MENTATION PAGE

Form Approved
OMB No 0704-0188

2

estimated to average 1 hour per response including the time for reviewing instructions searching existing data sources
g and reviewing the collection of information Send comments regarding this burden estimate or any other aspect of this
g this burden to Washington Headquarters Services Directorate for Information Operations and Reports, 1215 Jefferson
to the Office of Management and Budget Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE

3. REPORT TYPE AND DATES COVERED
FINAL 1 Sep 89 - 31 Aug 92

4. TITLE AND SUBTITLE
"COMPUTATION & LEARNING IN NEURAL NETWORKS WITH BINARY WEIGHTS" (U)

5. FUNDING NUMBERS
61102F
2305/B3

6. AUTHOR(S)
Professor Santosh S. Venkatesh

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Pennsylvania
Dept of Electrical Engineering
Philadelphia PA 19104

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 93 0068

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
110 Duncan Ave Suite ~~100~~ B115
Bolling AFB DC 20332-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-89-0523

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE
UL

13. ABSTRACT (*Maximum 200 words*)

Under the aegis of the AFOSR grant they have been investigating computational learning attributes of networks of formal neurons. The formal neurons considered are linear threshold elements which produce binary outputs based on the sign of a linear form of a set of inputs. The researchers have been interested in (1) exploring the theoretical limitations on what can be computed or learnt in neural network architectures, and (2) developing and analysing learning algorithms which specify weights as a function of a set of examples of a computation.

14. SUBJECT TERMS

15. NUMBER OF PAGES
164

16. PRICE CODE

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | SAR |

# COMPUTATION AND LEARNING IN NEURAL NETWORKS WITH BINARY WEIGHTS

PI: Santosh S. Venkatesh
Department of Electrical Engineering

AWARD NUMBER: AFOSR 89-0523

FINAL REPORT

November 28, 1992

UNIVERSITY of PENNSYLVANIA
The Moore School of Electrical Engineering
PHILADELPHIA, PENNSYLVANIA 19104-6390

# COMPUTATION AND LEARNING IN NEURAL NETWORKS WITH BINARY WEIGHTS

PI: Santosh S. Venkatesh
Department of Electrical Engineering

AWARD NUMBER: AFOSR 89-0523

FINAL REPORT

November 28, 1992

# PROPOSAL TITLE: COMPUTATION AND LEARNING IN NEURAL NETWORKS WITH BINARY WEIGHTS

PI: Santosh S. Venkatesh
Department of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104
*E-Mail*: venkatesh@ee.upenn.edu

AWARD NUMBER: AFOSR 89-0523

**FINAL REPORT**
November 28, 1992

# 1 PROBLEM DESCRIPTION AND RESULTS

## 1.1 Brief Overview

Under the aegis of the AFOSR grant we have been investigating computational and learning attributes of networks of formal neurons. The formal neurons we consider are linear threshold elements which produce binary outputs based on the sign of a linear form of a set of inputs. In particular, each neuron is characterised by a vector of weights $\mathbf{w}$, and given a set of inputs $\mathbf{u}$, produces an output $v = \text{sgn}\langle\mathbf{w}, \mathbf{u}\rangle = \text{sgn}\sum_i w_i u_i$.[1] In a given neural network architecture the degrees of freedom reside in the specification of the neural weights; in particular, each choice of weights specifies a particular computation. We have been interested in (1) exploring the theoretical limitations on what can be computed or learnt in neural network architectures, and (2) developing and analysing learning algorithms which specify weights as a function of a set of examples of a computation.

Since information about any computation realisable in a neural network resides in the selection of the weights, a cogent question relevant to the understanding of the efficiency of this computational structure is: ***How much information can be stored per bit of weight***? A satisfactory resolution of this question will have a direct import on the dynamic range for the weights that will be demanded of purveyors of neural network hardware. Classical learning algorithms such as Perceptron Training and Backpropagation are typically operational in situations where there is no dynamic range limitation on the weights; if, as we demonstrate, dynamic range requirements are not extreme, then another question arises: ***Do there exist efficient algorithms for learning weights in a dynamic range limited network structure***? In the analysis of these issues we present results for two distinct scenarios: in one we consider networks with weights restricted to being binary—in a network of $N$ neurons this corresponds to spreading an available dynamic range of $N$ bits uniformly across the neurons—; in the other we analyse a class of sparsely interconnected neural networks, a situation which corresponds to packing available dynamic range in a few weights.

On another front, we have also investigated the enhancement in computational capability that results if the degrees of freedom in specifying the weights are increased. In particular, we have

[1] This is the basic neural model proposed by McCulloch and Pitts (1943). A real threshold can be incorporated in the model, but is not essential to our discussion.

analysed a family of recurrent neural networks with the linear threshold neural model replaced by a polynomial threshold model where each higher order neuron produces a binary output according to the sign of a polynomial form of the inputs. The intuition here is that the extra degrees of freedom in specifying the polynomial coefficients (weights) should result in more powerful computational structures.

We have also been developing a theoretical basis in which the analysis of the above and similar problems can be carried out. The framework that is evolving rests upon a statistical notion of the computational capacity of a network architecture and an associated algorithm. The notion of capacity is turning out to be a fundament in the analysis of intrinsic computational and learning attributes of computational structures, and has been brought to prominence in the PAC learning model of Valiant (1984).

In the following we briefly summarise the results we have obtained, providing a road map as it were to the attached papers where we carry out more searching investigations of the subject matter. We also present prognoses and brief summaries of work in progress.

## 1.2 Binary Weights

Consider a neuron with weights restricted to be binary, $w_i \in \{-1, 1\}$, $i = 1, \ldots, n$. If we consider inputs to be binary as well, each assignment of weights to the neuron results in the realisation of one of $2^n$ distinct Boolean functions of $n$ Boolean variables, and in particular, one of $2^n$ majority functions of a set of $n$ literals: for every $\mathbf{u} \in \{-1, 1\}^n$, $f(u_1, \ldots, u_n) = \mathrm{sgn}\sum_{i=1}^n w_i u_i$. An immediate question is what is the computational capacity of such an element (vis à vis a neuron where the weights are allowed to be real). Consider a randomly chosen $m$-set of points $\mathbf{u}^1, \ldots, \mathbf{u}^m$ drawn from $\{-1, 1\}^n$ with components drawn from a sequence of symmetric Bernoulli trials and an associated set of desired binary ($\{-1, 1\}$) classifications $v^1, \ldots, v^m$. We are interested in whether there exists (with high probability) a vector of binary weights $\mathbf{w} \in \{-1, 1\}^n$ such that each of the points is classified correctly:

$$\mathrm{sgn}\sum_{i=1}^{n} w_i u_i^\alpha = v^\alpha, \qquad \alpha = 1, \ldots, m. \tag{1}$$

It is clear that for each $i = 1, \ldots, n$, the *binary* weight $w_i \in \mathbf{B}$ has to retain information about the corresponding $m$ input components $u_i^1, \ldots, u_i^m$, and the desired classifications $v^1, \ldots, v^m$. The difficulty, of course, is that we have to store information about $m$ bits in a single bit, and, especially if the learning procedure is on-line, it may perhaps appear doubtful if this is possible at all.

Let us consider without loss of generality that the $m$ desired classifications are all $+1$. The assignments in (1) are hence more likely to be realised if, for each fixed $i$, $\min_\alpha \mathbf{E} w_i u_i^\alpha$ can be made as large as possible; i.e., the probability that a weight has the same sign as a corresponding pattern component is made as large as possible. Using randomisation in the algorithm as a tool we show that for local on-line procedures $\sup \min_\alpha \mathbf{E} w_i u_i^\alpha = \Theta(1/n)$ for each $i$. Here the sup is taken over all local, on-line procedures. For local off-line procedures the analogous result is $\sup \min_\alpha \mathbf{E} w_i u_i^\alpha = \Theta(1/\sqrt{n})$. A detailed development of these and related results are contained in [1], and is included as the first attachment. The basic conclusion that may be drawn from these investigations is that fairly large capacities are attainable for networks of neurons with binary weights, and that these capacities are comparable to those when the weights are unrestricted reals.

While large capacities may be attainable in principle, there are practical difficulties in the design of binary weight learning algorithms: learning binary weights is equivalent to integer programming and is NP-complete. We might hence anticipate that for any binary weight learning algorithm there exists at least one problem instance which is intractable (i.e., takes exponential time). We have investigated the use of random algorithms, however, as a technique to partially circumvent the NP-completeness of the problem by providing good average-case performance. In particular, we have developed a family of local, on-line randomised algorithms (dubbed Directed Drift) which provide good average-case performance in certain regimes. Details are provided in [2] which constitutes the next attachment. R. Meir has communicated to us that simulations indicate that for local, on-line algorithms Directed Drift appears to have an optimal character.

**Prospectus** We have been investigating further randomisation ideas in the development of on-line and off-line learning algorithms and these are to be reported in [3]. Combining these ideas with the Directed Drift family of algorithms we have developed heuristics for learning binary weights for arbitrary network configurations. Early simulation results indicate very promising performance of these randomised algorithms. The implications to hardware development can be profound as these early results indicate that it suffices to have very small dynamic ranges for weights (one bit suffices in many applications), and with on-line algorithms such as those described here this can lead to very low complexity neural network hardware with on-line learning capabilities.

We have begun to study batching in on-line algorithms as a tool in the study of the tradeoffs in performance and complexity when we move from on-line to off-line procedures. In particular, we have been examining a randomised batch version of Directed Drift, the on-line algorithm described in the previous report. (Batch learning is a process intermediate in complexity between on-line and off-line learning where learning still takes place in a sequence of trials, but a (small) batch of examples is available to the algorithm at each epoch.) Surprising and unlooked for results have emerged in this consideration [4]. While Directed Drift converges very rapidly when the number of examples is small, it slows down substantially when the number of examples becomes large, a regime where, effectively, the examples are numerous enough to uniquely identify the generating function. In this latter regime, batch versions of the algorithm, however, show improvements in convergence time of several orders of magnitude, even for very small batch sizes. Improvements saturate quickly with increasing batch size leading to the conjecture that a modified on-line learning algorithm with very small batch sizes can achieve off-line performance. These and other results are to be reported at the Conference on Neural Networks for Computing, Snowbird, 1992.

## 1.3 Sparse Networks

Sparsity in networks can arise either as a result of architectural constraints or can arise as a consequence of damage to the network. In either case sparsity can be viewed as a situation where a total available dynamic range in bits for the weights is distributed among a few weights in the network.

When sparsity occurs as a result of damage to the network, the principal concern is whether the network continues to function effectively, i.e., whether the network is structurally *robust*. In contexts with large interconnectivity, neural folklore tells us that networks will continue to function efficiently, albeit with some degradation, in the presence of component damage or loss. We have rigourously examined this premise for a fully-interconnected network of neurons in an associative memory application by introducing the "devil"[2] in the network as an agent that produces sparsity by snipping connections between neurons. A consideration of a malicious (or at best neutral) devil which removes connections at random yields the following strong validation of the robustness hypothesis: *in a network of $n$ neurons each neuron needs retain only of the order of $\log n$ random links (on average) of a total of $n$ possible links with other neurons for useful computational properties to emerge.* Memory storage capacity degrades very gracefully as the probability of losing links increases. Details are presented in [5, 6] which constitute the next two attachments.

When network sparsity arises as a consequence of architectural constraints it may be possible to demarcate classes of problems which are well suited to the sparse structure. We have investigated this in a recurrent neural network situation where the neurons are partitioned into fully-interconnected sub-blocks with few or no connections between blocks (*nested sparsity* and *block sparsity*, repectively). For an associative memory application we identify memories to be stored as *codewords* and the collection of admissible memories as *codes*—the neural network is a *decoder* which corrects errors in memories. We show that for networks of $n$ neurons in nested or block architectures, storage

[2] Well, maybe an imp.

capacities as large as $2^{cn}$ memories for any $c < 1$ can be achieved for a family of codes (admissible sets of memories) which is exponential in size. More precise statements of the results and details of constructions and proofs can be found in the attachments [5, 7].

**Prospectus** Sparse network structures are again practically motivated as hardware would appear to favour certain regular, sparse interconnectivity patterns. The characterisation of problems best fitted to these structures is still an open question which the above investigations answer only partially. In an effort to understand how computation and capacity scale with increasing sparsity we have done extensive simulations in a feedforward network environment. The results are reported in attachment [20]. In general, capacity decreases with increasing sparsity roughly in proportion to the loss in the degrees of freedom. This is reflected by a concurrent improvement in learning times.

## 1.4 Polynomial Neural Interactions

Higher order neural networks have been proposed in the literature as a means of enhancing the computational capability of these networks. A higher order neuron is a polynomial threshold element which computes the sign of a polynomial form of its inputs. In particular, a higher order neuron of degree $d$ and $n$ inputs is characterised by a set of $\binom{n}{d}$ weights $w_{i_1,\ldots,i_d}$, $1 \leq i_1 < \cdots < i_d \leq n$. In response to an input $\mathbf{u} = (u_1 \cdots u_n)$ it produces an output

$$v = \operatorname{sgn} \sum_{1 \leq i_1 < \cdots < i_d \leq n} w_{i_1,\ldots i_d} u_{i_1} \cdots u_{i_d}.$$

The increased degrees of freedom give rise to a commensurate improvement in computational capability. We have obtained rigourous results on the computational gains that accrue in recurrent networks of higher order neurons. The main results can be summarised as follows: the information storage capacity of a recurrent higher order neural network is of the order of one bit per polynomial interaction coefficent (weight), this result being independent of the choice of the algorithm. We provide exact results on associative storage capability and error correction for a variety of algorithms in attachments [9, 10].

We have also carried out a complementary analysis of the structure of fixed points in symmetric recurrent higher order networks when the weights $w_{i_1,\ldots,i_d}$ are standard normal $\mathcal{N}(0,1)$ random variables. This corresponds to higher order spin glasses in statistical physics. We obtain expressions for the expected number of fixed points as a function of their margin of stability. In particular, we show that there exists a critical margin of stability below which the expected number of fixed points increases exponentially in $n$, and above which the expected number of fixed points actually decreases exponentially with $n$. A formal statement of these results and proofs is provided in attachment [11].

On another tack, we have developed algorithms for recurrent neural networks for an associative memory application. In particular, we have shown that it is possible to store memories with simulaneous memory-specific as well as feature- (or direction-) specific error correction while retaining high storage capacity [13, 14].

**Prospectus** The computational gains that accrue from higher order neural networks agree with intuition—they correspond with the increase in the degrees of freedom in the network. An attractive feature of these networks is that each higher order neuron can be replaced by a functionally equivalent small network of linear threshold elements so that hardware can be standardised with the formal neuron (linear threshold element) as the basic building block. The low complexity algorithms described in [9, 10] indicate that the higher capacity latent in higher order networks can actually be realised. There are open issues on the nature of problems that efficiently fit networks of polynomial threshold elements; for instance, it is not known whether the family of poly-sized two layer higher order networks is functionally strictly subsumed within the class of poly-sized three layer higher order networks. We are investigating these issues.

## 1.5 Capacity and Learning Sample Complexity

A common thread running through our analysis of the nature of information storage in the weights of a neural network has been the notion of the statistical capacity of a network architecture and an algorithm. This is a distribution dependent notion which captures, loosely speaking, the largest size of a randomly specified set of input points which can be mapped into a corresponding independently specified set of output points with high probability by the network specified by the algorithm. This statistical notion of capacity plays a critical role in determining the minimum size of labeled sample (the sample complexity) needed to identify a given function realisable in a given network architecture. The statistical notion of capacity that we espouse is related to a combinatorial parameter known as the VC-dimension which is a critical parameter in a distribution-free learning model. We develop a fairly general set of definitions of capacity in the attachment [12] and have presented the material in [15]. Definitions of capacity can also be found in the earlier reported work, and in particular, in [5, 6, 7, 9, 10].

One particular problem we have considered is the gains that may be realisable in computational capacity if errors are permitted in the output. Our main results here are that allowing a linear number of output errors improves the constants, but not the rate of growth of capacity. The exact constants depend upon what protocol governs the errors. Exact expressions and derivations are provided in attachment [16, 17].

As aforementioned, capacities govern the sample complexities needed for learning. A related issue of both practical and theoretical interest is the rate of convergence that can be expected of a learning algorithm as a function of the sample size. In an effort to shed light on what may be the worst case behaviour we considered the classical nearest neighbour algorithm which has been suggested to be representative of the best non-parametric learning algorithms. (The specification of a host neural network architecture for a learning algorithm in sharp contradistinction imposes a parametrisation. The learning algorithm seeks to find weights—the parameters—for the architecture which best fit the data. If the parametrisation, i.e., the choice of architecture, is appropriate this should result in substantially better behaviour than a non-parametrised approach.) It is known that in the infinite sample limit the nearest neighbour algorithm has performance no worse than twice the Bayes risk, and an old result of T. M. Cover shows that for one-dimensional feature spaces the convergence rate to the infinite sample limit is as rapid as $\Theta(m^{-2})$ where $m$ is the sample size. In attachment [18, 19] we present a precise statement of a generalisation of Cover's result to $n$-dimensional feature spaces which has been hitherto lacking. We show that the performance of the nearest neighbour algorithm converges to its infinite sample limit as rapidly as $\Theta(m^{-2/n})$, where $n$ is the dimensionality of the space and $m$ is the sample size. This result holds under mild conditions on the input distribution. (Alternatively, the sample complexities needed for learning are exponential in $n$.) Clearly Bellman's "curse of dimensionality" is made evident in the drastic reduction in convergence rates as the input dimensionality increases.

On another tack, we have been extending these results to precisely estimate the value of side-information in learning, with special reference to a problem proposed by T. Cover: *How many unlabelled examples is each labelled example worth in learning?* The question has import when unlabelled examples exist in relative profusion, but there are few labelled examples or where labelling examples is expensive. The answer in general depends on how much side-information is present. We have early results and are investigating further [20].

**Prospectus** We are seeking to combine the various elements described above into a theory of evolutionary learning in a neural network setting. A result of J. S. Judd indicates that the problem of deciding whether a given problem instance can be loaded into a given architecture may be intractable (read NP-complete). It may be possible to circumvent this problem by adopting a suitable evolutionary protocol where the network architecture is allowed to grow in time. To keep the complexity manageable we have been considering binary weight networks, using the randomised algorithms we have been developing for training at every stage of network evolution. Training periods at each stage of the evolution are governed by the statistical capacity of the network at that stage in the evolution. Results are only preliminary at this stage. The convergence rate calculations for the nearest neighbour algorithm indicate the importance of choosing a proper evolutionary protocol. A very loose evolutionary structure would be essentially unparametrised leading to very large times for convergence.

## 1.6 Coin Tossing and Randomised Algorithms

The last attachment [21] is not directly related to issues in neural network compuatation, but indirectly in that it examines limitations of randomised procedures. The basic question analysed is the expected minimum duration of a coin tossing game which carries the essence of several randomised search procedures in cryptography. The paper provides precise and rather complete estimates of the minimum duration of the game and provides constructions for generating the optimal strategy.

**Prospectus** We are seeking efficient learning algorithms to learn discrete weights for neural networks. In this randomisation is turning out to be a very effective tool to attack some formally intractable learning problems. We are investigating among other issues, a characterisation of the effectiveness of randomisation.

# References

[1] S. S. Venkatesh and J. Franklin, "How Much Information Can One Bit of Memory Retain About a Bernoulli Sequence?" *IEEE Transactions on Information Theory*, vol. IT-37, pp. 1595–1604, 1991. Presented at *IEEE International Symposium on Information Theory*, San Diego, California, 1990, and *Workshop on Optical Neural Networks*, Jackson, Wyoming, 1990.

[2] S. S. Venkatesh, "Directed Drift: A New Linear Threshold Algorithm for Learning Binary Weights On-Line," *Journal of Computer and Systems Sciences*, vol. 46, 1993, in press.

[3] S. S. Venkatesh, "On learning binary weights for majority functions," in *Proceedings of the Fourth Workshop on Computational Learning Theory*, (eds. L. G. Valiant and M. K. Warmuth). San Mateo, California: Morgan Kaufmann, 1991. Also presented at *Workshop on Neural Networks for Compnting*, Snowbird, Utah, 1991.

[4] S. Fang and S. S. Venkatesh, "Probabilistic algorithms for on-line learning in a binary weight setting," presented at *Conference on Neural Networks for Computing*, Snowbird, 1991.

[5] S. Biswas and S. S. Venkatesh, "The devil and the network: what sparsity implies to robustness and memory," in *Advances in Neural Information Processing Systems 3*, (eds. D. Touretzky and R. Lipmann). San Mateo, California: Morgan Kaufmann, 1991.

[6] S. S. Venkatesh, "Robustness in Neural Computation: Random Graphs and Sparsity," *IEEE Transactions on Information Theory*, vol. 38, no. 3, pp. 1114–1118, May 1992. Presented at *IEEE Workshop on Information Theory*, Veldhoven, the Netherlands, 1990.

[7] S. Biswas and S. S. Venkatesh, "Codes, sparsity, and capacity in neural associative memory," *IEEE Transactions on Information Theory*, submitted for publication. Presented at *IEEE Workshop on Information Theory*, Veldhoven, the Netherlands, 1990.

[8] J. Ratsaby and S. S. Venkatesh, "Empirical investigations into the effects of sparsity on network capacity," *Technical Report*, 1990.

[9] S. S. Venkatesh and P. Baldi, "Programmed Interactions in Higher-Order Neural Networks: Maximal Capacity," *Journal of Complexity*, vol. 7, no. 3, pp. 316–337, 1991.

[10] S. S. Venkatesh and P. Baldi, "Programmed Interactions in Higher-Order Neural Networks: The Outer-Product Algorithm," *Journal of Complexity*, vol. 7, no. 4, pp. 443–479, 1991.

[11] S. S. Venkatesh and P. Baldi, "Random Interconnections in Higher-Order Neural Networks," *IEEE Transactions on !nformation Theory*, vol. 39, no. 1, January 1993, in press.

[12] S. S. Venkatesh, "Computation and learning in the context of neural network capacity," in *Neural Networks for Perception*, (ed. H. Wechsler). New York: Academic Press, 1991. [Invited contribution.]

[13] S. S. Venkatesh, G. Pancha, D. Psaltis, and G. Sirat, "Shaping Attraction Basins in Neural Networks," *Neural Networks*, vol. 3, no. 6, pp. 613–624, 1990.

[14] G. Pancha and S. S. Venkatesh, "Feature and Memory Selective Error Correction in Neural Associative Memory," in *Associative Neural Memories: Theory and Implementation* (ed. M. H. Hassoun). New York: Oxford University Press, 1992. [Invited contribution.]

[15] S. S. Venkatesh, "Probabilistic capacity and links to distribution dependent learning," *DIMACS Workshop on Theoretical Issues in Neural Nets*, Rutgers University, New Brunswick, New Jersey, 1991.

[16] S. S. Venkatesh and D. Psaltis, "On Reliable Computation with Formal Neurons", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 87–91, 1992. Presented at *Workshop on Neural Networks for Computing*, Snowbird, Utah, 1990.

[17] S. S. Venkatesh, "The Science of Making *Erors*," *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, no. 2, pp. 135–144, April 1992. [Invited paper.]

[18] R. Snapp, D. Psaltis, and S. S. Venkatesh, "Asymptotic Slowing Down of the Nearest Neighbour Classifier," in *Advances in Neural Information Processing Systems 3*, (eds. D. S. Touretzky and R. Lippman). San Mateo, California: Morgan Kaufmann, 1991.

[19] S. S. Venkatesh, R. Snapp, and D. Psaltis, "Bellman Strikes Again—The Rate of Growth of Sample Complexity with Dimension for the Nearest Neighbour Classifier," in *Proceedings of the Fifth Workshop on Computational Learning Theory.* San Mateo, California: Morgan Kaufmann, 1992.

[20] J. Ratsaby and S. S. Venkatesh, "Learning classification with few labelled examples," in *Proceedings of the Conference on Neural Information Processing Systems*, Denver 1992. Also presented at the *Conference on Neural Networks for Computation*, Snowbird, 1991.

[21] I. Hu and S. S. Venkatesh, "On the Minimum Expected Duration of a Coin Tossing Game," *IEEE Transactions on Information Theory*, March 1993, in press. To be presented at the *IEEE International Symposium on Information Theory*, San Antonio, January 1993.

# How Much Information Can One Bit of Memory Retain About a Bernoulli Sequence?

Santosh S. Venkatesh, *Member, IEEE*, and Joel Franklin

***Abstract*—The maximin problem of the maximization of the minimum amount of information that a single bit of memory retains about the entire past is investigated. Specifically, a random binary sequence of ± 1 inputs drawn from a sequence of symmetric Bernoulli trials is given. A family of (time dependent, deterministic or probabilistic) memory update rules that at each epoch produce a new bi. ( − 1 *or* 1) *of memory* depending solely on the epoch, the current input, and the current state of memory is also given. The problem is to estimate the supremum over all possible sequences of update rules of the minimum information that the bit of memory at epoch $(n+1)$ retains about the previous $n$ inputs. Using only elementary techniques we show that the *maximin covariance* between the memory at epoch $(n+1)$ and past inputs is $\Theta(1/n)$, the *maximum average covariance* is $\Theta(1/n)$, and the *maximin mutual information* is $\Omega(1/n^2)$. In a consideration of related issues, we also provide an exact count of the number of Boolean functions of $n$ variables that can be obtained recursively from Boolean functions of two variables, discuss extensions and applications of the original problem, and indicate links with issues in neural computation.**

***Index Terms*—Bernoulli sequence, Boolean functions, memory, covariance, mutual information, neuron, capacity.**

## I. A Problem in Information Storage

János Komlós posed the following problem: Given a single bit of memory and a random binary sequence of inputs, at any epoch in time what is the maximum amount of information that the memory can retain about the *entire* binary sequence?

More precisely, let $\{X_n\}_{n=1}^{\infty}$ be a sequence of symmetric Bernoulli trials, with

$$X_n = \begin{cases} -1 & \text{with probability } 1/2 \\ 1 & \text{with probability } 1/2. \end{cases}$$

Let $M_n \in \{-1, 1\}$ denote the state of a one bit memory at epoch $n$. The memory states are updated by a sequence of (possibly random) Boolean functions, $f_n$, of two Boolean variables: $M_{n+1} = f_n(M_n, X_n)$. (The initial memory state, $M_1$, is arbitrary.) For each $n$ we are required to estimate

$$I_n = \max_{f_1,\cdots,f_n} \min_{1 \le k \le n} E(M_{n+1}X_k). \tag{1}$$

Is $I_n$ bounded away from zero? Can we identify functions $f_1^*,\cdots,f_n^*$ that achieve $I_n$?

Komlós' problem can be generalized in various ways with other measures of information used instead of the covariance. Specificaliy, we can consider the determination of

$$J_n = \max_{f_1,\cdots,f_n} \min_{1 \le k \le n} I(M_{n+1}; X_k), \tag{2}$$

where $I(M_{n+1}; X_k)$ denotes the mutual information of $M_{n+1}$ and $X_k$. Another measure of (average) information about the past that we investigate is

$$K_n = \max_{f_1,\cdots,f_n} \frac{1}{n} \sum_{k=1}^{n} E(M_{n+1}X_k). \tag{3}$$

The following are the main results[1]:

$$I_n = \Theta\left(\frac{1}{n}\right),$$

$$J_n = \Omega\left(\frac{1}{n^2}\right),$$

$$K_n = \Theta\left(\frac{1}{n}\right).$$

The last result is due to Komlós, Rejtö, and Tusnády [1] who have recently investigated the average covariance, $K_n$, in a control problem. In this paper, we show that the result holds as a direct consequence of arguments adduced in the consideration of the maximin problem $I_n$. We also show that the maximum average covariance is $\Theta(1/\sqrt{n})$ when we allow update rules with unlimited access to past inputs. Specifically, let $\mathscr{F}$ denote the family

Manuscript recieved June 8, 1990; revised March 19, 1991. This work was supported by the Air Force Office of Scientific Research under Grant AFOSR-89-0523. This work was presented in part at the IEEE International Symposium on Information Theory, San Diego, CA, January 14–19, 1990.

S. S. Venkatesh is with the Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104.

J. Franklin is with the Department of Applied Mathematics, Room 217-50, California Institute of Technology, Pasadena, CA 91125.

IEEE Log Number 9102255.

[1] *On Notation.* If $\{x_n\}$ and $\{y_n\}$ are positive sequences, we denote: $x_n = O(y_n)$ if there is a positive constant $K$ such that $x_n/y_n < K$ for all $n$; $x_n = \Omega(y_n)$ if there is a positive constant $L$ such that $x_n/y_n > L$ for all $n$; $x_n = \Theta(y_n)$ if $x_n = O(y_n)$ and $x_n = \Omega(y_n)$; and $x_n \sim y_n$ if $x_n/y_n \to 1$ as $n \to \infty$.

of all update rules mapping $\{-1,1\}^n$ into $\{-1,1\}$. Then

$$\max_{f\in\mathcal{F}n}\frac{1}{n}\sum_{k=1}^{n}E(X_k f(X_1,\cdots,X_n))\sim\frac{\sqrt{2}}{\sqrt{\pi n}}\qquad(n\to\infty).$$

In the proof of the results, it also develops that the maximin and average *absolute value* of covariances is also $\Theta(1/n)$, with

$$\frac{1}{n}\le\max_{f_1,\cdots,f_n}\min_{1\le k\le n}|E(M_{n+1}X_k)|<\frac{2}{n},$$

and

$$\frac{1}{n}\le\max_{f_1,\cdots,f_n}\frac{1}{n}\sum_{k=1}^{n}|E(M_{n+1}X_k)|<\frac{2}{n}.$$

If we restrict attention to a reasonable family of update rules—*monotone symmetric rules*—we demonstrate, in fact, that $\max\min E(M_{n+1}X_k)=1/n$ and $\max\min I(M_{n+1};X_k)\sim 1/2n^2\ln 2$.[2]

In Section III, we will conclude by looking briefly at some related issues. In particular, we will: provide an exact count of the number of Boolean functions of $n$ variables that can be obtained by a recursive application of $(n-1)$ Boolean functions of two variables, with the variables taken in sequence—there are exactly $(0.4)6^n+1.6$ such Boolean functions—examine extensions of the results and raise some open questions when more than one bit of memory is available; and link these results with issues in information storage in neural networks.

## II. Information Bounds

### A. Probabilistic Rules

In the most general setting the update rules, $M_{k+1}=f_k(M_k,X_k)$, are probabilistic and can be characterized in terms of probabilities conditioned upon the epoch, $k$, the current state of memory, $M_k$, and the current input, $X_k$, as follows: if $M_k=i\in\{-1,1\}$ and $X_k=j\in\{-1,1\}$, then set

$$M_{k+1}=\begin{cases}-M_k & \text{with probability } p_k(i,j),\\ M_k & \text{with probability } \bar{p}_k(i,j)=1-p_k(i,j).\end{cases}$$

Alternatively,

$$p_k(i,j)=P\{M_{k+1}=-i\mid M_k=i,X_k=j\},$$
$$\bar{p}_k(i,j)=P\{M_{k+1}=i\mid M_k=i,X_k=j\}.$$

Each update rule, $f_k$, can hence be defined by four (independently specifiable) probabilities, $p_k(-1,-1)$, $p_k(-1,1)$, $p_k(1,-1)$, and $p_k(1,1)$, each of which represents the probability, given the epoch, and current values of memory and input, that the memory update results in a *change of sign of memory*.

We define the family of *monotone symmetric* update rules to be update rules satisfying: $p_j(-1,-1)=p_j(1,1)=0$, and $p_j(-1,1)=p_j(1,-1)$, $j\ge 1$. The first of the two symmetry requirements, in particular, enforces no change in memory state if the current input agrees with the current state of memory—an intuitively appealing procedure.

We first evaluate the unconditional probabilities

$$\omega_k\triangleq P\{M_k=1\},$$
$$\bar{\omega}_k\triangleq P\{M_k=-1\}.$$

(Clearly, $\bar{\omega}_k=1-\omega_k$; we introduce the additional notation for later convenience.) Let us assume, without loss of generality, that we generate the initial value of the memory, $M_1$, by flipping a fair coin.[3] Hence, $\omega_1=\bar{\omega}_1=1/2$. For $j\ge 1$, define

$$\psi_j\triangleq p_j(-1,-1)+p_j(-1,1)+p_j(1,-1)+p_j(1,1).\quad(4)$$

For convenience, let us also define

$$p_0(-1,-1)=p_0(-1,1)=p_0(1,-1)=p_0(1,1)=1/2.$$

*Assertion 1:* For $k=0,1,\cdots,$ the unconditional probabilities for the state of the memory at epoch $k+1$ are given by

$$\omega_{k+1}=\sum_{i=0}^{k}\frac{1}{2}[p_i(-1,-1)+p_i(-1,1)]\prod_{j=i+1}^{k}\left(1-\frac{\psi_j}{2}\right),\quad(5)$$

$$\bar{\omega}_{k+1}=\sum_{i=0}^{k}\frac{1}{2}[p_i(1,1)+p_i(1,-1)]\prod_{j=i+1}^{k}\left(1-\frac{\psi_j}{2}\right).\quad(6)$$

*Proof:* We can obtain the following recursion by noting that $\bar{\omega}_k=1-\omega_k$.

$$\omega_{k+1}=\omega_k+\frac{\bar{\omega}_k}{2}[p_k(-1,-1)+p_k(-1,1)]-\frac{\omega_k}{2}[p_k(1,1)+p_k(1,-1)],$$

$$\bar{\omega}_{k+1}=\bar{\omega}_k-\frac{\bar{\omega}_k}{2}[p_k(-1,-1)+p_k(-1,1)]+\frac{\omega_k}{2}[p_k(1,1)+p_k(1,-1)].$$

The result can now be established by induction. □

For $k\ge 1$, let us now define

$$\phi_k\triangleq[\bar{\omega}_k p_k(-1,1)+\omega_k p_k(1,-1)]-[\bar{\omega}_k p_k(-1,-1)+\omega_k p_k(1,1)].\quad(7)$$

*Assertion 2:* For any choices of $n$ and $k$ with $k\le n$,

$$P\{M_{n+1}=X_k\}=\frac{1}{2}\left[1+\phi_k\prod_{j=k+1}^{n}\left(1-\frac{\psi_j}{2}\right)\right],\quad(8)$$

$$P\{M_{n+1}=-X_k\}=\frac{1}{2}\left[1-\phi_k\prod_{j=k+1}^{n}\left(1-\frac{\psi_j}{2}\right)\right].\quad(9)$$

*Remark:* We adopt the convention $\prod_{j=r}^{s}(\cdot)=1$ if $r>s$.

[2] We conjecture, in fact, that $\max\min E(M_{n+1}X_k)=1/n$ and $\max\min I(M_{n+1};X_k)\sim 1/2n^2\ln 2$, with the maximum being taken over all functions $f_1,\cdots,f_n$. This is not true for absolute values of covariances. However, J. Komlós has recently communicated a construction to us that demonstrates $\max\min|E(M_{n+1}X_k)|>1/n$.

[3] The initial choice of memory bit can have no information about the data sequence to come. The obvious optimal procedure would be to choose the update rule $M_2=f_1(M_1,X_1)=X_1$

*Proof:* To prove the assertion we use double induction on $k$ and $n$.

*Base:* For every choice of $n \geq 1$ and $k = n$, we have

$$\begin{aligned} P\{M_{n+1} = X_n\} &= \frac{\bar{\omega}_n}{2}[1 - p_n(-1,-1) + p_n(-1,1)] \\ &\quad + \frac{\omega_n}{2}[p_n(1,-1) + 1 - p_n(1,1)] \\ &= \frac{1}{2}[1 + \phi_n]. \end{aligned}$$

*Inductive Hypothesis:* Assume that for some choice of $n$ and $k < n$, we have

$$P\{M_n = X_k\} = \frac{1}{2}\left[1 + \phi_k \prod_{j=k+1}^{n-1}\left(1 - \frac{\psi_j}{2}\right)\right].$$

Now consider

$$\begin{aligned} &P\{M_{n+1} = X_k = 1\} \\ &\quad = P\{M_{n+1} = 1, M_n = 1, X_k = 1\} \\ &\qquad + P\{M_{n+1} = 1, M_n = -1, X_k = 1\} \\ &\quad = P\{M_{n+1} = 1 | M_n = 1, X_k = 1\} P\{M_n = 1, X_k = 1\} \\ &\qquad + P\{M_{n+1} = 1 | M_n = -1, X_k = 1\} P\{M_n = -1, X_k = 1\}. \end{aligned} \tag{10}$$

Now, given $M_n$, the random variable $M_{n+1}$ is conditionally independent of the random variable $X_k$. Hence,

$$\begin{aligned} P\{M_{n+1} = 1 | M_n = 1, X_k = 1\} &= P\{M_{n+1} = 1 | M_n = 1\} \\ &= \frac{1}{2}[1 - p_n(1,-1) + 1 - p_n(1,1)]. \end{aligned} \tag{11}$$

In similar fashion, we obtain

$$P\{M_{n+1} = 1 | M_n = -1, X_k = 1\} = \frac{1}{2}[p_n(-1,-1) + p_n(-1,1)]. \tag{12}$$

We now claim that

$$P\{M_n = -1, X_k = 1\} = \frac{1}{2} - P\{M_n = X_k = 1\}. \tag{13}$$

In fact, we have

$$\begin{aligned} P\{M_n = -1, X_k = 1\} &= P\{M_n = -1 | X_k = 1\} P\{X_k = 1\} \\ &= \frac{1}{2}(1 - P\{M_n = 1 | X_k = 1\}) \\ &= \frac{1}{2}\left(1 - \frac{P\{M_n = 1, X_k = 1\}}{P\{X_k = 1\}}\right), \end{aligned}$$

so that (13) follows. Substituting the results of (11)–(13) in (10), we obtain

$$\begin{aligned} P\{M_{n+1} = X_k = 1\} &= \frac{1}{4}[p_n(-1,-1) + p_n(-1,1)] \\ &\quad + \left(1 - \frac{\psi_n}{2}\right) P\{M_n = X_k = 1\}. \end{aligned}$$

An entirely analogous procedure yields

$$\begin{aligned} P\{M_{n+1} = X_k = -1\} &= \frac{1}{4}[p_n(1,1) + p_n(1,-1)] \\ &\quad + \left(1 - \frac{\psi_n}{2}\right) P\{M_n = X_k = -1\}. \end{aligned}$$

Combining the two results gives

$$\begin{aligned} &P\{M_{n+1} = X_k\} \\ &\quad = P\{M_{n+1} = X_k = 1\} + P\{M_{n+1} = X_k = -1\} \\ &\quad = \frac{\psi_n}{4} + \left(1 - \frac{\psi_n}{2}\right) P\{M_n = X_k\}. \end{aligned}$$

The base of the induction argument establishes the first part of the assertion, (8), for $k = n$, and the inductive hypothesis completes the induction for $k < n$. Equation (9) follows trivially from the observation that $P\{M_{n+1} = -X_k\} = 1 - P\{M_{n+1} = X_k\}$. □

*Assertion 3:* For $n \geq 1$ and $1 \leq k \leq n$,

$$\begin{aligned} &P\{M_{n+1} = X_k = 1\} \\ &\quad = \frac{\omega_k}{2} + \frac{1}{2}[\bar{\omega}_k p_k(-1,1) - \omega_k p_k(1,1)] \prod_{j=k+1}^{n}\left(1 - \frac{\psi_j}{2}\right) \\ &\qquad + \frac{1}{4}\sum_{i=k+1}^{n} [\{\bar{\omega}_k p_i(-1,1) - \omega_k p_i(1,-1)\} \\ &\qquad + \{\bar{\omega}_k p_i(-1,-1) - \omega_k p_i(1,1)\}] \prod_{j=i+1}^{n}\left(1 - \frac{\psi_j}{2}\right), \end{aligned}$$

$$\begin{aligned} &P\{M_{n+1} = X_k = -1\} \\ &\quad = \frac{\bar{\omega}_k}{2} + \frac{1}{2}[\omega_k p_k(1,-1) - \bar{\omega}_k p_k(-1,-1)] \prod_{j=k+1}^{n}\left(1 - \frac{\psi_j}{2}\right) \\ &\qquad - \frac{1}{4}\sum_{i=k+1}^{n} [\{\bar{\omega}_k p_i(-1,1) - \omega_k p_i(1,-1)\} \\ &\qquad + \{\bar{\omega}_k p_i(-1,-1) - \omega_k p_i(1,1)\}] \prod_{j=i+1}^{n}\left(1 - \frac{\psi_j}{2}\right), \end{aligned}$$

$$\begin{aligned} &P\{M_{n+1} = -1, X_k = 1\} \\ &\quad = \frac{\omega_k}{2} - \frac{1}{2}[\bar{\omega}_k p_k(-1,1) - \omega_k p_k(1,1)] \prod_{j=k+1}^{n}\left(1 - \frac{\psi_j}{2}\right) \\ &\qquad + \frac{1}{4}\sum_{i=k+1}^{n} [\{\bar{\omega}_k p_i(-1,1) - \omega_k p_i(1,-1)\} \\ &\qquad + \{\bar{\omega}_k p_i(-1,-1) - \omega_k p_i(1,1)\}] \prod_{j=i+1}^{n}\left(1 - \frac{\psi_j}{2}\right), \end{aligned}$$

$$\begin{aligned} &P\{M_{n+1} = 1, X_k = -1\} \\ &\quad = \frac{\bar{\omega}_k}{2} - \frac{1}{2}[\omega_k p_k(1,-1) - \bar{\omega}_k p_k(-1,-1)] \prod_{j=k+1}^{n}\left(1 - \frac{\psi_j}{2}\right) \\ &\qquad - \frac{1}{4}\sum_{i=k+1}^{n} [\{\bar{\omega}_k p_i(-1,1) - \omega_k p_i(1,-1)\} \\ &\qquad + \{\bar{\omega}_k p_i(-1,-1) - \omega_k p_i(1,1)\}] \prod_{j=i+1}^{n}\left(1 - \frac{\psi_j}{2}\right). \end{aligned}$$

*Proof:* These results can be verified, as in Assertion 2, by induction. □

*Remark:* The previous identities simplify considerably for the family of monotone symmetric update rules; in particular, update rules governed by probabilities of the form $p_j(-1,-1)=p_j(1,1)=0$, and $p_j(-1,1)=p_j(1,-1)=p_j$, $j \geq 1$. Substituting in (4)–(7) we have $\psi_k = 2p_k$, $\omega_k = \bar{\omega}_k = 1/2$, and $\phi_k = p_k$, for $k \geq 1$. Substituting these relations in the above expressions, we have

$$P\{M_{n+1}=X_k=1\} = P\{M_{n+1}=X_k=-1\}$$
$$= \frac{1}{4}\left[1 + p_k \prod_{j=k+1}^{n} (1-p_j)\right], \qquad (14)$$

and

$$P\{M_{n+1}=1, X_k=-1\} = P\{M_{n+1}=-1, X_k=1\}$$
$$= \frac{1}{4}\left[1 - p_k \prod_{j=k+1}^{n} (1-p_j)\right]. \qquad (15)$$

## B. Maximin Covariance

A direct application of (8) and (9) yields the following general result.

*Assertion 4:* For any choice of positive integers $n$ and $k$ with $1 \leq k \leq n$,

$$E(M_{n+1}X_k) = \phi_k \prod_{j=k+1}^{n} \left(1 - \frac{\psi_j}{2}\right), \qquad (16)$$

where $\psi_j$ and $\phi_k$ are given by (4) and (7), respectively.

Some examples may serve to fix the result.

*Example—Follow the Leader:* Consider the choice of rule $M_{j+1} = X_j$, $j \geq 1$, corresponding to the selection

$$p_j(-1,-1) = p_j(1,1) = 0,$$
$$p_j(-1,1) = p_j(1,-1) = 1.$$

From the defining equation (4), we clearly have $\psi_j = 2$ for every $j \geq 1$. Applying (5) and (6), we have the unconditional probabilities of the state of the memory given by $\omega_k = \bar{\omega}_k = 1/2$, so that applying (7) we have $\phi_k = 1$. Hence,

$$E(M_{n+1}X_k) = \begin{cases} 0, & \text{if } 1 \leq k \leq n-1, \\ 1, & \text{if } k = n, \end{cases}$$

in agreement with the intuitive result. Consequently, $\min_{k \leq n} E(M_{n+1}X_k) = 0$.

*Example—Parity:* Consider the sequence of update rules which, at any epoch $n$, set $M_{n+1} = 1$ iff an odd number of the random variables, $X_1, \cdots, X_n$ have taken on the value 1. The update rules determining $M_2$ and $M_{k+1}$, $k \geq 2$ are shown in Figs. 1 and 2. The probabilities corresponding to the update rules are, hence,

$$p_1(-1,-1) = p_1(1,1) = 0,$$
$$p_1(-1,1) = p_1(1,-1) = 1,$$

when $k = 1$, and

$$p_k(-1,-1) = p_k(1,-1) = 0,$$
$$p_k(-1,1) = p_k(1,1) = 1, \qquad k \geq 2.$$

| $M_1$ \ $X_1$ | $-1$ | $1$ |
|---|---|---|
| $-1$ | $-1$ | $1$ |
| $1$ | $-1$ | $1$ |

Fig. 1. Odd parity update for $M_2$.

| $M_k$ \ $X_k$ | $-1$ | $1$ |
|---|---|---|
| $-1$ | $-1$ | $1$ |
| $1$ | $1$ | $-1$ |

Fig. 2. Odd parity update for $M_{k+1}$.

Evaluating the various parameters we obtain

$$\psi_j = 2, \qquad j \geq 1,$$
$$\omega_k = 1/2, \qquad k \geq 1,$$
$$\bar{\omega}_k = 1/2, \qquad k \geq 1,$$
$$\phi_k = \begin{cases} 1, & \text{if } k = 1, \\ 0, & \text{if } k \geq 2. \end{cases}$$

*Substituting these into (16) yields*

$$E(M_{n+1}X_k) = 0, \qquad k = 1, \cdots, n.$$

For $n \geq 1$, this again yields $\min_{k \leq n} E(M_{n+1}X_k) = 0$.

These examples illustrate that it suffices, hence, to restrict attention to update rules that yield nonnegative covariances, $E(M_{n+1}X_k)$, for *every* $k \leq n$. The following example illustrates that a nonzero covariance can, in fact, be obtained between a memory and every past input using a purely deterministic sequence of update rules.

*Example—Unbroken Runs:* Consider the sequence of update rules which store a 1 in the memory iff there has been an unbroken run of inputs taking the value 1. The update rules determining $M_2$ and $M_{k+1}$, $k \geq 2$ are shown in Figs. 3 and 4. The probabilities corresponding to the update rules are, hence,

$$p_1(-1,-1) = p_1(1,1) = 0,$$
$$p_1(-1,1) = p_1(1,-1) = 1,$$

when $k = 1$, and

$$p_k(-1,-1) = p_k(-1,1) = p_k(1,1) = 0,$$
$$p_k(1,-1) = 1, \qquad k \geq 2.$$

| $X_1$ | $-1$ | $1$ |
|---|---|---|
| $M_1$ | | |
| $-1$ | $-1$ | $1$ |
| $1$ | $-1$ | $1$ |

Fig. 3. Unbroken run update for $M_2$.

| $X_k$ | $-1$ | $1$ |
|---|---|---|
| $M_k$ | | |
| $-1$ | $-1$ | $-1$ |
| $1$ | $-1$ | $1$ |

Fig. 4. Unbroken run update for $M_{k+1}$.

Evaluating the various parameters we obtain

$$\psi_j = \begin{cases} 2, & \text{if } j = 1, \\ 1, & \text{if } j \geq 2, \end{cases}$$

$$\omega_k = \begin{cases} 1/2 & \text{if } k = 1, \\ 2^{-k+1}, & \text{if } k \geq 2, \end{cases}$$

$$\bar{\omega}_k = \begin{cases} 1/2, & \text{if } k = 1, \\ 1 - 2^{-k+1}, & \text{if } k \geq 2, \end{cases}$$

$$\phi_k = \begin{cases} 1, & \text{if } k = 1 \\ 1 - 2^{-k+1}, & \text{if } k \geq 2. \end{cases}$$

Substituting these into (16) yields

$$E(M_{n+1}X_k) = \begin{cases} 2^{-n+1}, & \text{if } k = 1, \\ 2^{-n+1}(2^{k-1} - 1), & \text{if } k \geq 2. \end{cases}$$

Hence, $\min_{k \leq n} E(M_{n+1}X_k) = 2^{-n+1}$ for $n \geq 1$.

While the minimum covariance in the above example is nonzero, it is still exponentially small. To obtain somewhat larger minimum covariances we resort to probabilistic update rules.

*Example—Harmonic Updates:* For each $k \geq 1$ we prescribe the update rule $f_k$ by setting $p_k(-1,-1) = p_k(1,1) = 0$ and $p_k(-1,1) = p_k(1,-1) = 1/k$. This is equivalent to the following prescription:

1) if $X_k = M_k$, then set $M_{k+1} = M_k$;
2) if $X_k \neq M_k$, then set

$$M_{k+1} = \begin{cases} -M_k, & \text{with probability } 1/k \\ M_k, & \text{with probability } 1 - 1/k; \end{cases}$$

specifically, we do not change the current state of the memory if the current input matches the sign of the memory, and change the state of the memory probabilistically (but with increasing reluctance) in case of a mismatch in signs. Estimating the various parameters gives

$$\psi_j = \frac{2}{j}, \qquad j \geq 1,$$

$$\omega_k = \frac{1}{2}, \qquad k \geq 1,$$

$$\bar{\omega}_k = \frac{1}{2}, \qquad k \geq 1,$$

$$\phi_k = \frac{1}{k}, \qquad k \geq 1.$$

Substituting in (16) yields $E(M_{n+1}X_k) = 1/n$ for $k \leq n$. It, hence, follows that, in fact, $\min_{k \leq n} E(M_{n+1}X_k) = 1/n$.

*Theorem 1:* For every positive integer $n$,

$$\frac{1}{n} \leq \max_{f_1, \cdots, f_n} \min_{1 \leq k \leq n} E(M_{n+1}X_k) < \frac{2}{n}. \tag{17}$$

*Proof:* The lower bound of $1/n$ follows immediately from the construction of the harmonic update rule in the last example. For $k \geq 1$ let us define

$$\hat{\phi}_k = |\phi_k|, \tag{18}$$

$$\hat{\psi}_k = \begin{cases} \dfrac{\psi_k}{2}, & \text{if } 0 \leq \psi_k \leq 2, \\ 2 - \dfrac{\psi_k}{2}, & \text{if } 2 \leq \psi_k \leq 4. \end{cases} \tag{19}$$

Note that $0 \leq \psi_k \leq 4$, so that the definition above achieves a sort of "normalization": $0 \leq \hat{\psi}_k \leq 1$. An immediate consequence of the definition is the equality

$$\left|1 - \frac{\psi_k}{2}\right| = 1 - \hat{\psi}_k, \qquad k \geq 1.$$

We now claim that $\hat{\phi}_k \leq 2\hat{\psi}_k$ for every positive integer $k$. Indeed, we have from (7) and (19) that

$$\begin{aligned} \hat{\phi}_k &= |\bar{\omega}_k(p_k(-1,1) - p_k(-1,-1)) \\ &\quad + \omega_k(p_k(1,-1) - p_k(1,1))| \\ &\leq p_k(-1,1) + p_k(-1,-1) + p_k(1,-1) + p_k(1,1) \\ &= 2\hat{\psi}_k, \qquad \text{if } 0 \leq \psi_k \leq 2. \end{aligned}$$

Also, setting $\bar{p}_k(i,j) = 1 - p_k(i,j)$, for $i \in \{-1,1\}$ and $j \in \{-1,1\}$, we have

$$\begin{aligned} \hat{\phi}_k &= |\bar{\omega}_k(\bar{p}_k(-1,-1) - \bar{p}_k(-1,1)) \\ &\quad + \omega_k(\bar{p}_k(1,1) - \bar{p}_k(1,-1))| \\ &\leq \bar{p}_k(-1,-1) + \bar{p}_k(1,1) + \bar{p}_k(-1,1) + \bar{p}_k(1,-1) \\ &= 4 - \psi_k \\ &= 2\hat{\psi}_k, \qquad \text{if } 2 \leq \psi_k \leq 4. \end{aligned}$$

This proves the claim. □

Now consider (16). From the definitions (18) and (19), the "normalization" of $\psi_k$, and the claim,

$$E(M_{n+1}X_k) \le |\phi_k| \prod_{j=k+1}^{n} \left|1 - \frac{\psi}{2}\right|$$
$$= \hat{\phi}_k \prod_{j=k+1}^{n} \left(1 - \hat{\psi}_j\right) \le 2\hat{\psi}_k \prod_{j=k+1}^{n} \left(1 - \hat{\psi}_j\right). \quad (20)$$

To establish the validity of the upper bound in (17) we begin by showing that

$$\max_{\hat{\psi}_1,\cdots,\hat{\psi}_n} \min_{1 \le k \le n} \hat{\psi}_k \prod_{j=k+1}^{n} \left(1 - \hat{\psi}_j\right) \le \frac{1}{n}.$$

(Here the variables, $\hat{\psi}_j$, take values in the closed interval $[0,1]$, as previously noted.) For notational simplicity, denote

$$F_k = \hat{\psi}_k \prod_{j=k+1}^{n} \left(1 - \hat{\psi}_j\right) \qquad (1 \le k \le n). \qquad (21)$$

Consider first the choice $\hat{\psi}_j = 1/j$ for each $j$. Direct substitution yields that $F_k = 1/n$ for each $k = 1,\cdots,n$. Hence, $\min_{k \le n} F_k = 1/n$ for this choice of $\hat{\psi}_j$. We now claim that we can, without loss of generality, consider only choices $1 \ge \hat{\psi}_j \ge 1/j$ for each value of $j$. To see this, assume $\hat{\psi}_j < 1/j$ for some choices of $j \le n$. Let $k$ be the largest such $j$. We then have $\prod_{j=k+1}^{n}(1 - \hat{\psi}_j) \le k/n$ as $\hat{\psi}_j \ge 1/j$ for $j > k$, and $\hat{\psi}_k < 1/k$. Hence, $\min_{k \le n} F_k < 1/n$ if there is any $j \le n$ for which $\hat{\psi}_j < 1/j$.

We will now show that, in fact, $\max\min F_k = 1/n$, with the maximum achieved, as just seen, for the choice, $\hat{\psi}_j = 1/j$, for each $j$. By the result just shown, without loss of generality, for each $j$ we need consider only choices for $\hat{\psi}_j$ in the closed interval $[1/j, 1]$. Now consider

$$F_1 = \hat{\psi}_1 \prod_{j=2}^{n} \left(1 - \hat{\psi}_j\right).$$

For each $j$, we have $1/j \le \hat{\psi}_j \le 1$, and in particular, for $j = 1$ we have $\hat{\psi}_1 = 1^4$. Hence, we must necessarily obtain $F_1 < 1/n$, and consequently $\min_{k \le n} F_k < 1/n$, if there exists any $j$ with $\hat{\psi}_j > 1/j$.

We have, hence, shown that $\max\min F_k = 1/n$. From (20) and (21) we, then, have

$$\max_{f_1,\cdots,f_n} \min_{1 \le k \le n} E(M_{n+1}X_k)$$
$$\le 2 \max_{\hat{\psi}_1,\cdots,\hat{\psi}_n} \min_{1 \le k \le n} \hat{\psi}_k \prod_{j=k+1}^{n} \left(1 - \hat{\psi}_j\right) = \frac{2}{n}.$$

To complete the proof, we need to show that the upper bound $2/n$ is strict. To see this, note that $\max\min F_k = 1/n$ is achieved only for the unique choice of $\hat{\psi}_j = 1/j$ for each $j \le n$. An examination of the bounding technique used in deriving the bound of equation (20) shows that a necessary condition for the upper bound in (17) to be realizable is that $\hat{\phi}_j = 2\hat{\psi}_j = 2/j$ for each $j$. But for $j = 1$ this is already impossible as can be verified from (5)–(7), and (18). Hence, $\max\min E(M_{n+1}X_k) < 2/n$. □

*Remarks:* In this proof, we used the bound $\hat{\phi}_k \le 2\hat{\psi}_k$ valid for every $k$. This is, however, not the tightest possible as we saw above; in particular, the bound is not achievable when the best results (the bound of $2/n$) are obtained for the choice of parameters, $\hat{\psi}_k = 1/k$. A more careful analysis should see improvement in the upper bound. (In particular, the harmonic update rule is a persuasive candidate for being, in fact, the *optimal* update rule. If true, this would imply, of course, that $\max\min E(M_{n+1}X_k) = 1/n$.)

Note also that the proof yields the following stronger result: the same maximin bounds hold for the *absolute value* of the covariances, viz.,

$$\frac{1}{n} \le \max_{f_1,\cdots,f_n} \min_{1 \le k \le n} |E(M_{n+1}X_k)| < \frac{2}{n}.$$

### *C. Maximin Mutual Information*

Now consider the problem (2). Here the maximin problem is to maximize the mutual information between past inputs and the current memory state. In order to evaluate the mutual information, $I(M_{n+1}; X_k)$, for a general family of update rules, in general, we have recourse to Assertion 3. We obtain the lower bound below for $J_n$ by maximizing the minimum mutual information over a restricted set of update rules where the probabilities derived in Assertion 3 are somewhat more manageable.

*Theorem 2:*

$$\max_{f_1,\cdots,f_n} \min_{1 \le k \le n} I(M_{n+1}; X_k) = \Omega(n^{-2}) \qquad (n \to \infty).$$

*More specifically,*

$$\max_{f_1,\cdots,f_n} \min_{1 \le k \le n} I(M_{n+1}; X_k) \ge \frac{1}{2n^2 \ln 2} + O\left(\frac{1}{n^4}\right) \qquad (n \to \infty).$$

*Proof:* Let us restrict attention to the family of *monotone symmetric* update rules: $p_j(-1,-1) = p_j(1,1) = 0$, and $p_j(-1,1) = p_j(1,-1) = p_j$, $j \ge 1$. For simplicity let us denote

$$z_k = p_k \prod_{j=k+1}^{n} (1 - p_j).$$

From (14) and (15) we then have

$$P\{M_{n+1} = X_k = 1\} = P\{M_{n+1} = X_k = -1\} = \frac{1}{4}(1 + z_k),$$

and

$$P\{M_{n+1} = 1, X_k = -1\} = P\{M_{n+1} = -1, X_k = 1\}$$
$$= \frac{1}{4}(1 - z_k).$$

Noting that for the class of monotone symmetric update rules, the r.v.'s $M_{n+1}$ are symmetric, and take on the values $-1$ and $1$ with equal probability $1/2$, we have the

[4] In fact, the variable $\hat{\psi}_1$ appears only in the expression for $F_1$ where it appears as a product term. We can then maximize the value of $F_1$ without affecting any of the other $F_k$'s by setting $\hat{\psi}_1 = 1$.

following expression for the conditional uncertainty of $X_k$ given $M_{n+1}$:

$$\begin{aligned} H(X_k|M_{n+1}) &= -\frac{1}{2}(1+z_k)\log_2\frac{1}{2}(1+z_k) \\ &\quad -\frac{1}{2}(1-z_k)\log_2\frac{1}{2}(1-z_k) \\ &= h\left(\frac{1+z_k}{2}\right), \end{aligned}$$

where $h$ is the binary entropy function

$$h(y) = -y\log_2 y - (1-y)\log_2(1-y), \qquad 0 \le y \le 1.$$

(As usual, we define $0\log 0 = 0$.) Hence,

$$I(M_{n+1};X_k) = H(X_k) - H(X_k|M_{n+1}) = 1 - h\left(\frac{1+z_k}{2}\right).$$

By the same inductive argument used in establishing the upper bound for Theorem 1 we obtain that $\min_{k\le n} z_k$ is maximized among the class of monotone symmetric update rules for the unique choice of the harmonic update rule: $p_j = 1/j$ for each $j$. For this choice of update rule we have

$$z_k = \frac{1}{k}\prod_{j=k+1}^{n}\left(1-\frac{1}{j}\right) = \frac{1}{n}, \qquad k = 1,\cdots,n.$$

Using the monotone decreasing property of $h(y)$ for $1/2 \le y \le 1$ we have that $\min_{k\le n} I(M_{n+1};X_k\}$ is also maximized among the class of monotone symmetric update rules for the harmonic update rule. This estimate forms a useful lower bound for $J_n = \max\min I(M_{n+1};X_k)$. Hence,

$$\max_{f_1,\cdots,f_n}\ \min_{1\le k\le n} I(M_{n+1};X_k) \ge 1 - h\left(\frac{1}{2}+\frac{1}{2n}\right).$$

The Taylor series expansion for $\ln(1+y)$, $|y|<1$ yields the required asymptotic form in the statement of the theorem. □

*Remarks:* A general examination of $J_n$ over all possible update rules using the results of Assertion 3 appears somewhat difficult in view of the lack of symmetry in the various probabilities. A reasonable candidate hypothesis may be that it suffices to consider only monotone symmetric rules—$p_k(-1,-1) = p_k(1,1) = 0$ and $p_k(-1,1) = p_k(1,-1) = p_k$ for each $k \ge 1$. (If true this would, of course, yield the estimate $J_n \sim 1/2n^2 \ln 2$.) As noted earlier, this enforces symmetry and the intuitively appealing procedure of effecting no change in memory state if the current input agrees with the current state of memory. While it is relatively easy to show that we can, without loss of generality, set $p_n(-1,-1) = p_n(1,1) = 0$, the proof does not seem to extend simply to all $p_k(-1,-1)$ and $p_k(1,1)$.

### *D. Maximum Average Covariance*

J. Komlós has recently communicated to us results of joint work with L. Rejtö and G. Tusnády on the maximal expected payoff of a finite automaton with binary inputs [1]. Their results include the estimate $\Theta(1/n)$ for the maximal average covariance, $K_n$, which they obtain using conditioning on inputs coupled with an inductive argument. We show this estimate here as an (almost) direct consequence of the proof of Theorem 1.

*Theorem 3:* For every positive integer $n$,

$$\frac{1}{n} \le \max_{f_1,\cdots,f_n}\frac{1}{n}\sum_{k=1}^{n} E(M_{n+1}X_k) < \frac{2}{n}.$$

*Proof:* The lower bound follows from the lower bound for $I_n$. Now consider (21). Writing $F_k = F_{k,n}$ explicitly as a function of $n$, we have

$$F_{k,n} = \hat{\psi}_k\prod_{j=k+1}^{n}\left(1-\hat{\psi}_j\right) \qquad (1 \le k \le n,\ n = 1,2,\cdots).$$

Recall from equation (19) that $0 \le \hat{\psi}_j \le 1$ for every $j$, and that $\hat{\psi}_j$ depends solely on $j$ and not on $n$. Now form the sequence of sums, $\{S_n\}$, by setting

$$S_n = \sum_{k=1}^{n} F_{k,n} \qquad (n \ge 1).$$

Noting that

$$F_{k,n} = F_{k,n-1}\left(1-\hat{\psi}_n\right), \qquad \text{if } 1 \le k \le n-1,$$

we have

$$S_n = \left(1-\hat{\psi}_n\right)S_{n-1} + \hat{\psi}_n.$$

As $0 \le S_1 = \hat{\psi}_1 \le 1$, an easy inductive argument shows that $S_n$ is an iteration of convex combinations of numbers less than one, so that $S_n \le 1$. From (20) and the concluding remarks of the proof of Theorem 1, we have

$$E(M_{n+1}X_k) < 2F_{k,n},$$

so that

$$\max_{f_1,\cdots,f_n}\frac{1}{n}\sum_{k=1}^{n} E(M_{n+1}X_k) < 2\max_{f_1,\cdots,f_n}\frac{S_n}{n} \le \frac{2}{n}.$$

This completes the proof. □

*Remarks:* In fact, this convex combination argument can be used in lieu of the argument presented in the proof of Theorem 1. Note also that the bound of (20) is easily improved to $|E(M_{n+1}X_k)| < 2F_{k,n}$. The proof of Theorem 3 then yields the stronger result

$$\frac{1}{n} \le \max_{f_1,\cdots,f_n}\frac{1}{n}\sum_{k=1}^{n}|E(M_{n+1}X_k)| < \frac{2}{n} \qquad (n \ge 1).$$

Substantial improvements in the maximum average covariance may be obtained if memory updates are allowed access to all past inputs (and not just the last input). Let $\mathcal{F}$ denote the family of all (probabilistic) functions mapping $\{-1,1\}^n$ into $\{-1,1\}$.

*Theorem 4:* For every positive integer $n$,

$$\max_{f_1,\cdots,f_n} \frac{1}{n}\sum_{k=1}^{n} E(M_{n+1}X_k) < \max_{f\in\mathcal{F}} \frac{1}{n}\sum_{k=1}^{n} E(X_k f(X_1,\cdots,X_n)) \sim \frac{\sqrt{2}}{\sqrt{\pi n}} \qquad (n\to\infty).$$

*Proof:* The first inequality is immediate. Now, for any $f\in\mathcal{F}$, we have

$$\sum_{k=1}^{n} E(X_k f(X_1,\cdots,X_n)) = E\left(f(X_1,\cdots,X_n)\sum_{k=1}^{n} X_k\right) \le E\left|\sum_{k=1}^{n} X_k\right|,$$

(as $f(X_1,\cdots,X_n)\in\{-1,1\}$), with equality if $f$ is chosen to be the *majority* function: for any choice of Boolean variables $x_1,\cdots,x_n\in\{-1,1\}$ let $N^+$ denote the number of variables, $x_i$, that take the value $+1$, and let $N^- = n - N^+$ denote the number of variables, $x_j$, that take the value $-1$; we define the majority function, $f^M(x_1,\cdots,x_n)$, by

$$f^M(x_1,\cdots,x_n) = \begin{cases} -1, & \text{if } N^- > N^+, \\ 1, & \text{if } N^- \le N^+. \end{cases}$$

Let us denote by $S_n$ the random walk

$$S_n = \sum_{k=1}^{n} X_k.$$

We then have

$$\begin{aligned}\max_{f\in\mathcal{F}} \frac{1}{n}\sum_{k=1}^{n} E(X_k f(X_1,\cdots,X_n)) &= \frac{1}{n}E(|S_n|) \\ &= \frac{1}{n}\sum_{j=0}^{\lfloor \frac{n}{2}\rfloor}[n-2j]\binom{n}{j}2^{-n+1} \\ &= \binom{n-1}{\lfloor \frac{n}{2}\rfloor}2^{-n+1},\end{aligned}$$

with the last equality following by the application of standard binomial identities. An application of Stirling's formula now yields the required result. □

The average covariance cannot, hence, exceed the order of $1/\sqrt{n}$ even if we allow (binary) update rules with unlimited access to past history.

## III. Related Issues

Thus far, we have been mainly concerned with update rules with two Boolean arguments and producing one Boolean variable. The state of memory at epoch $n+1$ is, hence, a Boolean function of $n$ Boolean variables (the inputs, $X_1,\cdots,X_n$) taken in sequence and passed through a cascade of Boolean functions of two Boolean variables. A natural question that arises is how many *deterministic* Boolean functions of $n$ variables can be constructed in this fashion out of the total of $2^{2^n}$ Boolean functions of $n$ variables?

Let $g_k\colon\{-1,1\}^2\to\{-1,1\}$, $k\ge 2$ denote a sequence of (deterministic) Boolean functions of two Boolean variables. We recursively form a sequence of Boolean functions of $k$ Boolean variables, $f_k\colon\{-1,1\}^k\to\{-1,1\}$, for $k\ge 2$, as follows:

$$\begin{aligned} f_2(X_1,X_2) &= g_2(X_1,X_2), \\ f_k(X_1,\cdots,X_{k-1},X_k) &= g_k(f_{k-1}(X_1,\cdots,X_{k-1}),X_k) \qquad (k\ge 3).\end{aligned}$$

Let $\mathcal{F}_k$ denote the family of all (deterministic) Boolean functions of $k$ Boolean variables, $f_k$, constructed recursively, for every choice of functions $g_k$.

*Theorem 5:*

$$|\mathcal{F}_n| = \frac{2}{5}6^n + \frac{8}{5}, \qquad n\ge 2.$$

*Remark:* In fact, it is easy to see that $2^n \le |\mathcal{F}_n| \le 16^n$. Clearly, this count falls far short of the $2^{2^n}$ possible Boolean functions of $n$ Boolean variables.

*Proof:* The demonstration is inductive in nature. For $n=2$ we clearly have

$$|\mathcal{F}_2| = 16,$$

as there are $2^4$ Boolean functions of two Boolean variables. Now, for $n\ge 3$ we claim the following recursion holds:

$$|\mathcal{F}_n| = 4 + 12\left(\frac{|\mathcal{F}_{n-1}|}{2} - 1\right) = 6|\mathcal{F}_{n-1}| - 8.$$

To establish this it is helpful to consider the table of all 16 Boolean functions of two Boolean variables, $X$ and $Y$, illustrated in Fig. 5. Note that two of the possible functions (the first row) are the constant functions, which depend on neither $X$ nor $Y$, and that two more functions (the second row) depend only on $X$ and not on $Y$. All the remaining 12 functions depend explicitly on $Y$. Let us call a set of Boolean functions *independent* if no function in the set is the complement of another function in the set. Now, by symmetry, the complement of every function in $\mathcal{F}_{n-1}$ is also in $\mathcal{F}_{n-1}$. Hence, we can find a maximal set of $|\mathcal{F}_{n-1}|/2$ independent functions in $\mathcal{F}_{n-1}$. Clearly, one of these functions is the constant function so that there are $|\mathcal{F}_{n-1}|/2-1$ functions in a maximal set of independent functions in $\mathcal{F}_{n-1}$ which depend *explicitly* on one or more of the variables $X_1,\cdots,X_{n-1}$.

Now consider functions, $g_n(f_{n-1}(X_1,\cdots,X_{n-1}),X_n)$. Let us identify with $X_n$ the variable $X$ and with

| $g(X,Y)$ | $\bar{g}(X,Y)$ |
|---|---|
| $1$ | $-1$ |
| $X$ | $\bar{X}$ |
| $Y$ | $\bar{Y}$ |
| $X \wedge Y$ | $\bar{X} \vee \bar{Y}$ |
| $X \wedge \bar{Y}$ | $\bar{X} \vee Y$ |
| $\bar{X} \wedge Y$ | $X \vee \bar{Y}$ |
| $\bar{X} \wedge \bar{Y}$ | $X \vee Y$ |
| $(X \wedge Y) \vee (\bar{X} \wedge \bar{Y})$ | $(X \wedge \bar{Y}) \vee (\bar{X} \wedge Y)$ |

Fig. 5. A tabulation of the 16 possible Boolean functions of two Boolean variables, $X \in \{-1,1\}$ and $Y \in \{-1,1\}$. The first column enumerates a set of eight distinct Boolean functions of these two variables, none of which is a complement of another function in the column. The second column lists the complements of the functions listed in the first column; (each row gives a function and its complement.) We use the notation $\bar{}$ to denote complement (logical NOT), $\wedge$ to denote conjunction (logical AND), and $\vee$ to denote disjunction (logical OR).

$f_{n-1}(X_1,\cdots,X_{n-1})$ the variable $Y$ in the table of Boolean functions of two Boolean variables. Each of the independent, nonconstant functions, $Y$, in $\mathscr{F}_{n-1}$ yields 12 distinct functions depending explicitly on $Y$ in $\mathscr{F}_n$, as can be verified from Fig. 5. (By symmetry, the complement, $\bar{Y}$, of each independent, nonconstant function $Y$ in $\mathscr{F}_{n-1}$ yields the same set of 12 distinct functions as does $Y$.) There are, hence, $12(|\mathscr{F}_{n-1}|/2-1)$ distinct functions in $\mathscr{F}_n$ that depend explicitly on one or more of the variables $X_1,\cdots,X_{n-1}$. Adding in the four functions—the two constant functions, and the functions returning the values $X_n$ and $\bar{X}_n$—which are independent of the variables $X_1,\cdots,X_{n-1}$ completes the count. □

A natural extension to the maximin problem is to consider how much information can be stored about the past if now (say) $m \geq 1$ bits of memory are available. This issue is still open. The simple strategy of interleaving the input sequence across the memory bits (equivalently, partitioning the input sequence into $m$ equal length subsequences and apportioning one bit of memory to each subsequence), for instance, effectively reduces the problem to a one bit memory problem with an equivalent "reduced sequence length" of $n/m$. With the mutual information measure, for instance, if $m$ bits are available for the memory, we have

$$\sup\min I(M_{n+1};X_k) \geq \frac{m^2}{2n^2 \ln 2} + O\left(\frac{1}{n^4}\right).$$

Another approach giving the same results is to update each bit of memory independently. Substantial improvements over these straightforward gains may, however, be possible if more complex update strategies are used.

The tightening of the information bounds shown in the previous section is open. Specifically, it appears plausible that we need to consider only monotone symmetric update rules. As noted earlier, if this conjecture holds true, then $I_n = 1/n$ and $J_n \sim 1/2n^2 \ln 2$ with equality holding in both cases for a choice of the harmonic update rule.

Another extension of the problem is to consider input sequences drawn from nonsymmetric Bernoulli trials, and in general, i.i.d. inputs $X_k$, $k \geq 1$ drawn from a distribution on the real line (with a suitable second moment constraint). The maximin problem with one or more bits of available memory is open for this case.

The maximin problem analyzed here has implications to questions on the information storage capacity of neural networks. A formal *McCulloch–Pitts neuron* is characterized by $n$ real weights, $w_1,\cdots,w_n$; it accepts $n$ binary inputs, $u_1,\cdots,u_n \in \{-1,1\}$ and produces a binary output $v \in \{-1,1\}$ according to the threshold rule

$$v = \begin{cases} -1, & \text{if } \sum_{j=1}^{n} w_j u_j < 0, \\ 1, & \text{if } \sum_{j=1}^{n} w_j u_j \geq 0. \end{cases}$$

In a network of formal neurons information can be regarded as being stored in the weights. If the weights are allowed to range over only a finite set of values, a cogent question is *how much information is stored per bit of weight*?

As a specific instance, consider a classification problem on vertices of the $n$ cube. Let $\boldsymbol{u}^1,\cdots,\boldsymbol{u}^m \in \{-1,1\}^n$ be $m$ randomly chosen patterns (with components drawn from symmetric Bernoulli trials). Let $\mathscr{A}(n,m)$ denote the attribute (of the $m$-set of patterns) that there is a choice of weight vector, $\boldsymbol{w}$, such that $\langle \boldsymbol{w}, \boldsymbol{u}^q \rangle > 0$, $q = 1,\cdots,m$. (Alternatively, $\mathscr{A}(n,m)$ is the attribute that a formal neuron classifies each of the patterns properly.) We say that $C_n$ is a *capacity function* for the attribute $\mathscr{A}(n,m)$ if, for every $\lambda > 0$, as $n \to \infty$:

a) $P\{\mathscr{A}(n,m)\} \to 1$, if $m \leq (1-\lambda)C_n$;
b) $P\{\mathscr{A}(n,m)\} \to 0$, if $m \geq (1+\lambda)C_n$.

The capacity function specifies, in a sense, the largest size of random problem that can be reliably done by a linear threshold element or formal neuron. Equivalently, it can be thought of as specifying the maximum amount of information that can be reliably stored in the weights. This interpretation is particularly persuasive when the neural weights are constrained to be binary. In this case, each weight, $w_j \in \{-1,1\}$, has to store information about the $j$th component of each pattern,

$$u_j^1,\cdots,u_j^m \in \{-1,1\},$$

so that the information stored per bit of weight is directly related to the capacity. In this form the problem can be seen to be strongly related to the maximin problem we have analyzed here. A rigorous analysis shows that the capacity of a neuron with binary weights is, in fact, linear in $n$.[5] In a succeeding paper, we illustrate how the ideas developed in this paper can be used in the training of formal neurons with binary weights, and provide rigorous capacity calculations [4].

## Acknowledgment

The authors are indebted to J. Komlós for bringing the problem analyzed in this paper to our attention, and for going through an earlier version of the paper with helpful comments. The authors are also much obliged to A. Barron who showed them the simple convex combination argument using which Theorem 3 follows directly from the proof of Theorem 1.

## References

[1] J. Komlós, L. Rejtö, and G. Tusnády, "Learning with finite memory," preprint.

[2] T. Cover, "On geometrical and statistical properties of systems of linear inequalities with applications to pattern recognition," *IEEE Trans. Elect. Comput.*, vol. EC-14, pp. 326–334, June 1965.

[3] Z. Füredi, "Random polytopes in the $d$-dimensional cube," *Discrete Comput. Geom.*, vol. 1, pp. 315–319, 1986.

[4] S. S. Venkatesh, "On learning binary weights for majority functions," in *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, L. G. Valiant and M. K. Warmuth, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 257–266.

[5] The capacity function for a neuron with real weights is $2n$[2, 3], so that the restriction to binary weights does not seem to seriously reduce capacity.

To appear:
J. of Computer & Systems Sciences
vol. 46, 1993 (in press)

# DIRECTED DRIFT: A NEW LINEAR THRESHOLD ALGORITHM FOR LEARNING BINARY WEIGHTS ON-LINE*

Santosh S. Venkatesh[†]
Department of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104
*E-mail*: venkatesh@ee.upenn.edu

Submitted May 1, 1990
Revised January 31, 1991

**Abstract**

Learning real weights for a McCulloch-Pitts neuron is equivalent to linear programming and can hence be done in polynomial time. Efficient local learning algorithms such as Perceptron Learning, further, guarantee convergence in finite time. The problem becomes considerably harder, however, when it is sought to learn *binary* weights; this is equivalent to integer programming which is known to be NP-complete. A family of probabilistic algorithms which learn binary weights for a McCulloch-Pitts neuron with inputs constrained to be binary is proposed here, the target functions being majority functions of a set of literals. These algorithms have low computational demands and are essentially local in character. Rapid (average-case) quadratic rates of convergence for the algorithm are predicted analytically and confirmed through computer simulations when the number of examples is within capacity. It is also shown that for the functions under consideration, Perceptron Learning converges rapidly (but to an, in general, *non-binary* solution weight vector).

## 1 INTRODUCTION

We consider learning in the context of *linearly separable* functions. Given an arbitrary linearly separable dichotomy of a finite set of patterns, the Perceptron Training Algorithm [1] guarantees convergence in *finite time* of an iteratively updated sequence of weight vectors to a *real* solution vector which separates the dichotomy. The problem becomes considerably harder, however, when we are required to learn *binary* weights for a linearly separable problem. The problem of learning real weights for a McCulloch-Pitts neuron is equivalent to linear programming, for which there exist polynomial time algorithms. (The Perceptron Learning Rule is an on-line procedure which, as we will see in the sequel, can converge extremely rapidly under moderate conditions. Similar

*Presented in part at the *Workshop on Neural Networks for Computing*, Snowbird, Utah, April 1989, and the *IEEE International Symposium on Information Theory*, San Diego, California, January 1990.

[†] This research was supported in part by the National Science Foundation under research grant EET-8709198 and by the Air Force Office of Scientific Research under research grant AFOSR-89-0523.

conclusions are also reported by Baum [2] under slightly different hypotheses.) Learning binary weights for a McCulloch-Pitts neuron is, however, equivalent to integer programming, which is known to be NP-complete [3].

Notwithstanding the apparent difficulty of the learning problem in this case, the potentially lower cost and simplicity of neural networks comprised of binary interconnections as opposed to real weights makes such circuits rather appealing practically. Recent theoretical results also bolster the usage of such circuits: the computational capacity of a neuron with binary weights remains comparable to that of a neuron with real weights [4, 5]. The development of efficient heuristics for learning binary weights (paralleling the development of such algorithms as backpropagation for neural circuits with real interconnections) is, hence, critical if the cost advantages promised by binary circuits are to be realised.

We present here a new family of probabilistic algorithms which learn binary weights for a neuron in an on-line setting. The target functions here are weighted linear threshold functions with weights from $\{-1, 1\}$ which are defined on a domain of binary $n$-tuples, $\{-1, 1\}^n$. In particular, the target functions are majority functions of a set of literals, that is, majority functions that may have any of their inputs complemented. Given a partial Boolean function defined on a subset of $m$ points (patterns) from this class, or alternatively, given a dichotomy of $m$ points in $\{-1, 1\}^n$ which can be linearly separated with weights from $\{-1, 1\}$, the randomised algorithm described here iteratively adjusts the weights until a solution (binary) weight vector which separates the dichotomy is found. The principal advantage the proposed algorithm has over Perceptron Learning is that, not only is the solution vector generated by the procedure binary, but the weights remain confined to the domain $\{-1, 1\}$ throughout the entire learning process. The algorithm, as we will see, converges rapidly to a solution when the number of patterns to be dichotomised is within the computational capacity of the neuron. An interesting feature of the algorithm is that it is *local*, which makes it appealing from an implementation perspective.

In the next section we set up the learning protocol and describe the algorithm. We derive some preliminary results on the expected time of first passage of random walks to given boundaries in Section 3. In Section 4 we analyse the algorithm and show quadratic initial rates of convergence when the number of training examples is within the computational capacity of the threshold element. The analysis here is for the average case.* We also compare the results obtained with the rate of convergence of the Perceptron Training Algorithm: we show that the Perceptron Algorithm converges in the worst case with a mistake bound $O(n^2)$ to a real solution vector under the constraint that there exists a binary solution vector within the solution space. We also present an average case analysis of a modification of the Perceptron Learning Algorithm, in the spirit of the proposed Directed Drift Algorithm, wherein a single weight component is updated at a time. In Section 5 we present simulations and discussions of the algorithm.

*On notation*: We will use the symbol $\mathbb{B}$ to denote the set $\{-1, 1\}$. If $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$ are points in real Euclidean $n$-space, we denote by $\langle \mathbf{x}, \mathbf{y} \rangle$ the inner-product $\sum_{j=1}^{n} x_j y_j$. Following J. Riordan we use the word *epoch* to denote *points* on the time axis. A physical weight update may take some time, but we will assume updates are timeless and occur at epochs.† We define the function sgn $: \mathbb{R} \rightarrow \mathbb{B}$ by $\operatorname{sgn} x = x/|x|$ if $x \neq 0$ and $\operatorname{sgn} 0 = 0$. All logarithms in the exposition are to base $e$. Finally, if $\{x_n\}$ and $\{y_n\}$ are positive sequences, we

*Directed Drift is a randomised algorithm, and arbitrarily bad worst-case results are possible. The probability of such occurences is small, however, and is governed by the extreme tails of the underlying probability distribution.

†In his text, W. Feller credits J. Riordan with initiating the usage of the word epoch in such situations [7, page 73].

denote: $x_n = O(y_n)$ if there is a positive constant $K$ such that $x_n/y_n < K$ for all $n$; $x_n \sim y_n$ if $x_n/y_n \to 1$ as $n \to \infty$.

# 2 LEARNING

## 2.1 The Setting

We are given a set of *patterns*, $\mathcal{U} \subset \mathbb{R}^n$, and a function $f : \mathcal{U} \to \mathbb{B}$ which is linearly separable: specifically, there exists a *solution weight vector*, $\mathbf{w}^s \in \mathbb{R}^n$, such that

$$\operatorname{sgn}\{\langle \mathbf{w}^s, \mathbf{u}\rangle\} = f(\mathbf{u}) \tag{1}$$

for every choice of pattern $\mathbf{u} \in \mathcal{U}$. We call the function $f$ the *target function* (also known as the *target concept* in the literature on Learning Theory). The target functions are, hence, majority functions of a set of literals. Clearly, $f$ realises a dichotomy of $\mathcal{U}$. Without loss of generality we assume that $f(\mathbf{u}) = 1$ for every pattern $\mathbf{u} \in \mathcal{U}$.[‡]

An algorithm for *learning from examples* is a procedure where learning takes place in a sequence of trials. The protocol is as follows:

1° At epoch $t$ the system is characterised by a weight vector, $\mathbf{w}[t]$, and receives an example pattern, $\mathbf{u}[t]$, drawn from $\mathcal{U}$.

2° The system produces a *response*, -1 or 1, according to the sign of $\langle \mathbf{w}[t], \mathbf{u}[t]\rangle$.

3° A new weight vector, $\mathbf{w}[t+1]$, is generated based on the current response, weight vector, $\mathbf{w}[t]$, and example, $\mathbf{u}[t]$.

The procedure is carried out iteratively, and is terminated if a solution weight vector is obtained. We call the sequence of examples, $\{\mathbf{u}[t]\}_{t=1}^{\infty}$, the *training sequence*, and the sequence of weight vectors, $\{\mathbf{w}[t]\}_{t=1}^{\infty}$, the *learning sequence.* If the procedure terminates in a finite time, we say that the learning algorithm has *learnt* the function $f$. We will be interested in the *mistake bound*—the number of classification mistakes the learning algorithm makes on the set of examples before it learns the given function. For our purposes, the mistake bound is equal to the number of updates of the weight vector before the function is learnt. We denote the mistake bound by $T$.

In the sequel, we will further restrict the set of patterns, $\mathcal{U}$, to be drawn from the vertices of the $n$-cube, $\mathbb{B}^n$, and require that there is a binary solution weight vector, $\mathbf{w}^s \in \mathbb{B}^n$, for $f$. A fourth restriction that we will require of any binary learning algorithm is:

4° The initial choice of weight vector, $\mathbf{w}[1]$, is arbitrary, subject only to its being chosen from $\mathbb{B}^n$, and the learning algorithm generates binary weight vectors, $\mathbf{w}[t] \in \mathbb{B}^n$, at each epoch of the learning process.

We will, thus, be constrained to looking at algorithms which make only bit changes in the weight vector at each epoch. Specifically, the weights are confined to the domain $\{-1, 1\}$ throughout the learning process. This situation may be compared to Perceptron Learning, where the weights typically grow in magnitude during the learning process.

[‡] If $f(\mathbf{u}) = -1$ then $f(-\mathbf{u}) = 1$ as can be easily seen from (1). Replacing each pattern in $\mathcal{U}$ for which $f(\mathbf{u}) = -1$ by $-\mathbf{u}$ we obtain a corresponding set of patterns $\hat{\mathcal{U}}$; if $\mathbf{w}^s$ is any solution weight vector separating the dichotomy of $\mathcal{U}$ specified by $f$ then all patterns in $\hat{\mathcal{U}}$ lie on the same (positive) side of the hyperplane corresponding to $\mathbf{w}^s$, and conversely.

## 2.2 Directed Drift Algorithms

We present here a family of probabilistic algorithms for binary learning. We call these algorithms *Directed Drift Algorithms* because, as we shall see, they share some similarities with asymmetric random walks with a preferred direction toward a solution.

Let $\mathcal{U}$ be any subset of patterns from $\mathbb{B}^n$, and let $\{\mathbf{u}[t]\}$ be any training sequence such that each of the patterns in $\mathcal{U}$ appears infinitely often.§ Let $\{\mathbf{w}[t]\}$ denote a binary learning sequence. For each epoch, $t$, we denote by $J[t]$ the subset of indices for which the corresponding components of $\mathbf{w}[t]$ and $\mathbf{u}[t]$ are opposite in sign:

$$J[t] = \{\, j : w_j[t] \neq u_j[t] \,\}.$$

**Single bit updates** We begin with the simplest version of the algorithm where no more than a single component of the weight vector is updated per epoch.

BASE: $\mathbf{w}[1] \in \mathbb{B}^n$ *is chosen arbitrarily.*

ITERATION: *The algorithm's response is predicated upon whether a correct or incorrect response is obtained at the current epoch, $t$.*

- *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle > 0$, *then the weight vector is left unchanged:* $\mathbf{w}[t+1] = \mathbf{w}[t]$.
- *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle \leq 0$, *then an index $j[t]$ is picked at random from the set of indices, $J[t]$, of mismatched components. The new weight vector is now formed according to the following rule:*

$$w_j[t+1] = \begin{cases} w_j[t] & \text{if } j \neq j[t] \\ -w_j[t] & \text{if } j = j[t]. \end{cases} \tag{2}$$

The intuition behind the algorithm is as follows. If a binary solution vector, $\mathbf{w}^s \in \mathbb{B}^n$, exists, then necessarily we must have $\langle \mathbf{w}^s, \mathbf{u} \rangle = \sum_{j=1}^n w_j^s u_j > 0$ for each pattern $\mathbf{u} \in \mathcal{U}$. As there is a contribution of +1 to the sum if two corresponding components of $\mathbf{w}^s$ and $\mathbf{u}$ have the same sign, and -1 if the signs are mismatched, it follows that the binary solution vector has more component sign matches than mismatches with *each* pattern in $\mathcal{U}$.

Now the algorithm updates the current estimate of the weight vector if and only if the current pattern from the training sequence is misclassified. A weight vector update results in a randomly chosen mismatched component of the weight vector being flipped to the sign of the corresponding pattern component. Since there is a probability better than a half that a randomly specified component of any pattern has the same sign as the corresponding component of the binary solution vector, it follows that at each epoch the *a priori* probability that the weight vector update is in the direction of the binary solution vector is better than a half. We will explore this more formally in the sequel.

**Several bit updates** The algorithm can be simply extended to accommodate more than a single bit update per epoch. Let $\{N_t\}$ be a sequence of integers with $0 \leq N_t \leq n/2$.

§Note that $\mathcal{U} \subset \mathbb{B}^n$ is a finite set of patterns. If $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ is an $m$-set of patterns, then we can, for instance, obtain valid training sequences by cycling through the patterns or choosing a pattern randomly at each epoch.

BASE: *$\mathbf{w}[1] \in \mathbb{B}^n$ is chosen arbitrarily.*

ITERATION: *As before, updates are made only if the current pattern from the training sequence is misclassified.*

- *If $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle > 0$, then the weight vector is left unchanged: $\mathbf{w}[t+1] = \mathbf{w}[t]$.*
- *If $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle \leq 0$, then $N_t$ indices $j_1[t], \ldots, j_{N_t}[t]$ are picked at random from the set of indices, $J[t]$, of mismatched components. The new weight vector is now formed according to the following rule:*

$$w_j[t+1] = \begin{cases} w_j[t] & \text{if } j \notin \{j_1[t], \ldots, j_{N_t}[t]\} \\ -w_j[t] & \text{if } j \in \{j_1[t], \ldots, j_{N_t}[t]\}. \end{cases}$$

The sequence $N_t$ specifies the number of bits to be changed at each update epoch, and the proper choice of this sequence is clearly critical to the functioning of the algorithm. This is analogous to choosing an appropriate *cooling schedule* for *simulated annealing* [6].

### 2.3 Perceptron Training Algorithm

A geometrical appreciation of the Directed Drift Algorithm can be obtained from a consideration of the classical Perceptron Training Algorithm. Let $\{\mathbf{u}[t]\}$ be a training sequence of patterns, and let $\{\mathbf{w}[t]\}$ denote a learning sequence of *real* weight vectors.

**Fixed increment Perceptron Training** This is the simplest form of Perceptron learning. Let $\beta > 0$ be fixed.

BASE: *The initial choice of weight vector is arbitrary. For simplicity we take $\mathbf{w}[1] = \mathbf{0}$.*

ITERATION: *As before, weight vector updates are made only if a pattern is misclassified.*

- *If $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle > 0$, then the weight vector is left unchanged: $\mathbf{w}[t+1] = \mathbf{w}[t]$.*
- *If $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle \leq 0$, then set $\mathbf{w}[t+1] = \mathbf{w}[t] + \beta \mathbf{u}[t]$.*

The Perceptron Training Algorithm is known to converge to a real solution vector (if it exists) in finite time [1]. Geometrically speaking, the situation is as depicted in Figure 1(a). When a pattern from the training sequence is misclassified by the current estimate of the weight vector, the weight vector update is in the direction of the misclassified pattern. This idea of updating in the direction of the misclassified pattern is extended in the Directed Drift Algorithms. The situation is as depicted schematically in Figure 1(b). The updates, being constrained to be binary, are not directly in the direction of the misclassified pattern; nevertheless, the update lies in the positive half space corresponding to the binary pattern vector so that the updated weight vector is more apt to classify the pattern correctly.

**Single component Perceptron Training** The basic randomisation idea behind single bit update Directed Drift is easily extended to single component Perceptron Learning, where a single component of the weight vector is modified at each update epoch (as opposed to traditional Perceptron Learning where all components are modified at each update epoch).

For each epoch, $t$, let $I[t]$ denote the subset of indices for which the corresponding components of $\mathbf{w}[t]$ and $\mathbf{u}[t]$ are opposite in sign:

$$I[t] = \{\, i : u_i[t] \neq \operatorname{sgn} w_i[t] \,\}.$$

BASE: *For simplicity, take* $\mathbf{w}[1] = \mathbf{0}$.

ITERATION: *The algorithm's response is predicated, as usual, upon whether a correct or incorrect response is obtained at the current epoch, $t$.*

- *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle > 0$, *then the weight vector is left unchanged:* $\mathbf{w}[t+1] = \mathbf{w}[t]$.
- *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle \leq 0$, *then an index $i[t]$ is picked at random from the set of indices, $I[t]$, of mismatched components. The new weight vector is now formed according to the following rule:*

$$w_i[t+1] = \begin{cases} w_i[t] & \textit{if } i \neq i[t] \\ w_i[t] + u_i[t] & \textit{if } i = i[t]. \end{cases} \tag{3}$$

In this variation, a single bit is added (subtracted) from a randomly chosen component at each update epoch. The mistake bound hence coincides with the number of component updates, as in single bit update Directed Drift. For fixed increment Perceptron Learning, of course, the number of component updates is $n$ times the mistake bound.

# 3 RANDOM WALKS

An estimate of the rates of convergence of the randomised algorithms described above may be obtained by appealing to notions from random walks and the geometric theory of paths.

Let $\{X_j\}$ be a sequence of Bernoulli trials with success probability $p_n = 1/2 + \beta_n$, $0 < \beta_n \leq 1/2$ depending on a parameter $n$:

$$X_j = \begin{cases} 1 & \text{with probability } p_n = 1/2 + \beta_n \\ -1 & \text{with probability } q_n = 1/2 - \beta_n. \end{cases}$$

Let $S_k = \sum_{j=1}^{k} X_j$ denote a random walk with positive drift, $\mathbf{E} S_k = 2k\beta_n$. We are interested in estimating the expected time of first passage of the random walk to some specified boundary, $B(n,k)$.

## 3.1 Fixed Boundary

Let us first consider the case of a fixed (one-sided) boundary at $n$. Define

$$T_1(n) = \inf\{k : S_k \geq n\}.$$

For this case, the theory of generating functions can be readily invoked to estimate the expected time of first passage to the boundary (cf. Feller [7, Chapter III]). We have the following estimate:

**Proposition 3.1** $\mathbf{E} T_1(n) = n/2\beta_n$ *for every $n$.*

PROOF: Let $\alpha_l(k)$ denote the probability that the random walk makes a first passage $l$ units to the right of the starting point in $k$ steps. We then have $\mathbf{E} T_1(n) = \sum_{k=0}^{\infty} k\alpha_n(k)$.

As a first passage through $n$ must necessarily involve a prior first passage through $n-1$, we immediately have the convolutional relation

$$\alpha_n(k) = \sum_{j=1}^{k-1} \alpha_{n-1}(j)\alpha_1(k-j). \tag{4}$$

$$P\{M_{n+1} = X_k = 1\} = \frac{1}{4}[p_n(-1,-1) + p_n(-1,1)] + \left(1 - \frac{\psi_n}{2}\right) P\{M_n = X_k = 1\}.$$

$$-\frac{}{4} \sum_{i=k+1} [\{\bar{\omega}_k p_i(-1,1) - \omega_k p_i(1,-1)\} + \{\bar{\omega}_k p_i(-1,-1) - \omega_k p_i(1,1)\}] \prod_{j=i+1}^{n} \left(1 - \frac{\psi_j}{2}\right).$$

Let $A_l(s)$ denote the generating function for the probability distribution $\alpha_l(k)$; i.e.,

$$A_l(s) = \sum_{k=0}^{\infty} \alpha_l(k) s^k.$$

From equation (4) we hence have

$$A_n(s) = A_{n-1}(s) A_1(s) = [A_1(s)]^n, \tag{5}$$

with the latter equality following by induction.

To evaluate $A_1(s)$ we need to evaluate the probabilities, $\alpha_1(k)$, of a first transition one unit to the right in $k$ steps. We note that

$$\begin{aligned} \alpha_1(0) &= 0 & (6) \\ \alpha_1(1) &= p_n. & (7) \end{aligned}$$

Now, a first transition one unit to the right in $k \geq 2$ steps must necessarily involve an initial step to the left, followed by a first transition one unit to the right (back to the origin), and a final first transition one more unit to the right. Hence

$$\alpha_1(k) = q_n \sum_{j=1}^{k-2} \alpha_1(j) \alpha_1(k-j-1), \qquad k = 2, 3, \ldots. \tag{8}$$

Using equations (6), (7), and (8) we now have

$$\begin{aligned} A_1(s) &= \sum_{k=0}^{\infty} \alpha_1(k) s^k \\ &= p_n s + \sum_{k=2}^{\infty} \alpha_1(k) s^k \\ &= p_n s + q_n \sum_{k=2}^{\infty} \sum_{j=1}^{k-2} \alpha_1(j) \alpha_1(k-j-1) s^k \\ &= p_n s + q_n s [A_1(s)]^2. \end{aligned}$$

Solving for $A_1(s)$ we finally obtain

$$A_1(s) = \frac{1 - \sqrt{1 - 4 p_n q_n s^2}}{2 q_n s}.$$[¶]

Substituting in (5) we obtain

$$A_n(s) = \sum_{k=0}^{\infty} \alpha_n(k) s^k = \left[\frac{1 - \sqrt{1 - 4 p_n q_n s^2}}{2 q_n s}\right]^n.$$

We can now directly compute the expected time of first passage $n$ units to the right by

$$\mathbf{E}\, T_1(n) = A_n'(1) = \frac{n}{p_n - q_n}.$$

The substitution $p_n - q_n = 2\beta_n$ completes the proof. ∎

[¶] We discard the positive root as it grows without bound as $s \to 0$, and we require $A_1(0) = 0$.

### 3.2 Receding Boundary

We now consider the case of a receding (two-sided) boundary at $\pm\sqrt{kn}$. Define

$$T_2(\sqrt{kn}) = \inf\{k : |S_k| \geq \sqrt{kn}\}.$$

It is difficult to get explicit closed-form expressions when the boundary is not fixed, and we will be satisfied with an asymptotic estimate for $\mathbf{E}\,T_2(\sqrt{kn})$ as $n \to \infty$. Our development follows Siegmund [8, Chapter IX] where a general analysis is presented in the context of nonlinear renewal theory.

Now $T_2(\sqrt{kn})$ (if finite) is the first integer $k$ for which the random variable $S_k^2/k$ exceeds $n$. Now, simple algebraic manipulations yield

$$\frac{S_k^2}{k} = \underbrace{4k\beta_n^2 + 4\beta_n(S_k - 2k\beta_n)}_{A_k} + \underbrace{\frac{1}{k}(S_k - 2k\beta_n)^2}_{B_k}.$$

By the strong law of large numbers we have

$$\begin{aligned} \tfrac{A_k}{k} &\to 4\beta_n^2 \quad \text{with probability one} \\ \tfrac{B_k}{k} &\to 0 \quad \text{with probability one.} \end{aligned} \tag{9}$$

Further, as an easy consequence of Kolmogorov's inequality, we have

$$\frac{1}{k}\max_{1\leq j\leq k} B_j \to 0 \quad \text{in probability.} \tag{10}$$

**Proposition 3.2** *The following assertions hold:*

*a)* $\mathbf{P}\left\{T_2(\sqrt{kn}) < \infty\right\} = 1$ *for all* $n$*;*

*b)* $\frac{T_2(\sqrt{kn})}{n/4\beta_n^2} \to 1$ *in probability as* $n \to \infty$.

Proof: The observation (9) yields $S_k^2/k^2 \to 4\beta_n^2$ with probability one, so that part (a) of the proposition follows.

Now let $K_n = n/4\beta_n^2$. Fix $0 < \epsilon < 1$ and set $K_n' = K_n(1+\epsilon)$. From (9) and (10) we have

$$\frac{1}{k}\max_{1\leq j\leq k}\frac{S_j^2}{j} \to 4\beta_n^2 \quad \text{in probability.}$$

Hence, as $n \to \infty$, we have

$$\begin{aligned} \mathbf{P}\left\{T_2(\sqrt{kn}) > \frac{n}{4\beta_n^2}(1+\epsilon)\right\} &= \mathbf{P}\left\{\max_{1\leq j\leq K_n'}\frac{S_j^2}{j} < n\right\} \\ &= \mathbf{P}\left\{\frac{1}{4\beta_n^2 K_n'}\max_{1\leq j\leq K_n'}\frac{S_j^2}{j} < \frac{1}{1+\epsilon}\right\} \longrightarrow 0. \end{aligned}$$

A similar argument shows that $\mathbf{P}\left\{T_2(\sqrt{kn}) < \frac{n}{4\beta_n^2}(1-\epsilon)\right\} \to 0$. ∎

**Proposition 3.3** $\mathbf{E}T_2(\sqrt{kn}) \sim \frac{n}{4\beta_n^2}$ *as* $n \to \infty$.

PROOF: Let $K_n = n/4\beta_n^2$, as before. For $k \geq 2K_n$ we have

$$\begin{aligned} \mathbf{P}\{T_2(\sqrt{kn}) > k\} \leq \mathbf{P}\left\{\frac{S_k^2}{k} < n\right\} &\leq \mathbf{P}\{A_k < n\} \\ &= \mathbf{P}\left\{\sum_{j=1}^{k}(X_j - 2\beta_n) < -k\beta_n + \frac{n}{4\beta_n}\right\} \\ &\leq \mathbf{P}\left\{\sum_{j=1}^{k}(X_j - 2\beta_n) < -\frac{k\beta_n}{2}\right\}. \end{aligned}$$

The bound above is just the probability of an event in the left tail of the binomial distribution. An application of Chernoff's bound [9] yields:

$$\mathbf{P}\{T_2(\sqrt{kn}) > k\} \leq e^{-kC_n},$$

where

$$C_n = -\left(\frac{1}{2} + \frac{3\beta_n}{4}\right)\log\left(1 + \frac{\beta_n}{2 + 3\beta_n}\right) - \left(\frac{1}{2} - \frac{3\beta_n}{4}\right)\log\left(1 - \frac{\beta_n}{2 - 3\beta_n}\right).$$

We now claim that

$$\sum_{k \geq 2K_n} \mathbf{P}\{T_2(\sqrt{kn}) > k\} \to 0 \qquad (n \to \infty). \tag{11}$$

If $\beta_n$ is bounded away from zero this is clear: $C_n > D > 0$ for some absolute positive constant $D$, and the sum in (11) is just a sum over the exponential tail (note that $K_n \to \infty$ as $n \to \infty$). Now consider the case where $\beta_n \to 0$. Using the Taylor series approximation

$$\log(1 + x) = x - x^2/2 + O(x^3) \qquad (|x| \to 0)$$

it is easy to see that

$$\mathbf{P}\{T_2(\sqrt{kn}) > k\} \leq \exp\left[-\frac{k\beta_n^2}{8}\{1 + O(\beta_n)\}\right].$$

It is now readily verified that the first term in the series in (11) decreases exponentially fast with $n$. This completes the proof of the claim.

The elementary observation $\mathbf{E}(T_2(\sqrt{kn}) | T_2(\sqrt{kn}) > 4K_n) \geq 4K_n$ together with the claim now yields

$$\begin{aligned} \mathbf{E}(T_2(\sqrt{kn}); T_2(\sqrt{kn}) > 4K_n) &\leq 2\mathbf{E}(T_2(\sqrt{kn}) - 2K_n; T_2(\sqrt{kn}) > 4K_n) \\ &\leq 2\mathbf{E}(T_2(\sqrt{kn}) - 2K_n; T_2(\sqrt{kn}) > 2K_n) \\ &\leq 2\sum_{k \geq 2K_n} \mathbf{P}\{T_2(\sqrt{kn}) > k\} \longrightarrow 0 \qquad (n \to \infty). \end{aligned}$$

Hence, the random variables $\left(\frac{n}{4\beta_n^2}\right)^{-1} T_2(\sqrt{kn})$, $n \geq 1$ are uniformly integrable. In addition, by Proposition 3.2(b), $\left(\frac{n}{4\beta_n^2}\right)^{-1} T_2(\sqrt{kn}) \to 1$ in probability. The result follows. ∎

# 4 ANALYSIS

## 4.1 Directed Drift

We consider single bit updates for simplicity. Assume that we have a finite set of patterns, $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\} \subset \mathbb{B}^n$, chosen independently with pattern components drawn from a sequence of symmetric Bernoulli trials. Let $\{\mathbf{u}[t]\}$ be a training sequence, and $\{\mathbf{w}[t]\}$ the learning sequence specified by the rule (2). Let $\{t_k\}$ denote the subsequence of epochs at which patterns from the training sequence are misclassified; i.e.,

$$\langle \mathbf{w}[t_k], \mathbf{u}[t_k] \rangle \leq 0, \qquad k = 1, 2, \ldots.$$

We can write the weight vector updates of equation (2) in the form

$$\mathbf{w}[t_{k+1}] = \mathbf{w}[t_k] + \mathbf{v}[t_k],$$

where $\mathbf{v}[t_k]$ is a vector whose components satisfy

$$v_j[t_k] = \begin{cases} 0 & \text{if } j \neq j[t_k] \\ -2w_j[t_k] & \text{if } j = j[t_k]. \end{cases} \tag{12}$$

Assume that there is a binary solution vector, $\mathbf{w}^s \in \mathbb{B}^n$. Consider the estimate errors

$$\begin{aligned} \mathcal{E}_{k+1} &\triangleq \|\mathbf{w}[t_{k+1}] - \mathbf{w}^s\|^2 \\ &= \|\mathbf{w}[t_k] + \mathbf{v}[t_k] - \mathbf{w}^s\|^2 \\ &= \|\mathbf{w}[t_k] - \mathbf{w}^s\|^2 + \|\mathbf{v}[t_k]\|^2 + 2\langle \mathbf{w}[t_k] - \mathbf{w}^s, \mathbf{v}[t_k]\rangle \\ &= \mathcal{E}_k + 4 + 2\langle \mathbf{w}[t_k], \mathbf{v}[t_k]\rangle - 2\langle \mathbf{w}^s, \mathbf{v}[t_k]\rangle. \end{aligned}$$

Using (2) and (12) we hence obtain

$$\mathcal{E}_{k+1} = \mathcal{E}_k - 4w^s_{j[t_k]} u_{j[t_k]}[t_k].$$

Define the ±1 random variables

$$X_i = w^s_{j[t_i]} u_{j[t_i]}[t_i].$$

By induction we obtain

$$\mathcal{E}_{k+1} = \mathcal{E}_1 - 4\sum_{i=1}^{k} X_i.$$

Upper bounding $\mathcal{E}_1$ by $4n$, and setting $S_k = \sum_{i=1}^k X_i$ we finally obtain

$$0 \leq \mathcal{E}_{k+1} \leq 4(n - S_k).$$

The procedure terminates at the value of $k$ for which the random sum $S_k$ first exceeds $n$. The mistake bound, $T$, hence satisfies $S_T \geq n$, and $S_k < n$, for $k = 1, \ldots, T-1$. The mistake bound is infinite if there exists no such value of $k$, or if there exists no binary solution vector for the choice of patterns, $\mathcal{U}$.

The above is reminiscent of a random walk with a fixed boundary at $n$. In fact, if the $m$-set of patterns $\mathcal{U}$ is chosen independently, then the random variables $X_i$ corresponding to different patterns must be independent.[‖] Let us assume that the training sequence is obtained by cycling through the patterns. For a random initial choice of weight vector a substantial number of patterns will be misclassified, so that a pattern will recur in the update sequence only after $\Omega(m)$ epochs. Through the initial progress of the algorithm, hence, the random sum $S_k = \sum_{i=1}^{k} X_i$ is a sum of independent, identically distributed, ±1 random variables with

$$X_i = \begin{cases} 1 & \text{with probability } p_n = 1/2 + \beta_n \\ -1 & \text{with probability } q_n = 1/2 - \beta_n \end{cases}$$

for some $\beta_n \in (0, 1/2]$. The specific value of the probability $p_n$—the relative frequency of the number of component matches between a binary solution vector and a pattern—depends on the size, $m$, of the set of patterns, $\mathcal{U}$. We clearly have

$$p_n \geq \frac{n/2+1}{n} = \frac{1}{2} + \frac{1}{n},$$

so that $\beta_n \geq 1/n$.

The above argument indicates that the expected time of first passage to $n$ of a one-dimensional random walk, $S_k$, with positive drift, $2k\beta_n$, may be used as an estimate of the expected mistake bound for single bit update Directed Drift. Substituting $\beta_n \geq 1/n$ in Proposition 3.1 we then obtain the estimate $O(n^2)$ for the expected mistake bound when the number of patterns is within capacity.

## 4.2 Perceptron Training

It is instructive to compare the above convergence rates with rates that obtain for Perceptron Training. The classical proofs of the Perceptron Training procedure only guarantee that the procedure converges in finite time if a solution exists: convergence time, however, is strongly dependent on the distribution of (real) patterns, and in the worst case can be exponential in the number of bits needed to specify the pattern distribution. When constraints are placed on the allowable choices of patterns, however, convergence can be much more rapid. To compare mistake bounds with Directed Drift, let us consider Perceptron Training when the patterns are binary, and under the condition that there exists a binary solution vector. (Note, however, that we only require that the Perceptron Training Algorithm return a *real* solution vector.) We show first that the fixed increment Perceptron Training Algorithm converges in the worst case with a mistake bound which increases no faster than quadratically in $n$; alternatively, the total number of component updates before convergence is $O(n^3)$. We follow this with an average case analysis, similar in flavour to the analysis for Directed Drift, for randomised, single component update Perceptron Training, which yields similar results.

Let $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\} \subset \mathbb{B}^n$ be a finite set of patterns, $\{\mathbf{u}[t]\}$ the training sequence, and $\{\mathbf{w}[t]\}$ the learning sequence. As before, let $\{t_k\}$ denote the subsequence of epochs at which patterns from the training sequence are misclassified.

[‖] To facilitate ease of analysis for the nonce we assume that the number of patterns is within the *computational capacity* of a linear threshold element with binary weights. (The capacity is quite large—linear in $n$—and capacities of the order of $n/\log n$ can be easily achieved for rather simple algorithms [4, 5].) For a random choice of patterns we are then assured with arbitrarily high probability asymptotically that there exists a binary solution vector.

**Fixed increment training** The weight vector updates result in

$$\mathbf{w}[t_{k+1}] = \mathbf{w}[t_k] + \beta\mathbf{u}[t_k], \qquad k = 1, 2, \ldots .$$

Assume that there is a binary solution vector, $\mathbf{w}^s \in \mathbb{B}^n$. For a choice of parameter $c > 0$ to be specified shortly, consider now the estimate errors

$$\begin{aligned}\mathcal{F}_{k+1} &\triangleq \|\mathbf{w}[t_{k+1}] - c\mathbf{w}^s\|^2 \\ &= \|\mathbf{w}[t_k] + \beta\mathbf{u}[t_k] - c\mathbf{w}^s\|^2 \\ &= \mathcal{F}_k + \beta^2\|\mathbf{u}[t_k]\|^2 + 2\beta\langle\mathbf{w}[t_k] - c\mathbf{w}^s, \mathbf{u}[t_k]\rangle.\end{aligned}$$

Since $\mathbf{w}^s$ is a binary solution vector we must have that $\langle\mathbf{w}^s, \mathbf{u}[t]\rangle \geq 1$ for each pattern in the training sequence. Furthermore, $\|\mathbf{u}[t]\|^2 = n$ for each pattern in the training sequence, and as $\mathbf{w}[t_k]$ misclassifies pattern $\mathbf{u}[t_k]$ by definition, we also have $\langle\mathbf{w}[t_k], \mathbf{u}[t_k]\rangle \leq 0$. Hence

$$0 \leq \mathcal{F}_{k+1} \leq \mathcal{F}_k + \beta^2 n - 2c\beta.$$

With the assumption that $\mathbf{w}[1] = \mathbf{0}$ we have $\mathcal{F}_1 = \|\mathbf{w}[1] - c\mathbf{w}^s\|^2 = c^2 n$. By induction on the above inequality we then obtain

$$0 \leq \mathcal{F}_{k+1} \leq c^2 n - \beta(2c - \beta n)k.$$

The procedure terminates with a mistake bound

$$T \leq \frac{c^2 n}{\beta(2c - \beta n)}.$$

Choosing $c = \beta n$ minimises this upper bound for the mistake bound. With this choice of $c$ we obtain that the mistake bound for the fixed increment Perceptron Training Rule is $T \leq n^2$ under the constraints of a binary pattern space, and with the requirement that there exist a binary solution vector. Note that this bound is independent of the number of binary patterns, and their distribution. The sole requirement for this estimate of convergence time for the Perceptron Training Rule to hold is that there exist a vertex of the $n$-cube (a binary solution vector) within the convex polyhedral cone defined by the space of real solution vectors. While the mistake bound gives the number of weight updates before convergence, it must also be noted that in the Perceptron Training Rule each update is a synchronous update in which each of the $n$ components of the weight vector are modified (as opposed to the single bit modifications in the simpler version of the Directed Drift Algorithm) and each component modification requires the addition of a real scalar. Thus, in the worst case, the procedure terminates after no more than $O(n^3)$ component updates.

**Single component updates** The weight vector updates of equation (3) can be written in the form

$$\mathbf{w}[t_{k+1}] = \mathbf{w}[t_k] + \mathbf{x}[t_k]$$

where $\mathbf{x}[t_k]$ denotes the vector with components

$$x_i[t_k] = \begin{cases} 0 & \text{if } i \neq i[t_k] \\ u_i[t_k] & \text{if } i = i[t_k]. \end{cases}$$

Let $\mathbf{w}^s \in \mathbb{B}^n$ be a binary solution vector. We have the following bounds on the length-square of the weight vector estimates:

$$\begin{aligned}\mathcal{L}_{k+1}^2 \triangleq \|\mathbf{w}[t_{k+1}]\|^2 &= \|\mathbf{w}[t_k] + \mathbf{x}[t_k]\|^2 \\ &= \mathcal{L}_k^2 + \|\mathbf{x}[t_k]\|^2 + 2w_{i[t_k]}[t_k]u_{i[t_k]}[t_k] \\ &\leq \mathcal{L}_k^2 + 1.\end{aligned}$$

We hence have $\mathcal{L}_{k+1}^2 \leq k$ as a consequence of the choice $\mathbf{w}_1 = 0$. Also, as a consequence of the Cauchy-Schwarz inequality, we have

$$\begin{aligned}\mathcal{L}_{k+1}^2 &\geq \frac{|\langle \mathbf{w}[t_{k+1}], \mathbf{w}^s\rangle|^2}{\|\mathbf{w}^s\|^2} = \frac{1}{n}|\langle \mathbf{w}[t_k], \mathbf{w}^s\rangle + \langle \mathbf{w}^s, \mathbf{x}[t_k]\rangle|^2 \\ &= \frac{1}{n}\left|\sum_{h=1}^{k}\langle \mathbf{w}^s, \mathbf{x}[t_h]\rangle\right|^2 = \frac{1}{n}\left|\sum_{h=1}^{k} w_{i[t_h]}^s u_{i[t_h]}[t_h]\right|^2 .\end{aligned}$$

Define the ±1 random variables

$$X_h = w_{i[t_h]}^s u_{i[t_h]}[t_h],$$

and let $S_k = \sum_{h=1}^k X_h$. We then have

$$\frac{|S_k|}{\sqrt{n}} \leq \mathcal{L}_{k+1} \leq \sqrt{k}.$$

The algorithm terminates at the first instant $k$ for which $|S_k|$ exceeds $\sqrt{kn}$. Arguing as for Directed Drift, we use as an estimate for the expected mistake bound the expected time of first passage of a random walk with positive drift $2k\beta_n \geq 2k/n$ to the two-sided boundary at $\pm\sqrt{kn}$ : Proposition 3.3 then yields the asymptotic estimate $O(n^3)$ for the expected mistake bound as $n \to \infty$.

## 5 SIMULATIONS

Computer simulations indicate that the rapid convergence times predicted by analysis hold when the number of patterns to be loaded lies within the capacity. Mistake bounds are plotted as a function of $n$ in Figure 2. In each plot mistake bounds for each choice of $m$ and $n$ were averaged over 1000 runs of the single bit update Directed Drift Algorithm. In each run of the algorithm an independent set of patterns was drawn from a standard pseudo-random binomial number generator. (To ensure the existence of a binary solution weight vector, a binary $n$-tuple was selected at random as the solution vector, and those patterns lying in the negative half space of the solution vector were reflected.) A random initial binary weight vector was selected as the initial estimate of the weight vector presented to the single bit update Directed Drift Algorithm with the training sequence obtained by cyclically presenting the patterns. At convergence the number of adaptations of the weight vector were stored as the estimate of the mistake bound. The expected mistake bound for the choice of $m$ and $n$ was evaluated by averaging the number of adaptations before convergence over 1000 independent runs (each on an independently chosen data set).

In Figures 2(a) and 2(b) the number of patterns, $m$, was fixed within capacity (at $m = n/4$ and $m = n/2$, respectively) and very rapid convergence times are seen. Expected mistake bounds increase significantly around capacity as illustrated in Figure 2(c) with $m = n$. Mistake bounds finally saturate above capacity at around $2n$ (plotted in Figure 2(d)). When the number of patterns is well above capacity it is not clear whether the algorithm still converges polynomially fast, and this is under investigation. Convergence is still, however, several orders of magnitude faster than an exhaustive search through the vertices of the $n$-cube. (For instance, when $n = 20$ and $m = 40$, exhaustive search requires of the order of $10^6$ steps while Directed Drift converges in about $10^4$ steps.) Early results indicate that order of magnitude improvements in mistake bound may be obtained by updating several bits at a time with an appropriate choice of cooling schedule in the algorithm. We do not have good heuristics at this time, however, for choice of good cooling schedules.

Figures 3 and 4 show a plot of the expected mistake bound, $T$, for single bit update Directed Drift as the number of patterns, $m$, increases for a fixed value of $n$. We observe that the expected mistake bound saturates to a fixed value depending on $n$ when the number of patterns exceeds approximately $3n$ in the range of $n$ we considered in our simulations. Note also the rather abrupt threshold behaviour when the number of patterns exceeds the capacity ($n$). We conjecture that there is a threshold function for the expected mistake bound around the capacity.

The saturation of the mistake bound when the number of patterns exceeds capacity is caused essentially by the shrinkage in the solution space—when the number of patterns exceeds roughly $3n$, then the binary solution vector, if one exists, is essentially unique. We illustrate this in Figure 4: for a fixed value of $n$ we plot simultaneously the expected mistake bound and the relative frequency with which the algorithm terminates in an initially chosen binary solution vector. The saturation in mistake bound around $3n$ is again evident, as well as the threshold behaviour around capacity. Note that the probability that there are multiple solution vectors is the dual of the mistake bound curve: while for a small number of patterns there exist many binary solution vectors, a precipitous drop in the probability of multiple solutions is evidenced around the capacity, and finally around $3n$ there exists only one solution vector with high probability.

This observation has an important consequence from the point of view of generalisation in learning. If the observed saturation of the expected stopping time around $3n$ patterns extends uniformly for all $n$, then *any linearly separable Boolean function for which there exists a binary solution vector can be learnt with no more than $3n$ examples (of the total of $2^n$ instances) of the function drawn at random.* In ongoing work we are attempting to make this rigourous.

In Figure 5 we plot the average number of component updates before convergence versus the number of patterns (with $n = 10$ fixed) for fixed increment Perceptron Training. [The expected mistake bound is an order of magnitude smaller: the average number of component updates is $n$ times the mistake bound for fixed increment Perceptron Training. For single update Directed Drift, as noted before, the mistake bound coincides with the number of component updates.] The sharp threshold behaviour seen in Directed Drift is not so much in evidence here. Saturation again appears to be around twice the capacity ($2n$ for real weights). Note that the average mistake bound is an order of magnitude lower than the worst-case upper bound $O(n^2)$. The derived worst-case bound may, hence, be too conservative. On the same figure we also plot the normalised length, $L/\sqrt{n}$, of the solution vector returned by the algorithm. A similar saturation phenomenon is in evidence, with the length of the solution vector saturating at a value somewhat larger than the length, $\sqrt{n}$, of the binary solution vector.

# 6 CONCLUSIONS

While the general problem of learning binary weights is NP-complete, the rapid convergence of the Directed Drift Algorithms indicates that the *typical* problem may well be tractable even if there exist, perhaps pathological, intractable bad instances. The simplicity of these probabilistic (binary) learning algorithms allows of several possible extensions to networks of neurons—in particular, feedforward structures. This is clearly of some theoretical and practical import, and is under investigation.

**Acknowledgement**

My thanks to: Stephen Judd for staying up working on the convergence rate; David Miller for his assistance in running the simulations of Figures 2 and 3; Inchi Hu for bringing Siegmund's book to my attention; and the two referees who helped improve the presentation of the material.

## References

[1] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Washington, D.C.: Spartan Books, 1962.

[2] E. Baum, "The Perceptron algorithm is fast for non-malicious distributions," preprint.

[3] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: W. H. Freeman, 1979.

[4] S. S. Venkatesh, "Computation and learning in networks with binary synapses," *Workshop on Neural Networks for Computing*, Snowbird, Utah, April 1989.

[5] S. S. Venkatesh and J. Franklin, "How much information can one bit of memory retain about a Bernoulli sequence?" submitted for publication.

[6] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimisation by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[7] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. I, third edition. New York: John Wiley, 1968.

[8] D. Siegmund, *Sequential Analysis: Tests and Confidence Intervals.* New York: Springer-Verlag, 1985.

[9] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on a sum of observations," *Ann. Math. Stat.*, vol. 23, pp. 493–507, 1952.

## Figure Captions

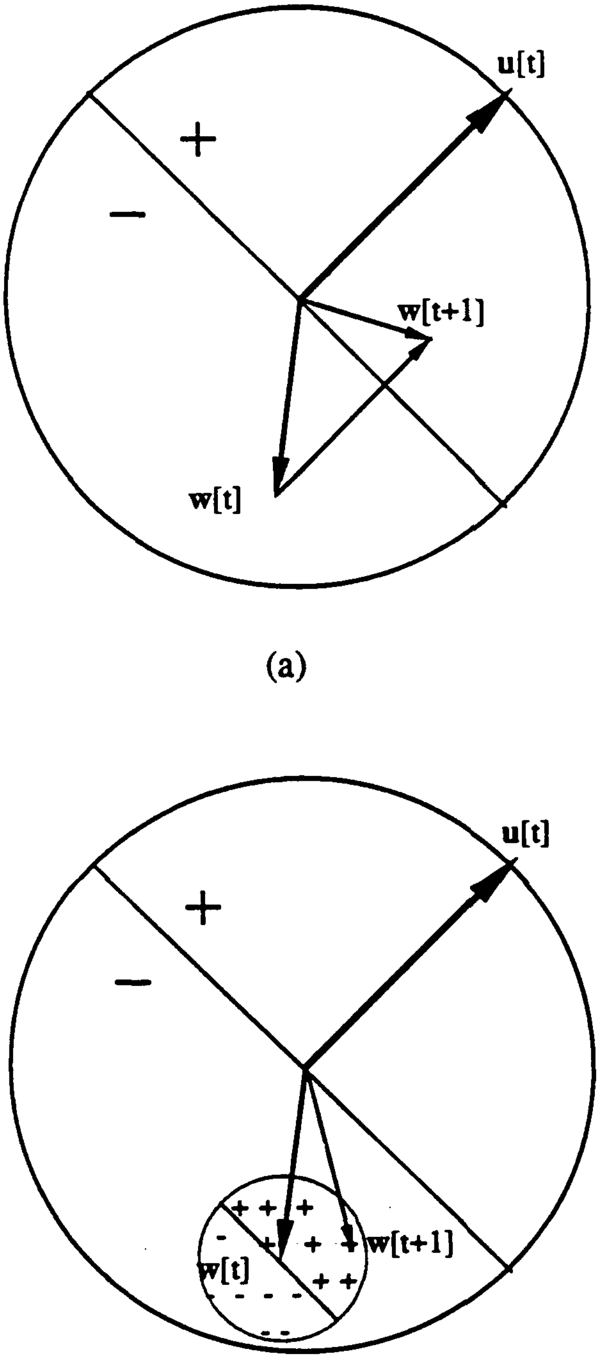
**Fig. 1** (a) A schematic depiction of the incremental change in the current weight vector, $\mathbf{w}[t]$, made by the Perceptron Training Rule. The change is in the direction of the misclassified pattern, $\mathbf{u}[t]$, and the resultant weight vector, $\mathbf{w}[t+1]$, is more apt to classify the pattern correctly. (b) The corresponding scenario for Directed Drift. Here the algorithm is constrained to make only single bit changes in the current (binary) weight vector. Bit changes in the positive hemisphere (in the direction of the misclassified pattern) improve the possibility of correct classification.

**Fig. 2** Plots of the expected mistake bound (averaged over 1000 trials) of the Directed Drift Algorithm as a function of $n$ for four cases with the number of patterns, $m$, chosen equal to $n/4$, $n/2$, $n$, and $2n$. The first two cases are within capacity, and the curves reflect best fit quadratics. For $m = n$ and $m = 2n$ the number of patterns exceed capacity, and the curves fitted are polynomials of degree 4 and 5, respectively. The best fit exponential curve, $2^{\alpha n}$, above capacity has an exponent of roughly $0.75n$ in the range considered.

**Fig. 3** The expected mistake bound (averaged over 1000 independent trials) of Directed Drift is plotted against the number of patterns for $n = 20$. A threshold phenomenon is observed around capacity when the mistake bound rises abruptly. The mistake bound saturates to a fixed value when the number of patterns exceeds approximately $3n$.

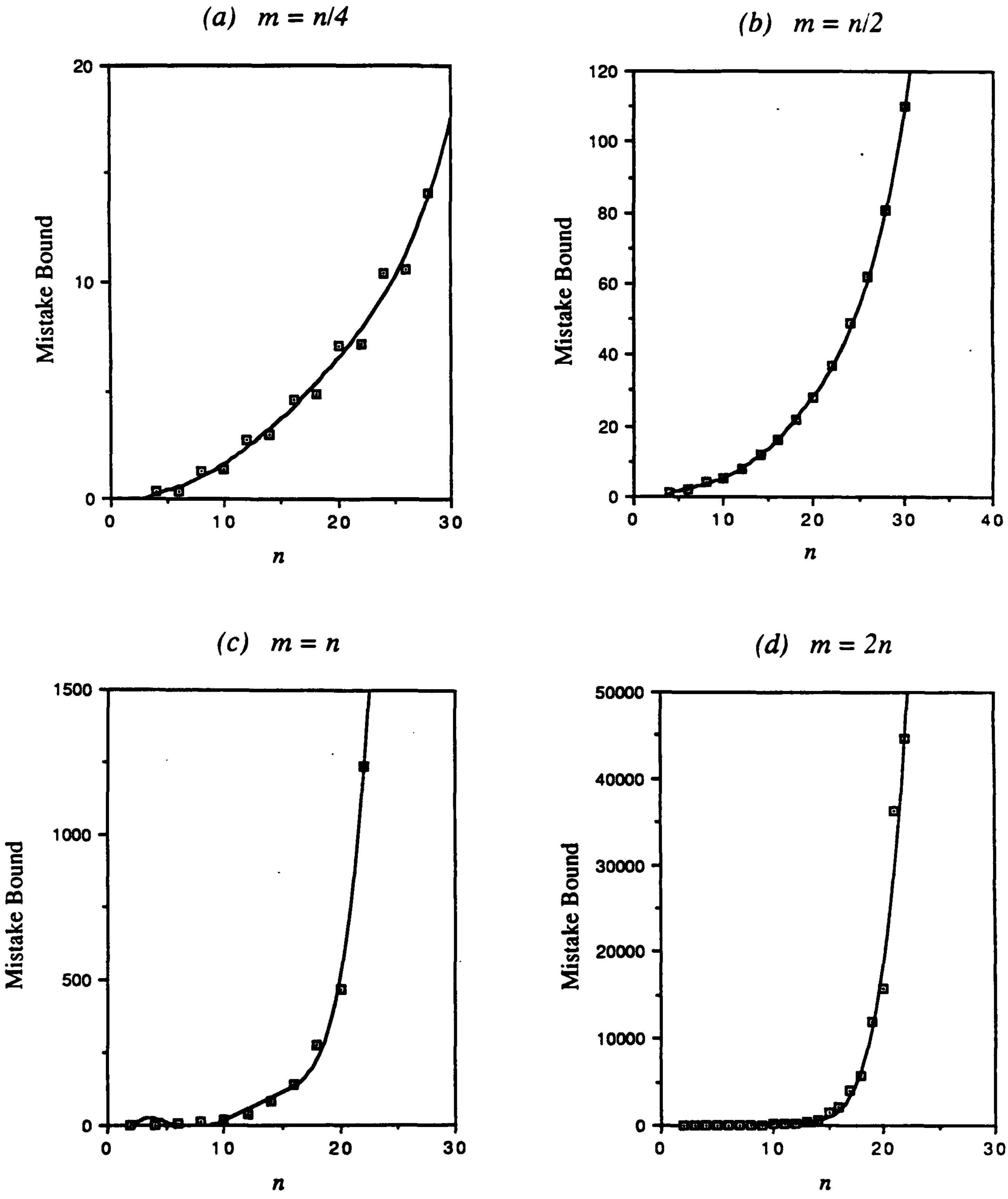
**Fig. 4** The expected mistake bound and the probability that Directed Drift terminates in a different solution vector than the one specified (at random) initially are plotted as a function of the number of patterns for $n = 10$. (Results are averaged over 100 independent trials.) Note the same threshold behaviour around capacity and the saturation phenomenon for the mistake bound as observed in Figure 3. The probability that there is more than one binary solution is a dual of the curve for the mistake bound: the probability of many binary solution vectors is high when the number of patterns is small, and plunges abruptly around capacity to essentially zero around $3n$. The saturation of the mistake bound around $3n$ patterns thus seems to be a consequence of the reduction in the binary solution space till there is only a unique binary solution vector around $3n$ patterns. Specifying more examplar patterns then does not yield any further information on the solution vector.

**Fig. 5** The expected mistake bound and the length of the solution vector (normalised by $\sqrt{n} = \sqrt{10}$) at convergence is plotted against the number of patterns for $n = 10$ for fixed increment Perceptron Training. (Results were averaged over a 100 independent trials.) The averaged mistake bound saturates around $6n$ patterns, reflecting the larger capacity, while the normalised length of the solution vector returned by the algorithm saturates at a value somewhat larger (about a factor of 2.5) than the length of a binary solution vector.



(a)

(b)



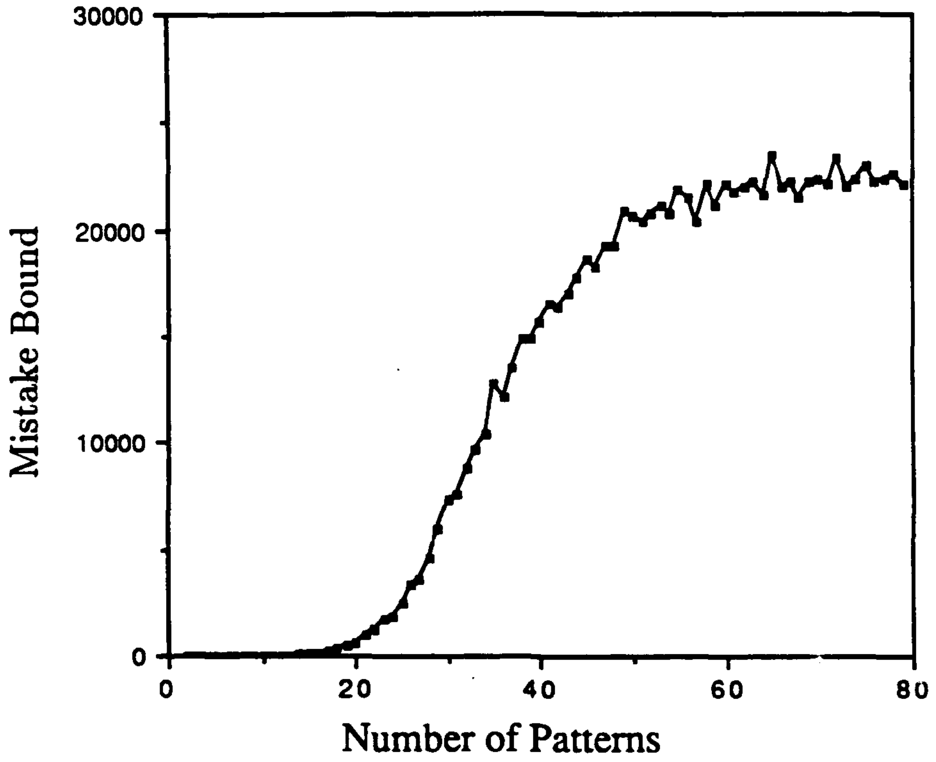
*(a) m = n/4*

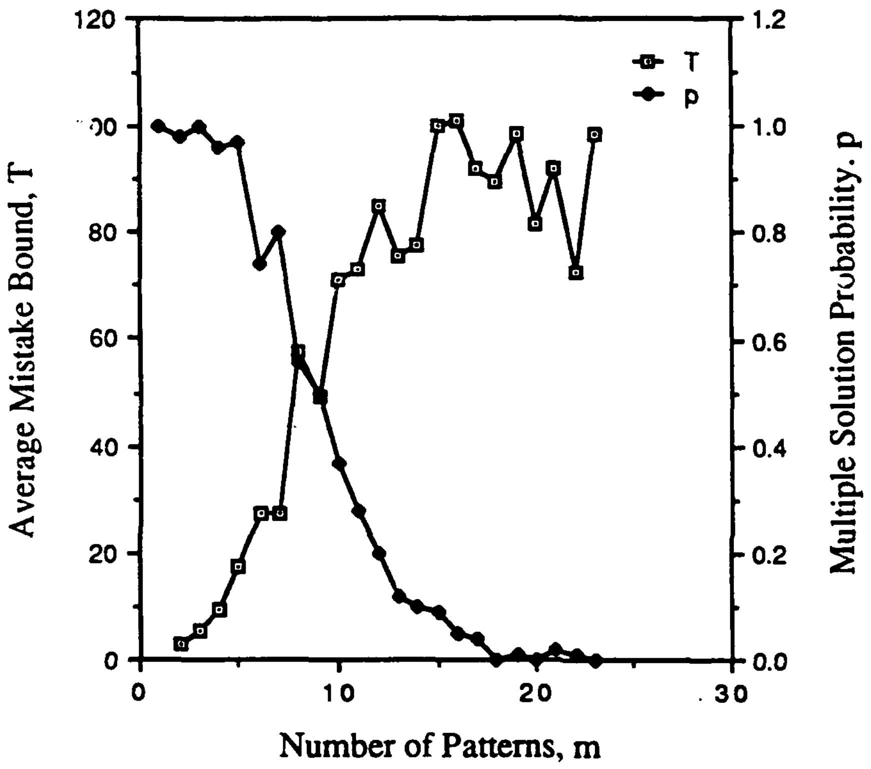
*(b) m = n/2*

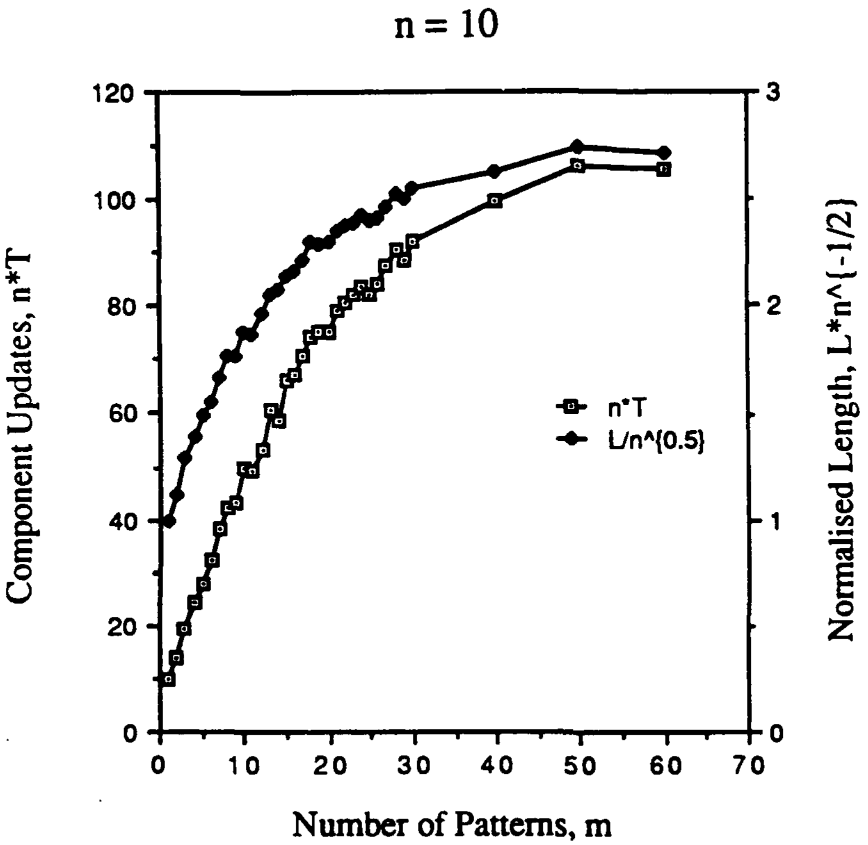
*(c) m = n*

*(d) m = 2n*

n = 20


30000
20000
10000
0
Mistake Bound
0
20
40
60
80
Number of Patterns


n = 10
120
100
80
60
40
20
0
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
T
p
Average Mistake Bound, T
Multiple Solution Probability. p
Number of Patterns, m


n = 10
Component Updates, n*T
Normalised Length, L*n^{-1/2}
Number of Patterns, m
n*T
L/n^{0.5}
0
20
40
60
80
100
120
0
1
2
3
0
10
20
30
40
50
60
70

To appear:
Proceedings of the 4th Workshop
on Ccmputational Learning Theory
(L.G. Valiant & M.K. Warmuth, eds)
Morgan Kaufmann, San Mateo, CA
1991

# On Learning Binary Weights for Majority Functions

**Santosh S. Venkatesh**
Department of Electrical Engineering
University of Pennsylvania
Phildelphia, PA 19104
*Email*: venkatesh@ee.upenn.edu

## Abstract

We investigate algorithms for learning *binary* weights from examples of majority functions of a set of literals. In particular, given a set of (randomly drawn) input-output pairs, with inputs being binary ±1 vectors, and the outputs likewise being ±1 classifications, we seek to find a vector of *binary* (±1) weights for a linear threshold element (or formal neuron) which provides a linearly separable hypothesis consistent on the set of examples. We present three algorithms—Directed Drift, Harmonic Update, and Majority Rule—for learning binary weights in this context, and examine their characteristics. In particular, we formally define a distribution dependent notion of algorithmic *capacity* (which is related to the distibution free notion of the VC dimension) and provide estimates of the capacity of the proposed algorithms.

## 1 INTRODUCTION

Recent results have indicated that large dynamic ranges may not be needed for the weights in neural networks [VF91]. In particular, for many applications, binary weights may suffice for the weights; alternatively, a network with real interconnection weights can be replaced by an equivalent network of binary weights realising the same Boolean function with a slight increase in the size of the network. Concomittant with the birth of a theory validating the computational capabilities of networks with binary (or limited dynamic range) weights, there has been a development of a capability to produce large hardware implementations of such networks [GH90].

A question of some practical import is whether there are algorithms which can succesfully exploit the latent information storage capabilities of these networks by *learning* binary weights for a given architecture from instances of the function to be represented. Unfortunately, theory may proscribe a general solution: the problem of learning binary weights is NP-complete. The issue is, however, open whether there are (randomised) learning algorithms which converge rapidly *on average*. As a first step in the consideration of this problem, we consider learning binary weights in the context of *linearly separable* functions; i.e., restrict consideration to learning binary weights for a single neuron.[1]

Given an arbitrary linearly separable dichotomy of a finite set of patterns, the Perceptron Training Algorithm [Ros62] guarantees convergence in *finite time* of an iteratively updated sequence of weight vectors to a *real* solution vector which separates the dichotomy. Perceptron Training is an on-line procedure which has good average-case convergence times, but which can occasionally exhibit a worst-case exponential time convergence. Worst-case polynomial running times can, however, be guaranteed for the problem with off-line procedures such as Karmarkar's algorithm for linear programming. Learning binary weights for a neuron is, however, equivalent to integer programming, which is known to be NP-complete [GJ79].[2]

We present three approaches to the problem of learning binary weights for linear threshold functions, the target functions being majority functions of a set of $n$ literals. In the first approach we present a randomised, local, and homogeneous on-line procedure—which we call *Harmonic Update* [Ven91(a)]—for learning binary weights from a single pass of a set of examples. The second algorithm we present is a homogeneous off-line procedure we call *Majority Rule* [Ven91(a)]. In the third approach we develop a family of randomised algorithms—dubbed *Directed Drift* [Ven91(b)]—which are on-line, local, and mistake driven.

A key parameter we estimate is the *capacity*, an algorithm and distribution dependent parameter linked to the VC dimension. The main results here are that Harmonic Update has a capacity of the order of $\sqrt{n}/\sqrt{\log n}$, Majority Rule has a capacity of the order of $n/\log n$, and Directed Drift has a capacity of order

[1] In this discussion, we shall use the term "neuron" synonymously with a linear threshold element.

[2] Some problems are born to NP-completeness, some attain NP-completeness, and other have NP-completeness thrust upon 'em.

between $n/\log n$ and $n$. Furthermore, these capacities are maximal among algorithms with the respective features of these three algorithms.

The Harmonic Update Algorithm, while on-line, is not mistake driven and terminates after a single pass of the set of examples. Mistake bounds or convergence times for the Directed Drift Algorithm are, however, much harder to obtain. A feeling for the problem can be obtained, however, by appealing to analogous situations in the theory of random walks and the geometric theory of paths. The corresponding problem here involves the estimation of the expected time of first passage of a random walk with positive drift to a fixed boundary at $n$. We obtain the estimate $O(n^2)$ for the expected time of first passage to the boundary and argue heuristically that this may hold as an estimate for the expected mistake bound for Directed Drift when the number of examples is within the capacity of the algorithm. In an appendix we also provide a comparison of the rate of convergence of Directed Drift with Perceptron Training: we show that the corresponding worst-case and average-case number of component updates for Perceptron Training is $O(n^3)$.

*On notation*: We will use the symbol $\mathbb{B}$ to denote the set $\{-1,1\}$. If $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$ are points in real Euclidean $n$-space, we denote by $\langle \mathbf{x}, \mathbf{y} \rangle$ the inner-product $\sum_{j=1}^{n} x_j y_j$. We use the word *epoch* to denote *points* on the time axis. A physical weight update may take some time, but we will assume updates are timeless and occur at epochs.[3] We define the function sgn : $\mathbb{R} \to \mathbb{B}$ by $\operatorname{sgn} x = x/|x|$ if $x \neq 0$ and $\operatorname{sgn} 0 = 1$. All logarithms in the exposition are to base $e$. If $\{x_n\}$ and $\{y_n\}$ are positive sequences, we denote: $x_n \lesssim y_n$ if $x_n \leq y_n$ for $n$ large enough; $x_n \gtrsim y_n$ if $x_n \geq y_n$ for $n$ large enough.

## 2 THE SETTING

We are given a set of *patterns*, $\mathcal{U} \subset \mathbb{B}^n$, and a function $f : \mathcal{U} \to \mathbb{B}$ which is linearly separable: specifically, there exists a (binary) *solution weight vector*, $\mathbf{w}^s \in \mathbb{B}^n$, such that

$$\operatorname{sgn} \langle \mathbf{w}^s, \mathbf{u} \rangle = f(\mathbf{u}) \qquad (1)$$

for every choice of pattern $\mathbf{u} \in \mathcal{U}$. We call the function $f$ the *target function*; these are, hence, majority functions of a set of literals. Given $\mathcal{U}$ and a linearly separable target function $f$, the goal is to efficiently find a (binary) solution weight vector $\mathbf{w} \in \mathbb{B}^n$. Note that $f$ dichotomises the set of patterns $\mathcal{U}$. Without loss of generality we assume that $f(\mathbf{u}) = 1$ for every pattern $\mathbf{u} \in \mathcal{U}$.[4]

### 2.1 ALGORITHMS

In an *off-line* binary learning algorithm, weights $w_i \in \mathbb{B}$, $i = 1, \dots, n$ are produced directly as a function of the set of patterns $\mathcal{U}$: more specifically, $w_i = g_i(\mathcal{U})$, where the functions $g_i : \mathcal{U} \to \mathbb{B}$ are specified by the algorithm. We say that: an off-line algorithm is *local* if $w_i$ is determined solely from the $i$th component of the patterns, $w_i = g_i(\{u_i : \mathbf{u} \in \mathcal{U}\})$, for each $i = 1, \dots, n$;[5] a local off-line algorithm is ***homogeneous*** if there is a function $g$ such that $w_i = g(\{u_i : \mathbf{u} \in \mathcal{U}\})$, for each $i = 1, \dots, n$.

In contrast, an *on-line* algorithm for *learning from examples* is a procedure where learning takes place in a sequence of trials. The protocol is as follows:

1° At epoch $t$ the system is characterised by a weight vector, $\mathbf{w}[t] \in \mathbb{B}^n$, and receives an example pattern, $\mathbf{u}[t] \in \mathbb{B}^n$, drawn from $\mathcal{U}$.

2° The system produces a *response* $v[t] \in \mathbb{B}$ according to the sign of $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle$.

3° A new weight vector, $\mathbf{w}[t+1] \in \mathbb{B}^n$, is generated based on the current response $v[t] \in \mathbb{B}$, weight vector $\mathbf{w}[t] \in \mathbb{B}^n$, and example $\mathbf{u}[t] \in \mathbb{B}^n$.

The procedure is carried out iteratively, and is terminated *if a solution weight vector* is obtained. Note that we restrict ourselves to on-line algorithms which generate binary weight vectors, $\mathbf{w}[t] \in \mathbb{B}^n$, at *each* epoch of the learning process; specifically, the weights are confined to the domain $\{-1,1\}$ throughout learning. This situation may be compared to Perceptron Learning where the weights typically grow in maginitude during the learning process. We call the sequence of examples, $\{\mathbf{u}[t]\}_{t=1}^{\infty}$, the *training sequence*, and the sequence of weight vectors, $\{\mathbf{w}[t]\}_{t=1}^{\infty}$, the *learning sequence*. If the procedure terminates in a finite time, we say that the learning algorithm has *learnt* the function $f$. We will be interested in the ***mistake bound*** $T$—the number of classification mistakes the learning algorithm makes on the training sequence before it learns the given function. In particular, the mistake bound is equal to the number of epochs for which $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle$ is not positive. For our purposes, the mistake bound is equal to the actual number of updates of the weight vector before the function is learnt.

We say that an on-line learning algorithm is *local* if each weight, $w_i[t+1] \in \mathbb{B}$, is updated solely as a function of $w_i[t]$, $u_i[t]$, and $v[t]$, i.e., for each $i = 1$,

[3] In his text, W. Feller [Fel68, page 73] credits J. Riordan with initiating the usage of the word epoch in such situations.

[4] For the nonce, extend $f$ to the domain $\mathbb{B}^n$ using the relation (1). If $f(\mathbf{u}) = -1$ then $f(-\mathbf{u}) = 1$ as can be easily seen from (1). Replacing each pattern in $\mathcal{U}$ for which $f(\mathbf{u}) = -1$ by $-\mathbf{u}$ we obtain a corresponding set of patterns $\hat{\mathcal{U}}$; if $\mathbf{w}^s$ is any solution weight vector separating the dichotomy of $\mathcal{U}$ specified by $f$ then all patterns in $\hat{\mathcal{U}}$ lie on the same (positive) side of the hyperplane corresponding to $\mathbf{w}^s$, and conversely.

[5] We abuse notation somewhat here by retaining the same functional notation $g_i$ over different domains.

..., $n$, there is a sequence of (possibly probabilistic) functions $f_{i,t}$ such that

$$w_i[t+1] = f_{i,t}(w_i[t], u_i[t], v[t]), \qquad i = 1, \ldots, n.$$

In analogy with the corresponding situation for off-line algorithms, we say that a local on-line algorithm is *homogeneous* if all weights at a given epoch have the same update rule, i.e., there is a sequence of (possibly probabilistic) functions $f_t$ such that

$$w_i[t+1] = f_t(w_i[t], u_i[t], v[t]), \qquad i = 1, \ldots, n.$$

In addition, we say that an on-line algorithm is *single-pass* if each pattern $\mathbf{u} \in \mathcal{U}$ *occurs exactly once in the* training sequence.

## 2.2 CAPACITY

For positive integers $m$ and $n$, let $\mathcal{U}_n^m = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ be a random $m$-set of patterns chosen independently from $\mathbb{B}^n$; specifically, for each $\alpha = 1, \ldots, m$, the patterns $\mathbf{u}^\alpha = (u_1^\alpha, \ldots, u_n^\alpha)$ are chosen independently, with the components $u_i^\alpha$ of each pattern drawn from a sequence of symmetric Bernoulli trials

$$\mathbf{P}\{u_i^\alpha = +1\} = \mathbf{P}\{u_i^\alpha = -1\} = 1/2.$$

Let $\mathcal{A}$ be an algorithm for learning binary weights, and let $P_{\mathcal{A}}(n,m)$ be the probability that $\mathcal{A}$ produces a solution weight vector for the $m$-set of patterns $\mathcal{U}_n^m$, i.e., $P_{\mathcal{A}}(n,m)$ is the probability that, given the $m$-set of patterns $\mathcal{U}_n^m$, the algorithm $\mathcal{A}$ yields a weight vector $\mathbf{w}_{\mathcal{A}}(\mathcal{U}_n^m) \in \mathbb{B}^n$ such that $\langle \mathbf{w}_{\mathcal{A}}(\mathcal{U}_n^m), \mathbf{u} \rangle > 0$ for every $\mathbf{u} \in \mathcal{U}_n^m$.

**Definition 2.1** We say that a sequence $C_n$ is a *capacity function* (or simply, *capacity*) for $\mathcal{A}$ if, for any choice of $0 < \lambda < 1$, the following two properties hold:

a) $P_{\mathcal{A}}(n, m_n) \to 1$ as $n \to \infty$ for every sequence $\{m_n\}$ which is such that $m_n \lesssim (1-\lambda)C_n$;

b) $P_{\mathcal{A}}(n, m_n) \to 0$ as $n \to \infty$ for every sequence $\{m_n\}$ which is such that $m_n \gtrsim (1+\lambda)C_n$.

We also say that $C_n$ is a *lower capacity* if property (a) holds, and that $C_n$ is an *upper capacity* if property (b) holds.

This notion of the capacity function has a counterpart in the theory of random graphs in the notion of a *threshold function of an attribute.* Loosely speaking, the capacity function quantifies the capability of the algorithm under consideration by specifying the size of the "largest typical set" of patterns for which "most" dichotomies are separated by the algorithm with high probability. Note that the capacity function depends implicitly upon the choice of distribution for the patterns. We could allow other distributions, or more generally, a family of distributions.

Lower and upper capacities in a natural sense provide lower and upper bounds on the capability of an algorithm. Note that while lower and upper capacities are guaranteed to exist, the capacity function itself may not (though it frequently does). Capacity requires a sharp threshold characterisitic in the computational capability of the structure and the algorithm. Capacity functions are not unique, even when they exist. The following result (an easy consequence of the definition) shows, however, that if capacity functions exist, then they are not very different from each other asymptotically.

**Proposition 2.2** *If $C_n$ is a capacity function then so is $C_n(1 \pm o(1))$. Conversely, if $C_n$ and $C_n'$ are any two capacity functions, then $C_n \sim C_n'$ as $n \to \infty$.*

The capacity definition can be seen to be a sort of distribution dependent analogue of the VC dimension. For learning in a distribution free setting, Blumer, *et al* [BEHW89] invoke the sufficient conditions for uniform convergence of relative frequencies of events to their probabilities derived in the seminal paper of Vapnik and Cĕrvonenkis [VC71] to show that the sample complexity for learning is proportional to the VC dimension of the hypothesis class under consideration. A similar argument utilising the necessary and sufficient conditions derived in the Vapnik-Cĕrvonenkis paper can be adduced to show that the sample complexity for learning in a distribution dependent setting should be proportional to the (probabilistic) capacity.[6]

A natural question then is how this (distribution dependent) notion of capacity is linked to the VC dimension. We are dealing here with a sequence of hypothesis classes $H_n$—the family of half-spaces corresponding to binary weight vectors from $\mathbb{B}^n$—with corresponding VC dimensions $d_n \le n$. Using the fact that the number of dichotomies of an $m$-set of patterns induced by the hypothesis class $H_n$ is majorised by $m^{d_n} + 1$, it is easy to show the following general result.

**Theorem 2.3** *Any lower capacity function $\underline{C_n}$ satisfies $\underline{C_n} = O(d_n \log^* d_n)$ as $n \to \infty$.*

The above result holds for all choices of algorithm and distribution. (In fact, the VC dimension can be thought of as a special case of the capacity when the algorithm allows an exhaustive search of the hypothesis class, and all distributions are allowed.) The capacity can, hence, never exceed the VC dimension by very much, whatever be the choice of algorithm and distribution family. It is possible, however, to have capacities substantially smaller than the VC dimension [Ven91(c)]. Distribution families for which this is true will then demand much smaller sample complexities than the distribution free case.

[6] Some slight additions have to be made to the definition, but these are not critical in a network setting.

The capacity definition above can be readily extended to more general situations where we have an arbitrary sequence of computational structures (hypothesis classes $H_n$), distribution families, algorithms, and computational attributes more complex than correct classification (such as associative memory with error tolerance) [Ven91(c), Ven91(d)]. (For examples of capacity calculations within this framework in a neural network setting see [KP88(a), KP88(b), MPRV87, N88, Ven91(e), VB91(a), VB91(b), VP91], for instance.) The basic properties derived above (Proposition 2.2 and Theorem 2.3) continue to hold in the general case.

## 3 HARMONIC UPDATE

The first algorithm we introduce, dubbed Harmonic Update, is a single-pass, homogeneous, on-line randomised algorithm for learning binary weights [Ven91(a)]. As before, let $\mathcal{U}_n^m = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ be a random $m$-set[7] of patterns in $\mathbb{B}^n$ drawn independently, and with components drawn from a sequence of symmetric Bernoulli trials. The training sequence consists of the patterns, $\mathbf{u}^1$, ..., $\mathbf{u}^m$, presented in turn. Let the initial choices of the weights, $w_i[1] \in \mathbb{B}$, be arbitrary, and let $\mathbf{w} \equiv \mathbf{w}[m+1]$ be the final weight vector returned by the algorithm. Harmonic Update is a randomised algorithm which prescribes weight updates as follows.

*For $i = 1, \ldots, n$, and epochs $t = 1, \ldots, m$:*

- *If $w_i[t] = u_i^t$, then set $w_i[t+1] = w_i[t]$.*
- *If $w_i[t] = -u_i^t$, then set $w_i[t+1] = -w_i[t]$ with probability $1/t$, and $w_i[t+1] = w_i[t]$ with probability $1 - 1/t$.*

Clearly, Harmonic Update[8] is a randomised on-line algorithm, and, as claimed, it is homogeneous and single-pass. The algorithm is not mistake driven, and as each example pattern is seen exactly once, the algorithm terminates after the minimal number of steps, $m$. The effect of this randomised procedure is to ensure that each weight retains an equal amount of information about the corresponding component of every pattern. In particular,

$$\mathbf{E}\, w_i u_i^\alpha = \frac{1}{m}, \qquad \alpha = 1, \ldots, m, \quad i = 1, \ldots, n, \quad (2)$$

as can be readily seen by induction. The following estimate now holds:

**Theorem 3.1** *The sequence $\sqrt{n}/\sqrt{\log n}$ is a capacity function for the Harmonic Update Algorithm. Moreover, no other homogeneous, single-pass algorithm has a capacity function with a more rapid rate of growth.*

REMARKS: An application of the second moment method shows that $\sqrt{n}/\sqrt{\log n}$ is a lower capacity, and for brevity, we will restrict ourselves to proving this here. To prove that $\sqrt{n}/\sqrt{\log n}$ is also an upper capacity for Harmonic Update calls for some delicate footwork with weakly dependent random variables. The main ideas involved are the observation that the random variables $Y_j^\alpha$ defined below are exchangeable, together with a large deviation "Poissonisation" argument which shows that the errors are Poisson distributed asymptotically. The proof that this capacity is maximal for homogeneous, single-pass algorithms utilises some maximin inequalities proved in [VF91]. Details and the complete proof are given in [Ven91(a)].

PROOF: [Sketch.] Define the random variables

$$Y_j^\alpha = w_j u_j^\alpha, \qquad j = 1, \ldots, n, \quad \alpha = 1, \ldots, m.$$

Note that by the locality of the Harmonic Update Algorithm, for each $j = 1, \ldots, n$, the weight $w_j$ depends solely on the $j$th components $u_j^1$, ..., $u_j^m$ of the patterns. By independence of the pattern components, and by symmetry, it follows that the weights $w_1$, ..., $w_n$ are i.i.d., symmetric Bernoulli random variables taking values -1 and 1 only, each with probability 1/2. It hence follows that for every fixed $\alpha$, the random variables $Y_1^\alpha$, ..., $Y_n^\alpha$ are independent, $\pm 1$ random variables, and as Harmonic Update is homogeneous, they are identically distributed as well. An inductive argument similar (and only slightly more detailed) than the one used to establish (2) yields

$$\begin{aligned} \mathbf{P}\{Y_j^\alpha = -1\} &= \frac{1}{2} - \frac{1}{2m} = q \\ \mathbf{P}\{Y_j^\alpha = +1\} &= \frac{1}{2} + \frac{1}{2m} = p. \end{aligned}$$

Now form the random sums

$$X^\alpha = \sum_{j=1}^n Y_j^\alpha, \qquad \alpha = 1, \ldots, m.$$

If $\mathbf{w}$ is to be a solution vector, we require that each $X^\alpha$ be positive. Let us estimate instead the probability that a particular pattern, say $\mathbf{u}^\alpha$, is not correctly classified by $\mathbf{w}$. The following exponential inequality due to Hoeffding now proves useful.

**Lemma 3.2** *[Hoeffding] Let $Z_1$, ..., $Z_n$ be independent random variables with zero means and bounded*

[7] For capacity calculations we seek $m$, the number of patterns, as an explicit function of $n$. To keep the notation simple, however, we write $m$ instead of $m_n$, which would make the dependence explicit, with the tacit understanding that the number of patterns is actually a function of $n$.

[8] The name arises from the choice of the sequence of probabilities $\{1/t, t \geq 1\}$ in the algorithm: at epoch $t$, $1/t$ is the probability that a weight update results in a change in sign of the weight when the current weight and the corresponding component of the pattern from the training sequence disagree in sign.

*ranges:* $a_j \leq Z_j \leq b_j$. *Then for every* $\eta > 0$,

$$\mathbf{P}\left\{\sum_{j=1}^{n} Z_j \leq -\eta\right\} \leq \exp\left[\frac{-2\eta^2}{\sum_{j=1}^{n}(b_j - a_j)^2}\right].$$

As $X^\alpha$ is the sum of $n$ i.i.d. ±1 random variables with mean $p-q = 1/m$, Hoeffding's inequality directly yields[9]

$$\mathbf{P}\{X^\alpha \leq 0\} \leq \exp\left(-\frac{n}{2m^2}\right).$$

The probability $P_{HU}(n,m)$ that Harmonic Update correctly classifies each of the patterns in $\mathcal{U}_n^m$ is bounded below by a simple application of Boole's inequality:

$$P_{HU}(n,m) \geq 1 - m\,\mathbf{P}\{X^\alpha \leq 0\} \geq 1 - m\exp\left(-\frac{n}{2m^2}\right).$$

Now, for any $\epsilon > 0$, the choice

$$m = \frac{\sqrt{n}}{\sqrt{\log n}}\left[1 + \frac{\log\log n + 2\log\epsilon}{2\log n} - O_\epsilon\left(\frac{\log\log n}{(\log n)^2}\right)\right]$$

yields $P_{HU}(n,m) \gtrsim 1-\epsilon$ as $n \to \infty$. It follows that $\sqrt{n}/\sqrt{\log n}$ is a lower capacity function for Harmonic Update. ▮

# 4 MAJORITY RULE

If off-line procedures are permitted, substantial gains in capacity can be made. The Majority Rule described below is a homogeneous, off-line algorithm with near maximal capacity.

As before, let $\mathcal{U}_n^m = \{\mathbf{u}^1, \dots, \mathbf{u}^m\}$ be a random $m$-set of patterns in $\mathbb{B}^n$, with components chosen from a sequence of symmetric Bernoulli trials. The Majority Rule prescribes weights as follows:

*For* $i = 1, \dots, n$, *let*

$$\begin{aligned} U_i^+ &= \{\mathbf{u}^\iota \in \mathcal{U} : u_i^\iota = +1\} \\ U_i^- &= \{\mathbf{u}^\iota \in \mathcal{U} : u_i^\iota = -1\}. \end{aligned}$$

*Set*

$$w_i = \begin{cases} +1 & \text{if } |U_i^+| \geq |U_i^-| \\ -1 & \text{if } |U_i^+| < |U_i^-|. \end{cases}$$

In other words, $w_i = +1$ if patterns whose $i$th component is $+1$ are in the majority, and $w_i = -1$ otherwise. Clearly, Majority Rule is an off-line algorithm which is local and homogeneous. The following estimate can be obtained:

**Theorem 4.1** *The sequence* $n/\pi \log n$ *is a capacity function for the Majority Rule Algorithm. Moreover, no other homogeneous, off-line algorithm has a capacity function with a more rapid rate of growth.*

REMARKS: Again, we will content ourselves here with providing a sketch of the proof that $n/\pi\log n$ is a lower capacity for Majority Rule. Details may be found in [Ven91(a)].

PROOF: [Sketch.] We begin by noting that with probability one we can write

$$w_j = \operatorname{sgn}\left(\sum_{\beta=1}^{m} u_j^\beta\right), \qquad j = 1, \dots, n.$$

As before, for $\alpha = 1, \dots, m$, form the sums

$$X^\alpha = \sum_{j=1}^{n} w_j u_j^\alpha = \sum_{j=1}^{n} Y_j^\alpha,$$

where the ±1 random variables $Y_j^\alpha$ are defined by

$$\begin{aligned} Y_j^\alpha &= w_j u_j^\alpha = \operatorname{sgn}\left(u_j^\alpha \sum_{\beta=1}^{m} u_j^\beta\right) \\ &= \operatorname{sgn}\left(1 + \sum_{\beta\neq\alpha} u_j^\alpha u_j^\beta\right). \qquad (3) \end{aligned}$$

Note that, as before, for fixed $\alpha$, $Y_1^\alpha, \dots, Y_n^\alpha$ are i.i.d., ±1 random variables. Now, the summands in the sum in (3) are i.i.d., symmetric Bernoulli, ±1 random variables, so that the sum is just a symmetric random walk over $m-1$ steps. Let $m$ grow without bound as $n \to \infty$. An application of Stirling's formula then yields

$$\begin{aligned} \mathbf{P}\{Y_j^\alpha = -1\} &\sim \frac{1}{2} - \frac{1}{\sqrt{2\pi m}} \\ \mathbf{P}\{Y_j^\alpha = +1\} &\sim \frac{1}{2} + \frac{1}{\sqrt{2\pi m}}. \end{aligned}$$

It follows that

$$\mathbf{E}\,Y_j^\alpha \sim \frac{\sqrt{2}}{\sqrt{\pi m}} \qquad (n \to \infty).$$

An application of Hoeffding's inequality as in Theorem 3.1 hence yields

$$\mathbf{P}\{X^\alpha \leq 0\} \leq \exp\left(-\frac{n}{\pi m}\right).$$

Using Boole's inequality, as before, yields that the probability $P_{MR}(n,m)$ that Majority Rule correctly classifies each of the patterns in $\mathcal{U}_n^m$ is bounded below by

$$P_{MR}(n,m) \geq 1 - m\,\mathbf{P}\{X^\alpha \leq 0\} \geq 1 - m\exp\left(-\frac{n}{\pi m}\right).$$

[9] A slightly more involved argument invoking the large deviation version of the classical De Moivre-Laplace central limit theorem (cf. Feller's text [Fel68], for instance) yields that if $m$ grows with $n$ such that $m = o(\sqrt{n})$ and $m/n^{1/6} \to \infty$, then $\mathbf{P}\{X^\alpha \leq 0\} \sim \frac{m}{\sqrt{2\pi n}}\exp\left(-\frac{n}{2m^2}\right)$ as $n \to \infty$. This more precise estimate is needed to show that $\sqrt{n}/\sqrt{\log n}$ is also an upper capacity for Harmonic Update.

Now, for any $\epsilon > 0$, the choice

$$m = \frac{n}{\pi \log n}\left[1 + \frac{\log\log n + \log \pi\epsilon}{\log n} - O_\epsilon\left(\frac{\log\log n}{(\log n)^2}\right)\right]$$

yields $P_{MR}(n,m) \gtrsim 1 - \epsilon$ as $n \to \infty$. It follows that $n/\pi \log n$ is a lower capacity function for Majority Rule. ∎

# 5 DIRECTED DRIFT

We conclude with a family of randomised, on-line, local (but non-homogeneous)[10] algorithms for binary learning [Ven91(b)]. We call these algorithms *Directed Drift* because, as we shall see, they share some similarities with asymmetric random walks with a preferred direction toward a solution.

Let $\mathcal{U}$ be any subset of patterns from $\mathbb{B}^n$, and let $\{\mathbf{u}[t]\}$ be any training sequence such that each of the patterns in $\mathcal{U}$ appears infinitely often.[11] Let $\{\mathbf{w}[t]\}$ denote a binary learning sequence. For each epoch, $t$, we denote by $J[t]$ the subset of indices for which the corresponding components of $\mathbf{w}[t]$ and $\mathbf{u}[t]$ are opposite in sign:

$$J[t] = \{\, j : w_j[t] \neq u_j[t] \,\}.$$

## 5.1 SINGLE BIT UPDATES

In the simplest version of Directed Drift, no more than à single component of the weight vector is updated per epoch.

> BASE: $\mathbf{w}[1] \in \mathbb{B}^n$ *is chosen arbitrarily.*
>
> ITERATION: *Weight updates are predicated upon whether a correct or incorrect response is obtained at the current epoch, t.*
>
> - *If* $\langle \mathbf{w}[t], \mathbf{u}[t]\rangle > 0$, *then the weight vector is left unchanged:* $\mathbf{w}[t+1] = \mathbf{w}[t]$.[12]
> - *If* $\langle \mathbf{w}[t], \mathbf{u}[t]\rangle \leq 0$, *then an index* $j[t]$ *is picked at random from the set of indices,* $J[t]$, *of mismatched components. The new weight vector is now formed according to the following rule:*
>
> $$w_j[t+1] = \begin{cases} w_j[t] & \text{if } j \neq j[t] \\ -w_j[t] & \text{if } j = j[t]. \end{cases} \qquad (4)$$

The intuition behind the algorithm is as follows. If a binary solution vector, $\mathbf{w}^s \in \mathbb{B}^n$, exists, then necessarily we must have $\langle \mathbf{w}^s, \mathbf{u}\rangle = \sum_{j=1}^n w_j^s u_j > 0$ for each pattern $\mathbf{u} \in \mathcal{U}$. As there is a contribution of +1 to the sum if two corresponding components of $\mathbf{w}^s$ and $\mathbf{u}$ have the same sign, and -1 if the signs are mismatched, it follows that the binary solution vector has more component sign matches than mismatches with *each* pattern in $\mathcal{U}$.

Now the algorithm updates the current estimate of the weight vector if and only if the current pattern from the training sequence is misclassified. A weight vector update results in a randomly chosen mismatched component of the weight vector being flipped to the sign of the corresponding pattern component. Since there is a probability better than a half that a randomly specified component of any pattern has the same sign as the corresponding component of a binary solution vector, it follows that at least during the initial progress of the algorithm, the *a priori* probability that the weight vector update is in the direction of the binary solution vector is better than a half. We will explore this more formally in the sequel.

## 5.2 ANALYSIS

Let $\mathbf{w}^s \in \mathbb{B}^n$ denote a solution vector, and let $\{t_k\}$ denote the subsequence of epochs at which patterns from the training sequence are misclassified; i.e.,

$$\langle \mathbf{w}[t_k], \mathbf{u}[t_k]\rangle \leq 0, \qquad k = 1, 2, \ldots .$$

Let

$$\mathcal{E}_{k+1} = \|\mathbf{w}[k+1] - \mathbf{w}^s\|^2$$

denote the estimate error at epoch $k$ for single bit update Directed Drift. A straightforward inductive argument then gives

$$\mathcal{E}_{k+1} = \mathcal{E}_1 - 4\sum_{i=1}^{k} X_i,$$

where we define the ±1 random variables $X_i$ by

$$X_i = w^s_{j[t_i]}\, u_{j[t_i]}[t_i].$$

Upper bounding $\mathcal{E}_1$ by $4n$ and setting $S_k = \sum_{i=1}^k X_i$ we then obtain

$$0 \leq \mathcal{E}_{k+1} \leq 4(n - S_k).$$

The procedure terminates at the value of $k$ for which the random sum $S_k$ first exceeds $n$. The mistake bound $T$ hence satisfies $S_T \geq n$, and $S_k < n$ for $k = 1, \ldots, T-1$. The mistake bound is infinite if there exists no such value of $k$, or if there exists no binary solution vector for the choice of patterns $\mathcal{U}$.

The above is reminiscent of a random walk with a fixed boundary at $n$. Let the $m$-set of patterns $\mathcal{U}$ be chosen independently, with components drawn from a sequence of symmetric Bernoulli trials, and assume $m$ is within the capacity of the algorithm. (With high probability, then, there exists a solution vector.) Through

[10] The algorithms actually have a slightly non-local preamble at each epoch. We will ignore this non-locality and continue to call the algorithms local.

[11] Note that $\mathcal{U} \subset \mathbb{B}^n$ is a finite set of patterns. If $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ is an $m$-set of patterns, then we can, for instance, obtain valid training sequences by cycling through the patterns or by choosing a pattern randomly at each epoch.

[12] If it ain't broke, don't fix it.

the initial progress of the algorithm then, the random variables $X_i$ will be independent with

$$X_i = \begin{cases} -1 & \text{with probability } q_{n,m} = 1/2 - \beta_{n,m} \\ 1 & \text{with probability } p_{n,m} = 1/2 + \beta_{n,m}, \end{cases}$$

for some $\beta_{n,m} \in (0, 1/2]$. The specific value of the probability $p_{n,m}$—the relative frequency of the number of component matches between a binary solution vector and a pattern—depends explictly both on $n$ and the number of patterns $m$. Clearly,

$$p_{n,m} \geq \frac{n/2+1}{n} = \frac{1}{2} + \frac{1}{n},$$

so that $\beta_{n,m} \geq 1/n$. A heuristic estimate of the expected mistake bound for single bit update Directed Drift when $m$ is within capacity can hence be obtained from the expected time of first passage to $n$ of a one-dimensional random walk $S_k$, with positive drift $2k\beta_{n,m}$. The following result is an application of Wald's equation:

**Lemma 5.1** *Let $\{X_j\}$ be i.i.d., $\pm 1$ random variables with mean $\mathbf{E}\,X_1 = 2\beta_{n,m} \geq 2/n$, and let $S_k = \sum_{j=1}^{k} X_j$ denote a random walk with positive drift $\mathbf{E}\,S_k = 2k\beta_{n,m} \geq 2k/n$. Let*

$$T_1 = \inf\{k : S_k = n\}$$

*denote the time of first passage to the fixed boundary at $n$. Then*

$$\mathbf{E}\,T_1 = \frac{n}{2\beta_{n,m}}$$

*for every $n$.*

Proof: Let $\mathcal{F}_k = \sigma(X_1, \dots, X_k)$, so that $\{\mathcal{F}_k, k \geq 1\}$ is an increasing sequence of sub-$\sigma$-algebras. $T_1$ is clearly a stopping time with respect to $\{\mathcal{F}_k\}$. Now, as $j \to \infty$, we have

$$\begin{aligned} \mathbf{P}\{T_1 \geq j\} &= \mathbf{P}\{S_1 < n, \dots, S_{j-1} < n\} \\ &\leq \mathbf{P}\{S_{j-1} < n\} = O(e^{-cj}) \end{aligned}$$

for a positive constant $c$, as, for fixed $n$, the penultimate expression is the probability in the extreme left tail of the binomial distribution. Hence,

$$\mathbf{E}\,T_1 = \sum_{j=1}^{\infty} \mathbf{P}\{T_1 \geq j\} < \infty$$

as the terms of the series decrease exponentially fast. For any measurable set $A$, let $I_A$ be the indicator random variable for $A$. We now have

$$\begin{aligned} \mathbf{E}\,|S_{T_1}| &\leq \mathbf{E} \sum_{j=1}^{\infty} |X_j|\, I_{\{T_1 \geq j\}} \\ &= \sum_{j=1}^{\infty} \mathbf{P}\{T_1 \geq j\} < \infty. \end{aligned}$$

Now, the random variables $X_j$ and $I_{\{T_1 \geq j\}}$ are independent for every $j \geq 1$, so that by the dominated convergence theorem and the definition of the stopping time $T_1$ we have

$$\begin{aligned} n &= \mathbf{E}\,S_{T_1} = \mathbf{E} \sum_{j=1}^{\infty} X_j\, I_{\{T_1 \geq j\}} \\ &= \sum_{j=1}^{\infty} \mathbf{E}(X_j)\mathbf{P}\{T_1 \geq j\} = \mathbf{E}(X_1)\mathbf{E}(T_1). \end{aligned}$$

The result follows. ∎

Using $\mathbf{E}\,T_1$ as a rough estimate for the expected mistake bound for single bit update Directed Drift results in the estimate $O(n^2)$. Simulations confirm this rapid convergence of Directed Drift when the number of examples is within the capacity of the algorithm.

There are a number of open issues about the algorithm which we are currently in the process of resolving. A rigourous and general analysis of stopping times for the algorithm involves careful consideration of the matrix of transition probabilities of a finite Markov chain, the transition probabilities depending both on $n$ and $m$. Capacity estimates for the algorithm are currently between the orders of $n/\log n$ and $n$. The upper capacity estimate of $n$ is an immediate consequence of Boole's inequality: the probability that there exists a binary solution vector for $m$ randomly drawn patterns is less than $2^{n-m}$, and if $m$ exceeds $n$, this probability plunges below 1/2. The lower capacity estimate of the order of $n/\log n$ follows from the estimate of the capacity of the Majority Rule algorithm: by construction of the Majority Rule, if $m$ is less than the order of $n/\log n$ then there exists a binary solution vector with high probability. An analysis of the transition probability matrix for Directed Drift indicates that the probability that the system stays forever among the (finite) set of transient states is zero when there exists at least one solution vector (which constitutes an absorbing state), so that the lower capacity is at least of the order of $n/\log n$. We conjecture that, in fact, the capacity of the algorithm is $n$.

Simulations indicate that there are two distinct regimes of behaviour—a regime below capacity where convergence is very rapid (in quadratic time) in consonance with the rough analysis above, and a regime above capacity where the analytical picture is much less clear and where convergence takes substantially longer. An abrupt transition around the capacity of the algorithmis is seen between the two regimes of convergence time. R. Meir has recently communicated to us that in Monte Carlo simulations and comparisons with genetic algorithms, Directed Drift appears to have an optimal character [Mei91]. A slightly more detailed analysis and specifics of simulation results are included in [Ven91(b)].

### 5.3 SEVERAL BIT UPDATES

The algorithm can be simply extended to accommodate more than a single bit update per epoch. Let $\{N_t\}$ be a sequence of integers with $0 \leq N_t \leq n/2$.

BASE: $\mathbf{w}[1] \in \mathbb{B}^n$ *is chosen arbitrarily.*

ITERATION: *As before, updates are made only if the current pattern from the training sequence is misclassified.*

- *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle > 0$, *then the weight vector is left unchanged:* $\mathbf{w}[t+1] = \mathbf{w}[t]$.
- *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle \leq 0$, *then* $N_t$ *indices* $j_1[t], \ldots, j_{N_t}[t]$ *are picked at random from the set of indices,* $J[t]$, *of mismatched components. The new weight vector is now formed according to the following rule: if* $j \notin \{j_1[t], \ldots, j_{N_t}[t]\}$ *then set* $w_j[t+1] = w_j[t]$; *else if* $j \in \{j_1[t], \ldots, j_{N_t}[t]\}$ *then set* $w_j[t+1] = -w_j[t]$.

The sequence $N_t$ specifies the number of bits to be changed at each update epoch, and the proper choice of this sequence is clearly critical to the functioning of the algorithm. This is analogous to choosing an appropriate *cooling schedule* for *simulated annealing* [KGV83]. Anecdotal evidence from simulations indicates that significant improvements in convergence can be obtained over single bit updates by appropriate choices of the sequence $N_t$.

## 6 CONCLUSIONS

The investigations reported here constitute initial forays into two areas: (1) using randomisation as a tool in the development of efficient learning algorithms for networks with binary weights (or, more generally, dynamic range constrained weights); and (2) developing notions of probabilistic capacity which, in distribution dependent situations, yield results on sample complexities for learning analogous to the distribution free results that derive from the VC dimension. Notwithstanding the theoretical stumbling blocks in learning binary weights—intractable worst cases may exist as a consequence of the NP-completeness of the problem—there is a strong practical motivation to develop learning algorithms for this case because of the lower cost and simplicity of circuits comprised of binary interconnections. The success (albeit limited) of the randomised algorithms reported here suggest that these may repay further investigation; in particular, we might be able to hope for good average case behaviour in certain regimes. We are currently investigating certain extensions of these ideas in networks with more complex interconnectivity patterns than the single neuron considered here. The parallel development of notions of distribution dependent capacity presented here in brief is aimed at providing a better understanding of (distribution dependent) problems where practitioners report a wide gulf between the sample complexities needed in practice and those predicted in the distribution free model.

## A PERCEPTRON TRAINING

It is instructive to compare convergence rates for Directed Drift with those that obtain for Perceptron Training. Let $\{\mathbf{u}[t]\}$ be a training sequence of patterns, and let $\{\mathbf{w}[t]\}$ denote a learning sequence of *real* weight vectors. We will assume that there exists a binary solution vector $\mathbf{w}^* \in \mathbb{B}^n$. (Perceptron Training will, in general, not converge to a binary solution, however, even if one exists.)

### A.1 FIXED INCREMENT PERCEPTRON TRAINING

This is the simplest form of Perceptron Training. Let $\beta > 0$ be fixed.

BASE: *The initial choice of weight vector is arbitrary. For simplicity we take* $\mathbf{w}[1] = 0$.

ITERATION: *As before, weight vector updates are made only if a pattern is misclassified.*

- *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle > 0$, *then the weight vector is left unchanged:* $\mathbf{w}[t+1] = \mathbf{w}[t]$.
- *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle \leq 0$, *then set* $\mathbf{w}[t+1] = \mathbf{w}[t] + \beta \mathbf{u}[t]$.

Note that fixed increment Perceptron Training is homogeneous and on-line.

We now claim that the procedure will converge to an, in general, non-binary solution with a worst-case mistake bound of $n^2$ if there exists a binary solution vector. Let $\mathbf{w}^* \in \mathbb{B}^n$ be a binary solution vector, and as before, let $\{t_k\}$ denote the subsequence of epochs at which patterns from the training sequence are misclassified; i.e.,

$$\langle \mathbf{w}[t_k], \mathbf{u}[t_k] \rangle \leq 0, \qquad k = 1, 2, \ldots. \tag{5}$$

Set $\mathbf{w}[1] \equiv 0$, and, for a value of parameter $c > 0$ to be specified, consider the estimate errors

$$\mathcal{F}_{k+1} \triangleq \|\mathbf{w}[t_{k+1}] - c\mathbf{w}^*\|^2.$$

Using (5), a standard inductive argument then yields the bounds

$$0 \leq \mathcal{F}_{k+1} \leq c^2 n - \beta(2c - \beta n)k.$$

We hence have the worst-case mistake bound

$$T \leq \frac{c^2 n}{\beta(2c - \beta n)}.$$

Minimising the bound with respect to $c$ yields the $n^2$ upper bound for the mistake bound. (For better mistake bounds see [Lit88].) The fixed increment Perceptron Training Algorithm hence terminates after $O(n^3)$ component updates if a binary solution vector exists.

## A.2 SINGLE COMPONENT PERCEPTRON TRAINING

The basic randomisation idea behind single bit update Directed Drift is easily extended to single component Perceptron Training, where a single component of the weight vector is modified at each update epoch (as opposed to fixed increment Perceptron Training where all components are modified at each epoch).

For each epoch, $t$, let $I[t]$ denote the subset of indices for which the corresponding components of $\mathbf{w}[t]$ and $\mathbf{u}[t]$ are opposite in sign:

$$I[t] = \{\, i : u_i[t] \neq \operatorname{sgn} w_i[t] \,\}.$$

> Base: *For simplicity, take* $\mathbf{w}[1] = 0$.
>
> Iteration: *Weight updates are predicated, as usual, upon whether a correct or incorrect response is obtained at the current epoch, $t$.*
>
> - *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle > 0$, *then the weight vector is left unchanged:* $\mathbf{w}[t+1] = \mathbf{w}[t]$.
> - *If* $\langle \mathbf{w}[t], \mathbf{u}[t] \rangle \leq 0$, *then an index $i[t]$ is picked at random from the set of indices, $I[t]$, of mismatched components. The new weight vector is now formed according to the following rule:*
>
> $$w_i[t+1] = \begin{cases} w_i[t] & \text{if } i \neq i[t] \\ w_i[t] + u_i[t] & \text{if } i = i[t]. \end{cases} \qquad (6)$$

Just as for Directed Drift, single component Perceptron Training is local, non-homogeneous, randomised, and on-line.

As before, we can heuristically estimate the average-case performance of the algorithm by appealing to ideas from random walks. Let $\{t_k\}$ satisfying (5) denote the subsequence of epochs at which patterns from the training sequence are misclassified. Let $\mathcal{L}_{k+1} = \|\mathbf{w}[t_{k+1}]\|$ denote the length of the weight vector at epoch $t_{k+1}$. By definition of the algorithm we inductively obtain the upper bound

$$\mathcal{L}_{k+1}^2 \leq \mathcal{L}_k^2 + 1 \leq k,$$

while the Cauchy-Schwarz inequality yields the lower bound

$$\mathcal{L}_{k+1}^2 \geq \frac{|\langle \mathbf{w}[t_{k+1}], \mathbf{w}^s \rangle|^2}{\|\mathbf{w}^s\|^2} = \frac{1}{n} \left| \sum_{h=1}^{k} w^s_{i[t_h]} u_{i[t_h]}[t_h] \right|^2 .$$

Define the ±1 random variables

$$X_h = w^s_{i[t_h]} u_{i[t_h]}[t_h],$$

and set $S_k = \sum_{h=1}^k X_h$. We then have the bounds

$$\frac{|S_k|}{\sqrt{n}} \leq \mathcal{L}_{k+1} \leq \sqrt{k}.$$

The algorithm terminates at the first instant $k$ for which $|S_k|$ exceeds $\sqrt{kn}$. Within capacity again, the situation is reminiscent of a random walk $S_k$ with positive drift $\mathbf{E}\, S_k = 2k\beta_n \geq 2k/n$, and absorbing boundaries at $\pm\sqrt{kn}$. We refer the reader to [Ven91(b)] for the proof of the following result:

**Lemma A.1** *Let $\{X_j\}$ be i.i.d., ±1 random variables with mean $\mathbf{E}\, X_1 = 2\beta_{n,m} \geq 2/n$, and let $S_k = \sum_{j=1}^k X_j$ denote a random walk with positive drift $\mathbf{E}\, S_k = 2k\beta_{n,m} \geq 2k/n$. Let*

$$T_2 = \inf\{k : |S_k| \geq \sqrt{kn}\}$$

*denote the time of first passage to the receding (two-sided) boundary at $\pm\sqrt{kn}$. Then*

$$\mathbf{E}\, T_2 \sim \frac{n}{4\beta_{n,m}^2} \qquad (n \to \infty).$$

Using $\mathbf{E}\, T_2$ as a rough estimate for the expected mistake bound (when $m$ is within capacity), we get the asymptotic estimate $O(n^3)$ for the expected mistake bound of single component Perceptron Training as $n \to \infty$.

### Acknowledgements

The support of research grant AFOSR-89-0523 from the Air Force Office of Scientific Research is gratefully acknowledged.

## References

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the Vapnik-Cěrvonenkis dimension," *Jnl. Assoc. Comput. Machinery*, vol. 36, no. 4, pp. 929–965, 1989.

[Fel68] W. Feller, *An Introduction to Probability Theory and its Applications*, vol. I, third edition. New York: John Wiley, 1968.

[GJ79] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman, 1979.

[GH90] H. P. Graf and D. Henderson, "A reconfigurable CMOS neural network," *Digest of Technical Papers of the IEEE Int'l Solid State Circuits Conf.*, San Franscisco, pp. 144–145, Feb. 1990.

[KGV83] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimisation by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[KP88(a)] J. Komlós and R. Paturi, "Convergence results in an associative memory model," *Neural Networks*, vol. 1, pp. 239–250, 1988.

[KP88(b)] J. Komlós and R. Paturi, "Effect of connectivity in associative memory models," *Tech. Report*, CS88-131, University of California, San Diego, 1988.

[Lit88] N. Littlestone, "Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm," *Machine Learning*, vol. 2, pp. 285–318, 1988.

[MPRV87] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 461–482, 1987.

[Mei91] R. Meir, private communication.

[N88] C. Newman, "Memory capacity in neural network models: rigourous lower bounds," *Neural Networks*, vol. 1, pp. 223–238, 1988.

[Ros62] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, D.C.: Spartan Books, 1962.

[VC71] V. N. Vapnik and A. Ya. Cĕrvonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theor. Prob. Appl.*, vol. 16, pp. 264–280, 1971.

[Veu91(a)] S. S. Venkatesh, "Off-line and on-line approaches to learning binary weights for majority functions from examples." Unpublished manuscript, 1991.

[Ven91(b)] S. S. Venkatesh, "Directed drift: a new linear threshold algorithm for learning binary weights on-line," *Jnl. Comp. Sci. and Sys.*, 1991. [To appear.]

[Ven91(c)] S. S. Venkatesh, "Computation and learning in the context of neural network capacity," in *Neural Networks for Perception*, (ed. H. Wechsler). New York: Academic Press, 1991.

[Ven91(d)] S. S. Venkatesh, "Probabilistic capacity and links to distribution dependent learning," *Workshop on Theoretical Issues in Neural Networks*, Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, May 1991.

[Ven91(e)] S. S. Venkatesh, "Robustness in neural computation: random graphs and sparsity," *IEEE Trans. Inform. Theory*, 1991. [In press.]

[VB91(a)] S. S. Venkatesh and P. Baldi, "Higher order neural networks: maximal capacity," *Jnl. Compl.*, 1991. [In press.]

[VB91(b)] S. S. Venkatesh and P. Baldi, "Higher order neural networks: the outer product algorithm," *Jnl. Compl.*, 1991. [In press.]

[VF91] S. S. Venkatesh and J. Franklin, "How much information can one bit of memory retain about a Bernoulli sequence?" *IEEE Trans. Inform. Theory*, 1991. [In press.]

[VP91] S. S. Venkatesh and D. Psaltis, "On reliable computation with formal neurons," *IEEE Trans. Pattern Anal. Machine Intelligence*, 1991. [In press.]

To appear:
Advances in Neural Information
Processing Systems, 3
(D. Touretzky & R. Lipmann, eds.)
Morgan Kauffmann, San Mateo, CA

1991

# The Devil and the Network: What Sparsity Implies to Robustness and Memory

**Sanjay Biswas** and **Santosh S. Venkatesh**
Department of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104

## Abstract

Robustness is a commonly bruited property of neural networks; in particular, a folk theorem in neural computation asserts that neural networks—in contexts with large interconnectivity—continue to function efficiently. albeit with some degradation, in the presence of component damage or loss. A second folk theorem in such contexts asserts that dense interconnectivity between neural elements is a *sine qua non* for the efficient usage of resources. These premises are formally examined in this communication in a setting that invokes the notion of the "devil"[1] in the network as an agent that produces sparsity by snipping connections.

## 1 ON REMOVING THE FOLK FROM THE THEOREM

Robustness in the presence of component damage is a property that is commonly attributed to neural networks. The content of the following statement embodies this sentiment.

> *Folk Theorem 1: Computation in neural networks is not substantially affected by damage to network components.*

While such a statement is manifestly not true in general—witness networks with "grandmother cells" where damage to the critical cells fatally impairs the computational ability of the network—there is anecdotal evidence in support of it in

[1] Well, maybe an imp.

situations where the network has a more "distributed" flavour with relatively dense interconnectivity of elements and a distributed format for the storage of information. Qualitatively, the phenomenon is akin to holographic modes of storing information where the distributed, non-localised format of information storage carries with it a measure of security against component damage.

The flip side to the robust folk theorem is the following observation, robustness notwithstanding:

> *Folk Theorem 2: Dense interconnectivity is a sine qua non for efficient usage of resources; in particular, sparser structures exhibit a degradation in computational capability.*

Again, disclaimers have to be thrown in on the applicability of such a statement. In recurrent network architectures, however, this might seem to have some merit. In particular, in associative memory applications, while structural robustness might guarantee that the loss in memory storage capacity with increased interconnection sparsity may not be catastrophic, nonetheless intuitively a drop in capacity with increased sparsity may be expected.

This communication represents an effort to mathematically codify these tenets. In the setting we examine we formally introduce sparse network interconnectivity by invoking the notion of a (puckish) devil in the network which severs interconnection links between neurons. Our results here involve some surprising consequences—viewed in the light of the two folk theorems—of sparse interconnectivity to robustness and to memory storage capability. Only the main results are stated here; for extensions and details of proofs we refer the interested reader to Venkatesh (1990) and Biswas and Venkatesh (1990).

**Notation** We denote by $\mathbb{B}$ the set $\{-1,1\}$. For every integer $k$ we denote the set of integers $\{1,2,\ldots,k\}$ by $[k]$. By ordered multiset we mean an ordered collection of elements with repetition of elements allowed, and by $k$-set we mean an ordered multiset of $k$ elements. All logarithms in the exposition are to base $e$.

## 2 RECURRENT NETWORKS

### 2.1 INTERCONNECTION GRAPHS

We consider a recurrent network of $n$ formal neurons. The allowed pattern of neural interconnectivity is specified by the edges of a ***(bipartite) interconnectivity graph***, $G_n$, on vertices, $[n] \times [n]$. In particular, the existence of an edge $\{i,j\}$ in $G_n$ is indicative that the state of neuron $j$ is input to neuron $i$.[2] The network is characterised by an $n \times n$ matrix of weights, $\mathbf{W} = [w_{ij}]$, where $w_{ij}$ denotes the (real) weight modulating the state of neuron $j$ at the input of neuron $i$. If $\mathbf{u} \in \mathbb{B}^n$ is the current state of the system, an update, $u_i \mapsto u_i'$ of the state of neuron $i$ is

[2] Equivalently, imagine a devil loose with a pair of scissors snipping those interconnections for which $\{i,j\} \notin G_n$. For a complementary discussion of sparse interconnectivity see Komlós and Paturi (1988).

specified by the linear threshold rule

$$u'_i = \operatorname{sgn}\left(\sum_{j:\{i,j\}\in G} w_{ij}u_j\right).$$

The network dynamics describe trajectories in a state space comprised of the vertices of the $n$-cube.[3] We are interested in an associative memory application wh^re we wish to store a desired set of states—the *memories*—as fixed points of the network, and with the property that errors in an input representation of a memory are corrected and the memory retrieved.

## 2.2 DOMINATORS

Let $\mathbf{u} \in \mathbb{B}^n$ be a memory and $0 \leq \rho < 1$ a parameter. Corresponding to the memory $\mathbf{u}$ we generate a probe $\hat{\mathbf{u}} \in \mathbb{B}^n$ by independently specifying the components, $\hat{u}_j$, of the probe as follows:

$$\hat{u}_j = \begin{cases} u_j & \text{with probability } 1-\rho \\ -u_j & \text{with probability } \rho. \end{cases} \tag{1}$$

We call $\hat{\mathbf{u}}$ a *random probe with parameter* $\rho$.

**Definition 2.1** We say that a memory, $\mathbf{u}$, *dominates over a radius* $\rho n$ if, with probability approaching one as $n \to \infty$, the network corrects all errors in a random probe with parameter $\rho$ in one synchronous step. We call $\rho$ the *(fractional) dominance radius.* We also say that $\mathbf{u}$ is *stable* if it is a 0-dominator.

REMARKS: Note that stable memories are just fixed points of the network. Also, the expected number of errors in a probe is $\rho n$.

## 2.3 CODES

For given integers $m \geq 1$, $n \geq 1$, a *code*, $\mathcal{K}_n^m$, is a collection of ordered multisets of size $m$ from $\mathbb{B}^n$. We say that an $m$-set of memories is *admissible* iff it is in $\mathcal{K}_n^m$.[4] Thus, a code just specifies which $m$-sets are allowable as memories. Examples of codes include: the set of all multisets of size $m$ from $\mathbb{B}^n$; a single multiset of size $m$ from $\mathbb{B}^n$; all collections of $m$ mutually orthogonal vectors in $\mathbb{B}^n$; all $m$-sets of vectors in $\mathbb{B}^n$ in general position.

Define two ordered multisets of memories to be *equivalent* if they are permutations of one another. We define the *size* of a code, $\mathcal{K}_n^m$, to be the number of distinct equivalence classes of $m$-sets of memories. We will be interested in codes of relatively large size: $\log|\mathcal{K}_n^m|/n \to \infty$ as $n \to \infty$. In particular, we require at least an exponential number of choices of (equivalence classes of) admissible $m$-sets of memories.

[3] As usual, there are Liapunov functions for the system under suitable conditions on the interconnectivity graph and the corresponding weights.

[4] We define admissible $m$-sets of memories in terms of ordered multisets rather than sets so as to obviate certain technical nuisances.

## 2.4 CAPACITY

For each fixed $n$ and interconnectivity graph, $G_n$, an *algorithm*, $\mathcal{X}$, is a prescription which, given an $m$-set of memories, produces a corresponding set of interconnection weights, $w_{ij}$, $i \in [n]$, $\{i,j\} \in G_n$. For $m \geq 1$ let $\mathcal{A}(\mathbf{u}^1, \ldots, \mathbf{u}^m)$ be some attribute of $m$-sets of memories. (The following, for instance, are examples of attributes of admissible sets of memories: all the memories are stable in the network generated by $\mathcal{X}$; almost all the memories dominate over a radius $\rho n$.) *For given $n$ and $m$, we choose a random $m$-set of memories, $\mathbf{u}^1, \ldots, \mathbf{u}^m$, from the uniform distribution on $\mathcal{K}_n^m$.*

**Definition 2.2** Given interconnectivity graphs $G_n$, codes $\mathcal{K}_n^m$, and algorithm $\mathcal{X}$, a sequence, $\{C_n\}_{n=1}^{\infty}$, is a *capacity function for the attribute $\mathcal{A}$* (or *$\mathcal{A}$-capacity* for short) if for $\lambda > 0$ arbitrarily small:

a) $\mathbf{P}\{\mathcal{A}(\mathbf{u}^1, \ldots, \mathbf{u}^m)\} \to 1$ as $n \to \infty$ whenever $m \leq (1-\lambda)C_n$;

b) $\mathbf{P}\{\mathcal{A}(\mathbf{u}^1, \ldots, \mathbf{u}^m)\} \to 0$ as $n \to \infty$ whenever $m \geq (1+\lambda)C_n$.

We also say that $C_n$ is a *lower $\mathcal{A}$-capacity* if property (a) holds, and that $C_n$ is an *upper $\mathcal{A}$-capacity* if property (b) holds.

For $m \geq 1$ let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of memories chosen from a code $\mathcal{K}_n^m$. The *outer-product algorithm* specifies the interconnection weights, $w_{ij}$, according to the following rule: for $i \in [n]$, $\{i,j\} \in G_n$,

$$w_{ij} = \sum_{\beta=1}^{m} u_i^{\beta} u_j^{\beta}. \tag{2}$$

In general, if the interconnectivity graph, $G_n$, is symmetric then, under a suitable mode of operation, there is a Liapunov function for the network specified by the outer-product algorithm. *Given graphs $G_n$, codes $\mathcal{K}_n^m$, and the outer-product algorithm, for fixed $0 \leq \rho < 1/2$ we are interested in the attribute $\mathcal{D}_\rho$ that each of the $m$ memories dominates over a radius $\rho n$.*

# 3 RANDOM GRAPHS

We investigate the effect of a random loss of neural interconnections in a recurrent network of $n$ neurons by considering a random bipartite interconnectivity graph $RG_n$ on vertices $[n] \times [n]$ with

$$\mathbf{P}\ \{\{i,j\} \in RG_n\} = p$$

for all $i \in [n]$, $j \in [n]$, and with these probabilities being mutually independent. The interconnection probability $p$ is called the *sparsity parameter* and may depend on $n$. The system described above is formally equivalent to beginning with a fully-interconnected network of neurons with specified interconnection weights $w_{ij}$, and then invoking a devil which randomly severs interconnection links, independently retaining each interconnection weight $w_{ij}$ with probability $p$, and severing it (replacing it with a zero weight) with probability $q = 1 - p$.

Let $\mathcal{CK}_n^m$ denote the *complete code* of all choices of ordered multisets of size $m$ from $\mathbb{B}^n$.

**Theorem 3.1** *Let $0 \leq \rho < 1/2$ be a fixed dominance radius, and let the sparsity parameter $p$ satisfy $pn^2 \to \infty$ as $n \to \infty$. Then $(1-2\rho)^2 pn/2 \log pn^2$ is a $\mathcal{D}_\rho$-capacity for random interconnectivity graphs $RG_n$, complete codes $\mathcal{CK}_n^m$, and the outer-product algorithm.*

REMARKS: The above result graphically validates Folk Theorem 1 on the fault-tolerant nature of the network; specifically, the network exhibits a *graceful degradation* in storage capacity as the loss in interconnections increases. Catastrophic failure occurs only when $p$ is smaller than $\log n/n$: *each neuron need retain only of the order of $\Omega(\log n)$ links of a total of $n$ possible links with other neurons for useful associative properties to emerge.*

## 4 BLOCK GRAPHS

One of the simplest (and most regular) forms of sparsity that a favourably disposed devil might enjoin is block sparsity where the neurons are partitioned into disjoint subsets of neurons with full-interconnectivity within each subset and no neural interconnections between subsets. The weight matrix in this case takes on a block diagonal form, and the interconnectivity graph is composed of a set of disjoint, complete bipartite sub-graphs.

More formally, let $1 \leq b \leq n$ be a positive integer, and let $\{I_1, \ldots, I_{n/b}\}$ partition $[n]$ such that each subset of indices, $I_k$, $k = 1, \ldots, n/b$, has size $|I_k| = b$.[5] We call each $I_k$ a *block* and $b$ the *block size*. We specify the edges of the *(bipartite) block interconnectivity graph* $BG_n$ by $\{i,j\} \in BG_n$ iff $i$ and $j$ lie in a common block.

**Theorem 4.1** *Let the block size $b$ be such that $b = \Omega(n)$ as $n \to \infty$, and let $0 \leq \rho < 1/2$ be a fixed dominance radius. Then $(1-2\rho)^2 b/2 \log bn$ is a $\mathcal{D}_\rho$-capacity for block interconnectivity graphs $BG_n$, complete codes $\mathcal{CK}_n^m$, and the outer-product algorithm.*

**Corollary 4.2** *Under the conditions of theorem 4.1 the fixed point memory capacity is $b/2 \log bn$.*

**Corollary 4.3** *For a fully-interconnected graph, complete codes $\mathcal{CK}_n^m$, and the outer-product algorithm, the fixed point memory capacity is $n/4 \log n$.*

Corollary 4.3 is the main result shown by McEliece, Posner, Rodemich, and Venkatesh (1987). Theorem 4.1 extends the result and shows (formally validating the intuition espoused in Folk Theorem 2) that increased sparsity causes a loss in capacity if the code is complete, i.e., all choices of memories are considered admissible. It is possible, however, to design codes to take advantage of the sparse interconnectivity structure, rather at odds with the Folk Theorem.

[5] Here, as in the rest of the paper, we ignore details with regard to integer rounding.

Without loss of generality let us assume that block $I_1$ consists cf the first $b$ indices, $[b]$, block $I_2$ the next $b$ indices, $[2b]-[b]$, and so on, with the last block $I_{n/b}$ consisting of the last $b$ indices, $[n]-[n-b]$. We can then partition any vector $\mathbf{u} \in \mathbb{B}^n$ as

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{n/b} \end{pmatrix}, \tag{3}$$

where for $k = 1, \ldots, n/b$, $\mathbf{u}_k$ is the vector of components corresponding to block $I_k$. For $M \geq 1$ we form the *block code* $\mathcal{BK}_n^{M^{n/b}}$ as follows: to each ordered multiset of $M$ vectors, $\mathbf{u}^1, \ldots, \mathbf{u}^M$ from $\mathbb{B}^n$, we associate a unique ordered multiset in $\mathcal{BK}_n^{M^{n/b}}$ by lexicographically ordering all $M^{n/b}$ vectors of the form

$$\begin{pmatrix} \mathbf{u}_1^{\alpha_1} \\ \mathbf{u}_2^{\alpha_2} \\ \vdots \\ \mathbf{u}_{n/b}^{\alpha_{n/b}} \end{pmatrix}, \qquad \alpha_1, \alpha_2, \ldots, \alpha_{n/b} \in [M].$$

Thus, we obtain an admissible set of $M^{n/b}$ memories from any ordered multiset of $M$ vectors in $\mathbb{B}^n$ by "mixing" the blocks of the vectors. We call each $M$-set of vectors, $\mathbf{u}^1, \ldots, \mathbf{u}^M \in \mathbb{B}^n$, the *generating vectors* for the corresponding admissible set of memories in $\mathcal{BK}_n^{M^{n/b}}$.

EXAMPLE: Consider a case with $n = 4$, block size $b = 2$, and $M = 2$ generating vectors. To any 2-set of generating vectors there corresponds a unique $4(=M^{n/b})$-set in the block code as follows:

$$\left(\begin{array}{c} u_1^1 \\ u_2^1 \\ \hline u_3^1 \\ u_4^1 \end{array}\right), \left(\begin{array}{c} u_1^2 \\ u_2^2 \\ \hline u_3^2 \\ u_4^2 \end{array}\right) \longmapsto \left(\begin{array}{c} u_1^1 \\ u_2^1 \\ \hline u_3^1 \\ u_4^1 \end{array}\right), \left(\begin{array}{c} u_1^1 \\ u_2^1 \\ \hline u_3^2 \\ u_4^2 \end{array}\right), \left(\begin{array}{c} u_1^2 \\ u_2^2 \\ \hline u_3^1 \\ u_4^1 \end{array}\right), \left(\begin{array}{c} u_1^2 \\ u_2^2 \\ \hline u_3^2 \\ u_4^2 \end{array}\right).$$

**Theorem 4.4** *Let $0 \leq \rho < 1/2$ be a fixed dominance radius. Then we have the following capacity estimates for block interconnectivity graphs $BG_n$, block codes $\mathcal{BK}_n^m$, and the outer-product algorithm:*

*a) If the block size $b$ satisfies $n \log\log bn / b \log bn \to 0$ as $n \to \infty$ then the $\mathcal{D}_\rho$-capacity is*

$$\left[\frac{(1-2\rho)^2 b}{2 \log bn}\right]^{n/b}.$$

*b) Define for any $\nu$*

$$C_n(\nu) = 2^{\frac{n \log b}{b \log 2}\left[1 - \frac{\log\log bn + \log\left(2(1-2\rho)^{-2}\right)}{\log b} + \frac{\nu \log\log bn}{(\log b)(\log bn)}\right]}.$$

*If the block size $b$ satisfies $b/\log n \to \infty$ and $b \log bn / \log\log bn = O(n)$ as $n \to \infty$, then $C_n(\nu)$ is a lower $\mathcal{D}_\rho$-capacity for any choice of $\nu < 3/2$ and $C_n(\nu)$ is an upper $\mathcal{D}_\rho$-capacity for any $\nu > 3/2$.*

**Corollary 4.5** *If, for fixed $t \geq 1$, we have $b = n/t$, then, under the conditions of theorem 4.4, the $\mathcal{D}_\rho$-capacity is*

$$(1-2\rho)^{2t} t^{-t} 4^{-t} \left(\frac{n}{\log n}\right)^t .$$

**Corollary 4.6** *For any fixed dominance radius $0 \leq \rho < 1/2$, and for any $\tau < 1$, a constant $c > 0$ and a code of size $\Omega\left(2^{cn^{2-\tau}}\right)$ can be found such that it is possible to achieve lower $\mathcal{D}_\rho$-capacities which are $\Omega\left(2^{n^\tau}\right)$ in recurrent neural networks with interconnectivity graphs of degree $\Theta\left(n^{1-\tau}\right)$.*

REMARKS: If the number of blocks is kept fixed as $n$ grows (i.e., the block size grows linearly with $n$) then capacities polynomial in $n$ are attained. If the number of blocks increases with $n$ (i.e., the block size grows sub-linearly with $n$) then super-polynomial capacities are attained. Furthermore, we have the surprising result rather at odds with Folk Theorem 2 that very large storage capacities can be obtained at the expense of code size (while still retaining large code sizes) in *increasingly sparse networks.*

## Acknowledgements

The support of research grants from E. I. Dupont de Nemours, Inc. and the Air Force Office of Scientific Research (grant number AFOSR 89-0523) is gratefully acknowledged.

## References

Biswas, S. and S. S. Venkatesh (1990), "Codes, sparsity, and capacity in neural associative memory," submitted for publication.

Komlós, J. and R. Paturi (1988), "Effects of connectivity in associative memory models," Technical Report CS88-131, University of California, San Diego, 1988.

McEliece, R. J., E. C. Posner, E. R. Rodemich, and S. S. Venkatesh (1987), "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 461-482.

Venkatesh, S. S. (1990), "Robustness in neural computation: random graphs and sparsity," to appear *IEEE Trans. Inform. Theory*.

# Correspondence

## Robustness in Neural Computation: Random Graphs and Sparsity

Santosh S. Venkatesh, *Member, IEEE*

*Abstract*—**Robustness is a commonly bruited property of neural networks; in particular, a folk theorem in neural computation asserts that fully-interconnected neural networks continue to function efficiently in the presence of component damage. This communication is an effort to mathematically codify this belief. Component damage is introduced in a fully-interconnected neural network model of $n$ neurons by randomly deleting links between neurons. An analysis of the outer-product algorithm for this random graph model of sparse interconnectivity using a simple generalisation of Chebyshev's inequality yields the following main result: *if the probability of losing any given link between two neurons is $1 - p$, then the outer product algorithm can store of the order of $pn/\log pn^2$ stable memories correcting a linear number of random errors.* In particular, the average degree of the interconnectivity graph dictates the memory storage capability, and functional storage of memories as stable states is feasible abruptly when the average number of neural interconnections retained by a neuron exceeds the order of $\log n$ links (of a total of $n$ possible links) with other neurons. This work complements the results of Komlós and Paturi on worst case error correction for fixed underlying interconnectivity graphs.**

*Index Terms*—**Neural networks, robustness, random graph, sparsity, outer-product algorithm.**

### I. Introduction

### *A. The Problem*

Robustness in the presence of component damage is a property that is common attributed to neural networks. The content of the following statement embodies this sentiment.

*Folk Theorem:* Computation in neural networks is not substantially affected by damage to network components.

While such a statement cannot hold true in general—witness networks with "grandmother cells" where damage to the critical cells fatally impairs the computational ability of the network—there is anecdotal evidence in support of it in situations where the network has a more "distributed" flavor with a relatively dense interconnectivity of elements. In such situations, experimental evidence indicates that networks of neural elements do indeed possess a measure of fault-tolerance [1]. Qualitatively, the phenomenon is akin to holographic modes of storing information where the distributed, nonlocalized format of information storage carries with it a measure of security against component damage.

Neural models for associative memory are natural candidates for investigation of fault-tolerant properties. These models typically consist of a fully-interconnected network of formal neurons (linear threshold elements). Information is stored in these models in the interconnections between neural elements.

The *outer-product algorithm*, which we described in the sequel, is a particularly simple algorithmic prescription for storing memories in a fully-interconnected, recurrent neural network. The algorithm has good associative properties, and consequently, has been the subject of some searching mathematical investigations: McEliece *et al.* [2] showed that the algorithm can store of the order of $n/\log n$ memories with correction of a linear number of random errors; subsequent investigations by Komlós and Paturi [3] showed that the storage capacities derived by McEliece *et al.* persist even in the case of worst case errors; complementary results due to Newman [4] indicate that storage capacities linear in $n$ can be achieved in the outer-product algorithm if errors can be tolerated in the recall of the memories. Nonrigourous results qualitatively similar to those above have also been reported by Hopfield [1] and Amit *et al.* [5].

We investigate robustness in the model by invoking a devil (well, maybe an imp) which randomly severs interconnection links in a fully interconnected network of $n$ neurons, with weights specified by the outer-product algorithm. The sparse network that results is essentially specified by an underlying random interconnectivity graph. The following are our main results, which provide a graphic validation of the folk theorem in this instance.

If the probability of retaining any given link between two neurons is $p$, then the outer-product algorithm can store of the order of $pn/\log pn^2$ stable memories with correction of a linear number of random errors. Functional storage of memories as stable states is feasible when the average degree of the random interconnectivity graph exceeds the order of $\log n$; memories will be stable with respect to a linear number of random errors in components if the average degree of the random interconnectivity graph exceeds the order of $\log^3 n$.

These results are consistent with results of Komlós and Paturi [6] who have analysed *worst case errors* in networks with interconnectivities specified by *fixed* underlying graphs. Using sophisticated and powerful techniques from large deviation probability theory they show results on convergence times and the radius of attraction within which *all* points are attracted to the memories in terms of the spectrum of the underlying graph. The *random graph* model analysed here provides great attendant simplicity in the analysis of the correction of *random errors*. In fact, as we will see in the sequel, the main results fall out of a rather simple application of Chebyshev's inequality.

*Notation:* We denote by $\mathbb{B}$ the set $\{-1, 1\}$. For any positive integer $k$, we denote by $[k]$ the set $\{1, \cdots, k\}$. All logarithms in the exposition are to the Napier base $e$. We also use $c_1, c_2, \cdots$, to denote absolute positive constants. We invoke standard asymptotic notation in the sequel; in addition, if $\{x_n\}$ is a positive sequence and $\{y_n(\epsilon)\}$ is another positive sequence which is a function of a real parameter $\epsilon$, we denote $y_n(\epsilon) = O_\epsilon(x_n)$ if, for every fixed value of $\epsilon$, we can find $K(\epsilon) > 0$ (independent of $n$) such that $y_n(\epsilon) < K(\epsilon)x_n$ for every $n$.

### *B. The Setting*

We consider a network of $n$ formal neurons. Each neuron in the system assumes one of two binary states, $-1$ or $+1$, and the network as a whole evolves in the state space, $\mathbb{B}^n$, of binary vectors of length $n$. Neural interconnectivity is specified by a random

Manuscript received August 16, 1990; revised January 7, 1991. This work was supported by E. I. Dupont de Nemours, Inc. and Air Force Office of Scientific Research Grant No. AFOSR 89-0523. This work was presented in part at the Neural Information Processing Systems Conference, Denver, CO, November 1990.

The author is with the Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104.

IEEE Log Number 9105774.

bipartite interconnectivity graph $G_n$ on vertices $[n] \times [n]$ with

$$P\{\{i, j\} \in G_n\} = p,$$

for all $i \in [n]$, $j \in [n]$, and with these probabilities being mutually independent. The interconnection probability $p$ is called the *sparsity parameter* and may depend on $n$. A real interconnection weight $w_{ij}$ is associated with each edge $\{i, j\} \in G_n$. We adopt the convention $w_{ij} \equiv 0$ if $\{i, j\} \notin G_n$. The state of each neuron is updated based on the sign of a linear form computed by the interconnection weights and the current state of the system: if $u \in \mathbb{B}^n$ is the current state of the system, an update, $u_i \to u_i'$ of the state of the $i$th neuron is specified by the linear threshold rule[1]

$$u_i' = \operatorname{sgn}\left(\sum_{j:\{i,j\} \in G_n} w_{ij} u_j\right) = \operatorname{sgn}\left(\sum_{j=1}^{n} w_{ij} u_j\right).$$

Neural updates may be either synchronous, with every neuron being updated in concert, or asynchronous, with at most one neuron being updated at any instant.

The system previously described is formally equivalent to beginning with a fully-interconnected network of neurons with specified interconnection weights $w_{ij}$, and then invoking a devil which randomly severs interconnection links, independently retaining each interconnection weight $w_{ij}$ with probability $p$, and severing it (replacing it with a zero weight) with probability $q = 1 - p$. Note that the expected number of weights retained by any neuron in the network is $pn$, and the expected number of nonzero weights in the network is $pn^2$.[2]

### C. The Algorithm

As in any recurrent dynamical system, we are interested in the fixed points of the system. In particular, we focus on an associative memory application where we wish to store a desired set of states—*the fundamental memories*—as fixed points of the network, and with the property that errors in an input representation of a memory are corrected and the memory retrieved.

Let $u^1, \cdots, u^m \in \mathbb{B}^n$ be an $m$-set of fundamental memories whose components, $u_j^\beta$, are drawn independently from a sequence of symmetric Bernoulli trials; viz., for $j = 1, \cdots, n$, and $\beta = 1, \cdots, m$,

$$u_j^\beta = \begin{cases} -1, & \text{with probability } 1/2, \\ 1, & \text{with probability } 1/2. \end{cases}$$

The outer-product algorithm specifies interconnection weights, $\hat{w}_{ij}$, according to the following prescription:[3] for $i \in [n]$, $j \in [n]$,

$$\hat{w}_{ij} = \begin{cases} \sum_{\beta=1}^{m} u_i^\beta u_j^\beta, & \text{if } j \neq i, \\ 0, & \text{if } j = i. \end{cases}$$

In our sparse interconnectivity model, each weight $\hat{w}_{ij}$ is independently severed with probability $q = 1 - p$, and retained with probability $p$. More formally, let $\pi_{ij}$, $i \in [n]$, $j \in [n]$ be a sequence of i.i.d. random variables with

$$\pi_{ij} = \begin{cases} 0, & \text{if } \{i, j\} \notin G_n, \\ 1, & \text{if } \{i, j\} \in G_n. \end{cases}$$

For $i \in [n]$ and $j \in [n]$ we can now define the interconnection weights of the sparse, random network by

$$w_{ij} = \pi_{ij}\hat{w}_{ij} = \begin{cases} \pi_{ij} \sum_{\beta=1}^{m} u_i^\beta u_j^\beta, & \text{if } j \neq i, \\ 0, & \text{if } j = i. \end{cases} \tag{1}$$

The variables $\pi_{ij}$ are simply the indicator random variables for the edges of the random bipartite interconnectivity graph $G_n$.

## II. Stable Memories

A basic requirement that we would like to impose is that the memories are *stable*, i.e., fixed points of the network:

$$u_i^\alpha = \operatorname{sgn}\left(\sum_{j=1}^{n} w_{ij} u_j^\alpha\right), \qquad i = 1, \cdots, n, \qquad \alpha = 1, \cdots, m.$$

We begin by estimating the number of memories that can be made stable in the outer product algorithm for a random interconnectivity graph with sparsity parameter $p$. The following theorem is our main result.

*Theorem 1:* Let the sparsity parameter $p$ satisfy $pn^2 \to \infty$ as $n \to \infty$. For any fixed $\epsilon > 0$, we then have the following.

a) If, as $n \to \infty$, we choose the number of memories, $m$, such that

$$m \leq \frac{pn}{2\log pn^2}\left[1 + \frac{\log\log pn^2 + \log 2\epsilon}{\log pn^2} - O_\epsilon\left(\frac{\log\log pn^2}{\log^2 pn^2}\right)\right], \tag{2}$$

then the probability that all $m$ memories are fixed points is at least as large as $1 - \epsilon - o(1)$.

b) If, as $n \to \infty$, we choose the number of memories, $m$, such that

$$m \leq \frac{pn}{2\log n}\left[1 + \frac{\log \epsilon}{\log n} + O_\epsilon\left(\frac{1}{\log^2 n}\right)\right], \tag{3}$$

then the expected number of memories that are fixed points is at least as large as $[1 - \epsilon - o(1)]m$.

*Remarks:* In particular, we can store at least $pn/2\log pn^2$ memories if *all* the memories are required to be stable, and at least $pn/2\log n$ memories if only *most* of the memories are required to be stable. This result reduces to the capacity result for full interconnectivity of McEliece *et al.* [2] if we set $p = 1$, i.e., no interconnections are severed.

This result illustrates graphically the fault-tolerant nature of the network; specifically, the network exhibits a *graceful degradation* in storage capacity as the loss in interconnections increases. Memory storage is achieved if the sparsity parameter, $p$, is at least of the order of $\log n/n$, i.e., each neuron retains essentially of the order of $\log n$ weights out of its original complement of $n$ weights. In particular, if $p = K\log n/n$ then the network can store at least $K/2$ memories; if $p = n^{-\tau}$ for any $0 \leq \tau < 1$ then the network can store at least $n^{1-\tau}/2(2 - \tau)\log n$ memories; if $p$ is equal to a constant $0 < c \leq 1$ then the network can store at least $cn/4\log n$ memories. As a graphic example, the network can loose half its

[1] We define the sgn function by $\operatorname{sgn} x = x/|x|$ for all $x \neq 0$ and $\operatorname{sgn} 0 = 1$.

[2] We could, if we wished, enforce symmetry in the sparse network by considering the links between neurons as bidirectional so that severing a link automatically produces symmetric zeroes in the weight matrix. For the purposes of this correspondence it is immaterial which random graph model we select.

[3] Variations are possible with diagonal terms $\hat{w}_{ii} \neq 0$, but are all functionally equivalent.

interconnections with essentially no change in the storage characteristics. If $p = o(\log n/n)$, i.e., the average degree of the interconnectivity graph is $o(\log n)$, we should, of course, expect catastrophic failure of the memory.

*Proof:* Let us define the doubly indexed random variables, $X_i^\alpha$, by

$$X_i^\alpha = u_i^\alpha \sum_{j=1}^{n} w_{ij} u_j^\alpha, \qquad i = 1, \cdots, n, \qquad \alpha = 1, \cdots, m.$$

It is readily seen that $X_i^\alpha > 0$ implies that the $i$th component of the $\alpha$th memory is stable. Thus, we will require that $X_i^\alpha > 0$ for each $i \in [n]$ and $\alpha \in [m]$ if each of the memories is to be a fixed point of the network.

Let us first consider the requirements that must be satisfied for a single component of a memory to be fixed. Substituting for the weights, $w_{ij}$, from (1) we have

$$X_i^\alpha = \sum_{j \neq i} \pi_{ij} \sum_{\beta=1}^{m} u_i^\alpha u_j^\alpha u_i^\beta u_j^\beta = \sum_{j \neq i} \pi_{ij}\Big(1 + \sum_{\beta \neq \alpha} Z_{ij}^{\alpha\beta}\Big),$$

where we define

$$Z_{ij}^{\alpha\beta} = u_i^\alpha u_j^\alpha u_i^\beta u_j^\beta.$$

We hold the indices $i$ and $\alpha$ fixed for the nonce, and for notational simplicity suppress the $i$ and $\alpha$ dependence of both $X_i^\alpha$ and $Z_{ij}^{\alpha\beta}$. We will need the following result which estimates the probability that a single component of a memory is not stable.

*Lemma 1:* If, as $n \to \infty$, the parameters $p$ and $m$ vary such that $p\sqrt{n}/m \to 0$, then

$$P\{X \leq 0\} \leq [1 + o(1)] \exp\Big(-\frac{pn}{2m}\Big) \qquad (n \to \infty).$$

*Proof:* For fixed $i$ and $\alpha$, the random variables $Z_j^\beta$ are i.i.d. and symmetric, and take on the values $-1$ and 1 with equal probability 1/2. (This follows from the fact that the memory components are i.i.d., symmetric ± 1 random variables, and that the distinct component $u_j^\beta$ appears solely in the expression for $Z_j^\beta$.) Applying the generalized Chebyshev inequality of Lemma A1, we have the following estimate for the probability that there is an error in the retrieval of a single component of a memory:

$$\begin{aligned} P\{X \leq 0\} &\leq \inf_{r \geq 0} E(e^{-rX}) \\ &= \inf_{r \geq 0} E\Big[\exp\Big\{-\sum_{j \neq i} r\pi_{ij}\Big(1 + \sum_{\beta \neq \alpha} Z_j^\beta\Big)\Big\}\Big] \\ &= \inf_{r \geq 0} E\Big[\prod_{j \neq i} \exp\Big\{-r\pi_{ij}\Big(1 + \sum_{\beta \neq \alpha} Z_j^\beta\Big)\Big\}\Big]. \end{aligned}$$

The (1, 0) random variables, $\pi_{ij}$, $j \neq i$ are i.i.d., as are the ± 1 random variables $Z_j^\beta$, $j \neq i$, $\beta \neq \alpha$. It follows that the terms in the product are also i.i.d. random variables. For notational simplicity, we set $M = m - 1$ and $N = n - 1$. We now have

$$\begin{aligned} P\{X \leq 0\} &\leq \inf_{r \geq 0}\Big[E \exp\Big\{-r\pi_{ij}\Big(1 + \sum_{\beta \neq \alpha} Z_j^\beta\Big)\Big\}\Big]^N \\ &= \inf_{r \geq 0}\Big[pE \exp\Big\{-r\Big(1 + \sum_{\beta \neq \alpha} Z_j^\beta\Big)\Big\} + q\Big]^N \\ &= \inf_{r \geq 0}\big[pe^{-r}(Ee^{-rZ_j^\beta})^M + q\big]^N \\ &= \inf_{r \geq 0}\big[pe^{-r}(\cosh r)^M + q\big]^N. \end{aligned}$$

Now, for every real $r$ we have $\cosh r \leq e^{r^2/2}$. Hence,

$$\begin{aligned} P\{X \leq 0\} &\leq \inf_{r \geq 0}\big[pe^{-r + Mr^2/2} + q\big]^N \leq \big[pe^{-1/2M} + q\big]^N \\ &= \big[1 - p(1 - e^{-1/2M})\big]^N. \end{aligned} \qquad (4)$$

Recalling that $M = m - 1$ it is easy to verify that for $m > 1$

$$1 - e^{-1/2M} = \frac{1}{2m} + \frac{3}{8m^2} + O\Big(\frac{1}{m^3}\Big) > \frac{1}{2m}.$$

It follows that

$$\begin{aligned} P\{X \leq 0\} &\leq \Big[1 - \frac{p}{2m}\Big]^N = \exp\Big[N \log\Big\{1 - \frac{p}{2m}\Big\}\Big] \\ &= \exp\Big[-\frac{pn}{2m} + O\Big(\frac{p}{m}\Big) - O\Big(\frac{p^2 n}{m^2}\Big)\Big] \qquad (n \to \infty). \end{aligned}$$

To obtain the last equality we have used the Taylor series approximation

$$\log(1 - x) = -x - O(x^2) \qquad (x \to 0)$$

and recalled that $N = n - 1$. The condition on $p$ and $m$ completes the proof. □

The probability, $\mathcal{P}_e$, that one or more components of any of the memories is not stable can be readily estimated by an application of the union bound and (4):

$$\mathcal{P}_e \leq nmP\{X \leq 0\} \leq nm\big[pe^{-1/2(m-1)} + q\big]^{n-1}.$$

Note that the bound of (4) holds for all choices of $p$, $m$, and $n$, so that the above estimate for $\mathcal{P}_e$ also holds unrestricted. It is clear that the upper bound for $\mathcal{P}_e$ increases monotonically as $m$ increases, so it suffices to prove the theorem with inequality replaced by equality in (2) and (3). Now, with $m$ chosen as in (2) the condition on $p$ and $m$ in Lemma 1 is satisfied. Hence, for this choice of $m$

$$\mathcal{P}_e \leq [1 + o(1)]\, nm \exp\Big(-\frac{pn}{2m}\Big) \leq \epsilon + o(1) \qquad (n \to \infty).$$

This establishes part a) of the theorem.

In similar fashion we can establish the second part of the theorem by noting that the probability that a given memory is not a fixed point is bounded from above by $nP\{X \leq 0\}$ by the union bound. For a choice of $m$ according to (3) this probability is bounded above by $\epsilon + o(1)$. Part b) of the theorem follows as the expected number of memories that are not fixed points is just $m$ times the probability that one memory is not fixed.

## III. Error Correction

Let us now investigate how sparsity in the model affects the ability of the system to retrieve fundamental memories from probes which are "noisy" versions of the memories. The particular model of error correction that we will investigate is the ability of the (sparse) network to correct *random errors* in the memories in *one synchronous step*. As we will see, the moment inequality technique of the previous section still serves to analyse this situation, albeit at the cost of some additional complexity.

Let $0 \leq \rho < 1/2$ be fixed. Corresponding to each memory, $\boldsymbol{u}^\alpha$, we generate a random probe, $\hat{\boldsymbol{u}}^\alpha \in \mathbb{B}^n$, by independently specifying the components, $\hat{u}_j^\alpha$, of the probe as follows:

$$\hat{u}_j^\alpha = \begin{cases} u_j^\alpha, & \text{with probability } 1 - \rho, \\ -u_j^\alpha, & \text{with probability } \rho. \end{cases} \qquad (5)$$

Note that the expected number of errors in the probe (i.e., the expected number of components of the probe, $\hat{u}^\alpha$, which are not equal to the corresponding components of the memory, $u^\alpha$) is $\rho n$.

*Definition 1:* We say that a memory, $u^\alpha$, *dominates over a radius* $\rho n$ if, with probability approaching one as $n \to \infty$, the network corrects all the errors in a random probe generated according to prescription (5) in one synchronous step. We call $\rho$ the *(fractional) radius of dominance* of the memory.

*Remarks:* An application of Lemma A.2 in the appendix yields that for any $\delta > 0$ there is a large enough constant $C$ such that with probability $1 - \delta$ the number of errors in the probe lies between $\rho n - C\sqrt{n}$ and $\rho n + C\sqrt{n}$. Hence, a memory that dominates over a radius $\rho n$ corrects random errors in essentially $\rho n$ components with high probability.

An alternative—and perhaps more appealing—model for generating random probes is to choose the probe at random from the Hamming ball of radius $\rho n$ surrounding the memory. The notion of a radius of dominance for the memory is intuitively and geometrically much clearer for this model. However, by the sphere hardening effect, almost all probes generated in this model are concentrated at the surface of the Hamming ball so that the number of errors is again essentially $\rho n$. The analytical results that derive for this model are formally indistinguishable from the model we have adopted in (5). The present format is, however, slightly more convenient mathematically.

We will prove the following theorem which is our main result of this section.

*Theorem 2:* Let $0 \le \rho < 1/2$ be any desired radius of dominance, and let the sparsity parameter, $p$, satisfy $p = \Omega(\log^\gamma n/n)$ for some fixed $\gamma > 3$. For any $\epsilon > 0$, we then have the following.

a) If, as $n \to \infty$, we choose the number of memories, $m$, such that

$$m \le \frac{(1-2\rho)^2 pn}{2\log pn^2}\left[1 + \frac{\log\log pn^2 + \log 2\epsilon/(1-2\rho)^2}{\log pn^2} - O_\epsilon\left(\frac{\log\log pn^2}{\log^2 pn^2}\right)\right], \quad (6)$$

then the probability that all $m$ memories dominate over a radius $\rho n$ is at least as large as $1 - \epsilon - o(1)$.

b) If, as $n \to \infty$, we choose the number of memories, $m$, such that

$$m \le \frac{(1-2\rho)^2 pn}{2\log n}\left[1 + \frac{\log \epsilon}{\log n} + O_\epsilon\left(\frac{1}{(\log n)^2}\right)\right], \quad (7)$$

then the expected number of memories that dominate over a radius $\rho n$ is at least as large as $[1 - \epsilon - o(1)]m$.

*Remarks:* We can store at least $(1-2\rho)^2 pn/2\log pn^2$ memories *all* of which dominate over a radius $\rho n$, and at least $(1 - 2\rho)^2 pn/2\log n$ memories *most* of which dominate over a radius $\rho n$. These lower estimates of capacity are also tight from above. This can be demonstrated extending the technique used by McEliece, *et al.* [2]. The proof, as in the original, is long and replete with technical details. We will not go into it here.

*Proof:* We will first estimate the probability that a single component of a memory is retrieved from a random probe. The use of the union bound, as before, will then complete the proof of the theorem.

Let us form the random sums

$$\hat{X}_i^\alpha = u_i^\alpha \sum_{j=1}^{n} w_{ij}\hat{u}_j^\alpha, \qquad i = 1,\cdots,n, \qquad \alpha = 1,\cdots,m. \quad (8)$$

If random errors are to be corrected in one synchronous step for each memory we will require that $\hat{X}_i^\alpha > 0$ for each $i \in [n]$ and $\alpha \in [m]$ with high probability. Let us first estimate the probability that a particular component of a memory is not retrieved in one synchronous step from a random probe. We again hold $i$ and $\alpha$ fixed and suppress the dependence of variables on these indices except where required for clarity.

Substituting for the weights, $w_{ij}$, from (1) in (8) we have

$$\hat{X} = \sum_{j\ne i}\pi_{ij}\sum_{\beta=1}^{m} u_i^\alpha \hat{u}_j^\alpha u_i^\beta u_j^\beta = \hat{Y} + \sum_{j \ne i}\pi_{ij}\sum_{\beta\ne\alpha}\hat{Z}_j^\beta, \quad (9)$$

where we define

$$\hat{Z}_j^\beta = u_i^\alpha \hat{u}_j^\alpha u_i^\beta u_j^\beta, \qquad j \ne i, \qquad \beta = 1,\cdots,m,$$

and

$$\hat{Y} = \sum_{j\ne i}\pi_{ij}\hat{Z}_j^\alpha = \sum_{j\ne i}\pi_{ij}u_j^\alpha \hat{u}_j^\alpha. \quad (10)$$

We are interested in estimating the probability that $\hat{X} \le 0$, i.e., the probability that the $i$th component of memory $u^\alpha$ is not retrieved from the random probe $\hat{u}^\alpha$ in one synchronous step. The following is the central result.

*Lemma 2:* Let $0 \le \rho < 1/2$ be any desired fractional radius of dominance, and let $\tau$ be a fixed parameter with $2/3 < \tau < 1$. If, as $n \to \infty$, the sparsity parameter, $p$, and the number of memories, $m$, vary such that $pn \to \infty$ and $m = \Omega((pn)^\tau)$ then

$$P\{\hat{X} \le 0\} \le [1 + o(1)]\exp\left(-\frac{(1-2\rho)^2 pn}{2m}\right) \qquad (n \to \infty). \quad (11)$$

*Proof:* The demonstration is in three parts. We first show that the sum over the index $j$ in (9) can be formally replaced by a sum over essentially $pn$ indices; we next show that the random variable $\hat{Y}$ can be formally replaced by the fixed value $(1 - 2\rho)pn$; we finally invoke the inequality involving the moment generating function described in the previous section to complete the proof.

Let $J \subseteq [n] \setminus \{i\}$ be the random subset of indices defined by

$$J = \{j: \pi_{ij} = 1\}.$$

We then have

$$\hat{X} = \hat{Y} + \sum_{j\in J}\sum_{\beta\ne\alpha}\hat{Z}_j^\beta. \quad (12)$$

Let the random variable $\Lambda = |J|$ denote the cardinality of $J$. Clearly, $\Lambda = \sum_{j\ne i}\pi_{ij}$. It follows that

$$E(\Lambda) = pN,$$

where we set $N = n - 1$ as before. Let $\delta$ be chosen such that $(1-\tau)/2 < \delta < 1/6$. An application of Lemma A2 yields

$$P\{|\Lambda - pN| > (pN)^{1/2+\delta}\} = O(e^{-c_1(pN)^{2\delta}}). \quad (13)$$

Now, from (10) we have

$$\hat{Y} = \sum_{j\in J}u_j^\alpha \hat{u}_j^\alpha.$$

By independence of the components of the memories, the expectation of $\hat{Y}$ conditioned upon a sample realisation of the random set of indices $J$ depends only on the cardinality, $\Lambda$, of $J$. Hence,

$$E\hat{Y} = \sum_{k=0}^{N} E(\hat{Y} \mid \Lambda = k) P\{\Lambda = k\}$$

$$= \sum_{k=0}^{N} k(1-2\rho)\binom{N}{k}(1-p)^k p^{N-k} = (1-2\rho)pN.$$

Using (13) and the large deviation Lemma A2 hence yields

$$P\{|\hat{Y} - (1-2\rho)pN|> (pN)^{1/2+\delta}\} = O(e^{-c_2(pN)^{2\delta}}). \quad (14)$$

Let $\mathcal{S}$ be the set of sample points over which the following inequalities hold jointly:

$$|\Lambda - pN| \le (pN)^{1/2+\delta},$$

$$|\hat{Y} - (1-2\rho)pN| \le (pN)^{1/2+\delta}.$$

From (13) and (14), we then have

$$P\{\mathcal{S}\} = 1 - O(e^{-c_3(pN)^{2\delta}}). \quad (15)$$

We say that an assignment of values to $\Lambda$ and $\hat{Y}$ is *allowable* if they occur in $\mathcal{S}$. A subset of indices from $[n] \setminus \{i\}$ is allowable if the number of indices in the set is allowable.

Let us now return to a consideration of (12). Using (15) we have from elementary considerations that

$$P\{\hat{X} \le 0\} = P\left\{\sum_{j\in J}\sum_{\beta\ne\alpha} \hat{Z}_j^\beta \le -\hat{Y}\right\}$$

$$= P\left\{\sum_{j\in J}\sum_{\beta\ne\alpha} \hat{Z}_j^\beta \le -\hat{Y} \,\middle|\, \mathcal{S}\right\} + O(e^{-c_3(pN)^{2\delta}}). \quad (16)$$

Let $J' \subseteq [n] \setminus \{i\}$ be any subset of indices, and let $\lambda = |J'|$. For positive $\lambda$ and $y$ define

$$f(\lambda, y) \triangleq P\left\{\sum_{j\in J'}\sum_{\beta\ne\alpha} \hat{Z}_j^\beta \le -y\right\}.$$

Applying Lemma A1 as in the last section, we have

$$f(\lambda, y) \le \inf_{r\ge 0} e^{-ry} E(e^{-r\sum_{j\in J'}\sum_{\beta\ne\alpha}\hat{Z}_j^\beta}) \le e^{-y^2/2\lambda M}.$$

Now consider a choice of $\lambda = pN \pm O((pN)^{1/2+\delta})$ and $y = (1-2\rho)pN \pm O((pN)^{1/2+\delta})$. Recalling that $N = n-1$, $M = m-1$, and that from the statement of the lemma $pn \to \infty$ and $m = \Omega((pn)^\tau) \to \infty$ for $2/3 < \tau < 1$, we have

$$f(\lambda, y) \le \exp\left\{-\frac{(1-2\rho)^2 pN}{2M} + O\left(\frac{(pN)^{1/2+\delta}}{M}\right)\right\}$$

$$= \left[1 + O\left(\frac{(pn)^{1/2+\delta}}{m}\right)\right]\exp\left\{-\frac{(1-2\rho)^2 pn}{2m}\right\}. \quad (17)$$

The last equality follows from the choice $(1-\tau)/2 < \delta < 1/6$; this yields $1/2 + \delta < 2/3 < \tau$ so that by choice of $m = \Omega((pn)^\tau)$ we have $(pn)^{1/2+\delta} = o(m)$.

Returning to (16) we note that the random variables $\hat{Z}_j^\beta$ are independent of the random variable $\hat{Y}$ and the random subsets $J$. Hence, we have

$$P\{\hat{X} \le 0\} = \sum_{\text{allowable } y,\, J'} P\left\{\sum_{j\in J'}\sum_{\beta\ne\alpha}\hat{Z}_j^\beta \le -y \,\middle|\, \hat{Y} = y,\right.$$

$$\left. J = J', \mathcal{S}\right\} P\{\hat{Y} = y, J = J' \mid \mathcal{S}\}$$

$$+ O(e^{-c_3(pn)^{2\delta}})$$

$$= \sum_{\text{allowable } y,\, J'} P\left\{\sum_{j\in J'}\sum_{\beta\ne\alpha}\hat{Z}_j^\beta \le -y\right\}$$

$$\cdot P\{\hat{Y} = y, J = J' \mid \mathcal{S}\} + O(e^{-c_3(pn)^{2\delta}})$$

$$= \sum_{\text{allowable } y,\, J'} f(\lambda, y) P\{\hat{Y} = y, J = J' \mid \mathcal{S}\}$$

$$+ O(e^{-c_3(pn)^{2\delta}}),$$

where $\lambda = |J'|$. For allowable $\lambda$ and $y$, however, we have

$$|\lambda - pN| = O((pN)^{1/2+\delta}),$$

$$|y - (1-2\rho)pN| = O((pN)^{1/2+\delta}),$$

by definition. The bound (17), hence, holds for every term, $f(\lambda, y)$, in the sum above. It follows that

$$P\{\hat{X} \le 0\} \le \left[1 + O\left(\frac{(pn)^{1/2+\delta}}{m}\right)\right]$$

$$\cdot \exp\left\{-\frac{(1-2\rho)^2 pn}{2m}\right\} + O(e^{-c_3(pn)^{2\delta}}).$$

The exponent $(pn)^{2\delta}$ dominates $pn/m$ as $m = \Omega((pn)^\tau)$ and $2\delta > 1 - \tau$. Further, $(pn)^{1/2+\delta}/m = o(1)$ as $1/2 + \delta < \tau$. The statement of the lemma follows.

As before, the probability that one or more memory components is not retrieved increases monotonically as $m$ increases, so it suffices to show that the theorem holds with $m$ given by equality in (6) and (7). Now let $\gamma > 3$ be as in the statement of the theorem, and set $\tau = 1 - 1/\gamma$. A choice of a number of memories according to (6) or (7) satisfies the requirements of Lemma 2, so that the asymptotic bound of (11) holds for the probability that a single memory component is not retrieved from a random probe. The theorem is now proved using the union bound as in the last section. □

## IV. Conclusion

The results of this correspondence and those of Komlós and Paturi [6] imply that the folk theorem on robustness is well founded in situations where there is a distributed storage of information in the network. In such instances the neural network would appear to be relatively resilient to the loss or damage of interconnection weights. For the outer-product algorithm, in particular, each neuron needs to retain only of the order of $\Omega(\log n)$ interconnection weights out of a total of $n$ possible links with other neurons for useful associative properties to emerge. These results also appear to generalize to other, more complex situations, and this is under investigation.

In an evocative alternate line of thought we could consider situations where the devil in the network is not malicious but is actively well disposed towards producing useful sparse structures. The issue here is whether we can exploit carefully designed sparsity to design codes (families of allowed subsets of memories) which have high storage capacities. Specifically, we would like to store large numbers of memories (high capacity) where the allowed sets of fundamental memories that can be picked is specified by a (large) code. The intuition here is that large gains in storage capacity may be obtained by excluding certain pathological sets of memories from consideration in the code, and that such resulting codes may be designed to fit suitable sparse architectures. We provide an illustration of the gains that are possible in a succeeding paper [7].

## Appendix A

### Large Deviations

We quote the following technical lemmas without proof. Lemma A1 is a generalisation of the classical Chebyshev inequality and provides a large deviation estimate in terms of generating functions. Lemma A2 is a straightforward generalisation of a classical large deviation central limit theorem for sums of binary random variables which provides good uniform estimates for the probability that the sum has a large deviation from the mean. (The corresponding version of the result for indicator random variables (taking values 0 and 1 only) can be found, for instance, in Feller's text [8].)

*Lemma A1:* Let $X$ be a random variable and $x \geq 0$ any nonnegative number. Then

$$P\{X \leq -x\} \leq \inf_{r \geq 0} e^{-rx} E(e^{-rX}).$$

*Lemma A2:* Let $x_1 < x_2$ be any two real numbers and let $\{\zeta_j\}$ be a sequence of i.i.d. random variables drawn from a sequence of Bernoulli trials with

$$\zeta_j = \begin{cases} x_1, & \text{with probability } q = 1 - p, \\ x_2, & \text{with proability } p, \end{cases}$$

where $0 < p < 1$. For each $K$ let $S_K = \sum_{j=1}^{K} \zeta_j$. If as $K \to \infty$ the real number $v$ varies such that $v/\sqrt{K} \to \infty$ and

$$v = \begin{cases} o(K^{2/3}), & \text{if } p \neq q, \\ o(K^{3/4}), & \text{if } p = q = 1/2, \end{cases}$$

then

$$P\{|S_K - K(px_2 + qx_1)| > v(x_2 - x_1)\} \sim \frac{\sqrt{2pqK}\, e^{-v^2/2pqK}}{\sqrt{\pi}\, v}.$$

## References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational properties," *Proc. Nat. Acad. Sci.*, vol. 79, 1982, pp. 2554-2558.

[2] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. 33, pp. 461-482, July 1987.

[3] J. Komlós and R. Paturi, "Convergence results in an associative memory model," *Neural Networks*, vol. 1, no. 3, pp. 239-250, 1988.

[4] C. Newman, "Memory capacity in neural network models: rigourous lower bounds," *Neural Networks*, vol. 1, no. 3, pp. 223-238, 1988.

[5] D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Storing infinite numbers of patterns in a spin-glass model of neural networks," *Phys. Rev. Lett.*, vol. 55, pp. 1530-1533, 1985.

[6] J. Komlós and R. Paturi, "Effect of connectivity in associative memory models," tech. Rep. CS88-131, Univ. of California, San Diego, 1988.

[7] S. Biswas and S. S. Venkatesh, "Codes, sparsity, and capacity in neural associative memory," submitted for publication.

[8] W. Feller, *An Introduction to Probability Theory and its Applications*, vol. 1, 3rd ed. New York: Wiley, 1968.

# Balanced Codes and Nonequiprobable Signaling

A. R. Calderbank, *Member, IEEE*, and M. Klimesh

*Abstract*—**The problem of shaping signal constellations that are designed for the Gaussian channel is considered. The signal constellation consists of all points from some translate of a lattice $\Lambda$, that lie within a region $\mathscr{R}$. The signal constellation is partitioned into $T$ annular subconstellations $\Omega_0, \cdots, \Omega_{T-1}$ by scaling the region $\mathscr{R}$. Signal points in the same subconstellation are used equiprobably, and a shaping code selects region $\Omega_i$ with frequency $f_i$. If the signal constellation is partitioned into annular subconstellations of unequal size, then absent some cleverness, the transmission rate will vary with the choice of codeword in the shaping code, and it will be necessary to queue the data in buffers. It is described how balanced binary codes constructed by Knuth can be used to avoid a data rate that is probabilistic. The basic idea is that if symbols *0* and *1* *represent constellations of unequal size*, and if all shaping codewords have equally many 0's and 1's, then the data rate will be deterministic.**

*Index Terms*—**Bandwidth efficient communication, shaping codes, nonequiprobable signaling.**

## I. Introduction

We start with a basic region $\mathscr{R}$ in $\mathbb{R}^N$, and by scaling we obtain a nested sequence $\mathscr{R} = \alpha_0 \mathscr{R}, \alpha_1 \mathscr{R}, \cdots, \alpha_{T-1} \mathscr{R}$ of copies of $\mathscr{R}$. Let $\Omega$ be the signal constellation comprising all points from (some fixed translate of) a lattice $\Lambda$ that lie within the region $\alpha_{T-1} \mathscr{R}$. Then $\Omega_0 = \Lambda \cap \mathscr{R}$ and $\Omega_i = \Lambda \cap (\alpha_i \mathscr{R} \setminus \alpha_{i-1} \mathscr{R})$, $i = 1, \cdots, T - 1$, give a partition of $\Omega$ into annular subconstellations with increasing average power.

The reason we consider signal constellations drawn from lattices is that signal points are distributed regularly throughout $N$-dimensional space. If signals are equiprobable, then the average signal power $P_0$ of the constellation $\Omega_0$ is approximately the average power $P(\mathscr{R})$ of a probability distribution that is uniform within $\mathscr{R}$ and zero elsewhere; thus

$$P_0 \approx P(\mathscr{R}) = \frac{1}{NV(\mathscr{R})} \int_{\mathscr{R}} \|x\|^2 \, dv, \tag{1}$$

where

$$V(\mathscr{R}) = \int_{\mathscr{R}} dv$$

is the volume of the region $\mathscr{R}$. We rewrite (1) as

$$P_0 \approx G(\mathscr{R}) V(\mathscr{R})^{2/N}, \tag{2}$$

where

$$G(\mathscr{R}) = \frac{\int_{\mathscr{R}} \|x\|^2 \, dv}{NV(\mathscr{R})^{1+2/N}} \tag{3}$$

is the normalized or dimensionless second moment. Since $G(\mathscr{R})$ is dimensionless, it is not changed by scaling the region $\mathscr{R}$. It measures the effect of the shape of the region $\mathscr{R}$ on average signal power.

Manuscript received April 18, 1991; revised October 10, 1991.

A. R. Calderbank is with AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974.

M. Klimesh is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 9105782.

0018-9448/92$03.00 © 1992 IEEE

To appear:
IEEE Trans. Information
Theory

# CODES, SPARSITY, AND CAPACITY IN NEURAL ASSOCIATIVE MEMORY

Sanjay Biswas and Santosh S. Venkatesh[*]
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104

1 October 1990 INDEX TERMS: *Neural Networks, Interconnection Graphs, Codes, Capacity, Sparsity, Associative Memory.*

**Abstract**

Neural associative memories viewed as a coding system have been subject to the criticism that the codes have very low rates. In a fully-interconnected network of $n$ neurons, for instance, the outer-product algorithm has a storage capacity of the order of $n/\log n$ memories: specifically, almost all choices of $n/4\log n$ memories are stored as fixed points by the outer-product algorithm when the interconnectivity graph has degree $n$. In this communication it is shown that storage capacities, $C$—the maximum number of memories that can be stored—can be improved substantially at the expense of the size of the code—the family of *admissible* $C$-sets of memories. In particular, a relation between code size, capacity, and degree of the interconnectivity graph is shown: *for any* $\tau < 1$, *a constant* $c > 0$ *and a code of size* $\Omega\left(2^{cn^{2-\tau}}\right)$ *can be found such that it is possible to achieve memory storage capacities which are* $\Omega\left(2^{n^{\tau}}\right)$ *in recurrent neural networks with interconnectivity graph of degree* $\Theta\left(n^{1-\tau}\right)$. Thus, near-exponential capacities can be obtained for codes which are still exponential in size. An interesting and useful side effect of the constructions employed in this paper is that large capacities can be obtained in very sparsely interconnected structures for suitably chosen codes.

# 1 INTRODUCTION

A folk theorem in neural computation asserts that dense interconnectivity is a *sine qua non* for efficient usage of resources, and, in particular, that sparser structures exhibit a degradation in computational capability. This communication represents an effort to formally examine this tenet in the context of neural associative memory and a recurrent neural network structure. An appreciation of the results may be best produced in terms of a coding theoretic analogue. The memories to be stored can be thought of as *codewords* with the neural network being the *decoder* which corrects errors in memories. Any set of memories to be stored is chosen from a collection of *admissible* sets of memories which forms the *code*. In applications hitherto the code has typically been the set of *all* subsets of binary vectors—the power set of $\{-1,1\}^n$.[*] An *algorithm*

[*] The support of research grants from E. I. DuPont de Nemours, Inc. and the Air Force Office of Scientific Research (AFOSR 89–0523) is gratefully acknowledged.

[*] There have been some explorations, however, of *sparse encoding*, which has particular significance when the vector representation of the codewords (or memories) is in terms of 1's and 0's instead of 1's and -1's. In such cases the codewords (or memories) are typically chosen such that the number of components taking value 1 is small compared to the number of components taking value 0. In terms of electrical circuit realisations of such networks, each memory can be represented by relatively few active electrical lines. The code corresponding to such a sparse encoding is clearly much more dilute.

for neural associative memory is a procedure which, given an admissible set of memories from a given code, produces a network which stores the memories (typically as fixed points). For a given code the two main parameters characterising the efficacy of an algorithm are the *capacity*—the largest admissible set of memories that can be stored by the algorithm—and the *attraction radius*—the number of component errors in any memory that can be corrected by the network. With a code consisting of all subsets of binary $n$-vectors, the outer-product algorithm, for instance, has a capacity of the order of $n/\log n$ memories correcting a linear number of random component errors [1, 2].

In the general case the network may not be fully-interconnected, but may have connections specified by an interconnectivity graph which may be rather sparse. If we still insist that almost all choices of memories within capacity be stored—i.e., the code consists of all subsets of $\{-1,1\}^n$—then the capacity of an algorithm inevitably decreases as the degree of sparsity increases.[†] In an evocative alternate line of thought, however, we might consider designing codes which are proper subsets of $\{-1,1\}^n$ to take advantage of a given sparse interconnectivity structure. This is the situation examined in this paper. The main result that emerges from our investigations is the following: *for any $\tau < 1$ we can achieve memory storage capacities which are $\Omega\left(2^{n^\tau}\right)$ in recurrent neural networks with interconnectivity graph of degree $\Theta\left(n^{1-\tau}\right)$ for carefully chosen codes of log-size $\Theta\left(n^{2-\tau}\right)$.* (In other words, we can find an exponential number of choices of $2^{n^\tau}$ memories that can be stored as attractors in suitably chosen sparse networks.) The trade-off here is in increased capacity at the expense of code size. Interestingly, increased sparsity in the interconnectivity graph can increase storage capacity—for code choices which, as it will turn out, are exponential in size as long as the interconnectivity graph has degree $\Omega(\log n)$. The design methodology here lies in the choice of the code (subsets of admissible memories) as a function of the network sparsity, and the selection of an algorithm to specify the strengths of the interconnections for the given interconnectivity graph as a function of the memories to be stored. Note that unlike the situation in the fully-interconnected case (the interconnectivity graph having degree $n$), all possible choices of $m = \Omega\left(2^{n^\tau}\right)$ memories cannot be stored in the sparse network. Rather, the memories to be stored must be taken from the set of admissible memories which form the code.

In the next section we briefly review the neural model and formally introduce the notions of codes and capacity in the context of neural associative memory. In section 3 we introduce a simple model of block sparsity where the neurons are partitioned into mutually non-communicating sets. For this structure we demonstrate a code which is sufficiently rich while yielding large capacities for the classical outer-product algorithm. In section 4 we show how the results for the simple block sparsity model extend to other sparse structures, and in particular, the nested model introduced by Baram [5]. We present a generalisation of a spectral based algorithm for the block sparsity model in section 5, and show concomitant increases in capacity. Theorems are stated in the body of the paper while their proofs and relevant technical lemmas are developed in sequence in the appendices.

**Notation** We employ usual asymptotic notation, and introduce two (non-standard) notations: if $\{x_n\}$ and $\{y_n\}$ are positive sequences, we say that

- $x_n = \Omega(y_n)$ if there exists $K$ such that $x_n/y_n \geq K$ for all $n$;
- $x_n = O(y_n)$ if there exists $L$ such that $x_n/y_n \leq L$ for all $n$;
- $x_n = \Theta(y_n)$ if $x_n = \Omega(y_n)$ and $x_n = O(y_n)$;
- $x_n \sim y_n$ if $x_n/y_n \to 1$ as $n \to \infty$; we also say that $x_n \gtrsim y_n$ if $x_n \geq y_n$ for $n$ large enough, and $x_n \lesssim y_n$ if $x_n \leq y_n$ for $n$ large enough;
- $x_n = o(y_n)$ if $x_n/y_n \to 0$ as $n \to \infty$.

We denote by $\mathbb{B}$ the set $\{-1,1\}$, and by $[n]$ the set of integers $\{1,2,\ldots,n\}$. By ordered multiset we mean an ordered collection of elements with repetition of elements allowed. We will use the terminology $m$-set and ordered multiset of size $m$ interchangeably. All logarithms in the exposition are to base $e$.

[†] The decrease in capacity with increased sparsity is not catastrophic, however, and network performance as an associative memory degrades gracefully [3, 4].

# 2 ASSOCIATIVE MEMORY

## 2.1 Recurrent Neural Networks

A neuron (after McCulloch and Pitts [6]) is formally a linear threshold element characterised by $n$ real weights, $\mathbf{w} \in \mathbb{R}^n$, which, in response to a vector of $n$ (real or binary) inputs, $\mathbf{u}$, produces a binary output, $v \in \mathbb{B}$, as the sign of the weighted sum $\sum_{j=1}^{n} w_j u_j$.[1]

We consider a network of $n$ formal neurons. The state of the network at any epoch is the $n$-vector, $\mathbf{u} \in \mathbb{B}^n$, of neural outputs at that epoch. Neural outputs at each epoch are fed back and constitute the inputs to each neuron at the next update epoch. The allowed pattern of neural interconnectivity is specified by the edges of a *(bipartite) interconnectivity graph*, $G_n$, on vertices, $[n] \times [n]$. In particular, the existence of an edge $\{i, j\}$ in $G_n$ is indica  aat the output of the $j$-th neuron is fed back as input to the $i$-th neuron. The network is characterised by an $n \times n$ matrix of weights, $\mathbf{W} = [w_{ij}]$, where $w_{ij}$ denotes the (real) weight linking the output of neuron $j$ to the input of neuron $i$. (We adopt the convention that a weight, $w_{ij}$, is zero if $\{i, j\} \notin G_n$.) If $\mathbf{u} \in \mathbb{B}^n$ is the current state of the system, an update, $u_i \mapsto u_i'$ of the state of the $i$-th neuron is specified by the linear threshold rule

$$u_i' = \operatorname{sgn}\left(\sum_{j:\{i,j\}\in G_n} w_{ij} u_j\right).$$

The two extreme modes of neural updates are synchronous, with every neuron being updated in concert, and asynchronous, with at most one neuron being updated at any instant. Mixed modes of operation between the two extremes are, of course, feasible. For any mode of operation the network dynamics describe trajectories in a state space comprised of the vertices of the $n$-cube.

The utility of this network model as an associative memory hinges upon the observation that under suitable symmetry conditions there are Lyapunov functions for the system [7, 8]. In particular, for each state $\mathbf{u} \in \mathbb{B}^n$ define the *energy function*, $E(\mathbf{u})$, as the quadratic form

$$E(\mathbf{u}) = -\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} u_i u_j = -(\mathbf{u}, \mathbf{W}\mathbf{u}).$$

If $\mathbf{W}$ is symmetric, non-negative definite, then the function $E$ is non-increasing along any trajectory in *any* mode of operation [9].

We can, hence, think in terms of an "energy landscape" with states embedded in it. Trajectories in this landscape tend to go "downhill." States which form local "energy" minima, hence, determine system dynamics; each such state possesses a *basin of attraction* comprised of neighbouring states of higher "energy" which are mapped into the state at the local minimum. This geometric picture is particularly persuasive for an associative memory application where we wish to store a desired set of states—the *memories*—as fixed points of the network, and with the property that errors in an input representation of a memory are corrected and the memory retrieved. The challenge here is to choose a matrix of weights such that the desired memories are located at energy minima.

Let $\mathbf{u} \in \mathbb{B}^n$ be a memory and $0 \leq \rho < 1$ a parameter. Corresponding to the memory $\mathbf{u}$ we generate a probe $\hat{\mathbf{u}} \in \mathbb{B}^n$ by independently specifying the components, $\hat{u}_j$, of the probe as follows:

$$\hat{u}_j = \begin{cases} u_j & \text{with probability } 1-\rho \\ -u_j & \text{with probability } \rho. \end{cases} \tag{1}$$

We call $\hat{\mathbf{u}}$ a *random probe with parameter* $\rho$.

**Definition 2.1** We say that a memory, $\mathbf{u}$, is a *monotone $\rho$-attractor* if, with probability approaching one as $n \to \infty$, the network corrects all errors in a random probe with parameter $\rho$ in one synchronous step. We call $\rho$ the *(fractional) attraction radius*. We also say that $\mathbf{u}$ is *stable* if it is a monotone 0-attractor.

[1] The model allows for a real *threshold* as well, but this will not be important to our discussion. We will throughout assume a zero threshold for each neuron. We also adopt the convention $\operatorname{sgn} 0 = 1$.

REMARKS: Note that stable memories are just fixed poir.ts of the network. Also, by the Borel strong law, the fraction of the number of components in the probe which are in error (i.e., not equal to the corresponding components of the memory) is concentrated at the expected value $\rho$.[§]

For $m \geq 1$ let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of memories to be stored. The ***outer-product algorithm*** specifies the interconnection weights, $w_{ij}$, according to the following rule: for $i \in [n]$, $\{i,j\} \in G_n$,

$$w_{ij} = \sum_{\beta=1}^{m} u_i^\beta u_j^\beta. \tag{2}$$

In the fully-interconnected situation, for instance, $\mathbf{W}$ is symmetric, non-negative definite so that suitable associative properties result. In general, if the interconnectivity graph, $G_n$, is symmetric then, under a suitable mode of operation, there is a Lyapunov function for the network specified by the outer-product algorithm.

## 2.2 Codes and Capacity

For given integers $m \geq 1$, $n \geq 1$, a ***code***, $\mathcal{K}_n^m$, is a collection of ordered multisets of size $m$ from $\mathbb{B}^n$. We say that an $m$-set of memories is ***admissible*** iff it is in $\mathcal{K}_n^m$.[¶] Thus, a code just specifies which $m$-sets are allowable as memories. Henceforth when we refer to a memory we mean a binary $n$-tuple in some admissible set from a code $\mathcal{K}_n^m$. Examples of codes include: the set of all ordered multisets of size $m$ from $\mathbb{B}^n$; a single multiset of size $m$ from $\mathbb{B}^n$; all collections of $m$ mutually orthogonal vectors in $\mathbb{B}^n$; all $m$-sets of vectors in $\mathbb{B}^n$ in general position.

Clearly, if all memories in an admissible $m$-set of memories are stable (or are monotone $\rho$-attractors), then so are the $m!$ ordered multisets generated by all permutations of the original $m$-set. (For $m$ linear in $n$, for instance, the number of permutations is of the order of $2^{cn \log n}$ for some constant $c$.) We hence need to guard against defining trivial codes generated by permutations of a few basic ordered multisets of memories. Define two ordered multisets of memories to be ***equivalent*** if they are permutations of one another. We define the ***size*** of a code, $|\mathcal{K}_n^m|$, to be the number of distinct equivalence classes of $m$-sets of memories. We will be interested in codes of relatively large size: $\log|\mathcal{K}_n^m|/n \to \infty$ as $n \to \infty$. In particular, we require at least an exponential number of choices of (equivalence classes of) admissible $m$-sets of memories. For a given code, $\mathcal{K}_n^m$, we confer a probability distribution on memories by choosing an $m$-set of memories from the uniform distribution on $\mathcal{K}_n^m$.

For each fixed $n$ and interconnectivity graph, $G_n$, an ***algorithm***, $\mathcal{X}$, is a prescription which, given an $m$-set of memories, produces a corresponding set of interconnection weights, $w_{ij}$, $i \in [n]$, $\{i,j\} \in G_n$. Let $\mathcal{K}_n^m$, $m \geq 1$, $n \geq 1$ be a doubly-indexed sequence of codes, and let $\mathcal{X}$ be an algorithm (corresponding to an underlying interconnectivity graph, $G_n$). For $m \geq 1$ let $\mathcal{A}(\mathbf{u}^1, \ldots, \mathbf{u}^m)$ be some attribute of $m$-sets of memories. (The following, for instance, are examples of attributes of admissible sets of memories: all the memories are stable in the network generated by $\mathcal{X}$; almost all the memories are monotone $\rho$-attractors.) For given $n$ and $m$, we choose a random $m$-set of memories, $\mathbf{u}^1, \ldots, \mathbf{u}^m$, from the uniform distribution on $\mathcal{K}_n^m$.

**Definition 2.2** A sequence, $\{C_n\}_{n=1}^{\infty}$, is a ***capacity function for the attribute*** $\mathcal{A}$ (or $\mathcal{A}$-***capacity*** for short) if for $\lambda > 0$ arbitrarily small:

[§] An alternative—and perhaps more appealing—model for generating random probes is to choose the probe at random from the Hamming ball of radius $\rho n$ at $\mathbf{u}$. The notion of a radius of attraction is intuitively and geometrically much clearer for this model. However, by the sphere hardening effect, almost all probes generated in this model are concentrated at the surface of the Hamming ball surrounding the memory, so that the number of errors is again essentially $\rho n$. The analytical capacity results that derive for this model are formally indistinguishable from the model we have adopted in equation (1), though the technical details are somewhat different. The present format is, however, slightly more convenient mathematically.

[¶] We define admissible $m$-sets of memories in terms of ordered multisets rather than sets so as to obviate certain technical nuisances.

a) $\mathrm{P}\{\mathcal{A}(\mathbf{u}^1, \dots, \mathbf{u}^m)\} \to 1$ as $n \to \infty$ whenever $m \le (1-\lambda)C_n$;

b) $\mathrm{P}\{\mathcal{A}(\mathbf{u}^1, \dots, \mathbf{u}^m)\} \to 0$ as $n \to \infty$ whenever $m \ge (1+\lambda)C_n$.

We also say that $C_n$ is a *lower $\mathcal{A}$-capacity* if property (a) holds, and that $C_n$ is an *upper $\mathcal{A}$-capacity* if property (b) holds.

REMARK: The capacity function implicitly depends upon the sequence of interconnectivity graphs, $G_n$, the sequence of codes, $\mathcal{K}_n^m$, and the algorithm, $\mathcal{X}$, as well as on the desired attribute, $\mathcal{A}$, of the memories.

## 3 BLOCK SPARSITY

Sparse interconnectivity graphs are of importance in practical realisations. Besides the obvious advantages in programming when there are relatively few weights, cost considerations strongly favour sparse interconnectivity as interconnections dominate the silicon real estate in hardware realisations of these networks. One of the simplest forms of sparsity we might enjoin is block sparsity where the neurons are partitioned into disjoint subsets of neurons with full-interconnectivity within each subset and no neural interconnections between subsets. The weight matrix in this case takes on a block diagonal form, and the interconnectivity graph is composed of a set of disjoint complete bipartite sub-graphs.

More formally, let $1 \le b \le n$ be a positive integer, and let $\{I_1, \dots, I_{n/b}\}$ partition $[n]$ such that each subset of indices, $I_k$, $k = 1, \dots, n/b$, has size $|I_k| = b$.[‖] We call each $I_k$ a *block* and $b$ the *block size*. We specify the edges of the *(bipartite) block interconnectivity graph* $BG_n$ by $\{i,j\} \in BG_n$ iff $i$ and $j$ lie in a common block. For any given $m$-set of memories, $\mathbf{u}^1, \dots, \mathbf{u}^m$, we specify the interconnection weights, $w_{ij}$, $i \in [n]$, $\{i,j\} \in BG_n$, by the outer-product algorithm of prescription (2).

**Proposition 3.1** *With interconnectivities specified by the block interconnectivity graph, $BG_n$, and weights by the outer-product algorithm, the energy function, $E$, is non-increasing along any trajectory in any mode of operation.*

PROOF: Let $\mathbf{W}_k$ be the sub-matrix of weights corresponding to the components of block $I_k$. Note that $\mathbf{W}_k$ is symmetric, non-negative definite for each $k = 1, \dots, n/b$. Now, for any vector $\mathbf{u} \in \mathbb{B}^n$, let $\mathbf{u}_k \in \mathbb{B}^b$ denote the binary $b$-tuple of components of $\mathbf{u}$ in block $I_k$. We can then write the energy function as

$$E(\mathbf{u}) = -\langle \mathbf{u}, \mathbf{W}\mathbf{u} \rangle = -\sum_{k=1}^{n/b} \langle \mathbf{u}_k, \mathbf{W}_k \mathbf{u}_k \rangle = \sum_{k=1}^{n/b} E_k(\mathbf{u}_k),$$

where, for each $k$, $E_k$ denotes the energy function for block $I_k$. As the blocks are disjoint, two distinct vectors $\mathbf{u}_k$ and $\mathbf{u}_l$ do not share any components. Consequently, each $E_k$ is non-increasing along any trajectory in $\mathbb{B}^n$, and thus, so is $E$. ∎

Let $\mathcal{CK}_n^m$ denote the *complete code* of all choices of ordered multisets of size $m$ from $\mathbb{B}^n$.

**Theorem 3.2** *Let the block size $b$ be such that $b = \Omega(\log n)$ as $n \to \infty$, and let $0 \le \rho < 1/2$ be a fixed attraction radius. Then, for block interconnectivity graphs $BG_n$, complete codes $\mathcal{CK}_n^m$, and the outer-product algorithm, the monotone $\rho$-attractor capacity is $(1-2\rho)^2 b/2 \log bn$.*

**Corollary 3.3** *Under the conditions of theorem 3.2 the fixed point memory capacity is $b/2 \log bn$.*

[‖] Here, as in the rest of the paper, we ignore details with regard to rounding to the nearest integer in an effort to simplify notation. The modifications to be made for formal correctness will be obvious, and do not affect the results.

**Corollary 3.4** *For a fully-interconnected graph, complete codes $\mathcal{CK}_n^m$, and the outer-product algorithm, the fixed point memory capacity is $n/4\log n$.*

Corollary 3.4 is the main result shown by McEliece, *et al* [1]. Theorem 3.2 is a slight extension of the result and shows the natural result that increased sparsity causes a loss in capacity if the code is complete, i.e., all choices of memories are considered admissible. It is possible, however, to design codes to take advantage of the sparse interconnectivity structure as the following simple construction indicates.

Without loss of generality let us assume that block $I_1$ consists of the first $b$ indices, $[b]$, block $I_2$ the next $b$ indices, $[2b]-[b]$, and so on, with the last block $I_{n/b}$ consisting of the last $b$ indices, $[n]-[n-b]$. We can then partition any vector $\mathbf{u} \in \mathbb{B}^n$ as

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{n/b} \end{pmatrix}, \qquad (3)$$

where for $k=1,\ldots,n/b$, $\mathbf{u}_k$ is the sub-vector of components corresponding to block $I_k$. For $M \geq 1$ we form the *block code* $\mathcal{BK}_n^{M^{n/b}}$ as follows: to each ordered multiset of $M$ vectors, $\mathbf{u}^1,\ldots,\mathbf{u}^M$ from $\mathbb{B}^n$, we associate a unique ordered multiset in $\mathcal{BK}_n^{M^{n/b}}$ by lexicographically ordering all $M^{n/b}$ vectors of the form

$$\begin{pmatrix} \mathbf{u}_1^{\alpha_1} \\ \mathbf{u}_2^{\alpha_2} \\ \vdots \\ \mathbf{u}_{n/b}^{\alpha_{n/b}} \end{pmatrix}, \qquad \alpha_1,\alpha_2,\ldots,\alpha_{n/b} \in [M].$$

Thus, we obtain an admissible set of $M^{n/b}$ memories from any ordered multiset of $M$ vectors in $\mathbb{B}^n$ by "mixing" the blocks of the vectors. We call each $M$-set of vectors, $\mathbf{u}^1,\ldots,\mathbf{u}^M \in \mathbb{B}^n$, the *generating vectors* for the corresponding admissible set of memories in $\mathcal{BK}_n^{M^{n/b}}$.

EXAMPLE: Consider a case with $n=4$, block size $b=2$, and $M=2$ generating vectors. To any 2-set of generating vectors there corresponds a unique $4(=M^{n/b})$-set in the block code as follows:

$$\begin{pmatrix} u_1^1 \\ u_2^1 \\ \cdots \\ u_3^1 \\ u_4^1 \end{pmatrix}, \begin{pmatrix} u_1^2 \\ u_2^2 \\ \cdots \\ u_3^2 \\ u_4^2 \end{pmatrix} \longmapsto \begin{pmatrix} u_1^1 \\ u_2^1 \\ \cdots \\ u_3^1 \\ u_4^1 \end{pmatrix}, \begin{pmatrix} u_1^1 \\ u_2^1 \\ \cdots \\ u_3^2 \\ u_4^2 \end{pmatrix}, \begin{pmatrix} u_1^2 \\ u_2^2 \\ \cdots \\ u_3^1 \\ u_4^1 \end{pmatrix}, \begin{pmatrix} u_1^2 \\ u_2^2 \\ \cdots \\ u_3^2 \\ u_4^2 \end{pmatrix}.$$

The basic idea behind the formation of the block code is that if each sub-vector $\mathbf{u}_k^\alpha$ is stable with respect to the fully-interconnected submatrix of weights $\mathbf{W}_k$ (i.e., the energy $E_k(\mathbf{u}_k^\alpha)$ is a local minimum) then the vector $\mathbf{u}$ is stable with respect to the matrix of weights $\mathbf{W}$ (i.e., the energy $E(\mathbf{u})$ is also a local minimum) for the block interconnectivity graph $BG_n$. Thus, we can mix any combination of stable vectors $\mathbf{u}_k^\alpha$ to obtain a stable vector $\mathbf{u}$. Consequently, if we choose $M$ small enough that for most choices of $M$ vectors, $\mathbf{u}^1,\ldots,\mathbf{u}^M$, in $\mathbb{B}^n$, each of the vectors $\mathbf{u}_k^\alpha$, $\alpha=1,\ldots,M$ is stable for each of the blocks $k=1,\ldots,n/b$, then we can generate a relatively large number of stable vectors ($M^{n/b}$ in number) by mixing the blocks. We will take care of technical details in the appendix: specifically, we need stability of $M$ vectors for a large number of $n/b$ blocks simultaneously; further, to estimate capacity when there is error-correction we will have to guard against the possibility that $\rho n$ errors in a memory translates into a disproportionate share of errors in one or more blocks.

**Theorem 3.5** *Let $0 \leq \rho < 1/2$ be a fixed attraction radius. Then we have the following capacity estimates for block interconnectivity graphs $BG_n$, block codes $\mathcal{BK}_n^m$, and the outer-product algorithm.*

*a) If the block size $b$ satisfies $n \log\log bn/b \log bn \to 0$ as $n \to \infty$ then the monotone $\rho$-attractor capacity is*

$$\left[\frac{(1-2\rho)^2 b}{2\log bn}\right]^{n/b}.$$

*b) Define for any $\nu$*

$$C_n(\nu) = 2^{\frac{n\log b}{b\log 2}\left[1-\frac{\log\log bn+\log 2(1-2\rho)^{-2}}{\log b}+\frac{\nu\log\log bn}{(\log b)(\log bn)}\right]}.$$

*If the block size $b$ satisfies $b/\log n \to \infty$ and $b\log bn/\log\log bn = O(n)$ as $n \to \infty$, then $C_n(\nu)$ is a lower monotone $\rho$-attractor capacity for any choice of $\nu < 3/2$ and $C_n(\nu)$ is an upper monotone $\rho$-attractor capacity for any $\nu > 3/2$.*

**Corollary 3.6** *If, for fixed $t \geq 1$, we have $b = n/t$, then for any fixed attraction radius $0 \leq \rho < 1/2$, graphs $BG_n$, codes $B\mathcal{K}_n^m$, and the outer-product algorithm, the monotone $\rho$-attractor capacity is*

$$(1-2\rho)^{2t}t^{-t}4^{-t}\left(\frac{n}{\log n}\right)^t.$$

**Corollary 3.7** *For any fixed attraction radius $0 \leq \rho < 1/2$, and for any $\tau < 1$, a constant $c > 0$ and a code of size $\Omega\left(2^{cn^{2-\tau}}\right)$ can be found such that it is possible to achieve lower monotone $\rho$-attractor capacities which are $\Omega\left(2^{n^\tau}\right)$ in recurrent neural networks with interconnectivity graph of degree $\Theta\left(n^{1-\tau}\right)$.*

REMARKS: If the number of blocks is kept fixed as $n$ grows (i.e., the block size grows linearly with $n$) then capacities polynomial in $n$ are attained. If the number of blocks increases with $n$ (i.e., the block size grows sub-linearly with $n$) then super-polynomial capacities are attained.

# 4 NESTED SPARSITY

Recently, Baram [5] has proposed the investigation of certain nested codes geared towards exploiting certain classes of sparsely interconnected neural networks. The basic model can be described in terms of a nesting of block interconnectivity graphs: a hierarchy of blocks is defined with blocks at any given nesting level derived recursively from blocks at the previous level. More precisely, let $b$ as before denote the block size, $1 \leq b \leq n$, and let the positive integer $1 \leq h \leq \log n/\log b$ denote the *nesting depth*. For each *nesting level* $l = 1, \ldots, h$, we recursively define a disjoint collection of blocks, $I_1^l, \ldots, I_{nb^{-l}}^l$, as follows.

BASE: As in the block interconnectivity graph, at nesting level 1 the blocks $I_1^1, \ldots, I_{n/b}^1$ partition $[n]$, with each block having size $b$.

RECURSION: Let $I_1^l, \ldots, I_{nb^{-l}}^l$ be blocks corresponding to nesting level $l$. For $k = 1, \ldots, nb^{-l}$ let $i_k^l \in I_k^l$ be a specification of indices. The blocks corresponding to nesting level $(l+1)$ are now chosen so as to partition the specified set of indices $\{i_1^l, \ldots, i_{nb^{-l}}^l\}$, and such that each block has size $b$.

We specify the edges of the *(bipartite) nested interconnectivity graph* $NG_n$ by $\{i,j\} \in NG_n$ iff $i$ and $j$ lie in a common block at any nesting level. The *nested code* $\mathcal{NK}_n^m$ we consider is just the block code defined for the lowest nesting level, $l = 1$. Again, for any $m$-set of memories, $\mathbf{u}^1, \ldots, \mathbf{u}^m$, we specify the interconnection weights, $w_{ij}$, $i \in [n]$, $\{i,j\} \in NG_n$, by the outer-product algorithm of prescription (2).

The nested interconnection graph structure is very similar to that of the block interconnection graph with fully-interconnected disjoint subsets of neurons. For the nested structure, however, a small number of interconnections are permitted *between* blocks. At the first nesting level the structure is that of the block interconnectivity graph with $n/b$ disjoint blocks of fully-interconnected neurons. For the next nesting layer, one neuron is specified from each of the $n/b$ blocks of the first layer and these are grouped into $n/b^2$ blocks of fully-interconnected neurons. Thus, a specified neuron in each block in layer 1 is permitted connections with neurons in an additional $b-1$ blocks. This exercise is repeated recursively for each of the remaining layers.

We hence have essentially a block interconnectivity graph except that certain rare "long range interactions" are allowed across blocks of fully-interconnected neurons. The intuitive idea behind this setup is to have a sparsely interconnected network very reminiscent of a block structure in which a certain limited amount of communication is allowed between blocks of distinct features.

It turns out that the ideas developed in the analysis of block sparsity can be applied to nested structures as well, and in fact, the relatively small number of inter-block interconnections do not alter capacity.

**Theorem 4.1** *Consider nested interconnectivity graphs $NG_n$ with block size $b$ and nesting depth $h$, nested codes $\mathcal{NK}_n^m$, and the outer-product algorithm. Then, for any $0 \leq \rho < 1/2$, the monotone $\rho$-attractor capacity estimates of theorem 3.5 continue to hold under the same conditions on block size $b$. In particular, the capacity estimates are independent of the nesting depth.*

# 5 SPECTRAL ALGORITHM

The spectral algorithm was proposed (cf. Venkatesh and Psaltis [9] and Personnaz, *et al* [10]) as an alternative to the outer-product algorithm in the context of neural associative memory. At the expense of some additional algorithmic complexity the algorithm circumvents the need for orthogonality of memories in the outer-product algorithm and improves the fixed-point storage capacity from the sub-linear capacities that obtain (for fully-interconnected networks) to capacities linear in $n$ [9]. Concomitant increases in capacity may now be obtained for block interconnectivity graphs and block codes by extending the spectral algorithm in a manner analogous to the treatment earlier.

More specifically, consider a block interconnectivity graph $BG_n$ with block size $b$, and the block code $\mathcal{BK}_n^{M^{n/b}}$. (As before, assume the indices are assigned sequentially to the blocks $I_1, \ldots, I_{n/b}$.) Consider a choice of an admissible $M^{n/b}$-set of memories from $\mathcal{BK}_n^{M^{n/b}}$ corresponding to the $M$-set of generating vectors, $\mathbf{u}^1, \ldots, \mathbf{u}^M \in \mathbb{B}^n$. Now consider the $k$-th block, $k \in [n/b]$. Define the $b \times M$ matrix of column vectors

$$\mathbf{U}_k = [\ \mathbf{u}_k^1 \ \ \mathbf{u}_k^2 \ \ \cdots \ \ \mathbf{u}_k^M\ ].$$

(Recall that $\mathbf{u}_k^\alpha$ is the vector of components corresponding to block $I_k$ of the generating vector $\mathbf{u}^\alpha$.) Let $\lambda_{k1}, \ldots, \lambda_{kM}$ be fixed positive numbers and let $\Lambda_k = \mathrm{dg}(\lambda_{k1}, \ldots, \lambda_{kM})$. For each $k$ define the sub-matrix of weights, $\mathbf{W}_k$, corresponding to the components of block $I_k$ by

$$\mathbf{W}_k = \mathbf{U}_k \Lambda_k \mathbf{U}_k^\dagger,$$

where $\mathbf{U}_k^\dagger$ denotes the pseudo-inverse of $\mathbf{U}_k$. (If $\mathbf{U}_k$ is full-rank then $\mathbf{U}_k^\dagger = (\mathbf{U}_k^T\mathbf{U}_k)^{-1}\mathbf{U}_k^T$, where $\mathbf{U}_k^T$ denotes the transpose of $\mathbf{U}_k$.) The above prescription generalises the spectral algorithm to block interconnectivity graphs and block codes. [The case of a single block ($b = n$) yields the original algorithm.]

**Proposition 5.1** *For each $k$, let $\lambda_{k1} = \cdots = \lambda_{kM} = \lambda_k > 0$, and assume that $M$ generating vectors are chosen from a sequence of symmetric Bernoulli trials. Then with interconnectivities specified by the block interconnectivity graph, $BG_n$, and weights by the spectral algorithm, the energy function, $E$, is non-increasing along any trajectory in any mode of operation with probability approaching one asymptotically.*

PROOF: As before, $\mathrm{E}(\mathbf{u}) = \sum_{k=1}^{n/b} E_k(\mathbf{u}_k)$, where $E_k$ is the energy function corresponding to block $I_k$. Now each $\mathbf{U}_k$ is full-rank with probability approaching one asymptotically as a consequence of a theorem of Kahn, Komlós, and Szemerédi (see Appendix E). Thus, with high probability, each sub-matrix of weights, $\mathbf{W}_k$, is of the form $\mathbf{W}_k = \lambda_k\mathbf{U}_k\left(\mathbf{U}_k^T\mathbf{U}_k\right)^{-1}\mathbf{U}_k^T$. Thus, $\mathbf{W}_k$ is symmetric and its only eigenvalues are 0 and $\lambda_k > 0$, so that it is non-negative definite. Consequently, $E_k$ decreases along any trajectory with high probability, and hence, so does $E$. ∎

**Theorem 5.2** *If the block size satisfies $b/\log n \to \infty$ as $n \to \infty$ then, for block interconnectivity graphs $BG_n$, block codes $\mathcal{BK}_n^m$, and the spectral algorithm, the fixed point capacity is $b^{n/b}$.*

**Corollary 5.3** *If, for fixed $t \geq 1$, the block size $b = n/t$, then under the conditions of theorem 4.1 the fixed point capacity is $t^{-t}n^t$.*

**Corollary 5.4** *If, for fixed $0 < \tau \leq 1$, the block size $b = n^\tau$, then under the conditions of theorem 4.1 the fixed point capacity is $n^{\tau n^{1-\tau}}$.*

In particular, slightly better capacities obtain for the same code sizes than for the outer-product algorithm. The main technical problem here concerns the stability requirement that *all* the matrices $\mathbf{U}_k$ be full rank.

# 6 CONCLUSIONS

We have demonstrated that code size can be traded off for increased storage capacity, and that very large capacities obtain for carefully selected codes which are still exponential in size. The analysis carried out here for block sparsity where we have a uniform block size can be readily extended in obvious fashion for non-uniform block sizes, where each block $I_k$ has a distinct block size $b_k$. For instance, if each $b_k = \Theta(n)$ then the monotone $\rho$-attractor capacity for graphs $BG_n$ (with these block sizes), codes $\mathcal{BK}_n^m$, and the outer-product algorithm becomes $\prod_k (1-2\rho)^2 b_k / 2 \log b_k n$. (The block code is again formed in the indicated manner by mixing blocks.) Nesting blocks does not change the intrinsic capacity significantly; however, nesting may be used as a vehicle for introducing "long range interactions" between distinct blocks of features.

A feature of the analysis of block sparsity in this paper is that we confer a probability distribution on admissible memories from the uniform distribution on the code. The probability distribution we consider is, hence, on ordered multisets of memories and not on individual memories; in particular, the distribution is not a product distribution (except in simple cases such as the complete code). This may be seen as a limitation in the technique espoused here. We do not currently know whether it is possible to simultaneously achieve large capacities and code sizes to take advantage of specific sparse structures with memories drawn from a suitable product distribution. A general open question along these lines is the design of codes of large size and achieving large capacity for any given (sparse) interconnectivity graph.

# A Preliminary Lemmas

Consider a set of $N$ fully-interconnected neurons. Let $\mathbf{u}^1, \ldots, \mathbf{u}^M \in \mathbb{B}^N$ be an $M$-set of memories with components drawn from a sequence of symmetric Bernoulli trials, i.e., the memory components are i.i.d. with

$$\mathbf{P}\{u_i^\alpha = -1\} = \mathbf{P}\{u_i^\alpha = 1\} = 1/2, \qquad i = 1, \ldots, N, \quad \alpha = 1, \ldots, M.$$

Note that we are considering the complete code here and that the product distribution on memories above induces a uniform distribution on admissible $M$-sets of memories in the code $\mathcal{CK}_N^M$. The weights are specified by the outer-product algorithm. Specifically,

$$w_{ij} = \sum_{\beta=1}^{M} u_i^\beta u_j^\beta, \qquad i, j = 1, \ldots, N.$$

Let $0 \leq \rho < 1/2$ be the desired attraction radius. Corresponding to each memory, $\mathbf{u}^\alpha$, let $\hat{\mathbf{u}}^\alpha$ denote a random probe at mean distance $\rho n$ from $\mathbf{u}^\alpha$ generated according to prescription (1). If each of the $m$ memories is to be a monotone $\rho$-attractor then we will require that each of the $NM$ random sums

$$X_i^\alpha = u_i^\alpha \sum_{j=1}^{N} w_{ij} \hat{u}_j^\alpha, \qquad i = 1, \ldots, N, \quad \alpha = 1, \ldots, M,$$

be positive with high probability. Form the random variables

$$\zeta_{ij}^{\alpha\beta} = u_i^\alpha \hat{u}_j^\alpha u_i^\beta u_j^\beta, \qquad j = 1, \ldots, N, \quad \beta = 1, \ldots, M, \tag{4}$$

Substituting for the weights $w_{ij}$ we then have

$$X_i^\alpha = Y_i^\alpha + Z_i^\alpha, \tag{5}$$

where

$$Y_i^\alpha = \sum_{j \neq i} \zeta_{ij}^{\alpha\alpha} + M\zeta_{ii}^{\alpha\beta}, \tag{6}$$

and

$$Z_i^\alpha = \sum_{j \neq i} \sum_{\beta \neq \alpha} \zeta_{ij}^{\alpha\beta}. \tag{7}$$

Let us begin by estimating the probability that a particular memory component is not retrieved from a random probe in one synchronous step. We need the following technical result on large deviations.

**Lemma A.1** *Let $x_1 < x_2$ be any two real numbers and let $\{\zeta_j\}$ be a sequence of i.i.d. random variables drawn from a sequence of Bernoulli trials with*

$$\zeta_j = \begin{cases} x_1 & \textit{with probability } q = 1 - p \\ x_2 & \textit{with probability } p, \end{cases}$$

*where $0 < p < 1$. For each $K$ let $S_K = \sum_{j=1}^K \zeta_j$. If as $K \to \infty$ the real number $v$ varies such that $v/\sqrt{K} \to \infty$ and*

$$v = \begin{cases} o(K^{2/3}) & \textit{if } p \neq q \\ o(K^{3/4}) & \textit{if } p = q = 1/2 \end{cases}$$

*then*

$$\mathbf{P}\left\{S_K - K(px_2 + qx_1) < -v(x_2 - x_1)\right\} \sim \mathbf{P}\left\{S_K - K(px_2 + qx_1) > v(x_2 - x_1)\right\} \sim \frac{\sqrt{pqK}\, e^{-v^2/2pqK}}{\sqrt{2\pi}\, v}.$$

The result is just a slight extension of the classical large deviation form of the DeMoivre-Laplace limit theorem for sums of $(0, 1)$ random variables.

For notational simplicity denote $N_\rho = (1 - 2\rho)^2 N$. For $0 \leq \rho < 1/2$ define the function

$$f(\rho) = \left((1 - \rho)e^{-(1-2\rho)} + \rho e^{1-2\rho}\right). \tag{8}$$

**Lemma A.2** *If $M$ satisfies $MN^{-2/3} \to \infty$ and $M = o(N)$ as $N \to \infty$, then*

$$\mathbf{P}\left\{X_i^\alpha \leq 0\right\} \sim q_1 = \frac{f(\rho)\sqrt{M}}{\sqrt{2\pi N_\rho}} \exp\left\{-\left(\frac{N_\rho}{2M}\right)\right\}.$$

Remark: The asymptotic form for component error above is somewhat different from the similar expression derived in McEliece, *et al* [1] as a consequence of differing modes adopted for the generation of random probes.

Proof: Fix the component index $i$ and the memory index $\alpha$. Note that the random variables $\zeta_{ij}^{\alpha\beta}$ defined in equation (4) are independent, so that the random variables $Y_i^\alpha$ and $Z_i^\alpha$ are independent and comprised of sums of independent, ±1 random variables.

Fix $0 < \delta < 1/6$. For $j = i$ or $\beta = \alpha$ the r.v.'s $\zeta_{ij}^{\alpha\beta}$ are i.i.d. and take on values -1 and 1 with probabilities $\rho$ and $1-\rho$, respectively. Let $\mathcal{E}^+$ denote the event that $\zeta_{ii}^{\alpha\beta} = \hat{u}_i^\alpha u_i^\alpha = 1$. We then have by an application of lemma A.1 that

$$\begin{aligned} \mathbf{P}\left\{ |Y_i^\alpha - (1-2\rho)(N-1) - M| > N^{1/2+\delta} \middle| \mathcal{E}^+ \right\} &= \mathbf{P}\left\{ \left| \sum_{j \neq i} \zeta_{ij}^{\alpha\alpha} - (1-2\rho)(N-1) \right| > N^{1/2+\delta} \right\} \\ &= O\left(e^{-c_1 N^{2\delta}}\right). \end{aligned} \tag{9}$$

Conditioned upon $\mathcal{E}^+$ let $\mathcal{S}^+$ denote the set of sample points for which the following holds:

$$|Y_i^\alpha - (1-2\rho)(N-1) - M| \leq N^{1/2+\delta}.$$

We then have from equation (9) that

$$\mathbf{P}\left\{ \mathcal{S}^+ | \mathcal{E}^+ \right\} = 1 - O\left(e^{-c_1 N^{2\delta}}\right).$$

From elementary considerations we then have that

$$\mathbf{P}\left\{ X_i^\alpha \leq 0 \mid \mathcal{E}^+ \right\} = \mathbf{P}\left\{ Z_i^\alpha \leq -Y_i^\alpha \mid \mathcal{E}^+ \right\} = \mathbf{P}\left\{ Z_i^\alpha \leq -Y_i^\alpha \mid \mathcal{S}^+, \mathcal{E}^+ \right\} + O\left(e^{-c_1 N^{2\delta}}\right).$$

For $j \neq i$ and $\beta \neq \alpha$ the random variables $\zeta_{ij}^{\alpha\beta}$ are i.i.d. and symmetric, and take values in $\mathbb{B}$. Hence, $Z_i^\alpha$ is just a symmetric random walk over $(N-1)(M-1)$ steps. Also, conditioned upon $\mathcal{E}^+$ we have that within $\mathcal{S}^+$ the r.v. $Y_i^\alpha$ takes on values whose deviation from $(1-2\rho)(N-1)+M$ is at most $N^{1/2+\delta}$. For each sample value taken by $Y_i^\alpha$ (conditioned upon $\mathcal{E}^+$ and $\mathcal{S}^+$) lemma A.1, hence, applies for $M$ as in the statement of the lemma. In particular, let us say a deviation, $y$, is *allowable* if $|y| \leq N^{1/2+\delta}$, and let us denote by $p(y)$ the probability that $y$ is allowable:

$$p(y) = \mathbf{P}\left\{ Y_i^\alpha = (1-2\rho)(N-1) + M + y \mid \mathcal{S}^+, \mathcal{E}^+ \right\}.$$

For each allowable $y$, and for $M$ as in the statement of the lemma, we then have by lemma A.1 that

$$\mathbf{P}\left\{Z_i^\alpha \leq -(1-2\rho)(N-1) - M - y\right\} \sim \frac{\sqrt{M}}{\sqrt{2\pi N_\rho}} \exp\left(-\frac{N_\rho}{2M} - (1-2\rho)\right).$$

Hence,

$$\begin{aligned} \mathbf{P}\left\{ X_i^\alpha \leq 0 \mid \mathcal{E}^+ \right\} &= \sum_{\text{allowable } y} p(y)\, \mathbf{P}\left\{Z_i^\alpha \leq -(1-2\rho)(N-1) - M - y\right\} + O\left(e^{-c_1 N^{2\delta}}\right) \\ &\sim \frac{\sqrt{M}}{\sqrt{2\pi N_\rho}} \exp\left(-\frac{N_\rho}{2M} - (1-2\rho)\right). \end{aligned}$$

As $\mathcal{E}^+$ occurs independently with probability $(1-\rho)$ it follows that

$$\mathbf{P}\left\{ X_i^\alpha \leq 0, \mathcal{E}^+ \right\} \sim \frac{(1-\rho)\sqrt{M}}{\sqrt{2\pi N_\rho}} \exp\left(-\frac{N_\rho}{2M} - (1-2\rho)\right). \tag{10}$$

In similar spirit let us define the event $\mathcal{E}^-$ that $\zeta_{ii}^{\alpha\beta} = \hat{u}_i^\alpha u_i^\alpha = -1$, and conditioned upon $\mathcal{E}^-$ the set of sample points $\mathcal{S}^-$ for which

$$|Y_i^\alpha - (1-2\rho)(N-1) + M| \leq N^{1/2+\delta}.$$

As before, we obtain

$$\mathbf{P}\left\{ \mathcal{S}^- | \mathcal{E}^- \right\} = 1 - O\left(e^{-c_1 N^{2\delta}}\right).$$

Proceeding in similar vein we can now demonstrate that

$$\mathbf{P}\left\{X_i^\alpha \leq 0, \mathcal{E}^-\right\} \sim \frac{\rho\sqrt{M}}{\sqrt{2\pi N_\rho}} \exp\left(-\frac{N_\rho}{2M} + (1-2\rho)\right). \tag{11}$$

Combining equations (10) and (11) and equation (8) completes the proof of the lemma. ▮

The next lemma concerns the joint distribution of $r$ sums, $X_{i_g}^{\alpha_g}$, $g = 1, \ldots, r$. The result is essentially the main lemma demonstrated in McEliece, *et al* [1], except for certain correction factors corresponding to the terms $f(\rho)$ arising from a different generation of random probes. (The result we show below is actually a slightly stronger version which appears in Venkatesh [12, pages 220–239].)

**Lemma A.3** *Let $r$ be any fixed, positive integer, and let $(i_g, \alpha_g) \in [N] \times [M]$, $g = 1, \ldots, r$ be distinct pairs of integers. If $M \geq N^\sigma$ with $3/4 < \sigma < 1$, then, under the hypothesis of lemma A.2,*

$$\mathbf{P}\left\{X_{i_g}^{\alpha_g} \leq 0,\ g = 1, \ldots, r\right\} \sim q_1^r, \qquad (N \to \infty).$$

Essentially the same proof that appears in McEliece, *et al* [1] or Venkatesh [12] can be modified to show the following result, care being taken to rigourously account for all correction terms. We will not go into it here.

## B Proof of Theorem 3.2

Consider a set of $n$ neurons, interconnected according to a block interconnectivity graph with block size $b$. Let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of memories with components drawn from a sequence of symmetric Bernoulli trials, and let the weights be specified by the outer-product algorithm. Note that, as before, we are considering the complete code here and that the product distribution on memories above induces a uniform distribution on admissible $m$-sets of memories in the code $\mathcal{CK}_n^m$.

For the memories to be monotone $\rho$-attractors a necessary and sufficient condition is that the components of the memories corresponding to each block be retrieved from a random probe. Let us consider block $I_k$ for definiteness, and, as before, let $\mathbf{u}_k^1, \ldots, \mathbf{u}_k^m \in \mathbb{B}^b$ be the vectors of components of the memories corresponding to block $I_k$. Now, within each block the components of a memory are updated independently of the components in other blocks as a consequence of block sparsity. As the components of the memories as well as the random probes are drawn independently, the results of appendix A apply here with $M = m$ and $N = b$. We will need the following version of the inclusion-exclusion principle.

**Lemma B.1** *Let $E_1, \ldots, E_N$ be measurable subsets of a probability space. For $1 \leq r \leq N$, let $\sigma_r$ be the sum of probabilities of all sets formed by intersecting $r$ of the events $E_1, \ldots, E_N$:*

$$\sigma_r = \sum_{1 \leq j_1 < j_2 < \cdots < j_r \leq N} \mathbf{P}\left\{\bigcap_{g=1}^r E_{j_g}\right\}.$$

*Then for every $K$, $1 \leq K \leq N/2$,*

$$\sum_{r=1}^{2K} (-1)^{r-1}\sigma_r \leq \mathbf{P}\left\{\bigcup_{j=1}^N E_j\right\} \leq \sum_{r=1}^{2K-1} (-1)^{r-1}\sigma_r.$$

For fixed $r$, let $\sigma_r$ denote the sum over the $\binom{mb}{r}$ distinct choices of $r$ memory components in block $I_k$ of the probabilities that a distinct choice of $r$ memory components is not retrieved:

$$\sigma_r = \sum_{\{(i_1,\alpha_1),\ldots,(i_r,\alpha_r)\} \subset I_k \times [m]} \mathbf{P}\left\{\bigcap_{g=1}^r X_{i_g}^{\alpha_g} \leq 0\right\},$$

where

$$X_{i_s}^{\alpha_s} = u_{i_s}^{\alpha_s} \sum_{j:\{i_s,j\}\in BG_n} w_{i_s j}\hat{u}_j^{\alpha_s} = \sum_{j\in I_k}\sum_{\beta=1}^{m} u_{i_s}^{\alpha_s}\hat{u}_j^{\alpha_s}u_{i_s}^{\beta}u_j^{\beta}.$$

Lemma A.3 applies to this case, so that as $b \to \infty$ we have

$$\sigma_r \sim \frac{(mbq_1)^r}{r!}.$$

Here $q_1$ is given by lemma A.2 as

$$q_1 = \frac{f(\rho)\sqrt{m}}{\sqrt{2\pi b_\rho}} \exp\left\{-\left(\frac{b_\rho}{2m}\right)\right\}.$$

with $f(\rho)$ as defined in equation (8) and $b_\rho = (1-2\rho)^2 b$. We can now apply lemma B.1 to estimate the probability, $q_b$, that one or more memory components is not retrieved inside block $I_k$. Let us choose a rate of growth for the number of memories, $m$, (as a function of the block size, $b$) small enough that the term $mbq_1$ is bounded. By choosing larger and larger (but fixed) sizes $K$ in lemma B.1, as $b \to \infty$ we can make both the upper and lower bound on the probability $q_b$ approach arbitrarily close to $1 - e^{-mbq_1}$. Alternatively, the probability, $p_b = 1 - q_b$, that, for each memory, all the components in the block are retrieved from a random probe with parameter $0 \le \rho < 1/2$ is given by $p_b \sim e^{-mbq_1}$. As there are no interconnections in between blocks, the retrieval of memory components is independent across blocks. Hence, the probability of retrieval of all the components of all the memories is $p_b^{n/b}$. We have thus established the following

**Lemma B.2** *With $q_1$ as given above, let $m$ increase slowly enough with $b$ so that $mbq_1$ remains bounded as $b \to \infty$. Let $p(m,n,\rho)$ denote the probability that, for each memory, a random probe with parameter $0 \le \rho < 1/2$ is mapped into the memory in one synchronous step (i.e., all memory components are retrieved from the probe in one synchronous step). Then, for any $0 < t < 1 < s$ we have*

$$e^{-smnq_1} \lesssim p(m,n,\rho) \lesssim e^{-tmnq_1}, \qquad (b \to \infty).$$

Now, for any fixed choice of $\delta$ let $M$ be an integer sequence such that as $n \to \infty$

$$M = \frac{(1-2\rho)^2 b}{2\log bn}\left[1 + \frac{\frac{3}{2}\log\log bn + \log\left(\frac{4\delta\sqrt{\pi}}{f(\rho)(1-2\rho)^2}\right)}{\log bn} - O\left(\frac{\log\log bn}{\log^2 bn}\right)\right]. \tag{12}$$

It is easy to check that $mbq_1$ remains bounded as $n \to \infty$ if $m$ is chosen equal to $M$ for any fixed choice of $\delta$. Lemma B.2 hence applies.

Now, for any $\lambda > 0$ (chosen arbitrarily small), consider a number of memories

$$m' = \frac{(1+\lambda)(1-2\rho)^2 b}{2\log bn}.$$

For any choice of $\epsilon > 0$ fix $0 < t < 1$ and choose $\delta = -t^{-1}\log\epsilon$. Choose $M$ according to equation (12) for such a choice of $\delta$. Using the upper bound for $p(m,n,\rho)$ in lemma B.2 with $m$ replaced by $M$ yields that for such a choice of $M$, $p(M,n,\rho) \lesssim \epsilon$. Now it is clear that as $n \to \infty$ we will have $m' \gtrsim M$ whatever be the choices of $\lambda$, $\epsilon$, and $t$. By uniformity, hence, $p(m',n,\rho) \lesssim p(M,n,\rho) \lesssim \epsilon$. As $\epsilon$ can be chosen arbitrarily small it follows that $(1-2\rho)^2 b/2\log bn$ is an upper monotone $\rho$-attractor capacity.

Now again, for a choice of $\lambda > 0$ small, consider a number of memories

$$m'' = \frac{(1-\lambda)(1-2\rho)^2 b}{2\log bn}.$$

For any choice of $\epsilon > 0$ chosen arbitrarily small fix $s > 1$ and choose $\delta = -s^{-1}\log(1-\epsilon)$. Choose $M$ according to equation (12) for such a choice of $\delta$. Now using the lower bound for $p(m,n,\rho)$ in lemma B.2 with $m$ replaced by $M$ yields that for such a choice of $M$, $p(M,n,\rho) \gtrsim 1-\epsilon$. Now we have $m'' \lesssim M$ as $n \to \infty$ whatever be the choices of $\lambda$, $\epsilon$, and $s$. By uniformity, hence, $p(m'',n,\rho) \gtrsim p(M,n,\rho) \gtrsim 1-\epsilon$. As $\epsilon$ can be chosen arbitrarily small it follows that $(1-2\rho)^2 b/2\log bn$ is also a lower monotone $\rho$-attractor capacity. This concludes the proof of theorem 3.2. ∎

# C Proof of Theorem 3.5

Let $\mathbf{u}^1, \ldots, \mathbf{u}^M \in \mathbb{B}^n$ be a randomly chosen $M$-set of generating vectors with components drawn from a sequence of symmetric Bernoulli trials. Corresponding to the $M$-set of generating vectors there is a unique $M^{n/b}$-set of memories, $\tilde{\mathbf{u}}^1, \ldots, \tilde{\mathbf{u}}^{M^{n/b}} \in \mathbb{B}^n$, in the block code $\mathcal{BK}_n^{M^{n/b}}$. Note that the product distribution on generating vectors induces the uniform distribution on the block code.[**] For $\{i,j\} \in BG_n$ the weights prescribed by the outer-product algorithm are given by

$$w_{ij} = \sum_{\beta=1}^{M^{n/b}} \tilde{u}_i^\beta \tilde{u}_j^\beta .$$

As $\beta$ runs through the indices 1 through $M^{n/b}$, for each of the $M$ generating vectors, $\mathbf{u}^\beta$, the corresponding term $u_i^\beta u_j^\beta$ occurs $M^{\frac{n}{b}-1}$ times in the sum above. (This follows from the construction of the block code: each vector of components $\mathbf{u}_k^\beta$ corresponding to block $I_k$ and generating vector $\mathbf{u}^\beta$ is used in the $k$-th block of exactly $M^{\frac{n}{b}-1}$ vectors in the generated $M^{n/b}$-set of memories in the block code.) Thus:

$$w_{ij} = M^{\frac{n}{b}-1} \sum_{\beta=1}^{M} u_i^\beta u_j^\beta , \qquad \{i,j\} \in BG_n. \tag{13}$$

Scaling all the weights by the positive scale factor $M^{\frac{n}{b}-1}$ does not affect capacity. The situation is now similar to that analysed earlier: the outer-product weights for the graph $BG_n$ are generated from a set of vectors whose components are drawn from a sequence of symmetric Bernoulli trials.

Now we claim that the $M^{n/b}$-set of memories are monotone $\rho$-attractors iff each vector of components, $\mathbf{u}_k^\beta$, corresponding to each block $I_k$, $k = 1, \ldots, n/b$, and each of the generating vectors, $\mathbf{u}^\beta$, $\beta = 1, \ldots, M$, is a monotone $\rho$-attractor.[††] This follows because of the disjoint nature of the blocks, and the indpendent assignment of components to the random probe. But by independence across the blocks this is equivalent to requiring that each of the $M$ generating vectors are monotone $\rho$-attractors for the matrix of weights specified by equation (13). Lemma B.2 now applies directly. In particular, let the number of generating vectors, $M$, be chosen as in equation (12). With a choice of $s > 1$, $\delta = -s^{-1}\log(1-\epsilon)$ all the generating vectors are monotone $\rho$-attractors, and hence so are the $M^{n/b}$ generated memories, with probability at least $1-\epsilon$; with a choice of $0 < t < 1$ and $\delta = -t^{-1}\log\epsilon$ some generating vectors fail to attract montonically over a radius $\rho$, and hence so do some of the $M^{n/b}$ generated memories, with probability at least $1-\epsilon$.

Consider first the case where the block size $b$ satisfies $n/b = o(\log bn/\log\log bn)$ as $n \to \infty$. Then

$$M^{n/b} = \left[\frac{(1-2\rho)^2 b}{2\log bn}\right]^{n/b} (1 + o_\delta(1)).$$

The choices $s > 1$, $\delta = -s^{-1}\log(1-\epsilon)$, and $0 < t < 1$, $\delta = -t^{-1}\log\epsilon$ are both absorbed in the $o_\delta(1)$ term, so that

$$\left[\frac{(1-2\rho)^2 b}{2\log bn}\right]^{n/b}$$

is both a lower and an upper monotone $\rho$-attractor capacity.

[**] Herein lies the reason we defined codes in terms of ordered multisets of memories instead of sets of memories. We would like to preserve a product distribution on memory components because—as we saw in the previous sections—this provides certain amenities in analysis. This, however, corresponds to an urn model with replacement, and there is a non-zero (albeit small) probability that the same memories are drawn again. If the code is defined in terms of sets of memories instead of ordered multisets this results in a small, but annoying, non-uniformity in the distribution induced on the code.

[††] The utility of the random probe with parameter $\rho$ model is apparent here. The statement would not continue to hold *in toto* if the probe were to be selected, for instance, randomly from the Hamming ball of radius $\rho n$ at the memory. The difficulty is that $\rho n$ component errors in a memory need not translate into $\rho b$ errors in each block, and a disproportionate assignment of component errors in any one block will cause non-convergence to the memory components in that block. A (provable) large deviation limit theorem for the hypergeometric distribution is needed here.

Now consider the case where the block size $b$ satisfies $n/b = \Omega(\log bn/\log\log bn)$ and $b/\log n \to \infty$ as $n \to \infty$. Define for any $\nu$

$$C_n(\nu) = 2^{\frac{n\log b}{b\log 2}\left[1-\frac{\log\log bn+\log 2(1-2\rho)^{-2}}{\log b}+\frac{\nu\log\log bn}{(\log b)(\log bn)}\right]}.$$

For any fixed choice of $\delta$ it follows that $C_n(\nu) \lesssim M^{n/b}$ if $\nu < 3/2$, while $C_n(\nu) \gtrsim M^{n/b}$ if $\nu > 3/2$. Thus, $C_n(\nu)$ is a lower monotone $\rho$-attractor when $\nu < 3/2$ and an upper monotone $\rho$-attractor when $\nu > 3/2$. This concludes the proof of the theorem. ∎

# D Proof of Theorem 4.1

Let $NG_n$ be a nested interconnectivity graph with block size $b$ and nesting depth $h$. As before, let $\mathbf{u}^1, \ldots, \mathbf{u}^M \in \mathbb{B}^n$ be a random $M$-set of generating vectors (corresponding to a random $M^{n/b}$-set of memories in the nested code $\mathcal{NK}_n^{M^{n/b}}$). As before, the product distribution on the components of the generating vectors induces the uniform distribution on $\mathcal{NK}_n^{M^{n/b}}$. For nesting level 1 the situation is identical to that of block sparsity. Specifically, if indices $i$ and $j$ lie in a common block at nesting level 1, i.e., $\{i,j\} \subset I_k^1$ for some $k \in [n/b]$, then the corresponding weight $w_{ij}$ is as given by equation (13).

Let $i_k^1 \in I_k^1$, $k = 1, \ldots, n/b$ denote the specified indices which comprise nesting level 2. Consider block $I_k^1$. The probability that, for each memory, all $b$ components corresponding to this block are retrieved from a random probe with parameter $\rho$ in one synchronous step is certainly less than the probability that, for each memory, the $b-1$ components corresponding to indices $j \in I_k^1 \setminus \{i_k^1\}$ are retrieved. (Equality iff the probability of retrieval of the $i_k^1$-th component is one.) But these $b-1$ indices are only interconnected with other indices in the block $I_k^1$, so that the retrieval of the memory components corresponding to these $b-1$ indices in each block (sans the specified indices $i_k^1$) is a stochastically independent event across the blocks. (The fact that there are interconnections across blocks through the specified indices, $i_k^1$, cannot affect the other $b-1$ indices in each block in the first synchronous step, but only on later steps: in one synchronous step, only the value of the $i_k^1$-th component contributes to the updates of the remaining $b-1$ components corresponding to block $I_k^1$.) Consequently, we can again partition the problem into $n/b$ independent blocks. It hence suffices to consider just the generating vectors rather than the vastly larger number of generated memories in the code. Specifically, the requirement that $b-1$ components of each memory be retrieved in each block is equivalent to just requiring that for each block the $b-1$ components (disregarding the specified components $i_k^1$) of each generating vector are retrieved from a random probe with parameter $\rho$. Let $P'$ denote this probability. (This argument would not be hold if we had to consider retrieval of the $i_k^1$-th components as well because of the dependencies caused by the inter-block connections.) An argument similar to that leading up to lemma B.2 now yields that for any $0 < t < 1$, and rate of growth of $M$ with $b$ such that $M(b-1)q_1'$ is bounded,

$$p(M,n,\rho) \le P' \lesssim \exp\left(-\frac{tM(b-1)q_1'n}{b}\right) = \exp\left(-tMnq_1'\left(1-\frac{1}{b}\right)\right),$$

where, with the appropriate substitution of parameters in lemma A.2 we have

$$q_1' = \frac{f(\rho)\sqrt{M}}{\sqrt{2\pi b_\rho}}\exp\left\{-\left(\frac{b_\rho}{2M}\right)\right\}.$$

With $M$ as defined in equation (12) for a choice of $< 0t < 1$ and $\delta = -t^{-1}\log\epsilon$, it follows that

$$p(M,n,\rho) \lesssim e^{-t\delta(1-t)} = \epsilon + O\left(\frac{1}{b}\right).$$

The upper monotone $\rho$-attractor capacity estimates of theorem 3.5, hence, continue to hold for the nested case.

To obtain lower capacity estimates, note that removing the $n/b$ neurons corresponding to the specified indices $i_k^1$ results in a collapse of the nested structure into a block interconnectivity graph $BG_{n(b-1)/b}$ with block size $b-1$. Now, an examination of the random variables, $X_i^\alpha$, shows that each added interconnection weight specified by the outer-product algorithm improves the probability of retrieval of the corresponding memory component. The connections from the indices $i_k^1$ within a block, hence, contribute a positive probability to the probability of retrieval of any memory component within the block. Furthermore, we have the uniformity property that the probability of retrieval of each component improves monotonically with $n$ for fixed $M$. Hence, the probability $p(M,n,\rho)$ that, for each memory, all the components are retrieved from a random probe with parameter $\rho$ exceeds the probability, $P''$, that the corresponding vectors with the specified $n/b$ components removed are retrieved in the graph $BG_{n(b-1)/b}$ with block size $b-1$. But, here again we have an independent partition into $n/b$ blocks, and it suffices to estimate the probability that the components of the corresponding generating vectors (with the specified $n/b$ components removed) are retrieved. The argument leading to lemma B.2 again works and we have for any $s > 1$, and rate of growth of $M$ with $b$ such that $M(b-1)q_1''$ is bounded,

$$p(M,n,\rho) \geq P'' \gtrsim \exp\left(-\frac{sM(b-1)q_1''n(b-1)}{b(b-1)}\right) = \exp\left(-sMnq_1''\left(1-\frac{1}{b}\right)\right),$$

where, with the appropriate substitution of parameters in lemma A.2 we have

$$q_1'' = \frac{f(\rho)\sqrt{M}}{(1-2\rho)\sqrt{2\pi(b-1)}} \exp\left\{-\left(\frac{(1-2\rho)^2(b-1)}{2M}\right)\right\}.$$

With $M$ as defined in equation (12) for a choice of $s > 1$ and $\delta = -s^{-1}\log(1-\epsilon)$, it follows that when $b$ is such that $b/\log n \to \infty$ as $n \to \infty$, then

$$p(M,n,\rho) \gtrsim e^{-s\delta\left(1-O\left(\frac{1}{b}\right)\right)} = 1-\epsilon+O\left(\frac{1}{b}\right).$$

The lower monotone $\rho$-attractor capacity estimates of theorem 3.5, hence, also continue to hold for the nested case, and this concludes the proof. ∎

# E Proof of Theorem 5.2

Consider the block interconnectivity graph $BG_n$ with block size $b$ and the block code $BK_n^{M^{n/b}}$. Let $\mathbf{u}^1, \ldots, \mathbf{u}^M \in \mathbb{B}^n$ be a randomly selected $M$-set of generating vectors, as before, and let

$$\mathbf{U}_k = [\ \mathbf{u}_k^1 \ \ \mathbf{u}_k^2 \ \ \cdots \ \ \mathbf{u}_k^M\ ]$$

be the corresponding $b \times M$ matrix of column vectors corresponding to the block $I_k$. To show that each of the generated memories is stable it suffices to show that, for each generating vector, $\mathbf{u}^\beta$, each of the $n/b$ vector of components, $\mathbf{u}_k^\beta$, is stable. If $\mathbf{U}_k$ is full-rank then the sub-matrix of weights corresponding to block $I_k$ is given by

$$\mathbf{W}_k = \mathbf{U}_k \Lambda_k \left(\mathbf{U}_k^T \mathbf{U}_k\right)^{-1} \mathbf{U}_k^T, \tag{14}$$

where $\Lambda$ is diagonal with positive diagonal terms $\lambda_{k1}, \ldots, \lambda_{kM}$. If the representation (14) holds (i.e., $\mathbf{U}_k$ is full-rank) then, for any generating vector, $\mathbf{u}^\beta$, we have

$$\mathbf{W}_k \mathbf{u}_k^\beta = \lambda_{k\beta} \mathbf{u}_k^\beta \xmapsto{\text{sgn}} \mathbf{u}_k$$

as $\lambda_{k\beta} > 0$. The proof of the theorem will be complete if the probability that each of the matrices $\mathbf{U}_k$ is full-rank approaches one for large $n$. This follows as a consequence of the following new result due to Kahn, Komlós, and Szemerédi [11].

**Lemma E.1** *Let $A_b$ be a random $b \times b$ binary matrix whose components are drawn from a sequence of symmetric Bernoulli trials. Then there is a constant $1 < d \leq 2$ such that the probability that $A_b$ is singular is no more than $d^{-b}$ asymptotically as $b \rightarrow \infty$.*

REMARK: The earlier (1977) estimate of Komlós of $b^{-1/2}$ for the probability that $A_b$ is singular does not suffice for our purposes for block sizes $b = O\left(n^{1/2}\right)$.

Using the above lemma, the probability that all the matrices $\mathbf{U}_k$, $k = 1, \ldots, n/b$ are full-rank is at least $(1 - d^{-b})^{n/b}$ which asymptotically tends to one rather quickly. ∎

# References

[1] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "On the capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 461-482, 1987.

[2] J. Komlós and R. Paturi, "Convergence results in an associative memory model," *Neural Networks*, vol. 1, pp. 239-250, 1988.

[3] J. Komlós and R. Paturi, "Effect of connectivity in associative memory models," *Tech. Report*, CS88-131, University of California, San Diego, 1988.

[4] S. S. Venkatesh, "Robustness in neural computation: random graphs and sparsity," submitted for publication.

[5] Y. Baram, "Nested neural networks and their codes," talk presented at the *IEEE Int'l Symp. Inform. Theory*, San Diego, 1990.

[6] W. W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115-133, 1943.

[7] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554-2558, 1982.

[8] E. Goles and G. Y. Vichniac, "Lyapunov functions for parallel neural networks," in *Neural Networks for Computing*, (J. Denker, ed.), AIP, 1986.

[9] S. S. Venkatesh and D. Psaltis, "Linear and logarithmic capacities in associative neural networks," *IEEE Trans. Inform. Theory*, vol. vol. IT-35, pp. 558-568, 1989.

[10] L. Personnaz, I. Guyon, and G. Dreyfus, "Information storage and retrieval in spin-glass like neural networks," *Jnl. Phys. Lett.*, vol. 46, pp. L359-L365, 1985.

[11] J. Kahn, J. Komlós, and E. Szemerédi, "On the determinant of random ±1 matrices," preprint.

[12] S. S. Venkatesh, *Linear Maps with Point Rules: Applications to Pattern Classification and Associative Memory.* Ph.D. Thesis, California Institute of Technology, 1987.

# EMPIRICAL INVESTIGATIONS INTO THE EFFECTS OF SPARSITY ON NETWORK CAPACITY

Joel Ratsaby and Santosh S. Venkatesh
Department of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104

*Email address: jer@pender.ee.upenn.edu*

Joel Ratsaby
Electrical Engineering Dept.
University of Pennsylvania
EE999 Project
April 23, 1990

## (1) Abstract

Areas in the brain that are believed to carry out the functions of pattern separation, categorization, and associative memory, exhibit variable amounts of sparsed connectivity between neurons. In Artificial Neural Networks, one of the parameters that distinguishes an architecture from another is the connectivity of the units. In feedforward layered networks, the connections are asymmetrical, and the information flows from the input layer through several hidden layers to a single output layer. Backpropagation is a learning algorithm for feedforward layered nets which (due mainly to the fact that it operates on networks that contains hidden units) can discover useful internal representations and thus can be a powerful tool for attacking difficult problems like image and speech recognition. Several papers have reported the correlation between the number of hidden units and the learning ability of the BP, but did not report any relationship between the number of connections between the units to properties such as learning times, memory capacities. In this paper I report results of computer simulations of learning with the BP algorithm on layered networks which differ in their connectivities. The results include comparison of learning rates between different levels of sparsed networks and the percentage of examples needed to reach the critical point in which a neural network is said to have learned the function in a probably approximately correct sense (PAC). These results can serve as a proposal for a more theoretical research on the learning capacity of layered networks.

## (2) Introduction

Neural nets have various parameters that influence the computational characteristics, mainly the number of neurons, number of hidden neurons (in the case of layered nets), activation functions (linear, semi-linear, non-linear threshold), deterministic/probabilistic output decision rule, and connectivity. In real biological networks, sparsness of connections is very common (basically due to the physical space limitation that make fully connected networks with millions of neurons impractical) and it is clear that sparsed networks do work. Is it better or worse (and by which factor) to have sparsed networks ? This is not a trivial question. First we have to identify variables of the neural network in question, and ask how does sparsness relates to them. In addition we have to divide the variables into two groups: the first includes variables that are independent of the learning algorithm that is used in teaching the neural-net, and the second includes the variables that are algorithm-dependent. One important algorithm-dependent variable is the learning time (or solution-convergence time). The most important algorithm-independent variable is the capacity of a neural network. In a fully connected network (like: Hopefield, 1982) capacity is the number of vectors that can be stored in a network and recalled reliably; in feedforward layered nets, capacity reflects the number of functions (input-output mappings) that can be realised by the network. The exact definition of capacity also depends on how do we want to define "reliably" in the above. I will follow the definition of learnability-capacity which is best described by the following: In the formal definition of probably approximately correct (PAC) learning, we start with some underlying function, and pick examples at random from this very same function. Specifically, we pick points in the input space randomly according to some probability distribution and generate ordered pairs (x_1, f(x_1)), ... , (x_M, f(x_M)) from the function f. The question is how many examples do we need to train the network on, before we can say that it has learned the function (L.G. Valiant, Nov 1984). This answer is tied to the capacity of the network architecture (it is proportional to, but larger than the capacity). Note, however, that we have assumed that the function f can be realised within the network architecture (by architecture I mean the number of neurons and the specific link connectivity between them); if it can't.

then the number of examples needed to be shown to the network will never be finite since the network can't learn to begin with.

The results that I will present will compare the learning times specifically to the Backpropagation algorithm on different sparsed networks, and also will compare the number of examples that are needed to be shown during the learning cycle for different sparsed networks, to achieve learnability. But before I present the results, I will give an overview of experimental and theoretical results that have been reported in the field.

## Computational functions of the hipocampus

This section describes experimental results and system level theory of the hipocampal functions; I will focus on the variety of sparsed real neural-nets architectures that this region in the brain has, and explain how the different levels of sparsness contribute to different important characteristics. The hipocampus is one of the oldest parts of the brain. It gets inputs from many different areas of the cortex, including the cerebral cortex, parietal cortex, the temporal lobe visual and auditory areas, and the frontal cortex. Effects of damage to the hipocampus show that the very long term memories are not influenced. Different experiments (Squire, 1986; Squire & Zola-Morgan, 1988) have repeatedly shown that the the hipocampus plays a vital role in the storage of declarative memories such as episodic memory and semantic memory (hiearchy of facts). Within the hipocampus, there is a three stage sequence of processing consisting of granule cells (which receive from the entorhinal cortex), the CA3 pyramidial cells, and the CA1 pyramidial cells. The CA3 area has an extensive recurrent structure with a relatively large contact probability (4.3% in the rat). It is believed that some sort of an autoassociation memory matrix (Kohonen, Oja, & Lehtio, 1981) is being represented there. Kohonen (1972) as shown that the probability of a connection between neurons must not be very low in order to maintain a large signal to noise ratio in the retrieval of an output vector from an associative matrix. This would answer very well why in the CA3 region the ratio of the number of neurons to the number of their inter-connections is relatively low i.e. having less neurons allows to have high connectivity which improves the quality of the associations. In the stage preceding the hipocampus the axons pass via a competitive network. Its function is to accept non-orthognal input vectors and orthogonalize them; Kohonen (1972) showed that an associative network has larger fidelity of recall for stored vectors which are closer to being fully mutually orthognal. Another preceding stage is the mossy fibers system connecting dendate gyrus cells to the CA3 cells. This system is characterized by very low link probability and is used to decorrelate input patterns; in the rat, the probability of a link in this region is 0.000078. Pattern separation can be achieved using this type of a sparsed network since the probability is very high that each CA3 cell is influenced by a different assembly of dentate granuale cells.

## Effect of Sparsness to Capacity in Hopefield-like Networks

Associative networks based on the architecture of the Hopefield Network have linear threshold units fully interconnected. Two popular schemes of storing methods are the outer-product method (J.J. Hopefield, April 1982), and the spectral method for construction of the weight matrix (Venkatesh/Psaltis, 1989). The relatively simple construction of the linear transformation matrix (whose elements are the weights of the connections between the neurons) by means of the outer-products method yields a storage capacity of N/(4 logN) stable memory states (where N is the number of neurons). The spectral algorithm yields storage capacity which is linear in N. However in contrast to the fully connected networks, nested neural networks (nested means not-fully connected and structured in tree-like architecture) who have the same characteristics as the Hopefield network, have been shown to have vastly greater number of stable memory states than fully connected networks programmed in either of the above schemes (Yoram Baram, March 1989). These type of networks resemble the fractal forms studied by Mandelbrot (B. Mandelbrot, 1983), but differ in allowing connections not only between neigboring layers but between all layers through several neurons shared in common. All nested networks require considerably less connections than fully connected ones. In one particular architecture having 1000 neurons divided into subnets each having 8 fully connected neurons, it was shown that the number of stable memory states is

$2^{(1006/7)}=1.8294\times10^{43}$ (given that the states in each subnet is selected to satisfy certain requirements assuring stability and error correction). For randomly chosen vectors that are to be stored, it was shown that the probability of picking a vector orthognal to a vector which is already stored, is inversely proportional to the square root of the number of neurons in a subnet ( a subnet being the elementary fully connected structure in the overall nested network). Since orthognal vectors satisfy the stability requirement for a memory vector, in nested networks consisting of relatively small subnetworks (i.e. subnets having small number of fully connected neurons), the orthogonality condition allows for the storage of stable vectors with relatively high probability. The capacity in less than ideal conditions (i.e. where the vectors stored in the subnets were only nearly orthognal) was also much higher compared to the fully connected Hopefield network; with a nested network of 81 neurons divided into 10 subnetworks, each having 9 fully interconnected neurons, it was possible to store 2048 stable vectors.

## Summary of existing experimental and theoretical results regarding the effects of sparsness on neural networks

Fully interconnected networks are not realisable when there are vast numbers of neurons, and it is evident in the brain that sparsed networks do work, and achieve extremely large numbers of memory state associations. In the above section, results of theoretical analysis have shown that not only can sparsed networks function reliably, but in certain architectures they improve the capacity over fully connected networks with the same number of neurons. It would be interesting to know what is the effect of sparsing feedforward layered networks; will the number of realisable input-output mappings increase ? If not, then how is capacity related to sparseness ?

The following sections will attempt to answer these questions.

## (3) Experimental results

In all the following experiments, I constructed 3 layer networks where sparsing was done only on the links that connect the input to the hidden layer. The following setup was used in every trial of the experiment:

(1) An original sparsed network is built according to a certain architecture (architecture being the number of neurons and a specific interconnection matrix. The main parameter that selects a given architecture is the probability (P) for the existence of a link connecting the input layer to the hidden layer); the weights assigned to the links are picked randomly. Then several binary vectors are placed individually at the input layer, and the network generates a real valued response at the output layer. The input-output pairs are then placed aside for use in the next steps.

(2) A sparsed network with same architecture (same P but different random weights for the links) is built. An Unsparse network is built (i.e. every neuron in the input layer connects to all neurons in the hidden layer, and every neuron in the hidden layer connects to all neurons in the output layer) with random weight assignment.

(3) Teach the unsparsed and sparsed networks on the vectors generated by the original sparsed net. Record the overall error versus the number of learning cycles (I fixed the maximum number of cycles). The reason that I generate vectors from an existing network, and teach them, instead of picking randomly a set of input-output pairs, is because not every mapping can be realised by the student networks; thus generating vectors to be taught by the same architecture-network as the original generating network guarantees that a solution exists for the mapping, and allows me to measure other interesting characteristics of the student network. As far as the unsparsed networks are concerned, I expect them to be able to learn any mapping that a sparsed, with the same number of neurons, generated.

## Sparseness versus Learning time

I ran the above procedure on a 20-10-2 layered network (20 inputs units, 10 hidden and 2 output units) with increasing values for the probability of a link (with P = 0.2, 0.3, 0.5, 0.7). From the Figure series (A) the following are evident:

(1) When the level of sparseness for the generating network is high (low P) both the unsparsed and sparsed learn very fast (Figure (A.1) . The explanation for that is as follows: the training vectors have very low variance (i.e. the output of each vector are very similar, within 1%); this is due to the very high sparsness which limits the output values of vectors to a narrow region in state space, since no matter which input vector we put, it will have the same effect as some input vector X which has many zeros; therefore many different inputs will be mapped to a vector X and will result in its output. So sparseness decreased the range of the input-output transformation, and there is now much less to learn (since all the vectors can be considered as a few distinct vectors), and that is why whoever is learning, will do very well fast.

(2) The variance in the output of the vectors that are to be taught, increases as sparsness of the generating network decreases, and therefore it takes the student networks more time to learn as the generating network becomes less sparse (since there are more distinct vectors to learn). There are several runs per figure, and one should look at the average behavior when judging the results. Note that the Backpropagation algorithm can get stuck in a local error minimum instead of converging to the global error minimum, and that is why some of the runs don't converge; this is the difficult part of trying to analyze characteristics of neural networks by studying them through an unperfect learning algorithm.

(3) On the average, sparsed networks learn faster than the unsparsed networks. In Figures (A.2),(A.3) the unsparsed is slower from the 0.2 sparsed by approximately 6%, in Figures (A.4),(A.5) the unsparsed is slower by 8.3%, in Figures (A.6),(A.7) the unsparsed is slower by 16%.

## Learning the 8-5-8 Encoder Problem

Here the taught vectors were not generated by an original network, but rather were:

| | Input | Output |
|---|---|---|
| Vector(1): | 10000000 | 10000000 |
| Vector(2): | 01000000 | 01000000 |
| Vector(3): | 00100000 | 00100000 |
| Vector(4): | 00010000 | 00010000 |
| Vector(5): | 00001000 | 00001000 |
| Vector(6): | 00000100 | 00000100 |
| Vector(7): | 00000010 | 00000010 |
| Vector(8): | 00000001 | 00000001 |

Eight different sparsed networks were built (each having 8 input units, 5 hidden units and 8 output units). As Figure (B.1) shows, the less sparsness the faster the learning. The explanation is as follows:

Sparsed nets have smaller capacity, because removing weights decreases the number of possible functions (mappings) that they can realise; this is simply because there are less variables to permute (i.e taking the view that each function is a permutation of weights); one might argue that since there are infinitely real numbers that weights may take, it would imply that removing weights will not reduce the number of realisable functions (since the remaining weights will still range over infinitely many values) but this is clearly wrong since it is not just the value of the link that characterizes a given function, but also its position in the overall architecture, and there is a finite number of positions that links can occupy (since there are finite number of neurons) therefore if enough links are removed to isolate an output neuron then although all other remaining links may take infinitely many values, they still cannot influence the isolated output neuron (therefore the number of functions realisable has been reduced). And for a given set of points (or vectors to be learned) it is much more probable to find a function that contains these points in its solution, in a network that can realise more functions to begin with (i.e. the unsparsed NN) whereas in the sparsed, the number of realisable functions is smaller, therefore the probability of finding a function (i.e. amongst the small number of realisable functions) that has the points in its solution set, is much smaller. Therefore, the unsparsed network can with a higher probability than the sparsed network, learn a given set of M vectors, therefore its capacity is larger than the sparsed.

In Figure (8.1), the function that was to be taught was not realisable by the very sparsed networks (as opposed to the previous section which dealt with teaching a realisable function to sparsed networks). That is the reason that the very sparsed networks were not able to learn; their capacity is too small. As we decreased the level of sparsness we in effect increased the capacities, and thus we see an improvement in the learning. This is because the function that we taught was realisable with a greater probability in the less sparsed networks.

In summary, the above two sections imply the following: a sparsed network will learn faster than an unsparsed when its architecture can realise the function that it is trying to learn. It learns faster because its architecture permits it to realise much fewer functions and when given M distinct points (vectors) it needs to sort out less possible functions (in contrast to an unsparsed network) that contain these points in their solution, and thus completes the sorting faster than an unsparsed. However, when given an unrealisable function (or function that is only nearly realisable) the sparsed network either cannot learn or learns slower than the unsparsed network (this follows from the same argument).

## Experimental results on capacity of feed-forward layered networks

In all the following experiments, sparsing was done only on the links that connect the input to the hidden layer. The following setup was used in every trial of the experiment:

(1) An original sparsed network is built according to a certain architecture interconnection matrix. The main parameter that selects a given architecture is the probability (P) for the existence of a link connecting the input layer to the hidden layer); the weights assigned to the links are picked randomly. Then several binary vectors are placed individually at the input layer, and the network generates a real valued response

at the output layer. The input-output pairs are then placed aside for use in the next steps.

(2) A sparsed network with same architecture (same P but different random weights for the links) is built. An Unsparse network is built.

(3) Teach the unsparsed and sparsed networks on the vectors generated by the original sparsed net, showing a randomly selected percentage of the vectors, and repeatedly learning on the range of (20%,90%). Record the overall error versus the number of learning cycles (I fixed the maximum number of cycles). The reason that I teach different percentage of the points that define the taught function, is because this will determine the critical percentage point which is (according to PAC learning) proportional to the capacity of the system that is doing the learning (in this case it is a neural network).

I started with a 5-4-2 architecture (Figure Series D) . I built a 0.2 probability original sparsed network with which I generated 32 binary-input real- output vectors. I built a 5-4-2 sparsed with the same connectivity and a 5-4-2 unsparsed and trained them on 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% of the above generated vectors (averaged over 6 runs per trial e.g. in the 50% trial it means that the number of vectors shown is 16. but the set of vectors shown in each case can be different 16 vectors). I recorded the resulting Error versus Number of learning cycles versus Percentage shown. I repeated the above for a 0.4, 0.5, 0.6, 0.8, 0.9 probability generating sparsed network.

When comparing the critical points for the unsparsed network with that of the corresponding sparsed network that learned the vectors generated by the same architecture network, we see that always, the unsparsed critical point is higher than the sparsed (as we expect). In Figures (D.6) and (D.7), I ran 0.5 sparsed networks that learned 4 different sets of 0.5-sparsed-generated vectors; it is evident that in all 4 cases, the critical point is approximately the same (50% shown). This result goes nicely with the expectation that the capacity (and thus the critical point which is proportional to it) is independent of the problem learned. Figure (E.1) shows the Error vs Percentage-Shown at the 700th learning cycle. The point where the error starts to follow the asymptote (i.e. where the derivative is relatively small), is the critical point. The relationship of sparseness to critical point for the 5-4-2 is as follows:

| Probability of a link | Critical Point: |
|---|---|
| 0.2 | 30% |
| 0.4 | 40% |
| 0.5 | 50% |
| 0.6 | 50% |
| 0.8 | 60% |
| 0.9 | 70% |

I repeated the above with a 6-4-1 architecture (Figure Series F); there were $2^6=64$ generated vectors to be learned. I only display (Figure F) as an example of one sparsed network (0.5 sparsed). In reality, 6 different sparsed networks were built for each percentage (20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%) of each sparsed case (i.e. for 0.2, 0.4, 0.5, 0.6, 0.8, 0.9). In total, 288 networks were built in this experiment. Then I averaged the 6 responses per a specific percentage, and plotted the graphs (Figures F.1 - F.4). As seen in Figure Series (F), the relationship of sparseness to critical point for the 6-4-1 is as follows:

| Probability of a link | Critical Point: |
|---|---|
| 0.2 | 40% |
| 0.4 | 50% |
| 0.5 | 50% |
| 0.6 | 35% |
| 0.8 | 45% |
| 0.9 | 60% |

I repeated the above with a 7-7-1 architecture (Figure Series G); there were $2^6=64$ generated vectors to be learned. In total, 288 networks were built in this experiment. Then I averaged the 6 responses per a specific percentage, and plotted the graphs (Figures G.1 - G.4). As seen, the relationship of sparseness to critical point for the 7-7-1 network is as follows:

| Probability of a link | Critical Point: |
|---|---|
| 0.2 | 40% |
| 0.4 | 50% |
| 0.5 | 40% |
| 0.6 | 50% |
| 0.8 | 70% |
| 0.9 | 80% |

Figure (K.1) shows the results of the critical point measurements for the 5-4-2, 6-4-1, 7-7-1 architectures.

In the above, each network was learning a problem of different size, i.e. the 5-4-2 learned a function that had 32 input-output vectors, the 6-4-1 learned a function that had 64 input-output vectors, the 7-7-1 learned a function that had 128 input-output vectors. In order to compare networks of different architectures that learn the same size problem, I ran the following experiment. I repeated the above 3-step procedure with an 8-8-1 and a 10-10-1 learning a set of vectors that was generated by a 6-4-1 network, i.e. the problem had 64 input-output vectors. In total, 576 networks were built in this experiment. The results are shown in Figure Series (H), and Figure Series (J). The relationship of sparseness to critical point in the 8-8-1 network is as follows:

| Probability of a link | Critical Point: |
|---|---|
| 0.2 | 30% |
| 0.4 | 40% |
| 0.5 | 50% |
| 0.6 | 50% |
| 0.8 | 60% |
| 0.9 | 70% |

And for the 10-10-1 network:

| Probability of a link | Critical Point: |
|---|---|
| 0.2 | 40% |
| 0.4 | 50% |
| 0.5 | 60% |
| 0.6 | 67% |
| 0.8 | 70% |
| 0.9 | 85% |

Figure (K.2) shows the results of the critical point measurements for the 6-4-1, 8-8-1, 10-10-1 architectures. Figure (K.3) shows the critical points as function of the maximum number of links.

## Section Summary

The following are evident from Figure Series (K):

(1) As sparseness decreases (increasing probability of a link beyond 0.5) the critical-point curves branch out from each other. This may suggest that the capacity is a linear function of sparsness with a slope determined by the number of units (the specific architecture) in the network.

(2) The plot (Figure K.2) of the networks that learned the same size problem was expected since the smaller network (6-4-1) should have a critical- point curve lower than the 8-8-1 (which is evident in the plot). But the plots of Figure (K.1) are not as trivial. I would expect the 5-4-2 network to have a critical-point curve lower than the 6-4-1 (which is not evident). This suggests that there may be some non-linear relationship between a specific-architecture network to its learnability of different size problems, thus allowing for a 6-4-1 network to learn a set

of 64 vectors with less percentage of examples than for a 5-4-2 network to learn a set of 32 vectors, and thus giving the false impression that the 6-4-1 has less capacity. In other words, when using an algorithm to measure the critical error, we must make sure that the initial conditions (in this case the problems being learned) are as uniform (i.e. of the same size)as possible across all the student networks.

## (4) Conclusions

This work was concerned in getting qualitative insight to the relationship of learning speed versus sparsness, and relative capacities versus sparsness. The experiments were carried out as computer simulations, and the majority of the results have been expected. The main points that can be suggested by the work are summarized below:

- A sparsed network will learn faster than an unsparse net (having the same number of neurons) when its architecture can realise the function that it is trying to learn. When given an unrealisable function (or function that is only nearly realisable) the sparsed network either cannot learn or learns slower than the unsparsed network.

- The capacity is approximately linear function of sparsness with a slope determined by the number of units (the specific architecture) in the network. The less sparsed the network, the greater the capacity and thus it is able to learn more types of input-output mappings, i.e. realise a bigger set of possible functions.

## (5) References

J.J.Hopefield (1982),"Neural networks and physical systems with emergent collective computational abilities" *Proceedings of the National Academy of Sciences* 79:2554-2558

L.G. Valiant (Nov 1984),"A Theory of the Learnable" *Communications of the ACM* 27:1134-1142

Squire, L. (1986) "Mechanisms of memory"*Science* 232:1612-1619

Squire & Zola-Morgan (1988) "Memory: Brain systems and behavior" *Trends in NeuroSciences* II:170-175

Kohonen, Oja, & Lehtio, (1981) "Storage and processing of information in distributed associative memory systems" In G.E.Hinton & J.A. Anderson, *Parallel models of associative memory* Hillsdale, NJ: Erlbaum.

Teuvo Kohonen (1972) "Correlation matrix memories" *IEEE Transactions on Computers* C-21:353-359

Venkatesh/Psaltis (1989) "Linear and logarithmic capacities in associative neural networks" *IEEE Transactions Information Theory*

Yoram Baram, March (1989) "Nested Neural Networks and their Codes" *Neural Networks*

B. Mandelbrot, (1983) *The fractal Geometry of Nature* W.H. Freeman and Company, New York

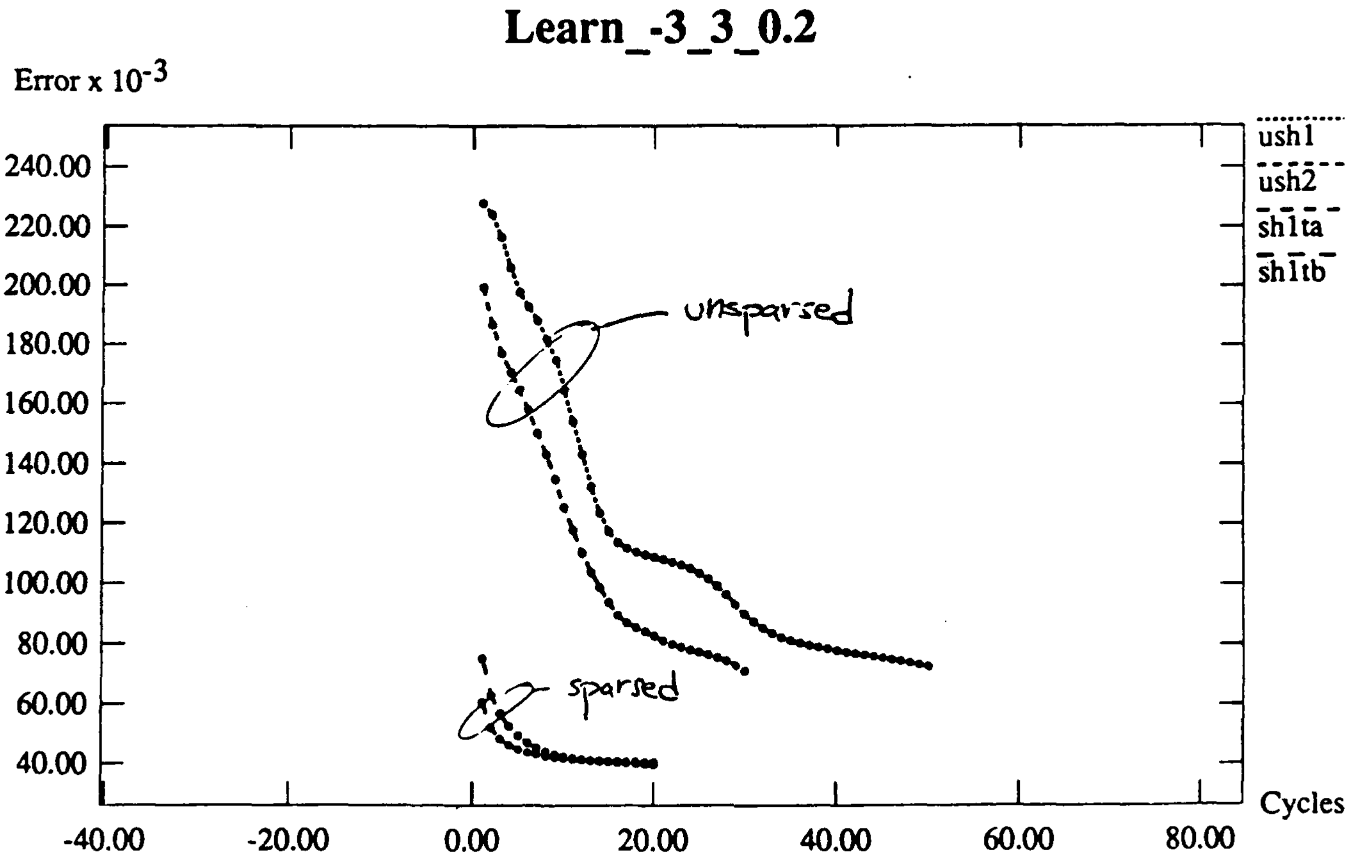


Figure A.1 : 20-10-2 Unsparsed and 0.2 sparsed learning.

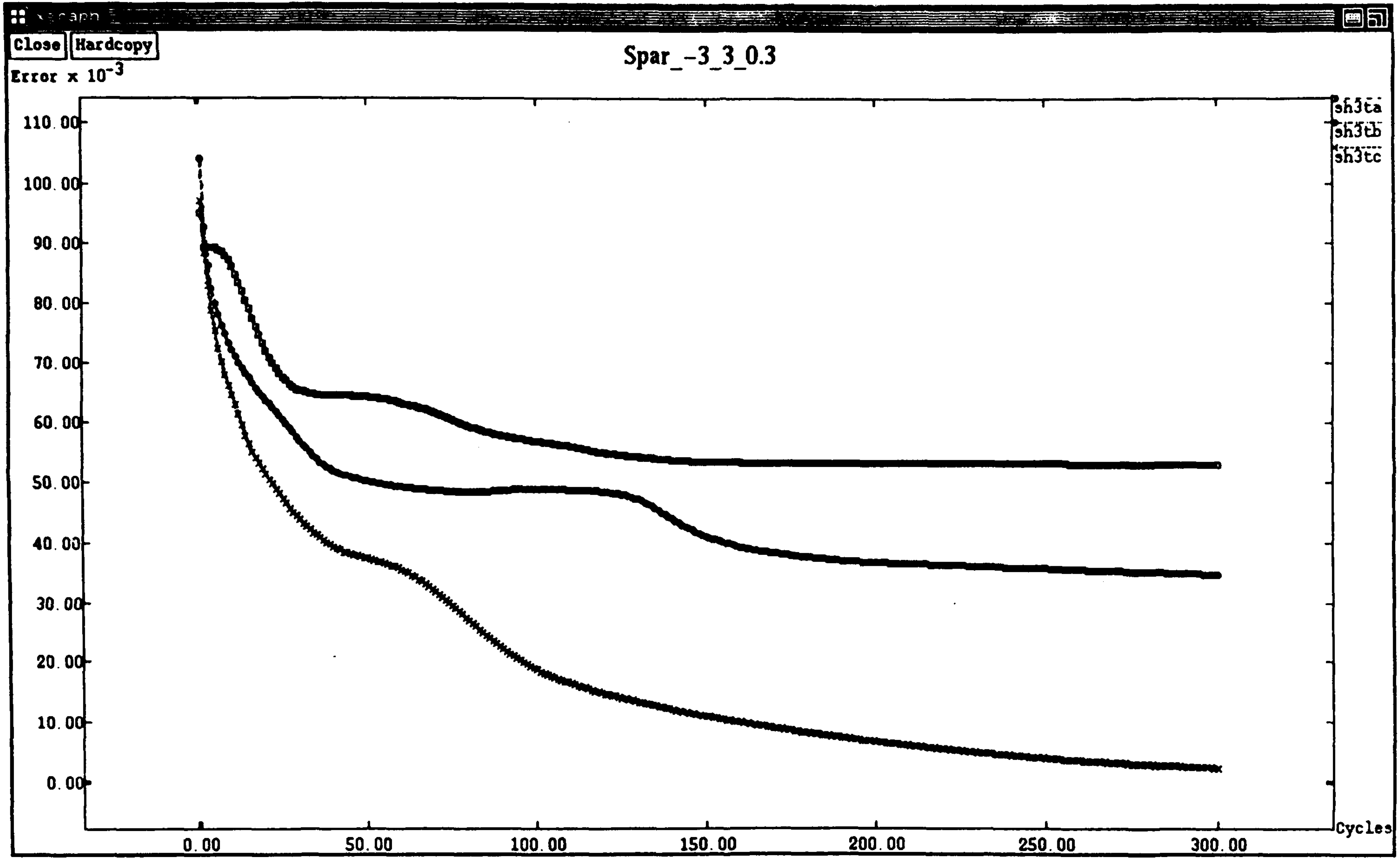


Figure A.2 : 20-10-2 0.3 sparsed learning

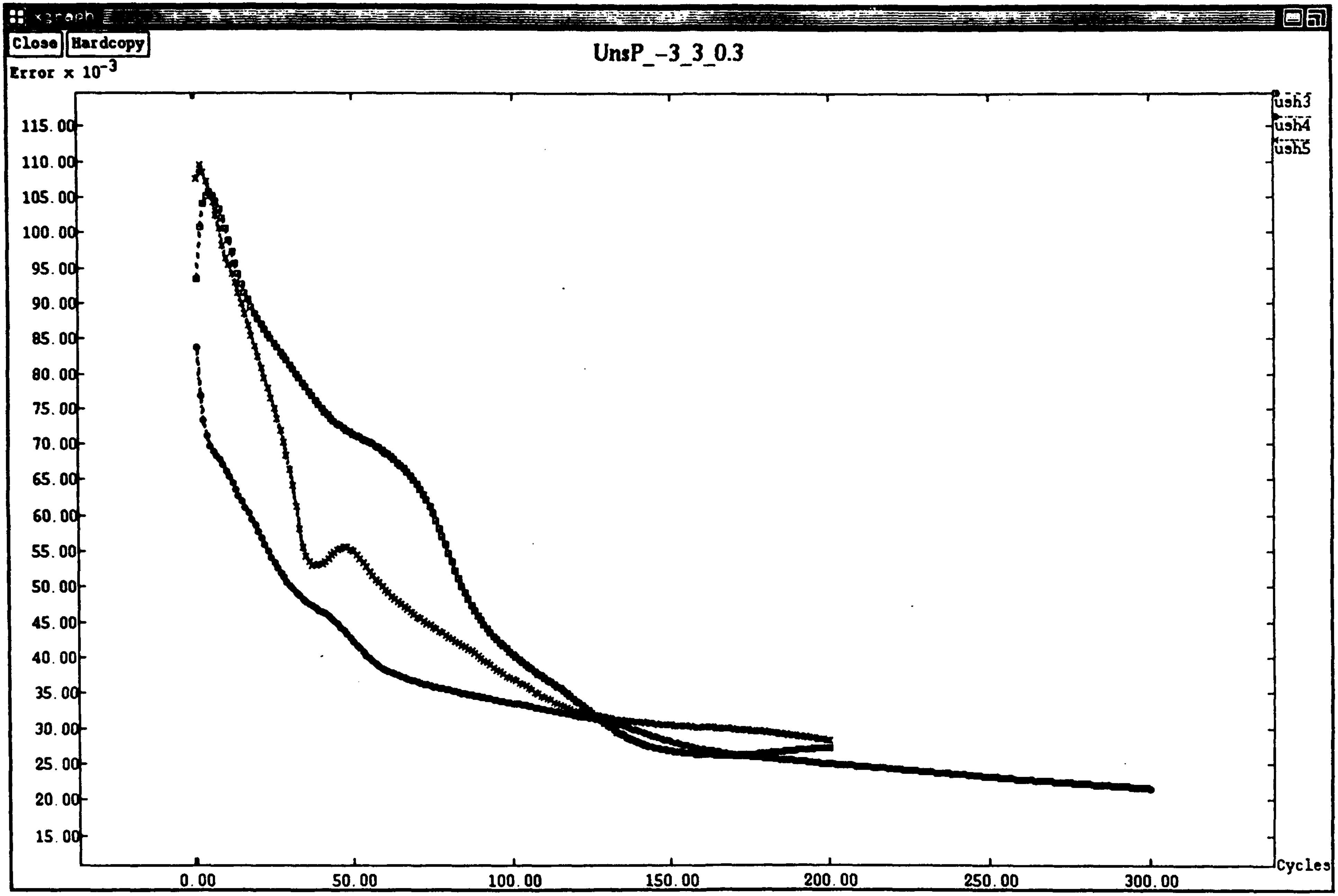


Figure A.3: Unsparsed Learning 0.3

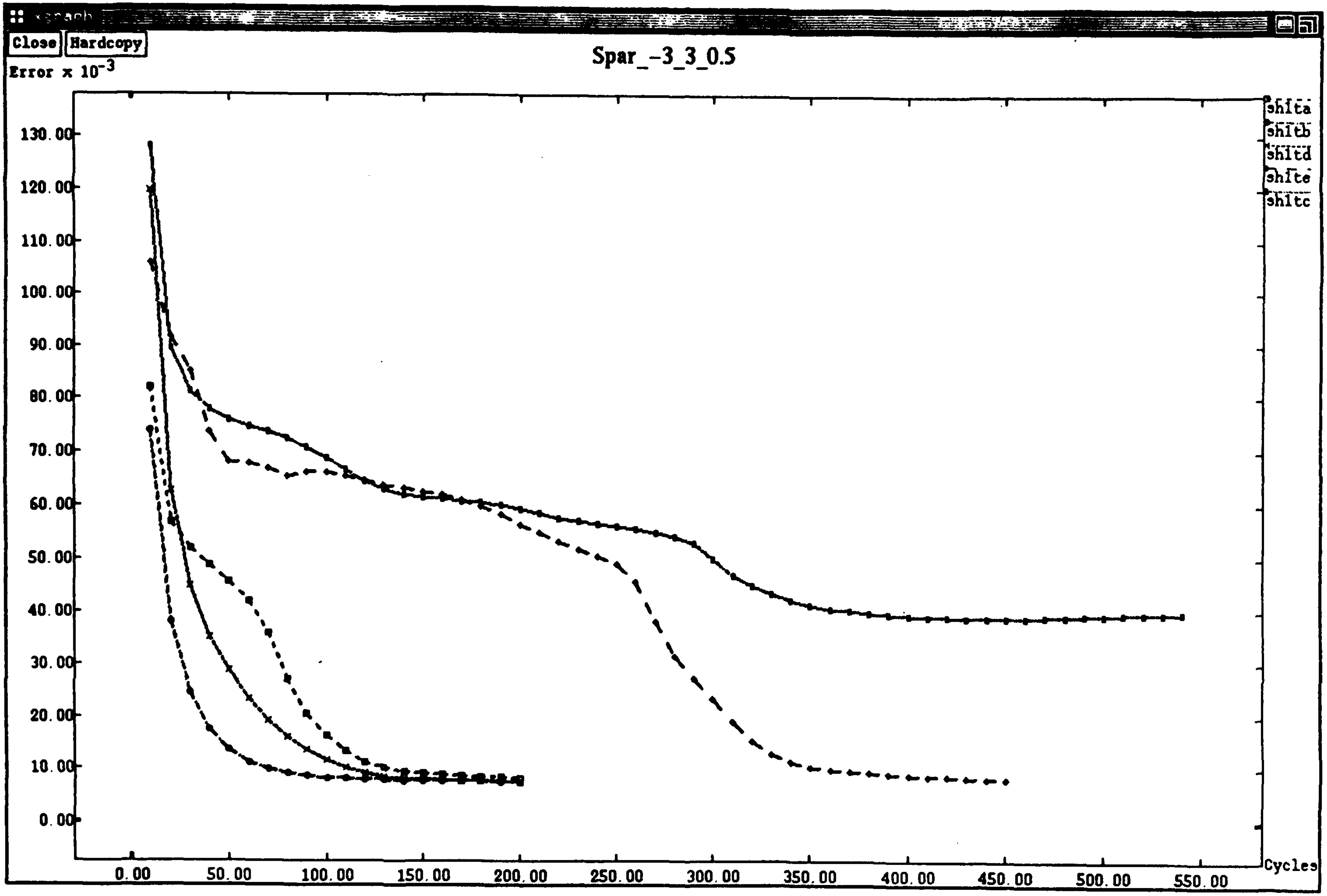


Figure A.4 : 20-10-2 0.5 Sparsed learning

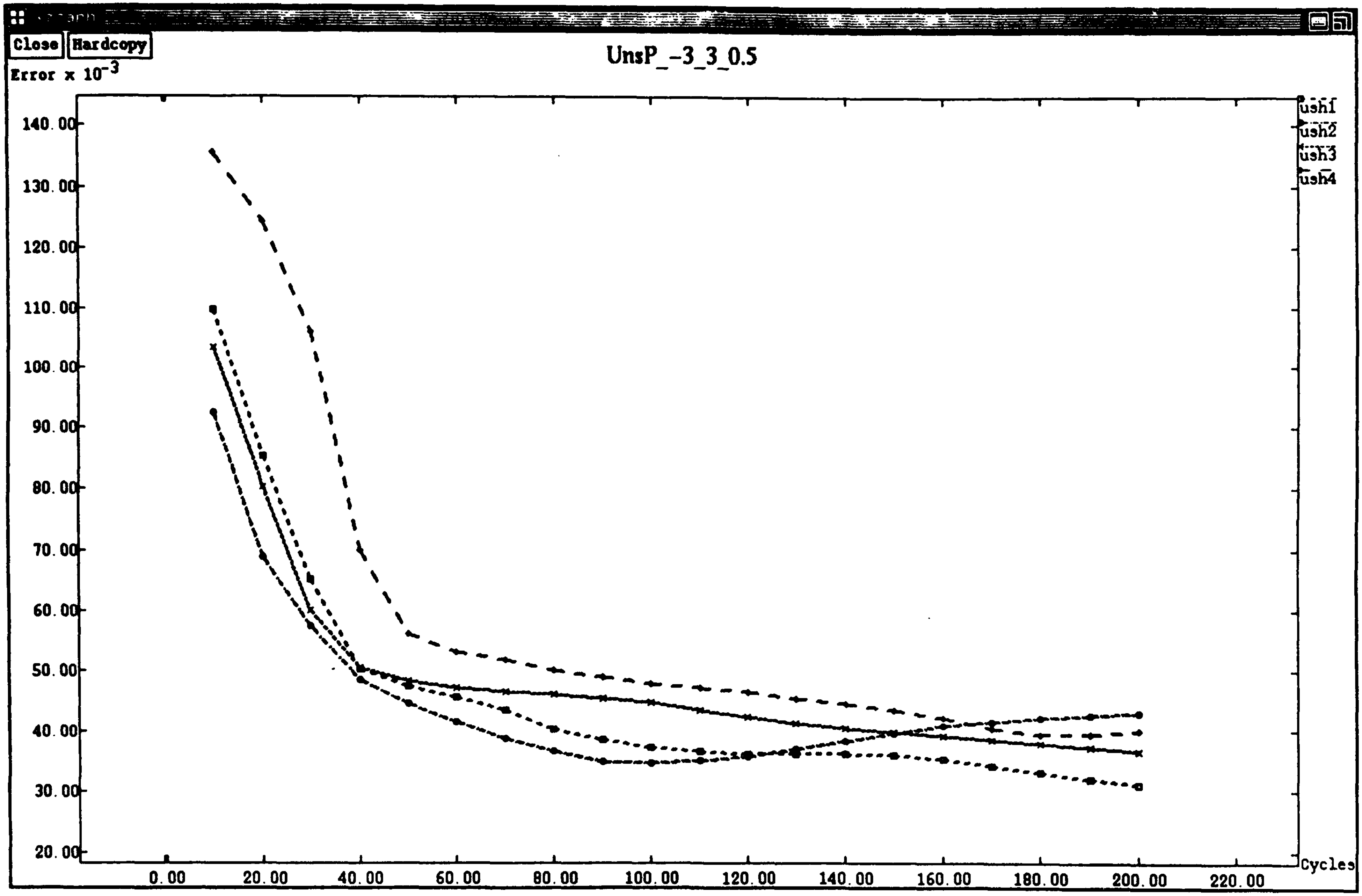


Figure A.5 : 20-10-2 Unsparsed learning 0.5

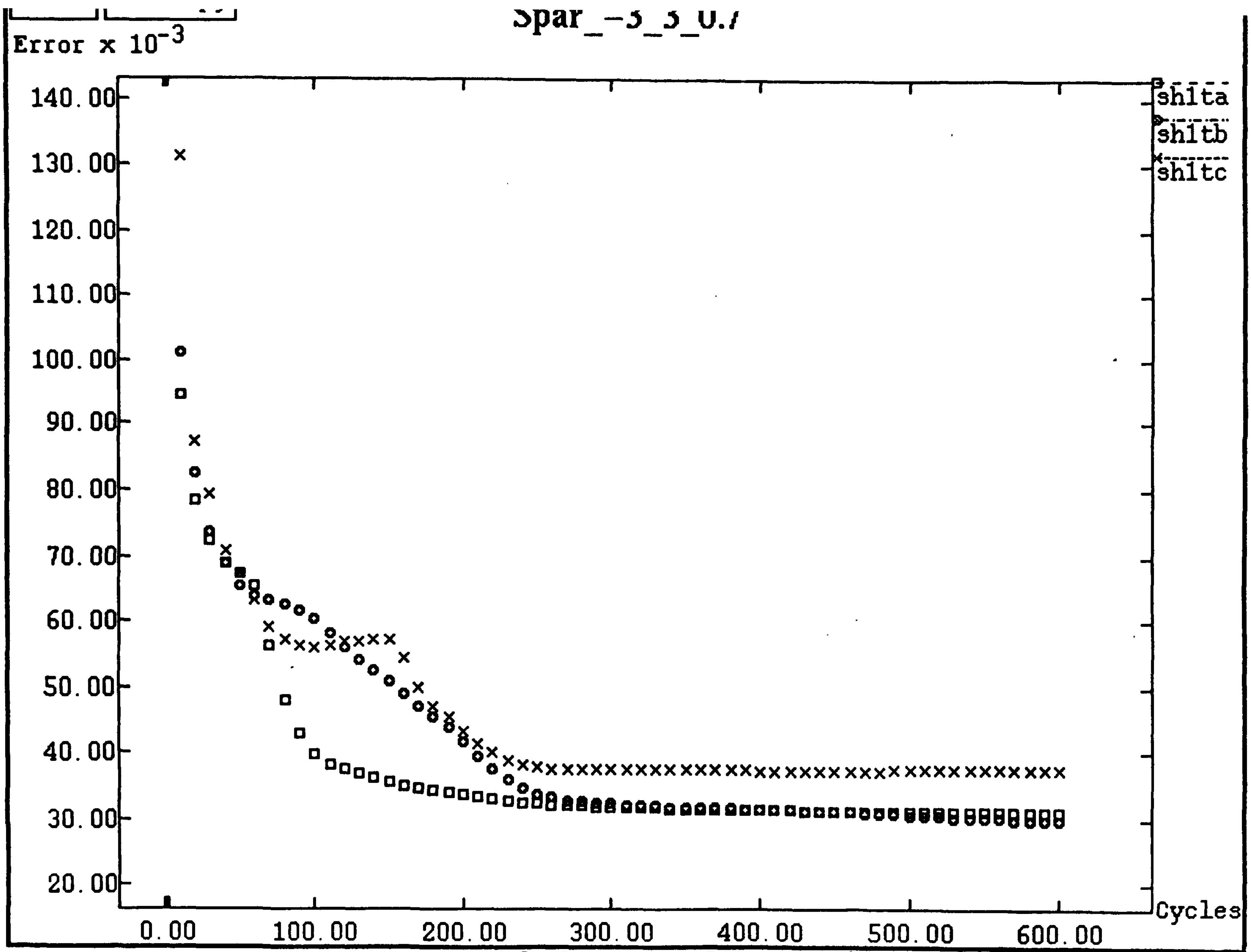


Figure A.6: 20-10-2 0.7 spaced learning

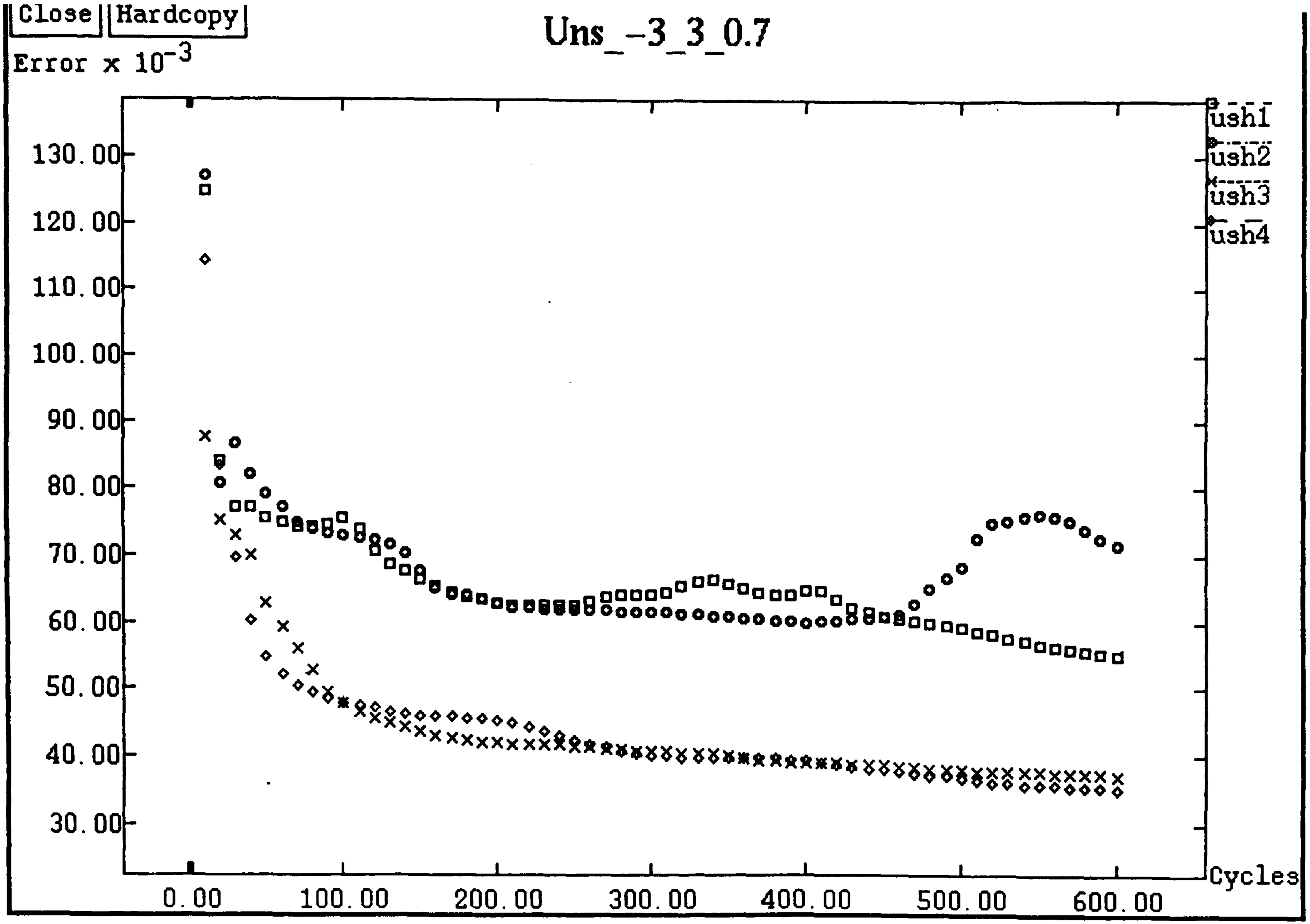


Figure A.7: 20-10-2 unsparsed learning 0.7

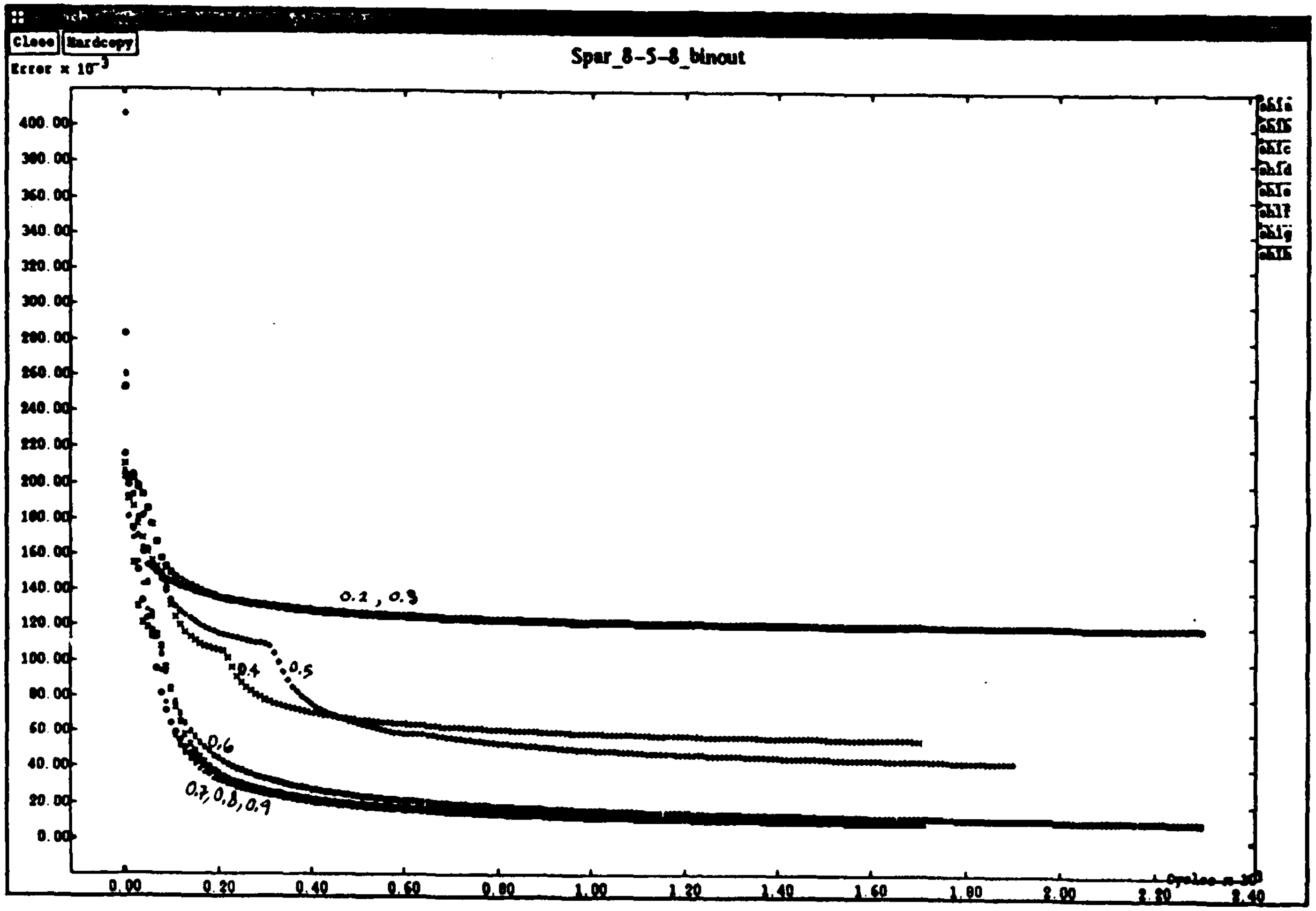


Figure B.1: Sparsed learning 8-5-8 Encoder

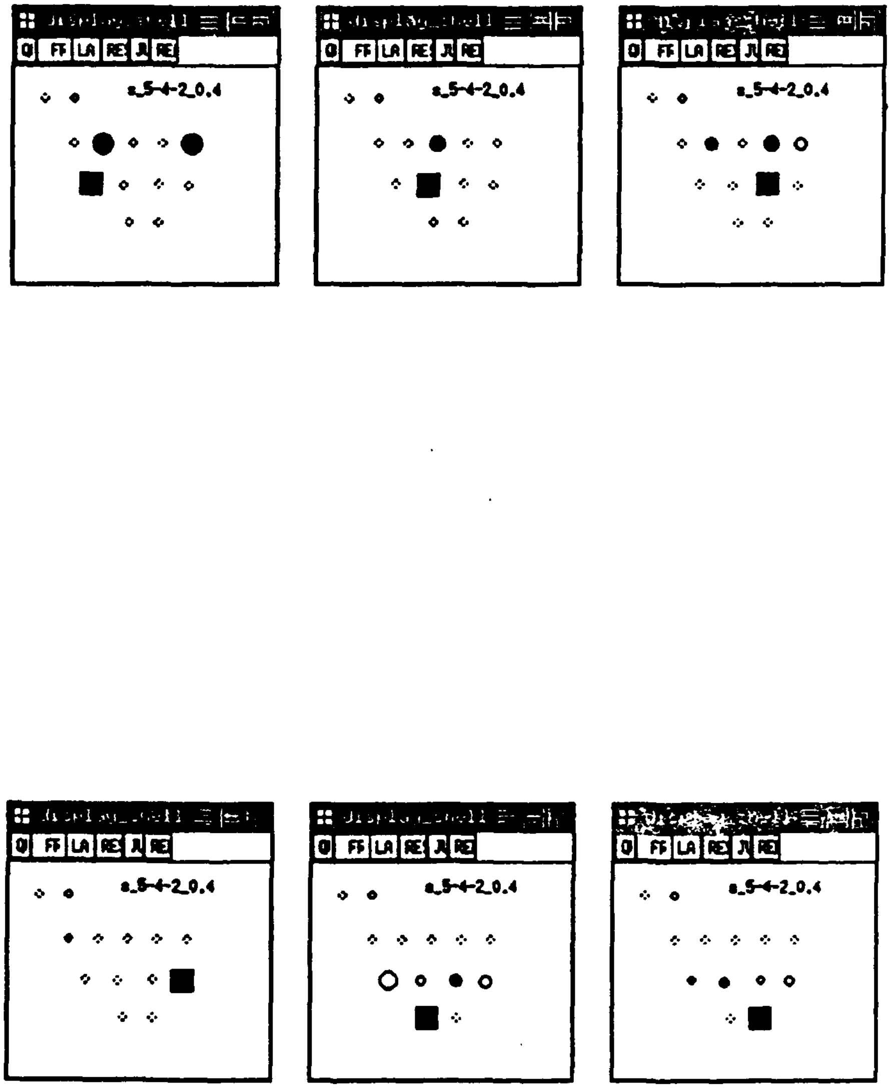
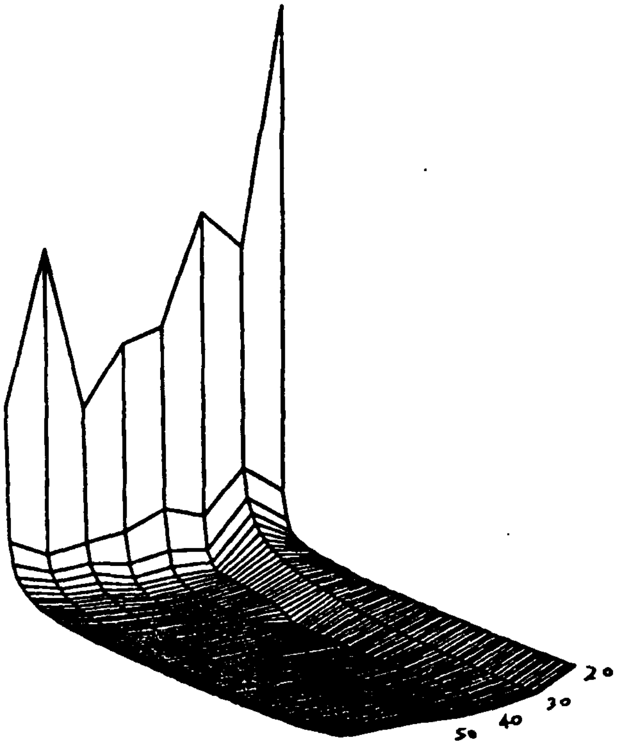
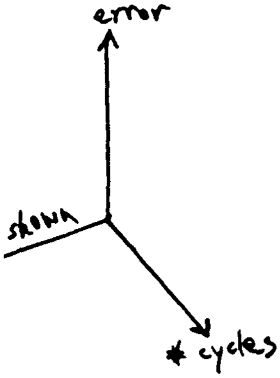


Figure D:Original Vector-Generating 0.4 Sparsed Net

Sparsed Learning 0.2 (overall error)





Unsparsed learning 0.2 (overall error)

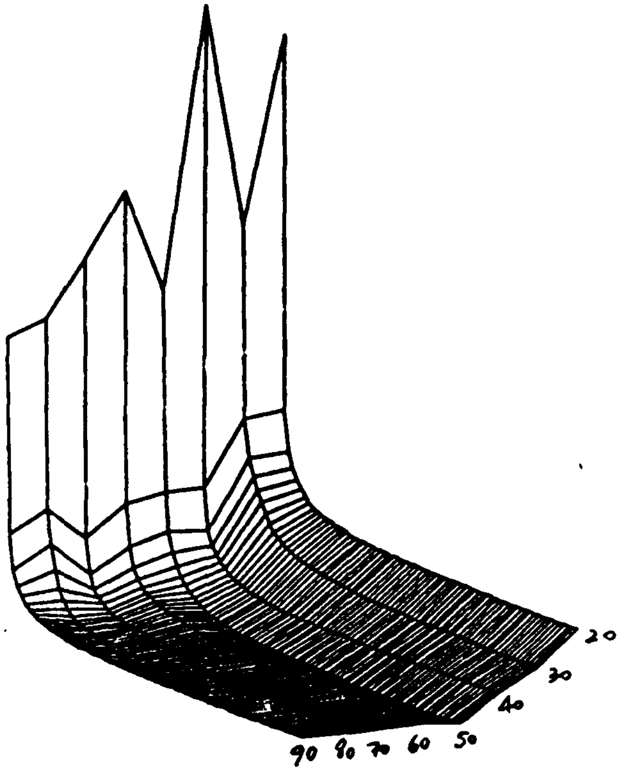
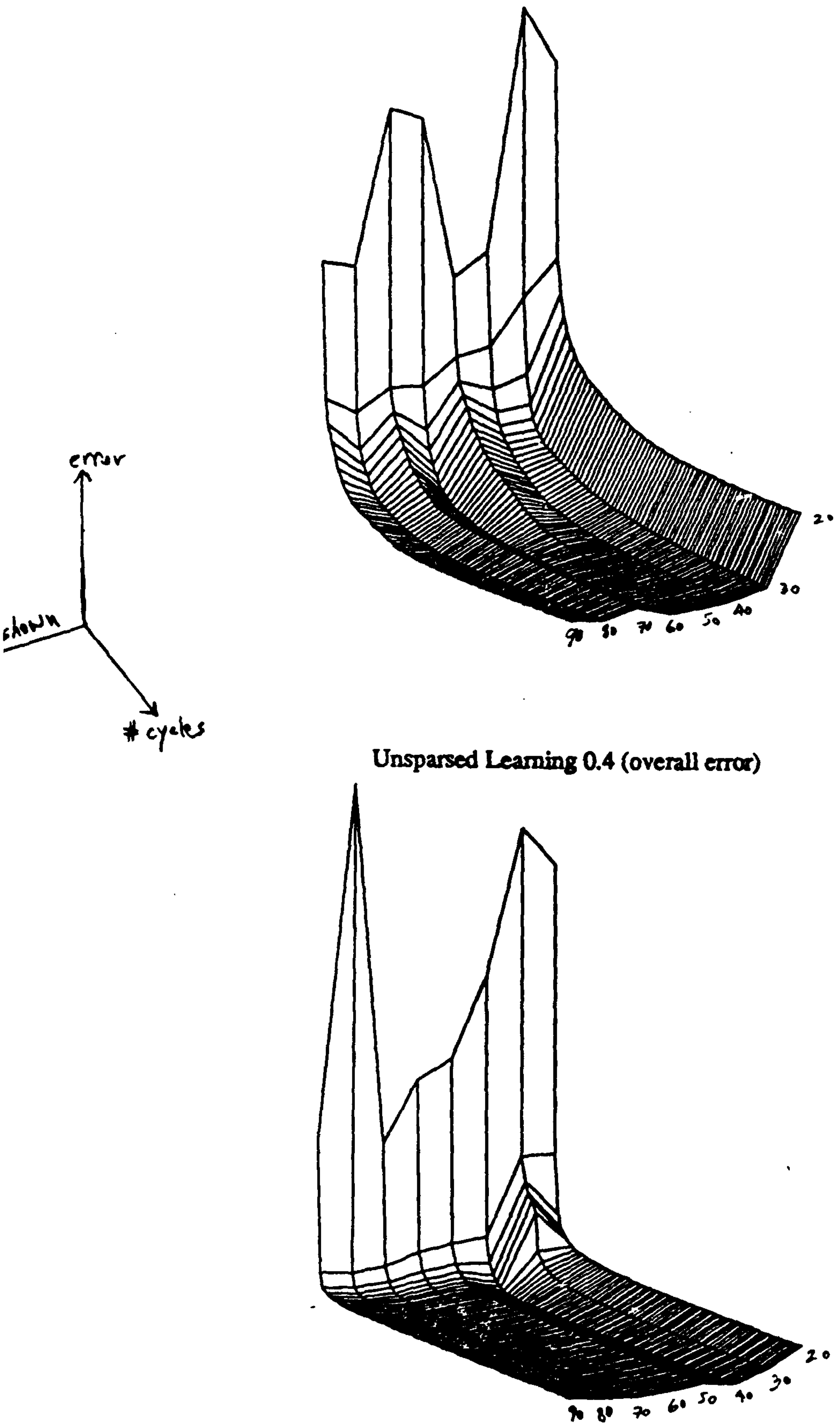


Figure D.1: 5-4-2 0.2 sparsed/unspared learning

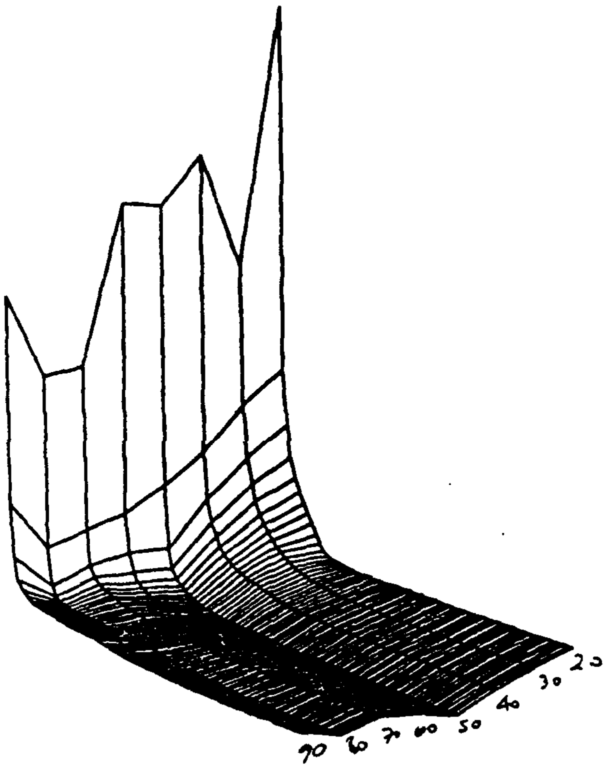


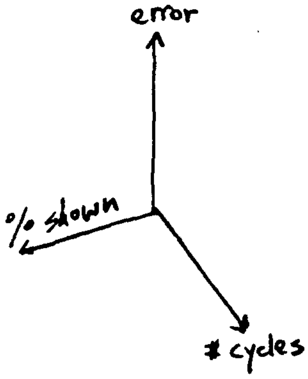
Sparsed Learning 0.4 (overall error)

Unsparsed Learning 0.4 (overall error)

Figure D.2: 5-4-2 0.4 Sparsed/Unsparsed Learning

Sparsed Learning 0.6 (overall error)





Unsparsed Learning 0.6 (overall error)

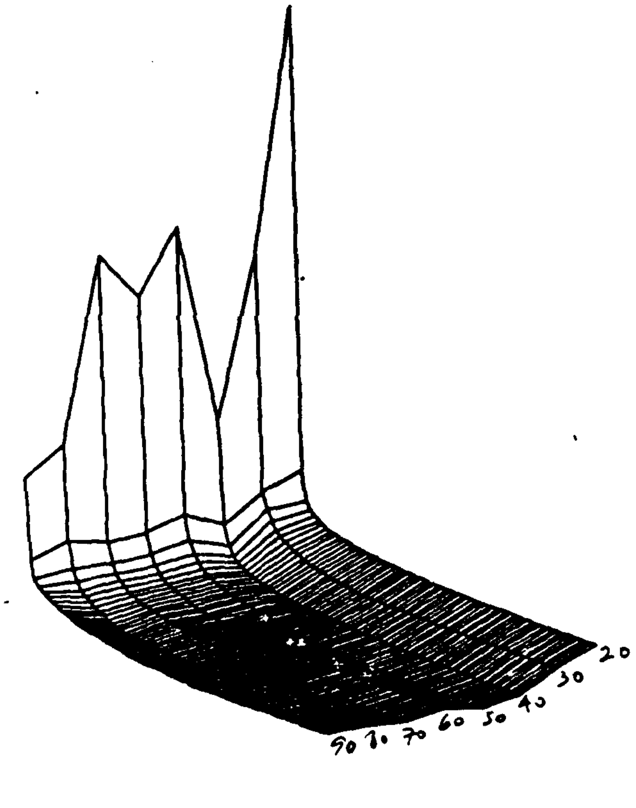
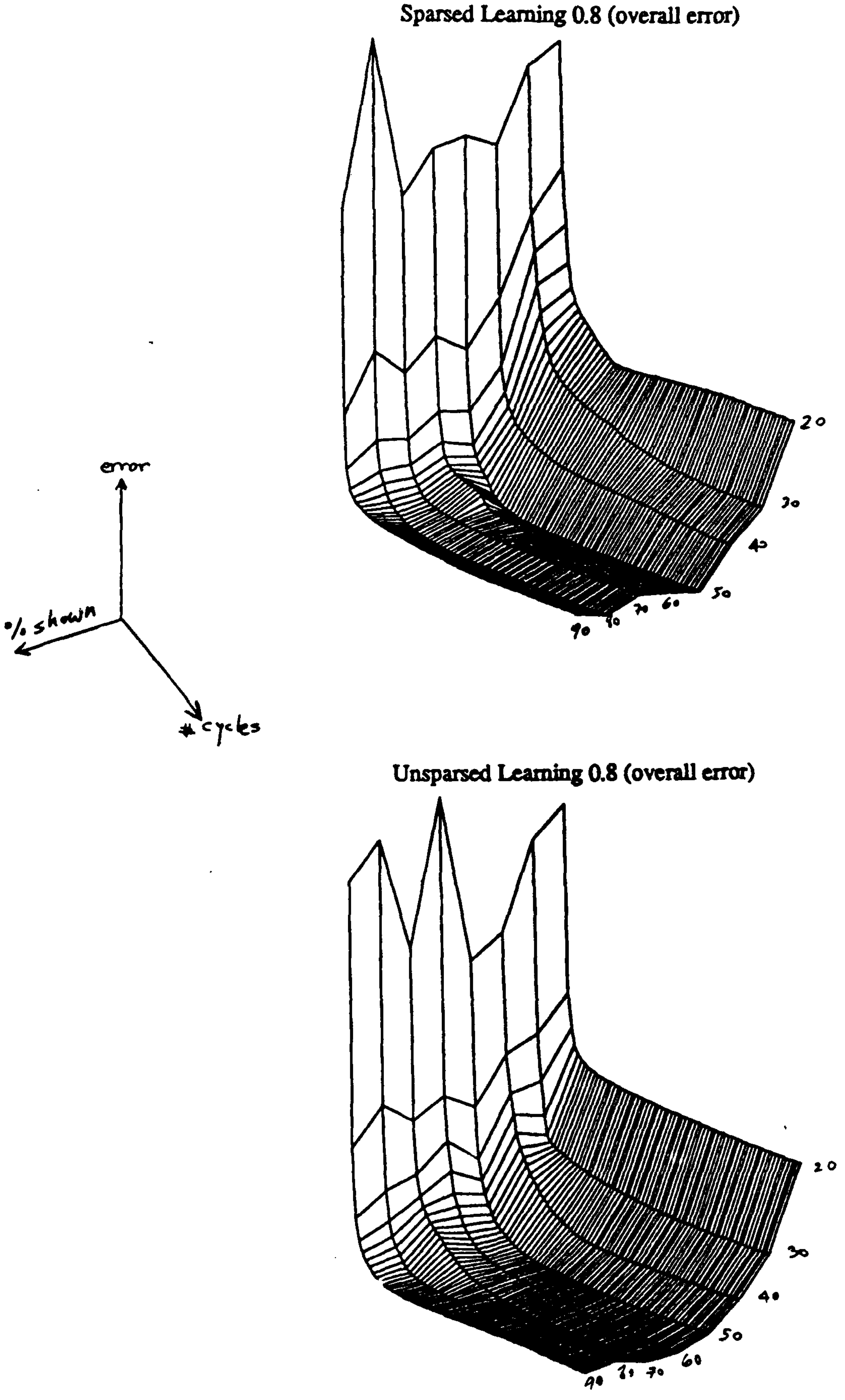


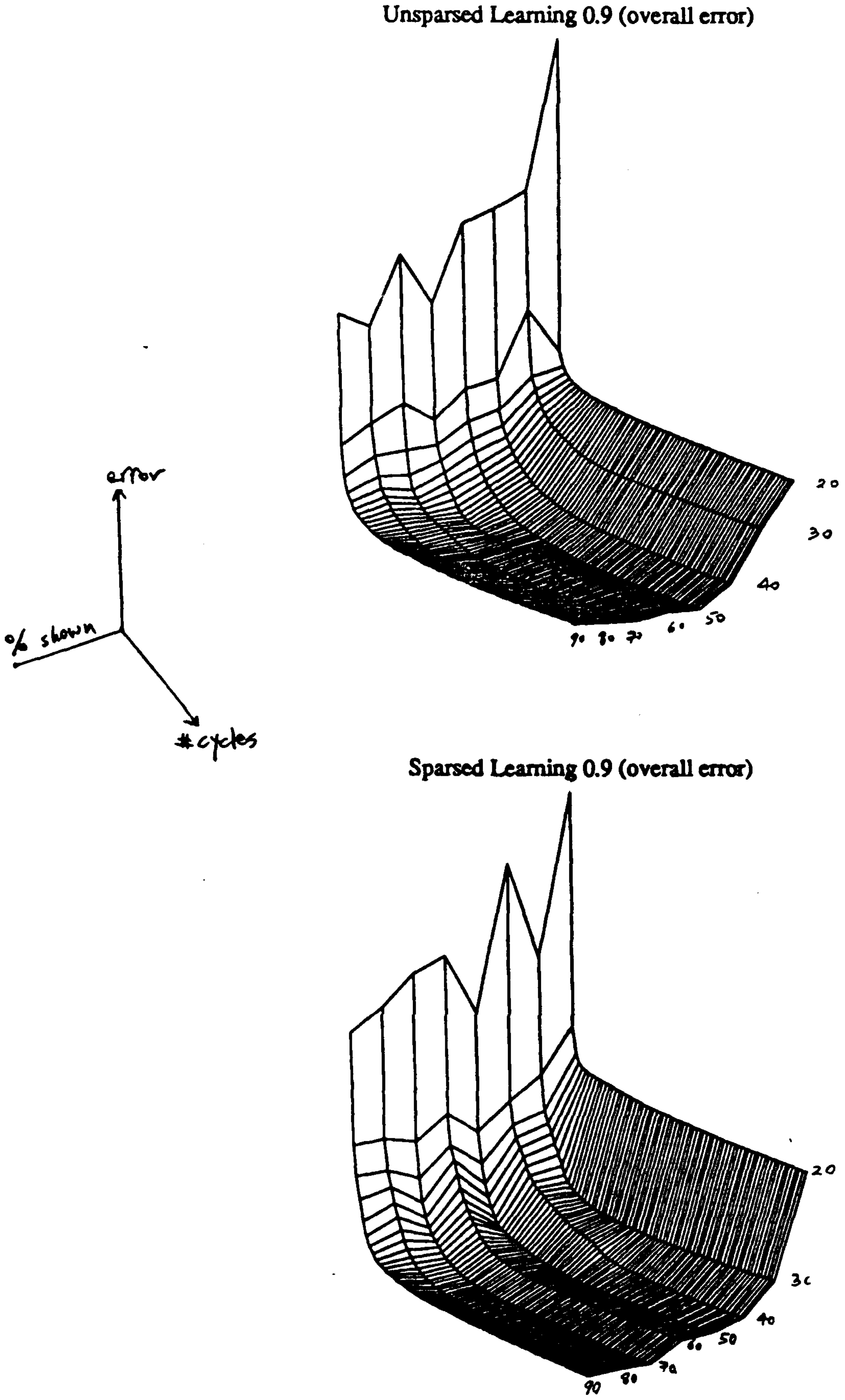
Figure D.3 5-4-2 0.6 Sparsed/Unsparsed learning



Sparsed Learning 0.8 (overall error)

Unsparsed Learning 0.8 (overall error)

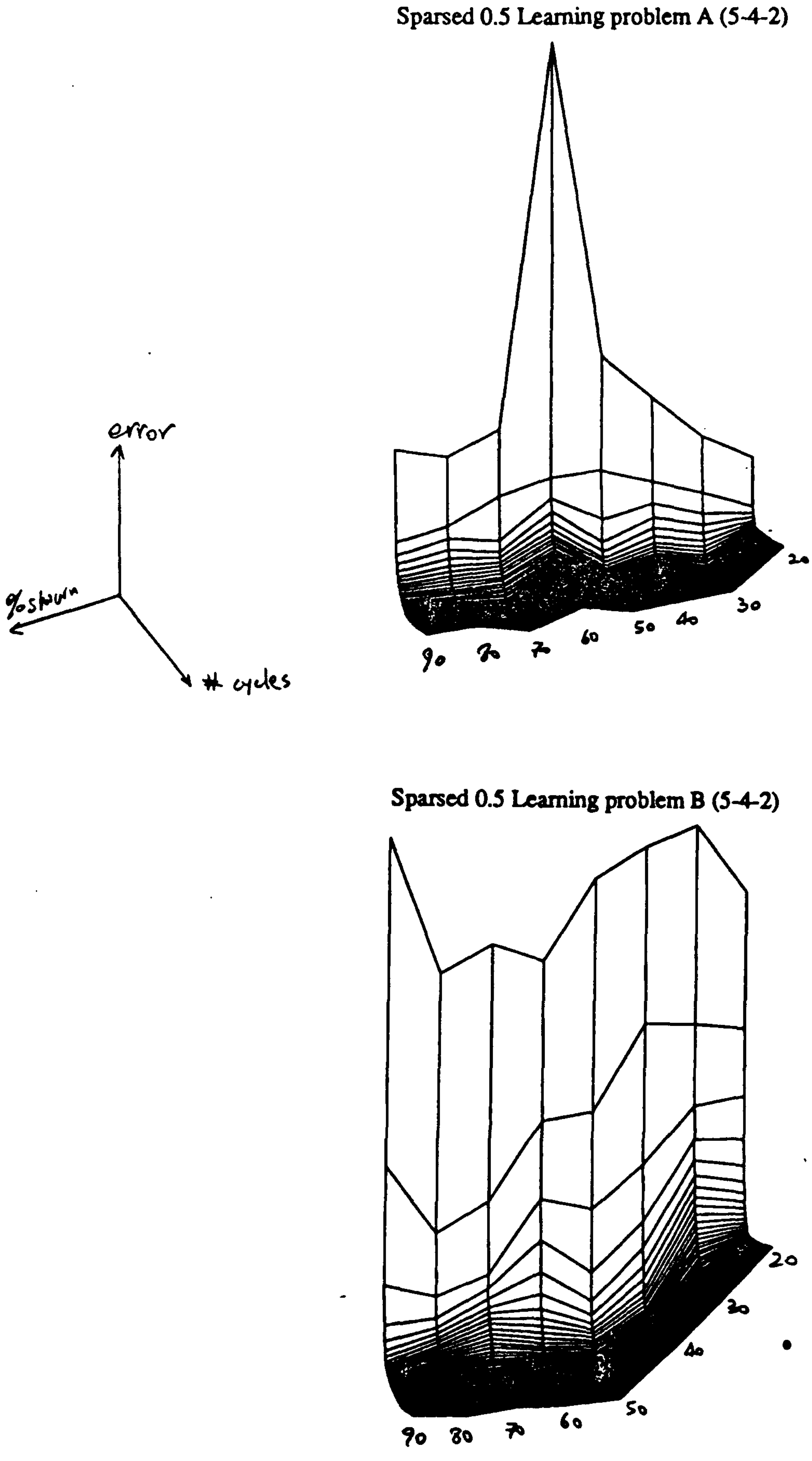
Figure D.4: 5-4-2 0.8 sparsed/unsparsed learning



Unsparsed Learning 0.9 (overall error)

Sparsed Learning 0.9 (overall error)

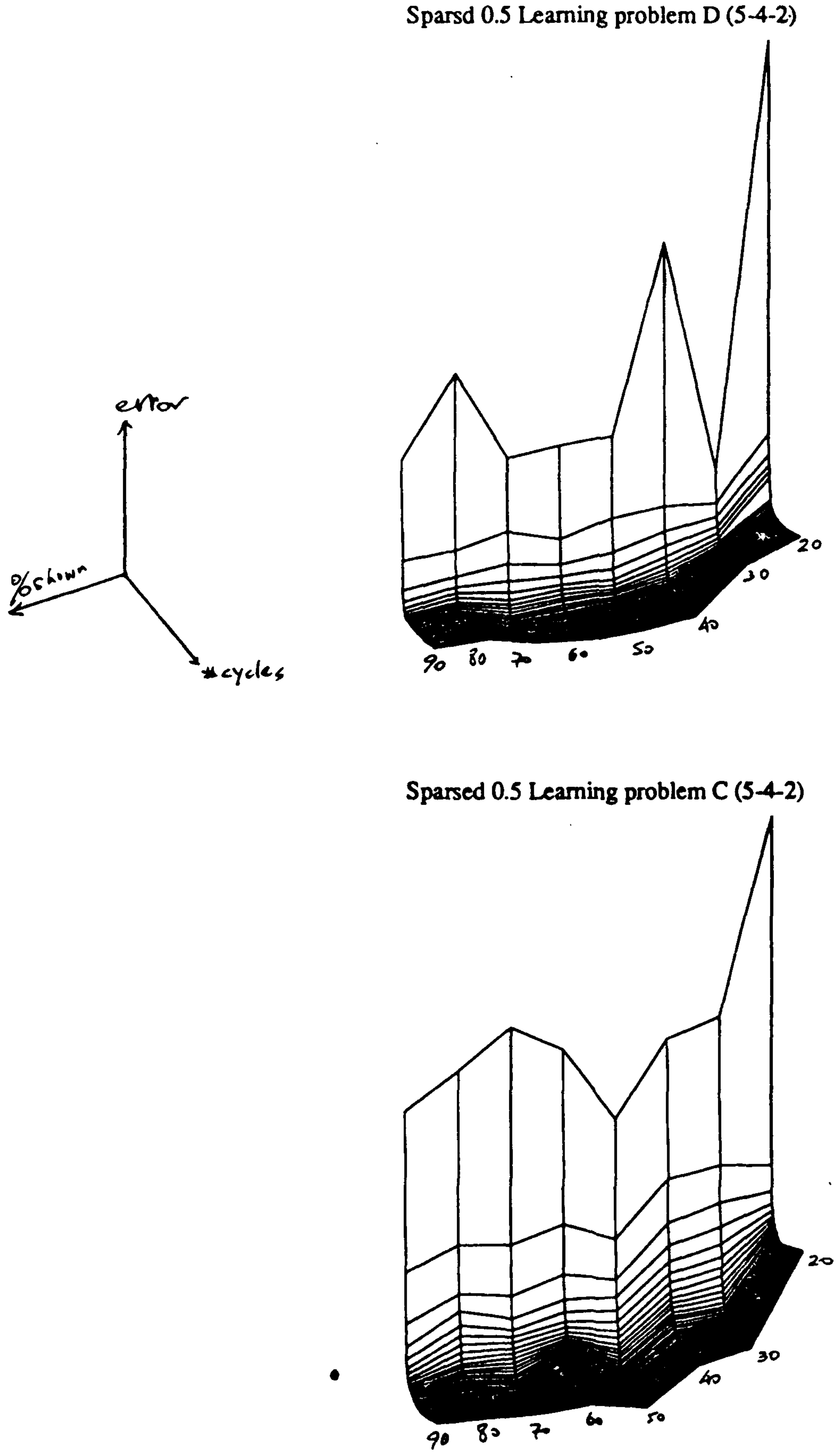
Figure D.5: 5-4-2 0.9 sparsed/Unsparsed learning



Sparsed 0.5 Learning problem A (5-4-2)

Sparsed 0.5 Learning problem B (5-4-2)

Figure D.6 : 5-4-2 sparsed 0.5 learning different problems



Sparsd 0.5 Learning problem D (5-4-2)

Sparsed 0.5 Learning problem C (5-4-2)

Figure D.7: 5-4-2 sparsed 0.5 learning different problems

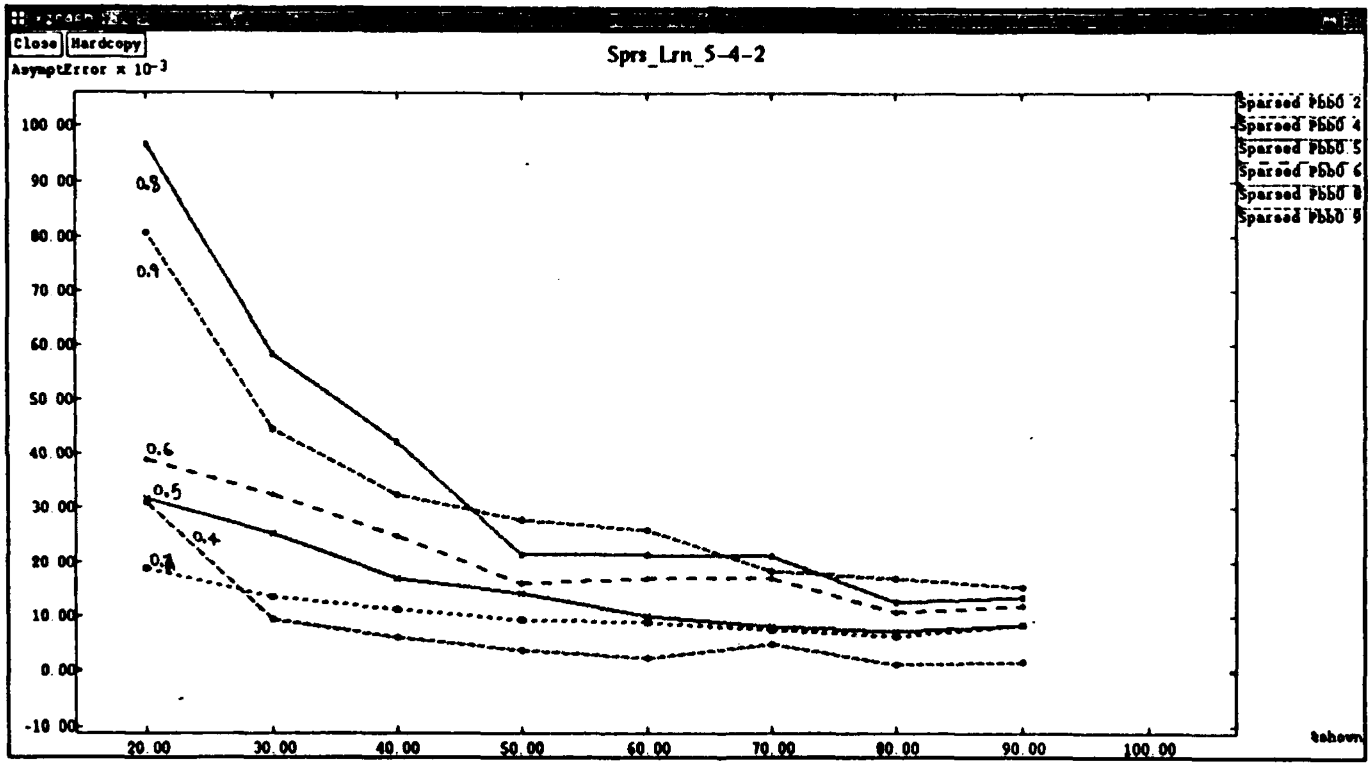


Figure E.1: Asymptotic error as function of sparsness and percentage of shown vectors.

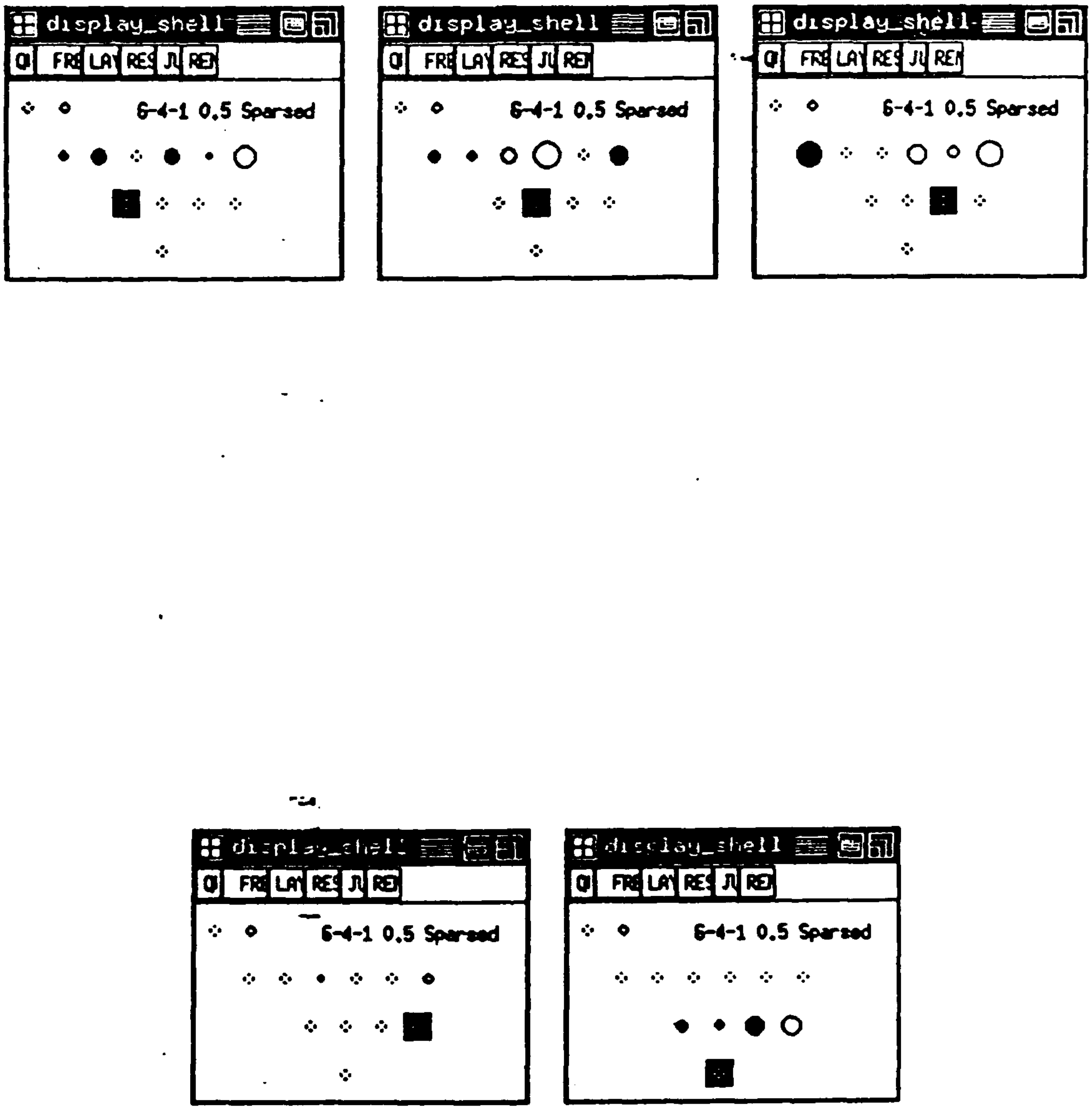
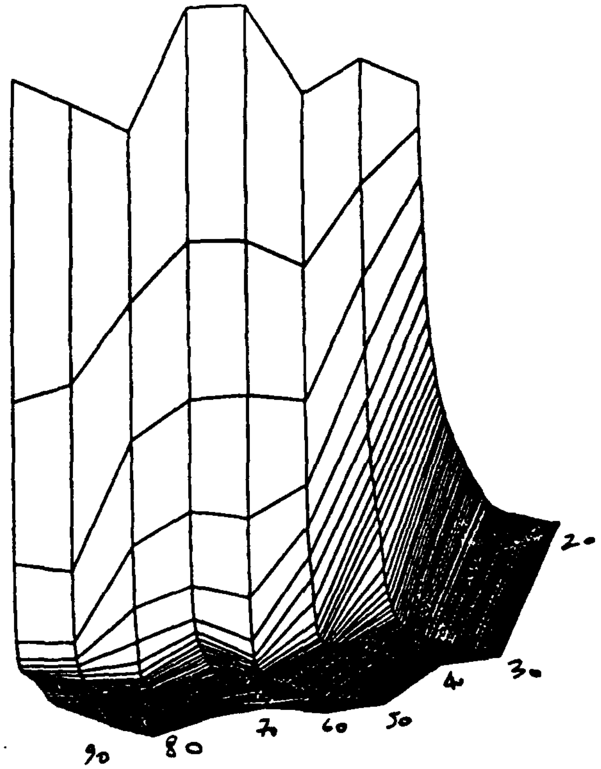
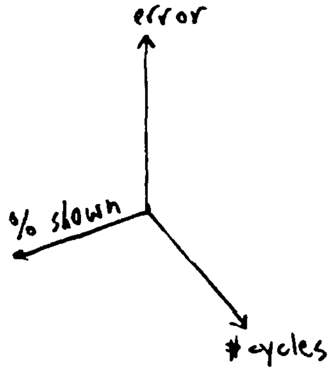


Figure F Original Vector-Generating 0.5 Sparsed Net(6-4-1)

6-4-1 Sparsed 0.4





6-4-1 Sarsed 0.2

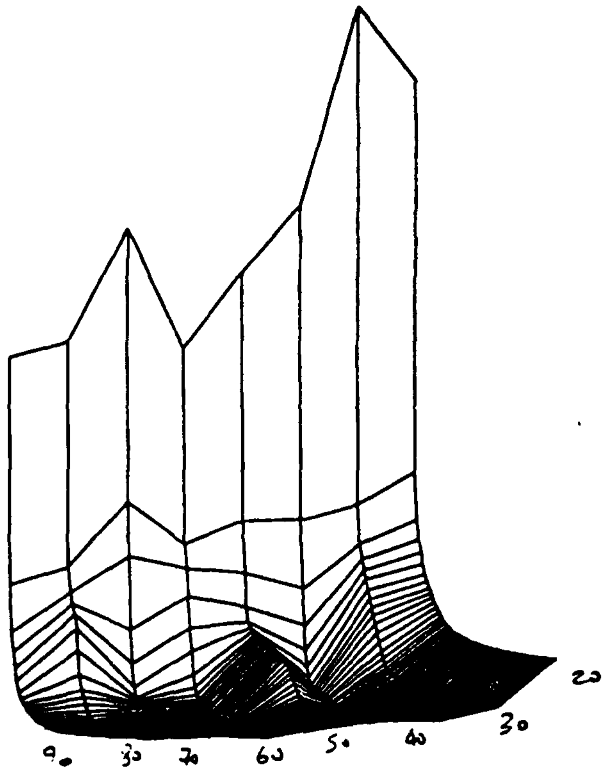
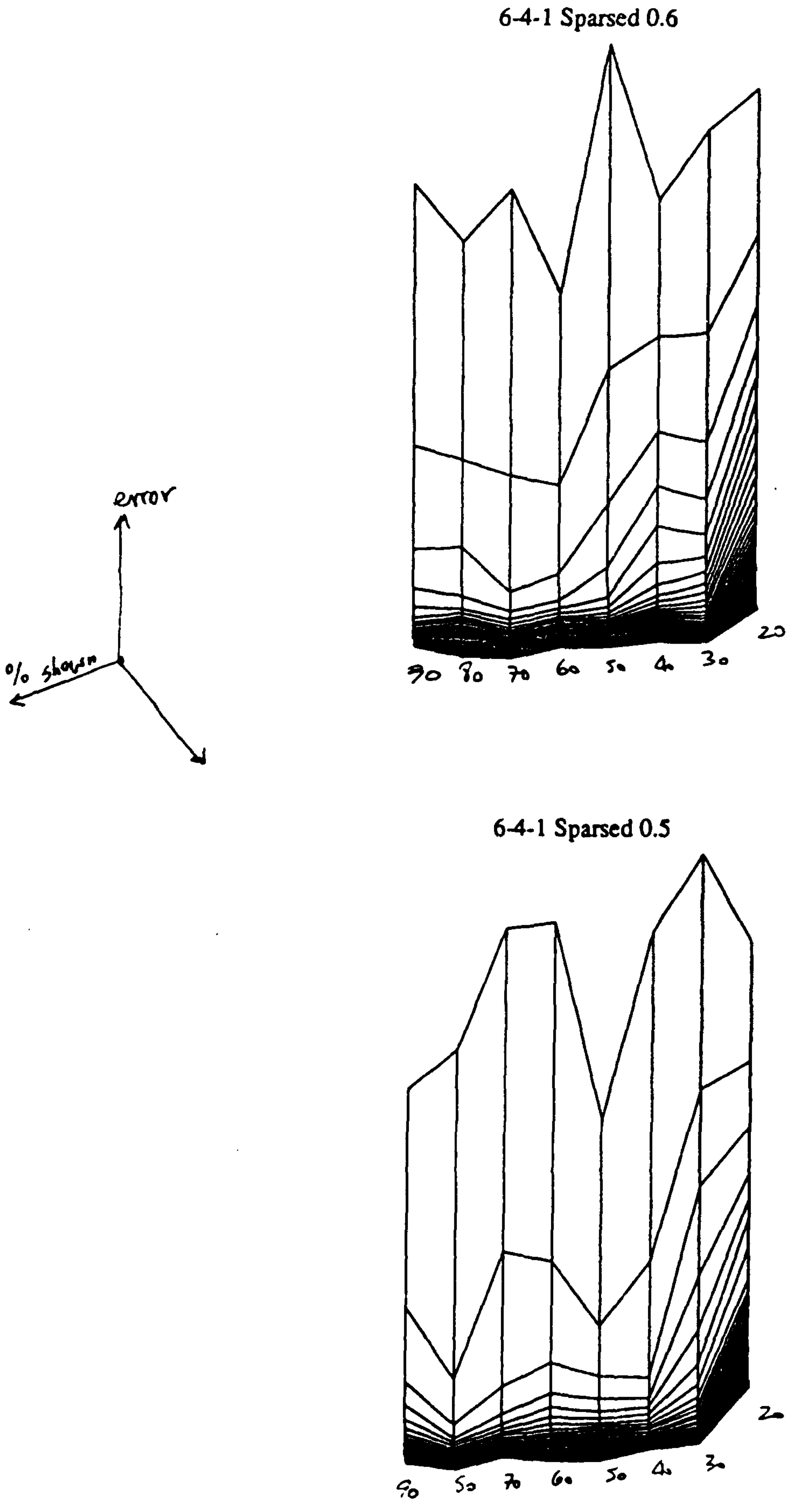
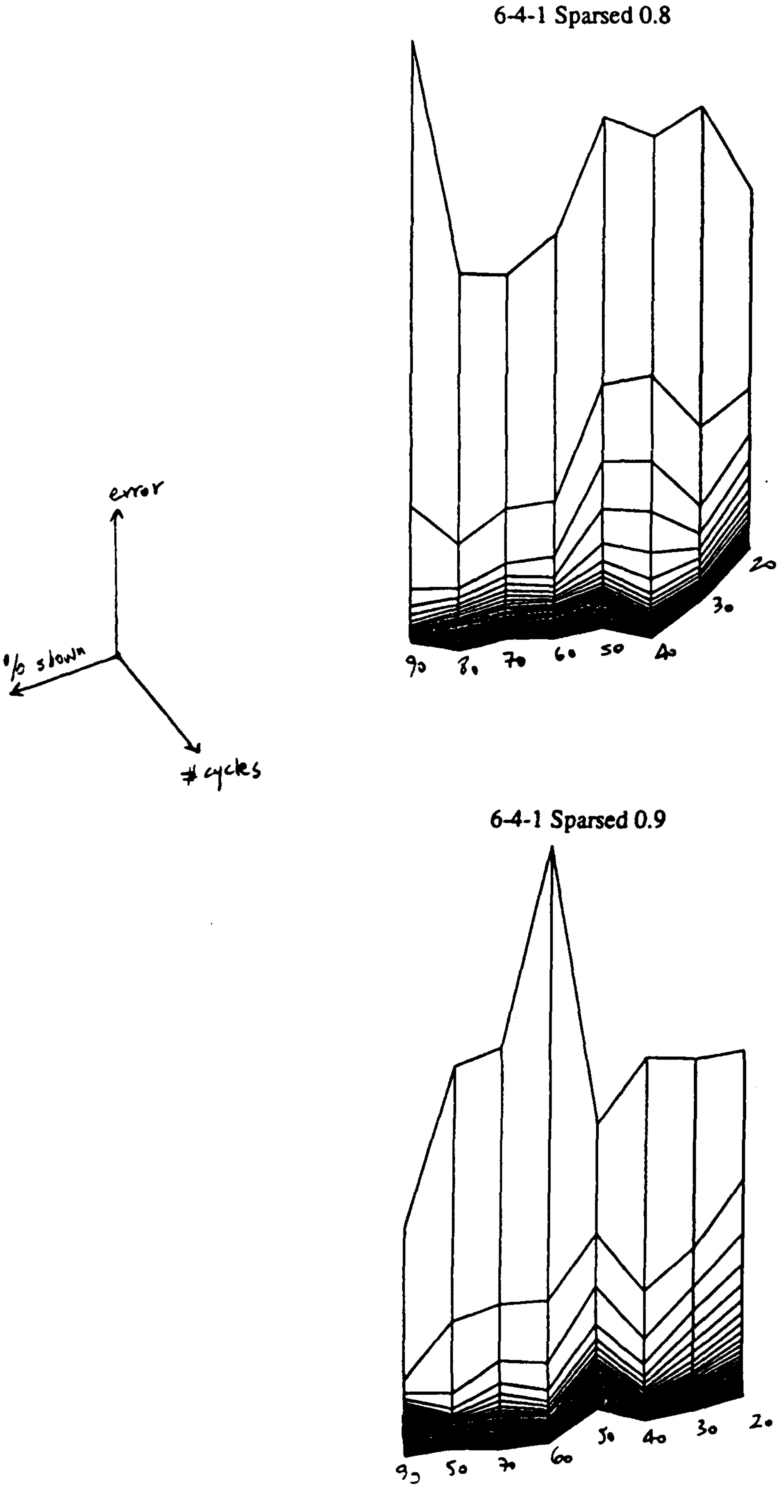


Figure F.1: 6-4-1 Sparsed 0.2, 0.4


6-4-1 Sparsed 0.6

6-4-1 Sparsed 0.5

Figure F.2: 6-4-1 Sparsed 0.5, 0.6

 
6-4-1 Sparsed 0.8

6-4-1 Sparsed 0.9

Figure F.3 : 6-4-1 Sparsed 0.8, 0.9

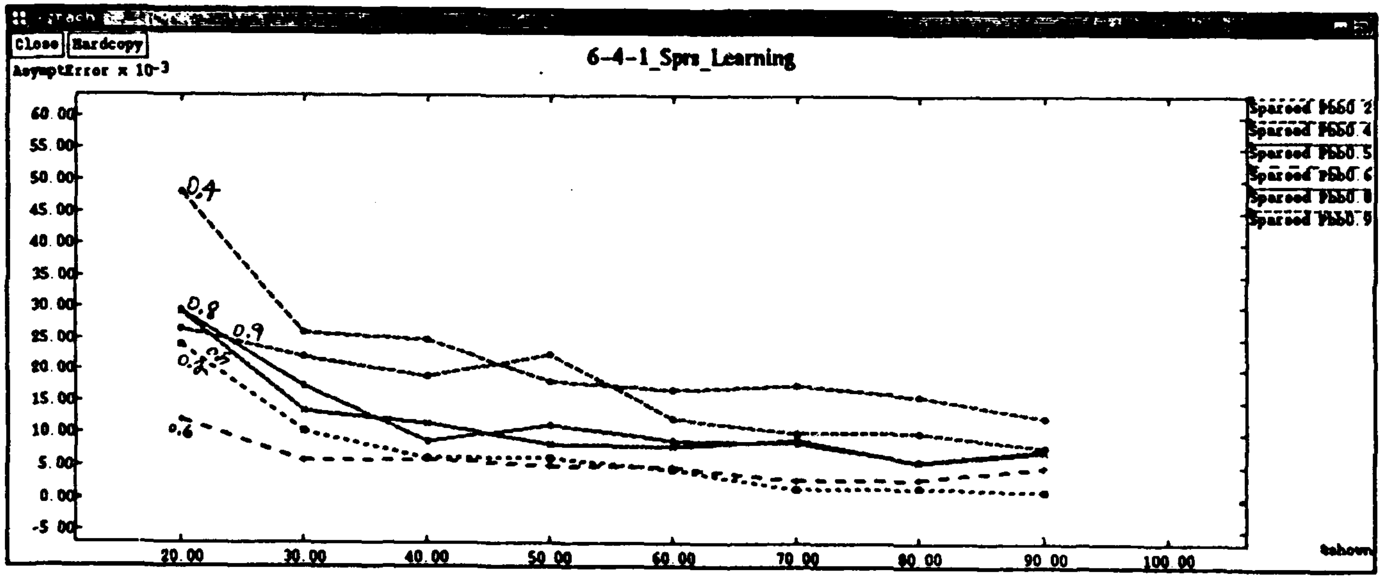


Figure F.4: Asymptotic Error as a function of Sparsness & percentage shown

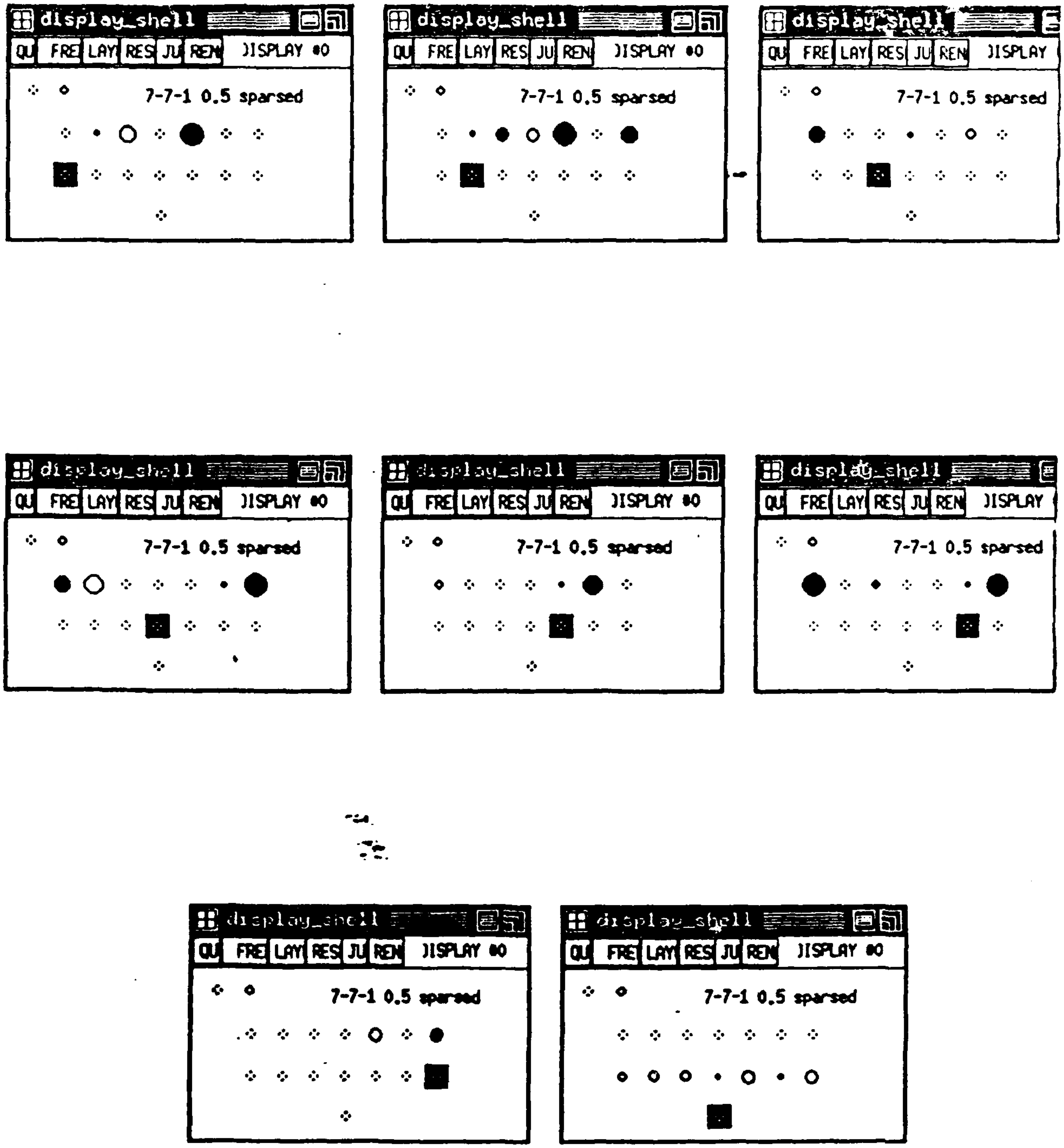
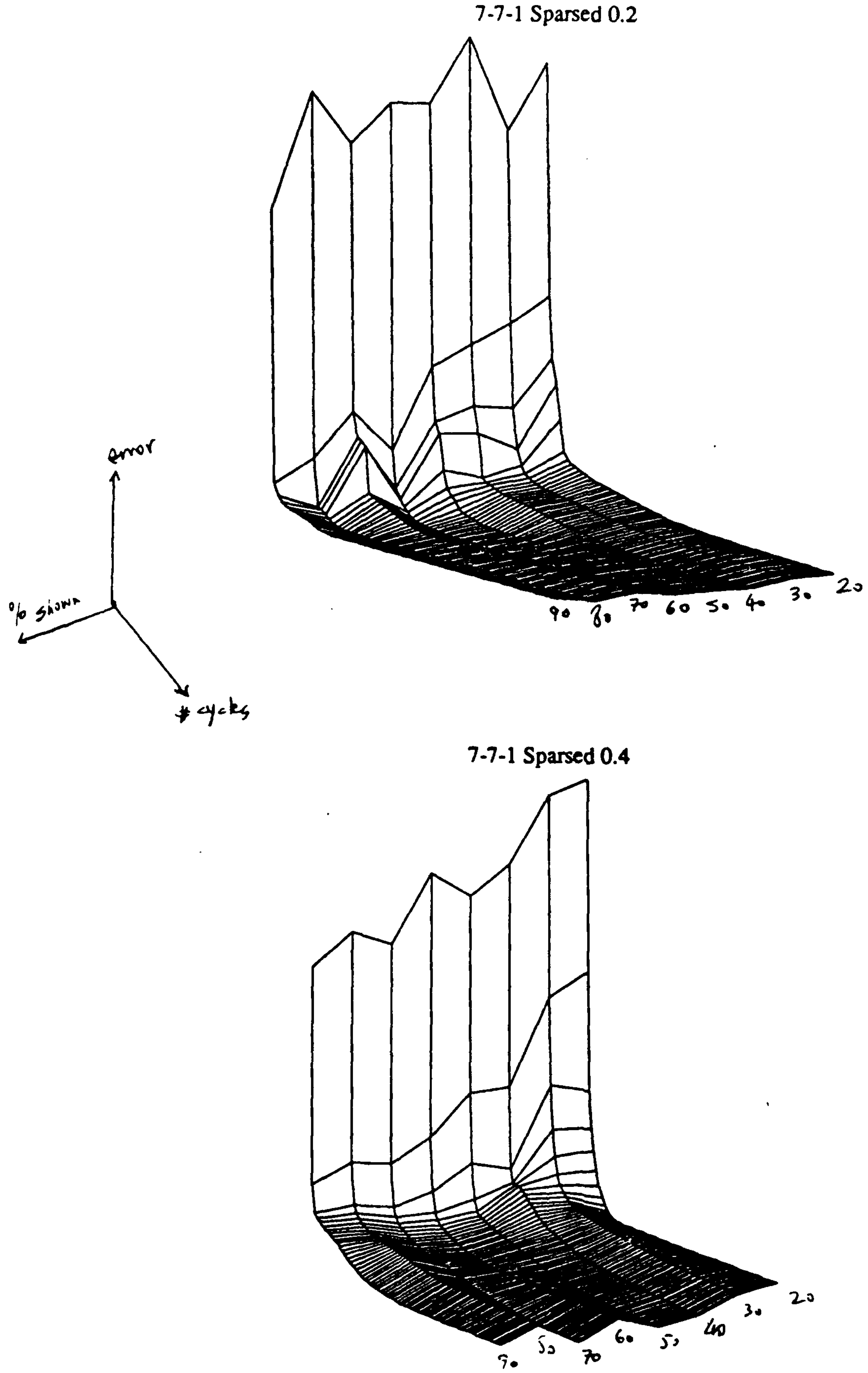


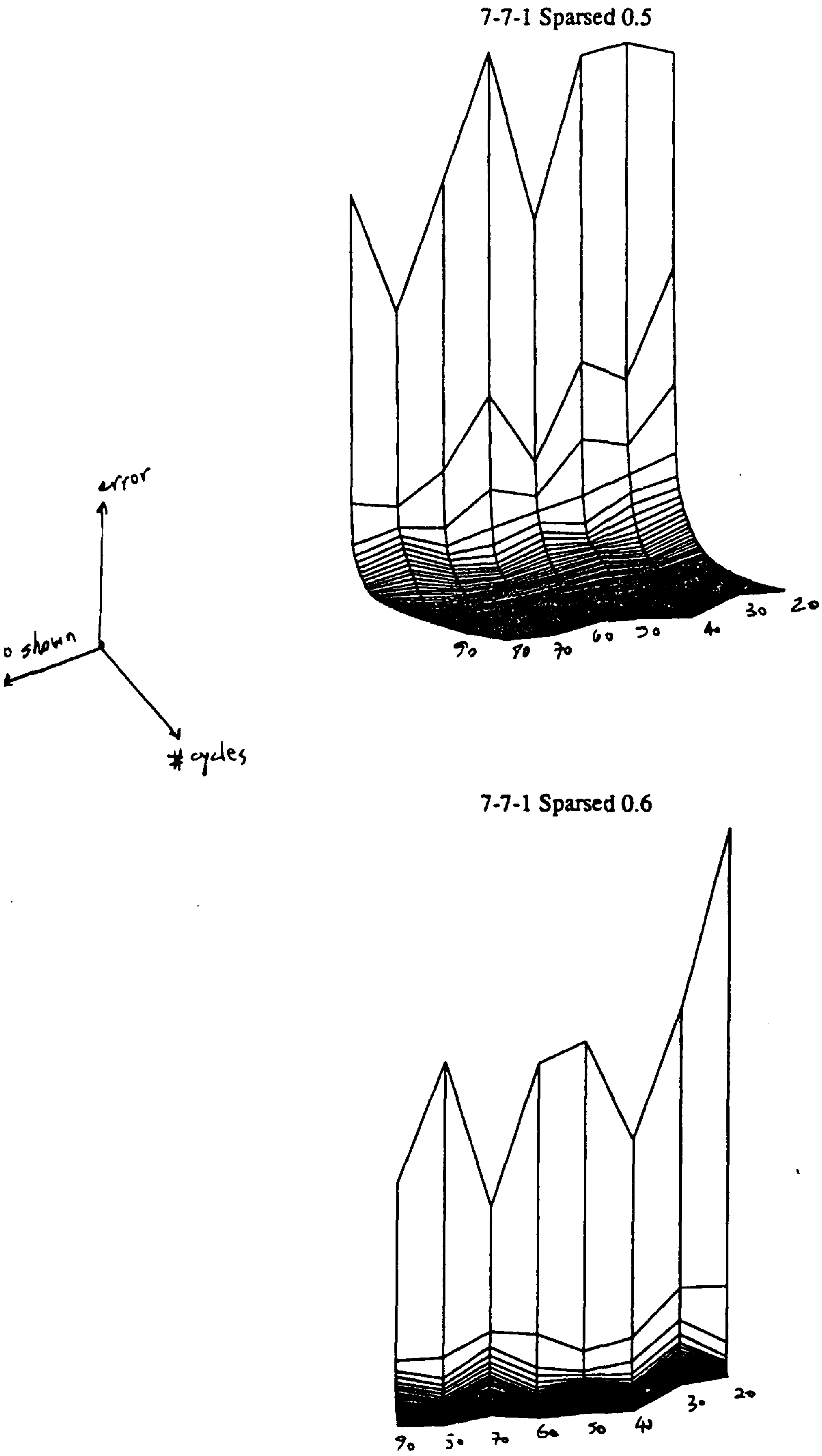
Figure 6 Original Vector-Generating 0.5 Sparsed Net(7-7-1)



7-7-1 Sparsed 0.2

7-7-1 Sparsed 0.4

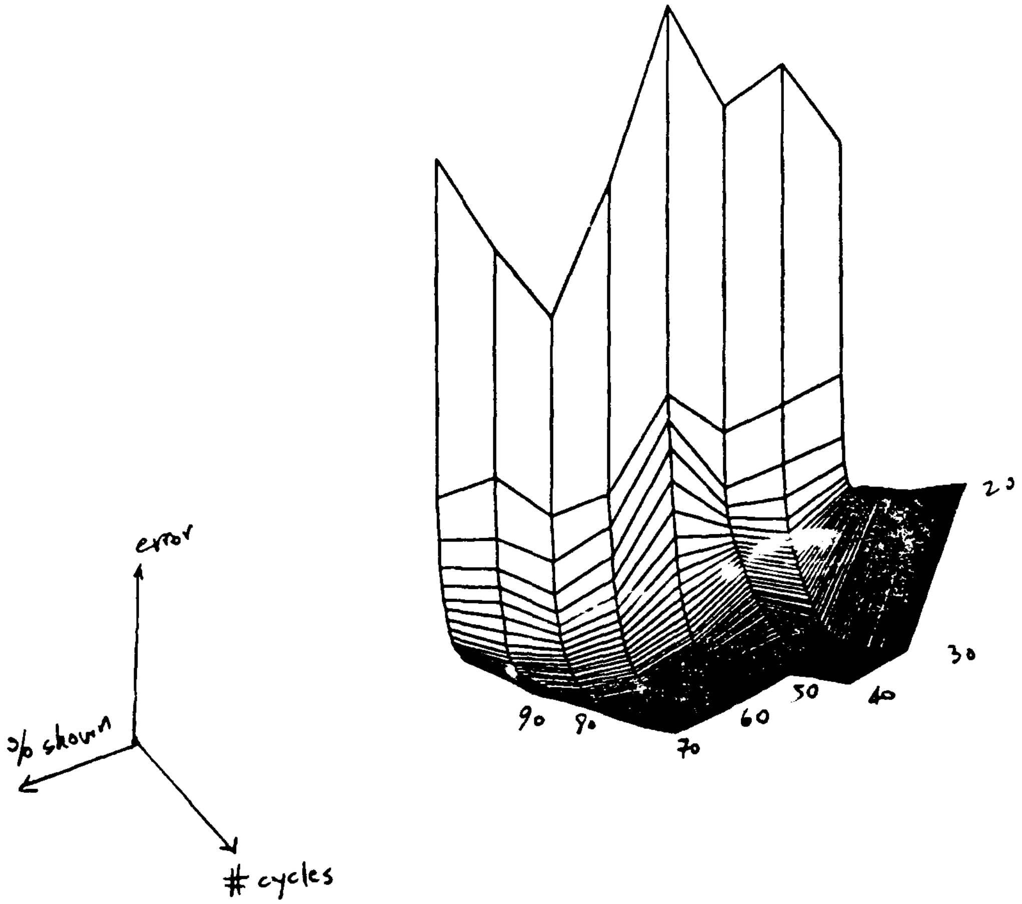
Figure G.1 : 7-7-1 Sparsed 0.2, 0.4


7-7-1 Sparsed 0.5

7-7-1 Sparsed 0.6

Figure G.2 : 7-7-1 Sparsed 0.5 0.6

7-7-1 Sparsed 0.8



7-7-1 Sparsed 0.9

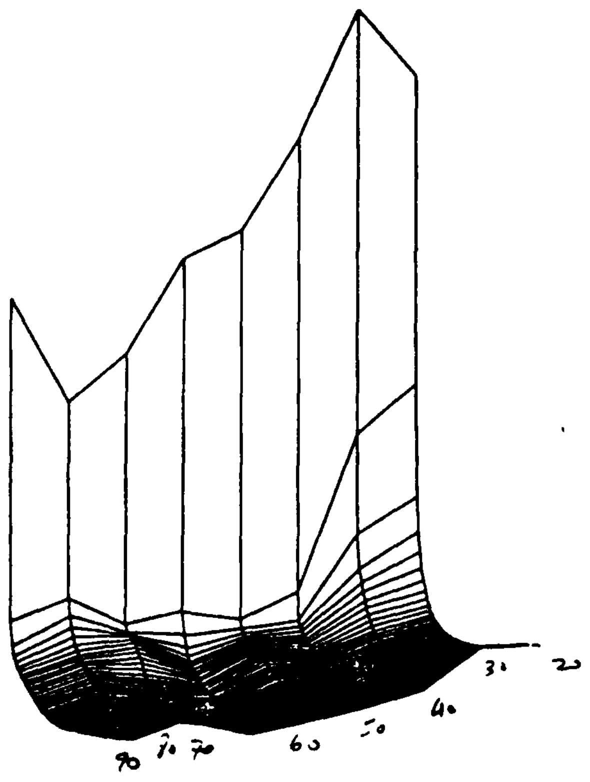


Figure G.3 : 7-7-1 Sparsed 0.8 0.9

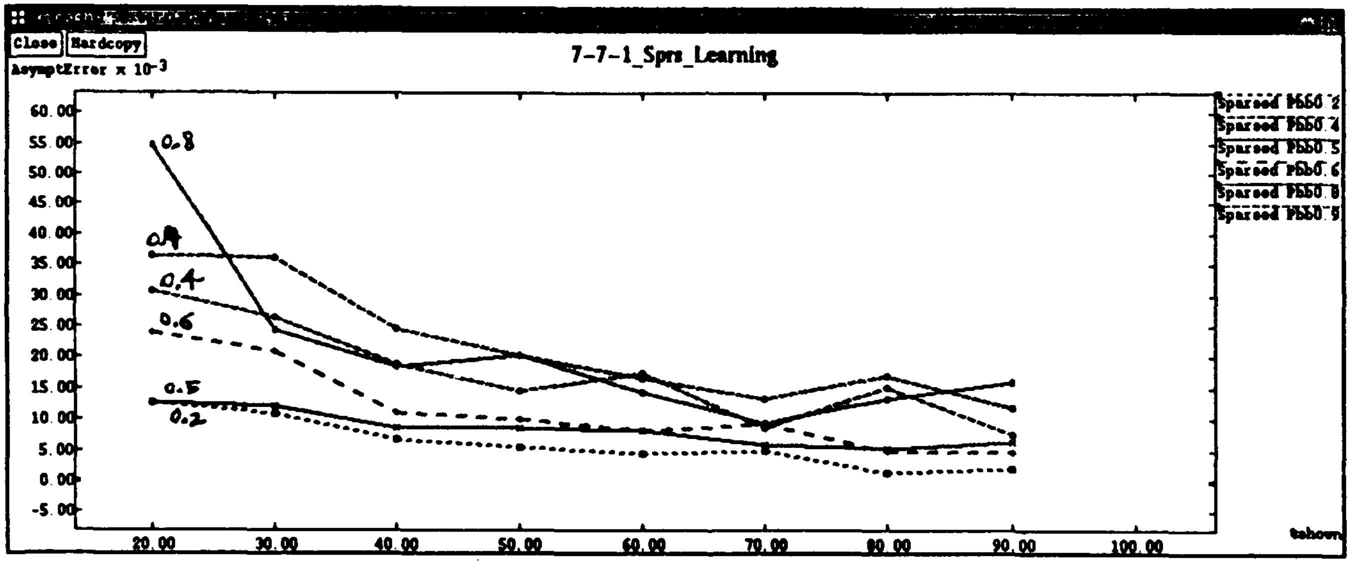
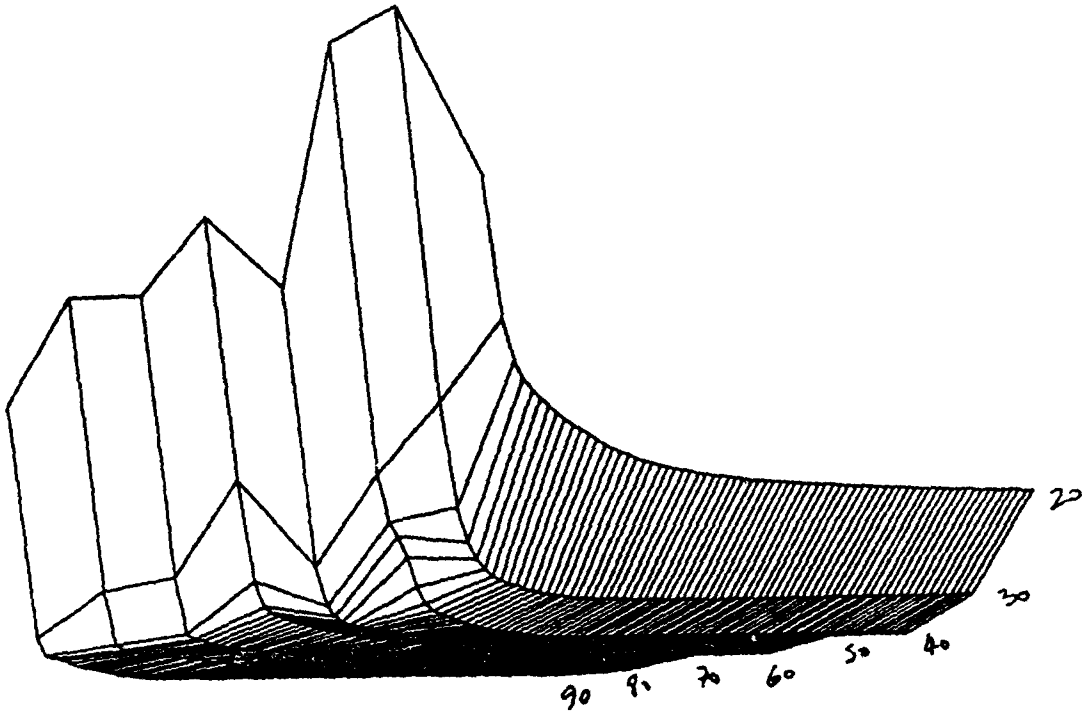
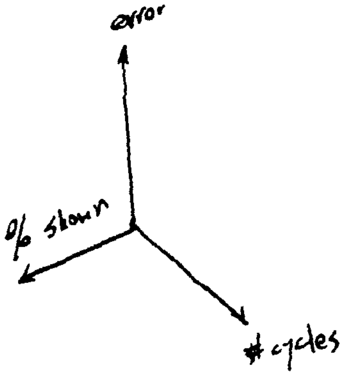


Figure G.4: Asymptotic Error as a function of Sparsness & percentage shown

8-8-1 Sparsed 0.4 learning 6-4-1





8-8-1 Sparsed 0.2 Learning 6-4-1

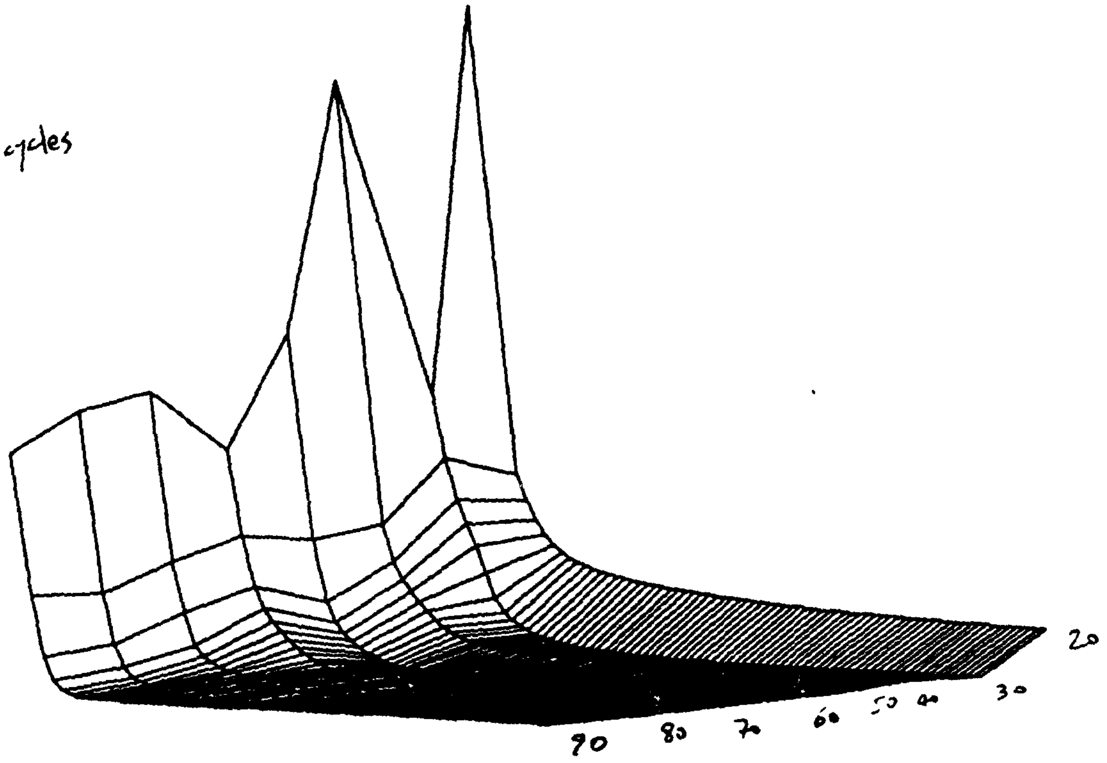
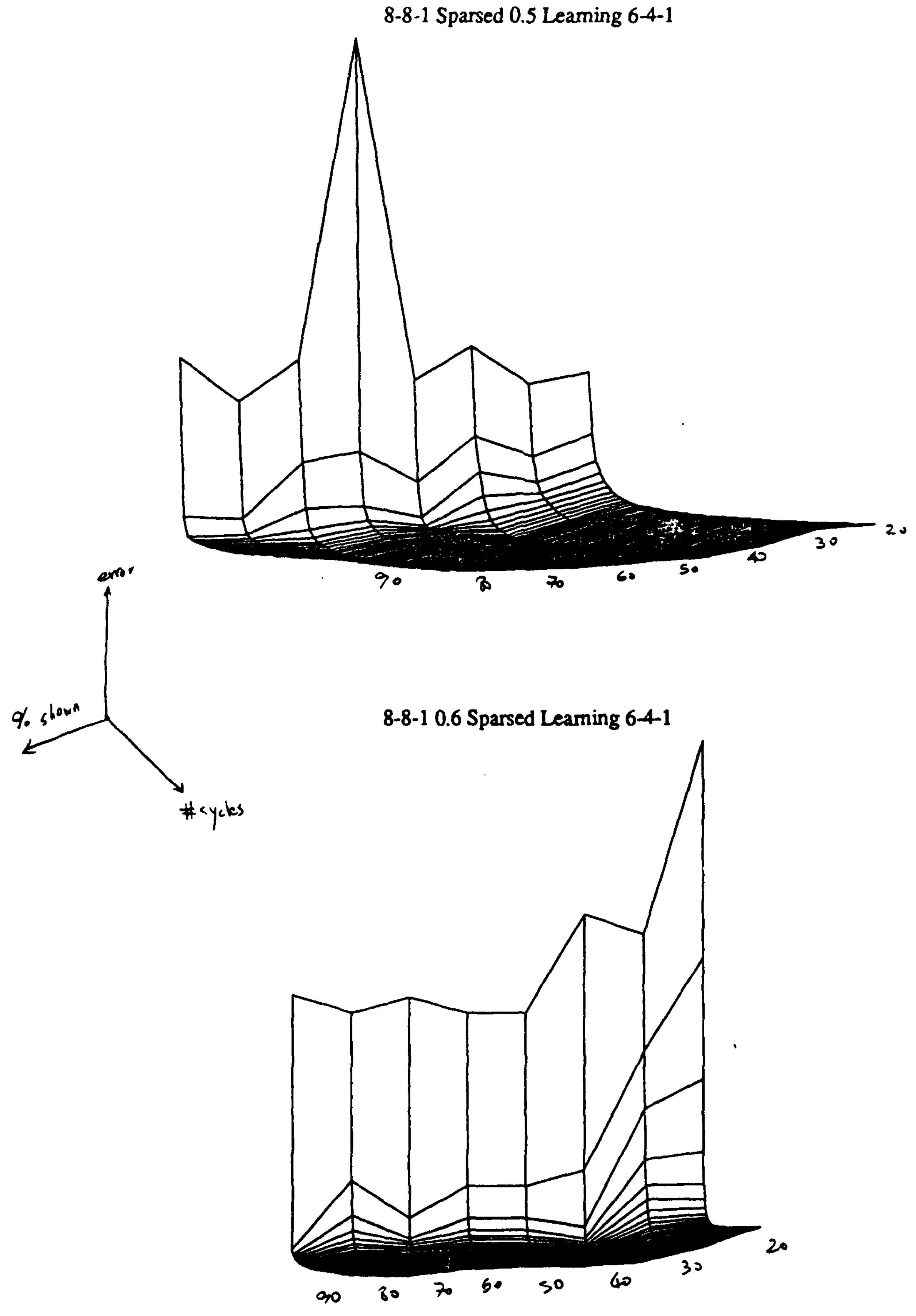


Figure H.1: 8-8-1 Sparsed 0.2, 0.4

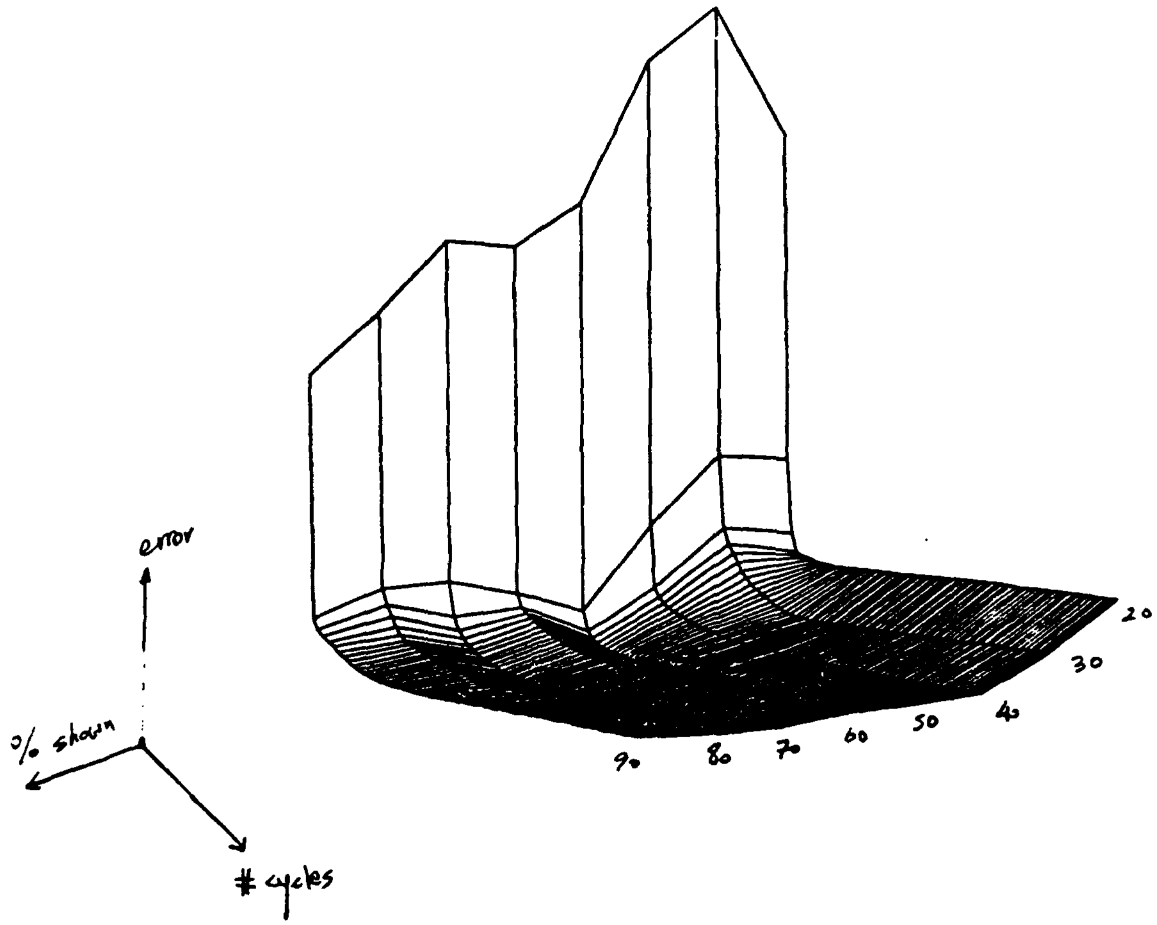


8-8-1 Sparsed 0.5 Learning 6-4-1

8-8-1 0.6 Sparsed Learning 6-4-1

Figure H.2 : 8-8-1 Sparsed 0.5 0.6

8-8-1 Sparsed 0.9 Learning 6-4-1



8-8-1 Sparsed 0.8 Learning 6-4-1

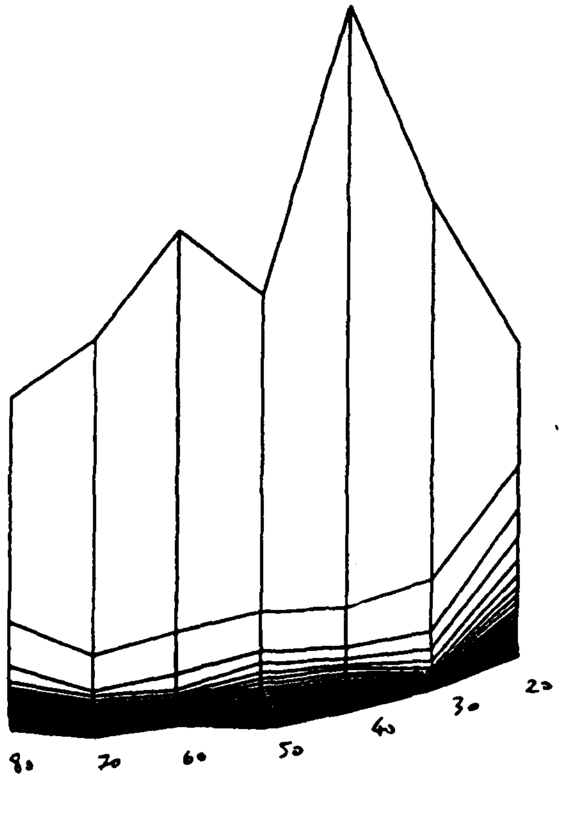


Figure 4.3 : 8-8-1 Sparsed 0.8 & 9

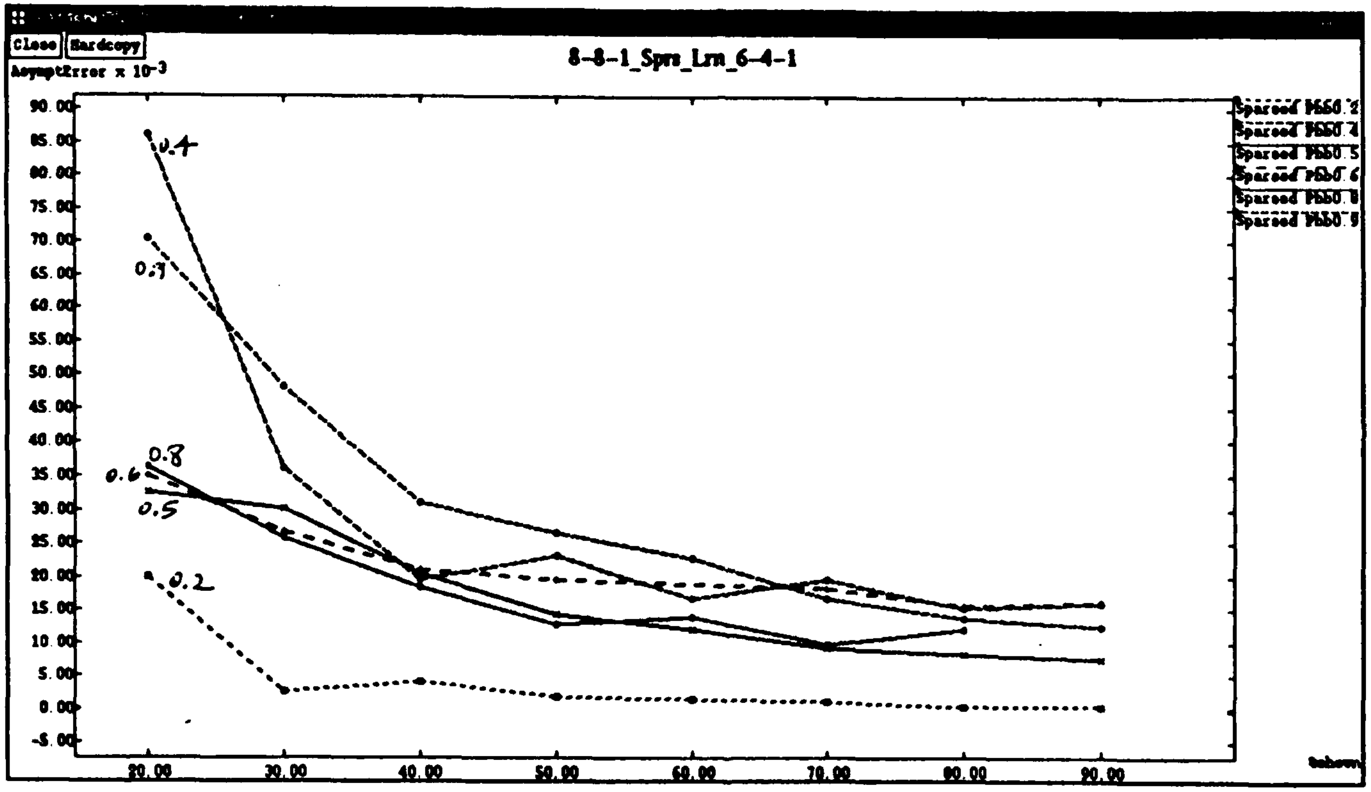
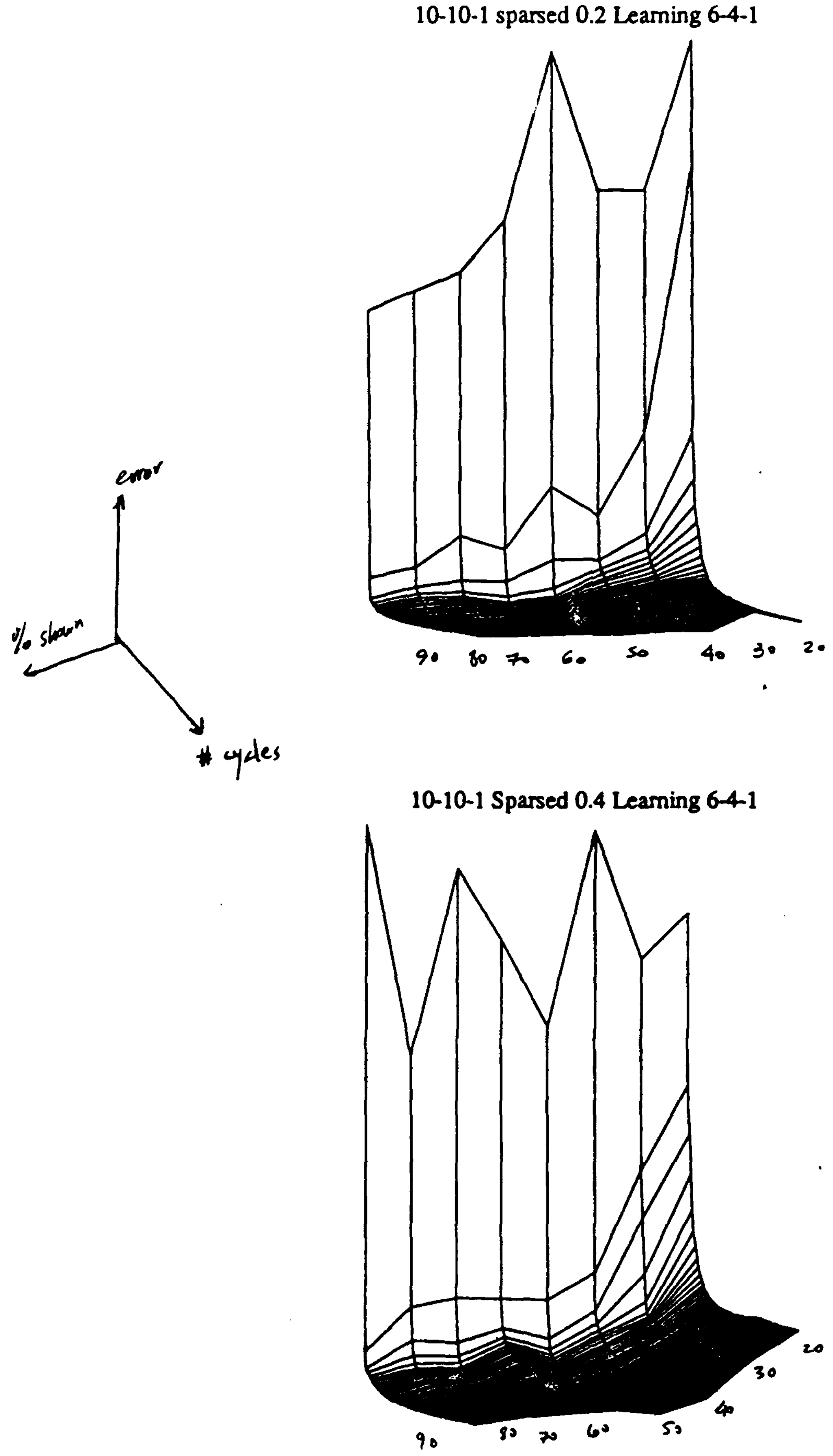


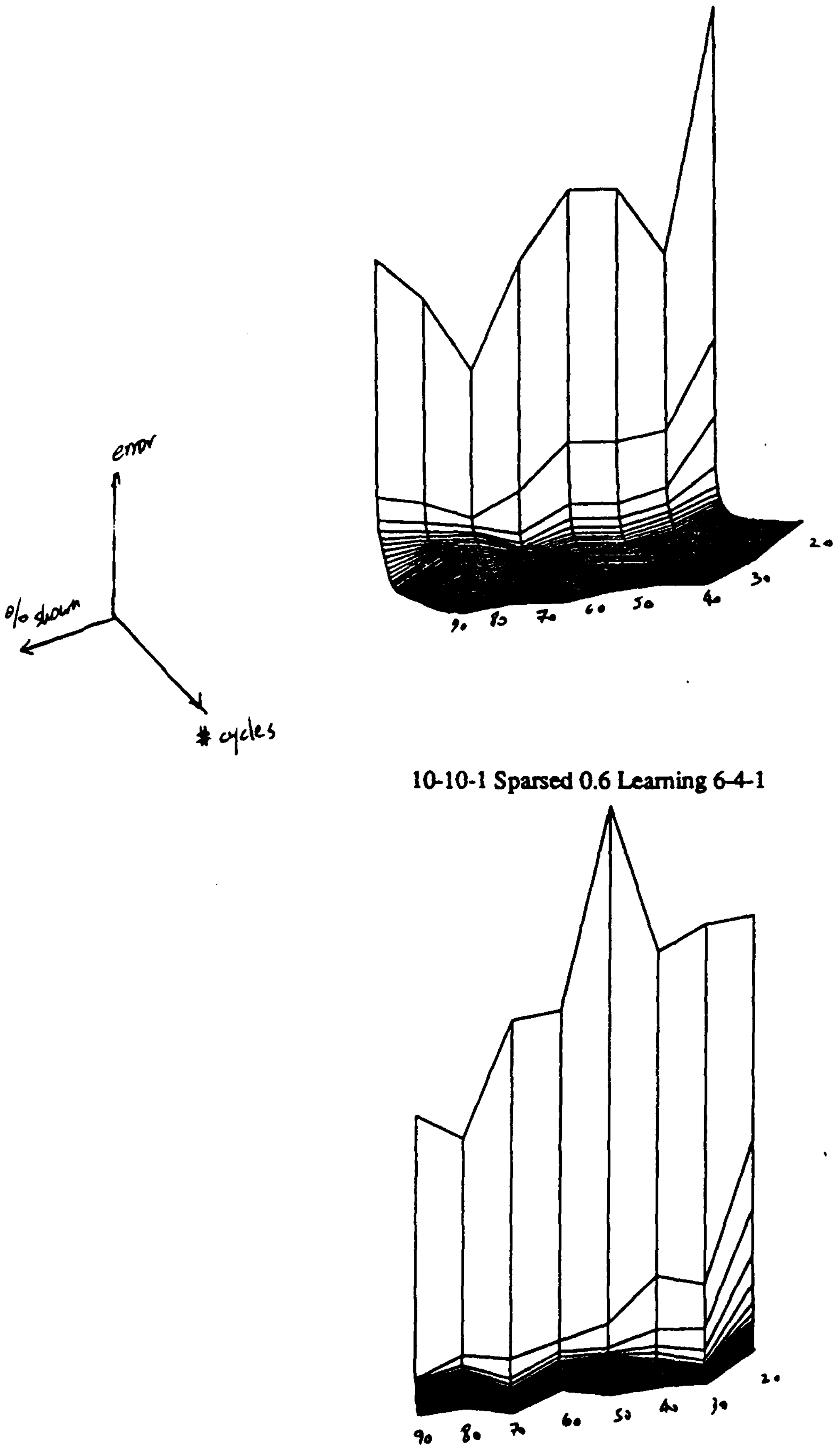
Figure H.4: Asymptotic Error as a function of Sparsness & percentage shown



10-10-1 sparsed 0.2 Learning 6-4-1

10-10-1 Sparsed 0.4 Learning 6-4-1

Figure J.1: 10-10-1 sparsed 0.2, 0.4

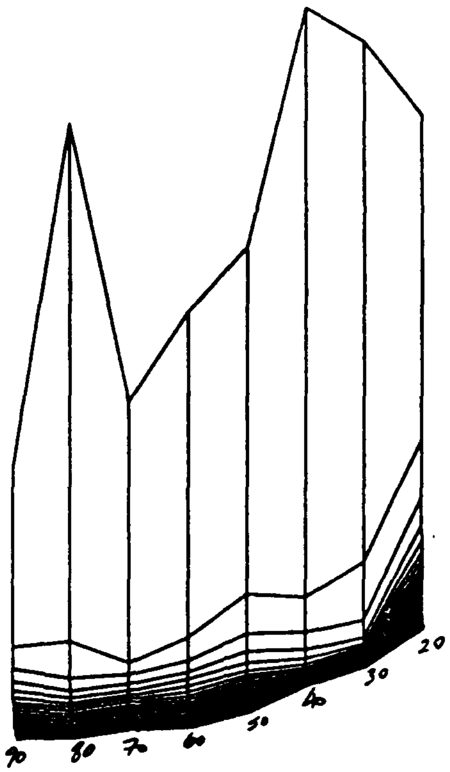


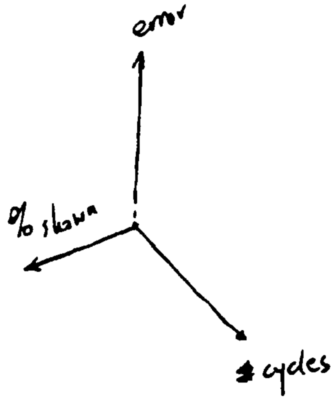
10-10-1 sparsed 0.5 Learning 6-4-1

10-10-1 Sparsed 0.6 Learning 6-4-1

Figure J.2: 10-10-1 Sparsed 0.5, 0.6

10-10-1 Sparsed 0.8 Learning 6-4-1





10-10-1 Sparsed 0.9 Learning 6-4-1

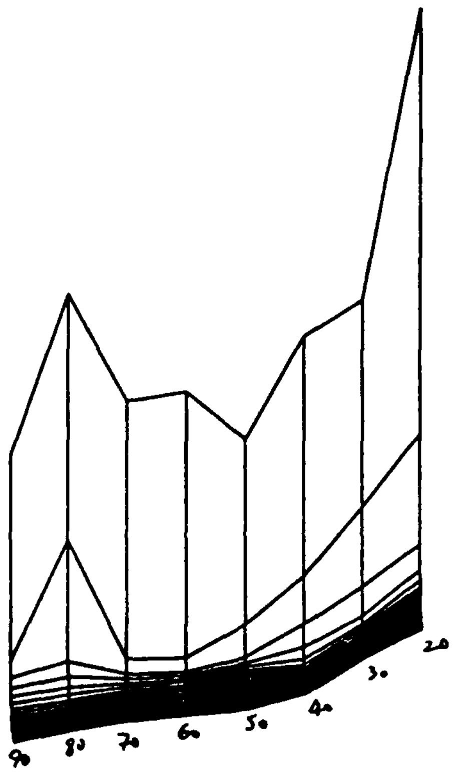


Figure J.3: 10-10-1 Sparsed 0.8, 0.9

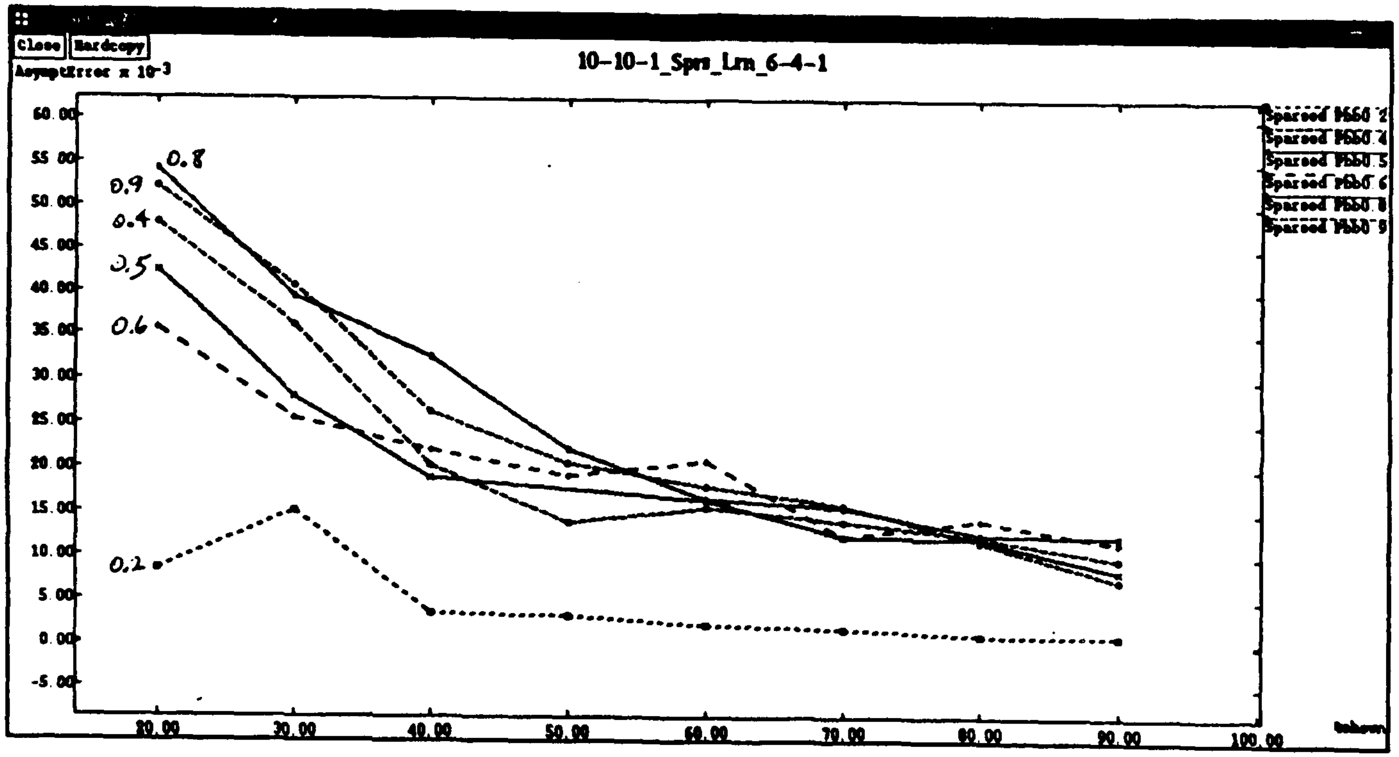


Figure J.4: Asymptotic Error as a function of sparsness and percentage shown

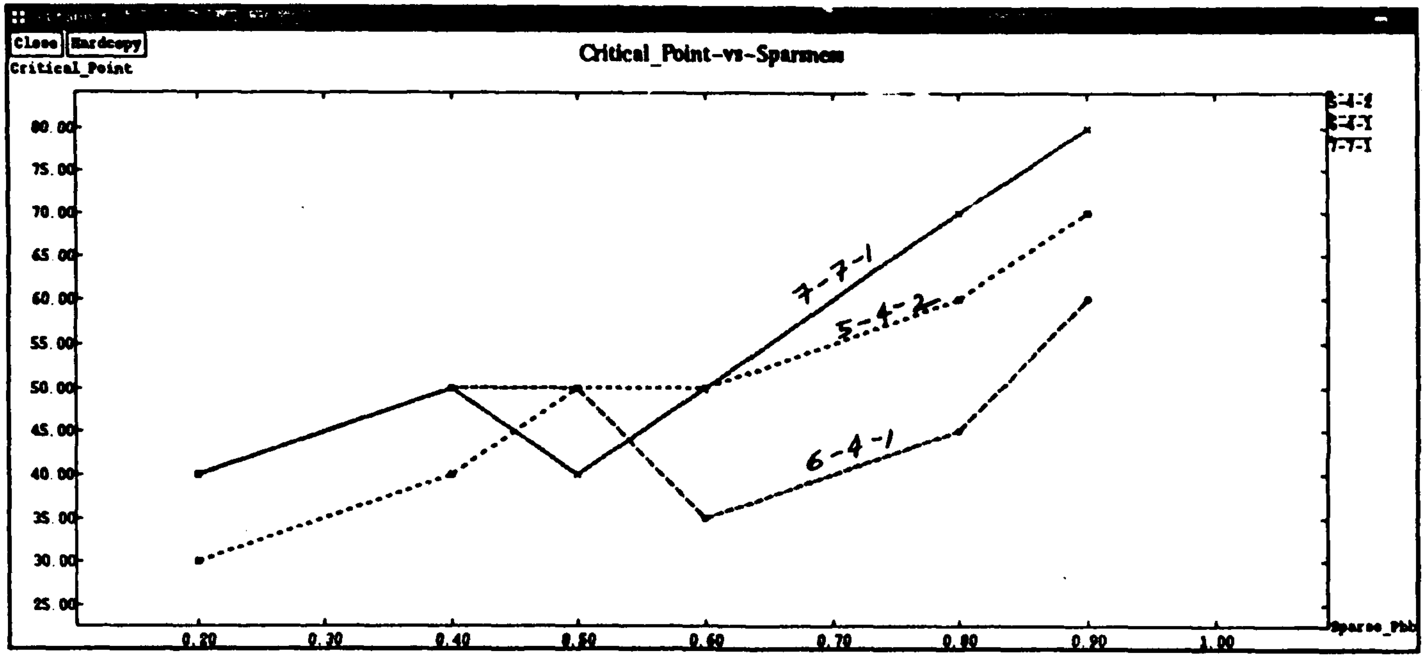


Figure K.1: Comparing Critical points of different networks.

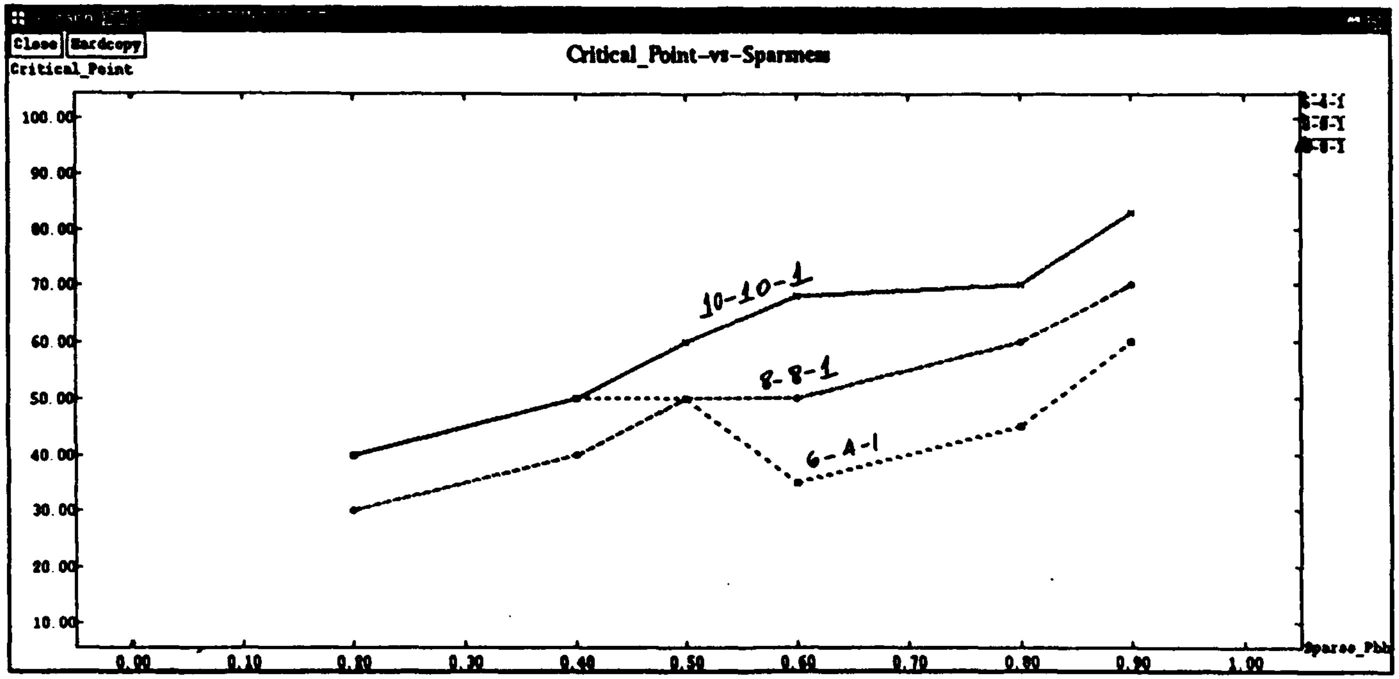


Figure K.2: Comparing Critical Points of different networks

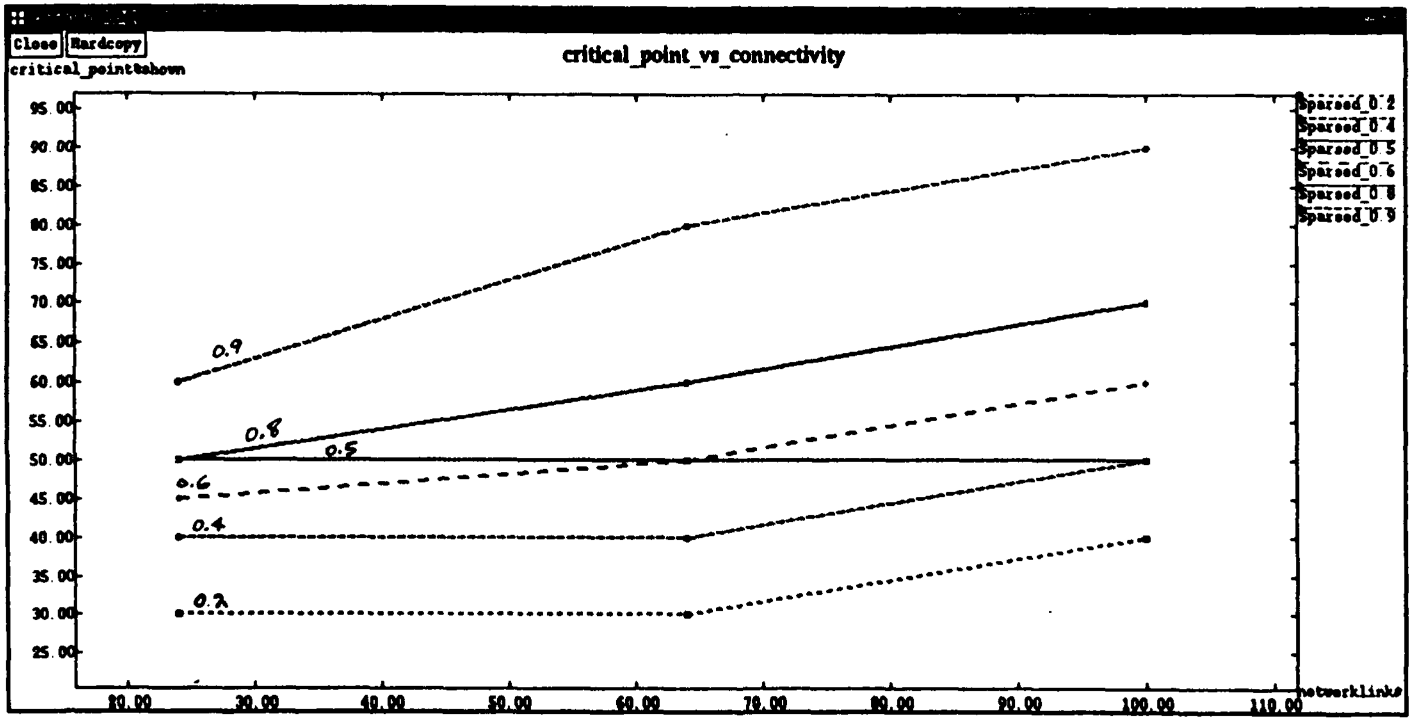


Figure K.3: Critical point (% shown) v.s. size of network (# links)

JOURNAL OF COMPLEXITY 7, 316–337 (1991)

# Programmed Interactions in Higher-Order Neural Networks: Maximal Capacity*

SANTOSH S. VENKATESH†

*Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania 19104*

AND

PIERRE BALDI‡

*Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109*

Received February 15, 1989

The focus of the paper is the estimation of the maximum number of states that can be made stable in higher-order extensions of neural network models. Each higher-order neuron in a network of $n$ elements is modeled as a polynomial threshold element of degree $d$. It is shown that regardless of the manner of operation, or the algorithm used, the storage capacity of the higher-order network is of the order of one bit per interaction weight. In particular, the maximal (algorithm independent) storage capacity realizable in a recurrent network of $n$ higher-order neurons of degree $d$ is of the order of $n^d/d!$. A generalization of a spectral algorithm for information storage is introduced and arguments adducing near optimal capacity for the algorithm are presented. © 1991 Academic Press, Inc.

## 1. INTRODUCTION

A formal neuron (after McCulloch and Pitts, 1943) is defined as a linear threshold element which accepts $n$ inputs and computes a binary output based on the sign of a linear form of the inputs. When $n$ such elements are

* Presented in part at the IEEE Conference on Neural Information Processing Systems, Denver, Colorado, November, 1987, and at the IEEE International Symposium on Information Theory, Kobe, Japan, June, 1988.

† Corresponding author.

‡ Also Division of Biology, California Institute of Technology, Pasadena, CA 91125.

0885-064X/91 $3.00

interconnected with the output of each neuron serving as input to all the neurons in the network, a closed feedback system results with dynamics described by trajectories on the vertices of the $n$-cube. Each vertex defines a possible state of the recurrent network, and we identify the vector of neural outputs as the (instantaneous) state of the system. The fixed points (or stable states) of such recurrent networks are of importance in their computational characterization; in particular, we are interested in the following question: What is the maximum number of arbitrarily specified vertices that can be made stable in a recurrent neural network by suitable selection of neural interconnectivity?

In this paper we focus on recurrent networks where the computational elements are higher-order extensions of the basic linear threshold neural model. Each higher-order neuron is a *polynomial threshold element* of a given degree $d$. If, in a recurrent network of $n$ higher-order neurons, the current outputs (states) of the neurons are $u_1, \ldots, u_n \in \{-1,1\}$, then an update, $u'_{i_1}$, of the state of the $i_1$th neuron is given by the sign of an algebraic form

$$u'_{i_1} = \operatorname{sgn}\left(\sum_{1\le i_2\le\cdots\le i_{d+1}\le n} w_{i_1 i_2 \cdots i_{d+1}} u_{i_2} \cdots u_{i_{d+1}}\right). \tag{1}$$

The number of degrees of freedom in choosing the interaction coefficients (or weights) $w_{i_1 i_2 \cdots i_{d+1}}$ is increased to $n^{d+1}$ from the $n^2$ weights for the case of linear interactions. The added degrees of freedom in the interaction coefficients can potentially result in enhanced flexibility and programming capability over the linear case as has been noted independently by several authors (Lee *et al.*, 1986; Psaltis and Park, 1986; Baldi and Venkatesh, 1987, 1988).

We rigorously estimate the storage capacity of recurrent higher-order neural networks: specifically, we calculate the maximum number of arbitrarily specified vectors that can be made stable in a recurrent network of $n$ polynomial threshold units of degree $d$.[1] All our results point in the following direction.

*Regardless of the manner of operation, or the algorithm utilized, the storage capacity of a higher-order network of degree d is of the order of* 1 *memory bit per interaction coefficient. And in particular:*

- *The storage capacity of the outer-product algorithm generalized to networks of degree d is of the order of $n^d/\log n$ memories (with constants depending on the variant employed);*

[1] Cases where networks have random interaction coefficients (instead of the programmed scenario here) lead to entirely different computational issues. We deal with these in a concurrent paper (Venkatesh and Baldi, 1989).

• *The maximal (algorithm independent) storage capacity realizable in a higher-order neural network of degree d is of the order of* $n^d/d!$*;*

• *Near optimal storage capacities of the order of* $n^d/d!$ *memories can be obtained by variants of the spectral algorithm.*

In this paper we set up the basic definitions in Section 2, construct a spectral based algorithm with near optimal capacity in Section 3, and rigorously estimate the maximal (algorithm independent) capacity of a network of given degree in Section 4. In a concurrent paper we include the capacity calculations for the outer-product algorithm generalized to degree $d$ (Venkatesh and Baldi, 1991).

*Notation.* Let $\{x_n\}$ and $\{y_n\}$ be positive sequences. We use the following standard asymptotic notation:

1. $x_n = O(y_n)$ if there is a positive constant $L$ such that $x_n/y_n \leq L$ for all $n$;
2. $x_n \sim y_n$ if $x_n/y_n \to 1$ as $n \to \infty$;
3. $x_n = o(y_n)$ if $x_n/y_n \to 0$ as $n \to \infty$.

By *almost all* we mean all but an asymptotically negligible subset: specifically, if $A_n$ denotes a sequence of finite sets, and $\mathcal{P}$ is some attribute, we say that almost all elements of $A_n$ exhibit $\mathcal{P}$ if the subsets $B_n \subseteq A_n$ for which $\mathcal{P}$ holds are such that $|B_n| \sim |A_n|$ as $n \to \infty$. We denote by $\mathbb{B}$ the set $\{-1, 1\}$, and by $[n]$ the set of indices $\{1, 2, \ldots, n\}$ for any positive integer $n$. Finally, by an *ordered multiset* we mean an ordered collection of elements where repetition is allowed.

## 2. Higher-Order Neural Networks

### 2.1. *Polynomial Threshold Units*

We consider recurrent networks of polynomial threshold units each of which yields an instantaneous state of $-1$ or $+1$. More formally, for positive integers $n$ and $d$, let $\mathcal{I}_d$ be the set of ordered multisets of cardinality $d$ of the set $[n]$. Clearly $|\mathcal{I}_d| = n^d$. For any subset $I$ of $[n]$, and for every $\mathbf{u} = (u_1 u_2 \cdots u_n) \in \mathbb{B}^n$, set $u_I = \prod_{i \in I} u_i$.

Definition 2.1. A *fully interconnected higher-order neural network of degree d* is characterized by a set of $n^{d+1}$ real *weights* $w_{(i,I)}$ indexed by the ordered pair $(i,I)$ with $i \in [n]$ and $I \in \mathcal{I}_d$, and a real *margin of operation* $\mathcal{M} \geq 0$. *The network dynamics are described by trajectories in a* state space of binary $n$-tuples, $\mathbb{B}^n$: for any state $\mathbf{u} \in \mathbb{B}^n$ on a trajectory, a component update $u_i \mapsto u_i'$ is permissible iff

$$u_i' = \begin{cases} -1 & \text{if } \sum_{I\in\mathcal{J}_d} w_{(i,I)}u_I < -\mathcal{B} \\ -u_i & \text{if } -\mathcal{B} \le \sum_{I\in\mathcal{J}_d} w_{(i,I)}u_I \le \mathcal{B} \\ 1 & \text{if } \sum_{I\in\mathcal{J}_d} w_{(i,I)}u_I > \mathcal{B}. \end{cases} \tag{2}$$

The evolution may be *synchronous* with all components of **u** being updated according to the rule (2) at each epoch, or *asynchronous* with at most one component being updated per epoch according to Eq. (2).

The network is said to be *symmetric* if $w_{(i,I)} = w_{(j,J)}$ whenever the $(d + 1)$-tuples of indices $(i, I)$ and $(j, J)$ are permutations of each other. The network is said to be *zero-diagonal* if $w_{(i,I)} = 0$ whenever any index repeats in $(i, I)$.

Let $\hat{\mathcal{J}}_d$ denote the set of all subsets of $d$ elements from $[n]$; $|\hat{\mathcal{J}}_d| = \binom{n}{d}$. Combining all redundant terms in Eq. (2), for symmetric, zero-diagonal networks a component update $u_i \mapsto u_i'$ is permissible iff

$$u_i' = \begin{cases} -1 & \text{if } \sum_{I\in\hat{\mathcal{J}}_d: i\notin I} w_{(i,I)}u_I < -\mathcal{B} \\ -u_i & \text{if } -\mathcal{B} \le \sum_{I\in\hat{\mathcal{J}}_d: i\notin I} w_{(i,I)}u_I \le \mathcal{B} \\ 1 & \text{if } \sum_{I\in\hat{\mathcal{J}}_d: i\notin I} w_{(i,I)}u_I > \mathcal{B}. \end{cases} \tag{3}$$

(If the network is symmetric and zero-diagonal then, for each nonzero coefficient $w_{(i,I)}$—i.e., coefficients $w_{(i,I)}$ for which no index repeats in $(i, I)$—the term $w_{(i,I)}u_I$ occurs $d!$ times in the sum $\sum_{I\in\mathcal{J}_d} w_{(i,I)}u_I$. Hence, $\sum_{I\in\mathcal{J}_d} w_{(i,I)}u_I = d! \sum_{I\in\hat{\mathcal{J}}_d: i\notin I} w_{(i,I)}u_I$. The constant scale factor $d!$ is removed in Eq. (3) as this is just equivalent to scaling the margin.)

The choice of margin of operation essentially specifies the "strength" of the desired interaction. A choice of margin $\mathcal{B} = 0$ leads to standard threshold operation. For a choice of nonzero margin of operation, a bit, $u_i$, retains its sign if and only if the corresponding weighted sum multiplied by $u_i$ exceeds $\mathcal{B}$; otherwise its sign is reversed.

These networks are seen to be natural generalizations to higher-order of the familiar case of *linear threshold networks* ($d = 1$). While networks of polynomial threshold units require more computationally powerful units than linear threshold functions, each polynomial threshold element (subscribing to rule (2) or to rule (3)) can be replaced by a small, equivalent network of linear threshold units. To see this note that it suffices to be able to realize each individual product of components, $u_I = \prod_{j=1}^{d} u_{i_j}$, for each choice of $I = (i_1, i_2, \ldots, i_d) \in \mathcal{J}_d$, as the results of all these computations can be combined with a single linear threshold gate to realize the desired output. Now, for each $I \in \mathcal{J}_d$, realizing the product of components $u_I$ is equivalent to checking the parity of the $d$ bits $u_{i_1}$, $u_{i_2}, \ldots, u_{i_d}$ in the product. It suffices, hence, to show that parity can be computed by small circuits of linear threshold units. But this, in fact, is a special case of a more general known result that any *symmetric Boolean* *function*—i.e., functions which are invariant under any permutation of the inputs, parity being an example—can be computed by small circuits of linear threshold elements. For completeness, we sketch a short proof of this result below.

PROPOSITION 2.2. *Any symmetric Boolean function on d variables can be computed by a linear threshold circuit of depth two and linear size; in particular, d threshold elements in the first layer and a single output threshold element in the second layer are always sufficient.*

*Proof.* The proof is constructive. Array the $2^d$ possible inputs of $\pm 1$ $d$-tuples in $(d + 1)$ rows with the elements in each row being permutations (i.e., all $d$-tuples in a row have the same number of $+1$'s), the lowest row containing the single $d$-tuple which has no $+1$'s, and with the number of $+1$'s increasing monotonically with the rows to the final $(d + 1)$th row which contains the single $d$-tuple whose components are all $+1$. Any symmetric Boolean function clearly assumes the same value for all elements (Boolean $d$-tuples) in a row. Hence, for any given symmetric function, contiguous rows where the function assumes the value $+1$ form *bands* which are separated by contiguous rows where the function assumes the value $-1$. This is illustrated schematically in Fig. 1a. Now assume there are $b$ bands where the function assumes the value $+1$. (There are at most $d/2$ such bands—the worst case occurring for the parity function.) The function can now be computed by a circuit with $2b$ linear threshold elements in the first layer and a single linear threshold element in the second layer as illustrated in Fig. 1b. (Each linear threshold unit produces a $+1$ if the weighted sum of all its inputs exceeds its threshold, and produces a $-1$ otherwise.) ■

## 2.2. *Capacity*

As in any dynamical system, the fixed points are important in the characterization of the system dynamics.

DEFINITION 2.3. Let $\mathcal{B} \geq 0$ be fixed. A state $\mathbf{u} \in \mathbb{B}^n$ of a fully interconnected network is said to be $\mathcal{B}$-*stable* iff

$$u_i \sum_{J \in \mathcal{J}_d} w_{(i,J)} u_J > \mathcal{B}, \qquad i = 1, \ldots, n.$$

Likewise, a state $\mathbf{u} \in \mathbb{B}^n$ of a zero-diagonal network is said to be $\mathcal{B}$-*stable* iff

$$u_i \sum_{J \in \mathcal{J}_d : i \notin J} w_{(i,J)} u_J > \mathcal{B}, \qquad i = 1, \ldots, n.$$

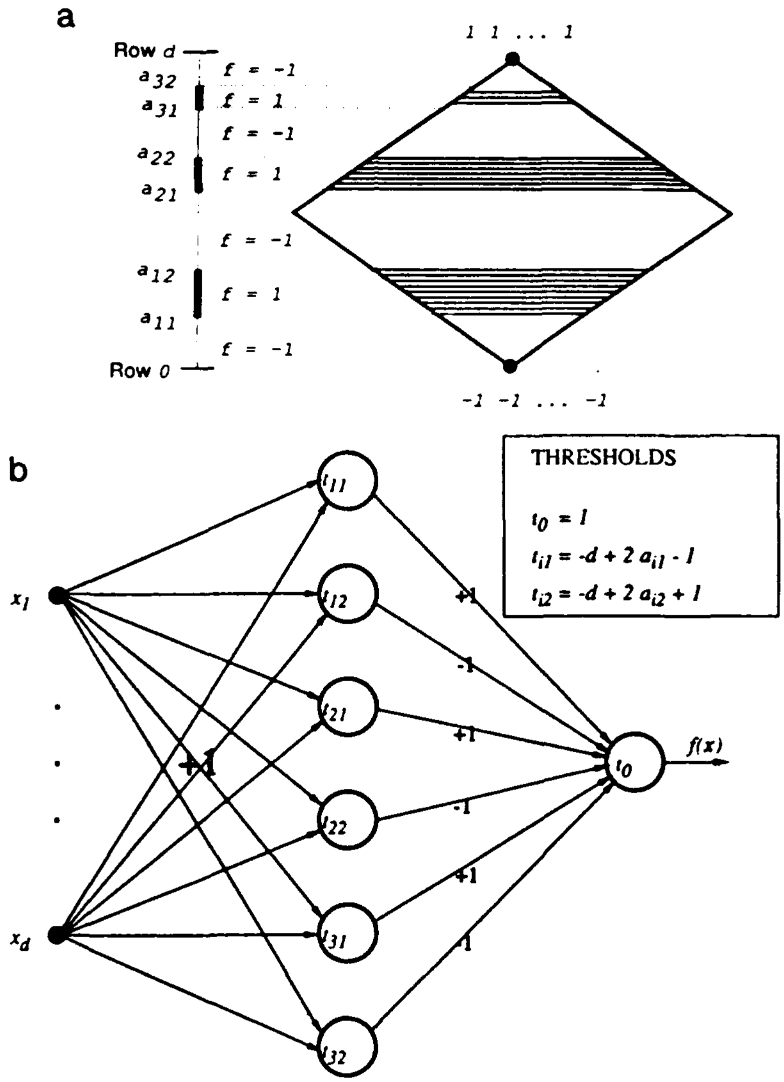


FIG. 1. (a) A symmetric Boolean function $f$ of $d$ inputs. (b) A realization of the symmetric Boolean function $f$ with a linear number (in $d$) of linear threshold elements arrayed in a depth 2 circuit.

It is easy to see that $\mathcal{B}$-stable states are fixed points of the higher-order network with evolution under a margin $\mathcal{B}$.

The fixed points of the network take on particular significance when the network interconnections are symmetric. In this case, under suitable modes of operation, Liapunov functions can be shown for the system

(Hopfield, 1982; Goles and Vichniac, 1986; Maxwell *et al.*, 1986; Venkatesh and Baldi, 1989). In particular, each fixed point exhibits an *attraction basin;* trajectories passing through states in the attraction basin of a fixed point ultimately converge to the fixed point. This geometric picture is particularly persuasive in associative memory applications; if, by appropriate choice of weights, data is stored as fixed points of the network, then the network functions as an error-correction mechanism and identifies states sufficiently similar to a stored datum with the datum.

In this paper we do not insist on symmetry in the choice of weights. We refer to the data to be stored as *memories*. By an *algorithm* for storing memories we mean a prescription for generating the interaction weights of a higher-order network of degree $d$ as a function of any given set of memories. We investigate the maximum number of arbitrarily specified memories that can be made fixed in the network by an algorithm; this is a measure of the *capacity* of the algorithm to store data.

Let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of memories to be stored in a higher-order network of degree $d$. We assume that the memories are chosen randomly from the probability space of an unending series of symmetric Bernoulli trials: specifically, the memory components, $u_i^\alpha$, $i \in [n]$, $\alpha \in [m]$, are i.i.d. random variables with

$$\mathbf{P}\{u_i^\alpha = -1\} = \mathbf{P}\{u_i^\alpha = +1\} = \tfrac{1}{2}.$$

In the following we assume that the network architecture is specified to be a higher-order network of degree $d$ operating under a margin $\mathscr{B}$.

DEFINITION 2.4. We say that $\underline{C}_n$ is a *lower capacity function* (or simply, *lower capacity*) for an algorithm if for every $0 < \lambda < 1$, and $m \leq (1 - \lambda)\underline{C}_n$, the probability that all the memories are fixed points of the network generated by the algorithm tends to one as $n \to \infty$.

Likewise, $\underline{C}_n$ is a *maximal lower capacity* if for every $0 < \lambda < 1$, and $m \leq (1 - \lambda)\underline{C}_n$, the probability that there is some network in which all the memories are $\mathscr{B}$-stable approaches one as $n \to \infty$.

DEFINITION 2.5. We say that $\overline{C}_n$ is an *upper capacity function* (or simply, *upper capacity*) for an algorithm if for every $0 < \lambda < 1$, and $m \geq (1 + \lambda)\overline{C}_n$, the probability that at least one of the memories is not a fixed point of the network generated by the algorithm tends to one as $n \to \infty$.

Likewise, $\overline{C}_n$ is a *maximal upper capacity* if for every $0 < \lambda < 1$, and $m \geq (1 + \lambda)\overline{C}_n$, the probability that there is a network in which all the memories are $\mathscr{B}$-stable approaches zero as $n \to \infty$.

*Remarks*. The first definition yields an underestimate of algorithm/network capability, while the second definition gives an overestimate. Note that the definitions of maximal capacity are algorithm independent, and bound any algorithmic capacity from above. It is clear that both lower and upper capacities always exist, and are not unique. What is more, there does not exist a largest lower capacity or a smallest upper capacity as the following proposition indicates. The proof is an immediate consequence of the definitions.

PROPOSITION 2.6. *(a) If $\underline{C}_n$ is a lower capacity, then so is $\underline{C}_n[1 \pm o(1)]$.*
*(b) If $\overline{C}_n$ is an upper capacity, then so is $\overline{C}_n[1 \pm o(1)]$.*

We combine the lower and upper estimates of capacity to obtain the following:

DEFINITION 2.7. $C_n$ is a *capacity function* (or simply, *capacity*) for an algorithm iff it is both a lower and an upper capacity for the algorithm; it is a *maximal capacity* iff it is both a maximal lower and a maximal upper capacity.

*Remarks*. Capacity follows a 0–1 law. The probabilistic setup we espouse requires almost all sequences of memories within capacity to be storable as fixed points within the network. Capacity, hence, reflects typical behavior.[2] Figures 2a and 2b indicate the threshold behavior of capacity.

Unlike lower and upper capacity functions, capacity functions are not guaranteed to exist. If a capacity function exists, however, then it is not unique.

PROPOSITION 2.8. *If $C_n$ is a capacity function, then so is $C_n[1 \pm o(1)]$; conversely, if $C_n$ and $C'_n$ are two capacity functions, then $C_n \sim C'_n$.*

*Proof.* The first part follows trivially because $C_n$ is both a lower and an upper capacity. To prove the converse, let $C_n$ and $C'_n$ be any two capacity functions. Without loss of generality, let $C'_n = [1 + \alpha_n]C_n$. We must prove that $|\alpha_n| = o(1)$.

Let $p$ denote the probability that all the memories are fixed points of the network. Fix $\lambda, \lambda' \in (0,1)$. For $m \leq (1 - \lambda')C'_n = (1 - \lambda')(1 + \alpha_n)C_n$, we have $p \to 1$ as $n \to \infty$. Further, for $m \geq (1 + \lambda)C_n$, we have $p \to 0$ as $n \to \infty$. Hence, for every choice of scalars $\lambda, \lambda' \in (0,1)$, we require that

$$1 + \alpha_n < \frac{1 + \lambda}{1 - \lambda'}$$

for large enough $n$. It hence follows that $|\alpha_n| = o(1)$. ■

[2] The definitions of capacity developed in this paper subsume within them most common notions of capacity, and can be easily extended in various ways to reflect properties of memories other than mere stability. For other variants, cf. Cover (1965), Vapnik (1982), Abu-Mostafa and St. Jacques (1985), Venkatesh (1986), Baldi (1988).

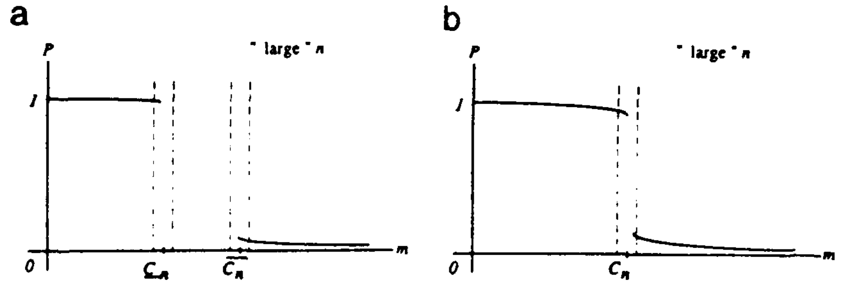


FIG. 2. (a) Lower and upper capacity functions; $P$ denotes the probability that each of $m$ randomly chosen memories is a fixed point of the network. (b) The 0–1 behavior of capacity.

Thus, if capacity functions do exist, they are not very different from each other asymptotically. Define the equivalence class $\mathcal{C}$ of (lower/upper) capacities by $C_n, C'_n \in \mathcal{C} \Leftrightarrow C_n \sim C'_n$. We call any member of $\mathcal{C}$ the (lower/upper) capacity (if $\mathcal{C}$ is nonempty).

## 3. The Spectral Algorithm

### 3.1. *The Linear Case*

For the linear case $d = 1$, Venkatesh and Psaltis (1989a), and Personnaz, Guyon, and Dreyfus (1985) have shown constructions which effectively shape the *spectrum* of the matrix of interconnection weights to ensure that the given set of memories is stable, while obtaining capacities linear in $n$. The construction entails a selection of weight matrix, $\mathbf{W}$, such that the memories $\mathbf{u}^\alpha$ are eigenvectors of $\mathbf{W}$ with positive eigenvalues. The basic notion used is that if a matrix $\mathbf{U}$ is of full rank the orthogonal projection of a vector $\mathbf{x}$ into the space spanned by the columns of $\mathbf{U}$ is given by $(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{x}$.

Let $\mathcal{B} \geq 0$ be some fixed margin of operation, and consider a fully interconnected network of degree $d = 1$. Fix $m \leq n$, and let $\lambda^{(1)}, \ldots, \lambda^{(m)} > \mathcal{B}$ be fixed (but arbitrary) positive real numbers. Let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of memories whose components are drawn from a sequence of symmetric Bernoulli trials. To each memory $\mathbf{u}^\alpha$ we associate the positive constant $\lambda^{(\alpha)}$. Let

$$\mathbf{U} = [\mathbf{u}^1 \quad \mathbf{u}^2 \quad \cdots \quad \mathbf{u}^m]$$

be the $n \times m$ matrix of memories, and let $\Lambda$ be the diagonal matrix

$$\Lambda = \begin{bmatrix} \lambda^{(1)} & 0 & \cdots & 0 \\ 0 & \lambda^{(2)} & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & \lambda^{(m)} \end{bmatrix}.$$

The spectral algorithm formally specifies the matrix of interaction weights, $\mathbf{W} = [w_{ij}]$, according to the following rule:.

$$\mathbf{W} = \mathbf{U}\Lambda(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T. \tag{4}$$

THEOREM 3.1. *For $d = 1$ and any choice of margin $\mathfrak{B} \geq 0$, the spectral algorithm has capacity $C_n = n$.*

In fact, if the prescription (4) yields well-defined weights, then we have

$$\mathbf{W}\mathbf{u}^\alpha = \lambda^{(\alpha)}\mathbf{u}^\alpha.$$

Each memory component is multiplied by a positive scalar, $\lambda^{(\alpha)} > \mathfrak{B}$, so that the memories are fixed points under evolution according to the rule (2). As a linear transformation can have at best $n$ eigenvectors with distinct eigenvalues, it follows that $n$ is an upper sequence of capacities for the algorithm. The fact that $n$ is, in fact, the capacity of the algorithm will follow if the prescription (4) is well defined for $m \leq n$ with arbitrarily high probability for large $n$. This is established by a new result of Kahn, Komlós, and Szemerédi (1990). (This is a refinement of the basic result proved by Komlós in 1967.)

PROPOSITION 3.2. *Almost all $n \times n$ matrices with $\pm 1$ components have full rank; more precisely, if the components of a random $n \times n$ matrix, $A_n$, are chosen independently and with equal probability $\frac{1}{2}$ from $\pm 1$, then there is a constant $1 < b < 2$ such that the probability that $A_n$ is nonsingular is $1 - O(b^{-n})$.*

The spectral rule amounts (in synchronous operation) to iteratively projecting states orthogonally into the linear space generated by $\mathbf{u}^1, \ldots, \mathbf{u}^m$, and then taking the closest point on the hypercube to this projection. While the algorithm appears to be non-Hebbian and nonlocal, nonetheless, a low complexity, recursive, local construction can be shown for the algorithm using Greville's theorem; the algorithm is, hence, attractive as an associative memory as it combines relatively low complexity with high capacity and efficient error-correction (Venkatesh and

Psaltis, 1989a). This approach can be extended to higher-orders as we now describe.

### 3.2. *Generalization to Higher-Order*

Let us consider the degree of interaction $d$ to be odd for definiteness. By combining terms we can replace the summation, $\sum_{I\in \mathcal{I}_d} w_{(i,I)}u_I$, for each $i = 1, \ldots, n$ in the evolution rule (2) by an equivalent sum of the form

$$\sum_{k \text{ odd}}^{d} \sum_{1\le i_1<\cdots<i_k\le n} w_{i,i_1,\ldots,i_k} u_{i_1} \cdots u_{i_k}. \tag{5}$$

For $\mathbf{u} \in \mathbb{B}^n$ to be a fixed point under evolution according to the rule (2) it, hence, suffices that

$$u_i \sum_{k \text{ odd}}^{d} \sum_{1\le i_1<\cdots<i_k\le n} w_{i,i_1,\ldots,i_k} u_{i_1} \cdots u_{i_k} > \mathfrak{B}, \qquad i = 1, \ldots, n. \tag{6}$$

Now, for any $\mathbf{u} \in \mathbb{B}^n$ let us define the *kth generation of* $\mathbf{u}$ to be the vector $\mathbf{u}[k] \in \mathbb{B}^{\binom{n}{k}}$ defined by

$$\mathbf{u}[k] = \begin{pmatrix} u_1u_2 \cdots u_{k-1}u_k \\ u_1u_2 \cdots u_{k-1}u_{k+1} \\ \cdot \\ \cdot \\ \cdot \\ u_{n-k+1}u_{n-k+2} \cdots u_{n-1}u_n \end{pmatrix}; \tag{7}$$

in other words, $\mathbf{u}[k]$ is the vector formed by lexicographically ordering the $\binom{n}{k}$ products of components of $\mathbf{u}$ taken $k$ at a time. We now form the vector $\hat{\mathbf{u}}$ from the first $\lceil d/2 \rceil$ odd generations of $\mathbf{u}$:

$$\hat{\mathbf{u}} = \begin{pmatrix} \mathbf{u}[1] \\ \mathbf{u}[3] \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{u}[d] \end{pmatrix}. \tag{8}$$

Now set

$$N_d = \sum_{k \text{ odd}}^{d} \binom{n}{k}.$$

Clearly, $\hat{\mathbf{u}}$ is a binary vector with $N_d$ components. Let $\hat{\mathbf{W}}$ denote the $n \times N_d$ matrix of coefficients, $w_{i,i_1,\cdots,i_k}$, in Eq. (5) arranged lexicographically; i.e.,

$$\hat{\mathbf{W}} = \left[\begin{matrix} w_{11} & \cdots & w_{1n} & w_{1123} & \cdots & w_{1,n-2,n-1,n} \\ w_{21} & \cdots & w_{2n} & w_{2123} & \cdots & w_{2,n-2,n-1,n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ w_{n1} & \cdots & w_{nn} & w_{n123} & \cdots & w_{n,n-2,n-1,n} \end{matrix}\right.$$

$$\left.\begin{matrix} w_{112345} & \cdots & w_{1,n-d+1,n-d+2,\ldots,n-1,n} \\ w_{212345} & \cdots & w_{2,n-d+1,n-d+2,\ldots,n-1,n} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ w_{n12345} & \cdots & w_{n,n-d+1,n-d+2,\ldots,n-1,n} \end{matrix}\right].$$

Let $\mathbf{U}$ be the $n \times m$ matrix of memories. Form the extended $N_d \times m$ binary matrix

$$\hat{\mathbf{U}} = [\hat{\mathbf{u}}^1 \quad \hat{\mathbf{u}}^2 \quad \cdots \quad \hat{\mathbf{u}}^m],$$

where $\hat{\mathbf{u}}^\alpha \in \mathbb{B}^{N_d}$ is as defined above. Let

$$\Lambda = \mathbf{dg}[\lambda^{(1)}, \ldots, \lambda^{(m)}]$$

be an $m \times m$ diagonal matrix with positive diagonal terms, $\lambda^{(\alpha)} > \mathfrak{B}$. We formally define the generalized spectral matrix of coefficients, $\hat{\mathbf{W}}$, by

$$\hat{\mathbf{W}} = \mathbf{U}\Lambda(\hat{\mathbf{U}}^T\hat{\mathbf{U}})^{-1}\hat{\mathbf{U}}^T. \tag{9}$$

Note that this yields stable memories as long as the matrix $\hat{\mathbf{U}}$ is full rank. Specifically, if the initial state is one of the memories, $\mathbf{u}^\alpha$, then we obtain

$$\hat{\mathbf{W}}\hat{\mathbf{u}}^\alpha = \lambda^{(\alpha)}\mathbf{u}^\alpha.$$

It is now easy to verify that Eq. (6) is satisfied for each component of memory $\mathbf{u}^\alpha$, so that $\mathbf{u}^\alpha$ is a fixed point under evolution according to the rule (2). If the degree of interaction, $d$, is even, the exposition follows as above with the first sum in Eq. (5) being over even $k$ instead of odd $k$. The maximal allowable rate of growth of $m$ with $n$ follows immediately.

**Theorem** 3.3. *An upper capacity of the generalized spectral algorithm of degree d is*

$$\sum_{j=0}^{\lfloor d/2 \rfloor} \binom{n}{d-2j}.$$

*In particular, if $d = o(n)$ then an upper capacity is $n^d/d!$.*

Anecdotal evidence in implementations indicates that the above estimate of upper capacity actually holds as an estimate of capacity as was the case for $d = 1$. There is some theoretical support for this though no complete proof. The main difficulty is that we cannot directly apply Proposition 3.2 to the matrix $\hat{\mathbf{U}}$ as the distribution induced on vertices of $\mathbb{B}^{N_d}$ as we build up generations according to Eqs. (7) and (8) is not uniform—indeed, we can only access $2^n$ out of the total of $2^{N_d}$ vertices. Note, however, that any two distinct vectors, $\mathbf{u}$ and $\mathbf{v}$, in $\mathbb{B}^n$ when expanded to vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ in $\mathbb{B}^{N_d}$ according to Eq. (8) become more and more nearly orthogonal as the number of generations increase. In fact, let $D$ be the Hamming distance between $\mathbf{u}$ and $\mathbf{v}$. Then it is easily verified that the Hamming distance, $\hat{D}$, between $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ is given by[3]

$$\hat{D} = \sum_{j \text{ odd}}^{d} \sum_{k \text{ odd}}^{D} \binom{D}{k}\binom{n-D}{j-k}.$$

(If $d$ is even replace the first sum by a sum over even indices, $j = 0, 2, \ldots, d$.) As $d$ increases the vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ approach orthogonality, and in fact, any pair of vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{B}^n$ result in orthogonal vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ in $\mathbb{B}^{2^{n-1}}$ when all odd (or even) generations are included—i.e., when $d$ is equal to $n$ or $n - 1$. To verify this note, for instance, that for any Hamming distance $0 < D < n$ between two vectors in $\mathbb{B}^n$ the corresponding Hamming distance $\hat{D}$ between the corresponding vectors in $\mathbb{B}^{2^{n-1}}$ when all odd generations are included is

$$\begin{aligned}\hat{D} &= \sum_{j \text{ odd}}^{n} \sum_{k \text{ odd}}^{D} \binom{D}{k}\binom{n-D}{j-k} \\ &= \sum_{k \text{ odd}}^{D} \binom{D}{k} \sum_{j \text{ odd}}^{n} \binom{n-D}{j-k}\end{aligned}$$

[3] For simplicity we use the convention $\binom{a}{b} = 0$ if $a < b$ or $b < 0$.

$$= 2^{n-D-1} \sum_{k \text{ odd}}^{D} \binom{D}{k}$$

$$= 2^{n-D-1} 2^{D-1}$$

$$= 2^{n-2}.$$

Hence $\langle \hat{\mathbf{u}}, \hat{\mathbf{v}} \rangle = 0$ for any two vectors $\mathbf{u} \neq -\mathbf{v}$ in $\mathbb{B}^n$ when all odd (or even) generations are included. The preceding analysis does not work when $D = n$; i.e., we start with two opposing vertices of the $n$-cube. However, even in this case note that the generated vectors $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$ become orthogonal if we include all even *and* odd generations. Thus, though the *statistical* dependence across components increases with the number of generations included, we may expect a concurrent building up of *linear* independence as the randomly chosen memories, $\mathbf{u}^\alpha$, result in more and more nearly orthogonal vectors $\hat{\mathbf{u}}^\alpha$. We may, hence, expect the nonsingularity probability estimate of Proposition 3.2 to improve for the generated matrices $\hat{\mathbf{U}}$.[4] In particular, let $N_d$ denote the length of the extended vectors $\hat{\mathbf{u}}$ for any choice of degree $d$ (which may depend on $n$).

CONJECTURE 3.4. *If the number of memories satisfies $m \leq N_d$ then the $N_d \times m$ extended matrix of memories, $\hat{\mathbf{U}}$, is full rank with probability approaching one as $n \to \infty$.*

This, in turn, would yield that the upper capacity estimate of Theorem 3.3 would actually be the estimate of the capacity of the higher-order spectral algorithm of degree $d$.

## 4. MAXIMAL CAPACITY

In this section we derive the maximal storage capacity of a higher-order neural network of degree $d$. The results are independent of any particular choice of algorithm, and depend only on the network architecture—a higher-order neural network of degree $d$. The maximal capacity, hence, delineates the upper limit on storage that can possibly be achieved by any particular choice of storage algorithm. We use a fundamental result due to Schläfli (1950) enumerating the number of linearly separable dichotomies of $m$ points in $N$-space.

Let $\mathbf{V} = \{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \subset \mathbb{R}^N$ be an $m$-set of points in $N$-space.

[4] The estimate of Proposition 3.2 may itself be rather weak. As conjectured by Komlós, we may expect the majority of singular ±1 matrices to be singular for the trivial reason that two rows or two columns coincide. If verified, this would, of course, improve the estimate of the probability of nonsingularity in Proposition 3.2 to $1 - O(n^2 2^{-n})$.

DEFINITION 4.1. A dichotomy $\mathcal{V} = \{\mathbf{V}^+, \mathbf{V}^-\}$ of $\mathbf{V}$ is *homogeneously linearly separable (hls)* if there is a vector $\mathbf{w} \in \mathbb{R}^N$ such that the inner product

$$\langle \mathbf{w}, \mathbf{v}\rangle \begin{matrix} > 0 & \text{if } \mathbf{v} \in \mathbf{V}^+ \\ < 0 & \text{if } \mathbf{v} \in \mathbf{V}^-. \end{matrix} \tag{10}$$

If Eq. (10) holds then $\mathbf{w}$ is said to be a *separating vector* for the dichotomy.

The following version of Schläfli's counting lemma estimates the probability that a randomly chosen dichotomy is homogeneously linearly separable. We give the proof for completeness. The presentation follows that of Wendel (1962) who utilizes the result in this form in a problem in geometric probability. [See also Cover (1965) for a slightly different approach.]

LEMMA 4.2. *Let* $\mathbf{V}$ *be an arbitrary m-set of points in* $\mathbb{R}^N$*, and let* $\mathcal{V}$ *be a dichotomy of* $\mathbf{V}$ *chosen independently of* $\mathbf{V}$*, and with equal probability,* $2^{-m}$*, from the set of dichotomies of* $\mathbf{V}$*. Then the probability,* $P_N^m$*, that* $\mathcal{V}$ *is homogeneously linearly separable is bounded by*

$$P_N^m \leq 2^{-(m-1)} \sum_{j=0}^{N-1} \binom{m-1}{j}. \tag{11}$$

*Moreover, a sufficient condition enabling us to replace the inequality above by equality is that the m-set of points* $\mathbf{V}$ *be chosen from a joint distribution which is such that* $\mathbf{V}$ *is in general position—i.e., all subsets of size N are linearly independent—with probability one.*

*Proof.* Let $D_N^m$ be the maximum number of dichotomies of an $m$-set of points in $\mathbb{R}^N$ that are hls. Then

$$P_N^m \leq 2^{-m} D_N^m.$$

In order to demonstrate the validity of Eq. (11) it suffices, hence, to show that

$$D_N^m = 2 \sum_{j=0}^{N-1} \binom{m-1}{j}. \tag{12}$$

Let $\mathbf{V}$ denote an $m$-set of points for which $D_N^m$ dichotomies are hls. (Such a set exists as $D_N^m \leq 2^m$ is finite.) Let $V^\alpha$ be the hyperplane orthogonal to $\mathbf{v}^\alpha$. Then $D_N^m$ is the number of path-components in $\mathbb{R}^N \setminus \bigcup_{\alpha=1}^m V^\alpha$ as each path-component is a maximally connected set of vectors all homogeneously separating the same dichotomy of $\mathbf{V}$.

Now consider the effect of deleting the hyperplane $V^m$. The remaining $m - 1$ hyperplanes determine $D_N^{m-1}$ path-components. These are of two types: (i) those path-components (say $Q_1$ in number) which have a nonnull intersection with the hyperplane $V^m$, and (ii) those path-components (say $Q_2$ in number) which do not intersect $V^m$. Clearly then, $D_N^{m-1} = Q_1 + Q_2$. With $V^m$ restored it cuts each path-component of type (i) in two, and leaves path-components of type (*ii*) undisturbed. Hence

$$D_N^m = 2Q_1 + Q_2 = D_N^{m-1} + Q_1.$$

Now the intersection of the $Q_1$ type (i) components with the hyperplane $V^m$ generates $Q_1$ path-components in $V^m \backslash \bigcup_{\alpha=1}^{m-1}(V^m \cap V^\alpha)$. As the sets $V^m \cap V^\alpha$ are just the hyperplanes in the $(N - 1)$-dimensional space $V^m$ orthogonal to the projection of the vectors $\mathbf{v}^\alpha$ into $V^m$, it follows that $Q_1 = D_{N-1}^{m-1}$. Hence

$$D_N^m = D_N^{m-1} + D_{N-1}^{m-1}.$$

This recursion with the obvious boundary conditions

$$D_N^1 = D_1^m = 2$$

yields the solution (12) which can be readily verified by induction.

To complete the proof we need to show that we can replace the inequality in Eq. (11) by equality if the $m$-set of points $\mathbf{V}$ is in general position with probability one. This follows immediately, however, from the simple observation that the proof above continues to work to estimate the number of hls dichotomies of any $m$-set of points which has an attribute which is preserved under projections. ■

We require the following technical result due to Chernoff (1952) which gives bounds for very large deviations in the tails of the binomial distribution.

LEMMA 4.3. *Fix* $\frac{1}{2} \leq c < 1$ *and let H denote the entropy function*

$$H(x) = -x \log_2 x - (1 - x)\log_2(1 - x) \qquad (0 < x < 1).$$

*Let p denote the probability that in M trials of a fair coin the number of successes is greater than or equal to cM. Then*

$$p = 2^{-M} \sum_{j=\lceil cM \rceil}^{M} \binom{M}{j} \leq 2^{-[1-H(c)]M}.$$

THEOREM 4.4. *$\overline{C_n} = 2\binom{n-1}{d}$ is a maximal upper capacity for zero-diagonal neural networks of degree $d$.*

*Proof.* Let $\mathbf{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ be a randomly specified $m$-set of memories whose components are generated from a sequence of symmetric Bernoulli trials. If each of the memories is to be $\mathscr{B}$-stable we require to find real coefficients, $w_{(i,I)}$, $I \in \hat{\mathcal{I}}_d$, $i \notin I$ such that for each $i \in [n]$, and $\alpha \in [m]$,

$$u_i^\alpha \sum_{I \in \hat{\mathcal{I}}_d : i \notin I} w_{(i,I)} u_I^\alpha > \mathscr{B}. \tag{13}$$

We first argue that without loss of generality we can restrict attention to a margin $\mathscr{B} = 0$. In fact, if there exist a choice of coefficients, $w_{(i,I)}$, such that

$$u_i^\alpha \sum_{I \in \hat{\mathcal{I}}_d : i \notin I} w_{(i,I)} u_I^\alpha > 0, \qquad i = 1, \ldots, n, \quad \alpha = 1, \ldots, m,$$

then, if $T > 0$ is the smallest of the sums above, the simple expedient of scaling all coefficients $w_{(i,I)}$ by a positive scalar greater than $\mathscr{B}/T$ will result in Eq. (13) being automatically satisfied.

Referring to the evolution rule (3) (with margin $\mathscr{B} = 0$) we see that each higher-order neuron in a zero-diagonal network of degree $d$ realizes a separating plane in $\binom{n-1}{d}$-space. For the memories to be fixed points we hence are required for each $i = 1, \ldots, n$ to find $N = \binom{n-1}{d}$ real coefficients $w_{(i,I)}$, $I \in \hat{\mathcal{I}}_d$, $i \notin I$ such that

$$u_i^\alpha = \operatorname{sgn}\left( \sum_{I \in \hat{\mathcal{I}}_d : i \notin I} w_{(i,I)} u_I^\alpha \right), \qquad \alpha = 1, \ldots, m. \tag{14}$$

Now fix $i$ and let $\mathscr{E}_n^i$ be the event that there is no weight vector $\mathbf{w}_i = [w_{(i,I)}]$ in $N$-space which separates the dichotomy of the extended $m$-set of memories, $[u_I^\alpha]$, with components varying over the set of indices $I \in \hat{\mathcal{I}}_d$: $i \notin I$, and $\alpha = 1, \ldots, m$, induced by Eq. (14)—i.e., the partition of the memories according to whether $u_i^\alpha$ is $-1$ or $+1$. Note that the term $u_i^\alpha$ does not appear anywhere in the sum or in the right-hand side of Eq. (14). As the components $u_i^\alpha$ are drawn from symmetric Bernoulli trials it follows that the dichotomy indicated in Eq. (14) is chosen independently of the extended $m$-set of memories. By Lemma 4.2 we hence have

$$\mathbf{P}\{\mathscr{E}_n^i\} = 1 - P_N^m \geq 1 - 2^{-(m-1)} \sum_{j=0}^{N-1} \binom{m-1}{j}. \tag{15}$$

Let $\mathcal{P}$ be the probability that there exists a zero-diagonal network of degree $d$ in which the fundamental memories are stable. Then

$$\mathcal{P} = 1 - \mathbf{P}\left\{\bigcup_{i=1}^{n} \mathscr{E}_n^i\right\} \le 1 - \mathbf{P}\{\mathscr{E}_n^i\}. \tag{16}$$

Set $M = m - 1$ for notational convenience. Using Eq. (15) with the upper bound for $\mathcal{P}$ in Eq. (16) we have

$$\mathcal{P} \le 2^{-M} \sum_{j=0}^{N-1} \binom{M}{j}.$$

Fix $\lambda > 0$ and choose $M = \lceil 2N(1 + \lambda)\rceil$. Then $N = c_1 M$ where $0 < c_1 < \frac{1}{2}$. Using Lemma 4.3 we hence have

$$\mathcal{P} \le 2^{-M} \sum_{j=0}^{c_1 M} \binom{M}{j} \le 2^{-[1-H(c_1)]M} \to 0, \qquad (n \to \infty).$$

Hence $2N + 1$ is a maximal upper capacity, and by Proposition 2.6 so is $2N = 2\binom{n-1}{d}$. ■

A maximal lower capacity of $N$ is readily demonstrated if an independence conjecture similar to the one earlier holds. Fix any index $i$ in $[n]$, and consider an extended set of $N$ memories $[u_I^\alpha]_{I\in\mathfrak{I}_d : i\in I}$, $\alpha \in [m]$, where each extended memory is a binary ($\pm 1$) vector of length $N$. Denote this set of (extended) memories by $\tilde{\mathbf{U}}$.

CONJECTURE 4.5. *The set of extended memories* $\tilde{\mathbf{U}}$ *is linearly independent with probability approaching one as* $n \to \infty$.

For a choice of $m \le \binom{n-1}{d}$, $P_N^m = 1$ for almost all choices of $m$ memories by Lemma 4.2 if the above holds. This will yield a lower maximal capacity of $N = \binom{n-1}{d}$. We can, however, hope for more: the following application of a result of Füredi (1986) provides a lower bound for the probability that a dichotomy of a randomly chosen $m$-set from the vertices of an $N$-cube is hls.

LEMMA 4.6. *Let an m-set of points be chosen independently from the uniform distribution over the vertices of the binary N-cube,* $\mathbf{B}^N$. *Then, if* $m \le 2N$, *the probability that an arbitrary dichotomy of the m-set of points is hls is bounded below by*

$$P_N^m = 2^{-(m-1)} \sum_{j=0}^{N-1} \binom{m-1}{j} - O(b^{-N}), \qquad (N \to \infty),$$

*where $b > 1$ is a fixed constant.*

*Remarks.* The exponentially small order term quoted above is a refinement of Füredi's original estimate of $O(N^{-1/2})$ using Proposition 3.2. The result eschews the general position requirement of Lemma 4.2. Specifically, the upper bound for $P_N^m$ in Eq. (11) is sharp if $m \leq 2N$ and the $m$-set of points is chosen independently from the uniform distribution on vertices of the $N$-cube.

Füredi's result makes it appear likely that, in fact, $2N = 2\binom{n-1}{d}$ is the maximal capacity of a zero-diagonal higher-order network of degree $d$. We again have a situation as in the previous section where we would like to apply the result not to the uniform distribution, but to the distribution corresponding to the $d$th generation of an $m$-set generated randomly from the uniform distribution on $\mathbb{B}^n$. If the above lemma continues to hold for this situation, then for $m \leq 2N$ we can replace the estimate (15) in the proof of the theorem above by

$$\mathbf{P}\{\mathscr{E}_n^i\} = 1 - 2^{-M} \sum_{j=0}^{N-1} \binom{M}{j} + O(b^{-N}),$$

where, again, we set $M = m - 1$. Using the union bound we have from Eq. (16) that

$$1 - n\mathbf{P}\{\mathscr{E}_n^i\} \leq \mathscr{P}.$$

Fix $0 < \lambda < \frac{1}{2}$ and choose $M = \lfloor 2N(1 - \lambda) \rfloor$. Under the above assumption we then have for $d = o(n)$ that

$$\begin{aligned} \mathscr{P} &\geq 1 - n\left[1 - 2^{-M} \sum_{j=0}^{N-1} \binom{M}{j} + O(b^{-N})\right] \\ &= 1 - n\left[2^{-M} \sum_{j=N}^{M} \binom{M}{j} + O(b^{-N})\right] \\ &= 1 - n\left[2^{-M} \sum_{j=c_\lambda M}^{M} \binom{M}{j} + O(b^{-N})\right], \end{aligned}$$

where $\frac{1}{2} < c_2 < 1$. As $M = \Omega(n^d)$ and $N \sim n^d/d!$ we then have by Chernoff's large deviation bound (Lemma 4.3) that for a choice of constant $c_3 > 0$

$$\mathcal{P} \geq 1 - n2^{-[1-H(c_2)]M} - O(nb^{-c_3 n^d}) \to 1, \qquad (n \to \infty).$$

So $2N + 1$ is a lower sequence of maximal capacities, and hence, so is $2N$ by Proposition 2.6 if Füredi's result holds in this case.

For the case $d = 1$ it is clear that Füredi's lemma holds *in toto* so that the above analysis works with $N = n - 1$. For the case of linear interactions, hence, we have shown the following

THEOREM 4.7. *The sequence $2n$ is the maximal capacity for zero-diagonal neural networks with linear interactions, $d = 1$.*

*Remark.* It is known that $2n$ is the capacity of a single linear threshold element [cf., for instance, Cover, 1965; Venkatesh and Psaltis, 1991]. The above result asserts that there is no decrease in capacity for the zero-diagonal network of $n$ neurons even though we now have a situation where $n$ neurons operate on the *same* set of memories.

## 5. Concluding Observations

1. For the case $d = 1$ Abu-Mostafa and St. Jacques (1985) demonstrate that with the requirement that *all* choices of $m$ vectors be stored as fixed points for some choice of zero-diagonal network, $m$ can be no larger than $n$. However, small pathological sets of vectors which cannot be stored can be found (Montgomery and Vijayakumar, 1986), and such pathologies make it difficult to achieve nontrivial deterministic capacities. The probabilistic setup adopted here essentially relaxes the requirement that *all* choices of $m$ vectors be storable to the requirement that *almost all* choices of $m$ memories be storable; pathological scenarios that cannot be stored form a set whose size is small compared to $\binom{2^n}{m}$, and are effectively ignored in this definition.

2. The maximal capacities for nonzero diagonal networks are of the same order as those for the zero-diagonal networks. Note, however, that we are required to put restrictions on the allowable choices of interactions. Specifically, consider the case $d = 1$. With a choice of identity matrix of interactions, $w_{ij} = \delta_{ij}$, it is clear that *all* states in $\mathbb{B}^n$ are stable with the same margin of stability. There is clearly no associative storage possible in this situation. To avoid situations of this type we have to put constraints on the allowable interactions so that the number of extraneous stable states do not become too large: specifically, the diagonal terms

should not dominate the nondiagonal terms. Similar examples hold for the higher-order cases.

3. The capacity estimates continue to hold if we are required to store random associations of the form $\mathbf{u}^\alpha \mapsto \mathbf{v}^\alpha$. We then call the vectors $\mathbf{v}^\alpha$ the *associated memories*. The spectral algorithm generalizes in a straightforward manner with the interaction matrix of coefficients of Eq. (9) modified to

$$\hat{\mathbf{W}} = \mathbf{V}\Lambda(\hat{\mathbf{U}}^T\hat{\mathbf{U}})^{-1}\hat{\mathbf{U}}^T$$

with $\mathbf{V}$ being the $n \times m$ matrix of associated memories.

4. The main unresolved issue in this work is the conjecture introduced in this paper that the linear independence property is preserved (strengthened!) when we consider higher generations of vectors chosen uniformly from $\mathbb{B}^n$. This is independent of the Komlós conjecture.

## Acknowledgments

We are grateful to the anonymous referee for his suggestions toward the improvement of the paper, and, in particular, for his comments on the linear independence issue. This work was supported in part by NSF Grants EET-8709198 and DMS-8800322, ONR Contract 411P006-01, and by Air Force Grant AFOSR 89-0523.

## References

Abu-Mostafa, Y. S., and St. Jacques, J. (1985), Information capacity of the Hopfield model, *IEEE Trans. Inform. Theory* **IT-31**, 461–464.

Baldi, P., and Venkatesh, S. S. (1987), Number of stable points for spin glasses and neural networks of higher orders, *Phys. Rev. Lett.* **58**, 913–916.

Baldi, P., and Venkatesh, S. S. (1988), On properties of networks of neuron-like elements, *in* "Neural Information Processing Systems" (D. Z. Anderson, Ed.), Amer. Inst. Phys., New York.

Baldi, P. (1988), Neural networks, orientations of the hypercube, and algebraic threshold functions, *IEEE Trans. Inform. Theory* **IT-34**, 523–530.

Chernoff, H. (1952), A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, *Ann. Math. Statist.* **23**, 493–507.

Cover, T. M. (1965), Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Trans. Elec. Comput.* **EC-14**, 326–334.

Füredi, Z. (1986), Random polytopes in the *d*-dimensional cube, *Discrete Comput. Geom.* **1**, 315–319.

Goles, E., and Vichniac, G. Y. (1986), Lyapunov functions for parallel neural networks, *in* "Neural Networks for Computing" (J. Denker, Ed.), Amer. Inst. Phys., New York.

Hopfield, J. J. (1982), Neural networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci. USA* **79**, 2554–2558.

Kahn, J., Komlós, J., and Szemerédi, E. (1990), Singularity probabilities for random ±1 matrices, preprint.

Lee, Y. C., Doolen, G., Chen, H. H., Sun, G. Z., Maxwell, T., Lee, H. Y., and Giles, C. L. (1986), Machine learning using a higher-order correlation network, *Physica* **22D**, 276–306.

Maxwell, T., Giles, C. L., Lee, Y. C., and Chen, H. H. (1986), Non-linear dynamics of artificial neural systems, *in* "Neural Networks for Computing" (J. Denker, Ed.), Amer. Inst. Phys., New York.

McCulloch, W. W., and Pitts, W. (1943), A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.* **5**, 115–133.

Montgomery, B. L., and Vijayakumar, B. V. K. (1986), Evaluation of the use of the Hopfield neural network model as a nearest neighbour algorithm, *Appl. Opt.* **25**, 3759–3766.

Personnaz, L., Guyon, I., and Dreyfus, G. (1985), Information storage and retrieval in spin-glass like neural networks, *J. Physique Lett.* **46**, L359–L365.

Psaltis, D., and Park, C. H. (1986), Nonlinear discriminant functions and associative memories, *in* "Neural Networks for Computing" (J. Denker, Ed.), Amer. Inst. Phys., New York.

Schläfli, L. (1950), "Gesammelte Mathematische Abhandlungen I," pp. 209–212, Verlag Birkhäuser, Basel, Switzerland.

Vapnik, V. N. (1982), Estimation of dependences based on empirical data, *in* "Springer Series in Statistics," Springer, New York/Berlin.

Venkatesh, S. S. (1986), "Linear Maps with Point Rules: Applications to Pattern Classification and Associative Memory," PhD thesis, California Institute of Technology.

Venkatesh, S. S., and Psaltis, D. (1989), Linear and logarithmic capacities in associative neural networks, *IEEE Trans. Inform. Theory* **IT-35**, 558–568.

Venkatesh, S. S., and Baldi, P. (1989), Random interactions in higher-order neural networks, submitted for publication.

Venkatesh, S. S., and Baldi, P. (1991), Programmed interactions in higher-order neural networks: The outer-product algorithm, *J. Compl.*, to appear.

Venkatesh, S. S., and Psaltis, D. (1991), On reliable computation with formal neurons, *IEEE Trans. Pattern Anal. Mach. Intell.*, to appear Aug. 1991.

Wendel, J. G. (1962), A problem in geometric probability, *Math. Scand.* **11**, 109–111.

JOURNAL OF COMPLEXITY 7, 443–479 (1991)

# Programmed Interactions in Higher-Order Neural Networks: The Outer-Product Algorithm*

SANTOSH S. VENKATESH†

*Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania 19104*

PIERRE BALDI‡

*Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109*

Received February 15, 1989

Recent results on the memory storage capacity of the outer-product algorithm indicate that the algorithm stores of the order of $n/\log n$ memories in a network of $n$ fully interconnected linear threshold elements when it is required that each memory be exactly recovered from a probe which is close enough to it. In this paper a rigourous analysis is presented of generalizations of the outer-product algorithm to higher-order networks of densely interconnected polynomial threshold units of degree $d$. Precise notions of memory storage capacity are formulated, and it is demonstrated that both static and dynamic storage capacities of all variants of the outer-product algorithm of degree $d$ are of the order of $n^d/\log n$.

## 1. INTRODUCTION

### 1.1. *Overview*

Formal neural network models of densely interconnected linear threshold gates have found considerable recent application in a variety of problems such as associative memory, error correction, and optimization. In

* Presented in part at the IEEE Conference on Neural Information Processing Systems, Denver, Colorado, November, 1987, and at the IEEE International Symposium on Information Theory, Kobe, Japan, June, 1988.

† Corresponding author.

‡ Also the Division of Biology, California Institute of Technology, Pasadena, CA 91125.

0885-064X/91 $3.00

these networks the model neurons are linear threshold elements with $n$ real inputs and a single binary output. Each neuron is characterized by $n$ real *weights*, say $w_{i1}, \ldots, w_{in}$, and a real *threshold* (which we assume to be zero for simplicity). Given inputs $u_1, \ldots, u_n$, the $i$th neuron produces an output $v_i \in \{-1, 1\}$ which is simply the sign of the weighted sum of inputs:

$$v_i = \operatorname{sgn}\left(\sum_{j=1}^{n} w_{ij} u_j\right). \tag{1}$$

A fully interconnected network of $n$ formal neurons is then completely characterized by an $n \times n$ matrix of real weights.

A number of authors have recently begun to investigate more general networks obtained by incorporating polynomial instead of linear interactions between the threshold processing elements. Specifically, the linear threshold elements of Eq. (1) are replaced by *polynomial threshold elements* of given degree $d$; the output, $v_{i_1}$, of the $i_1$th higher-order neuron in response to inputs $u_1, \ldots, u_n$, is given by the sign of an algebraic form

$$v_{i_1} = \operatorname{sgn}\left(\sum_{1 \le i_2 \le \cdots \le i_{d+1} \le n} w_{i_1 i_2 \cdots i_{d+1}} u_{i_2} \cdots u_{i_{d+1}}\right). \tag{2}$$

The number of interaction coefficients is increased to $n^{d+1}$ from the $n^2$ weights for the case of linear interactions. The added degrees of freedom in the interaction coefficients can potentially result in enhanced flexibility and programming capability over the linear case: in general, the computational gains match the added degrees of freedom (Venkatesh and Baldi, 1991).[1]

In this paper we estimate the maximum number of arbitrarily specified vectors (*memories*) that can be reliably stored by the outer-product algorithm in a higher-order network of degree $d$. We estimate both *static capacities*—where we require the memories to be stored as fixed points of the network—and *dynamic capacities*—where the specified memories are required to be *attractors* as well. Our principal results are as follows:

*The static and dynamic storage capacities of all variants of the outer-product algorithm generalized to degree $d$ are of the order of $n^d/\log n$ memories.*

The maximal storage capacities that can be realized in a higher-order network of degree $d$ are of the order of $n^d$ (Venkatesh and Baldi, 1991), so

[1] Higher-order neural with *random* interactions lead to rather different computational issues. We deal with these in a concurrent paper (Venkatesh and Baldi, 1989a).

that the outer-product prescription for storing memories loses a logarithmic factor in capacity. This, however, is somewhat offset by the ease of programmability and the simplicity of the algorithm.

*Notation.* We utilize standard asymptotic notation and introduce two (nonstandard) notations. Let $\{x_n\}$ and $\{y_n\}$ be positive sequences. We denote:

1. $x_n = \Omega(y_n)$ if there is a positive constant $K$ such that $x_n/y_n \geq K$ for all $n$;

2. $x_n = O(y_n)$ if there exists a positive constant $L$ such that $x_n/y_n \leq L$ for all $n$;

3. $x_n = \Theta(y_n)$ if $x_n = O(y_n)$ and $x_n = \Omega(y_n)$;

4. $x_n \sim y_n$ if $x_n/y_n \to 1$ as $n \to \infty$; we also use $x_n \lesssim y_n$ if $x_n/y_n \leq 1$ for $n$ large enough, and $x_n \gtrsim y_n$ if $x_n/y_n \geq 1$ for $n$ large enough;

5. $x_n = o(y_n)$ if $x_n/y_n \to 0$ as $n \to \infty$.

We also say that a positive sequence, $M_n$, is *polynomially increasing* if $\log M_n = \Theta(\log n)$ for any fixed base of logarithm. (All logarithms in the exposition are to the base $e$.) We denote by $\mathbb{B}$ the set $\{-1, 1\}$, and by $[n]$ the set $\{1, 2, \ldots, n\}$. Finally, by an *ordered multiset* we mean an ordered collection of elements where repetition is allowed.

*Organization.* The basic definitions were set up in a preceding paper (Venkatesh and Baldi, 1991), and we briefly summarize them in the rest of this section. In Section 2 we describe the generalization of the outer-product algorithm to higher-order networks. In Section 3 we present the main theorem on the static storage capacity of the outer-product algorithm. In Section 4 we prove the theorem for the simplest case of first-order interactions where the neurons are linear threshold elements; the proof techniques used here are somewhat simpler than those for the general case. In Section 5 we prove the main theorem on the static capacity of the higher-order outer-product algorithm. Following the proof of the main theorem, in Section 6 we then infer similar static capacity results for the outer-product algorithm when self-interconnections are proscribed—the zero-diagonal case. In Section 7 we consider the dynamic case. Theorems are proved in the body of the paper, while technical results needed in the proofs are confined to the Appendix.

### 1.2. *Higher-Order Neural Networks*

We consider recurrent networks of polynomial threshold units each of which yields an instantaneous state of $-1$ or $+1$. More formally, for positive integers $n$ and $d$, let $\mathcal{I}_d$ be the set of ordered multisets of cardinality $d$ of the set $[n]$. Clearly $|\mathcal{I}_d| = n^d$. For any subset $I = (i_1, i_2, \ldots, i_d) \in \mathcal{I}_d$, and for every $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{B}^n$, set $u_I = \prod_{j=1}^{d} u_{i_j}$.

DEFINITION 1.1. A *higher-order neural network of degree d* is characterized by a set of $n^{d+1}$ real weights $w_{(i,I)}$ indexed by the ordered pair $(i, I)$ with $i \in [n]$ and $I \in \mathcal{I}_d$, and a real *margin of operation* $\mathcal{B} \geq 0$. The network dynamics are described by trajectories in a state space of binary $n$-tuples, $\mathbb{B}^n$: for any state $\mathbf{u} \in \mathbb{B}^n$ on a trajectory, a component update $u_i \mapsto u_i'$ is permissible iff

$$u_i' = \begin{cases} -1 & \text{if } \sum_{I\in\mathcal{I}_d} w_{(i,I)}u_I < -\mathcal{B} \\ -u_i & \text{if } -\mathcal{B} \leq \sum_{I\in\mathcal{I}_d} w_{(i,I)}u_I \leq \mathcal{B} \\ 1 & \text{if } \sum_{I\in\mathcal{I}_d} w_{(i,I)}u_I > \mathcal{B}. \end{cases} \tag{3}$$

The evolution may be *synchronous* with all components of $\mathbf{u}$ being updated according to the rule (3) at each epoch, or *asynchronous* with at most one component being updated per epoch according to Eq. (3).

The network is said to be *symmetric* if $w_{(i,I)} = w_{(j,J)}$ whenever the $(d + 1)$-tuples of indices $(i, I)$ and $(j, J)$ are permutations of each other. The network is said to be *zero-diagonal* if $w_{(i,I)} = 0$ whenever any index repeats in $(i, I)$.

Let $\hat{\mathcal{I}}_d$ denote the set of subsets of $d$ elements from $[n]$; $|\hat{\mathcal{I}}_d| = \binom{n}{d}$. Combining all redundant terms in Eq. (3), for symmetric, zero-diagonal networks a component update $u_i \mapsto u_i'$ is permissible iff

$$u_i' = \begin{cases} -1 & \text{if } \sum_{I\in\hat{\mathcal{I}}_d : i\notin I} w_{(i,I)}u_I < -\mathcal{B} \\ -u_i & \text{if } -\mathcal{B} \leq \sum_{I\in\hat{\mathcal{I}}_d : i\notin I} w_{(i,I)}u_I \leq \mathcal{B} \\ 1 & \text{if } \sum_{I\in\hat{\mathcal{I}}_d : i\notin I} w_{(i,I)}u_I > \mathcal{B}. \end{cases} \tag{4}$$

As in the case of recurrent networks of linear threshold units, the dynamics of recurrent higher-order networks can be described by Lyapunov functions (Hopfield, 1982; Goles and Vichniac, 1986; Maxwell *et al.*, 1986; Psaltis and Park, 1988; Venkatesh and Baldi, 1989a) under suitable conditions on the interaction weights. Consider, in particular, a symmetric, zero-diagonal network of degree $d$. For $\mathbf{u} \in \mathbb{B}^n$ define the *algebraic Hamiltonian of degree d* by

$$\hat{H}_d(\mathbf{u}) = -\sum_{I\in\hat{\mathcal{I}}_{d+1}} w_I u_I.$$

We then have the following assertion which we give without proof.

PROPOSITION 1.2. *The function $\hat{H}_d$ is nonincreasing under the evolution rule* (4) *in asynchronous operation.*

In light of such results we are interested in the number of fixed points of the network and, in the associative memory application, in the trajectories leading into the fixed points.

DEFINITION 1.3. Let $\mathcal{B} \geq 0$ be fixed. A state $\mathbf{u} \in \mathbb{B}^n$ of a higher-order neural network of degree $d$ is said to be *$\mathcal{B}$-stable* iff

$$u_i \sum_{I \in \mathcal{I}_d} w_{(i,I)} u_I > \mathcal{B}, \qquad i = 1, \ldots, n.$$

Likewise, a state $\mathbf{u} \in \mathbb{B}^n$ of a zero-diagonal network is said to be $\mathcal{B}$-stable iff

$$u_i \sum_{I \in \mathcal{I}_d : i \notin I} w_{(i,I)} u_I > \mathcal{B}, \qquad i = 1, \ldots, n.$$

It is easy to see that $\mathcal{B}$-stable states are fixed points of the higher-order network with evolution under a margin $\mathcal{B}$. The notion of $\mathcal{B}$-stable states is explored further in Komlós and Paturi (1988) and Venkatesh and Baldi (1989a).

We refer to the data to be stored as *memories*. By an *algorithm* for storing memories we mean a prescription for generating the interaction weights of a higher-order network of degree $d$ as a function of any given set of memories. We will investigate the maximum number of arbitrarily specified memories that can be made fixed in the network by an algorithm; this is a measure of the *capacity* of the algorithm to store data.

### 1.3. *Memory Storage Capacity*

Let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of memories to be stored in a higher-order network of degree $d$. We assume that the memories are chosen randomly from the probability space of an unending series of symmetric Bernoulli trials: specifically, the memory components, $u_i^\alpha$, $i \in [n]$, $\alpha \in [m]$, are i.i.d. random variables with

$$\mathbf{P}\{u_i^\alpha = -1\} = \mathbf{P}\{u_i^\alpha = +1\} = \tfrac{1}{2}.$$

In the following we assume that the network architecture is specified to be a higher-order network of degree $d$.

DEFINITION 1.4. We say that $C_n$ is a *capacity function* (or simply, *capacity*) for an algorithm iff, for every choice of $\delta > 0$, the following two conditions hold as $n \to \infty$:

(a) The probability that all the memories are fixed points of the network generated by the algorithm tends to one whenever $m \leq (1 - \delta)C_n$;

(b) The probability that at least one of the memories is not a fixed point of the network generated by the algorithm tends to one whenever $m \geq (1 + \delta)C_n$.

If a sequence satisfies condition (a) we call it a *lower capacity function* and denote it by $\underline{C}_n$. Likewise, if a sequence satisfies condition (b) we call it an *upper capacity function* and denote it by $\overline{C}_n$.

Thus, if a capacity function exists for an algorithm, then it is both a lower and an upper capacity function for the algorithm. Define an equivalence class $\mathscr{C}$ of (lower/upper) capacity functions by $C_n, C'_n \in \mathscr{C} \Leftrightarrow C_n \sim C'_n$. We call any member of $\mathscr{C}$ the (lower/upper) capacity (if $\mathscr{C}$ is nonempty). Note that the definitions ensure that if any capacity function exists then the equivalence class of capacity functions is uniquely defined (Venkatesh and Baldi, 1991). (This is not true, however, for lower and upper capacities which are always guaranteed to exist.)

The above definitions of capacity require that *all* the memories are fixed points with probability approaching one. We obtain weaker definitions of capacity if we require just that *most* of the memories be fixed points.

DEFINITION 1.5. We say that $C_n^w$ is a *weak capacity function* (or simply, *weak capacity*) for an algorithm iff, for every choice of $\delta > 0$, the following two conditions hold as $n \to \infty$:

(a) The expected number of memories that are fixed points is $m(1 - o(1))$ whenever $m \leq (1 - \delta)C_n^w$;

(b) The expected number of memories that are fixed points is $o(m)$ whenever $m \geq (1 + \delta)C_n^w$.

If a sequence satisfies condition (a) we call it a *weak lower capacity function* and denote it by $\underline{C}_n^w$. Likewise, if a sequence satisfies condition (b) we call it a *weak upper capacity function* and denote it by $\overline{C}_n^w$.

We again define an equivalence class $\mathscr{C}^w$ of (lower/upper) capacity functions by $C_n^w, \hat{C}_n^w \in \mathscr{C}^w \Leftrightarrow C_n^w \sim \hat{C}_n^w$. We call any member of $\mathscr{C}^w$ the weak (lower/upper) capacity (if $\mathscr{C}^w$ is nonempty).

For the network to function as an associative memory we require that it corrects for errors in inputs sufficiently close to the stored memories.

DEFINITION 1.6. For a given mode of operation (synchronous or asynchronous) and a chosen time scale of operation (synchronous one-step, synchronous multiple-step, or asynchronous multiple-step) we say that a memory is a *$\rho$-attractor* for a choice of parameter $0 \leq \rho < \frac{1}{2}$ iff a randomly chosen state in the Hamming ball of radius $\rho n$ at the memory is mapped into the memory, within the given time scale, and for the given mode of operation, with probability approaching one as $n \to \infty$.[2]

In a manner cor.pletely analogous to the definitions of capacity above, we can now define *ρ-attractor capacities* and *weak ρ-attractor capacities* for the given mode of operation and the given time scale of operation by replacing the requirement of stable memories by the requirement that the memories be $\rho$-attractors.

## 2. The Outer-Product Algorithm

### 2.1. *The Classical Hebb Rule*

The outer-product algorithm (a special case of what is known as the Hebb rule) has been proposed by several authors as appropriate in a model of physical associative memory. While the algorithm is of some antiquity, formal analyses of the performance of the algorithm have, however, become available only recently (cf. McEliece *et al.*, 1987; Newman, 1988; and Kon:lós and Paturi, 1988). [For related nonrigourous results based upon replica calculations and statistical physics see, for instance, Amit *et al.*, 1985, and Peretto and Niez, 1986.]

Let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of memories. We will assume that the components, $u_i^\alpha$, $i = 1, \ldots, n$, $\alpha = 1, \ldots, m$, are drawn from a sequence of symmetric Bernoulli trials. For the linear case $d = 1$ the outer-product algorithm prescribes the interaction weights, $w_{ij}$, according to the rule

$$w_{ij} = \sum_{\nu=1}^{m} u_i^\nu u_j^\nu - gm\delta_{ij}, \qquad i, j = 1, \ldots, n,$$

where $g$ is a parameter with $0 \leq g \leq 1$, and $\delta_{ij}$ is the Kronecker delta.

It can be easily seen that in this algorithm the memories are stable with high probability provided $m$ is small compared to $n$; further, the construction utilizing outer-products of the memories results in a symmetric interaction matrix which in turn ensures that stable memories are attractors. The algorithm hence functions as a viable associative memory. McEliece *et al.* (1987) (cf. also Komlós and Paturi, 1988) carried out precise analytical calculations of the storage capacity of the outer-product algorithm

[2] For linear interactions, $d = 1$, Komlós and Paturi (1988) have investigated the more stringent case where they require the *entire* Hamming ball of radius $\rho n$ around a memory to be attracted to the memory.

under a variety of circumstances and showed that the capacity of the outer-product algorithm is of the order of $n/\log n$.[3]

The attractiveness of the outer-product algorithm for associative memory has led several investigators including Lee *et al.* (1986), Maxwell *et al.* (1986), Psaltis and Park (1986), and Baldi and Venkatesh (1987, 1988) to independently propose higher-order extensions of the algorithm.

### 2.2. *Outer-Products of Higher Degree*

While the results of McEliece *et al.* (1987) indicate that for the linear case $d = 1$, the capacity of the outer-product algorithm does not depend on whether self-connections are present or absent, the same does not continue to hold true for higher-order generalizations of the algorithm.

As before, we consider an $m$-set of memories, $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$, whose components are chosen from a sequence of symmetric Bernoulli trials. Consider first a network of $n$ higher-order neurons with dynamics specified by Eq. (3). For every $i$ in $[n]$ and ordered multiset $I \in \mathcal{I}_d$ the *outer-product algorithm of degree d* specifies the interaction coefficients, $w_{(i,I)}$, as a sum of generalized outer-products

$$w_{(i,I)} = \sum_{\nu=1}^{m} u_i^\nu u_I^\nu. \tag{5}$$

For the zero-diagonal case we use the same prescription to specify each $w_{(i,I)}$ with $i \in [n]$ and $I \in \hat{\mathcal{I}}_d$, and dynamics specified by Eq. (4).

While heuristic arguments suggest that the increase in the available degrees of freedom in the specification of the interaction coefficients would result in a commensurate increase in the fixed point storage capacity (Peretto and Niez, 1986; Baldi and Venkatesh, 1987), hitherto no rigorous estimates of storage capacity have been demonstrated.[4] We provide a formal analysis in the subsequent sections.

## 3. Fixed Points and Static Capacity

### 3.1. *The Main Result*

Consider a network of degree $d$. By the evolution rule (3), if the $i$th component of the $\alpha$th memory is to be stable, we require that

[3] The capacity estimates of McEliece *et al.* apply to the case where the memories are required to be stable—or, more generally, where they are required to be attractors—which will be our principal consideration in this paper. A somewhat different computational feature of the algorithm has been investigated by Newman (1988) and Komlós and Paturi (1988) who demonstrated that if errors are permitted in recall of the memories then the capacity of the outer-product algorithm can, in fact, increase linearly with $n$ [cf. also the epsilon capacity results of Venkatesh (1986) and Venkatesh and Psaltis (1991) in this regard].

[4] See Newman (1988), however, for investigations along a slightly different track.

$$u_i^\alpha \sum_{I\in\mathcal{I}_d} w_{(i,I)} u_I^\alpha > \mathcal{B}.$$

If each of the memories is to be a fixed point of the network we require $nm$ equations of the above form to be simultaneously satisfied, one per memory component.

Now select the coefficients $w_{(i,I)}$ according to prescription (5) for the outer-product algorithm of degree $d$. For each $n$ define the sequence of doubly indexed random variables $X_n^{i,\alpha}$ with

$$X_n^{i,\alpha} = u_i^\alpha \sum_{I\in\mathcal{I}_d} w_{(i,I)} u_I^\alpha = u_i^\alpha \sum_{\nu=1}^{m} \sum_{I\in\mathcal{I}_d} u_i^\nu u_I^\nu u_I^\alpha = n^d + \sum_{\nu\neq\alpha}\left(u_i^\alpha u_i^\nu \sum_{I\in\mathcal{I}_d} u_I^\alpha u_I^\nu\right). \tag{6}$$

Setting for $\nu \neq \alpha$

$$Y_n^{i,\alpha,\nu} = u_i^\alpha u_i^\nu \sum_{I\in\mathcal{I}_d} u_I^\alpha u_I^\nu = u_i^\alpha u_i^\nu \left(\sum_{j=1}^{n} u_j^\alpha u_j^\nu\right)^d, \tag{7}$$

we get

$$X_n^{i,\alpha} = n^d + \sum_{\nu\neq\alpha} Y_n^{i,\alpha,\nu}. \tag{8}$$

The evolution rule (3) will fail to retrieve the $i$th component of the $\alpha$th memory, $u_i^\alpha$, if $X_n^{i,\alpha} \leq \mathcal{B}$. If we identify the term $n^d$ as the "signal" term and the term $\Sigma_{\nu\neq\alpha} Y_n^{i,\alpha,\nu}$ as the "noise" term, a memory is $\mathcal{B}$-stable if the signal term less the margin exceeds the noise term for each component.

Let $\mathcal{E}_n^{i,\alpha}$ denote the event $\{X_n^{i,\alpha} \leq \mathcal{B}\}$, and let $\mathcal{E}_n = \bigcup_{i=1}^n \bigcup_{\alpha=1}^m \mathcal{E}_n^{i,\alpha}$ be the event that one or more memory components is not retrieved (i.e., is not $\mathcal{B}$-stable). We are interested in the probability, $\mathbf{P}\{\mathcal{E}_n\}$, of the event $\mathcal{E}_n$: we would like $m$ to be as large as possible while keeping the probability of $\mathcal{E}_n$ small, i.e., $m$ as large as possible while keeping the probability of *exact* retrieval of each of the memories high. For notational simplicity we henceforth suppress the $i$, $\alpha$ dependence of the random variables $X_n^{i,\alpha}$ and $Y_n^{i,\alpha,\nu}$ except where there is possibility of confusion. Denote

$$\mu_n \triangleq \mathbf{E}\{Y_n^\nu\},$$

and for each $d$ let

$$\lambda_d \triangleq \frac{(2d)!}{(d)!2^d}. \tag{9}$$

The following theorem is the main result of this section; it provides an estimate of the storage capacity of the outer-product algorithm of degree $d$.

THEOREM 3.1. *Consider a higher-order neural network of degree $d$ with weights chosen according to the outer-product algorithm of Eq.* (5) *and with a choice of margin $\mathcal{B} = m\mu_n$ in the evolution rule* (3). *For any fixed* $\varepsilon > 0$ and $\varpi > 0$:

1. *If, as $n \to \infty$, we choose $m$ such that*

$$m = \frac{(1-\varpi)n^d}{2(2d+1)\lambda_d \log n}\left[1 + \frac{2\log\log n + 2\log 2(d+1)\lambda_d\sqrt{\varepsilon}}{(2d+1)\log n} - O\left(\frac{\log\log n}{(\log n)^2}\right)\right], \tag{10}$$

*then the probability that each of the memories is $m\mu_n$-stable is $\gtrsim 1 - \varepsilon$;*

2. *If, as $n \to \infty$, we choose $m$ such that*

$$m = \frac{(1-\varpi)n^d}{2(d+1)\lambda_d \log n}\left[1 + \frac{\log\log n + \log 2\varepsilon(d+1)\lambda_d}{\log n} - O\left(\frac{\log\log n}{(\log n)^2}\right)\right], \tag{11}$$

*then the expected number of memories that are $m\mu_n$-stable is $\gtrsim m(1 - \varepsilon)$.*

*Remarks.* The size of the margin of operation is dictated by the expected size of the noise term for a typical state which is not a memory. As we will see in the subsequent development, the expected value of the noise term can be as large as the order of $mn^{(d-1)/2}$. If this is not compensated for in the margin of operation a large number of extraneous states (nonmemories) will also become fixed points of the system. Note also that relaxing the requirement that *all* the memories be stable to just requiring that *most* of the memories be stable effects (roughly) a twofold increase in the number of memories that can be stored.

COROLLARY 3.2. *For a given degree of interaction $d \gtrsim 1$ and margin $m\mu_n$, the sequence*

$$\underline{C}_n = \left(\frac{(d)!2^{d-1}}{(2d+1)!}\right)\frac{n^d}{\log n}$$

*is a lower capacity for the outer-product algorithm.*

COROLLARY 3.3. *For a given degree of interaction $d \geq 1$ and margin $m\mu_n$, the sequence*

$$\underline{C}_n^w = \left(\frac{(d)!2^{d-1}}{(2d)!(d+1)}\right)\frac{n^d}{\log n}$$

*is a weak lower capacity for the outer-product algorithm.*

*Remark.* Slightly sharper bounds are derived in Section 4 for the case $d = 1$.

### 3.2. *Outline of the Proof*

The main step involved in the proof of the theorem is the estimation of the probability, $\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} = \mathbf{P}\{X_n \leq m\mu_n\}$, that one component of a memory is not retrieved. We will utilize techniques from the theory of large deviations of a sum of random variables from its mean to estimate this probability. Over the next two sections we demonstrate that for the range of $m$ we consider, the following estimate holds: for any $\varpi > 0$

$$\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} \leq m \exp\left\{-\frac{(1-\varpi)n^d}{2\lambda_d m}\right\} \qquad (n \to \infty). \tag{12}$$

The probability that one or more memory components are not retrieved is less than $nm$ times the probability that one memory component is not retrieved; likewise, the expected fraction of memories that is not $\mathscr{B}$-stable is just the probability that one memory is not $\mathscr{B}$-stable, and this probability is bounded by $n$ times the probability that one memory component is not retrieved. Using the estimate of Eq. (12) together with a choice of $m$ according to Eq. (10) and (11), respectively, yields an upper bound of $\varepsilon$ for these probabilities, and concludes the proof.

The two corollaries follow as a consequence of uniformity: the probability that all the memories are $\mathscr{B}$-stable decreases monotonically as the number of memories increases. If, for instance, for any fixed $\delta > 0$ the number of memories is chosen to be equal to $(1 - \delta)$ times the capacity estimate of Corollary 3.2, it is easy to see that for large $n$ the number of memories will be less than that specified by Eq. (10). The resulting probability that all the memories are $\mathscr{B}$-stable will hence be asymptotically better than $1 - \varepsilon$. A similar line of reasoning also establishes Corollary 3.3.

The main idea in establishing Eq. (12) is to exploit the fact that the r.v.'s $Y_n^\nu$, $\nu \neq \alpha$ defined in (7) are i.i.d. Referring to (8), the probability that a memory component is not retrieved is just the probability that the sum, $\sum_{\nu\neq\alpha}(Y_n^\nu - \mu_n)$, of $(m - 1)$ zero-mean, i.i.d. r.v.'s is less than or equal to

$-n^d + \mu_n$. As we will see in the next two sections, a careful estimation of the mean, $\mu_n$, of the r.v.'s $Y_n^\nu$ will yield that $\mu_n = o(n^d)$. It suffices, hence, to estimate the probability that $\sum_{\nu \neq \alpha}(Y_n^\nu - \mu_n) \leq -n^d$; i.e., to estimate the probability that the sum of r.v.'s $Y_n^\nu$ deviates from the mean by the large deviation $n^d$.

For the case of first-order interactions, $d = 1$, the situation simplifies somewhat. For this case the r.v.'s $(Y_n^\nu - \mu_n)$ themselves turn out to be the sum of $(n - 1)$ i.i.d., symmetric $\pm 1$ r.v.'s, and the large deviation estimate for the probability that a memory component is not retrieved can be obtained by an application of the generalized Chebyshev inequality. We present the derivation of the probability estimate for this case in Section 4.

For $d > 1$ additional problems arise as the r.v. $Y_n^\nu$ has an infinite moment generating function. In particular, the Chebyshev estimates of Eqs. (33) and (34) in the Appendix work only trivially. We tackle this case in Section 5. The results needed here are two large deviation lemmas (A.6 and A.7) found in the Appendix.

## 4. First-Order Interactions

We begin with the following elementary observation.

*Fact* 4.1. Let $b_1, \ldots, b_N$ be i.i.d., symmetric, $\pm 1$ r.v.'s. Let $a_1, \ldots, a_N$ be any set of $\pm 1$ r.v.'s independent of the r.v.'s $b_k$, $k = 1, \ldots, N$. Then the r.v.'s $Z_k = a_k b_k$, $k = 1, \ldots, N$ are i.i.d., symmetric, $\pm 1$ r.v.'s.

*Remark.* Note that the r.v.'s $a_k$ need not be symmetric and may depend on each other.

Lemma 4.2 below is a particular application of Chebyshev's inequality. The result is an asymptotic expression for $\mathbf{P}\{\mathcal{E}_n^{i,\alpha}\}$, the probability that a particular memory component is not retrieved. The result agrees with what would be obtained by a naïve application of the Central Limit Theorem.

Lemma 4.2. *Let the order of interaction be $d = 1$ and let $\mathcal{B} = m$ be the margin of operation. If the number of memories, $m$, is chosen such that $m = o(n)$ and $m/\sqrt{n} \to \infty$, then*

$$\mathbf{P}\{\mathcal{E}_n^{i,\alpha}\} \leq \exp\left\{-\left(\frac{n}{2m}\right)\right\} \qquad (n \to \infty). \tag{13}$$

*Proof.* From Eq. (6) we can write

$$X_n = n + m - 1 + \sum_{\nu \neq \alpha} \sum_{j \neq i} Z_j^\nu,$$

where, for fixed $i$ and $\alpha$, we define the random variables $Z_j^\nu = u_i^\alpha u_i^\nu u_j^\alpha u_j^\nu$. Note that by Fact 4.1 the r.v.'s $Z_j^\nu, j \neq i, \nu \neq \alpha$ are i.i.d., symmetric, $\pm 1$ r.v.'s.[5] By Corollary A.2 we have for a choice of margin $\mathfrak{B} = m$ that

$$\begin{aligned}\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} = \mathbf{P}\{X_n \leq m\} &= \mathbf{P}\left\{\sum_{\nu\neq\alpha}\sum_{j\neq i} Z_j^\nu \leq -n+1\right\}\\ &\leq \inf_{r\geq 0} e^{-r(n-1)}\mathbf{E}\{e^{-r\Sigma_{\nu\neq\alpha}\Sigma_{j\neq i}Z_j^\nu}\}\\ &= \inf_{r\geq 0} e^{-r(n-1)}\mathbf{E}\left\{\prod_{\nu\neq\alpha}\prod_{j\neq i} e^{-rZ_j^\nu}\right\}.\end{aligned}$$

The terms in the product, $e^{-rZ_j^\nu}$, $\nu \neq \alpha, j \neq i$ are independent r.v.'s as the r.v.'s $Z_j^\nu$ are independent. The expectation of the product of r.v.'s above can, hence, be replaced by the product of expectations. Accordingly, denoting by $Z$ an r.v. which takes on values $-1$ and $1$ only, each with probability $\frac{1}{2}$, we have

$$\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} \leq \inf_{r\geq 0} e^{-r(n-1)}[\mathbf{E}(e^{-rZ})]^{(m-1)(n-1)} = \inf_{r\geq 0} e^{-r(n-1)}(\cosh r)^{(m-1)(n-1)}.$$

Now, for every $r \in \mathbb{R}$ we have $\cosh r \leq e^{r^2/2}$. Hence

$$\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} \leq \inf_{r\geq 0} \exp\left(\frac{r^2(m-1)(n-1)}{2} - r(n-1)\right) = \exp\left(-\frac{(n-1)}{2(m-1)}\right).$$

Equation (13) can now be readily verified recalling the condition $m/\sqrt{n} \to \infty$. ■

We are now equipped to complete the proof of the theorem for the case $d = 1$. We will, in fact, prove a slightly stronger version of the theorem with constants for the lower capacity which are larger than those given in Corollaries 3.2 and 3.3.

*Proof of Theorem* 3.1 ($d = 1$). From Eq. (7) we have that

$$Y_n^\nu = \sum_{j=1}^n u_i^\alpha u_i^\nu u_j^\alpha u_j^\nu = 1 + \sum_{j\neq i} u_i^\alpha u_i^\nu u_j^\alpha u_j^\nu.$$

Hence $\mu_n = \mathbf{E}\{Y_n^\nu\} = 1$, so that the requisite margin of operation in the theorem is $\mathfrak{B} = m\mu_n = m$. It is easy to verify that a choice of $m$ as in Eq. (10) with $d = 1$ satisfies the conditions of Lemma 4.2. Hence

[5] The critical fact here is that each r.v. $Z_j^\nu$ has a distinct multiplicative term $u_j^\nu$ which occurs solely in the expression for $Z_j^\nu$.

$$\mathbf{P}\{\mathscr{E}_n\} = \mathbf{P}\left\{\bigcup_{i=1}^{n} \bigcup_{\alpha=1}^{m} \mathscr{E}_n^{i,\alpha}\right\} \leq \sum_{i=1}^{n} \sum_{\alpha=1}^{m} \mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} \lesssim nm \exp\left\{-\left(\frac{n}{2m}\right)\right\}. \quad (14)$$

For a choice of

$$m = \frac{n}{4 \log n}\left[1 + \frac{\log \log n + \log 4\varepsilon}{2 \log n} - O\left(\frac{\log \log n}{(\log n)^2}\right)\right] \quad (15)$$

in Eq. (14) we have that $\mathbf{P}\{\mathscr{E}_n\} \lesssim \varepsilon$ as $n \to \infty$. As the probability that each of the memories is $m$-stable is exactly $1 - \mathbf{P}\{\mathscr{E}_n\}$, this establishes the first part of the theorem (with a slightly better constant for the critical number of memories).

The second part also follows similarly by noting that the probability that a particular memory is not stable is $\lesssim ne^{-n/2m}$ by the union bound, and for a choice of $m$ given by

$$m = \frac{n}{2 \log n}\left[1 + \frac{\log \varepsilon}{\log n} + O\left(\frac{1}{(\log n)^2}\right)\right], \quad (16)$$

this yields an upper bound of $\varepsilon$ for the probability. The result follows as the expected number of memories that are not stable is $m$ times the probability that one memory is not stable. (Again, the estimate for $m$ given in Eq. (16) is slightly sharper than that quoted in the theorem.) ■

The uniformity of the binomial distribution helps us to establish the lower capacity of the algorithm.

COROLLARY 4.3. *For a degree of interaction $d = 1$ and a margin of operation $\mathscr{B} = m$, the sequence*

$$\underline{C}_n = \frac{n}{4 \log n}$$

*is a lower capacity for the outer-product algorithm.*

COROLLARY 4.4. *For a degree of interaction $d = 1$ and a margin of operation $\mathscr{B} = m$, the sequence*

$$\underline{C}_n^w = \frac{n}{2 \log n}$$

*is a weak lower capacity for the outer-product algorithm.*

*Proof of Corollaries* 4.3 *and* 4.4. We will sketch the proof of Corollary 3.2; the proof of Corollary 3.3 is similar. Let $\tau_M$ explicitly denote the probability $\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\}$ that one component of a memory is not stable as a function of the number of memories, $M$. Fix any choice of $\delta > 0$, and consider a number of memories, $M = (1 - \delta)n/4 \log n$. For any $\varepsilon > 0$ chosen arbitrarily small in Theorem 3.1 we can choose $n$ large enough so that $M < m$ with $m$ chosen as in Eq. (15). The result now follows from Lemma 4.2 since the probability that at least one memory component is not retrievable is bounded from above by $nM\tau_M \lesssim nMe^{-n/2M} \lesssim nme^{-n/2m} \lesssim \varepsilon$. ■

*Remarks.* Corollary 4.3 provides an improvement of a factor of $\frac{3}{2}$ over the lower capacity claimed in Corollary 3.2, while Corollary 4.4 provides an improvement of a factor of 2 over the corresponding weak lower capacity claimed in Corollary 3.3. McEliece *et al.* (1987) show that $n/4 \log n$ is also an upper capacity for the outer-product algorithm for the linear interaction case $d = 1$, so that $n/4 \log n$ is, in fact, the capacity of the algorithm. (The constants obtained there for the $o(1)$ terms in Eq. (10) with $d = 1$ are slightly sharper—a coefficient of $\frac{3}{4}$ for the $\log \log n/\log n$ term instead of the coefficient $\frac{1}{2}$ that we obtain in Eq. (15)—but these do not affect the capacity results.) The proof of the main theorem in McEliece *et al.* (1987) also yields the estimate $n/2 \log n$ for the weak capacity.

## 5. Higher-Order Interactions

The above proof of the theorem for $d = 1$ fails, however, when the interaction order $d$ is larger than one: specifically, for $d \geq 3$ and $r > 0$, the r.v. $Y_n^\nu$ has an infinite moment generating function so that $\mathbf{E}\{e^{-rY_n^\nu}\}$ becomes unbounded and the generalized Chebyshev's inequality of Eq. (34) is too weak. (For $r = 0$ the Chebyshev bound is trivial.) To see this, consider $d = 3$, for instance, and $r > 0$; from Eq. (7) we obtain

$$Y_n^\nu = \left(1 + \sum_{j \neq i}^{n} u_i^\alpha u_i^\nu u_j^\alpha u_j^\nu\right)^3.$$

Let $U \sim \mathcal{N}(0, 1)$ be a standard normal r.v. By Fact 4.1 the summands $u_i^\alpha u_i^\nu u_j^\alpha u_j^\nu$, $j \neq i$ are i.i.d., ±1, symmetric r.v.'s so that by the Central Limit Theorem $Y_n^\nu$ converges in distribution to $(1 + \sqrt{n-1}\,U)^3$, and this has an infinite moment generating function. For $d = 2$, Chebyshev's inequality is workable, but the bound is terribly weak. We will hence need the large deviation lemma A.7 to cater to the higher-order cases.

Before proving the theorem for general interaction orders, we first establish some further properties of the random variables $Y_n^\nu$.

Definition 5.1. Let $Y$ be a discrete r.v. taking values in $\{\theta_j\}_{j=-\kappa}^{\kappa}$. We say that:

1. $Y$ is *skew-symmetric* if $\mathbf{P}\{Y = \theta_{-j}\} = \mathbf{P}\{Y = \theta_j\}$ for $j = 1, \ldots, \kappa$.

2. $Y$ is *unimodal* if $\mathbf{P}\{Y = \theta_{-j}\} < \mathbf{P}\{Y = \theta_{-j+1}\}$ and $\mathbf{P}\{Y = \theta_{j-1}\} > \mathbf{P}\{Y = \theta_j\}$ for $j = 2, \ldots, \kappa$.

We note that, in fact, the r.v.'s $Y_n^\nu$ are skew-symmetric and unimodal. Set $\xi_j^\nu = u_j^\alpha u_j^\nu$. For fixed $\alpha \neq \nu$, the r.v.'s $\xi_1^\nu, \ldots, \xi_n^\nu$ are i.i.d., symmetric, ±1 r.v.'s.

For $d$ even, $Y_n^\nu = \xi_i^\nu (\xi_i^\nu + \sum_{j\neq i}\xi_j^\nu)^d$. The r.v. $Y_n^\nu$ takes values in the set

$$\{-n^d, -(n-2)^d, \ldots, (n-2)^d, n^d\}$$

and for $k = 0, 1, \ldots, \lfloor n/2 \rfloor$

$$\mathbf{P}\{Y_n^\nu = -(n-2k)^d\} = \mathbf{P}\{Y_n^\nu = (n-2k)^d\} = \binom{n}{k} 2^{-n}.$$

Hence the r.v.'s $Y_n^\nu$ are symmetric (consequently, also skew-symmetric) and unimodal.

For $d$ odd, $Y_n^\nu = (1 + \sum_{j\neq i}\xi_i^\nu\xi_j^\nu)^d$. The r.v. $Y_n^\nu$ takes values in the set

$$\{-(n-2)^d, -(n-4)^d, \ldots, (n-2)^d, n^d\}$$

and for $k = 0, 1, \ldots, \lfloor (n-1)/2 \rfloor$

$$\mathbf{P}\{Y_n^\nu = -(n-2k-2)^d\} = \mathbf{P}\{Y_n^\nu = (n-2k)^d\} = \binom{n-1}{k} 2^{-(n-1)}.$$

Hence the r.v.'s $Y_n^\nu$ are skew-symmetric and unimodal.

Lemma 5.2. *For each $n$ the r.v.'s $Y_n^\nu$ are i.i.d. and as $n \to \infty$ satisfy*

$$\mathbf{E}(Y_n^\nu) \begin{array}{ll} = 0 & \textit{if } d \textit{ is even} \\ \sim d\lambda_{(d-1)/2}n^{(d-1)/2} & \textit{if } d \textit{ is odd, } d = o(n); \end{array} \tag{17}$$

$$\mathrm{Var}(Y_n^\nu) \sim \lambda_d n^d \qquad \textit{if } d = o(n). \tag{18}$$

*Remarks.* We actually show a little more than is claimed here. In Eq. (20) we show an *exact* expression for $\mu_n$. This is needed to set the margin of operation accurately.

*Proof.* Recall that we had defined $\mu_n \triangleq \mathbf{E}\{Y_n^\nu\}$, and $\lambda_t \triangleq (2t)!/(t)!2^t$ for every nonnegative integer $t$. As before, denoting $\xi_k^\nu = u_k^\alpha u_k^\nu$ for $k = 1, \ldots, n$, and $\nu \neq \alpha$ we can write

$$Y_n^\nu = \xi_i^\nu \left( \sum_{j=1}^{n} \xi_j^\nu \right)^d.$$

The r.v.'s $\xi_k^\beta$, $k = 1, \ldots, n$, $\beta \neq \alpha$ are mutually independent by Lemma 4.1. Furthermore, each r.v. $Y_n^\nu$ is determined by the distinct set of r.v.'s $\xi_1^\nu, \ldots, \xi_n^\nu$ which appear in no other $Y_n^\beta$, $\beta \neq \nu$. Consequently, the r.v.'s $Y_n^\nu$, $(\nu \neq \alpha)$, are i.i.d. for each $n$.

When $d$ is even, following Definition 5.1 we have established that $Y_n^\nu$ is symmetric so that $\mathbf{E}(Y_n^\nu) = 0$. Let us now consider $d$ odd. From Eq. (7) and by reason of the independent choices of the memories $\mathbf{u}^\alpha$ and $\mathbf{u}^\nu$ we have

$$\mathbf{E}(Y_n^\nu) = \sum_{j_1, \ldots, j_d = 1}^{n} \mathbf{E}(u_i^\alpha u_{j_1}^\alpha \cdots u_{j_d}^\alpha)\mathbf{E}(u_i^\nu u_{j_1}^\nu \cdots u_{j_d}^\nu). \tag{19}$$

We now use the elementary fact that if $x \in \mathbf{B}$ then

$$x^k = \begin{cases} x & \text{if } k \text{ is odd} \\ 1 & \text{if } k \text{ is even,} \end{cases}$$

together with the independence of the components $u_j^\nu$. Each expectation in the sum in Eq. (19) is over a product of an even number, $d + 1$, of $\pm 1$ r.v.'s corresponding to the fixed index $i$ and to each assignment of values to $j_1, \ldots, j_d$. The expectation will have value 1 iff an odd number of indices $j_k$ take the value $i$, and for every index value $h \neq i$ an even number (possibly zero) of indices $j_k$ take the value $h$; otherwise the expectation has value 0.

Let $N_q$ be the number of ways $j_1, \ldots, j_d$ can be chosen from $[n]$ such that precisely $q$ of the $j_k$ are equal to $i$ with each distinct value assigned to the remaining $d - q$ indices occurring an even number of times. We hence have

$$\mathbf{E}(Y_n^\nu) = \sum_{q \text{ odd}} N_q = \sum_{r=1}^{(d+1)/2} N_{2r-1}.$$

Now $2r - 1$ indices from $j_1, \ldots, j_d$ can be chosen equal to $i$ in $\binom{d}{2r-1}$ ways. We must enumerate the number of ways, $N'_{2r-1}$, that values $j \neq i$ can be assigned to the remaining $d - 2r + 1$ indices $j_k$ such that each index occurs an even number of times.

For $k = 1, \ldots, (d - 2r + 1)/2$ let $\mathbf{s} = (s_1, \ldots, s_k)$ be a vector such that $1 \leq s_k \leq s_{k-1} \leq \ldots \leq s_1 \leq (d - 2r + 1)/2$ and $\sum_{j=1}^{k} s_j = (d - 2r + 1)/2$. Let $S_1, \ldots, S_R$ partition $\{s_1, \ldots, s_k\}$ in such a way that each $S_l$ is a maximal collection of $s_j$'s that are equal, and let $\gamma_l = |S_l|$. Define the *redundancy factor*

$$f(\mathbf{s}) = \prod_{l=1}^{R} \gamma_l!.$$

We claim that

$$N'_{2r-1} = \sum_{k=1}^{(d-2r+1)/2} \sum_{\mathbf{s}} \left( \frac{(d-2r+1)!}{(2s_1)! \ldots (2s_k)! f(\mathbf{s})} \right) (n-1)(n-2) \ldots (n-k).$$

In fact, the inner sum over $\mathbf{s}$ enumerates the number of ways $k$ distinct values $j \neq i$ can be assigned to the $n - 2r + 1$ indices $j_k$ with each index occurring an even number, $2s_l$, of times. The redundancy factor, $f(\mathbf{s})$, is required to compensate for overcounting when some of the $s_l$'s are equal. (For instance, $f(\mathbf{s}) = k!$ if $s_1 = \cdots = s_k$, while $f(\mathbf{s}) = 1$ if each $s_l$ is distinct.) Thus (with the convention that $\Sigma_a^b(\cdot) = 1$ if $b < a$), we have

$$\begin{aligned}
\mu_n = \mathbf{E}(Y_n^\nu) &= \sum_{r=1}^{(d+1)/2} N_{2r-1} \\
&= \sum_{r=1}^{(d+1)/2} \binom{d}{2r-1} N'_{2r-1} \\
&= \sum_{r=1}^{(d+1)/2} \binom{d}{2r-1} \sum_{k=1}^{(d-2r+1)/2} \sum_{\mathbf{s}} \frac{(d-2r+1)!}{(2s_1)! \ldots (2s_k)! f(\mathbf{s})} \\
&\quad (n-1) \cdots (n-k) \qquad (20) \\
&= \frac{d!}{[(d-1)/2]! 2^{(d-1)/2}} n^{(d-1)/2} + O(n^{(d-3)/2}), \quad \text{if } d = o(n).^6 \qquad (21)
\end{aligned}$$

Now $(Y_n^\nu)^2 = (\Sigma_{j=1}^n u_j^\alpha u_j^\nu)^{2d}$. A similar argument to that above gives

$$\mathbf{E}\{(Y_n^\nu)^2\} = \frac{(2d)!}{(d)! 2^d} n^d + O(n^{d-1}), \qquad \text{if } d = o(n).^7 \qquad (22)$$

Equations (21) and (22) together complete the proof of the lemma. ■

[6] We can verify this by a standard CLT argument. Let $U \sim \mathcal{N}(0, 1)$ be a standard Gaussian r.v. For $d$ odd, as we saw from the earlier representation, $Y_n^\nu$ converges to $(1 + \sqrt{n-1}\,U)^d$ in distribution by the CLT. Using $\mathbf{E}U^k = 0$ if $k$ is odd and $\mathbf{E}U^k = k!/(k/2)!2^{k/2}$ if $k$ is even, the leading term in the binomial expansion of $\mathbf{E}(1 + \sqrt{n-1}\,U)^d$ yields the result.

We do not directly use this argument, however, as the *exact* representation of the mean is

*Remarks.* The previous result establishes the need for a margin of $\mathfrak{B} = m\mu_n$ in the evolution rule (3). For $d$ even, of course, the margin is precisely zero as the r.v.'s $Y_n^\nu$ are symmetric and have zero mean. For $d$ odd, however, the mean of the noise term in Eq. (6) will be of the order of $mn^{(d-1)/2}$. If $mn^{-d/2} \to \infty$ then this dominates the signal term, $n^d$, in Eq. (6). Hence, almost all states (not just the memories) are fixed points unde, an evolution rule with zero margin. Removing the bias due to this mean results in the evolution rule of Eq. (3) with a choice of margin $m\mu_n$. Clearly, we can expect the memories to be $m\mu_n$-stable because there is still a strong bias of the order of $n^d$ due to the signal term; most randomly chosen states, however, will not be $m\mu_n$-stable. The usage of a suitable margin hence ensures performance as a viable associative memory.

Note that for $d = 1$, however, we can dispense with the margin of $m$ as for $m = o(n)$ the signal term $n$ dominates the mean noise term $m$. Hence, for the linear case we could adopt any choice of margin $0 \leq \mathfrak{B} \leq m$, and obtain adequate performance with the same capacity (McEliece *et al.*, 1987).

The following main lemma uses the large deviation result of Lemma A.7 to estimate the probability that a single component of any given memory is not $\mathfrak{B}$-stable.

LEMMA 5.3. *For any interaction order $d \geq 1$, margin $\mathfrak{B} = m\mu_n$, and any choice of parameter $D > d$ if we choose $m$ such that $mn^{-d(D-1)/D} \to \infty$ and $m = O\,(n^d/\log n)$, then for every $\varpi > 0$*

$$\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} \leq m \exp\left\{-\frac{(1-\varpi)n^d}{2\lambda_d\, m}\right\}, \qquad \textit{as } n \to \infty. \tag{23}$$

*Proof.* Lemma 4.2 gives the result for $d = 1$. We hence consider the case $d > 1$. Define the normalized sequence of r.v.'s $T_n^\nu$ by

$$T_n^\nu = \lambda_d^{-1/2} n^{-d/2}(Y_n^\nu - \mu_n). \tag{24}$$

By Lemma 5.2 $\mathbf{E}(T_n^\nu) = 0$ and Var $(T_n^\nu) \to 1$ as $n \to \infty$. Set $M = m - 1$ for notational simplicity. Clearly $M \to \infty$ and $M = o(n^D)$. Using Lemma 5.2 with Eqs. (8) and (24) we have

$$\begin{aligned}\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} &= \mathbf{P}\{X_n \leq m\mu_n\}\\ &= \mathbf{P}\left\{\sum_{\nu\neq\alpha} T_n^\nu \leq -\frac{n^{d/2}}{\sqrt{\lambda_d}} + \tau_n\right\},\end{aligned}$$

important in determining the probability that a row-sum violation occurs. If we use only the highest-order term for the mean, the succeeding terms that were ignored will dominate the inequality as $n^d = o(mn^{(d-3)/2})$.

[7] Again, $(Y_n^\nu)^2$ converges to $(\sqrt{n}U)^{2d}$ in distribution, and $\mathbf{E}(\sqrt{n}U)^{2d} = (2d)!n^d/(d)!2^d$.

where $\tau_n = O(n^{-1/2})$. Now set

$$\gamma_n = \frac{n^{d/2}}{\sqrt{\lambda_d}} - \tau_n. \tag{25}$$

By the bounds on $m$ we have $\gamma_n = \Omega(\sqrt{M \log M})$ and $\gamma_n = o(M)$. If the conditions 1–4 of Lemma A.7 are met,[8] we would then have that as $n \to \infty$

$$\begin{aligned} \mathbf{P}\{\mathscr{E}_n^{i,\alpha}\} &\sim \mathbf{P}\left\{\sum_{\nu \neq \alpha} T_n^\nu \leq -\gamma_n\right\} \\ &\leq M \exp\left(-\frac{(1-\varpi)\gamma_n^2}{2M}\right) \\ &\sim m \exp\left(-\frac{(1-\varpi)n^d}{2\lambda_d m}\right). \end{aligned}$$

By construction, and by Lemma 5.2, the r.v.'s $T_n^\nu$ satisfy conditions 1 and 2 of Lemma A.7. Comparing Eqs. (25) and (38), we hence must show that conditions 3 and 4 are also met for the choice of parameter $D > d \geq 2$ in order to complete the proof. We show the result when the interaction order is odd, so that $D > d \geq 3$. The proof is similar when $d$ is even.

With a notation similar to that earlier, we have

$$\begin{aligned} |T_n^\nu| &= \lambda_d^{-1/2} n^{-d/2} \left| \left(1 + \sum_{j \neq i} \xi_i^\nu \xi_j^\nu\right)^d - \mu_n \right| \\ &\leq \lambda_d^{-1/2} n^{-d/2} |1 + U_{n-1}|^d + \lambda_d^{-1/2} n^{-d/2} \mu_n. \end{aligned} \tag{26}$$

By Lemma 5.2 we have that $\mu_n = O(n^{(d-1)/2})$. Further, it is easy to see that $|1 + U_{n-1}|^d \leq 1 + 2^d |U_{n-1}|^d$. Using the simple inequality $(A + B)^x \leq 2^x(A^x + B^x)$ valid for positive $A$, $B$, and $x$, it hence follows from Lemma A.6 that

$$\begin{aligned} &\limsup_{n \to \infty} \mathbf{E}\{\exp(x|T_n^\nu|^{2/D})\} \\ &\quad \leq \limsup_{n \to \infty} \mathbf{E}[\exp\{x 2^{2(d+1)/D} \lambda_d^{-1/D} |U_{n-1}|^{2d/D} n^{-d/D} + O(n^{-1/D})\}] \\ &\quad < \infty \end{aligned}$$

whenever we choose $x$ such that

[8] Note that $\gamma_n = o(M^{D/2(D-1)})$. Hence, the bound Eq. (38) in Lemma A.7 will hold trivially for any positive choice of $y < x$ once condition 3 is established.

$$x < 2^{-2(d+1)/D}\lambda_d^{1/D}\left(\frac{2d}{D}\right)^{-d/D}$$

This establishes Eq. (36). Now, noting that $T_n^\nu$ is a discrete r.v. which takes on only a finite set of real values with nonzero probability, we have for any choice of $K_n = \Omega\{(\log n)^{D/2}\}$ that

$$\begin{aligned}\int_{|t|>K_n} t^2\, dF_n^\nu(t) &\leq \sum_{|t|>A(\log n)^{D/2}} t^2\, \mathbf{P}\{T_n^\nu = t\}\\ &\leq \sum_{|t|>A(\log n)^{D/2}} t^2\, \mathbf{P}\{U_{n-1} = (t\lambda_d^{1/2}n^{d/2} + \mu_n)^{1/d} - 1\}.\end{aligned}$$

In the above $A > 0$ is a real constant, and the summation is over the finite set of real values that $T_n^\nu$ can assume in the range $|t| > A(\log n)^{D/2}$. Now, from Eq. (26) we have

$$|T_n^\nu| \leq \lambda_d^{-1/2}n^{d/2} + O(n^{-1/2}) \lesssim 2\lambda_d^{-1/2}n^{d/2}$$

as $|1 + U_{n-1}| \leq n$. Further, $U_{n-1}$ is a symmetric (binomially distributed), unimodal r.v. Hence, we can find $B = A^{1/d} \pm O\{n^{-1/2}(\log n)^{-D/2d}\}$ such that

$$\begin{aligned}&\mathbf{P}\{U_{n-1} = B\lambda_d^{1/2d}n^{1/2}(\log n)^{D/2d}\}\\ &\quad\gtrsim \max_{|t|>A(\log n)^{D/2}} \mathbf{P}\{U_{n-1} = (t\lambda_d^{1/2}n^{d/2} + \mu_n)^{1/d} - 1\}.\end{aligned}$$

It follows that

$$\begin{aligned}\int_{|t|>K_n} t^2\, dF_n^\nu(t) &\lesssim 4\lambda_d^{-1}n^d\, \mathbf{P}\{U_{n-1} = B\lambda_d^{1/2d}n^{1/2}(\log n)^{D/2d}\}\\ &\lesssim \frac{\sqrt{32}}{\sqrt{\pi}\,\lambda_d}\, n^{d-1/2} \exp\left(-\frac{B^2\lambda_d^{1/d}(\log n)^{D/d}}{2}\right), \qquad (27)\end{aligned}$$

by an application of Corollary A.5. But by the choice of $D$ we have that $D/d > 1$, so that the right-hand side of Eq. (27) is $o(n^{-D/2})$, and this concludes the proof. ■

*Proof of Theorem 3.1* ($d > 1$). An application of Lemma 5.3 together with the union bound finishes the proof of the theorem. For any fixed $\varpi > 0$

$$\begin{aligned}\mathbf{P}\{\mathscr{E}_n\} &= \mathbf{P}\left\{\bigcup_{i=1}^{n}\bigcup_{\alpha=1}^{m}\mathscr{E}_n^{i,\alpha}\right\}\\ &\leq nm\mathbf{P}\{\mathscr{E}_n^{i,\alpha}\}\\ &\leq nm^2 \exp\left(-\frac{(1-\varpi)n^d}{2\lambda_d m}\right).\end{aligned}$$

It can now be readily verified by substitution of Eq. (10) that $\mathbf{P}\{\mathcal{E}_n\} \leq \varepsilon$. Part 2 can be verified similarly. ■

A uniformity argument similar to the one used for Corollary 4.3 completes the proof of Corollaries 3.2 and 3.3 when $d > 1$. It appears plausible that, just as in the linear case $d = 1$, the rates of growth in Corollaries 3.2 and 3.3 also apply to upper capacities for higher interaction orders $d > 1$. The dependencies in the random variables, however, become rather more severe when $d > 1$, and, as yet, there are no rigorous proofs in this regard. In particular, the proof techniques used by McEliece *et al.* (1987) in establishing capacities for $d = 1$ cannot be used *in toto* for the higher-order case.

## 6. Zero-Diagonal Networks

As before, let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of memories, whose components are chosen from a sequence of symmetric Bernoulli trials. We now consider zero-diagonal networks with interconnection weights chosen according to prescription (5) for the zero-diagonal outer-product algorithm of degree $d$.

Analogously with the notation of the previous section, for each $n$ define the sequence of doubly indexed random variables $X_n^{i,\alpha}$ with

$$\begin{aligned} X_n^{i,\alpha} &= u_i^\alpha \sum_{I\in\mathcal{I}_d: i\notin I} w_{(i,I)} u_I^\alpha = u_i^\alpha \sum_{\nu=1}^{m} \sum_{I\in\mathcal{I}_d: i\notin I} u_i^\nu u_I^\nu u_I^\alpha \\ &= \binom{n-1}{d} + \sum_{\nu\neq\alpha}\left(u_i^\alpha u_i^\nu \sum_{I\in\mathcal{I}_d, i\notin I} u_I^\alpha u_I^\nu\right). \end{aligned} \tag{28}$$

Again suppressing the $i$, $\alpha$ dependence and setting

$$Y_n^\nu = u_i^\alpha u_i^\nu \sum_{I\in\mathcal{I}_d: i\notin I} u_I^\alpha u_I^\nu, \tag{29}$$

we get

$$X_n = \binom{n-1}{d} + \sum_{\nu\neq\alpha} Y_n^\nu.$$

For a margin of operation zero, the evolution will fail to retrieve the $i$th component of the $\alpha$th memory, $u_i^\alpha$, if $X_n^{i,\alpha} \leq 0$. As before, let $\mathcal{E}_n^{i,\alpha}$ denote the event $\{X_n^{i,\alpha} < 0\}$, and let $\mathcal{E}_n = \bigcup_{i=1}^n \bigcup_{\alpha=1}^m \mathcal{E}_n^{i,\alpha}$ be the event that one or more memory components is not stable.

Clearly $\mathbf{E}(Y_n^v) = 0$ and it is also easy to verify that $\mathrm{Var}(Y_n^v) = \binom{n-1}{d}$. The following result then follows analogously to Theorem 3.1 with virtually the same proof. (The situation is, in fact, simpler in the zero-diagonal case as the symmetric nature of the r.v.'s $Y_n^v$ ensures that Lemma A.7 readily applies in this instance.)

THEOREM 6.1. *Consider a zero-diagonal higher-order neural network of degree $d$ with weights chosen according to the outer-product algorithm of Eq.* (5) *and with a choice of margin* $\mathcal{B} = 0$ *in the evolution rule* (4). *For any fixed* $\varepsilon > 0$ *and* $\varpi > 0$:

1. *If, as* $n \to \infty$, *we choose* $m$ *such that*

$$m = \frac{(1-\varpi)n^d}{2(2d+1)(d)!\log n} \left[1 + \frac{2\log\log n + 2\log 2(2d+1)(d)!\sqrt{\varepsilon}}{(2d+1)\log n} - O\left(\frac{\log\log n}{(\log n)^2}\right)\right],$$

*then the probability that each of the memories is a fixed point is* $\gtrsim 1 - \varepsilon$;

2. *If, as* $n \to \infty$, *we choose* $m$ *such that*

$$m = \frac{(1-\varpi)n^d}{2(d+1)!\log n}\left[1 + \frac{\log\log n + \log 2\varepsilon(d+1)!}{(d+1)\log n} - O\left(\frac{\log\log n}{(\log n)^2}\right)\right],$$

*then the expected number of memories that are fixed points is* $\gtrsim m(1 - \varepsilon)$.

COROLLARY 6.2 *For a given degree of interaction* $d \geq 1$ *and margin* $\mathcal{B} = 0$ *the sequence*

$$\underline{C_n} = \frac{n^d}{2(2d+1)(d)!\log n}$$

*is a lower capacity for the zero-diagonal outer-product algorithm.*

COROLLARY 6.3 *For a given degree of interaction* $d \geq 1$ *and margin* $\mathcal{B} = 0$ *the sequence*

$$\underline{C_n^w} = \frac{n^d}{2(d+1)!\log n}$$

*is a weak lower capacity for the zero-diagonal outer-product algorithm.*

*Remarks.* Again, for $d = 1$ we can sharpen the results somewhat using the same techniques as in Section 4. The result is a capacity and weak capacity exactly given by Corollaries 4.3 and 4.4, respectively; i.e., for first-order interactions the presence or absence of the diagonal terms makes no difference to the capacity. This, as seen above, is not true for $d > 1$, however.

Note the somewhat surprising result that the zero-diagonal capacities are larger than their nonzero diagonal counterparts even though the signal term in the zero-diagonal case is somewhat lower than for the nonzero diagonal case. In fact, the ratio of the zero-diagonal capacity to the capacity when the diagonal terms are not set to zero is the rather substantial factor of $\lambda_d/(d)!$. For large interaction orders, therefore, the outer-product algorithm with diagonal terms set to zero picks up a factor of $2^d/\sqrt{\pi d}$ in capacity. This effect can be traced to the additional noise variance caused by the diagonal terms when they are present (Eq. (18)); the growth in the noise due to the nonzero diagonal terms exceeds the corresponding growth in the signal term. In particular, adding the diagonal terms causes an increase in the signal term from $\binom{n-1}{d}$ to $n^d$; however, the corresponding growth in the noise variance is somewhat larger, from $(m-1)\binom{n-1}{d}$ to $(m-1)\lambda_d n^d$.

## 7. Attractors and Dynamic Capacity

The capacity results derived above are readily extendable when the memories are required not just to be stable, but to be *attractors*. Let $\mathbf{u}^1, \ldots, \mathbf{u}^m \in \mathbb{B}^n$ be an $m$-set of randomly chosen memories and consider an outer-product network of degree $d$. Fix $0 \leq \rho < \frac{1}{2}$, and let $\mathbf{u}[\alpha]$ be a randomly chosen state within the Hamming ball of radius $\rho n$ surrounding an arbitrarily chosen memory $\mathbf{u}^\alpha$. We will require that system dynamics map $\mathbf{u}[\alpha]$ into the memory $\mathbf{u}^\alpha$ with high probability.

As before, we define the sequence of doubly indexed random variables $X_n^{i,\alpha}$ by

$$X_n^{i,\alpha} = u_i^\alpha \sum_{J \in \mathcal{J}_d} w_{(i,J)} u_J[\alpha] = u_i^\alpha \sum_{\nu=1}^{m} \sum_{J \in \mathcal{J}_d} u_i^\nu u_J^\nu u_J[\alpha].$$

Setting

$$S_n^{i,\alpha} = \left(\sum_{j=1}^{n} u_j[\alpha] u_j^\alpha\right)^d$$

and

$$Y_n^{i,\alpha,\nu} = u_i^\alpha u_i^\nu \left(\sum_{j=1}^{n} u_j[\alpha] u_j^\nu\right)^d,$$

we get

$$X_n^{i,\alpha} = S_n^{i,\alpha} + \sum_{\nu \neq \alpha} Y_n^{i,\alpha,\nu}.$$

Note that by the sphere hardening effect the random state $\mathbf{u}[\alpha]$ will lie on the surface of the Hamming ball of radius $\rho n$ surrounding the memory $\mathbf{u}^\alpha$ with high probability for large $n$. We hence have that the estimate $S_n^{i,\alpha} \sim n^d(1 - 2\rho)^d$ for the signal term holds with probability approaching one as $n \to \infty$. The signal term is reduced from its maximum value of $n^d$ because of the slight initial mismatch (essentially $\rho n$ components) between the probe vector $\mathbf{u}[\alpha]$ and the memory $\mathbf{u}^\alpha$. Now, for $d$ even the noise terms $Y_n^{i,\alpha,\nu}$ are symmetric r.v.'s. For $d$ odd we can write

$$\begin{aligned} Y_n^{i,\alpha,\nu} &= \left(u_i^\alpha u_i[\alpha] + \sum_{j \neq i} u_i^\alpha u_i^\nu u_j^\nu u_j[\alpha]\right)^d \\ &= \left(A^{i,\alpha} + \sum_{j \neq i} \xi^{j,\nu}\right)^d, \end{aligned}$$

where the r.v. $A^{i,\alpha} = u_i^\alpha u_i[\alpha]$ has mean approaching $1 - 2\rho$ for large $n$, and is independent of the symmetric, i.i.d., $\pm 1$ r.v.'s $\xi^{j,\nu} = u_i^\alpha u_i^\nu u_j^\nu u_j[\alpha]$ for $j \neq i$.

The evolution rule (3) will fail to retrieve the $i$th component of the $\alpha$th memory, $u_i^\alpha$, if $X_n^{i,\alpha} \leq \mathcal{B}$. As before, let $\mathcal{E}_n^{i,\alpha}$ denote the event $\{X_n^{i,\alpha} \leq \mathcal{B}\}$, and let $\mathcal{E}_n = \bigcup_{i=1}^n \bigcup_{\alpha=1}^m \mathcal{E}_n^{i,\alpha}$ be the event that one or more memory components are not retrieved (i.e., is not $\mathcal{B}$-stable). We are interested in the probability, $1 - \mathbf{P}\{\mathcal{E}_n\}$, that each of the fundamental memories attracts a randomly chosen state in the Hamming ball of radius $\rho n$ surrounding each memory in *one synchronous step*, as well as in the allied weak sense result.

Let $\lambda_d$ be as defined in Eq. (9), and let $\mu_n = \mathbf{E}\{Y_n^{i,\alpha,\nu}\}$. We see that the arguments used in the proof of Theorem 3.1 continue to work here, albeit with a slight reduction in the value of the signal term.

THEOREM 7.1. *Fix $\epsilon > 0$, $\varpi > 0$, and choose a margin $\mathcal{B} = m\mu_n$ in the evolution rule* (3) *for the outer-product algorithm of degree $d$. For any fixed radius of attraction, $\rho > 0$:*

1. *If, as $n \to \infty$, we choose $m$ such that*

$$\begin{aligned} m = \frac{(1 - \varpi)(1 - 2\rho)^{2d} n^d}{2(2d + 1)\lambda_d \log n} \Bigg[ 1 &+ \frac{2 \log\log n + 2 \log 2(d + 1)\lambda_d \sqrt{\varepsilon}}{(2d + 1) \log n} \\ &- O\left(\frac{\log\log n}{\log n)^2}\right)\Bigg], \end{aligned} \tag{30}$$

*then the probability that for each fundamental memory a randomly chosen state in the Hamming ball of radius $\rho n$ surrounding the memory is mapped into the memory in one synchronous step is $\gtrsim 1 - \varepsilon$;*

2. *If, as $n \to \infty$, we choose m such that*

$$m = \frac{(1-\varpi)(1-2\rho)^{2d}n^d}{2(2d+1)\lambda_d \log n}\left[1 + \frac{\log\log n + \log 2\varepsilon(d+1)\lambda_d}{\log n} - O\left(\frac{\log\log n}{(\log n)^2}\right)\right], \qquad (31)$$

*then the expected number of memories which attract a randomly chosen state in the Hamming ball of radius $\rho n$ surrounding the memory in one synchronous step is $\gtrsim m(1 - \varepsilon)$.*

Corollary 7.2. *For a given degree of interaction $d \geq 1$ and a fixed choice of $0 \leq \rho < \frac{1}{2}$ the sequence*

$$\underline{C_n}(\rho) = \left(\frac{(d)!(1-2\rho)^{2d}2^{d-1}}{(2d+1)!}\right)\frac{n^d}{\log n}$$

*is a lower $\rho$-attractor capacity in one-step synchronous operation for the outer-product algorithm of degree d.*

Corollary 7.3. *For a given degree of interaction $d \geq 1$ and a fixed choice of $0 \leq \rho < \frac{1}{2}$ the sequence*

$$\underline{C_n^w}(\rho) = \left(\frac{(d)!(1-2\rho)^{2d}2^{d-1}}{(2d)!(d+1)}\right)\frac{n^d}{\log n}$$

*is a weak lower $\rho$-attractor capacity in one-step synchronous operation for the outer-product algorithm of degree d.*

The fixed point capacity results of Corollaries 3.2 and 3.3 are hence weakened by just the multiplicative factor $(1 - 2\rho)^{2d}$ if we require, in addition, that there be attraction over a Hamming ball of radius $\rho n$ in *one* synchronous step. Analogous results hold for the zero-diagonal case. Specifically

Thforem 7.4. *Fix $\varepsilon > 0$, $\varpi > 0$, and choose a margin of zero in the evolution rule* (4) *for the zero-diagonal outer-product algorithm of degree d.*

1. *If, as $n \to \infty$, we choose m such that*

$$m = \frac{(1-\varpi)(1-2\rho)^{2d}n^d}{2(2d+1)(d)!\log n}\left[1 + \frac{2\log\log n + 2\log 2(2d+1)(d)!\sqrt{\varepsilon}}{(2d+1)\log n} - O\left(\frac{\log\log n}{(\log n)^2}\right)\right],$$

*then the probability that for each fundamental memory a randomly chosen state in the Hamming ball of radius ρn surrounding the memory is mapped into the memory in one synchronous step is ≳1 − ε;*

2. *If, as n → ∞, we choose m such that*

$$m = \frac{(1-\varpi)(1-2\rho)^{2d}n^d}{2(d+1)!\log n}\left[1 + \frac{\log\log n + \log 2\varepsilon(d+1)!}{(d+1)\log n} - O\left(\frac{\log\log n}{(\log n)^2}\right)\right],$$

*then the expected number of memories which attract a randomly chosen state in the Hamming ball of radius ρn surrounding the memory in one synchronous step is ≳m(1 − ε).*

Corollary 7.5. *For a given degree of interaction d ≥ 1 and a fixed choice of* $0 \le \rho < \frac{1}{2}$ *the sequence*

$$\underline{C}_n(\rho) = \left(\frac{(1-2\rho)^{2d}}{2(2d+1)(d)!}\right)\frac{n^d}{\log n}$$

*is a lower ρ-attractor capacity in one-step synchronous operation for the zero-diagonal outer-product algorithm of degree d.*

Corollary 7.6. *For a given degree of interaction d ≥ 1 and a fixed choice of* $0 \le \rho < \frac{1}{2}$ *the sequence*

$$\underline{C}_n^w(\rho) = \left(\frac{(1-2\rho)^{2d}}{2(d+1)!}\right)\frac{n^d}{\log n}$$

*is a weak lower ρ-attractor capacity in one-step synchronous operation for the zero-diagonal outer-product algorithm of degree d.*

The following nonrigorous argument (as in McEliece *et al.* (1987)) seems to indicate that if we allow nondirect convergence to the memories then we can, in fact, remove the factors of $(1-2\rho)^{2d}$ by which the capacity is reduced if we insist on direct convergence. Consider the nonzero diagonal situation again, for instance. Fix a small $\rho^* > 0$. If the number of fundamental memories is chosen to be

$$m = \frac{(1 - \varpi)(1 - 2\rho^*)^{2d} n^d}{(2d + 1)\lambda_d \log n},$$

then by Theorem 7.1 each fundamental memory directly attracts over a Hamming sphere of radius $\rho^* n$. Let $\rho < \frac{1}{2}$ be the desired (fractional) radius of attraction. Extending Lemma 5.3 for the direct convergence case (i.e., replacing $n$ in Eq. (23) by $n_\rho = (1 - 2\rho)n$) we obtain that the asymptotic probability, $\tau$, that a single component of a given memory is incorrectly labeled is bounded by

$$\tau = O\left(\frac{n^{d-(2d+1)(1-2\rho)^{2d}/(1-2\rho^*)^{2d}}}{\log n}\right).$$

It is easily seen that $\tau \to 0$ as $n \to \infty$ if the desired fractional radius of attraction, $\rho$, satisfies

$$\rho \le \frac{1}{2}\left(1 - \left(\frac{d}{2d + 1}\right)^{1/2d}\right). \tag{32}$$

In the multiple step synchronous case the probe vector has essentially $\rho n$ components incorrectly specified. The first synchronous state transition will map the probe vector to a state where essentially $n\tau$ components are wrong, with high probability. For any fixed $\rho^*$, however small, we can choose $n$ large enough so that the probability of component misclassification, $\tau$, becomes smaller still. Thus, for large enough $n$, the probe vector will be mapped within the confines of a Hamming sphere of (small) radius $\rho^*$ surrounding the memory. But by Theorem 7.1 the next state transition will converge directly to the fundamental memory with very high probability. This (nonrigorous) argument indicates that for every fixed (small) $\rho^*$, and every choice of attraction radius $\rho$ satisfying Eq. (32), we can find $n$ large enough that any randomly chosen state in the Hamming ball of radius $\rho n$ surrounding the memories will converge to the corresponding fundamental memories within two synchronous transitions. Now, keeping fixed, if we allow $\rho^*$ to approach zero it appears that the factor $(1 - 2\rho)^d$ can be dropped from the capacity expression for large enough $n$.[9]

## 8. Conclusions

We have established that the outer-product algorithm of degree $d$ (and its zero-diagonal variant) can store at least of the order of $n^d/\log n$ memo-

[9] The difficulty in making this rigorous is that we must estimate the probability of the conjunction of two successive events: one mapping a ball of radius $\rho n$ into a smaller ball of radius $\rho^* n$, and the other mapping the ball of radius $\rho^* n$ into the memory.

ries. Open questions include the determination of tight upper capacities, rates of convergence, and capacities when more than one synchronous step is allowed in the dynamics, and extending and tightening Newman's (1988) description of the energy landscape to obtain estimates of the number of memories that can be stored when a certain error-tolerance is permitted in recall. The key issue here is whether, as in the case $d = 1$, we can gain a factor of $\log n$ in capacity if errors are allowed in the retrieval of the memories.

## Appendix A: Large Deviations

The technical lemmas of this section principally deal with large deviations of a sum of random variables from its mean. Lemma A.1 is a generalization of the Chebyshev inequality. Lemma A.3 is a standard approximation of the tail of the normal distribution function. Lemma A.4 is the classical large deviation Central Limit Theorem for sums of (0,1) random variables. Lemma A.6 outlines an inequality for generating functions in the spirit of Khintchine's inequality. Finally, Lemma A.7 is a large deviation result which applies to deviations much larger than those handled by the Central Limit Theorem. The lemma is motivated by a large deviation result due to Newman for symmetric random variables. Lemmas A.1 to A.4 are standard results and we quote them without proof (cf. Feller (1968), for instance).

Lemma A.1 (Generalized Chebyshev Inequality). *Let $\psi_+$ be a monotonically increasing positive function on the real line. Let $Y$ be any random variable and suppose that $\mathbf{E}(\psi_+(Y))$ exists. Then for any $u$*

$$\mathbf{P}\{Y \geq u\} \leq \frac{\mathbf{E}(\psi_+(Y))}{\psi_+(u)}.$$

*Similarly, if $\psi_-$ is any monotonically decreasing positive function with $\mathbf{E}(\psi_-(Y)) < \infty$, then*

$$\mathbf{P}\{Y \leq -u\} \leq \frac{\mathbf{E}(\psi_-(Y))}{\psi_-(-u)}.$$

Corollary A.2. *For any random variable $Y$ and any $u \geq 0$*

$$\mathbf{P}\{Y \geq u\} \leq \inf_{r \geq 0} e^{-ru}\, \mathbf{E}(e^{rY}), \tag{33}$$

$$\mathbf{P}\{Y \leq -u\} \leq \inf_{)} e^{-ru}\, \mathbf{E}(e^{-rY}). \tag{34}$$

As usual, in the following we denote by $\varphi$ the normal density function

$$\varphi(x) = (2\pi)^{-1/2}e^{-x^2/2},$$

and by $\Phi$ the normal distribution function

$$\Phi(x) = \int_{-\infty}^{x} \varphi(y)\, dy.$$

LEMMA A.3. *$\Phi(-x) \sim \varphi(x)/x$ as $x \to \infty$.*

LEMMA A.4. *Let $\{\zeta_j\}$ be a sequence of i.i.d. random variables taking on values 0 and 1, each with probability $\frac{1}{2}$. For each $n$ let $S_n = \sum_{j=1}^{n} \zeta_j$, and let $a_k = \mathbf{P}\{S_n = \lceil n/2 \rceil + k\}$, and put*

$$h = \frac{2}{\sqrt{n}}.$$

*If $n \to \infty$ and $k$ is constrained to an interval $k < K_n$ where $K_n = o(n^{2/3})$ then there are constants $A$ and $B$ such that*

$$\left|\frac{a_k}{h\varphi(hk)} - 1\right| < \frac{A}{n} + \frac{BK_n^3}{n^2} \tag{35}$$

*uniformly in $k$; and, in fact, $h\varphi(hk)$ is an asymptotic upperbound for $a_k$ for any $k$. Further,*

$$\mathbf{P}\{S_n \geq \lceil n/2 \rceil + K_n\} = \sum_{k=K_n}^{\lfloor n/2 \rfloor} a_k \to \Phi\left(-\frac{2K_n}{\sqrt{n}}\right).$$

COROLLARY A.5. *Let $R_n$ denote the sum of $n$ i.i.d. random variables taking on values $-1$ and 1 only, each with probability $\frac{1}{2}$. Let $\hat{a}_k = \mathbf{P}\{R_n = k\}$. If, as $n \to \infty$, $k$ is constrained to an interval $k < K_n$ where $K_n = o(n^{2/3})$ then*

$$\hat{a}_k \begin{cases} = 0 & \text{if } n - k \text{ is odd} \\ \sim \dfrac{2}{\sqrt{n}}\, \varphi\left(\dfrac{k}{\sqrt{n}}\right) & \text{if } n - k \text{ is even.} \end{cases}$$

LEMMA A.6. *Let $\{\xi_j\}$ be a sequence of i.i.d. random variables taking on values $-1$ and 1, each with probability $\frac{1}{2}$. Let $U_n = \sum_{j=1}^{n} \xi_j$. Then for any choice of positive parameters $\omega \leq 2$ and $t < \omega^{-\omega/2}$ we have*

$$\limsup_{n \to \infty} \mathbf{E}(e^{t|U_n|^{\omega} n^{-\omega/2}}) < \infty.$$

*Remark.* Note that the function is of the form $\exp\{\alpha|U|^{\nu}\}$ so that Khintchine's inequality which requires that the test function be real analytic with all its derivatives being positive at the origin cannot be readily applied.

*Proof.* The basic strategy is to show that the sequence of partial sums corresponding to the Taylor series expansion for the generating function defined above converges uniformly. Accordingly, we first estimate $\mathbf{E}(|U_n|^z n^{-z/2})$ for $z > 0$. Set $\zeta_j = (\xi_j + 1)/2$ and let $S_n = \sum_{j=1}^{n} \zeta_j$. Now $U_n$ is a symmetric random variable and $S_n = (U_n + n)/2$. We have

$$\begin{aligned}\mathbf{E}(|U_n|^z n^{-z/2}) &= 2n^{-z/2} \sum_{k\geq 0} k^z\, \mathbf{P}\{U_n = k\}\\ &= 2n^{-z/2} \sum_{k\geq 0} k^z\, \mathbf{P}\{S_n = (k + n)/2\}\\ &< 2^{z+1} n^{-z/2} \sum_{l=0}^{\lfloor n/2\rfloor} (l + 1)^z\, a_l\end{aligned}$$

where $a_l = \mathbf{P}\{S_n = \lceil n/2\rceil + l\}$. Choosing $\frac{1}{2} < \tau < \frac{2}{3}$ we effect a partition of the above sum into three partial sums:

$$\mathbf{E}(|U_n|^z n^{-z/2}) < \underbrace{\sum_{l=0}^{\lfloor(\log n)/2\rfloor - 1}}_{\Sigma_1} + \underbrace{\sum_{l=\lfloor(\log n)/2\rfloor}^{\lfloor n^{\tau}/2\rfloor}}_{\Sigma_2} + \underbrace{\sum_{l=\lfloor n^{\tau}/2\rfloor+1}^{\lfloor n/2\rfloor}}_{\Sigma_3}.$$

Now

$$\begin{aligned}\Sigma_1 &\leq 2^{z+1} n^{-z/2} \left[\frac{\log n}{2}\right]^z \sum_{l=0}^{\lfloor(\log n)/2\rfloor} a_l\\ &\leq 2n^{-z/2}(\log n)^z,\end{aligned}$$

and using the results of Lemmas A.3 and A.4 we have

$$\begin{aligned}\Sigma_3 &\leq 2^{z+1} n^{-z/2} \left(\frac{n}{2}\right)^z \sum_{l=\lceil n^{\tau}/2\rceil}^{\lfloor n/2\rfloor} a_l\\ &\sim 2n^{z/2}\, \Phi(-n^{\tau-1/2})\\ &\sim \frac{\sqrt{2}}{\sqrt{\pi}}\, n^{z/2-\tau+1/2}\, e^{-n^{(2\tau-1)}/2}.\end{aligned}$$

Further, in the range $(\log n)/2 \leq l \leq n^{\tau}/2$ we have from Lemma A.4 that

$$\begin{aligned}(l+1)^z a_l &\sim \frac{l^z 2}{\sqrt{n}} \varphi\left(\frac{2l}{\sqrt{n}}\right)\left(1 + O\left(\frac{1}{\log n}\right)\right) \\ &\lesssim 2^{1-z} n^{z/2} \left[\int_{2(l-1/2)/\sqrt{n}}^{2(l+1/2)/\sqrt{n}} x^z \varphi(x)\, dx\right],\end{aligned}$$

where we have overestimated the $(1 + o(1))$ term by 2. It hence follows that

$$\begin{aligned}\Sigma_2 &\lesssim 4 \sum_{l=\lfloor(\log n)/2\rfloor}^{\lfloor n^\gamma/2\rfloor} \int_{2(l-1/2)/\sqrt{n}}^{2(l+1/2)/\sqrt{n}} x^z \varphi(x)\, dx \\ &\sim 4 \int_{n^{-1/2}(\log n - 1)}^{n^{-1/2}(n^\gamma + 1)} x^z \varphi(x)\, dx \\ &\sim 4 \int_0^\infty x^z \varphi(x)\, dx \\ &= 2^{z/2+1} \pi^{-1/2}\, \Gamma((z+1)/2).^{10}\end{aligned}$$

As the upper bounds for both $\Sigma_1$ and $\Sigma_3$ approach zero with $n$ it follows that

$$\mathbf{E}(|U_n|^z n^{-z/2}) \lesssim 2^{z/2+1} \pi^{-1/2}\, \Gamma((z+1)/2).$$

Using Stirling's formula[11] we then obtain for large $k$ and fixed $\omega > 0$ that

$$\frac{t^k}{k!} \mathbf{E}(|U_n|^{\omega k} n^{-\omega k/2}) \lesssim 2\pi^{-1/2}\, (t\omega^{\omega/2})^k\, e^{-(\omega/2-1)k}\, k^{(\omega/2-1)k-1/2}.$$

For large $k$, the $k$th term of the partial sum

$$Q_N = \sum_{k=0}^{N} \frac{t^k}{k!} \mathbf{E}(|U_n|^{\omega k} n^{-\omega k/2})$$

hence decreases exponentially provided $\omega \leq 2$ and $t < \omega^{-\omega/2}$. As the sequence of partial sums $Q_N$ converges to $\mathbf{E}(e^{t|U_n|^\omega n^{-\omega/2}})$ uniformly in $N$, it follows that $\mathbf{E}(e^{t|U_n|^\omega n^{-\omega/2}}) < \infty$. ■

LEMMA A.7. *Let $D \geq 2$ be some fixed parameter, and for each $n$ let $\{T_n^\nu\}_{\nu=1}^\infty$ be a sequence of independent random variables* (*with distribution function $F_n^\nu$*) *satisfying*:

1. $\mathbf{E}(T_n^\nu) = 0$;

[10] The *gamma function* is defined for any $y > 0$ by $\Gamma(y) = \int_0^\infty x^{y-1} e^{-x}\, dx$.

[11] For fixed $\omega > 0$ and $k$ large, $k! \sim \sqrt{2\pi}\, e^{-k} k^{k+1/2}$ and $\Gamma(\omega k) \sim \sqrt{2\pi}\, e^{-\omega k} (\omega k)^{\omega k - 1/2}$.

2. $\lim_{n\to\infty} \mathrm{Var}(T_n^\nu) = 1$;
3. *There is a number $x > 0$ such that*

$$\limsup_{n\to\infty} \mathbf{E}\{\exp(x|T_n^\nu|^{2/D})\} < \infty; \tag{36}$$

4. *For any $K_n = \Omega[(\log n)^{D/2}]$*

$$\int_{|t|>K_n} t^2\, dF_n^\nu(t) = o(n^{-D/2}), \qquad (n \to \infty). \tag{37}$$

*Let $M_n$ be a polynomially increasing sequence of integers satisfying $M_n = o(n^D)$, and let $\gamma_n$ be a sequence satisfying $\gamma_n = \Omega(\sqrt{M_n \log M_n})$, $\gamma_n = o(M_n)$, and such that for some positive $y < x$*

$$\gamma_n \lesssim (2^{(D-2)/D}\, yM_n)^{D/2(D-1)}. \tag{38}$$

*Then for any $\varpi > 0$*

$$\mathbf{P}\left\{\sum_{\nu=1}^{M_n} T_n^\nu \le \gamma_n\right\} \lesssim M_n \exp\left(-\frac{(1-\varpi)\gamma_n^2}{2M_n}\right), \qquad (n \to \infty).$$

*Remarks.* The above lemma is a generalization of a large deviation result for symmetric random variables due to Newman (1988). Note that condition 4 imposes a sort of "asymptotic symmetry" on the random variables $T_n^\nu$. In the application of the lemma to higher-order networks we will choose a parameter $D$ slightly larger than the degree of interaction $d$.

The deviations, $\gamma_n$, encountered in the lemma can be chosen to be as large as $M_n^{1/2+1/2(D-1)}$ which are much larger than the $\sqrt{M_n}$, deviations of the Central Limit Theorem.

The proof follows a standard truncation argument (cf. Newman, 1988). We will in fact show results slightly stronger than claimed, viz.,

$$\mathbf{P}\left\{\sum_{\nu=1}^{M_n} T_n^\nu \le \gamma_n\right\} = O\left(M_n \exp\left(-\frac{\gamma_n^2}{2M_n}\right)\right)$$

for the range of $M_n$ we will be interested in. This estimate can be further tightened by strengthening some of the cruder bounds in the proof.

*Proof.* Define the truncated random variables

$$\hat{T}_n^\nu = \begin{cases} T_n^\nu & \text{if } |T_n^\nu| \le \left(\dfrac{\gamma_n^2}{2yM_n}\right)^{D/2} \\ 0 & \text{otherwise.} \end{cases}$$

By a straightforward argument it follows that

$$\mathbf{P}\left\{\sum_{\nu=1}^{M_n} T_n^\nu \le -\gamma_n\right\} \le \underbrace{M_n \mathbf{P}\left\{|T_n^\nu| > \left(\frac{\gamma_n^2}{2yM_n}\right)^{D/2}\right\}}_{P_1} + \underbrace{\mathbf{P}\left\{\sum_{\nu=1}^{M_n} \hat{T}_n^\nu \le -\gamma_n\right\}}_{P_2}.$$

Choosing $r = y$ in Eq. (33) and invoking condition 3 (recall that $y < x$) we get for any choice of $\varpi > 0$

$$\begin{aligned} P_1 &\le M_n\, e^{-\gamma_n^2/2M_n} \quad \mathbf{E}\{\exp(y|T_n^\nu|^{2/D})\} \\ &\lesssim \frac{M_n}{2} \exp\left(-\frac{(1-\varpi)\gamma_n^2}{2M_n}\right), \qquad (n \to \infty). \end{aligned} \tag{39}$$

(The choice of constant ½ is solely for algebraic convenience and does not affect the capacity results.) Similarly, choosing $r = \gamma_n/M_n$ in Eq. (34) we get

$$\begin{aligned} P_2 &\le e^{-\gamma_n^2/M_n}\left[\mathbf{E} \exp\left(-\frac{\gamma_n \hat{T}_n^\nu}{M_n}\right)\right]^{M_n} \\ &= \exp\left\{-\frac{\gamma_n^2}{M_n}\left(1 - \frac{M_n^2}{\gamma_n^2} \log \mathbf{E}(e^{-\gamma_n \hat{T}_n^\nu/M_n})\right)\right\}. \end{aligned} \tag{40}$$

*Claim.* $\mathbf{E}\,(e^{-\gamma_n \hat{T}_n^\nu/M_n}) = 1 + \gamma_n^2/2M_n^2 + o(\gamma_n^2/M_n^2)$.

*Proof.* Setting $K_n = (\gamma_n^2/2yM_n)^{D/2}$ we have

$$\begin{aligned} \mathbf{E}(\hat{T}_n^\nu) &= \int_{|t| \le K_n} t\, dF_n^\nu(t) \\ &= -\int_{|t| > K_n} t\, dF_n^\nu(t), \end{aligned}$$

with the latter equality following because $T_n^\nu$ has zero mean. Using the lower bound on $\gamma_n$ and the fact that $M_n$ is polynomially increasing, we have $K_n = \Omega\{(\log n)^{D/2}\}$, so that by condition 4 and the bounds on $\gamma_n$ we have

$$|\mathbf{E}(\hat{T}_n^\nu)| = o(n^{-D/2}) = o(M_n^{-1/2}) = o\left(\frac{\gamma_n}{M_n}\right). \tag{41}$$

Further, condition 4 also ensures that

$$\mathbf{E}\{(\hat{T}_n^\nu)^2\} = \int_{|t| \le K_n} t^2\, dF_n^\nu(t) \to 1, \qquad (n \to \infty). \tag{42}$$

Define the function $g$ by

$$g(u) = e^{-u} - 1 + u - u^2/2.$$

To prove the claim it suffices now to show that

$$\limsup_{n\to\infty} \frac{M_n^2}{\gamma_n^2} \mathbf{E}\left\{|\hat{T}_n^\nu| \left|g\left(\frac{\gamma_n \hat{T}_n^\nu}{M_n}\right)\right|\right\} = D < \infty. \tag{43}$$

In fact, if Eq. (43) holds them for any $\delta > 0$ we can choose $r(\delta)$ such that $D/r(\delta) < \delta/2$. With such a choice of $r(\delta)$ we can now choose $n$ large enough that

$$\sup_{|t|\le r(\delta)} \frac{M_n^2}{\gamma_n^2}\left|g\left(\frac{\gamma_n t}{M_n}\right)\right| < \frac{\delta}{2}.$$

Hence, if Eq. (43) holds, then for every fixed $\delta > 0$ we can choose $n$ large enough so that

$$\begin{aligned}\left|\frac{M_n^2}{\gamma_n^2} \mathbf{E}\left\{g\left(\frac{\gamma_n \hat{T}_n^\nu}{M_n}\right)\right\}\right| &\le \frac{M_n^2}{\gamma_n^2} \int_{|t|\le r(\delta)} \left|g\left(\frac{\gamma_n t}{M_n}\right)\right| \, dF_n^\nu(t) \\ &\quad + \frac{M_n^2}{\gamma_n^2} \int_{r(\delta)<|t|\le K_n} \left|g\left(\frac{\gamma_n t}{M_n}\right)\right| \, dF_n^\nu(t) \\ &\le \sup_{|t|\le r(\delta)} \frac{M_n^2}{\gamma_n^2} \left|g\left(\frac{\gamma_n t}{M_n}\right)\right| \\ &\quad + \frac{M_n^2}{\gamma_n^2} \int_{r(\delta)<|t|\le K_n} \frac{|t|}{r(\delta)} \left|g\left(\frac{\gamma_n t}{M_n}\right)\right| \, dF_n^\nu(t) \\ &< \delta.\end{aligned}$$

Thus

$$\frac{M_n^2}{\gamma_n^2} \mathbf{E}\left\{\exp\left(-\frac{\gamma_n \hat{T}_n^\nu}{M_n}\right) - 1 + \frac{\gamma_n \hat{T}_n^\nu}{M_n} - \frac{\gamma_n^2 (\hat{T}_n^\nu)^2}{2M_n^2}\right\} \to 0 \qquad (n \to \infty)$$

whenever Eq. (43) holds, and by Eqs. (41) and (42) this would establish the claim. As $g(u) \le cu^2 e^{-u}$ for some finite $c$ and all $u$, it suffices hence to show that

$$\limsup_{n\to\infty} \mathbf{E}\left\{|\hat{T}_n^\nu|^3 \exp\left(-\frac{\gamma_n \hat{T}_n^\nu}{M_n}\right)\right\} < \infty. \tag{44}$$

Now, by the truncation of $\hat{T}_n^\nu$ and the bounds on $\gamma_n$ it follows that

$$\begin{aligned}\frac{\gamma_n \hat{T}_n^\nu}{M_n} &\leq \frac{1}{M_n} (2^{(D-2)/D} y M_n)^{D/2(D-1)} |\hat{T}_n^\nu| \\ &= (2^{(D-2)/D} y M_n^{-(D-2)/D})^{D/2(D-1)} |\hat{T}_n^\nu|^{(D-2)/D} |\hat{T}_n^\nu|^{2/D} \\ &< y |T_n^\nu|^{2/D}.\end{aligned}$$

It hence follows that

$$\limsup_{n\to\infty} \mathbf{E}\left\{|\hat{T}_n^\nu|^3 \exp\left(-\frac{\gamma_n \hat{T}_n^\nu}{M_n}\right)\right\} \leq \limsup_{n\to\infty} \mathbf{E}\{|T_n^\nu|^3 e^{y|T_n^\nu|^{2/D}}\}.$$

As $y > 0$, the exponential dominates the third power when $T_n^\nu$ assumes large values. Using the fact that $y < x$ we can now invoke condition 3 to establish Eq. (44). This establishes the claim.

As $\gamma_n/M_n \to 0$, we have from Eq. (40) that

$$\begin{aligned}P_2 &\leq \exp\left[-\frac{\gamma_n^2}{M_n}\left\{1 - \frac{M_n^2}{\gamma_n^2}\log\left(1 + \frac{\gamma_n^2}{2M_n^2} + o\left(\frac{\gamma_n^2}{M_n^2}\right)\right)\right\}\right] \\ &\sim \exp\left[-\frac{\gamma_n^2}{M_n}\left\{1 - \frac{M_n^2}{\gamma_n^2}\left(\frac{\gamma_n^2}{2M_n^2} + o\left(\frac{\gamma_n^2}{M_n^2}\right)\right)\right\}\right] \\ &= \exp\left(-\frac{\gamma_n^2}{2M_n}[1 - o(1)]\right).\end{aligned}$$

Then, for every $\varpi > 0$

$$P_2 \leq \frac{M_n}{2} \exp\left(-\frac{(1-\varpi)\gamma_n^2}{2M_n}\right). \tag{45}$$

Equations (39) and (45) complete the proof. ■

## Acknowledgments

We are grateful to the anonymous referee whose suggestions did much to improve the clarity of the presentation. This work was supported in part by NSF Grants EET-8709198 and DMS-8800322, by ONR Contract 41P006-01, and by Air Force Grant, AFOSR-89-0523.

## References

Amit, D. J., Gutfreund, H., and Sompolinsky, H. (1985), Storing infinite numbers of patterns in a spin-glass model of neural networks, *Phys. Rev. Lett.* **55**, 1530–1533.

BALDI, P., AND VENKATESH, S. S. (1987), Number of stable points for spin glasses and neural networks of higher orders, *Phys. Rev. Lett.* **58**, 913–916.

BALDI, P., AND VENKATESH, S. S. (1988), On properties of networks of neuron-like elements, *in* "Neural Information Processing Systems" (D. Z. Anderson, Ed.), AIP, New York.

FELLER, W. (1968), "An Introduction to Probability Theory and its Applications," Vol. I, Wiley, New York.

GOLES, E., AND VICHNIAC, G. Y. (1986), Lyapunov functions for parallel neural networks, *in* "Neural Networks for Computing" (J. Denker, Ed.), AIP, New York.

HOPFIELD, J. J. (1982), "Neural networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci. USA* **79**, 2554–2558.

KOMLÓS, J., AND PATURI, R. (1988), Convergence results in an associative memory model, *Neural Networks* **1**(3), 239–250.

LEE, Y. C., DOOLEN, G., CHEN, H. H., SUN, G. Z. MAXWELL, T., LEE, H. Y., AND GILES, C. L. (1986), Machine learning using a higher-order correlation network, *Physica* **22D**, 276–306.

MAXWELL, T., GILES, C. L., LEE, Y. C., AND CHEN, H. H. (1986), Non-linear dynamics of artificial neural systems, *in* "Neural Networks for Computing" (J. Denker, Ed.), AIP, New York.

MCELIECE, R. J., POSNER, E. C., RODEMICH, E. R. AND VENKATESH, S. S. (1987), The capacity of the Hopfield associative memory, *IEEE Trans. Inform. Theory* **IT-33**, 461–482.

NEWMAN, C. M. (1988), Memory capacity in neural network models: Rigorous lower bounds, *Neural Networks* **1**(3), 223–238.

PERETTO, P., AND NIEZ, J. J. (1986), Long term memory storage capacity of multiconnected neural networks, *Biol. Cybernet.* **54**, 53–63.

PSALTIS, D., AND PARK, C. H. (1986), Nonlinear discriminant functions and associative memories, *in* "Neural Networks for Computing" (J. Denker, Ed.), AIP, New York.

VENKATESH, S. S. (1986), Epsilon capacity of neural networks, *in* "Neural Networks for Computing" (J. Denker, Ed.), AIP, New York.

VENKATESH, S. S., AND BALDI, P. (1989a), Random interactions in higher-order neural networks, submitted for publication.

VENKATESH, S. S., AND BALDI, P. (1991), Programmed interactions in higher-order neural networks: Maximal capacity, *J. Compl.* **7**, 316–337.

VENKATESH, S. S., AND PSALTIS, D. (1991), On reliable computation with formal neurons, *IEEE Trans. Pattern Anal. and Machine Intelligence*, to appear (Dec.).

To appear:
IEEE Transactions on Information
Theory, vol. 39, no. 1,
January 1993, (in press)

# RANDOM INTERACTIONS IN HIGHER ORDER NEURAL NETWORKS*

Pierre Baldi[†] Santosh S. Venkatesh[‡]

Submitted 26 September 1988
Revised 12 April 1991

### Abstract

We consi ler recurrent neural networks of polynomial threshold units. We study the expected number of fixed points in the case of random, symmetric interactions, equivalent to higher order spin glass models of statistical physics. We derive precise asymptotic estimates for the expected number of fixed points as a function of the margin of stability. In particular, we show that there is a critical range of margins of stability (depending on the degree of interaction) such that the expected number of fixed points with margins below the critical range grows exponentially with the number of nodes in the network, while the expected number of fixed points with margins above the critical range decreases exponentially with the number of nodes in the network. We also briefly examine the random energy model.

## 1 INTRODUCTION

Recurrent networks of formal neurons have been popular in a variety of computational applications. The model neurons in such structures are typically linear threshold elements which compute the sign of a linear form of the inputs. A recurrent network results when such elements are fully interconnected, and as in any recurrent system, the fixed points are important in the characterisation of the computations done by the structure. A particular case of interest results when the interconnections between neurons are symmetric: in such cases network dynamics are regulated by a Hamiltonian or energy function (cf. Hopfield [1] for instance). In such an instance, we can imagine the state space of the network to be embedded in an energy landscape with fixed points residing at energy minima. A classical application of such networks is in associative memory where neural interactions are adjusted so that memories are stored as local attractors.

We consider here a natural extension of the model to recurrent networks comprised of higher order neurons which compute the sign of a *polynomial* form of the inputs. The added degrees of

[*] Presented in part at the *Sixth International Conference on Mathematical Modelling*, St. Louis, Missouri, 1987, and the *Conference on Neural Information Processing Systems*, Denver, Colorado, 1987.

[†] Jet Propulsion Laboratories, Pasadena, CA 91109, and Division of Biology, Caltech, Pasadena, CA 91125. Supported by grants DMS-8800322 from the National Science Foundation and AFOSR-89-0523 from the Air Force Office of Scientific Research.

[‡] Department of Electrical Engineering, University of Pennsylvaina, Philadelphia, PA 19104. Supported in part by the National Science Foundation under grant EET-8709198.

freedom in specifying the polynomial interaction coefficients can be expected to enrich the computational dynamics that result. Distinct features emerge, however, in the analysis of these structures depending on whether the higher order interactions are programmed (or "learnt") or random.

In the programmed scenario, the goal is to tailor the higher order interaction coefficients so as to obtain desired dynamical behaviours; this leads naturally to questions of capacity and efficiency of higher order networks of given degree of polynomial interaction. In two concurrent papers [2, 3] we present rigourous results on algorithmic capacity and efficiency in programmed situations for higher order networks (cf. also Newman [4]). The main results can be summarised briefly as follows: the computational gains in higher order networks parallel the extra degrees of freedom in specifying the polynomial interaction coefficients; in particular, regardless of the algorithm used to specify the interaction coefficients, the information storage capability of a higher order network is of the order of one bit per interaction coefficient.

Higher order systems where the polynomial interactions are random may be useful as models of disordered systems in statistical physics (spin glasses), or of neural networks, before any learning has occured, or in the limiting case when too much learning has occured (the onset of senility!). These will be our focus of analysis in this paper: in particular, we consider recurrent, higher order neural networks with symmetric, random polynomial interactions. We characterise the fixed points of these structures according to their *margin of stability*[*] which is a measure of how stable a fixed point is with respect to perturbations. Our main result may be informally stated as follows:

> *There exists a critical range of margins of stability (depending on the degree of polynomial interaction) such that the expected number of fixed points with margins of stability below the critical range increases exponentially in the size of the network while the expected number of fixed points with margins of stability above the critical range decays exponentially as the size of the network is increased.*

There is thus a threshold phenomenon in evidence for the expected number of fixed points around the critical range of the margin of stability. The fact that for a certain range of margins the expected number of fixed points grows exponentially with the number of nodes in the network is not unexpected; more counter-intuitive, perhaps, is the existence of a critical margin of stability above which the expected number of fixed points actually decays as more nodes are added. We also provide exact asymptotic expressions for the coefficients and exponents in the regime of exponential behaviour, and evaluate the critical margins of stability. While considerable attention has been focused on spin glass models in the statistical physics literature, at the time of writing rigourous results appear to have been confined to the case of linear interactions and to estimates of the expectation of the *total* number of fixed points (cf. Edwards and Tanaka [5], Gross and Mezard [6], and McEliece and Posner [7]). The estimates derived here provide a finer partition of the expected number of fixed points grouped according to their margins of stability, and extend the results to higher order cases with polynomial interactions where the statistical dependences grow more acute.

The basic analytical tool used is Laplace's method for integrals. The assumed random, independent, and symmetric nature of the interactions makes for some simplicity in analysis. The results derived here for the disordered case may also give some intuition in programmed situations where the interaction dependences are weak, though a corresponding analysis of the number of fixed points for the [illegible]grammed case is typically more complicated. The analysis for the programmed case depends [illegible]gly on the algorithm of choice, and is made harder by the presence of statistical

[*] In this context [illegible]s notion is due to Komlós and Paturi [9].

dependences in the interaction coefficients, especially in higher order cases. Rigourous estimates for the total number of fixed points are known only for the case of linear interactions programmed with the outer product algorithm: McEliece, *et al* [8] conjectured based on the corresponding situation with random interactions that the number of extraneous fixed points is exponential in the number of nodes and this was rigourously shown by Komlós and Paturi [9]. The issue remains open for other algorithms such as the spectral algorithm (cf. Venkatesh and Psaltis [10]) even for the linear interaction case. For the higher order cases no formal results have been shown for any algorithm.

In Section 2, we formally introduce recurrent higher order networks, and make precise the notion of the margin of stability of a fixed point. We also show that when the polynomial interactions are symmetric, network dynamics are regulated by an energy function. In Section 3 we consider random, homogeneous, higher order networks and prove the main Theorem 3.3; Table 1 contains a listing of critical margins of stability for certain fixed degrees of interaction; Corollaries 3.5 and 3.7 highlight two principal applications of the main theorem in showing an explicit expression for the expected number of fixed points in the exponential regime for the cases where the degree of interaction is fixed, and increases unboundedly with the number of nodes in the network, respectively; and Table 2 lists coefficient and exponent values for the exponential regime for certain fixed degrees of interaction. In Section 4 we deal with non-homogeneous networks, and also briefly examine a different model of randomness known in the literature as the random energy model (cf. Derrida [11]). The proofs of the main theorems are developed in the body of the paper, while technical lemmas and calculations are confined to the two appendices.

A brief word on notation: in addition to standard asymptotic conventions, we write $x_n \lesssim y_n$ if $x_n \leq y_n$ for $n$ large enough; all logarithms in the exposition are to base $e$; we also denote by $\mathbb{B}$ the set $\{-1, 1\}$.

## 2 POLYNOMIAL THRESHOLD NETWORKS

We consider systems of $n$ densely interacting threshold units each of which yields an instantaneous state -1 or +1. (This corresponds in the literature to a system of $n$ Ising spins, or alternatively, a system of $n$ neural states.) The state space is hence the set of vertices of the hypercube. We will in this discussion also restrict our attention throughout to *symmetric interaction systems.*

Let $\mathcal{I}_d$ be the family of all subsets of cardinality $d+1$ of the set $\{1, 2, \ldots, n\}$. Clearly $|\mathcal{I}_d| = \binom{n}{d+1}$. For any subset $I$ of $\{1, 2, \ldots, n\}$, and for every state $\mathbf{u} = \{u_1, u_2, \ldots, u_n\} \in \mathbb{B}^n$, set $u_I = \prod_{i \in I} u_i$.

**Definition 2.1** A *homogeneous polynomial threshold network* of degree $d$ is a network of $n$ threshold elements with interactions specified by a set of $\binom{n}{d+1}$ real coefficients $w_I$ indexed by $I$ in $\mathcal{I}_d$. The evolution rule is asynchronous, and for $i \in \{1, \ldots, n\}$ is given by

$$u_i^+ = \operatorname{sgn}\left(\sum_{I \in \mathcal{I}_d : i \in I} w_I u_{I \setminus \{i\}}\right). \tag{1}$$

A *(non-homogeneous) polynomial threshold network* of degree $d$ is a network of $n$ threshold elements with interactions specified by a set of $\sum_{j=1}^{d} \binom{n}{j+1}$ real coefficients $w_I$ indexed by $I$ in $\bigcup_{j=1}^{d} \mathcal{I}_j$, and, for $i \in \{1, \dots, n\}$, the asynchronous evolution rule

$$u_i^+ = \operatorname{sgn}\left(\sum_{j=1}^{d} \sum_{I \in \mathcal{I}_j : i \in I} w_I u_{I \setminus \{i\}}\right). \tag{2}$$

These networks are readily seen to be natural generalisations to higher order of the familiar case of *linear threshold networks* ($d = 1$). These systems can be identified either with higher order spin glasses at zero temperature, or higher order neural networks. Starting from an arbitrary configuration or state, the system evolves asynchronously by a sequence of single "spin" flips involving spins which are misaligned with the instantaneous "field." The dynamics of these symmetric higher order systems are regulated by higher order extensions of the classical quadratic Hamiltonian. We define the *homogeneous Hamiltonian of degree d* by

$$H_d(\mathbf{u}) = -\sum_{I \in \mathcal{I}_d} w_I u_I. \tag{3}$$

In like fashion we define the *(non-homogeneous) Hamiltonian of degree d* by

$$\hat{H}_d(\mathbf{u}) = -\sum_{j=1}^{d} \sum_{I \in \mathcal{I}_j} w_I u_I. \tag{4}$$

We briefly sketch the proof of the following result.

**Proposition 2.2** *The functions $H_d$ and $\hat{H}_d$ are non-increasing under the evolution rules (1) and (2), respectively.*

PROOF: We consider the case of $H_d$, the non-homogeneous case being similar. Assume $\mathbf{u} \mapsto \mathbf{u}^+$ is a mapping along some arbitrary trajectory in state space for a homogeneous polynomial threshold network of degree $d$. The proposition is trivially true if $\mathbf{u}$ is a fixed point. Consider the case where $\mathbf{u}$ and $\mathbf{u}^+$ are distinct. By the nature of the asynchronous mapping $\mathbf{u}$ and $\mathbf{u}^+$ differ only in a single component. Without loss of generality assume the $i$-th component of $\mathbf{u}$ changes sign: $u_i^+ = -u_i$ and $u_j^+ = u_j$ if $j \neq i$. Now consider $\delta H_d(\mathbf{u}) = H_d(\mathbf{u}^+) - H_d(\mathbf{u})$. Factoring out $u_i$ in equation (3) we can write

$$H_d(\mathbf{u}) = -u_i \sum_{I \in \mathcal{I}_d : i \in I} w_I u_{I \setminus \{i\}} - \sum_{I \in \mathcal{I}_d : i \notin I} w_I u_I.$$

Hence

$$\delta H_d(\mathbf{u}) = 2u_i \sum_{I \in \mathcal{I}_d : i \in I} w_I u_{I \setminus \{i\}}.$$

By assumption we have $u_i = -u_i^+ = -\operatorname{sgn}\left(\sum_{I \in \mathcal{I}_d : i \in I} w_I u_{I \setminus \{i\}}\right)$, so that $\delta H_d(\mathbf{u}) \leq 0$. ∎

In the terminology of spin glasses, the state trajectories of these higher order networks can be seen to be following essentially a zero-temperature Monte Carlo (or Glauber) dynamics. Because of the monotonicity of the Hamiltonians given by equations (3) and (4) under the asynchronous evolution rule (1) [resp. (2)], the system always reaches a stable state (fixed point) where the relation (1) [resp. (2)], is satisfied with $u_i^+ = u_i$ for each of the $n$ spins or neural states.

**Definition 2.3** Let $\mathcal{B}$ be a non-negative parameter (possibly depending on $n$). A fixed point, $\mathbf{u} \in \mathbb{B}^n$, of a homogeneous polynomial threshold network of degree $d$ is *$\mathcal{B}$-stable* if it satisfies

$$u_i \sum_{I \in \mathcal{I}_d : i \in I} w_I u_{I \setminus \{i\}} > \mathcal{B}, \qquad i = 1, \dots, n. \tag{5}$$

In like fashion, a *$\mathcal{B}$-stable* state, $\mathbf{u} \in \mathbb{B}^n$, of a non-homogeneous polynomial threshold network of degree $d$ satisfies

$$u_i \sum_{j=1}^{d} \sum_{I \in \mathcal{I}_j : i \in I} w_I u_{I \setminus \{i\}} > \mathcal{B}, \qquad i = 1, \dots, n. \tag{6}$$

For $\mathcal{B}$-stable states, $\mathcal{B}$ represents the *margin of stability* for the fixed point; we hence refer to $\mathcal{B}$ as the *margin*. We would expect $\mathcal{B}$-stable states with large margins to tend to exhibit correspondingly large basins of attraction, i.e., to be stable with respect to relatively large perturbations. Note that according to this definition, all fixed points are 0-stable states. Komlós and Paturi [9] utilise a similar notion in an analysis of the extraneous stable states for the case of a linear interaction network ($d = 1$) programmed using the outer product algorithm.

# 3 HOMOGENEOUS NETWORKS

## 3.1 Higher Order Spin Glasses

Consider homogeneous polynomial threshold networks whose weights $w_I$, $I \in \mathcal{I}_d$ are i.i.d., $\mathcal{N}(0,1)$ random variables. This is a natural generalisation to higher order of Ising spin glasses with Gaussian interactions. We will derive an asymptotic estimate for the expected number of $\mathcal{B}$-stable states of the structure. Asymptotic results for the number of 0-stable states (fixed points) for the usual case $d = 1$ of linear threshold networks with Gaussian interactions have been reported in the literature (cf. [5, 6, 7]). We extend the technique used by McEliece and Posner [7] to obtain the general result.

As a function of $n$, let $d_n$ explicitly represent the degree of the homogeneous threshold network, with the constraint $d_n < n - 1$. To avoid trivialities we restrict ourselves to $n \geq 3$. For any given $n$, and margin $\mathcal{B} > 0$, let $F_{(n,d_n,\mathcal{B})}$ denote the expected number of $\mathcal{B}$-stable states of a homogeneous network of degree $d_n$. In the following we will estimate $F_{(n,d_n,\mathcal{B})}$ under various assumptions on $d_n$ and $\mathcal{B}$.

Let $P_{(n,d_n,\mathcal{B})}$ denote the probability that a given state $\mathbf{u}$ is $\mathcal{B}$-stable. Clearly, $F_{(n,d_n,\mathcal{B})} = 2^n P_{(n,d_n,\mathcal{B})}$. Without loss of generality we assume that $\mathbf{u} = (1, 1, \dots, 1)$. For each $n$ and $i = 1, \dots, n$, define the sequence of random variables $S_n^i$ by

$$S_n^i = \sum_{I \in \mathcal{I}_{d_n} : i \in I} w_I.$$

For $\mathbf{u}$ to be $\mathcal{B}$-stable, we require that $S_n^i > \mathcal{B}$ for $i = 1, \dots, n$.

Now, for each $n$, the random variables $S_n^i$, $i = 1, \dots, n$ are zero-mean, identically distributed, and jointly normal. Set

$$\begin{aligned} p_n &= \binom{n-2}{d_n}^{1/2}, \\ q_n &= \binom{n-2}{d_n - 1}^{1/2}. \end{aligned} \tag{7}$$

Then we have

$$\mathbf{E}(S_n^i S_n^j) = \begin{cases} p_n^2 + q_n^2 & = \begin{pmatrix} n-1 \\ d_n \end{pmatrix}, & \text{if } i = j \\ q_n^2 & = \begin{pmatrix} n-2 \\ d_n - 1 \end{pmatrix}, & \text{if } i \neq j. \end{cases}$$

Now, let $\{c_n\}$ be the sequence

$$c_n = \frac{nq_n^2}{p_n^2} = \frac{d_n}{1 - \left(\frac{d_n+1}{n}\right)}, \tag{8}$$

and define the sequence of functions $f_n$ by

$$f_n(t) = \log \Phi(t) - \frac{(t + \mathcal{B}/p_n)^2}{2c_n}, \tag{9}$$

where, in usual notation, $\Phi(t) = \int_{-\infty}^{t} \varphi(s)\,ds$ is the normal distribution function, and $\varphi(s) = (2\pi)^{-1/2}e^{-s^2/2}$ is the standard normal density function.

**Proposition 3.1**

$$F_{(n,d_n,\mathcal{B})} = \frac{2^n\sqrt{n}}{\sqrt{2\pi c_n}} \int_{-\infty}^{\infty} e^{nf_n(t)}\,dt. \tag{10}$$

Proof: We use the principle of equivalent Gaussians. Let $X^0, X^1, \ldots, X^n$ be i.i.d., $\mathcal{N}(0,1)$ random variables. Construct the random variables $Y_n^i$, $i = 1, \ldots, n$ by

$$Y_n^i = q_n X^0 + p_n X^i.$$

The random variables $\{Y_n^i\}_{i=1}^n$ are jointly normal, and have the same expectations and covariances as the random variables $\{S_n^i\}_{i=1}^n$. Hence

$$\begin{aligned} P_{(n,d_n,\mathcal{B})} \;=\; \mathbf{P}\{S_n^i > \mathcal{B},\ i = 1,\ldots,n\} &= \mathbf{P}\{Y_n^i > \mathcal{B},\ i = 1,\ldots,n\} \\ &= \mathbf{P}\left\{X^i > -\frac{q_n}{p_n}X^0 + \frac{\mathcal{B}}{p_n},\ i = 1,\ldots,n\right\} \\ &= \int_{-\infty}^{\infty} \mathbf{P}\left\{X^i > -\frac{q_n}{p_n}t + \frac{\mathcal{B}}{p_n},\ i = 1,\ldots,n\right\} \varphi(t)\,dt \\ &= \int_{-\infty}^{\infty} \Phi^n\left(\frac{q_n}{p_n}t - \frac{\mathcal{B}}{p_n}\right) \varphi(t)\,dt. \end{aligned}$$

The result follows from the defining equations (8) and (9). ∎

We will estimate the expected number of $\mathcal{B}$-stable points given by equation (10) for large $n$ using variants of Laplace's technique to estimate the integral (cf. de Bruijn [12], for instance). Rather careful asymptotic estimates are required, however, as the integral depends critically on the functions $f_n$, and these depend both on $n$ and the interaction orders $d_n$.

It will be convenient to consider margins of the following form:

$$\mathcal{B} = \beta p_n c_n^\alpha, \qquad \alpha \geq 0, \quad \beta \geq 0.$$

For given degrees of interaction, $d_n$, the expected number of $\mathcal{B}$-stable states will depend solely on the choices of the parameters $\alpha \geq 0$ and $\beta \geq 0$.

### 3.2 $\mathcal{B}$-Stability

Define the positive function

$$\psi(t) = \frac{\varphi(t)}{\Phi(t)} = \frac{e^{-t^2/2}}{\int_{-\infty}^{t} e^{-s^2/2}\,ds}.$$

In Appendix A (lemma A.1) we show that, for every given degree $d_n$ and margin $\mathcal{B}$, the function $f_n(t)$ has a unique maximum at $t = t_n$ where $t_n$ satisfies

$$\psi(t_n) - \frac{t_n}{c_n} = \frac{\mathcal{B}}{p_n c_n}. \tag{11}$$

Note that $t_n$ depends implicitly on both the margin $\mathcal{B}$ and the degree of interaction $d_n$. The following lemma, which we prove in Appendix B, provides the sought after estimate for $F_{(n,d_n,\mathcal{B})}$.

**Lemma 3.2** *Let $\mathcal{B} = \beta p_n c_n^\alpha$ with $0 \leq \alpha \leq 1$ and $\beta \geq 0$. If $d_n = o(n)$, then*

$$F_{(n,d_n,\mathcal{B})} \sim \frac{2^{n\left(1+\frac{f_n(t_n)}{\log 2}\right)}}{\sqrt{-c_n f_n''(t_n)}} \qquad (n \to \infty). \tag{12}$$

We are now in position to state the main theorem.

**Theorem 3.3** *Let $d_n = o(n)$, and consider margins of the form $\mathcal{B} = \beta p_n \sqrt{c_n}$, with $\beta \geq 0$. Then there are constants $\beta_1$ and $\beta_2$, with $0 < \beta_1 \leq \beta_2$, such that as $n \to \infty$:*

*a) $F_{(n,d_n,\mathcal{B})}$ increases exponentially with $n$ whenever $0 \leq \beta < \beta_1$;*

*b) $F_{(n,d_n,\mathcal{B})}$ decreases exponentially with $n$ whenever $\beta > \beta_2$.*

PROOF: We consider the two cases, $\{d_n\}$ bounded, and $\{d_n\}$ unbounded separately.

CASE 1. $\{d_n\}$ is bounded.

From equation (8) it is clear that $c_n \sim d_n$ is bounded. Consequently, from equations (11), (25), and (26) it follows that in equation (12) the term $f_n(t_n)$ is bounded while the term $\sqrt{-c_n f_n''(t_n)}$ is bounded away from zero. It is clear then that, for large $n$, the behaviour of $F_{(n,d_n,\mathcal{B})}$ as $\beta$ varies is determined entirely by the sign of the exponent in equation (12). Now, from equation (25) we have

$$f_n(t_n) = -\frac{t_n^2}{2}\left(1+\frac{1}{d_n}\right) - \frac{\beta t_n}{\sqrt{d_n}} - \frac{\beta^2}{2} + \log\left(\frac{d_n}{\sqrt{2\pi}(t_n+\beta\sqrt{d_n})}\right) + O\left(\frac{1}{n}\right),$$

where $t_n$ is bounded and satisfies equation (11). It is easy to verify that if $\beta = 0$ then $1+f_n(t_n)/\log 2$ is positive and bounded away from zero, i.e., the expected number of fixed points (0-stable states) increases exponentially with $n$ (see Table 2 for a listing of exponents for some fixed degrees of interaction). Now, for every $n$, $F_{(n,d_n,\mathcal{B})}$ decreases monotonically as $\beta$ increases (the expected number of $\mathcal{B}$-stable states is a monotonically decreasing function of the margin), and an examination of the above asymptotic estimate for $f_n(t_n)$ shows that as $\beta$ increases $f_n(t_n)$ eventually decreases sufficiently for $1 + f_n(t_n)/\log 2$ to become negative. Recalling that $d_n$ takes values only in some finite set, by assumption, from the above equation we can find $0 < \beta_1 \leq \beta_2$ such that

$$\limsup \frac{1}{\log 2}\left[-\frac{t_n^2}{2}\left(1+\frac{1}{d_n}\right) - \frac{\beta_1 t_n}{\sqrt{d_n}} - \frac{\beta_1^2}{2} + \log\frac{d_n}{\sqrt{2\pi}(t_n+\beta_1\sqrt{d_n})}\right] = -1, \tag{13}$$

$$\liminf \frac{1}{\log 2}\left[-\frac{t_n^2}{2}\left(1+\frac{1}{d_n}\right) - \frac{\beta_2 t_n}{\sqrt{d_n}} - \frac{\beta_2^2}{2} + \log\frac{d_n}{\sqrt{2\pi}(t_n+\beta_2\sqrt{d_n})}\right] = -1. \tag{14}$$

As $\sqrt{-c_n f_n''(t_n)}$ is bounded above and away from zero, it follows from equation (12) that for every $\beta < \beta_1$ there is $\epsilon(\beta) > 0$ such that $F_{(n,d_n,\mathcal{B})} = \Omega(2^{n\epsilon(\beta)})$; similarly, for every $\beta > \beta_2$ we can find $\delta(\beta) > 0$ such that $F_{(n,d_n,\mathcal{B})} = O(2^{-n\delta(\beta)})$.

CASE 2. $d_n \to \infty$ such that $d_n = o(n)$.

For a choice of margin $\mathcal{B} = \beta p_n \sqrt{c_n}$, with $\beta > 0$, we have from equation (9) that

$$f_n(t_n) = \log\Phi(t_n) - \frac{(t_n + \beta\sqrt{c_n})^2}{2c_n} = -\sum_{k=1}^{\infty} \frac{\Phi(-t_n)^k}{k} - \frac{(t_n + \beta\sqrt{c_n})^2}{2c_n}. \tag{15}$$

Further, using equation (24) and the asymptotic form for the error function, we have

$$\Phi(-t_n) = \frac{\varphi(t_n)}{t_n}\left(1 - O\left(\frac{1}{t_n^2}\right)\right) = \frac{\beta}{\sqrt{c_n \log c_n}}\left(1 \pm O\left(\frac{1}{\log c_n}\right)\right).$$

Substituting from equations (15), (24), and (26) in equation (12) we then have

$$F_{(n,d_n,\mathcal{B})} = 2^{n\left\{1-\frac{\beta^2}{2\log 2}-O\left(\frac{\sqrt{\log c_n}}{\sqrt{c_n}}\right)-O\left(\frac{\log c_n}{n}\right)\right\}}.$$

Setting $\beta_1 = \beta_2 = \sqrt{2\log 2}$ in the theorem, it is clear that exponentially increasing behaviour attains when $0 < \beta < \sqrt{2\log 2}$, while, for $\beta > \sqrt{2\log 2}$, the expected number of $\mathcal{B}$-stable states decreases exponentially. To complete the proof we need to show that $F_{(n,d_n,\mathcal{B})}$ increases exponentially with $n$ when $\beta = 0$. But this follows immediately because the expected number of $\mathcal{B}$-stable states is a monotonically decreasing function of the margin. ∎

For $d_n = d =$ constant, and margin $\mathcal{B} = \beta p_n \sqrt{c_n}$, the critical quantities $\beta_1 = \beta_2 = \beta^*$ of equations (13) and (14) can be precisely calculated. The critical values $\beta^*$ are listed in Table 1 for a range of fixed interaction orders. Note that the critical values $\beta^*$ appear to increase to a maximum around $d = 25$, and then decrease monotonically.

| $d$ | $\beta^*$ |
|---|---|
| 1 | 0.0690 |
| 2 | 0.1214 |
| 3 | 0.1557 |
| 4 | 0.1792 |
| 5 | 0.1960 |
| 10 | 0.2349 |
| 25 | 0.2476 |
| 50 | 0.2316 |
| 100 | 0.2023 |
| 1000 | 0.0959 |

Table 1: Critical values of margin, $\beta^*$, for various choices of fixed degree, $d$

| $d$ | $k_d$ | $\omega_d$ |
|---|---|---|
| 1 | 1.0505 | 0.2874 |
| 2 | 1.1320 | 0.4265 |
| 3 | 1.2178 | 0.5124 |
| 4 | 1.3031 | 0.5721 |
| 5 | 1.3868 | 0.6165 |
| 10 | 1.7784 | 0.7382 |
| 25 | 2.7867 | 0.8541 |
| 50 | 4.2207 | 0.9104 |
| 100 | 6.7176 | 0.9466 |
| 1000 | 39.3421 | 0.9917 |

Table 2: The behaviour of the expected number of fixed points, $F_{(n,d,0)} \sim k_d 2^{n\omega_d}$, for different values of fixed degree of interaction, $d$.

More explicit results can be deduced from Lemma 3.2. In the range where the expected number of $\mathcal{B}$-stable points increases exponentially, the multiplying coefficients and exponents can themselves be precisely calculated given the interaction orders $d_n$. Particular cases of importance result when $\{d_n\}$ converges to some $d > 0$, and in particular, the case $d_n = d =$ constant, and the case $d_n \to \infty$ monotonically.

Consider the case where $d_n = d =$ constant. Let $\alpha \geq 0$ and $\beta \geq 0$ specify the margin $B$, and let $s$ be the unique solution of the equation

$$\psi(s) = \frac{s + \beta d^{\alpha}}{d}.$$

The location, $t_n$, of the maximum of $f_n$ (satisfying equation (11)) can be approximated up to terms of the order of $n^{-2}$ by

$$t_n = s + \frac{\kappa}{n} + O\left(\frac{1}{n^2}\right),$$

where $\kappa$ is independent of $n$ and satisfies

$$\kappa = \frac{d(d+1)[s + (1-\alpha)\beta d^{\alpha}]}{d + (s + \beta d^{\alpha})[s(d+1) + \beta d^{\alpha}]}. \tag{16}$$

Using the above approximation for $t_n$ in Lemma 3.2, and collecting all terms up to the order of $n^{-2}$ in the exponent in equation (12) yields the following result.

**Corollary 3.4** *If $d_n = d > 0$, and $B = \beta p_n c_n^{\alpha}$ with $\alpha \geq 0$ and $\beta \geq 0$, then, as $n \to \infty$, $F_{(n,d_n,B)} \sim k_d 2^{n\omega_d}$, where the multiplying coefficient, $k_d$, and exponent, $\omega_d$, are independent of $n$ (and depend solely on the interaction order, $d$, and the margin parameters, $\alpha$ and $\beta$); specifically,*

$$\omega_d = 1 - \frac{d+1}{2d\log 2}\left(s^2 + \frac{2s\beta d^{\alpha}}{d+1} + \frac{\beta^2 d^{2\alpha}}{d+1}\right) - \frac{1}{\log 2}\log\left(\frac{\sqrt{2\pi}(s + \beta d^{\alpha})}{d}\right),$$

*and $k_d$ can be expressed in the form $C\,e^D$ where*

$$C = \left[\frac{d}{s^2(d+1) + s\beta d^{\alpha}(d+2) + \beta^2 d^{2\alpha} + d}\right]^{1/2},$$

*and with $\kappa$ as in equation (16),*

$$\begin{aligned} D &= \frac{d+1}{2d}\left[s^2 - 2s\{\kappa - \beta d^{\alpha}(1-\alpha)\} + \beta^2 d^{2\alpha}(1-2\alpha) + 2d\right] \\ &\quad - \kappa\beta d^{\alpha-1} - \frac{\alpha\beta d^{\alpha}(d+1)}{s + \beta d^{\alpha}} - \frac{\kappa}{s + \beta d^{\alpha}}. \end{aligned}$$

An important special case results when we choose the margin to be identically zero.

**Corollary 3.5** *If $d_n = d > 0$, the expected number of fixed points is asymptotically $\sim k_d 2^{n\omega_d}$ where*

$$\begin{aligned} \omega_d &= 1 - \frac{1}{\log 2}\left[\frac{(d+1)s^2}{2d} + \log\left(\frac{s\sqrt{2\pi}}{d}\right)\right], \\ k_d &= \sqrt{\frac{d}{d + s^2(d+1)}}\exp\left(\frac{(d+1)s^2}{2d}\right). \end{aligned}$$

Table 2 lists the exponent $\omega_d$ and the multiplying coefficient $k_d$ for various choices of fixed interaction order $d$ with a choice of zero margin.

The monotonicity of $F_{(n,d_n,\mathcal{B})}$ with $\mathcal{B}$ yields

**Corollary 3.6** *Let $\mathcal{B} = \beta p_n c_n^\alpha$ be the margin. If $d_n \to \infty$ such that $d_n = o(n)$ then:*

*a) the expected number of $\mathcal{B}$-stable states increases exponentially with $n$ if $0 \leq \alpha < 1/2$ and $\beta \geq 0$;*

*b) the expected number of $\mathcal{B}$-stable states asymptotically tends to zero as $n \to \infty$ if $\alpha > 1/2$ and $\beta > 0$.*

Note from Table 2 that as $d$ becomes large $F_{(n,d_n,\mathcal{B})}$ approaches $2^n$. This is supported by the following result which gives the number of fixed points (0-stable states) when the interaction orders are allowed to grow large.

**Corollary 3.7** *If as $n \to \infty$, for any fixed choice of $\tau$ with $0 < \tau < 1$, $\{d_n\}$ satisfies $d_n = \Omega[n/(\log n)^\tau]$, and $d_n = o(n)$, then the expected number of fixed points (zero margin) is given by $F_{(n,d_n,0)} \sim k_{d_n} 2^{n\omega_{d_n}}$, as $n \to \infty$, where*

$$\begin{aligned} k_{d_n} &= \frac{d_n}{2\sqrt{2\pi}\log d_n}, \\ \omega_{d_n} &= 1 - \frac{\log d_n}{d_n \log 2} + \frac{\log\log d_n}{2d_n \log 2} + \frac{\log(\sqrt{4\pi}/e)}{d_n \log 2}. \end{aligned}$$

Proof: Consider the exponent in the integrand of equation (10). We have

$$nf_n(t_n) = n\log\Phi(t_n) - \frac{nt_n^2}{2c_n} = -\sum_{k=1}^{\infty} \frac{n}{k}\Phi(-t_n)^k - \frac{nt_n^2}{2c_n}.$$

Using the asymptotic formula for the tails of the normal distribution (cf. Feller's text [13], for instance) together with Lemma A.3 (equation (23)), and equations (8) and (26) in equation (12) completes the proof. ∎

Note that for this case, the multiplying coefficient $k_{d_n}$ and exponent $\omega_{d_n}$ assume particularly simple closed form expressions depending solely on the interaction order $d_n$. Note also that $\omega_{d_n} \to 1$ as $n \to \infty$, as is expected. The growth of $d_n$ with $n$ is rather rapid in Corollary 3.7. Results akin to Corollary 3.7 can be computed for slower rates of growth of $d_n$ (for instance, $d_n = n^\alpha$, $0 < \alpha < 1$). We do not yet have rigourous results, however, for the case where $d_n$ scales linearly with $n$.

# 4 NON-HOMOGENEOUS NETWORKS

## 4.1 Higher Order Spin Glasses

The non-homogeneous case has several more degrees of interconnection freedom. The results of the last section can, however, be simply extended to this case.

Analogously with equations (7) and (8) let

$$\hat{p}_n = \left[\sum_{k=1}^{d_n} \binom{n-2}{k}\right]^{1/2},$$

and

$$\hat{c}_n = \frac{n \sum_{j=1}^{d_n} \binom{n-2}{j-1}}{\sum_{j=1}^{d_n} \binom{n-2}{j}},$$

and to every choice of margin (fixed $\beta \geq 0$ and $\alpha \geq 0$) $\mathcal{B} = \beta p_n c_n^\alpha$ in the homogeneous case associate a margin $\hat{\mathcal{B}} = \beta \hat{p}_n \hat{c}_n^\alpha$ in the non-homogeneous case. Define the sequence of functions $\hat{f}_n$ (corresponding to equation (9)) by

$$\hat{f}_n(t) = \log \Phi(t) - \frac{(t + \hat{\mathcal{B}}/\hat{p}_n)^2}{2\hat{c}_n}. \tag{17}$$

Let $\hat{F}_{(n,d_n,\hat{\mathcal{B}})}$ denote the expected number of $\hat{\mathcal{B}}$-stable states of a non-homogeneous algebraic threshold network of degree $d_n$ with Gaussian interactions.

**Proposition 4.1**

$$\hat{F}_{(n,d_n,\hat{\mathcal{B}})} = \frac{\sqrt{n}\, 2^n}{\sqrt{2\pi \hat{c}_n}} \int_{-\infty}^{\infty} e^{n \hat{f}_n(t)}\, dt. \tag{18}$$

PROOF: For $i = 1, \ldots, n$ set

$$\hat{S}_n^i = \sum_{j=1}^{d_n} \sum_{I \in \mathcal{I}_j : i \in I} w_I.$$

Noting that $\hat{F}_{(n,d_n,\hat{\mathcal{B}})} = 2^n \mathbf{P}\{\hat{S}_n^i > \hat{\mathcal{B}},\, i = 1, \ldots, n\}$, the proof follows the same outline as that for Proposition 3.1. ∎

**Theorem 4.2** *If* $d_n = o(n)$ *then* $\hat{F}_{(n,d_n,\hat{\mathcal{B}})} \sim F_{(n,d_n,\mathcal{B})}$ *as* $n \to \infty$.

PROOF: We use the following inequality due to Blake and Darabian [14]. Set $r = d/(n - d + 1)$. Then

$$\frac{1}{1-r}\left(1 - \frac{(n+1)r^2}{d(n-d+1)(1-r)^2}\right) < \frac{\sum_{j=0}^{d} \binom{n}{j}}{\binom{n}{d}} < \frac{1}{1-r}.$$

For $d_n = o(n)$ we hence have

$$\hat{c}_n = \frac{n \sum_{j=1}^{d_n} \binom{n-2}{j-1}}{\sum_{j=1}^{d_n} \binom{n-2}{j}} = \frac{n \binom{n-2}{d_n - 1}}{\binom{n-2}{d_n}} \left(1 - \frac{1}{n} - O(\frac{d_n}{n^2})\right) = c_n \left(1 - \frac{1}{n} - O(\frac{d_n}{n^2})\right).$$

The analysis in Theorem 3.3 now continues to hold *in toto*. ∎

So far we have considered relatively small interaction orders, $d_n = o(n)$. A theoretically important case results when $d_n$ is allowed to grow linearly with $n$. In fact, as $d_n$ approaches $n$, almost all dichotomies of $2^n$ points in binary $n$-space can be separated by a non-homogeneous network (Venkatesh and Baldi [2]). It is useful, hence, to estimate the number of fixed points, $\hat{F}_{(n,d_n,0)}$, for the random case when $d_n$ grows linearly with $n$.

**Theorem 4.3** *If $d_n \sim n/2$ then $\hat{F}_{(n,d_n,0)} \sim 2^n/(n+1)$ as $n \to \infty$.*

Proof: If $d_n \sim n/2$ then $\hat{c}_n = n[1 \pm O(1/\sqrt{n})]$. Hence from equations (17) and (18),

$$\hat{F}_{(n,d_n,\hat{\mathcal{B}})} = \frac{2^n}{\sqrt{2\pi}} \left(1 \pm O(\frac{1}{\sqrt{n}})\right) \int_{-\infty}^{\infty} \Phi(t)^n e^{-t^2/2} e^{-t^2/v_n}\, dt, \tag{19}$$

where $v_n = \Omega(\sqrt{n})$. Fix $0 < \tau < 1/4$. Then

$$\frac{[\Phi(n^\tau)^{n+1} - \Phi(-n^\tau)^{n+1}]\, e^{-n^{2\tau}/v_n}}{n+1} < \frac{1}{\sqrt{2\pi}} \int_{-n^\tau}^{n^\tau} \Phi(t)^n e^{-t^2/2} e^{-t^2/v_n}\, dt < \frac{\Phi(n^\tau)^{n+1} - \Phi(-n^\tau)^{n+1}}{n+1},$$

while

$$0 \leq \frac{1}{\sqrt{2\pi}} \int_{|t| \geq n^\tau} \Phi(t)^n e^{-t^2/2} e^{-t^2/v_n}\, dt \leq e^{-n^{2\tau}/v_n} \left(\frac{1 - \Phi(n^\tau)^{n+1} + \Phi(-n^\tau)^{n+1}}{n+1}\right).$$

Now $\Phi(n^\tau)^{n+1} \to 1$, $\Phi(-n^\tau)^{n+1} \to 0$, and $n^{2\tau}/v_n \to 0$ as $n \to \infty$. Hence

$$\begin{aligned}
\frac{1}{\sqrt{2\pi}} \int_{-n^\tau}^{n^\tau} \Phi(t)^n e^{-t^2/2} e^{-t^2/v_n}\, dt &\sim \frac{1}{n+1} \qquad (n \to \infty),\\
\frac{1}{\sqrt{2\pi}} \int_{|t| \geq n^\tau} \Phi(t)^n e^{-t^2/2} e^{-t^2/v_n}\, dt &= o(\frac{1}{n+1}) \qquad (n \to \infty).
\end{aligned}$$

The proof is completed by substitution in equation (19). ∎

## 4.2 Random Energy Model

The dynamics of the symmetric interaction systems considered above are characterised by Hamiltonians or energies. The determination of the number of fixed points of such a system is hence equivalent to counting the number of states which form (local) energy minima. For higher order spin glasses, the energy of each state given by equation (3) is an $\mathcal{N}\left(0, \binom{n-1}{d_n-1}\right)$ random variable. The energies of different states are *dependent*, identically distributed normal random variables.

The random energy model (cf. Derrida [11]) is an allied system which *assigns* energies as i.i.d., $\mathcal{N}(0,1)$ random variables to the vertices of the hypercube. State energies are now independent normal random variables. Such an assignment of state energies results in random acyclic orientations of the vertices of the hypercube (cf. Baldi [15]) defined by state transitions: $\mathbf{u} \mapsto \mathbf{v}$ iff $H(\mathbf{u}) > H(\mathbf{v})$. For any given assignment of energies, the corresponding acyclic orientation can be realised by a (non-homogeneous) threshold network with degree $d_n = n$. In particular, we have $2^n$ interaction coefficients for such a system so that any particular assignment of $2^n$ state energies can be realised for a particular choice of coefficients.

Let $G_n$ be the number of local energy minima corresponding to a random acyclic orientation.

**Theorem 4.4**

$$\begin{aligned}\mathbf{E}(G_n) &= \frac{2^n}{n+1}\\ \mathrm{Var}(G_n) &= \frac{(n-1)2^{n-1}}{(n+1)^2}.\end{aligned}$$

Proof: Let $\mathbf{s}^i$, $i = 1, \ldots, 2^n$, enumerate the vertices of the hypercube. For $i = 1, \ldots, 2^n$, let the random variable $I^i$ be the indicator for state $\mathbf{s}^i$, i.e.,

$$I^i = \begin{cases} 1 & \text{if } \mathbf{s}^i \text{ is an energy minimum} \\ 0 & \text{otherwise.} \end{cases}$$

We then have $G_n = \sum_{i=1}^{2^n} I^i$. The probability, $p = \mathbf{P}\{I^i = 1\}$, that any particular state is a local minimum is just the probability that it is assigned a lower energy value than any of its $n$ nearest neighbours (at Hamming distance one from it). As the assigned energies are i.i.d. random variables, we have $p = 1/(n+1)$. Hence the expected number of fixed points is $2^n/(n+1)$. We now compute the joint probability that two states $\mathbf{s}^i$ and $\mathbf{s}^j$ are energy minima. Let $d_{ij}$ represent the Hamming distance between $\mathbf{s}^i$ and $\mathbf{s}^j$. It is easy to see that

$$\mathbf{P}\{I^iI^j = 1\} = \begin{cases} 1/(n+1)^2 & \text{if } d_{ij} > 2 \\ 1/n(n+1) & \text{if } d_{ij} = 2 \\ 0 & \text{if } d_{ij} = 1 \\ 1/(n+1) & \text{if } d_{ij} = 0. \end{cases}$$

Now, we have

$$\begin{aligned}\mathrm{Var}(G_n) = \mathbf{E}(G_n^2) - (\mathbf{E}G_n)^2 &= \sum_{i=1}^{2^n}\sum_{j=1}^{2^n} \mathbf{P}\{I^iI^j = 1\} - \left(\sum_{i=1}^{2^n} \mathbf{P}\{I^i = 1\}\right)^2 \\ &= \frac{2^n}{n+1} + 2\sum_{\substack{i<j\\ d_{ij}=2}} \frac{1}{n(n+1)} + 2\sum_{\substack{i<j\\ d_{ij}>2}} \frac{1}{(n+1)^2} - \left(\frac{2^n}{n+1}\right)^2.\end{aligned}$$

Collecting terms and simplifying yields the final result. ∎

Note that the result of Theorem 4.4 provides anecdotal support for the result of Theorem 4.3 as a sort of limiting result. Stronger results can be shown for the random energy model: the number of fixed points, $G_n$, exhibits central tendency. Let $G_n^*$ denote the normalised r.v.

$$G_n^* = \frac{G_n - \mathbf{E}\,G_n}{\sqrt{\mathrm{Var}\,G_n}}.$$

**Theorem 4.5** *There is an absolute positive constant $C$ such that for every $x$*

$$|\mathbf{P}\,(G_n^* < x) - \Phi(x)| \le C2^{-0.25n}.$$

We refer the reader to the papers by Baldi, *et al* [16, 17] for a proof of the theorem. It is an open question whether the number of fixed points of higher order spin glasses also exhibits central tendency.

# 5 CONCLUSIONS

We have rigourously estimated the expected number of stable points of higher order spin glasses with generalised Gaussian interactions. The critical feature observed here is the threshold phenomenon that is evidenced in the expected number of fixed points around a range of degree dependent critical margins. For margins below the critical range we have shown a precise exponentially increasing form of the solution, while for margins greater than the critical range we have shown that the expected number of fixed points decreases exponentially with the number of nodes in the network. Open questions remain on a more precise determination (than the mean) of the number of fixed points (as a function of the margin), and in particular, on whether there is central tendency as in the random energy model.

The results of this paper appear to have relevance to the programmed situation where interaction strengths are to be chosen for which *specified* collections of binary $n$-tuples are fixed points with some desired radius of attraction. In such cases it is important to be cognisant of the number of extraneous fixed points—and their radii of attraction—that are developed incidentally. Rigourous results have, however, been shown only for the linear interaction case ($d = 1$) with interactions programmed by the outer product algorithm (Komlós and Paturi [9]). The analysis appears to be substantially harder for higher order cases, even for the relatively simple outer product algorithm (cf. Newman's earlier paper [4] and our two concurrent papers [2, 3] for illustrations of the difficulties caused by the more severe statistical dependences in higher order cases). The extraneous fixed point structure of other algorithms, such as the spectral algorithm (Venkatesh and Psaltis [10]), is even less understood, especially in the higher order versions. It is not readily apparent whether the results derived here for the case of random, symmetric interactions (especially Theorem 3.3 and the corollaries) can be utilised in a rigourous analysis in programmed cases; nonetheless, these results may provide qualitative indications of behaviours that may be expected in programmed cases, especially when the dependences are weak.

**Acknowledgement**

We would like to thank the referees for their suggestions which served to help focus the main results of the paper.

# A Properties of $f_n$

**Lemma A.1** *For each n (and any choice of margin, $B > 0$, and degree, $d_n$):*

*a) $f_n$ is a convex $\cap$, strictly negative function with a unique maximum at $t = t_n$;*

*b) for $t \geq 1$, $f''_n$ increases monotonically to $-1/c_n$ as $t \to \infty$.*

PROOF: First we claim that $\psi$ is a positive, monotone decreasing function. Clearly $\psi(t) > 0$ for all $t$. Consider $\psi'(t) = -t\psi(t) - \psi(t)^2$. We have $\psi'(t) < 0$ for $t \geq 0$. Now, for $t > 0$ consider

$$\psi(-t) = \frac{\varphi(-t)}{\Phi(-t)} > \frac{\varphi(-t)}{\frac{1}{|t|}\varphi(-t)} = t > 0.$$

Hence $\psi'(t) < 0$ for all $t$ so that $\psi$ is monotone decreasing. By repeated differentiation of equation (9) we have

$$\begin{aligned} f_n'(t) &= \psi(t) - \frac{t + \mathcal{B}/p_n}{c_n}, & (20) \\ f_n''(t) &= -\psi(t)^2 - t\psi(t) - \frac{1}{c_n}. & (21) \end{aligned}$$

Now $f_n''(t) = \psi'(t) - 1/c_n < 0$ for all $t$ so that $f_n$ is strictly convex $\cap$, while the monotonicity of $\psi$ guarantees a unique solution at $t = t_n$ to $f_n'(t) = 0$. As $f_n(t) < 0$ for all $t$ by inspection of equation (9), part (a) follows. Now note that

$$[t\psi(t)]' = -\psi(t)[t^2 + t\psi(t) - 1] \leq -t\psi(t)^2 < 0 \qquad (t \geq 1).$$

Hence both $\psi(t)$ and $t\psi(t)$ decrease monotonically to zero so that (b) follows. ▮

**Lemma A.2** *For each $n$, $f_n$ has derivatives of all orders, and in fact, for $k \geq 3$, the derivatives $f_n^{(k)}$ are independent of $n$ and have the representation*

$$f_n^{(k)}(t) = \sum_{l=0}^{\lfloor k/2 \rfloor} \sum_{m=1}^{k-2l} c_{l,m}^{(k)}\, t^{k-2l-m}\, \psi(t)^m, \tag{22}$$

*where the coefficients $c_{l,m}^{(k)}$ are real constants independent of $n$, and $c_{0,1}^{(k)} = (-1)^{k-1}$.*

PROOF: Note that for $k \geq 3$ we have $f_n^{(k)}(t) = \frac{d^{k-1}\psi(t)}{dt^{k-1}}$. The result follows by induction. ▮

**Lemma A.3** *Let $\mathcal{B} = \beta p_n c_n^\alpha$ with $\alpha \geq 0$ and $\beta \geq 0$, and let $f_n$ achieve its maximum at $t_n$. Then as $n \to \infty$:*

*a) if $d_n \to d$, then $t_n \to s$ where $s$ satisfies*

$$\psi(s) - \frac{s}{d} = \beta d^{\alpha-1};$$

*b) if $d_n \to \infty$, and $\alpha = 0$ or $\beta = 0$, then*

$$t_n = \left[2\log c_n - \log\log c_n - \log(4\pi) + O\left(\frac{1}{\sqrt{\log c_n}}\right)\right]^{1/2}; \tag{23}$$

*c) if $d_n \to \infty$, $0 < \alpha < 1$, and $\beta > 0$, then*

$$t_n = \left[2(1-\alpha)\log c_n - 2\log(\beta\sqrt{2\pi}) - O\left(\frac{\sqrt{\log c_n}}{c_n^\alpha}\right)\right]^{1/2}. \tag{24}$$

PROOF: Part (a) follows by continuity of $\psi$ as $c_n \to d_n$ $(n \to \infty)$ from equation (8). Parts (b) and (c) can be verified by direct substitution. ∎

REMARKS:

$$\begin{aligned} f_n(t_n) &= -\frac{t_n^2}{2}\left(1+\frac{1}{c_n}\right) - \beta t_n c_n^{\alpha-1} - \frac{\beta^2}{2}c_n^{2\alpha-1} - \log\left(\frac{\sqrt{2\pi}}{c_n}(t_n+\beta c_n^{\alpha})\right), && (25)\\ f_n'(t_n) &= 0, \\ f_n''(t_n) &= -\frac{t_n^2}{c_n}\left(1+\frac{1}{c_n}\right) - t_n\beta c_n^{\alpha-1}\left(1+\frac{2}{c_n}\right) - \beta^2 c_n^{2\alpha-2} - \frac{1}{c_n}, && (26)\\ f_n^{(k)}(t_n) &\sim (-1)^{k-1}\frac{t_n^{k-1}}{c_n}(t_n+\beta c_n^{\alpha}), \qquad k \geq 3, \quad d_n \to \infty. && (27) \end{aligned}$$

Note that for $k \geq 3$, $f_n^{(k)}(t)$ *does not depend on* $n$ *any more* so that uniform bounds can be obtained.

We will seek to approximate the functions, $f_n(t)$, by the first few terms of a Taylor series expansion. Specifically, for particular choices of $\epsilon_n > 0$ and $\delta_n > 0$ we use

$$|t - t_n| < \delta_n \Longrightarrow \left| f_n(t) - f_n(t_n) - f_n''(t_n)\frac{(t-t_n)^2}{2}\right| < \epsilon_n \frac{(t-t_n)^2}{2}. \qquad (28)$$

The next two lemmas outline conditions under which the above holds.

**Lemma A.4** *If the sequence $\{d_n\}$ is bounded then for any specification of margin, $B = \beta p_n c_n^{\alpha}$ with $\alpha \geq 0$ and $\beta \geq 0$, and for any $\epsilon > 0$, we can find $\delta > 0$ uniform with respect to $n$ such that equation (28) holds with a choice of $\epsilon_n = \epsilon$ and $\delta_n = \delta$.*

PROOF: Set $g_n(t) = f_n(t) - f_n(t_n) - f_n''(t_n)(t-t_n)^2/2$. We have $g_n(t_n) = g_n'(t_n) = g_n''(t_n) = 0$. Applying the Mean Value Theorem, we can find $0 < \alpha < 1$ such that, $g_n(t_n + \alpha\zeta) = \zeta g_n'(t_n + \alpha\zeta)$, while $g_n'(t_n + \alpha\zeta) = o(\alpha\zeta)$ as $\zeta \to 0$. Hence $g_n(t_n + \zeta) = o(\zeta^2)$, $(\zeta \to 0)$. Thus, for each $n$, and every $\epsilon > 0$, we can find $\delta_n > 0$ such that $|g_n(t)| < \epsilon(t-t_n)^2/2$ whenever $|t - t_n| < \delta_n$.

Now assume without loss of generality that $d_n$ takes values from the finite set $\{\mu^1, \ldots, \mu^K\}$. For $i = 1, \ldots, K$, set

$$\begin{aligned} f^i(t) &= \log\Phi(t) - \frac{(t+\beta(\mu^i)^{\alpha})^2}{2\mu^i}, \\ g^i(t) &= f^i(t) - f^i(t^i) - f^{i\prime\prime}(t^i)\frac{(t-t^i)^2}{2}, \end{aligned}$$

where $f^i$ has its maximum at $t^i$. Then for every $\epsilon > 0$ there exists $\delta^i > 0$ such that $|g^i(t)| < \epsilon(t-t^i)^2/4$ whenever $|t - t^i| < \delta^i$. Now $c_n = d_n + O(1/n)$ so that from equation (11) it follows that $t_n = t^i + O(1/n)$ for some $i \in \{1, \ldots, K\}$. [As $\psi$ is monotone decreasing, we have:

$$t_n - t^i < x = \left(\frac{t^i}{d_n} - \frac{t^i}{c_n}\right) \Big/ \left(\frac{1}{c_n}\right) = t^i O\left(\frac{1}{n}\right).$$

As $d_n$ is bounded we are guaranteed that $t^i$ is also bounded, so that the result follows.] From equations (9), (25), and (26) it hence follows that

$$|g_n(t)| = |g^i(t)| + O\left(\frac{1}{n}\right) < \epsilon\frac{(t-t_n)^2}{4} + O\left(\frac{1}{n}\right) \qquad (|t-t_n| < \delta^i).$$

We can hence choose $N$ such that for $n \geq N$, we have $|g_n(t)| < \epsilon(t-t_n)^2/2$ whenever $|t - t_n|<\min\{\delta^1,\ldots,\delta^K\}$. We can finally choose a smallest $\delta = \min\{\delta_1,\ldots,\delta_N,\delta^1,\ldots,\delta^K\}$ to establish uniformity. ∎

**Lemma A.5** *Let $\mathcal{B} = \beta p_n c_n^{\alpha}$ be the margin. If $d_n \to \infty$ as $n \to \infty$, then equation (28) holds for $n$ large enough for the following choices of $\epsilon_n$ and $\delta_n$:*

*a) $\epsilon_n = \lambda \log c_n / c_n$ and $\delta_n = \lambda/\sqrt{128 \log c_n}$ if $\alpha = 0$ or $\beta = 0$;*

*b) $\epsilon_n = \lambda\sqrt{\log c_n}/c_n^{1-\alpha}$ and $\delta_n = \lambda/(8\beta(1-\alpha)\sqrt{\log c_n})$ if $0 < \alpha < 1$ and $\beta > 0$.*

*Here $\lambda > 0$ is a suitably small (but fixed) choice of parameter.*

Proof: We will prove the result for the case $\{\alpha = 0 \text{ or } \beta = 0\}$; the proof for the case $\{0 < \alpha < 1 \text{ and } \beta > 0\}$ is similar.

Consider a choice of margin $\mathcal{B} = \beta p_n c_n^{\alpha}$ with $\alpha = 0$ or $\beta = 0$. Set $\epsilon_n = \lambda \log c_n / c_n$ and $\delta_n = \lambda/\sqrt{128 \log c_n}$ for some $\lambda > 0$ to be specified suitably small. In the proof of Lemma A.4 set $\epsilon = \epsilon_n$. Now, it suffices to show that $|g_n'(t_n + \zeta)| < \epsilon_n|\zeta|/2$ whenever $|\zeta| < \delta_n = \lambda/\sqrt{128 \log c_n}$, $(n \to \infty)$. We have

$$|g_n'(t_n + \zeta)| = |f_n'(t_n + \zeta) - f_n''(t_n)\zeta|.$$

By the Mean Value Theorem, there exists $0 < \beta < 1$ such that

$$|f_n'(t_n + \zeta)| = |\zeta|\,|f_n''(t_n + \beta\zeta)|.$$

Now consider

$$|f_n'(t_n - \delta_n) + f_n''(t_n)\delta_n| = \delta_n|-f_n''(t_n - \beta\delta_n) + f_n''(t_n)| \leq \delta_n|-f_n''(t_n - \delta_n) + f_n''(t_n)|.$$

The last inequality follows from Lemma A.1(b) as $f_n''$ is negative and increases monotonically to $-1/c_n$ for large $t$, and by Lemma A.3(b) which ensures that $t_n \sim \sqrt{2 \log d_n} \to \infty$, $(n \to \infty)$. Using equation (21) with Lemma A.3(b), as $n \to \infty$ we have

$$\begin{aligned} |f_n'(t_n - \delta_n) + f_n''(t_n)\delta_n| &\leq \delta_n\,|\psi(t_n - \delta_n)^2 + (t_n - \delta_n)\psi(t_n - \delta_n) - \psi(t_n)^2 - t_n\psi(t_n)| \\ &\leq \frac{\delta_n t_n}{\sqrt{2\pi}}\, e^{-t_n^2/2}\left|e^{t_n\delta_n - \delta_n^2} - 1\right| + O\left(\frac{1}{c_n\sqrt{\log c_n}}\right). \end{aligned}$$

We have $t_n\delta_n = O(\lambda)$ so that for $\lambda$ sufficiently small

$$|f_n'(t_n - \delta_n) + f_n''(t_n)\delta_n| \lesssim \delta_n\left[\frac{\sqrt{2}\,\delta_n t_n^2}{\sqrt{\pi}}\, e^{-t_n^2/2}\right] \sim \epsilon_n\frac{\delta_n}{2} \qquad (n \to \infty).$$

Similarly

$$|f_n'(t_n - \delta_n) - f_n''(t_n)\delta_n| \lesssim \epsilon_n\frac{\delta_n}{2} \qquad (n \to \infty).$$

By Lemma A.1, the above inequalities hold in the $\delta_n$-neighbourhood of $t_n$, and this completes the proof. ∎

# B The Main Lemma

We prove here Lemma 3.2, restated below for convenience.

**Lemma 3.2** *Let $\mathcal{B} = \beta p_n c_n^\alpha$ with $0 \leq \alpha \leq 1$ and $\beta \geq 0$. If $d_n = o(n)$, then*

$$F_{(n,d_n,\mathcal{B})} \sim \frac{2^{n\left(1+\frac{f_n(t_n)}{\log 2}\right)}}{\sqrt{-c_n f_n''(t_n)}} \qquad (n \to \infty).$$

PROOF: We will consider separately the cases where $\{d_n\}$ is bounded and $\{d_n\}$ is unbounded.
CASE 1. $\{d_n\}$ is bounded.

The sequence $f_n''(t_n)$ is bounded strictly away from both zero and infinity. Hence set $\xi = \inf|f_n''(t_n)| > 0$ and $\kappa = \sup|f_n''(t_n)| < \infty$. Fix $\epsilon$ arbitrarily in the open interval $(0,\xi)$. Choose $\delta > 0$ uniform with respect to $n$ by Lemma A.4. By Proposition 3.1 we have

$$\frac{\sqrt{2\pi c_n}}{2^n\sqrt{n}} F_{(n,d_n,\mathcal{B})} = \int_{|t-t_n|<\delta} e^{nf_n(t)}\, dt + \int_{|t-t_n|\geq\delta} e^{nf_n(t)}\, dt.$$

Let $\vartheta$ be a parameter, $|\vartheta| \leq \epsilon$. Consider

$$\begin{aligned} \int_{t_n-\delta}^{t_n+\delta} e^{n(f_n''(t_n)+\vartheta)(t-t_n)^2/2}\, dt &= \left[\frac{-2\pi}{n(f_n''(t_n)+\vartheta)}\right]^{1/2} - 2\Phi\left(-\delta\sqrt{-n(f_n''(t_n)+\vartheta)}\right) \qquad (29)\\ &= \left[\frac{-2\pi}{n(f_n''(t_n)+\vartheta)}\right]^{1/2} - O\left(\frac{e^{-\delta^2 n(\xi-\vartheta)/2}}{\delta\sqrt{n(\xi-\vartheta)}}\right). \end{aligned}$$

By Lemma A.4 it then follows that

$$\begin{aligned} \left[\frac{-2\pi}{n(f_n''(t_n)-\epsilon)}\right]^{1/2} - O\left(\frac{e^{-q(\delta)n}}{\sqrt{n}}\right) &< e^{-nf_n(t_n)}\int_{t_n-\delta}^{t_n+\delta} e^{nf_n(t)}\, dt\\ &< \left[\frac{-2\pi}{n(f_n''(t_n)+\epsilon)}\right]^{1/2} - O\left(\frac{e^{-q(\delta)n}}{\sqrt{n}}\right) \end{aligned}$$

where $q(\delta) > 0$ depends solely on $\delta$. As $\epsilon$ was arbitrary we have

$$\int_{t_n-\delta}^{t_n+\delta} e^{nf_n(t)}\, dt \sim e^{nf_n(t_n)}\left(\frac{-2\pi}{nf_n''(t_n)}\right)^{1/2} \qquad (n \to \infty). \qquad (30)$$

Let $\zeta = \sup|f_n(t_n)|$. The sequence $\{f_n(t_n)\}$ is bounded so that $\zeta < \infty$. For each $n$ set

$$h_n(\delta) = \max\{f_n(t_n-\delta) - f_n(t_n),\ f_n(t_n+\delta) - f_n(t_n)\} < 0.$$

Then

$$\begin{aligned} \int_{|t-t_n|\geq\delta} e^{nf_n(t)}\, dt &= e^{nf_n(t_n)}\int_{|t-t_n|\geq\delta} e^{n(f_n(t)-f_n(t_n))}\, dt\\ &\leq e^{nf_n(t_n)}\left\{e^{(n-1)h_n(\delta)-f_n(t_n)}\int_{|t-t_n|\geq\delta}\Phi(t)e^{-t^2/2c_n}\, dt\right\}. \qquad (31) \end{aligned}$$

Now let $h(\delta) = \sup_n h_n(\delta)$. As $\{d_n\}$ is bounded, Lemma A.1 ensures that $h(\delta) < 0$ strictly. Furthermore, for each $n$, $0 > h_n(\delta) \geq f_n(t) - f_n(t_n)$ whenever $|t - t_n| \geq \delta$, by Lemma A.1. Hence

$$\int_{|t-t_n|\geq\delta} e^{nf_n(t)}\,dt \leq e^{nf_n(t_n)}\left\{\sqrt{2\pi c_n\, e^{(n-1)h(\delta)+\zeta}}\right\}.$$

Hence, there exists $\gamma(\delta)$, $p(\delta) > 0$ such that

$$\frac{\int_{|t-t_n|\geq\delta} e^{nf_n(t)}\,dt}{\int_{|t-t_n|<\delta} e^{nf_n(t)}\,dt} \lesssim \gamma(\delta)\sqrt{\kappa n c_n}\, e^{-p(\delta)n} \longrightarrow 0 \qquad (n \to \infty),$$

so that equation (12) follows.

CASE 2. $d_n \to \infty$ such that $d_n = o(n)$ as $n \to \infty$.

We prove the result for a choice of margin with $\alpha = 0$ or $\beta = 0$. Fix $\lambda > 0$ and choose $\iota_n = \lambda \log c_n / c_n$, and $\delta_n = \lambda/\sqrt{128 \log c_n}$. (Note that $c_n \sim d_n$ from equation (8).) Now, from equations (23) and (26), for $|\vartheta| \leq \epsilon_n$ and for small $\lambda$, as $n \to \infty$,

$$-(f_n''(t_n) + \vartheta) = \frac{2\log c_n}{c_n}[1 \pm O(\lambda)].$$

As $n/c_n \sim n/d_n \to \infty$, $(n \to \infty)$, the first term on the right hand side of equation (29) dominates the second, so that for a sufficiently small choice of $\lambda$, equation (30) continues to hold:

$$\int_{t_n-\delta_n}^{t_n+\delta_n} e^{nf_n(t)}\,dt \sim e^{nf_n(t_n)}\left(\frac{-2\pi}{nf_n''(t_n)}\right)^{1/2} \qquad (n \to \infty). \tag{32}$$

Now by Taylor's formula we have

$$f_n(t_n \pm \delta_n) - f_n(t_n) = \frac{f_n''(t_n)\delta_n^2}{2} + \frac{1}{2}\int_{t_n}^{t_n\pm\delta_n}(t - t_n)^2 f_n'''(t)\,dt.$$

By Lemma A.2 and equations (23) and (27) we have

$$\begin{aligned}\left|\frac{1}{2}\int_{t_n}^{t_n\pm\delta_n}(t-t_n)^2 f_n'''(t)\,dt\right| \leq \frac{\delta_n^3}{2}\sup_{|t-t_n|\leq\delta_n}|f_n'''(t)| &\doteq \frac{\delta_n^3 t_n^2 e^{-t_n^2/2}}{2\sqrt{2\pi}}(1 + o(1))\\ &= \frac{\lambda^3}{1024 c_n}(1 + o(1)).\end{aligned}$$

Substituting from equation (25) we then have for a small enough choice of $\lambda$ that

$$f_n(t_n \pm \delta_n) - f_n(t_n) = -\frac{\lambda^2}{128 c_n}[1 - O(\lambda)][1 + o(1)] \qquad (n \to \infty).$$

Substituting in equation (31) we have as $n \to \infty$

$$\int_{|t-t_n|\geq\delta_n} e^{nf_n(t)}\,dt \lesssim e^{nf_n(t_n)}\left\{\sqrt{2\pi c_n}\exp\left(-\frac{\lambda^2 n}{128 c_n}[1 + o(n)][1 - O(\lambda)] + \zeta\right)\right\}. \tag{33}$$

Noting that $\zeta = \sup|f_n(t_n)|$ is finite, and that $n/c_n \sim n/d_n \to \infty$ as $n \to \infty$, equation (12) follows from equations (32) and (33) by choosing $\lambda$ suitably small.

The proof for a choice of margin $B = \beta p_n c_n^\alpha$ with $0 < \alpha < 1$ and $\beta > 0$ is similar (with equation (24) giving the asymptotic form for $t_n$ in this case). ∎

# References

[1] J. J. Hopfield, "Neural networks and physical sytems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554–2558, 1982.

[2] S. S. Venkatesh and P. Baldi, "Programmed interactions in higher-order neural networks: maximal capacity," *Journal of Complexity*, to appear.

[3] S. S. Venkatesh and P. Baldi, "Programmed interactions in higher-order neural networks: the outer-product algorithm," *Journal of Complexity*, to appear.

[4] C. M. Newman, "Memory capacity in neural network models: rigourous lower bounds," *Neural Networks*, vol. 1, no. 3 pp. 223–238, 1988.

[5] S. F. Edwards and F. Tanaka, "Analytical theory of the ground state properties of a spin glass: I. Ising spin glass," *Jnl. Phys. F*, vol. 10, pp. 2769–2778, 1980.

[6] D. J. Gross and M. Mezard, "The simplest spin glass," *Nucl. Phys.*, vol. B240, pp. 431–452, 1984.

[7] R. J. McEliece and E. C. Posner, "The number of stable points of an infinite-range spin glass memory," *JPL Telecomm. and Data Acquisition Progress Report*, vol. 42–83, pp. 209–215, 1985.

[8] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. 33, pp. 461–482, 1987.

[9] J. Komlós and R. Paturi, "Convergence results in an associative memory model," *Neural Networks*, vol. 1, no. 3, pp. 239–250, 1988.

[10] S. S. Venkatesh and D. Psaltis, "Linear and logarithmic capacities in associative neural networks," *IEEE Trans. Inform. Theory*, vol. IT-35, pp. 558–568, 1989.

[11] B. Derrida, "Random-energy model: limit of a family of disordered models," *Phys. Rev. Lett.*, vol. 45, pp. 79–82, 1980.

[12] N. G. de Bruijn, *Asymptotic Methods in Analysis.* New York: Dover, 1981.

[13] W. Feller, *An Introduction to Probability Theory and its Applications, vol. I.* New York: Wiley, 1968.

[14] I. F. Blake and H. Darabian, "Approximations for the probability in the tails of the Binomial distribution," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 426–428, 1987.

[15] P. Baldi, "Neural networks, orientations of the hypercube, and algebraic threshold functions," *IEEE Trans. Inform. Theory*, vol. 34, pp. 523–530, 1988.

[16] P. Baldi and Y. Rinott, "Asymptotic normality of some graph related statistics," *Jnl. Appl. Prob.*, vol. 26, pp. 171–175, 1989.

[17] P. Baldi, Y. Rinott, and C. Stein, "A normal approximation for the number of local maxima of a random function on a graph," in *Probability, Statistics, and Mathematics: Papers in Honor of Samuel Karlin*, (eds. T. W. Anderson, K. B. Athreya, and D. L. Iglehart). New York: Academic Press, 1989.

*Neural Networks*, Vol. 3, pp. 613-623, 1990
Printed in the USA. All rights reserved.

0893-6080/90 $3.00 + .00

*ORIGINAL CONTRIBUTION*

# Shaping Attraction Basins in Neural Networks

SANTOSH S. VENKATESH AND GIRISH PANCHA

Moore School of Electrical Engineering, University of Pennsylvania

DEMETRI PSALTIS

Department of Electrical Engineering, California Institute of Technology

AND GABRIEL SIRAT

Groupe Optice de Matericle, École Nationale Supericure de Telecommunication

(*Received* 19 *January* 1989; *revised and accepted* 22 *February* 1990)

**Abstract**—*An interesting duality between two formally related schemes for neural associative memory is exploited to shape the attraction basins of stored memories. Considered are a family of spectral algorithms—based on specifying the spectrum of the matrix of weights as a function of the memories to be stored—and a class of dual spectral algorithms—based on manipulations of the orthogonal subspace of the memories, which are expanded here. These algorithms are shown to attain near maximal memory storage capacity of the order of* **n**, *and are shown to typically require the order of $n^3$ elementary operations for their implementation. Signal-to-noise ratio arguments are presented showing a duality in the error-correction behaviour of the two schemes: the spectral algorithm demonstrates memory-specific attraction around the memories, while the dual spectral algorithm demonstrates direction-specific attraction. Composite algorithms capable of joint memory-specific and direction-specific attraction are presented as a means of variably shaping attraction basins around desired memories. Computer simulations are included in support of the analysis.*

**Keywords**—Associative memory, Network dynamics.

## 1. INTRODUCTION

In this paper we develop the duality between two methods for training a fully connected network of $n$ McCulloch–Pitts neurons (McCulloch & Pitts, 1943). The sum of outer products is perhaps the most often used training method for such networks (Nakano, 1972; Amari, 1977; Hopfield, 1982). The memory storage capacity for this method is $n/4 \log n$ (McEliece, Posner, Rodermich, & Venkatesh, 1987; Psaltis & Venkatesh, 1989) whereas the maximal theoretical capacity for any storage algorithm is $2n$ (Cover, 1965; Venkatesh, 1986b). The spectral algorithm (Kohonen, 1977; Personnaz, Guyon, & Dreyfus, 1985; Venkatesh & Psaltis, 1989) and an algorithm we will refer to as the dual spectral algorithm (Maruani, Chevallier, & Sirat, 1987) are algorithms whose capacities approach the theoretical maximum. In this paper, we briefly review these two algorithms, establish the relationship between them, and define how a proper choice of parameters specifies their error correction properties.

In such networks, memories to be stored are typically programmed as fixed points of the structure. Error correction is obtained by attracting to one of the stored fixed points, initial states (or probes) of the system that are close to the fixed points. We show that in the spectral scheme the radius of attraction around each of the stored stable states is controlled by the relative size of the eigenvalues of the interconnection matrix. The dual spectral algorithm, on the other hand, leads to a method for programming the shape of the attraction basin around each of the elements of the stored vectors. We present a new method based on linear programming for selecting the parameters of the dual spectral algorithm which determine its attraction dynamics around each stored fixed point and we suggest a hybrid algorithm that can provide more arbitrary control of the shape of the attraction basin.

We consider a fully interconnected network of $n$ McCulloch–Pitts neurons with the instantaneous bi-

Acknowledgement: The work of the first two authors was supported in part by NSF grant EET-8709198. The work at Caltech is supported by DARPA and AFOSR.

Requests for reprints should be sent to Demetri Psaltis, Department of Electrical Engineering, Caltech, MS-116-81, Pasadena, CA 91125.

nary outputs ($-1$ or 1) of each of the neurons being fed back as inputs to the network: if $u_1[t]$, $u_2[t]$, . . . , $u_n[t]$ are the outputs of each of the $n$ neurons in the network at epoch $t$, then the neural update of the $i$th neuron results in a new state at epoch $t + 1$ according to the familiar threshold rule:

$$u_i[t+1] = \Delta\left(\sum_{j=1}^{n} w_{ij}u_j[t] - w_{i0}\right),$$

where

$$\Delta(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0. \end{cases}$$

The mode of operation may be synchronous (with all the neurons being updated simultaneously at each epoch) or asynchronous (with at most one neuron being updated at each epoch). In the application of these networks to associative memory both modes of operation lead to very similar associative behaviour (cf. Psaltis & Venkatesh, 1989, for instance) and we will not make a distinction in this paper as to the precise mode of operation.

The nature of flow in state space is completely determined once the neural interconnection strengths and the mode of operation is specified. We will be interested in specifying patterns of interconnectivity for which arbitrarily prescribed $m$-sets of memories $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)} \in \mathbb{B}^n$ can be stored in the network. In order for the network to act as an associative memory, we require that the memories themselves be stable (i.e., all subsequent operations on the memory $\mathbf{u}^{(\alpha)}$ give back $\mathbf{u}^{(\alpha)}$). Stable memories are hence fixed points of the network. Furthermore, we require states close to any of the memories to be mapped into the memory by the network. This is the associative or error correcting feature requisite in an associative memory. We call the average Hamming distance from a memory over which such error correction is exhibited the *attraction radius* of the memory.

The quadratic Hamiltonian (energy) and the Manhattan form have been shown to be Lyapunov functions for fully connected networks with symmetric connections (Hopfield, 1982; Goles & Vichniac, 1986; Peretto & Niez, 1986; Psaltis & Venkatesh, 1989), hence, guaranteeing that state trajectories of such networks will terminate in stable points. If the neural interconnection weights are chosen so that the desired memories are stable, then the existence of a Lyapunov function for the system indicates that the memories will exhibit an attraction radius of error correction. The outer product and the dual spectral algorithms lead to symmetric weights but this is not generally true for the spectral scheme. Nevertheless, the spectral scheme also exhibits very similar attraction dynamics (Psaltis & Venkatesh, 1989), even though there is no known Lyapunov function for the general case. In all these algorithms stability of the stored memories can be assured with high probability if the number of memories is within the storage capacity of the algorithm (McEliece et al., 1987; Psaltis & Venkatesh, 1989). The existence of Lyapunov functions then guarantees that the memories (being fixed points) lie at the minima of the Lyapunov functions.

## 2. ALGORITHMS

### 2.1. The Spectral Algorithm

In the spectral scheme, the interconnection matrix $\mathbf{W}^s$ is defined as follows:

$$\mathbf{W}^s = \mathbf{U}\Lambda(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T, \tag{1}$$

where $\Lambda = \mathbf{dg}[\lambda^{(1)}, \ldots, \lambda^{(m)}]$ is the $m \times m$ diagonal matrix of positive eigenvalues $\lambda^{(1)}, \ldots, \lambda^{(m)} > 0$, and $\mathbf{U} = [\mathbf{u}^{(1)}\mathbf{u}^{(2)} \cdots \mathbf{u}^{(m)}]$ is the $n \times m$ matrix of memory column vectors.

We note that

$$\mathbf{W}^s\mathbf{U} = \mathbf{U}\Lambda, \tag{2}$$

where $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$ are the eigenvectors of $\mathbf{W}^s$ and $\Lambda$ is the spectrum of $\mathbf{W}^s$ (Venkatesh & Psaltis, 1985; Personnaz, Guyon, & Dreyfus, 1985; Venkatesh & Psaltis, 1989). Therefore, we are guaranteed to have *stable memories* as long as $\mathbf{W}^s$ is well defined.

For the case of an $m$-fold degenerate spectrum $\lambda^{(1)}, \ldots, \lambda^{(m)} = \lambda > 0$, we see that the matrix $\mathbf{W}^s$ is symmetric with nonnegative eigenvalues (i.e., it is nonnegative definite). Therefore there exist Lyapunov functions in this case, and moreover it has been shown that the stored memories form *global* energy minima (Venkatesh & Psaltis, 1989).

For the general spectral matrix in eqn (1), exact Lyapunov functions are hard to come by. The signal-to-noise ratio, however, serves as a good ad hoc measure of attraction capability. Consider synchronous operations with $\mathbf{W}^s$ on a state vector $\mathbf{u} = \mathbf{u}^{(\alpha)} + \delta\mathbf{u} \in \mathbb{B}^n$. We have

$$\mathbf{W}^s\mathbf{u} = \mathbf{W}^s(\mathbf{u}^{(\alpha)} + \delta\mathbf{u}) = \mathbf{W}^s\mathbf{u}^{(\alpha)} + \mathbf{W}^s\delta\mathbf{u}.$$

Once again, there exists a "signal" term, $\mathbf{W}^s\mathbf{u}^{(\alpha)}$, and a "noise" term, $\mathbf{W}^s\delta\mathbf{u}$. We anticipate that the greater the signal-to-noise ratio, the greater the attraction around $\mathbf{u}^{(\alpha)}$. Let the Hamming distance between $\mathbf{u}$ and $\mathbf{u}^{(\alpha)}$, $d_H(\mathbf{u}, \mathbf{u}^{(\alpha)})$, equal $d$ (i.e., $\|\delta\mathbf{u}\| = 2\sqrt{d}$). The (strong) norm of the matrix $\mathbf{W}^s$ is defined as

$$\|\mathbf{W}^s\| = \sup_{\mathbf{x}} \frac{\|\mathbf{W}^s\mathbf{x}\|}{\|\mathbf{x}\|}, \quad \|\mathbf{x}\| \neq 0.$$

It follows (cf. Strang, 1980) that $\|\mathbf{W}^s\| = \sqrt{k}$, where $k$ is the largest eigenvalue of the matrix $(\mathbf{W}^s)^T\mathbf{W}^s$. For the case of the degenerate spectrum $\lambda^{(1)}, \ldots, \lambda^{(m)} = \lambda > 10$, $\mathbf{W}^s$ is symmetric, and $(\mathbf{W}^s)^T\mathbf{W}^s = (\mathbf{W}^s)^2$. Therefore, the maximum eigenvalue of

$(\mathbf{W}^s)^T\mathbf{W}^s = k = \lambda^2$, and the signal-to-noise ratio (SNR) is given by

$$\text{SNR} = \frac{\|\mathbf{W}^s\mathbf{u}^{(\alpha)}\|}{\|\mathbf{W}^s\delta\mathbf{u}\|} \geq \frac{\lambda^{(\alpha)}\sqrt{n}}{(\sqrt{k})(2\sqrt{d})} = \frac{1}{2}\sqrt{\frac{n}{d}}.$$

Thus, we would expect the attraction sphere around $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$ to increase as $n$ increases for the $m$-fold degenerate spectral scheme. For the general nondegenerate case, we expect that by varying the size of $\lambda^{(\alpha)}$, the SNR, and hence the attraction capability, be proportionately increased or decreased for the $\alpha$th memory $\mathbf{u}^{(\alpha)}$ (Figure 1).

Using a result of Komlós (1967) we can show that for all randomly chosen $n$-tuples $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)} \in \mathbf{B}^n$, and $m \leq n$, the probability that $\mathbf{W}^s$ is well defined approaches one as $n \to \infty$. It immediately follows that the static capacity of the spectral scheme is $n$, as a linear transformation has at most $n$ eigenvalues.

Let $N^s$ denote the number of elementary operations required to compute the weight matrix $\mathbf{W}^s$ directly from the $m$ memories to be stored. Then using the fact that $(\mathbf{U}^T\mathbf{U})^{-1}$ is symmetric, we can use the Cholesky decomposition to compute its inverse. This along with the rest of the matrix multiplications gives us that $N^s = mn^2 + m^2n + (m^3)/2 + O(n^2)$ (details can be found in Venkatesh and Psaltis (1989)).

### 2.2. Dual Spectral Algorithms

2.2.1. *Orthogonal Spaces and Duality*. The following scheme, formally related to the outer product and spectral algorithms, was introduced by Maruani et al. (1987).

Let $\mathbf{U} = [\mathbf{u}^{(1)}\mathbf{u}^{(2)} \cdots \mathbf{u}^{(m)}]$ be the matrix of memories as before. Let $\mathbf{x}^{(\beta)}$, $\beta = 1, \ldots, n - m$, be a set of linearly independent vectors in $\mathbf{R}^n$ which are individually orthogonal to each of the memories (i.e., $\mathbf{X}^T\mathbf{U} = \mathbf{0}$, where we define the $n \times (n - m)$ matrix $\mathbf{X} = [\mathbf{x}^{(1)}\mathbf{x}^{(2)} \cdots \mathbf{x}^{(n-m)}]$). Define a weight matrix $\mathbf{W}$ with weights $w_{ij}$ given by

$$w_{ij} = \begin{cases} -\sum_{\beta=1}^{n-m} x_{i\beta}x_{j\beta} & \text{if } i \neq j \\ 0 & \text{if } i = j, \end{cases}$$

where $x_{k\beta}$ is the $k$th component of $\mathbf{x}^{(\beta)}$. If we define $\hat{\mu}_i = \sum_{\beta=1}^{n-m} x_{i\beta}^2$, $i = 1, \ldots, n$, we see that

$$\mathbf{W} = \hat{\mathbf{M}} - \mathbf{X}\mathbf{X}^T, \tag{3}$$

where $\hat{\mathbf{M}} = \mathbf{dg}[\hat{\mu}_1, \ldots, \hat{\mu}_n]$. Thus,

$$\begin{aligned} \mathbf{W}\mathbf{U} &= \hat{\mathbf{M}}\mathbf{U} - \mathbf{X}\mathbf{X}^T\mathbf{U} \\ &= \hat{\mathbf{M}}\mathbf{U}. \end{aligned} \tag{4}$$

Comparing eqns (2) and (4) we see that the spectral and dual spectral algorithms exhibit an interesting duality. Since the parameters $\hat{\mu}_i$ are positive for each choice of $i$, it follows that

$$\Delta(\mathbf{W}\mathbf{u}^\alpha)_i = \Delta(\hat{\mu}_i u_i^\alpha) = u_i^\alpha,$$
$$\text{for each } i = 1, \ldots, n, \ \alpha = 1, \ldots, m.$$

So the memories $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$ are fixed points in the scheme as well.

$\mathbf{W}$ as defined in eqn (3) is a zero-diagonal symmetric matrix. Thus, we know that there exists some form of attraction behaviour. However, since the orthogonal basis $\mathbf{X}$ has been chosen arbitrarily, there is some lack of control in specifying attraction capability. Specifically, as we shall argue below, the $\hat{\mu}_i$'s essentially control directional attraction and we have no means of specifying these under the above

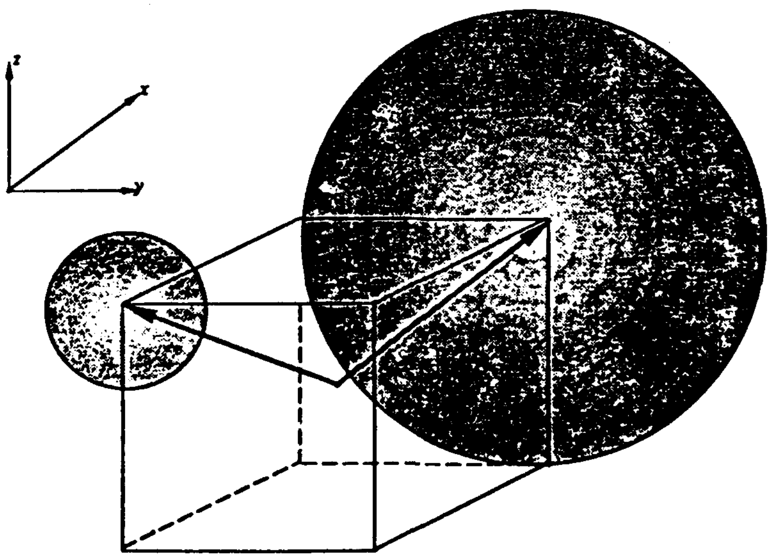


**FIGURE 1. Schematic representation of the attraction space in the spectral scheme for memories with different eigenvalues.**

approach. Our goal here will be to specify an algorithm where such control is possible.

2.2.2. *The Effect of the $\mu$-values.* In the spectral scheme, the eigenvectors of $\mathbf{W}^s$ are the memories, so that the column space of $\mathbf{W}^s$ is given by the span of the memories. Therefore, if the memories are far enough from each other and the initial state vector $\mathbf{u}$ is close enough to a memory, $\mathbf{W}^s$ combined with the thresholding operation *projects* $\mathbf{u}$ onto the memory.

On the other hand, in the dual spectral scheme, the weight matrix $\mathbf{W}^d$ is obtained by taking the correlation of vectors that are orthogonal to the memories and then setting the diagonal elements to be 0. In creating the zero diagonal, we essentially add perturbations to the left nullspace of $\mathbf{U}$ in the directions of the memories. The *strength* of the perturbations along any component $i$ is proportional to $\hat{\mu}_i$. Thus, each of the $\hat{\mu}_i$'s corresponds to a directional distortion, and we expect the SNR of the dual spectral scheme to vary from direction to direction proportionately with the value of $\hat{\mu}_i$. We therefore expect that the larger the $\hat{\mu}_i$, more information is lost if the $i$th bit is flipped and, hence, the smaller the attraction would be in the $i$th direction.

As an illustration, let us consider the case where $n = 3$, and $\mu_x \ll \mu_y, \mu_z$ (Figure 2). Each memory $\mathbf{u}$ would be preferentially attracted in the $dx$-direction, indicated schematically by an attraction cone in Figure 2 (i.e., a vector with a different $x$ component will probably map back to $\mathbf{u}$ but vectors with different $y$ and $z$ components will probably not be within the attraction region of $\mathbf{u}$). In other words,

$$\mathbf{P}\left[\begin{pmatrix} -u_x \\ u_y \\ u_z \end{pmatrix} \longrightarrow \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\right] > \mathbf{P}\left[\begin{pmatrix} u_x \\ u_y \\ -u_z \end{pmatrix} \longrightarrow \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\right],$$

$$\mathbf{P}\left[\begin{pmatrix} u_x \\ -u_y \\ u_z \end{pmatrix} \longrightarrow \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\right].$$

2.2.3. *Specifying Directional Attraction With Linear Programming.* The previous section's discussions point to a necessity of somehow specifying the $\mu$-values if we require direction-specific attraction. Specifically, for a *prescribed set* $\mu_1, \ldots, \mu_n > 0$ of directional attraction strengths, and $\mathbf{M} = \mathbf{dg}[\mu_1, \ldots, \mu_n]$, we require a weight matrix $\mathbf{W}^d$ such that

$$\mathbf{W}^d\mathbf{U} = \mathbf{MU}. \tag{5}$$

We define $\mathbf{W}^d$ such that:

$$w_{ij}^d = \begin{cases} -\sum_{\beta=1}^{n-m} (x_{i\beta}b_\beta)(x_{j\beta}b_\beta) & \text{if } i \neq j \\ 0 & \text{if } i = j, \end{cases} \tag{6}$$

where $x_{i\beta}$ is the $i$th component of the basis vector $\mathbf{x}^{(\beta)}$ as defined earlier, and $b_\beta$ is the $\beta$th component of a vector which we will specify shortly. Thus, given $\mu_1, \ldots, \mu_n$ we need to find a vector $\mathbf{b}$ such that with $\mathbf{Y} = \mathbf{Xb}$

$$\mathbf{W}^d = \mathbf{M} - \mathbf{YY}^T. \tag{7}$$

(Note that the columns of $\mathbf{Y}$, in general, are not orthogonal.)

Assuming that $\mathbf{W}^d$ has the form given in eqn (6), let us now consider the effect of $\mathbf{W}^d$ on the $i$th ele-

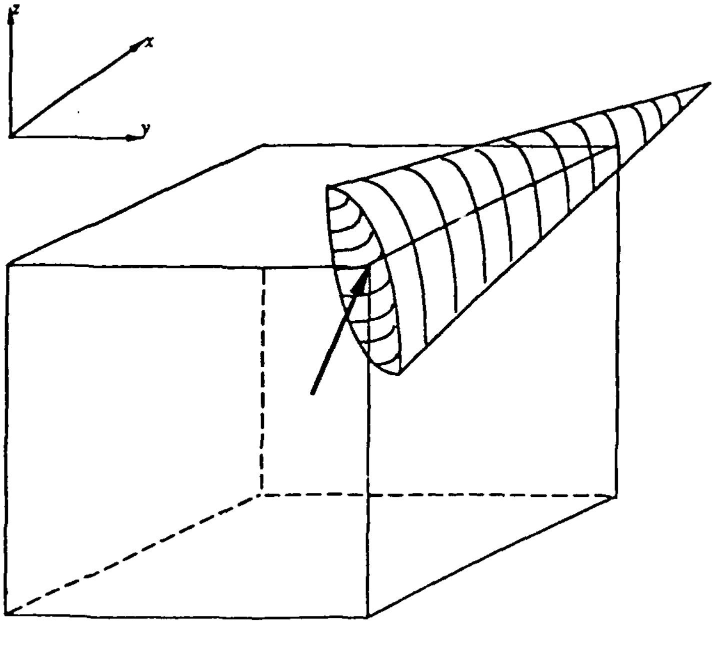


FIGURE 2. Schematic representation of the directional attraction space in the dual spectral scheme for a choice of $\mu_x \ll \mu_y, \mu_z$.

ment of a memory $\mathbf{u}^{(\alpha)}$:

$$
\begin{aligned}
[\mathbf{W}^d\mathbf{u}^{(\alpha)}]_i &= \sum_{j=1}^{n} w^d_{ij}u^{(\alpha)}_j \\
&= -\sum_{j\neq i}\sum_{\beta=1}^{n-m} b^2_\beta x_{i\beta}x_{j\beta}u^{(\alpha)}_j \\
&= \sum_{j=1}^{n}\sum_{\beta=1}^{n-m} b^2_\beta x_{i\beta}x_{j\beta}u^{(\alpha)} + \sum_{\beta=1}^{n-m} b^2_\beta x^2_{i\beta}u^{(\alpha)}_i \\
&= \sum_{\beta=1}^{n-m} b^2_\beta x^2_{i\beta}u^{(\alpha)}_i .
\end{aligned}
$$

We require from eqn (5) that

$$[\mathbf{W}^d\mathbf{u}^{(\alpha)}]_i = \mu_i u^{(\alpha)}_i.$$

where $\mu_i > 0$. By inspection, we obtain the relationship

$$\mu_i = \sum_{\beta=1}^{n-m} x^2_{i\beta}b^2_\beta .$$

Define $a_{i\beta} = x^2_{i\beta}$, and $c_\beta = b^2_\beta$. Then we require

$$\mathbf{Ac} = \mathbf{M}_\mu ,$$

where $\mathbf{A}$ is a known $n \times (n - m)$ matrix with nonnegative elements $a_{i\beta} = x^2_{i\beta}$, $\mathbf{c}$ is an unknown $(n - m)$-dimensional vector with $c_\beta = b^2_\beta$ constrained to be nonnegative, and $\mathbf{M}_\mu$ is a specified $n$-dimensional vector with positive components $\mu_1, \ldots, \mu_n$.

We notice that this is an overspecified system of $n$ equations with $(n - m)$ unknowns, where both $\mathbf{c}$ and $\mathbf{M}_\mu$ are constrained to have nonnegative elements. Linear programming techniques can be used to solve this system of equations. We can choose the $\mu$-values in a variety of ways. Two representative methods are suggested here.

*Specifying* $\mu_1, \ldots, \mu_k$, $k \leq n - m$. The canonical form of the linear programming problem that the simplex method solves is:

Minimize the *goal function* $\mathbf{c}^T\mathbf{y}$ subject to the constraints

$$\mathbf{Ay} = \mathbf{b},$$

where the vector $\mathbf{y}$ is unknown, and $\mathbf{y} > \mathbf{0}$.

In this case, we specify $k$ positive values of $\mathbf{M}_\mu$ and minimize the maximum of the $(n - k)$ unspecified values of $\mathbf{M}_\mu$ subject to the constraints $\mu_{k+1}, \ldots, \mu_n > 0$, and $c_1, \ldots, c_{n-m} > 0$. In other words, we have the following equations

$$
\begin{aligned}
a_{1,1}c_1 &+ \cdots + a_{1,n-m}c_{n-m} &&= \mu_1 \\
&\vdots \\
a_{k,1}c_1 &+ \cdots + a_{k,n-m}c_{n-m} &&= \mu_k \\
a_{k+1,1}c_1 &+ \cdots + a_{k+1,n-m}c_{n-m} &&\leq \varepsilon \\
&\vdots \\
a_{n,1}c_1 &+ \cdots + a_{n,n-m}c_{n-m} &&\leq \varepsilon,
\end{aligned}
$$

where $c_i \geq 0$, $\varepsilon > 0$, and we want to find $\mathbf{c}$ which minimises $\varepsilon$.

To convert the $n - k$ inequalities to equalities, we subtract $\varepsilon$ from both sides of the equation and add *slack variables* $z_1, \ldots, z_{n-k}$ to give us the following $n - k$ equations

$$
\begin{aligned}
a_{k+1,1}c_1 + \cdots + a_{k+1,n-m}c_{n-m} - \varepsilon + z_1 &= 0 \\
&\vdots \\
a_{n,1}c_1 + \cdots + a_{n,n-m}c_{n-m} - \varepsilon + z_{n-k} &= 0,
\end{aligned}
$$

in addition to the first $k$ equations. Now we have $n$ equations with $2n - m - k$ unknown nonnegative quantities $(c_1, \ldots, c_{n-m}, z_1, \ldots, z_{n-k})$.

Let us label $\varepsilon$ as $c_0$. By inspection, we see that the goal function to be minimised is $c_0$, subject to the constraints $\mathbf{A}'\mathbf{c}' = \mathbf{M}'_\mu$, where $\mathbf{c}'$ is a $(2n - m - k + 1)$-dimensional vector $\mathbf{M}'_\mu$ is a $n$-dimensional vector, and $\mathbf{A}'$ is an $n$ by $2n - m - k + 1$ matrix; that is, we require to solve

$$
\begin{pmatrix}
0 & & 0 & & \\
\vdots & & & \ddots & \\
0 & \mathbf{A} & & & 0 \\
-1 & & 1 & & \\
\vdots & & & \ddots & \\
-1 & & & & 1
\end{pmatrix}
\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-m} \\ z_1 \\ \vdots \\ z_{n-k} \end{pmatrix}
=
\begin{pmatrix} \mu_1 \\ \vdots \\ \mu_k \\ 0 \\ \vdots \\ 0 \end{pmatrix}
\tag{8}
$$

and $c_i$, $z_i \geq 0$. This is in the canonical form for the simplex method.

*Specifying* $\mu_1, \ldots, \mu_n$. In this case, we specify all the values of $\mathbf{M}_\mu$. We indicate two possible options when solving for $\mathbf{c}$.

1. Minimise the mean-square error given by

   $$\|\mathbf{Ac} - \mathbf{M}_\mu\|^2 = \sum_{i=1}^{n}(a_{i,1}c_1 + \cdots + a_{i,n-m}c_{n-m} - \mu_i)^2$$

   subject to the constraints $\mathbf{M}_\mu > 0$, $\mathbf{c} > \mathbf{0}$.

   This is a quadratic programming problem. However, this problem can be reformulated as a simplex method problem and can be solved using a variation of the traditional simplex method called Wolfe's method (Wolfe, 1959).
2. Minimise the largest absolute error $c_0$, given by

   $$\max(|\varepsilon_1|, \ldots, |\varepsilon_n|)$$

   where $\varepsilon_i$, the error in $\mu_i$, is

   $$\mu_i - (a_{i,1}c_1 + \cdots + a_{i,n-m}c_{n-m}), \quad i = 1, \ldots, n.$$

   Our problem now is to minimize $c_0$ subject to $c_0$, $c_1, \ldots, c_{n-m} \geq 0$. To solve this problem,[1] we

[1] This is known as Chebyshev's Approximation (Franklin, 1980, p. 8).

note that we have $n$ pairs of inequality contraints of the form

$$-c_0 + a_{i,1}c_1 + \cdots + a_{i,n-m}c_{n-m} \leq \mu_i,$$

$$-c_0 - a_{i,1}c_1 - \cdots - a_{i,n-m}c_{n-m} \leq -\mu_i.$$

The addition of slack variables puts the problem in canonical form.

2.2.4. *Characterisation of the Dual Spectral Scheme.* For simplicity, we consider algorithms employing the first linear programming approach outlined above. We have modified the initial basis for the nullspace of **U** using the results of the simplex method such that

$$\mathbf{W}^d = \mathbf{M} - \mathbf{Y}\mathbf{Y}^T,$$

where $\mathbf{M} = \mathbf{dg}[\mu_1, \mu_2, \ldots, \mu_n]$ with $\mu_1, \ldots, \mu_k > 0$ specified by us, and $0 < \mu_{k+1}, \ldots, \mu_n \leq \varepsilon < \min(\mu_1, \ldots, \mu_k)$, and $\mathbf{Y} = \mathbf{Xb}$ is a set of basis vectors for the left nullspace of **U**. Since $\mu_i$, $i = 1, \ldots, n$, are positive, we see that all the memories are strictly stable in the dual spectral scheme as long as the memories $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$ are linearly independent, and we are able to find the vector **c** in the system (8) through linear programming.

As asserted earlier, since $\mathbf{W}^d$ is a symmetric, zero-diagonal matrix, there exist Lyapunov functions for this scheme in both modes of operation. We have also conjectured that the attraction is directional in nature. The storage capacity of the dual spectral scheme of eqn (6) is directly $n - 1$. Specifically $n - 1$ is the number of memories for which we can still specify a left nullspace **X**. (By Komlós' result (Komlós, 1967), we are guaranteed that almost all choices of $n$ memories or fewer are linearly independent, so that for almost all choices of $n - 1$ memories there is an orthogonal subspace of dimension 1, while almost all choices of $n$ memories span the space $\mathbf{R}^n$ and therefore the orthogonal subspace is of dimension 0.)

To find an $n$-dimensional vector under constraints, the simplex method iterates from one feasible solution to another until it finds an optimal feasible solution. The maximum number of iterations that the simplex method can go through to find an $n$-dimensional vector is $2^n - 1$.[2] However, it has been widely reported (Chvátal, 1983; Murty, 1983) that, in practice, the number of iterations is almost always between 1 to 3 times the number of constraints. Thus, for the case of specifying $k$ values of $\mathbf{M}_\mu$, we would expect at the most $3n$ iterations. The computational complexity of each iteration is dependant on how the simplex method is implemented. For the revised simplex method, a good estimate of the average cost of each iteration in our scheme is $52n - 10m - 10k + 10$, while for the standard simplex method, a good estimate is $(2n^2 - mn - kn + n)/4$ (cf. Chvátal, 1983, p. 113). Thus, we estimate that the total cost of specifying $k$ values of $\mathbf{M}_\mu$ is $O(n^2)$ (using the revised simplex method). The cost of finding a basis for the nullspace of **U** (through Gram–Schmidt orthogonalisation) includes finding $(\mathbf{U}^T\mathbf{U})^{-1}$ and two other matrix multiplications and is given by $mn^2 + (m^2n)/2 - m^3/2 + O(n^2)$. Finally, the cost of finding $\mathbf{W}^d$ from **c** and **X** is $n^3 - n^2m + O(n^2)$. So, we can say that on the average,

$$N^d = n^3 + \tfrac{1}{2}m^2n - mn^2 - m^3/2 + O(n^2),$$

where $N^d$ is the number of elementary operations needed to compute $\mathbf{W}^d$.

There are a number of open questions involved with the dual spectral scheme arising from the nature of the construction of the $\mathbf{W}^d$ matrix. The number of directions $k$, that can be specified given a set of $m$ memories and $n$ neurons is of interest. It is obvious from the previous discussion about the dimensions of **A** and **c**, that we can surely specify no more than $n - m$ directions. However, there is a possibility (albeit small) that there exist no feasible solutions for pathological cases where $k < n - m$. This is seen particularly when the number $n - m$ is very small. Another quantity we are interested in is the size of $\varepsilon$, the largest of the unspecified $\mu$'s, compared to the size of the specified $\mu$'s since we have conjectured that this will affect directional attraction.

While there exists little theory for the simplex method which will enable us to gauge these parameters, simulations show that $\varepsilon$ is typically small compared to $\mu_i$ for the specified directions ($<0.5\mu_i$), and $k$ is typically of the order of $n/4$ in the ranges simulated. We conjecture that this behaviour continues to hold for large $n$.

## 2.3. Composite Algorithms

In section 2.1 we saw ways of increasing the radii of attraction-spheres around memories. In section 2.2 we say ways of specifying increased attraction in certain directions around each of the memories. A natural extension of these schemes is to create a composite scheme with weight matrix $\mathbf{W}^c$ given by

$$\mathbf{W}^c = \mathbf{W}^s + \mathbf{W}^d.$$

Since $\mathbf{W}^c$ is a linear combination of $\mathbf{W}^s$ and $\mathbf{W}^d$, we would expect memories to be stable in the composite scheme for reasons described in the previous sections. The idea of the composite scheme is to specify both memory-specific attraction by specifying $\lambda$ for each memory, and direction-specific attraction by specifying $\mu$ for the individual directions (Figure 3).

[2] This happens when the simplex method tests each vertex of the $n$-sided polyhedron that bounds the feasible region.

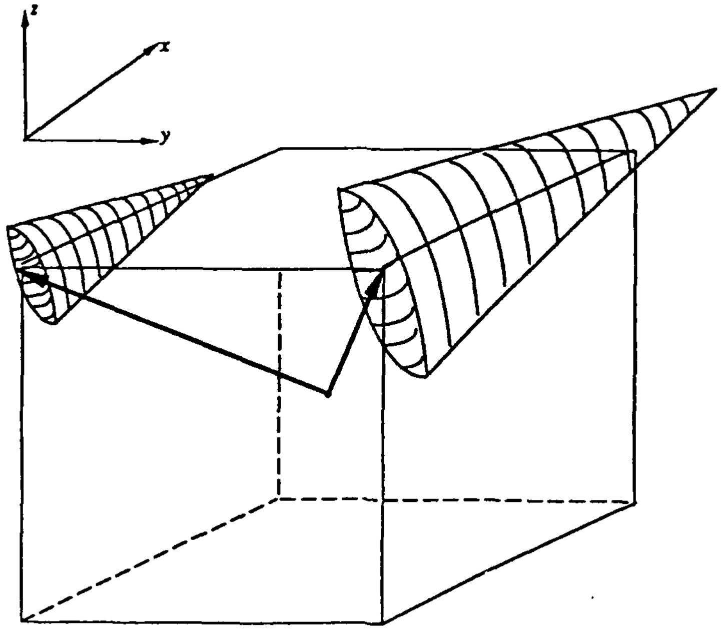


FIGURE 3. Schematic representation of the joint memory-specific and direction-specific attraction space for two memories in the composite scheme.

Here, the spectrum of $\mathbf{W}^s$ is no longer degenerate, and $\mathbf{W}^c$, consequently, is no longer symmetric. As the composite algorithm combines the memory-specific spectral algorithm, and the direction-specific dual spectral algorithm, it works effectively in shaping the attraction regions as desired. It should be noted that the relative values of the $\lambda^{(1)}, \ldots, \lambda^{(m)}$, compared to the $\mu_1, \ldots, \mu_n$, need to be considered in order not to lose the effects of one of the two parts of the composite scheme.

Note that the capacity of the composite scheme is $n - 1$. The algorithm complexity of the composite scheme is the sum of the complexities of the spectral and dual spectral schemes, except that we need not find $(\mathbf{U}^T\mathbf{U})^{-1}$ twice. Therefore the complexity $N^c$ is given by $3n^3 + O(n^2)$ for $m \leq n$.

## 3. SIMULATIONS

Computer simulations were carried out to verify the behaviour of the various schemes. Systems with state vectors of 32 bits were considered in the simulations. The memories were chosen randomly with a binomial pseudo-random number generator with equiprobable values 1 and $-1$. For each size of memory set $m$ that was investigated, simulations were carried out for each of the schemes, and the behaviour of the schemes was averaged out over between 20 and 100 trials, where over each trial a different random set of memories was generated. Error correction data were compiled at each trial by testing the convergence of randomly generated probes at increasing Hamming distance from a memory. Attraction radii were estimated by averaging the maximum error correction radius for each trial over the number of trials. The graphs included here were obtained from synchronous mode operations. However, we found that the schemes essentially behaved the same under an asynchronous mode of operation. The graphs show typical stability and attraction behaviour in each of the schemes. We can extract information on expected worst and best case behaviour for a set of random memories from these curves.

The behaviour of the outer product scheme is highlighted in Figures 4 and 5. As anticipated, the

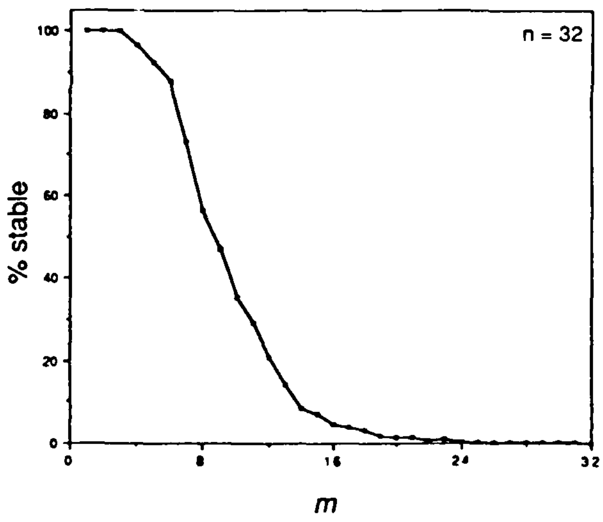


FIGURE 4. The percentage of stable memories plotted against the number of memories $m$ in the outer product scheme when $n = 32$.

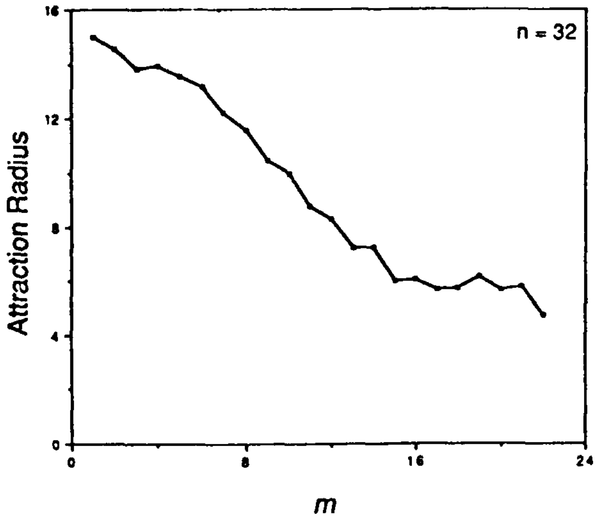


**FIGURE 5. The average radius of attraction around a stable memory is plotted versus the number of memories for $n = 32$ in the outer-product scheme. The attraction radius is estimated by averaging the maximum Hamming distance of error-correction around a stable memory over several independent runs.**

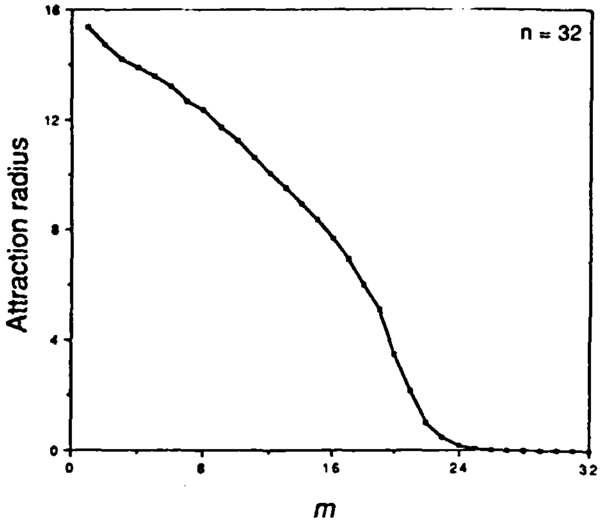


**FIGURE 6. The attraction radius around a typical memory plotted as a function of the number of memories $m$, in the degenerate spectral scheme where all the eigenvalues are chosen equal to $\lambda = n = 32$. Estimates of the attraction radius for a given number of memories were again obtained by averaging the maximum distance of error-correction around a memory over several independent runs.**

number of stable memories declines precipitously as $m$ increases beyond a certain point (the static capacity) as seen in Figure 4. While $n$ is quite small in these examples, the figures nonetheless are a precursor of the 0–1 behaviour which develops around the static capacity of $n/(4 \log n)$ for large $n$ (Venkatesh, 1986; McEliece et al., 1987; Komlós & Paturi, 1988). Figure 5 shows the graceful degradation of the average Hamming radius of attraction around the memories as the number of stored memories increases. (We averaged the maximum attraction radius for each of the memories over several independent trials to obtain estimates of the average radius of attraction.) The analysis in McEliece et al. (1987) indicates that the attraction is neither memory- nor direction-specific, and that we obtain uniform Hamming balls of attraction around each memory with high probability for large $n$.

Simulations highlighting the behaviour of the spectral scheme as a viable algorithm for associative memory are presented in Figures 6 and 7. The average Hamming radius of attraction again degrades gracefully as the number of memories increases, as illustrated in Figure 6, where the degenerate spectral algorithm exhibits uniform balls of attraction around the memories. (The static capacity here is clearly $n$ as outlined before and verified in our simulation.) As can be seen, the dynamical behaviour of the spectral scheme is qualitatively similar to the outer product scheme, but somewhat better over all ranges.

Investigations into attraction dynamics in the spectral scheme when there is a large deviation in eigenvalue size confirm theoretical predictions that the sizes of the attraction basins are memory-specific and increase with increase in the eigenvalue size of the corresponding memory. These trends are exemplified in the typical plot of Figure 7 where half the eigenvalues are fixed arbitrarily at $n$, and the other half of the eigenvalues are fixed at a fraction of $n$. The plots show the relative sizes of the Hamming balls of attraction for memories with large eigenvalue as compared to memories with small eigenvalue, as a function of the ratio of the two eigenvalues. The results are similar for other values of $m$ in the range of interest (i.e., values of $m$ for which there is significant attraction: the attraction radii around the memories is proportional to corresponding eigenvalue size).

The feasibility of forming the dual spectral matrix $\mathbf{W}^d$, using the simplex method when $\mu_1, \ldots, \mu_k$ are specified is confirmed in Figures 8 and 9. The success rate (the percentage of trials when the simplex method returns a feasible solution with $\varepsilon < \min(\mu_1, \ldots, \mu_n)$) is plotted in Figure 8 against the number of memories $m$, averaged over various choices of $k$. In Figure 9, the success rate is plotted as a function of the number of specified directions $k$, with $m$ as a parameter. Note that the success rate is almost 100% when $k$ is small, and drops gradually with failures occurring most often when $k$ approaches $n - m$ (Figure 9). Figure 10 exhibits plots of average $\varepsilon$ versus $k$ for various $m$. As can be seen, $\varepsilon$ increases with increasing $k$ and increasing $m$. Exhaustive simulations indicate that the values of $\varepsilon$ obtained by the simplex algorithm for $n = 16$ (fixed $m$, $k$) are approximately twice those for $n = 32$. Since the

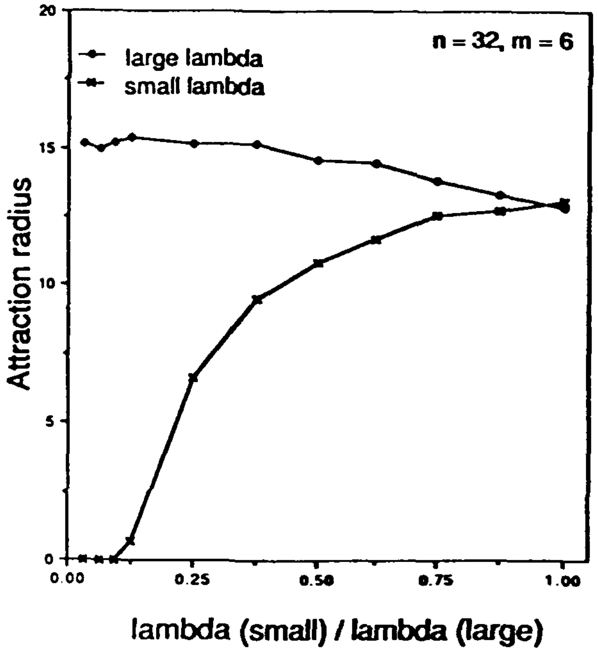


**FIGURE 7.** Demonstration of memory-specific attraction in the spectral scheme for $n = 32$ and $m = 6$. The memories were divided into two equal sized groups, one group with eigenvalue $\lambda$(large) $= n$, and the other group with eigenvalue $\lambda$(small) varying as a fraction of $n$. The respective attraction radii of the $\lambda$(large) memories and the $\lambda$(small) memories are plotted as the ratio $\lambda$(small)/$\lambda$(large) is increased from zero to one.

dynamic attraction behaviour of the dual spectral scheme is dependent on the size of $\varepsilon$, these curves are crude indicators of the limits on $m$ and $k$ in the dual spectral scheme.

Investigations into the attraction dynamics of the dual spectral scheme verify the analytical predictions of its performance as an associative memory. We will use the measure of attraction in a particular direction $x$ for a particular memory to be the average Hamming radius from which state vectors converge to that memory when bit $x$ is kept flipped. (*Specifically, if $mu_x$* is large, then inputs with bit $x$ opposite in sign to a memory will be unlikely to converge to the memory, and conversely if $\mu_x$ is small. Equivalently, if bit

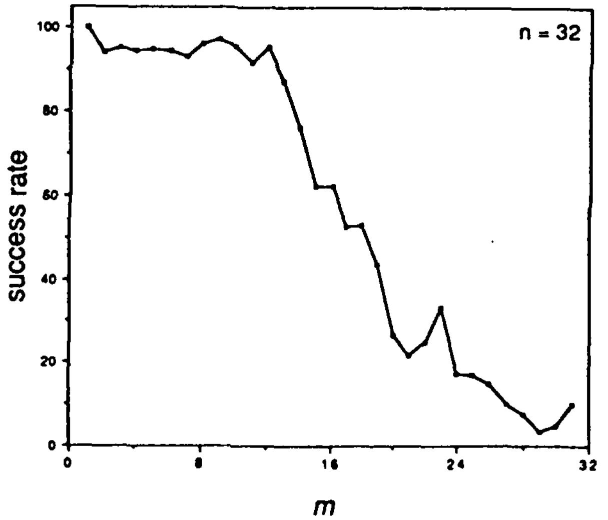


**FIGURE 8.** The percentage of trials when the simplex method returns a feasible solution (the success rate) in forming the dual spectral matrix $W^d$, averaged over various choices of $k$, the number of specified directional values, $\mu_1, \ldots, \mu_k$, plotted as a function of the number of memories $m$, when $n = 32$.

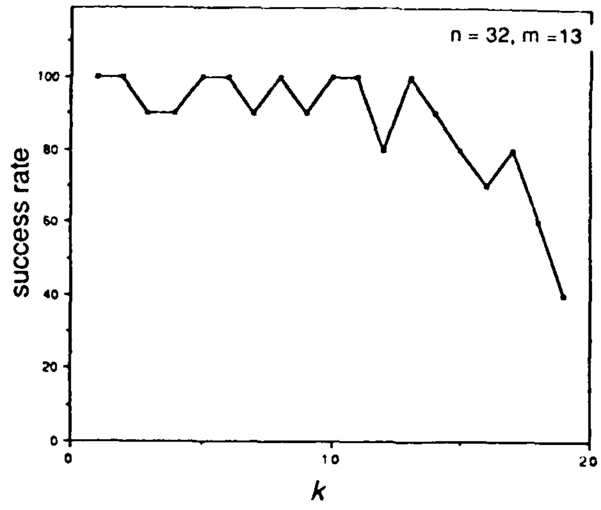


**FIGURE 9.** The percentage of trials when the simplex method returns a feasible solution for the dual spectral scheme (the success rate) plotted as a function of the number of specified directions $k$, for a choice of $n = 32$, $m = 13$. (Here $k$ denotes the number of directional values, $\mu_1, \ldots, \mu_k$, specified in the algorithm.)

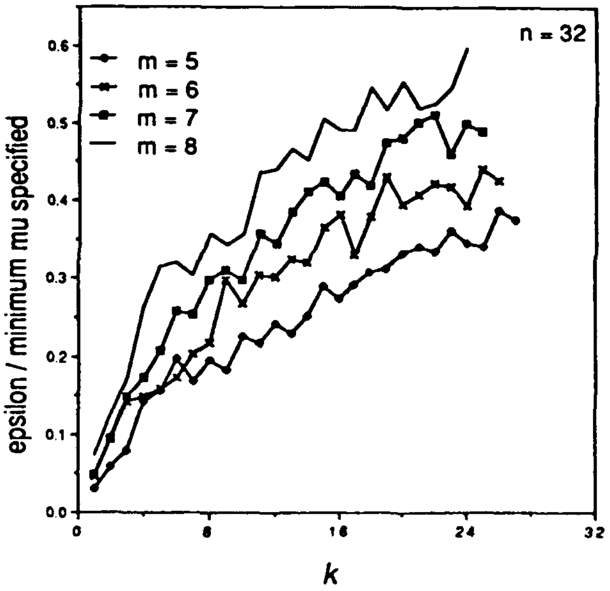


**FIGURE 10.** The ratio of the largest value $\epsilon$, of the unspecified directional parameters, $\mu_{k+1}, \ldots, \mu_n$, to the smallest of the specified directional parameters $\mu_{\min} = \min\{\mu_1, \ldots, \mu_k\}$, plotted versus $k$, the number of specified directional parameters with $m$ as a parameter for $n = 32$.

$x$ of the input vector is constrained to be correctly matched to the corresponding bit of the memory, then the algorithm will tend to correct for rather large distortions in the other components if $\mu_x$ is large, and conversely if $\mu_x$ is small.) Figure 11 exhibits plots of average attraction in both the specified (important) and the unspecified (unimportant) directions, where the component of the input in the direction being investigated was initially kept flipped. Here, the attraction characteristics have been averaged over all the memories for the two cases: (1) the specified directions (corresponding to large values of $\mu$), and (2) the unspecified directions (corresponding to small values of $\mu$). As can be seen, there exists a consistent difference in attraction in the large $\mu$ and small $\mu$ directions when $k$ is small, with a merging of the attraction capabilities for larger $k$.

The simulations indicate that we do have the capability of separately achieving memory-specific and direction-specific attraction. Investigations into the composite scheme indicate that attraction basins can indeed be shaped over a wide range. Varying the values of the specified $\mu_i$'s, and large eigenvalues ($\lambda_{lg}$) and small eigenvalues ($\lambda_{sm}$) lead to attraction basins that range from being purely directional to completely "spherical" around memories. A sample case where $\lambda_{sm} = 1$, $\lambda_{lg} = 3$, and $\mu_{lg} = 6$ (which gives us $\varepsilon \leq 3$ for moderate values of $k$ and $m$) is shown in Figures 12 and 13. Figure 12 exhibits plots of mem-

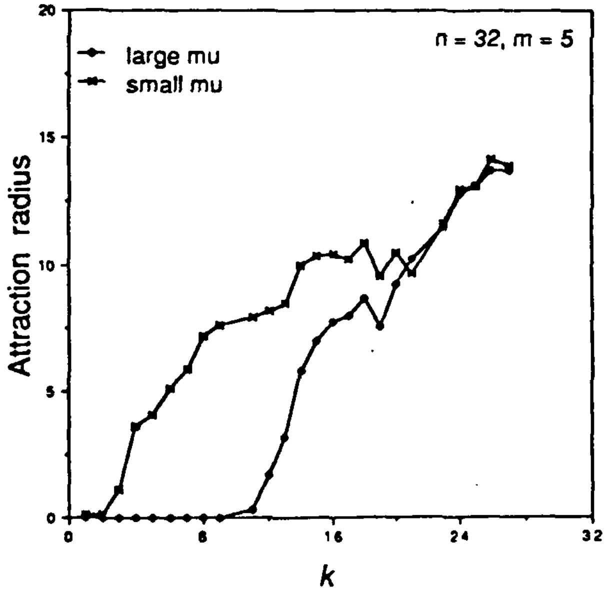


FIGURE 11. Demonstration of direction-specific attraction in the dual spectral scheme for $n = 32$ and $m = 5$. Curves of attraction radii versus the number of specified directional parameters $k$, are shown for two different directions—a specified (large $\mu$) direction and an unspecified (small $\mu$) direction. Attraction data for a given direction were generated by investigating probe vectors at various Hamming distances from a memory with the component of the probe in the direction being investigated being chosen to be opposite in sign to the corresponding component of the memory. (Flipping a bit in an important (large $\mu$) direction would reduce the attraction to the memory compared to an unimportant (small $\mu$) direction.)

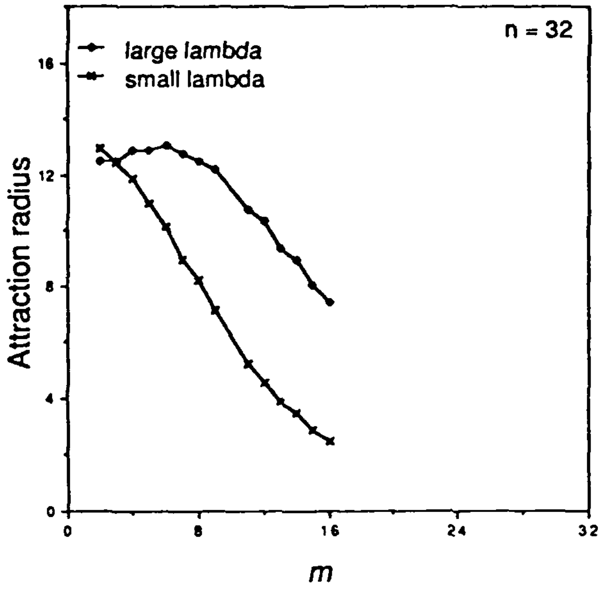


FIGURE 12. Demonstration of memory-specific attraction in the composite scheme for $n = 32$. The memories are divided into two groups, one group corresponding to a "large" eigenvalue $\lambda_{lg} = 3$, and the other group corresponding to a "small" eigenvalue $\lambda_{sm} = 1$. Attraction radii for a memory are plotted as a function of the number of memories $m$ for the two cases of the memory corresponding to eigenvalues $\lambda_{lg} = 3$ and $\lambda_{sm} = 1$.

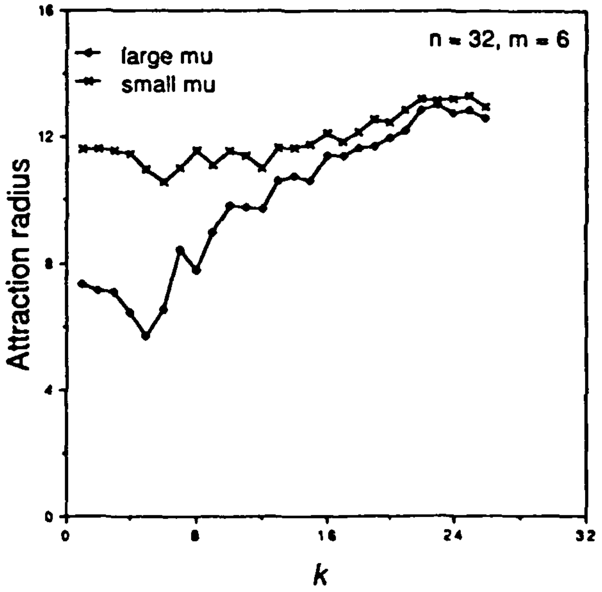


FIGURE 13. Demonstration of direction-specific attraction in the composite scheme for $n = 32$ and $m = 6$. Directional attraction parameters are specified to be all equal to $\mu_{lg} = 6$, while the largest of the unspecified directional parameters is kept below $\epsilon = 3$. Attraction radii are plotted in the large $\mu$ (specified) and small $\mu$ (unspecified) directions as a function of $k$, the number of specified directions.

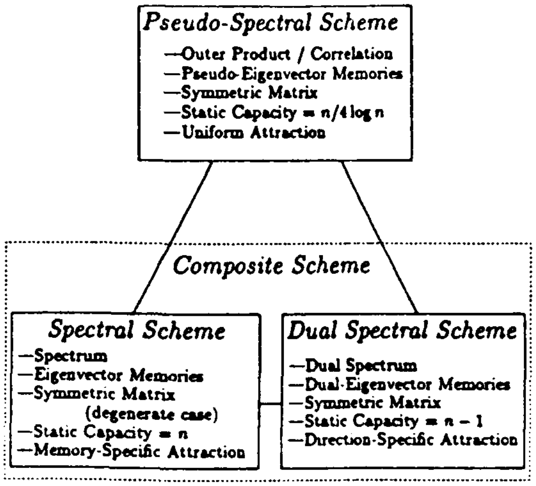


**FIGURE 14. An overview of the features of the family of spectral algorithms—the outer product (pseudo-spectral) algorithm, the spectral algorithm, the dual spectral algorithm, and the composite algorithm.**

ory-specific attraction against the number of memories. As can be seen, there is a superiority of between 6 to 8 Hamming bits of attraction for memories with large eigenvalues as opposed to memories with small eigenvalues. (For $n = 16$ we obtain a superiority of between 2 to 3 Hamming bits of attraction for the same choice of maximum and minimum eigenvalues.) Direction-specific attraction is mapped in Figure 13. As seen, we obtain a direction-specificity of about 4 bits in attraction capability when comparing the strong and weak directions. (For $n = 16$ we perceive a 2 to 3 bit difference in attraction capability between specified and unspecified directions for small values of $k$. When the number of memories $m$ is very small, however, only marginal direction-specificity is displayed.) We stress once again that by increasing the value of the specified $\mu$'s, we increase direction-specific attraction at the expense of memory-specific attraction.

Figure 14 summarises the main features of the three algorithms, and highlights their relationship with the spectral algorithm: in particular, the pseudo-spectral nature of the outer-product algorithm, and the dual spectral nature of the dual spectral algorithm is emphasised.

## REFERENCES

Amari, S. (1977). Neural theory of association and concept formation. *Biological Cybernetics*, **26**, 175–185.

Chvátal, V. (1983). *Linear programming*. New York: W. H. Freeman.

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, **EC-14**, 326–334.

Franklin, J. (1980). *Methods of mathematical economics*. New York: Springer-Verlag.

Goles, E., & Vichniac, G. Y. (1986). Lyapunov function for parallel neural networks. In J. Denker (Ed.), *Neural Networks for Computing*. AIP Conference Proceedings, **151**, 165–181.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, **79**, 2554–2558.

Kohonen, T. (1977). *Associative memory: A system-theoretic approach*. Berlin Heidelberg: Springer-Verlag.

Komlós, J. (1967). On the determinant of (0, 1) matrices. *Studia Scientarum Mathematicarum Hungarica*, **2**, 7–21.

Komlós, J., & Paturi, R. (1988). Convergence results in an associative memory model. *Neural Networks*, **1**, 239–250.

Maruani, A. D., Chevallier, R. C., & Sirat, G. (1987). Information retrieval in neural networks. I. Eigenproblems in neural networks. *Rev. Phys. Appl.*, **22**, 1321–1325.

McCulloch, W. W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin Mathematical Biophysics*, **5**, 115–133.

McEliece, R. J., Posner, E. C., Rodemich, E. R., & Venkatesh, S. S. (1987). The capacity of the Hopfield associative memory. *IEEE Transactions on Information Theory*, **IT-333**, 461–482.

Murty, K. G. (1983). *Linear programming*. New York: Wiley.

Nakano, K. (1972). Associatron—A model of associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-2**, 380–388.

Peretto, P., & Niez, J. J. (1986). Long term memory storage capacity of multiconnected neural networks. *Biological Cybernetics*, **54**, 53–63.

Personnaz, L., Guyon, I., & Dreyfus, G. (1985). Information storage and retrieval in spin-glass like neural networks. *Journal Physique Lettres* **46**, L359–L365.

Psaltis, D., & Venkatesh, S. S. (1989). Information storage in fully connected networks. In Y. C. Lee (Ed.), *Evolution, learning, and cognition* (pp. 51–89). Teaneck, New Jersey: World Scientific.

Strang, G. (1980). *Linear algebra and its applications*. New York: Academic Press.

Venkatesh, S. S., & Psaltis, D. (1985). Efficient strategies for information storage and retrieval in associative neural nets. *Workshop on Neural Networks for Computing*, Santa Barbara, California.

Venkatesh, S. S. (1986a). Epsilon capacity of neural networks. In J. Denker (Ed.), *Neural networks for computing*. AIP Conference Proceedings, **151**, 440–445.

Venkatesh, S. S. (1986b). *Linear maps with point rules: Applications to pattern classification and associative memory*. Ph.D. thesis, California Institute of Technology.

Venkatesh, S. S., & Psaltis, D. (1989). Linear and logarithmic capacities in associative neural networks. *IEEE Transactions on Information Theory*, IT-35, 558–568.

Wolfe, P. (1959). The simplex method for quadratic programming. *Econometrica*, **27**, 282–298.

in Associative Neural Memories:
Theory and Implementation
(M.H. Hassoun, ed.)
Oxford Univ. Press, NY, 1992
[invited contribution]

# FEATURE AND MEMORY SELECTIVE ERROR CORRECTION IN NEURAL ASSOCIATIVE MEMORY

Girish Pancha and Santosh S. Venkatesh*
Department of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104

## 1 INTRODUCTION

In this chapter we describe models of autoassociative memory based upon densely interconnected recurrent networks of McCulloch-Pitts neurons (McCulloch and Pitts, 1943). The model neurons in these networks are linear threshold elements which compute the sign of a linear form of their inputs. In a recurrent network, a collection of these neurons communicate with each other through linear synaptic weights and each neuron changes state based on the net synaptic potential from all the neurons in the network. The instantaneous state of the neural network is described by the collective states of the individual neurons, and the choice of synaptic weights and the neuron updating rule determines the nature of flow in the state space of the network. As in any dynamical system, the fixed points of the network play a critical role in determining its computational properties. In particular, such networks can be used for encoding a set of prescribed items identified with states **u** as fixed points of the network, i.e., states **u** which are fixed under the dynamics of the network. In

*This work was supported in part by the Air Force Office of Scientific Reseach under grant AFOSR 89-0523.

addition, some form of error correction is desired, so that states $\mathbf{u} + \delta\mathbf{u}$ in the vicinity of $\mathbf{u}$ are mapped into $\mathbf{u}$. When the item is to be retrieved from the network, the search can then be done on either the whole item or on part of it. Such searches by data association are especially useful in pattern recognition applications such as speech and vision. *We will describe methods of specifying in a probabilistic sense the conditions under which such error correction occurs.*

We now consider a fully interconnected network of $n$ McCulloch-Pitts neurons. Each neuron is capable of assuming two values: 1 (firing) and -1 (not firing). The instantaneous binary output of each neuron is fed back as an input to the network. At epoch $t$, if $u_1[t]$, $u_2[t], \dots, u_n[t]$ are the outputs of each of the $n$ neurons in the network, then at epoch $t+1$, the $i$th neuron updates itself according to the following threshold rule:

$$u_i[t+1] = \Delta\left(\sum_{j=1}^{n} w_{ij}u_j[t] - w_{i0}\right),$$

where

$$\Delta(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0. \end{cases} \tag{1}$$

Based on the preceding firing rule, each neuron is characterised by a set of $n$ real synaptic weights, and a real threshold value, and the network as a whole is characterised by a matrix of $n^2$ weights $w_{ij}$ and $n$ thresholds $w_{i0}$. Without loss of generality, we can confine our analysis to zero thresholds as thresholds are easily subsumed by the simple expedient of adding a constant input of -1 to each neuron.

The instantaneous state of the network is an $n$-tuple $\mathbf{u} = (u_1, \dots, u_n) \in \mathbb{B}^n$, where $\mathbb{B} = \{-1, 1\}$, and $u$ he $i$th component of $\mathbf{u}$ is the output value of the $i$th neuron. Our goal is to encode items (states in $\mathbb{B}^n$) that are to be stored as fixed points of the recurrent network by an appropriate choice of weight matrix. We label these prescribed items $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$, and hereafter refer to them as memories to distinguish them from other states of the network.

### 1.1 Lyapunov Functions and Error Correction

The network can operate under different modes of updating. If all neurons are simultaneously updated, the mode of operation is said to be synchronous. If at most one neuron is updated at each epoch, the network is said to operate in an asynchronous mode. It has been shown that both modes of operation lead to very similar associative behaviour in neural networks. Note that in this synchronous mode case, we have

$$\mathbf{u}[t+1] = \mathbf{\Delta}(\mathbf{W}\mathbf{u}[t])$$

where $\mathbf{\Delta} : \mathbb{R}^n \rightarrow \mathbb{B}^n$ is an $n$-ary pointwise threshold operator whose $i$th component $\Delta_i(x)$ is as in (1).

Given the arbitrarily prescribed set of $m$ memories, we are interested in specifying patterns of interconnectivity. The nature of flow in state space of the network is completely determined once the matrix of neural interconnection weights is computed. In order for our network to act as an associative memory, we require that the memories be stable. As described earlier, a memory $\mathbf{u}^{(\alpha)}$ is stable if all subsequent mappings return $\mathbf{u}^{(\alpha)}$, i.e., $\mathbf{u}^{(\alpha)}[t+1] = \mathbf{\Delta}(\mathbf{W}\mathbf{u}^{(\alpha)}[t])$ for all $t$. Furthermore, we required that the memories exercise a region of influence around themselves, i.e., states close to or similar to memories should map to the corresponding memories in the network, and thereby exhibit error correcting properties. The Euclidean distance between a state $\mathbf{u}$ and a memory $\mathbf{u}^{(\alpha)}$ is given by

$$d_E(\mathbf{u}, \mathbf{u}^{(\alpha)}) = \left[\sum_{i=1}^{n}(u_i{}^{(\alpha)} - u_i)^2\right]^{1/2} .$$

In $\mathbb{B}^n$, $(u_i{}^{(\alpha)} - u_i) = \pm 2$ if the components are mismatched, and 0 otherwise . Therefore $d_E = 2\sqrt{d_H}$ where the Hamming distance $d_H(\mathbf{u}, \mathbf{u}^{(\alpha)})$ is defined to be the number of mismatched components between the two states. We shall use the average Hamming distance from a memory over which error corrections is exhibited as a natural measure of attraction, and call it the *attraction radius, $\rho$,* of the memory.

From the theory of dynamical systems, we can expect attraction behaviour in neural networks if we can find functions on the systems that are bounded and monotone non-increasing along trajectories in the state space. Such *functions* are called Lyapunov functions. If such a function exists, the stable points of the system reside at minimas of the function. If the memories are programmed to be at these minimas, we can achieve the desired attraction behaviour around the memories.

Two such Lyapunov functions are the Hamiltonian Energy ($E(\mathbf{u})$) function and the Manhattan Norm ($F(\mathbf{u})$) function, where

$$E(\mathbf{u}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}u_iu_j$$

and

$$F(\mathbf{u}) = -\sum_{i=1}^{n}\left|\sum_{j=1}^{n} w_{ij}u_j\right| .$$

These functions act as Lyapunov functions for certain classes of weight matrices under particular modes of operation (cf. Hopfield, 1982; Goles and Vichniac, 1986; Venkatesh and

Psaltis, 1989).[1]

**Proposition 1** *$E(\mathbf{u})$ is non-increasing in asynchronous mode if $\mathbf{W}$ is symmetric and has non-negative diagonal elements; $E(\mathbf{u})$ is non-increasing in any mode if $\mathbf{W}$ is symmetric and non-negative definite.*

**Proposition 2** *$F(\mathbf{u})$ is non-increasing in synchronous mode if $\mathbf{W}$ is symmetric.*

While the existence of Lyapunov functions indicates attraction behaviour, the lack of one does not necessarily indicate that the network will not function as desired. In the following sections, *we shall discuss a family of near-optimal algorithms to compute the weight matrix* $\mathbf{W}$. All but one of these algorithms results in a symmetric weight matrix, and all of them exhibit both desired properties of stability and attraction. In the following sections, we will establish the relationship between the various algorithms, and evaluate their performance.

## 1.2 Capacity and Complexity

Two measures characteristic of any algorithm are the *algorithmic capacity* and *algorithmic complexity*. We will look at these measures for each of the alogrithms in the succeeding sections.

Capacity is the maximal number of memories that can be stored with high probability. It is useful to define capacity as a rate of growth rather than an exact number. Specifically, a sequence of numbers $\{C(n), n \geq 1\}$ is a sequence of capacities if and only if for every

[1] Proofs of propositions in the main text of the chapter are deferred to Appendix A.

$\lambda \in (0,1)$, as $n \to \infty$ the probability that each of the memories is stable approaches 1 whenever $m \leq (1-\lambda)C(n)$, and approaches 0 whenever $m \geq (1+\lambda)C(n)$. We note that a consequence of this is that a sequence of capacities, if it exists, is not unique, but rather determines an equivalence class of sequences $\mathcal{C}$ where, $C(n)$ and $C'(n)$ are sequences in $\mathcal{C}$ iff $C(n) \sim C'(n)$ as $n \to \infty$.[2]

We define complexity to be the number of elementary operations required to compute the matrix of weights. For the purposes of this discussion, the elementary operations are multiplication and addition of two real values. We are therefore interested in determining the complexity of an alogrithm given $m$ memories in $n$-space. In some cases, an algorithm may have a recursive definition which may result in a reduced algorithmic complexity from a practical standpoint when memories are added to the network one at a time.

# 2 OUTER PRODUCT ALGORITHM

Let $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$ be a selection of $m$ memories. The Outer Product Algorithm prescribes that the weight matrix $\mathbf{W}^{op}$ be chosen in the following manner (see Chapter 1 for additional details):

$$\mathbf{W}^{op} = \mathbf{U}\mathbf{U}^T , \tag{2}$$

where $\mathbf{U} = [\mathbf{u}^{(1)}\ \mathbf{u}^{(2)} \cdots \mathbf{u}^{(m)}]$.

This scheme uses the sum of outer-products of the memory vectors as correlation between the memories to form a weight matrix $\mathbf{W}^{op}$ so that an input vector close to a memory

[2] On asymptotic notation. If $\{x(n)\}$ and $\{y(n)\}$ are any two sequences, we denote: $x_n = \mathcal{O}(y_n)$ if there exists a constant $K$ such that $|x(n)| \leq K|y(n)|$ for every $n$; $x(n) = o(y(n))$ if $|x(n)|/|y(n)| \to 0$ as $n \to \infty$; $x(n) \sim y(n)$ if $x(n)/y(n) \to 1$ as $n \to \infty$; and $x(n) = \omega(y(n))$ if $|x(n)|/|y(n)| \to \infty$ as $n \to \infty$.

vector will pick out that memory vector. The functioning of the algorithm as an efficient associative memory has been well documented (cf. Hopfield, 1982), and theoretical results on the capacity have been derived (McEliece, Posner, Rodemich, and Venkatesh, 1987; Komlós and Paturi, 1988).[3] We will now review some results for the scheme.

## 2.1 Error Correction

From the definition it follows that $\mathbf{W}^{op}$ is symmetric, nonnegative definite. Therefore, the algorithmic flow in state space is towards the minimisation of bounded functionals (the Manhattan Norm $F$ in synchronous mode and the Energy $E$ in any mode).[4] The trajectories therefore will tend to terminate in stable states which are local minima of the functionals. If these stable states correspond to the stored memories, the Outer Product Algorithm satisfies the requirements of a physical associative memory. To examine its efficacy, however, we need to estimate its storage capacity, and the algorithmic complexity of computing the weights.

We first consider the effect of $\mathbf{W}^{op}$ on a memory $\mathbf{u}^{(\alpha)}$. We have

$$\begin{aligned}
\left[\mathbf{W}^{op}\mathbf{u}^{(\alpha)}\right]_i &= \sum_{j=1}^{n} w_{ij}^{op} u_j{}^{(\alpha)} \\
&= \sum_{j\neq i}\sum_{\beta=1}^{m} u_i{}^{(\beta)} u_j{}^{(\beta)} u_j{}^{(\alpha)} \\
&= (n-1)u_i{}^{(\alpha)} + \sum_{j\neq i}\sum_{\beta\neq\alpha} u_i{}^{(\beta)} u_j{}^{(\beta)} u_j{}^{(\alpha)} \\
&= (n-1)u_i{}^{(\alpha)} + \delta u_i{}^{(\alpha)}\,. \qquad (3)
\end{aligned}$$

We see that, in effect, there is a "signal" term and a "noise" term. Assuming that the

[3] See also Chapter 9.

[4] In some variations a zero diagonal is enforced for the matrix $\mathbf{W}^{op}$ in (2). The matrix is then symmetric, with non-negative diagonal elements so that the energy is non-increasing in asynchronous operation.

memories are chosen randomly from a sequence of symmetric Bernoulli trials, the noise term $\delta u_i^{(\alpha)}$ has a mean of 0 and a standard deviation of $\sqrt{(n-1)(m-1)}$. The mean of the absolute value of the signal term $(n-1)u_i^{(\alpha)}$ is $(n-1)$. Thus, if $m = o(n)$, the signal term dominates the error term, and we can write

$$\mathbf{W}^{op}\mathbf{U} = (n-1)\mathbf{U} + \delta\mathbf{U} \approx (n-1)\mathbf{U}.$$

It will follow that $\Delta(\mathbf{W}^{op}\mathbf{U}) = \mathbf{U}$ with high probability if $m = o(n)$.

## 2.2 Capacity and Complexity

The memories $\mathbf{u}^{(\alpha)}$, $\alpha = 1, \ldots, m$ can be identified as *pseudo-eigenvectors* of the linear operator $\mathbf{W}^{op}$ with *pseudo-eigenvalues* $n-1$. When randomly chosen, they are stable in a probabilistic sense only if the mean to standard deviation given by $\sqrt{(n-1)/(m-1)}$ is large. More precisely, the following assertion holds (cf. McEliece, Posner, Rodemich, and Venkatesh, 1987; Komlós and Paturi, 1988). Chapter 9 contains more details.

**Proposition 3** *The (stable state) capacity of the Outer Product Algorithm is $n/4\log n$.*

We sketch one side of the proof in Appendix A to illustrate some of the ideas involved. The gentle reader is also invited to delve into Chapter 1 for similar derivations.

Somewhat more can be shown than asserted above. In fact, if $\rho \in [0, 1/2)$ is any fixed quantity, and random probes are generated at a distance $\rho n$ from each of the memories, then all the errors in all the probes are corrected in one synchronous step with high probability if the number of memories $m$ increases no more rapidly with $n$ than $(1-2\rho)^2 n/4\log n$. In particular, this result implies that within capacity each of the memories has (asymptotically) an identically sized ball of attraction of radius $\rho n$. From a physical viewpoint this implies that all the memories are treated equivalently by the Outer Product Algorithm as are all the features (memory components).

If we label the number of elementary operations required to compute $\mathbf{W}^{op}$ for $m$ memories as $N^{op}$, then by counting the number of operations needed for matrix multiplication, by considering that that the weight matrix $\mathbf{W}^{op}$ is symmetrical, and by noting that the diagonal elements are trivially specified, we find that $N^{op} = (mn^2 - mn)/2$. In addition, by inspection of (3), we notice that the outer product matrix can be recursively computed via

$$\mathbf{W}^{op}[\alpha] = \mathbf{W}^{op}[\alpha - 1] + \mathbf{u}^{(\alpha)}(\mathbf{u}^{(\alpha)})^T, \qquad \alpha \geq 1,$$

where $\mathbf{W}^{op}[\alpha]$ denotes the outer-product weight matrix generated by the first $\alpha$ memories, and $\mathbf{W}^{op}[0] = \mathbf{0}$. This means that the incremental complexity $N^{op}[\alpha] = (n^2 - n)$, and the cost of computing the weight matrix incrementally is twice the cost of computing it from scratch for any given set of $m$ memories.

# 3 MEMORY SELECTIVE ALGORITHMS

In this section, we will discuss schemes to generate the weight matrix to yield a larger capacity than the outer product scheme. In addition, *these schemes will enable us to selectively increase attraction radii around specified memories.* (Compare with the Outer Product Algorithm where uniform attraction balls around the memories obtains.) The constructions are an extension of the outer product scheme to make the memories true eigenvectors of the linear operator $\mathbf{W}$, and then specifying the eigenvalues of the memories.

**Construction 1** Define the interconnection matrix $\mathbf{W}^s$ as follows:

$$\mathbf{W}^s = \mathbf{U}\Lambda(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T, \tag{4}$$

where $\Lambda = \mathbf{dg}\left[\lambda^{(1)}, \ldots, \lambda^{(m)}\right]$ is the $m \times m$ diagonal matrix of positive eigenvalues $\lambda^{(1)}, \ldots, \lambda^{(m)} > 0$, and $\mathbf{U} = \left[\mathbf{u}^{(1)}\ \mathbf{u}^{(2)} \cdots \mathbf{u}^{(m)}\right]$ is the $n \times m$ matrix of memory column vectors.

**Construction 2** Given $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$, choose any $(n-m)$ vectors $\mathbf{u}^{(m+1)}, \ldots, \mathbf{u}^{(n)} \in \mathbb{B}^n$ such that $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)}$ are linearly independent. Define the interconnection matrix $\mathbf{W}^s$as follows:

$$\mathbf{W}^s = \mathbf{U}_a\Lambda_a\mathbf{U}_a^{-1},$$

where the augmented matrices $\mathbf{U}_a$ and $\Lambda_a$ are defined as

$$\Lambda_a = \mathbf{dg}[\lambda^{(1)}, \ldots, \lambda^{(m)}, 0, \ldots, 0], \qquad \mathbf{U}_a = [\mathbf{u}^{(1)} \cdots \mathbf{u}^{(n)}].$$

We note that

$$\mathbf{W}^s\mathbf{U} = \mathbf{U}\Lambda, \qquad \mathbf{W}^s\mathbf{U}_a = \mathbf{U}_a\Lambda_a, \tag{5}$$

so that $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$ are eigenvectors of $\mathbf{W}^s$ and $\Lambda$ is the spectrum of $\mathbf{W}^s$ (Personnaz, Guyon, and Dreyfus, 1985; Venkatesh and Psaltis, 1989). Therefore, we are guaranteed to have stable memories as long as $\mathbf{W}^s$ is well defined.

Hybrids of the two methods described above can also be used where the matrix U is partially augmented and then the pseudo-inverse is computed.

## 3.1 Error Correction

For the case of an $m$-fold degenerate spectrum, $\lambda^{(1)}, \ldots, \lambda^{(m)} = \lambda > 0$, we see that the matrix $\mathbf{W}^s$ is symmetric with non-negative eigenvalues, i.e., it is non-negative definite.

Therefore there exist Lyapunov functions in this case. In fact, consider the energy function $E(\mathbf{u}) = -\frac{1}{2}\langle \mathbf{u}, \mathbf{W}\mathbf{u}\rangle$.

For each memory, $\mathbf{u}^{(\alpha)}$, the energy is hence given by

$$E(\mathbf{u}^{(\alpha)}) = -\frac{1}{2}\langle \mathbf{u}^{(\alpha)}, \mathbf{W}\mathbf{u}^{(\alpha)}\rangle = -\frac{\lambda n}{2} .$$

Let $\mathbf{u} \in \mathbb{B}^n$ be arbitrary. We can write $\mathbf{u}$ in the form

$$\mathbf{u} = \sum_{\alpha=1}^{m} c^{(\alpha)}\mathbf{u}^{(\alpha)} + \mathbf{u}_\perp = \mathbf{u}_{||} + \mathbf{u}_\perp ,$$

where $\mathbf{u}_{||}$ is the projection of $\mathbf{u}$ into the linear span of the $m$ memories, $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots, \mathbf{u}^{(m)}$, and $\mathbf{u}_\perp$ is a vector in the orthogonal subspace. Then we have $\langle \mathbf{u}_{||}, \mathbf{u}_\perp\rangle = 0$, and by the Pythagorean theorem,

$$\|\mathbf{u}\|^2 = \|\mathbf{u}_{||}\|^2 + \|\mathbf{u}_\perp\|^2 = \left\| \sum_{\alpha=1}^{m} c^{(\alpha)}\mathbf{u}^{(\alpha)} \right\|^2 + \|\mathbf{u}_\perp\|^2 .$$

The energy is then given by

$$\begin{aligned} E(\mathbf{u}) = -\frac{1}{2}\langle \mathbf{u}, \mathbf{W}\mathbf{u}\rangle &= -\frac{1}{2}\left\langle \sum_{\alpha=1}^{m} c^{(\alpha)}\mathbf{u}^{(\alpha)} + \mathbf{u}_\perp , \sum_{\alpha=1}^{m} \lambda c^{(\alpha)}\mathbf{u}^{(\alpha)} \right\rangle \\ &= -\frac{\lambda}{2}\left\| \sum_{\alpha=1}^{m} c^{(\alpha)}\mathbf{u}^{(\alpha)} \right\|^2 \\ &\geq -\frac{\lambda}{2}\|\mathbf{u}\|^2 = -\frac{\lambda n}{2}. \end{aligned}$$

It follows that the stored memories form *global* Energy minima.

For the general Spectral matrix in (4), exact Lyapunov functions are hard to come by. The signal-to-noise ratio, however, serves as a good *ad hoc* measure of attraction capability. Consider synchronous operations with $\mathbf{W}^s$ on a state vector $\mathbf{u}= \mathbf{u}^{(\alpha)}+ \delta\mathbf{u} \in \mathbb{B}^n$. We have

$$\mathbf{W}^s\mathbf{u} = \mathbf{W}^s(\mathbf{u}^{(\alpha)} + \delta\mathbf{u}) = \mathbf{W}^s\mathbf{u}^{(\alpha)} + \mathbf{W}^s\delta\mathbf{u}.$$

Once again, there exists a "signal" term, $\mathbf{W}^s\mathbf{u}^{(\alpha)}$, and a "noise" term, $\mathbf{W}^s\delta\mathbf{u}$. We anticipate that the greater the signal-to-noise ratio, the greater the attraction around $\mathbf{u}^{(\alpha)}$. Let the Hamming distance between $\mathbf{u}$ and $\mathbf{u}^{(\alpha)}$, $d_H(\mathbf{u}, \mathbf{u}^{(\alpha)})$, equal $d$, i.e., $||\delta\mathbf{u}|| = 2\sqrt{d}$. The (strong) norm of the matrix $\mathbf{W}^s$ is defined as

$$||\mathbf{W}^s|| = \sup_{\mathbf{x}} \frac{||\mathbf{W}^s\mathbf{x}||}{||\mathbf{x}||}, \qquad ||\mathbf{x}|| \neq 0.$$

It follows (cf. Strang, 1980) that $||\mathbf{W}^s|| = \sqrt{k}$, where $k$ is the the largest eigenvalue of the matrix $(\mathbf{W}^s)^T\mathbf{W}^s$. For the case of the degenerate spectrum, $\lambda^{(1)}, \ldots, \lambda^{(m)} = \lambda > 0$, $\mathbf{W}^s$ is symmetric, and $(\mathbf{W}^s)^T\mathbf{W}^s = (\mathbf{W}^s)^2$. Therefore, the maximum eigenvalue of $(\mathbf{W}^s)^T\mathbf{W}^s = k = \lambda^2$, and the signal-to-noise ratio (SNR) is given by

$$SNR = \frac{||\mathbf{W}^s\mathbf{u}^{(\alpha)}||}{||\mathbf{W}^s\delta\mathbf{u}||} \geq \frac{\lambda^{(\alpha)}\sqrt{n}}{(\sqrt{k})\,(2\sqrt{d})} = \frac{1}{2}\sqrt{\frac{n}{d}}\,.$$

Thus, we would expect the attraction sphere around $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$ to increase as $n$ increases for the $m$-fold degenerate Spectral Scheme. For the general non-degenerate case, we expect that by varying the size of $\lambda^{(\alpha)}$, the SNR, and hence the attraction capability, be proportionately increased or decreased for the $\alpha$th memory $\mathbf{u}^{(\alpha)}$(Figure 1).

*Figure 1 goes here*

## 3.2 Capacity and Complexity

To determine the capacity of these schemes, we use the following proposition (Komlós, 1967).

**Proposition 4** *Let $m$ increase with $n$ such that $m \leq n$. Then the probability that a randomly chosen set of $n$-tuples $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)} \in \mathbb{B}^n$ is linearly independent aproaches one as $n \rightarrow \infty$.*

It follows that the probability that $\mathbf{W}^s$is well defined approaches one as $n \to \infty$. Since a linear transformation has at most $n$ eigenvalues, the static capacity of the spectral scheme is $n$. (Dynamic capacities pose greater problems in evaluation here because of the added dependency structure. For some theoretical results in this regard see Dembo (1989); for numerical simulations see Venkatesh and Psaltis (1989) and Chapter 1 of this volume.)

Let $N^s$ denote the number of elementary operations required to compute the weight matrix $\mathbf{W}^s$ directly from the $m$ memories to be stored. Then using the fact that $(\mathbf{U}^T\mathbf{U})^{-1}$ is symmetric, we can use the Cholesky decomposition to compute its inverse. This along with the rest of the matrix multiplications yields $N^s = mn^2 + m^2n + m^3/2 + O(n^2)$.

When the eigenvalues $\lambda^{(\alpha)}$ are $m$-fold degenerate, Greville's algorithm can be used to recursively compute the pseudo-inverses which in turn results in a recursive construction for the weight matrices $\mathbf{W}^s[\alpha]$. Here $\mathbf{W}^s[\alpha]$ denotes the spectral weight matrix corresponding to the first $\alpha$ memories. In fact, let $\lambda^{(\alpha)} = \lambda > 0$, $\alpha \geq 1$. For each $\alpha \geq 1$ let $\mathbf{e}^{(\alpha)}$ be the $n$-vector defined by

$$\mathbf{e}^{(\alpha)} = (\lambda \mathbf{I} - \mathbf{W}^s[\alpha - 1])\mathbf{u}^{(\alpha)},$$

where we define $\mathbf{W}^s[0] = \mathbf{0}$. Then it is easy to verify by induction that

$$\mathbf{W}^s[\alpha] = \mathbf{W}^s[\alpha - 1] + \frac{\mathbf{e}^{(\alpha)}(\mathbf{e}^{(\alpha)})^T}{(\mathbf{u}^{(\alpha)})^T\mathbf{e}^{(\alpha)}}, \qquad \alpha \geq 1. \tag{6}$$

Now let $N^s[\alpha]$ denote the number of elementary operations needed to compute the update of the weight matrix according to the recursion (6). Again counting the number of multiplications (the number of additions is of the same order), we get the following cost estimate: $N^s[\alpha] = 2n^2 + 2n$, $\alpha \geq 1$. Note that for all choices of $m < n$, we have $mN^s[\alpha] \gtrsim 2N^s$, so that, especially for large $n$, the recursive construction of $\mathbf{W}^s$ through the updates (6) is computationally about twice as expensive as the direct estimation of $\mathbf{W}^s$. Note that the cost is only about four times more than that of the simple Outer Product Algorithm.

# 4 FEATURE SELECTIVE ALGORITHMS

## 4.1 Dual Spectral Alogrithm

The following scheme, formally related to the Outer Product and Spectral Algoritnms, was introduced by Maruani, Chevallier, and Sirat, (1987). Let $\mathbf{U} = [\mathbf{u}^{(1)}\ \mathbf{u}^{(2)}\ \cdots\ \mathbf{u}^{(m)}]$ be the matrix of memories as before. Let $\mathbf{x}^{(\beta)}$, $\beta = 1, \dots, n-m$, be a set of linearly independent vectors in $\mathbb{R}^n$ which are individually orthogonal to each of the memories, i.e., $\mathbf{X}^T\mathbf{U} = \mathbf{0}$, where we define the $n \times (n-m)$ matrix $\mathbf{X} = [\mathbf{x}^{(1)}\ \mathbf{x}^{(2)}\ \cdots\ \mathbf{x}^{(n-m)}]$. Define a weight matrix $\mathbf{W}$ with weights $w_{ij}$ given by

$$w_{ij} = \begin{cases} -\sum_{\beta=1}^{n-m} x_{i\beta}x_{j\beta} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases},$$

where $x_{k\beta}$ is the $k$th component of $\mathbf{x}^{(\beta)}$. If we define $\hat{\mu}_i = \sum_{\beta=1}^{n-m} x_{i\beta}^2$, $i = 1, \dots, n$, we see that

$$\mathbf{W} = \hat{\mathbf{M}} - \mathbf{X}\mathbf{X}^T \tag{7}$$

where $\hat{\mathbf{M}} = \mathbf{dg}[\hat{\mu}_1, \dots, \hat{\mu}_n]$. Thus,

$$\begin{aligned} \mathbf{W}\mathbf{U} &= \hat{\mathbf{M}}\mathbf{U} - \mathbf{X}\mathbf{X}^T\mathbf{U} \\ &= \hat{\mathbf{M}}\mathbf{U}. \end{aligned} \tag{8}$$

## 4.2 Error Correction

Comparing (5) and (8) we see that the Spectral and Dual Spectral Algorithms exhibit an interesting duality.

In the Spectral Scheme, the eigenvectors of $\mathbf{W}^s$ are the memories, so that the column space of $\mathbf{W}^s$ is given by the span of the memories. Therefore, if the memories are far enough from each other and the initial state vector $\mathbf{u}$ is close enough to a memory, $\mathbf{W}^s$ combined with the thresholding operation *projects* $\mathbf{u}$ onto the memory.

In the Dual Spectral Scheme, since the parameters $\hat{\mu}_i$ are positive for each choice of $i$, it follows that

$$\Delta(\mathbf{W}\mathbf{u}^\alpha)_i = \Delta(\hat{\mu}_i u_i^\alpha) = u_i^\alpha, \qquad \text{for each } i = 1, \ldots, n, \quad \alpha = 1, \ldots, m.$$

So the memories $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$ are fixed points in the scheme as well. $\mathbf{W}$ as defined in (7) is a zero-diagonal symmetric matrix. Thus, we know that there exist Lyapunov functions in both modes of operation and that the network will exhibit some form of attraction behaviour. The weight matrix $\mathbf{W}^d$ is obtained by taking the correlation of vectors that are orthogonal to the memories and then setting the diagonal elements to be 0. In creating the zero diagonal, we essentially add perturbations to the left nullspace of $\mathbf{U}$ in the directions of the memories. The *strength* of the perturbations along any component, $i$, is proportional to $\hat{\mu}_i$. Thus, each of the $\hat{\mu}_i$'s corresponds to a directional distortion, and we expect the SNR of the Dual Spectral Scheme to vary from direction to direction proportionately with the value of $\hat{\mu}_i$. We therefore expect that the larger the $\hat{\mu}_i$, the more the information that is lost if the $i$th bit is flipped and, hence, the smaller the attraction in the $i$th direction.

As an illustration, let us consider the case where $n = 3$, and $\mu_y \gg \mu_x, \mu_z$. Each memory **u** would be preferentially attracted in the $x$ and $z$ direction, indicated schematically by an attraction spheroid in Figure 2; i.e., a vector with a different $y$ component is less likely to map back to **u** but vectors with different $x$ and/or $z$ components will probably be within the attraction region of **u**. In other words,

*Figure 2 goes here*

$$\mathbf{P}\left[\begin{pmatrix} u_x \\ -u_y \\ u_z \end{pmatrix} \rightarrow \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\right] < \mathbf{P}\left[\begin{pmatrix} u_x \\ u_y \\ -u_z \end{pmatrix} \rightarrow \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\right], \mathbf{P}\left[\begin{pmatrix} -u_x \\ u_y \\ u_z \end{pmatrix} \rightarrow \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}\right] .$$

### 4.3 Constructing Feature Selective Weights

In the previous section, the orthogonal basis, **X**, was chosen arbitrarily and therefore resulted in some lack of control in specifying attraction capability. As we argued above, the $\hat{\mu}_i$'s essentially control directional attraction and we have no means of specifying these under the above approach. We now suggest a few *schemes for constructing the weight matrices that specify the $\mu$-values and thereby achieve direction-specific attraction.* Specifically, for a *prescribed set* $\mu_1, \ldots, \mu_n > 0$ of directional attraction strengths, and $\mathbf{M} = \mathbf{dg}\,[\mu_1, \ldots, \mu_n]$, we require a weight matrix $\mathbf{W}^d$ such that

$$\mathbf{W}^d\mathbf{U} = \mathbf{MU}\,. \tag{9}$$

We define $\mathbf{W}^d$ such that:

$$w_{ij}{}^d = \begin{cases} -\sum_{\beta=1}^{n-m}(x_{i\beta}b_\beta)(x_{j\beta}b_\beta) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}, \tag{10}$$

where $x_{i\beta}$ is the $i$th component of the basis vector $\mathbf{x}^{(\beta)}$ as defined earlier, and $b_\beta$ is the $\beta$th component of a vector which we will specify shortly. Thus, given $\mu_1, \ldots, \mu_n$ we need to find a vector **b** such that with **Y** =**Xb**

$$\mathbf{W}^d = \mathbf{M} - \mathbf{Y}\mathbf{Y}^T \,. \tag{11}$$

(Note that the columns of **Y**, in general, are not orthogonal.)

Assuming that $\mathbf{W}^d$ has the form given in (10), let us now consider the effect of $\mathbf{W}^d$ on the $i$th element of a memory $\mathbf{u}^{(\alpha)}$:

$$\begin{aligned}
[\mathbf{W}^d\mathbf{u}^{(\alpha)}]_i &= \sum_{j=1}^{n} w_{ij}{}^d u_j^{(\alpha)} \\
&= -\sum_{j\neq i}\sum_{\beta=1}^{n-m} b_\beta^2\, x_{i\beta}\, x_{j\beta}\, u_j^{(\alpha)} \\
&= -\sum_{j=1}^{n}\sum_{\beta=1}^{n-m} b_\beta^2\, x_{i\beta}\, x_{j\beta}\, u_j^{(\alpha)} + \sum_{\beta=1}^{n-m} b_\beta^2 x_{i\beta}^2\, u_i^{(\alpha)} \\
&= \sum_{\beta=1}^{n-m} b_\beta^2\, x_{i\beta}^2\, u_i^{(\alpha)} \,.
\end{aligned}$$

We require from (9) that $[\mathbf{W}^d\mathbf{u}^{(\alpha)}]_i = \mu_i u_i^{(\alpha)}$ where $\mu_i > 0$. By inspection, we obtain the relationship

$$\mu_i = \sum_{\beta=1}^{n-m} x_{i\beta}^2\, b_\beta^2 \,.$$

Define $a_{i\beta} = x_{i\beta}^2$, and $c_\beta = b_\beta^2$. Then we require $\mathbf{A}\mathbf{c} = \mathbf{M}_\mu$, where **A** is a known $n \times (n-m)$ matrix with non-negative elements $a_{i\beta} = x_{i\beta}^2$, **c** is an unknown $(n-m)$-dimensional vector with $c_\beta = b_\beta^2$ constrained to be non-negative, and $\mathbf{M}_\mu$ is a specified $n$-dimensional vector with positive components $\mu_1, \ldots, \mu_n$.

We notice that this is an overspecified system of $n$ equations with $(n-m)$ unknowns, where both **c** and $\mathbf{M}_\mu$ are constrained to have non-negative elements. Linear programming techniques can be used to solve this system of equations. We can choose the $\mu$-values in a variety of ways. A few representative methods are:

1. Specify $\mu_1,\ldots,\mu_k,\ k \le n-m$, let $\mu_k+1,\ldots,\mu_n-m < \epsilon$, and minimise $\epsilon$.

2. Specify $\mu_1, \ldots, \mu_n$, and

   (a) Minimise the mean-square error given by

   $$\|\mathbf{Ac} - \mathbf{M}_\mu\|^2 = \sum_{i=1}^{n} (a_{i,1}c_1 + \cdots + a_{i,n-m}c_{n-m} - \mu_i)^2$$

   subject to the constraints $\mathbf{M}_\mu > 0$, $\mathbf{c} > \mathbf{0}$.

   (b) Minimise the largest absolute error, $c_0$, given by

   $$\max(|\epsilon_1|, \ldots, |\epsilon_n|)$$

   where $\epsilon_i$, the error in $\mu_i$, is

   $$\mu_i - (a_{i,1}c_1 + \cdots + a_{i,n-m}c_{n-m}), \qquad i = 1, \ldots, n \,.$$

For simplicity, we consider algorithms employing the first linear programming approach outlined above. We have modified the initial basis for the nullspace of $\mathbf{U}$ using the results of the simplex method such that

$$\mathbf{W}^d = \mathbf{M} - \mathbf{Y}\mathbf{Y}^T \,,$$

where $\mathbf{M} = \mathbf{dg}\,[\mu_1, \mu_2, ..., \mu_n]$ with $\mu_1, \ldots, \mu_k > 0$ specified by us, $0 < \mu_{k+1}, \ldots, \mu_n \le \epsilon < \min(\mu_1, \ldots, \mu_k)$, and $\mathbf{Y} = \mathbf{Xb}$ is a set of basis vectors for the left nullspace of $\mathbf{U}$. Since $\mu_i$, $i = 1, \ldots, n$, are positive, we see that all the memories are strictly stable in the Dual

Spectral Scheme as long as the memories, $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)}$, are linearly independent, and we are able to find the vector $\mathbf{c}$ in the system (12) through linear programming. In Appendix B we outline the linear programming approach in some detail.

## 4.4 Capacity and Complexity

By Komlós' result (Proposition 4), we are guaranteed that almost all choices of $n$ memories or fewer are linearly independent, so that for almost all choices of $n-1$ memories there is an orthogonal subspace of dimension 1, while almost all choices of $n$ memories span the space $\mathbb{R}^n$ and therefore the orthogonal subspace is of dimension 0. The storage capacity of the Dual Spectral Scheme of (10) is directly $n-1$, since $n-1$ is the number of memories for which we can still specify a left nullspace $\mathbf{X}$.

To find an $n$-dimensional vector under constraints, the Simplex Method iterates from one feasible solution to another until it finds an optimal feasible solution. In the worst case, we need to test each vertex of the feasible region which is an $n$-sided polyhedron, leading to $2^n - 1$ iterations. However, such cases are rare and require careful specification of the constraints designed solely to approach the worst case behaviour. In practice, it has been widely reported (Chvátal, 1983; Murty, 1983) that the number of iterations is almost always between 1 to 3 times the number of constraints. Thus, for the case of specifying $k$ values of $\mathbf{M}_\mu$, we would expect at the most $3n$ iterations. For the (Revised) Simplex Method, a good estimate of the average cost of each iteration in our scheme is $52n - 10m - 10k + 10$, while for the Standard Simplex Method, a good estimate is $2n^2 - mn - kn + n)/4$ (cf. Chvátal, 1983, p. 113). Thus, we estimate that the total cost of specifying $k$ values of $\mathbf{M}_\mu$ is $O(n^2)$ (using the Revised Simplex Method). The cost of finding a basis for the nullspace of **U** (through Gram-Schmidt orthogonalisation) includes finding $(\mathbf{U}^T\mathbf{U})^{-1}$ and two matrix multiplications and is given by $mn^2 + (m^2n)/2 - m^3/2 + O(n^2)$. Finally, the cost of finding $\mathbf{W}^d$ from **c** and **X** is $n^3 - n^2m + O(n^2)$. So, we can say that on the average,

$$N^d = n^3 + \frac{1}{2}m^2n - mn^2 - \frac{m^3}{2} + O(n^2)\,,$$

where $N^d$ is the number of elementary operations needed to compute $\mathbf{W}^d$.

# 5 COMPOSITE ALGORITHMS

In Section 3 we saw ways of increasing the radii of attraction-spheres around memories. In Section 4.3 we saw ways of specifying increased attraction in certain directions around each of the memories. A natural extension of these schemes is to create a Composite Scheme (Venkatesh, Pancha, Psaltis, and Sirat, 1990) with weight matrix $\mathbf{W}^c$ given by

$$\mathbf{W}^c = \mathbf{W}^s + \mathbf{W}^d.$$

Since $\mathbf{W}^c$ is a linear combination of $\mathbf{W}^s$ and $\mathbf{W}^d$, we would expect memories to be stable in the Composite Scheme for reasons decribed in the previous sections. The idea of the Composite Scheme is to specify both memory-specific attraction by specifying $\lambda$ for each memory, and direction-specific attraction by specifying $\mu$ for the individual directions (Figure 3).

*Figure 3 goes her*

Here, the spectrum of $\mathbf{W}^s$ is no longer degenerate, and $\mathbf{W}^c$, consequently, is no longer symmetric. As the Composite Algorithm combines the memory-specific Spectral Algorithm, and the direction-specific Dual Spectral Algorithm, it works effectively in shaping the attraction regions as desired. It should be noted that the relative values of the $\lambda^{(1)}, \ldots, \lambda^{(m)}$, compared to the $\mu_1, \ldots, \mu_n$, need to be considered in order not to lose the effects of one of the two parts of the composite scheme.

Note that the capacity of the Composite Scheme is $n - 1$. The algorithm complexity of the Composite Scheme is the sum of the complexities of the Spectral and Dual Spectral Schemes, except that we need not find $(\mathbf{U}^T\mathbf{U})^{-1}$ twice. Therefore the complexity, $N^c$, is given by $3n^3 + O(n^2)$ for $m \lesssim n$.

# 6 SIMULATIONS

There are a number of open questions involved with the schemes outlined above. In this section, we discuss trends observed in computer simulations that validate some of our conjectures. All memories were chosen randomly using a binomial pseudo-random generator. Test input vectors at specified Hamming distances from the memories were generated by reversing the signs of randomly chosen components for the Outer-Product and Spectral schemes. In the case of the Dual Spectral schemes, test input vectors were generated by reversing the signs of randomly chosen components with the specified or unspecified $\mu$ values.

Analytical bounds are difficult to arrive at for the attraction radius as a function of the number of memories and the dimension of the state space. Figure 4 plots the attraction radius Spectral Scheme. The attraction radius was estimated by averaging the maximum Hamming distance of error-correction around stable memories over several independent runs. The memories were divided into two equal sized groups, one group with eigenvalue $\lambda(\text{large}) = 3n$, and the other group with eigenvalue $\lambda(\text{small}) = n$. The respective attraction

*Figure 4 goes here*

radii of the $\lambda$(large) memories and the $\lambda$(small) memories are plotted against the number of memories $m$.

In the Dual Spectral Schemes, there is the question of the number of directions, $k$, that can be specified given a set of $m$ memories and $n$ neurons, arising from the nature of the construction of the $\mathbf{W}^d$ matrix. It is obvious from the previous discussions about the dimensions of $\mathbf{A}$ and $\mathbf{c}$ that we can surely specify no more than $n - m$ directions. However, there is a possibility (albeit small) that there exist no feasible solutions for pathological cases where $k < n - m$. Another quantity we are interested in is the size of $\epsilon$, the largest of the unspecified $\mu$'s, compared to the size of the specified $\mu$'s since we have conjectured that this will affect directional attraction.

While there exists little theory for the Simplex Method which will enable us to gauge these parameters, simulations show that $\epsilon$ is typically small (Figure 5) compared to $\mu_i$ for the specified directions ($< 0.5\,\mu_i$), and $k$ is typically of the order of $n/4$ in the ranges simulated. We conjecture that this behaviour continues to hold for large $n$. Figures 6(a) and 6(b) plot directional attraction for the Dual Spectral Scheme. Attraction data for a given direction were generated by investigating probe vectors at various Hamming distances from a memory with the component of the probe in the direction being investigated being chosen to be opposite in sign to the corresponding component of the memory. Flipping a bit in an imporant (large $\mu$) direction almost always reduced the attraction to the memory compared to an unimportant (small $\mu$) direction.

*Figure 5 goes he*

*Figures 6( and 6(b) go here.*

Figures 7(a) and 7(b) plot the results for the composite scheme. The memories were

*Figures 7(a and 7(b) go here*

divided into two groups, one group corresponding to a "large" eigenvalue, $\lambda_{lg} = 3$, and the other group corresponding to a "small" eigenvalue, $\lambda_{sm} = 1$. Directional attraction parameters for $k$ directions were set equal to $\mu_{lg} = 16$. Attraction radii were determined for the large $\mu$ (specified) and small $\mu$ (unspecified) directions as a function of $k$, for both the large eigenvalue memories and the small eigenvalue memories. As can be seen, there is degradation of the behaviour when compared to the memory-specific and feature-specific schemes. However, the attraction behaviour is in keeping with our expectations, and we conjecture that the behaviour of these networks improves as $n$ increases.

# 7 SUMMARY

The recurrent neural network paradigm for associative memory is attractive in its simplicity and computational tractability. The classical Outer Product Algorithm, for instance, has very low implementation complexity and yet exhibits near-linear memory storage capacities with correction of a linear number of random errors uniformly in balls around the memories. In this chapter we have shown how it is possible to exercise a macroscopic degree of nonuniform error correction around the stored memories. In particular, both memorv and feature selective error correction is feasible using a composite algorithm which exploits the spectral characteristics of the interconnectivity weight matrix. These spectral based approaches are near-optimal in character and, in particular, are characterised by low implementation complexities and linear storage capacities. Extensions of these approaches are possible to higher-order neural networks (where the model neurons compute the sign of polynomial forms of their inputs) with concomitant increases in the storage capacities of the networks (cf. Venkatesh and Baldi, 1991a, 1991b).

# APPENDIX A: PROPOSITIONS

**Proposition 1** *$E(\mathbf{u})$ is non-increasing in asynchronous mode if* $\mathbf{W}$ *is symmetric and has non-negative diagonal elements; $E(\mathbf{u})$ is non-increasing in any mode if* $\mathbf{W}$ *is symmetric and non-negative definite.*

PROOF: (a) Consider either a synchronous or asynchronous mode of operation. For any state $\mathbf{u}$, the algorithm results in a flow in state space defined by $\mathbf{u} \mapsto \mathbf{u} + \delta\mathbf{u}$. We note that $\delta\mathbf{u}$ is an $n$-tuple with each component taking on one of the values 0, -2, or 2. The change in $E$ is given by

$$\begin{aligned} \delta E &= E(\mathbf{u}+\delta\mathbf{u}) - E(\mathbf{u}) \\ &= -\frac{1}{2}[\langle\delta\mathbf{u},\mathbf{W}\mathbf{u}\rangle + \langle\mathbf{u},\mathbf{W}\delta\mathbf{u}\rangle + \langle\delta\mathbf{u},\mathbf{W}\delta\mathbf{u}\rangle]. \end{aligned}$$

Since $\mathbf{W}$ is symmetric, $\langle\delta\mathbf{u},\mathbf{W}\mathbf{u}\rangle = \langle\mathbf{u},\mathbf{W}\delta\mathbf{u}\rangle$. Hence, we have

$$\delta E = -\langle\delta\mathbf{u},\mathbf{W}\mathbf{u}\rangle - \frac{1}{2}\langle\delta\mathbf{u},\mathbf{W}\delta\mathbf{u}\rangle.$$

We note that the nature of the algorithm is such that the sign of each component of $\delta\mathbf{u}$ is the same as that of the corresponding component of $\mathbf{W}\mathbf{u}$. Thus, the inner product $\langle\delta\mathbf{u},\mathbf{W}\mathbf{u}\rangle \geq 0$ for every state vector $\mathbf{u} \in \mathbb{B}^n$. Furthermore, if $\mathbf{W}$ is non-negative definite, the quadratic form $\langle\delta\mathbf{u},\mathbf{W}\delta\mathbf{u}\rangle \geq 0$. Thus $\delta E \leq 0$ and $E$ is a monotone non-increasing function

(b) In the synchronous mode, assume that the $k$th neuron updates itself at epoch $t$. We therefore have

$$u_i[t+1] - u_i[t] = \begin{cases} 0 & \text{if } i \neq k \\ -2u_k[t] & \text{if } i = k. \end{cases} \cdot$$

The change in energy $\delta E$ is given by

$$\delta E = -\delta u_k \left( \sum_{j=1}^{n} w_{kj} u_j \right) - \frac{1}{2} (\delta u_k)^2 w_{kk} \, .$$

Once again, the algorithm ensures that $\delta u_k$ is of the same sign as $\sum_{j=1}^{n} w_{kj} u_j$, and since $w_{kk}$ is non-negative, we see that $\delta E \leq 0$, and $E$ is montone non-increasing. ∎

**Proposition 2** *$F(\mathbf{u})$ is non-increasing in synchronous mode if $\mathbf{W}$ is symmetric.*

PROOF: In the synchronous mode of operation the change in the Manhattan Norm is given by

$$\begin{aligned} \delta F &= F(\mathbf{u}[t+1]) - F(\mathbf{u}[t]) \\ &= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} u_i[t+1] w_{ij} u_j[t] + \frac{1}{2} \sum_{p=1}^{n} \sum_{q=1}^{n} u_p[t] w_{pq} u_q[t-1]. \end{aligned}$$

As $\mathbf{W}$ is symmetric, we have $w_{pq} = w_{qp}$ and $\sum_{p=1}^{n} u_p[t] w_{qp} = \sum_{j=1}^{n} w_{ij} u_j[t]$. So

$$\delta F = -\frac{1}{2} \sum_{i=1}^{n} (u_i[t+1] - u_i[t-1]) \sum_{j=1}^{n} w_{ij} u_j[t].$$

Let $I$ be the set of indices for which $u_i[t+1] = -u_i[t-1]$ (Note that $I$ can be empty). Then

$$\delta F = -\sum_{i \in I} u_i[t+1] \sum_{j=1}^{n} w_{ij} u_j[t].$$

nce again we note that the nature of the algorithm guarantees us that the two sums in the above equation are of the same sign. Thus,

$$\delta F = -\sum_{i \in I} | \sum_{j=1}^{n} w_{ij} u_j[t] \leq 0,$$

and $F$ is a monotone non-increasing function along trajectories in state space. ∎

**Proposition 3** *The (stable state) capacity of the Outer Product Algorithm is $n/4\log n$.*

PROOF: We will prove only that $n/4\log n$ is a lower bound for the stable state capacity of the Outer Product Algorithm. Consider the random sums

$$X_n^{(i,\alpha)} = u_i^{(\alpha)} \sum_{j=1}^{n} w_{ij}^{op} u_j^{(\alpha)} = n + m - 1 + \sum_{j \neq i} \sum_{\beta \neq \alpha} u_i^{(\alpha)} u_i^{\beta} u_j^{(\alpha)} u_j^{(\beta)}, \qquad 1 \leq i \leq n, \quad 1 \leq \alpha \leq m.$$

Fix $i$ and $\alpha$, and write simply $X_n$ instead of $X_n^{(i,\alpha)}$. We can now write

$$X_n = n + m - 1 + \sum_{\beta \neq \alpha} \sum_{j \neq i} Z_j^{(\beta)},$$

where, for fixed $i$ and $\alpha$, we define the random variables $Z_j^{(\beta)} = u_i^{(\alpha)} i_i^{(\beta)} u_j^{(\alpha)} u_j^{(\beta)}$. Note that the random variables $Z_j^{(\beta)}$, $j \neq i$, $\beta \neq \alpha$ are i.i.d., symmetric, $\pm 1$ random variables.[5] Now recall that a simple application of Chebyshev's inequality yields that for any random variable $Y$ and any $t \geq 0$,

$$\mathbf{P}\{Y \leq -t\} \leq \inf_{r \geq 0} e^{-rt} \mathbf{E}(e^{-rY}).$$

[5] The critical fact here is that each random variable $Z_j^{(\beta)}$ has a distinct multiplicative term $u_j^{(\beta)}$ which occurs solely in the expression for $Z_j^{(\beta)}$.

Applying this result here we obtain

$$\begin{aligned} \mathbf{P}\{X_n \le 0\} &= \mathbf{P}\left\{ \sum_{\beta\neq\alpha} \sum_{j\neq i} Z_j^{(\beta)} \le -n - m + 1 \right\} \\ &\le \inf_{r\ge 0} e^{-r(n+m-1)}\, \mathbf{E}\left\{ e^{-r\sum_{\beta\neq\alpha}\sum_{j\neq i} Z_j^{(\beta)}} \right\} \\ &= \inf_{r\ge 0} e^{-r(n+m-1)}\, \mathbf{E}\left\{ \prod_{\beta\neq\alpha}\prod_{j\neq i} e^{-rZ_j^{(\beta)}} \right\}. \end{aligned}$$

The terms in the product, $e^{-rZ_j^{(\beta)}}$, $\beta \neq \alpha$, $j \neq i$ are independent random variables as the random variables $Z_j^{(\beta)}$ are independent. The expectation of the product of random variables above can, hence, be replaced by the product of expectations. Accordingly, denoting by $Z$ a random variable which takes on values -1 and 1 only, each with probabilty 1/2, we have

$$\mathbf{P}\{X_n \le 0\} \le \inf_{r\ge 0} e^{-r(n+m-1)} \left[\mathbf{E}(e^{-rZ})\right]^{(m-1)(n-1)} = \inf_{r\ge 0} e^{-r(n+m-1)} (\cosh r)^{(m-1)(n-1)}.$$

Now, for every $r \in \mathbb{R}$ we have $\cosh r \le e^{r^2/2}$. Hence

$$\mathbf{P}\{X_n \le 0\} \le \inf_{r\ge 0} \exp\left( \frac{r^2(m-1)(n-1)}{2} - r(n+m-1) \right) = \exp\left( -\frac{(n+m-1)^2}{2(n-1)(m-1)} \right).$$

We hence obtain that the probability that any given component of a memory is not stable is bounded by

$$\mathbf{P}\{X_n^{(1,\alpha)} \le 0\} \le e^{-n/2m}$$

for large enough $n$ provided $m$ grows so that $m = o(n)$ and $m = \omega(\sqrt{n})$.

Denote the event

$$\mathcal{E}_n = \left\{ \bigcup_{i=1}^{n} \bigcup_{\alpha=1}^{m} \{X_n^{(i,\alpha)} \le 0\} \right\}$$

that one or more of the $nm$ memory components is not stable. A simple application of Boole's inequality now yields

$$\mathbf{P}\{\mathcal{E}_n\} \leq \sum_{i=1}^{n}\sum_{\alpha=1}^{m} \mathbf{P}\{X_n^{(i,\alpha)} \leq 0\} \leq nm \exp\left\{-\left(\frac{n}{2m}\right)\right\}.$$

For a choice of

$$m = \frac{n}{4\log n}\left[1 + \frac{\log\log n + \log 4\epsilon}{2\log n} - O\left(\frac{\log\log n}{(\log n)^2}\right)\right]$$

we hence obtain that $\mathbf{P}\{\mathcal{E}_n\} \leq \epsilon$ as $n \to \infty$. As the probability that each of the memories is stable is exactly $1 - \mathbf{P}\{\mathcal{E}_n\}$, this establishes that the capacity is at least $n/4\log n$. ∎

**Proposition 4** *Let m increase with n such that $m \leq n$. Then the probability that a randomly chosen set of n-tuples* $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(m)} \in \mathbb{B}^n$ *is linearly independent aproaches one as* $n \to \infty$.

This result follows directly from a result of Komlós (1967) asserting that the probability that a random $n \times n$ ±1 matrix is nonsingular approaches one as $n \to \infty$.

# APPENDIX B: LINEAR PROGRAMMING

1. Specify $\mu_1, \ldots, \mu_k$, $k \leq n - m$, let $\mu_k + 1, \ldots, \mu_n - m < \epsilon$, and minimise $\epsilon$. The canonical form of the linear programming problem that the Simplex Method solves is:

   Minimise the *goal function* $\mathbf{c}^T\mathbf{y}$ subject to the constraints

$$\mathbf{A}\mathbf{y} = \mathbf{b}\,,$$

   where the vector $\mathbf{y}$ is unknown, and $\mathbf{y} > \mathbf{0}$.

In this case, we specify $k$ positive values of $\mathbf{M}_\mu$ and minimise the maximum of the $(n-k)$ unspecified values of $\mathbf{M}_\mu$ subject to the constraints $\mu_{k+1}, \ldots, \mu_n > 0$, and $c_1, \ldots, c_{n-m} > 0$. In other words, we have the following equations

$$\begin{array}{lll}
a_{1,1}c_1 & + \cdots + a_{1,n-m}c_{n-m} & = \mu_1 \\
 & \vdots & \\
a_{k,1}c_1 & + \cdots + a_{k,n-m}c_{n-m} & = \mu_k \\
a_{k+1,1}c_1 & + \cdots + a_{k+1,n-m}c_{n-m} & \leq \epsilon \\
 & \vdots & \\
a_{n,1}c_1 & + \cdots + a_{n,n-m}c_{n-m} & \leq \epsilon
\end{array}$$

where $c_i \geq 0$, $\epsilon > 0$, and we want to find $\mathbf{c}$ which minimises $\epsilon$.

To convert the $n-k$ inequalities to equalities, we subtract $\epsilon$ from both sides of the equation and add *slack variables* $z_1, \ldots, z_{n-k}$ to give us the following $n-k$ equations

$$\begin{array}{lll}
a_{k+1,1}c_1 + \cdots + & a_{k+1,n-m}c_{n-m} - \epsilon + z_1 & = 0 \\
 & \vdots & \\
a_{n,1}c_1 + \cdots + & a_{n,n-m}c_{n-m} - \epsilon + z_{n-k} & = 0 \, ,
\end{array}$$

in addition to the first $k$ equations. Now we have $n$ equations with $2n-m-k$ unknown non-negative quantities $(c_1, \ldots, c_{n-m}, z_1, \ldots, z_{n-k})$. Let us label $\epsilon$ as $c_0$. By inspection, we see that the goal function to be minimised is $c_0$, subject to the constraints $\mathbf{A}'\mathbf{c}' = \mathbf{M}_\mu{}'$ where $\mathbf{c}'$ is a $(2n-m-k+1)$-dimensional vector $\mathbf{M}_\mu{}'$ is a

$n$-dimensional vector, and $\mathbf{A}'$ is an $n \times (2n - m - k + 1)$ matrix; i.e., we require to solve

$$\begin{pmatrix} 0 & & 0 & & \\ \vdots & & & \ddots & \\ 0 & \mathbf{A} & & & 0 \\ -1 & & 1 & & \\ \vdots & & & \ddots & \\ -1 & & & & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-m} \\ z_1 \\ \vdots \\ z_{n-k} \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{12}$$

and $c_i, z_i \geq 0$. This is in the canonical form for the Simplex Method.

2. Specify $\mu_1, \ldots, \mu_n$, and

   (a) Minimise the mean-square error given by

   $$||\mathbf{Ac} - \mathbf{M}_\mu||^2 = \sum_{i=1}^{n} (a_{i,1}c_1 + \cdots + a_{i,n-m}c_{n-m} - \mu_i)^2$$

   subject to the constraints $\mathbf{M}_\mu > 0$, $\mathbf{c} > \mathbf{0}$.

   This is a quadratic programming problem. However, this problem can be reformulated as a Simplex Method problem and can be solved using a variation of the traditional simplex method called Wolfe's method (Wolfe, 1959).

   (b) Minimise the largest absolute error, $c_0$, given by

   $$\max(|\epsilon_1|, \ldots, |\epsilon_n|)$$

   where $\epsilon_i$, the error in $\mu_i$, is

   $$\mu_i - (a_{i,1}c_1 + \cdots + a_{i,n-m}c_{n-m}), \qquad i = 1, \ldots, n\,.$$

Our problem now is to minimize $c_0$ subject to $c_0, c_1, \ldots, c_{n-m} \geq 0$. To solve this problem,[6] we note that we have $n$ pairs of inequality contraints of the form

$$\begin{aligned} -c_0 + a_{i,1}c_1 + \cdots + a_{i,n-m}c_{n-m} &\leq \mu_i \\ -c_0 - a_{i,1}c_1 - \cdots - a_{i,n-m}c_{n-m} &\leq -\mu_i \end{aligned}$$

The addition of slack variables puts the problem in canonical form.

## References

Chvátal, V. (1983). *Linear Programming.* W. H. Freeman, New York.

Dembo, A. (1989). On the capacity of associative memories with linear threshold functions. *IEEE Trans. Inform. Theory,* **35**, 709–720.

Franklin, J. (1980). *Methods of Mathematical Economics.* Springer-Verlag, New York.

Goles, E., and Vichniac, G. Y. (1986). Lyapunov functions for parallel neural networks. *Neural Networks for Computing,* (ed. J. Denker). AIP, New York, **151**, 165–181.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sciences USA,* **79**, 2554–2558.

Komlós, J. (1967). On the determinant of (0,1) matrices. *Studia Scientarum Mathematicarum Hungarica,* **2**, 7–21.

Komlós, J., and Paturi, R. (1988). Convergence results in an associative memory model. *Neural Networks,* **1**, 239–250.

Maruani, A. D., Chevalier, R. C., and Sirat, G. (1987). Information retrieval in neural

[6] This is known as Chebyshev's Approximation (Franklin, 1980, p. 8).

networks. I. Eigenproblems in neural networks. *Rev. Phys. Appl.*, **22**, 1321–1325.

McCulloch, W. W., and Pitts, W. (1943). A logical calculus of the ideas immanent in neural activity. *Bull. Math. Biophys.*, **5**, 115–133.

McEliece, R. J., Posner, E. C., Rodemich, E. R., and Venkatesh, S. S. (1987). The capacity of the Hopfield associative memory. *IEEE Trans. Inform. Theory*, **33**, 461–482.

Murty, K. G. (1983). *Linear Programming.* Wiley, New York.

Personnaz, L., Guyon, I., and Dreyfus, G. (1985). Information storage and retrieval in spin-glass like neural networks. *J. Physique Lett.*, **46**, L359–L365.

Strang, G. (1980). *Linear Algebra and Its Applications.* Academic Press, New York.

Venkatesh, S. S., and Baldi, P. (1991a). Programmed interactions in higher-order neural networks: Maximal capacity. *J. Complexity*, **7**, 316–337.

Venkatesh, S. S., and Baldi, P. (1991b). Programmed interactions in higher-order neural networks: The outer product algorithm. *J. Complexity*, **7**, 443–479.

Venkatesh, S. S., Pancha, G., Psaltis, D., and Sirat, G. (1990). Shaping attraction basins in neural networks. *Neural Networks*, **3**, 613–623.

Venkatesh, S. S., and Psaltis, D. (1989). Linear and logarithmic capacities in associative neural networks. *IEEE Trans. Inform. Theory*, **35**, 558–568.

Wolfe, P. (1959). The simplex method for quadratic programming. *Econometrica*, **27**, 282–298.

## Figure Captions

**Fig. 1** Schematic representation of the directional attraction space around two memories with different eigenvalues with Spectral algorithms.

**Fig. 2** Schematic representation of the directional attraction space in the Dual Spectral Scheme with $\mu_y >> \mu_x, \mu_z$.

**Fig. 3** Schematic representation of the joint memory-specific and direction-specific attraction space for two memories in the Composite Scheme.

**Fig. 4** Average attraction radii around stable memories in the Outer-Product Scheme.

**Fig. 5** Attraction radii in the Spectral Scheme.

**Fig. 6** Variation of $\epsilon$, the largest of the unspecified directional parameters, $\mu_{k+1}, \ldots, \mu_n$, as a ratio of the specifed directional parameters with $k$, the number of directions specified.

**Fig. 7** Demonstration of direction-specific attraction in the Dual Spectral Scheme.

**Fig. 8** Demonstration of memory-specific and direction-specific attraction in the Composite Scheme.

Fig. 1

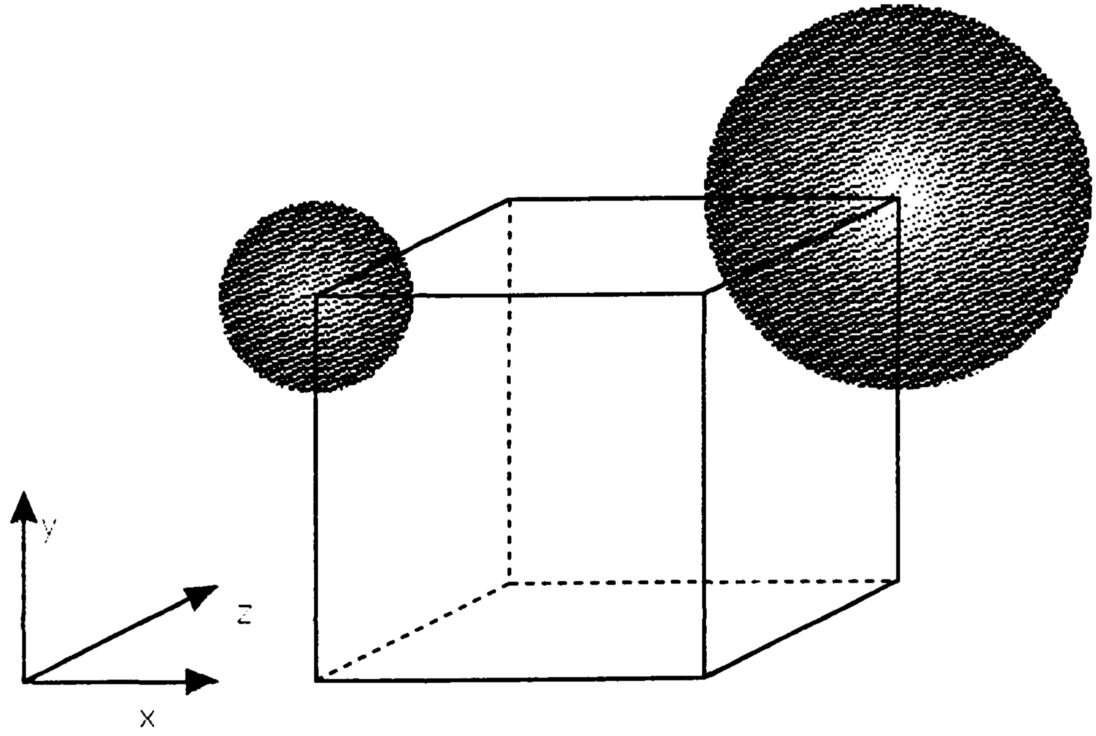


Fig. 2

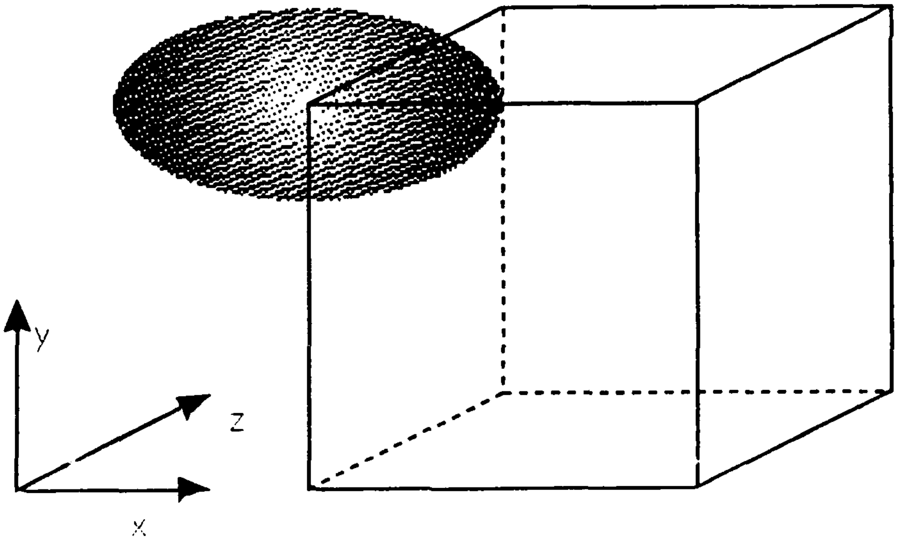
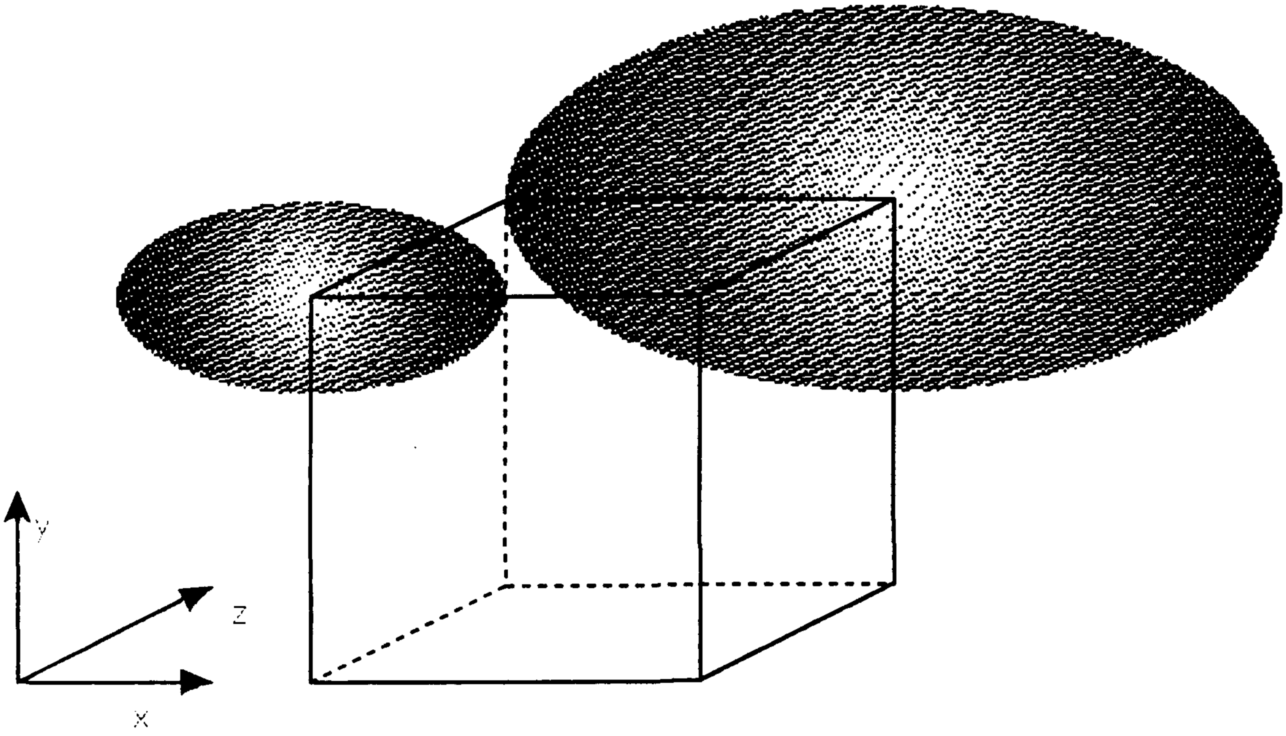


Fig. 3



~~fig7~~

Fig. 2

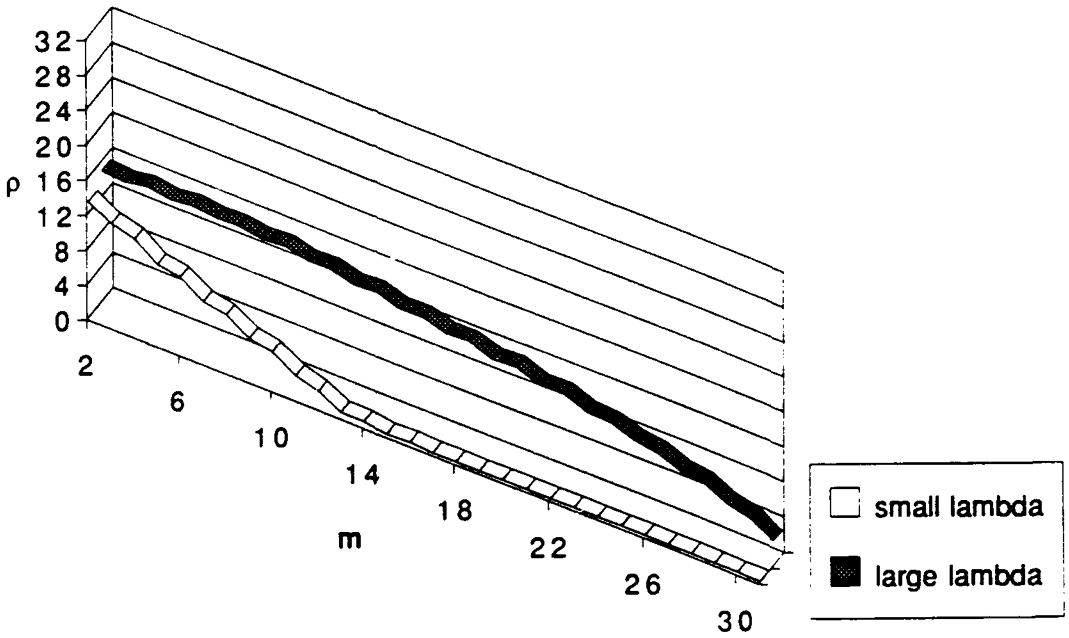
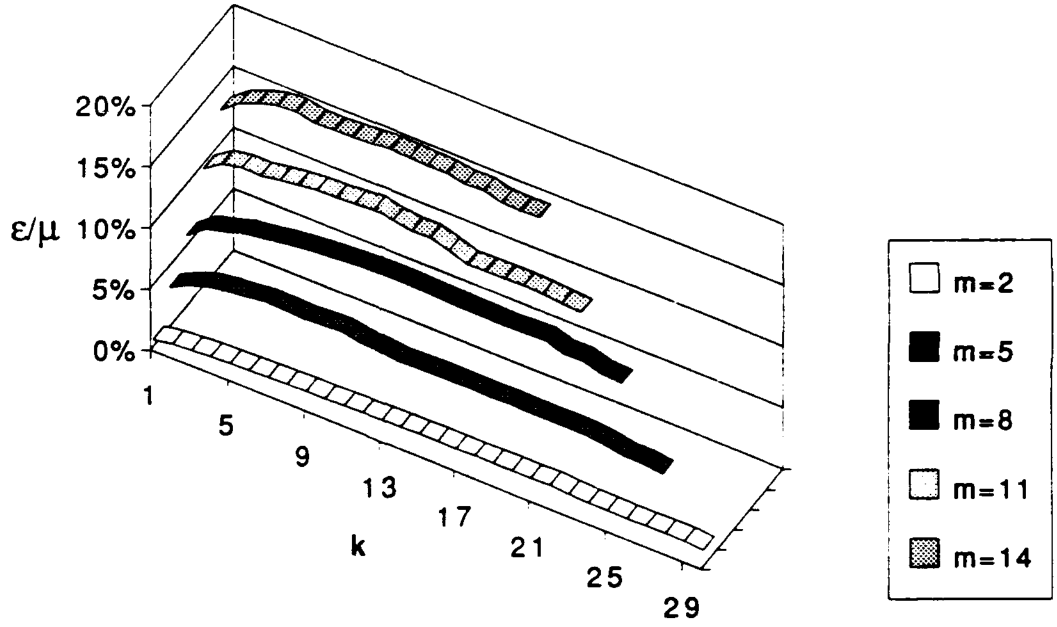

ρ
32
28
24
20
16
12
8
4
0
2
6
10
14
18
22
26
30
m
□ small lambda
▨ large lambda

Fig. 5

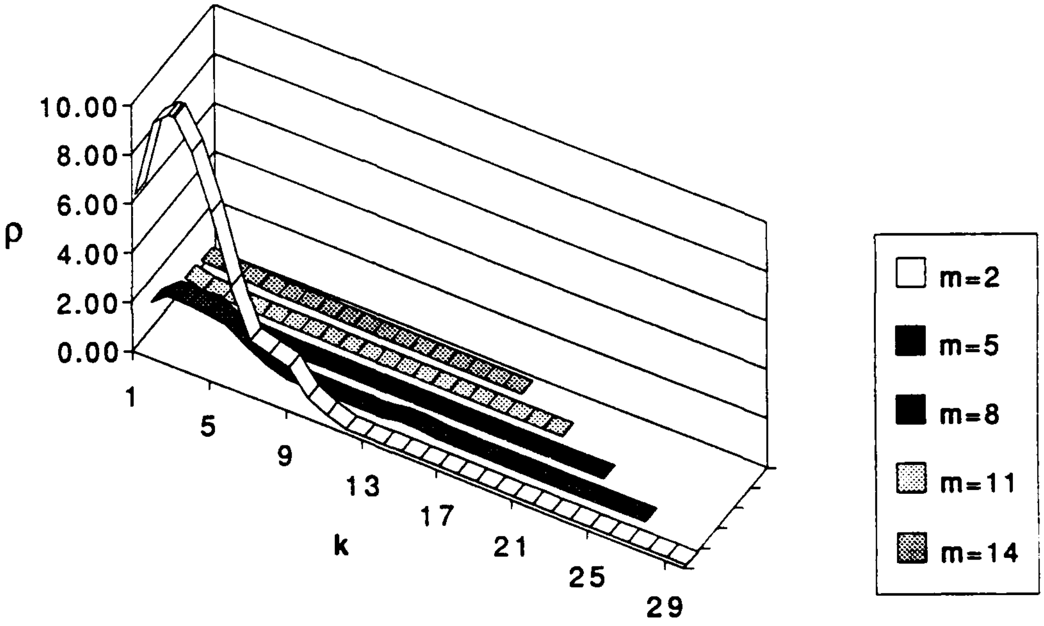
## Unspecified μ values vs. m, k



~~fig5a~~

Fig. 6(a)

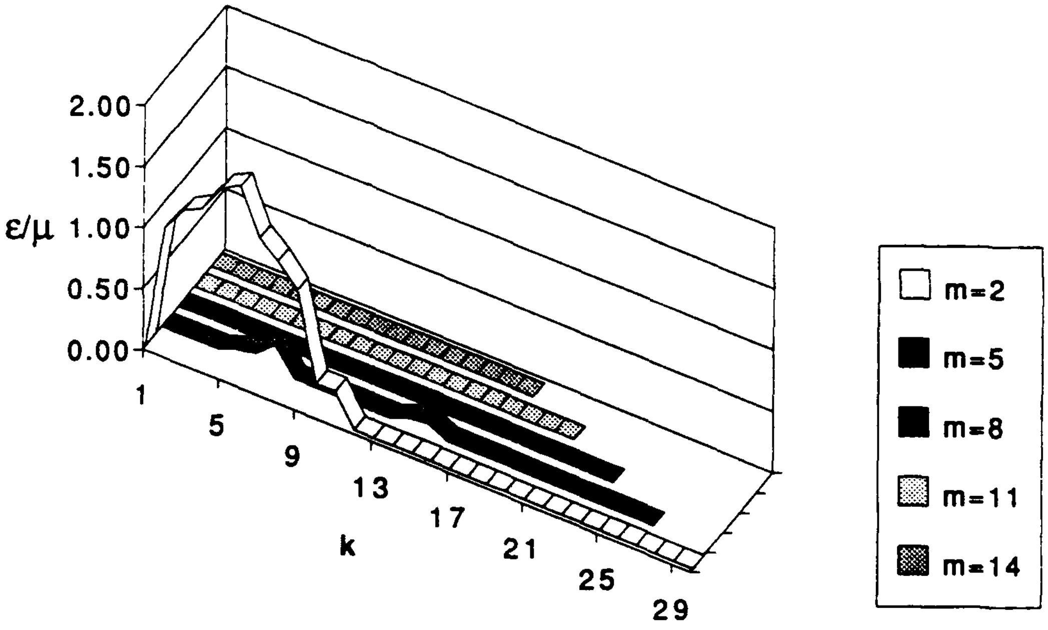
Dual Spectral Scheme, $\mu$=n



~~fig9b~~

Fig. 6(b)

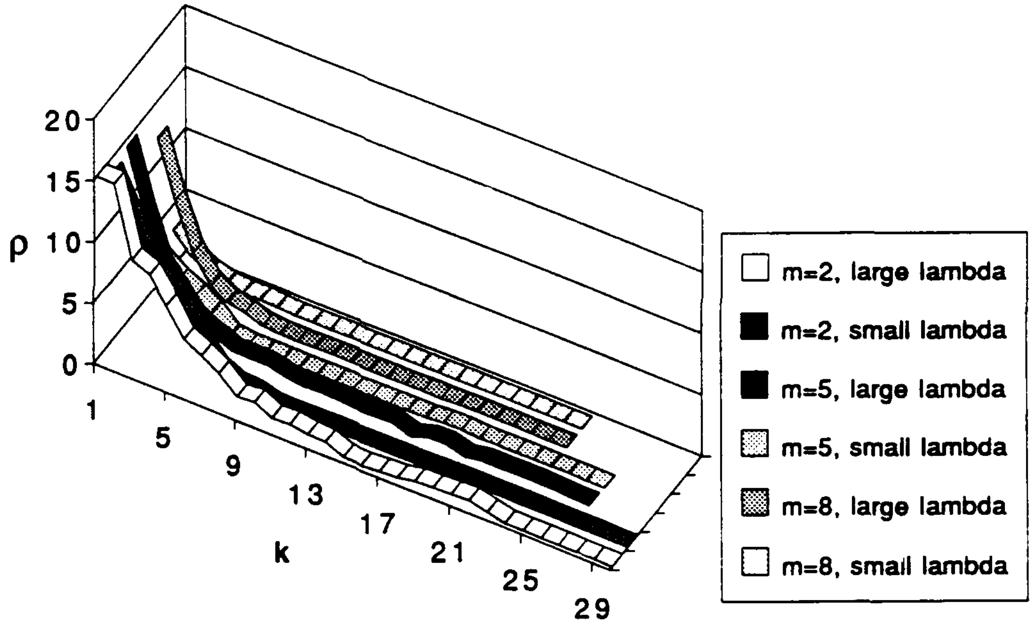
**Dual Spectral Scheme, μ=16n**



~~fig10a~~

Fig. 7(a)

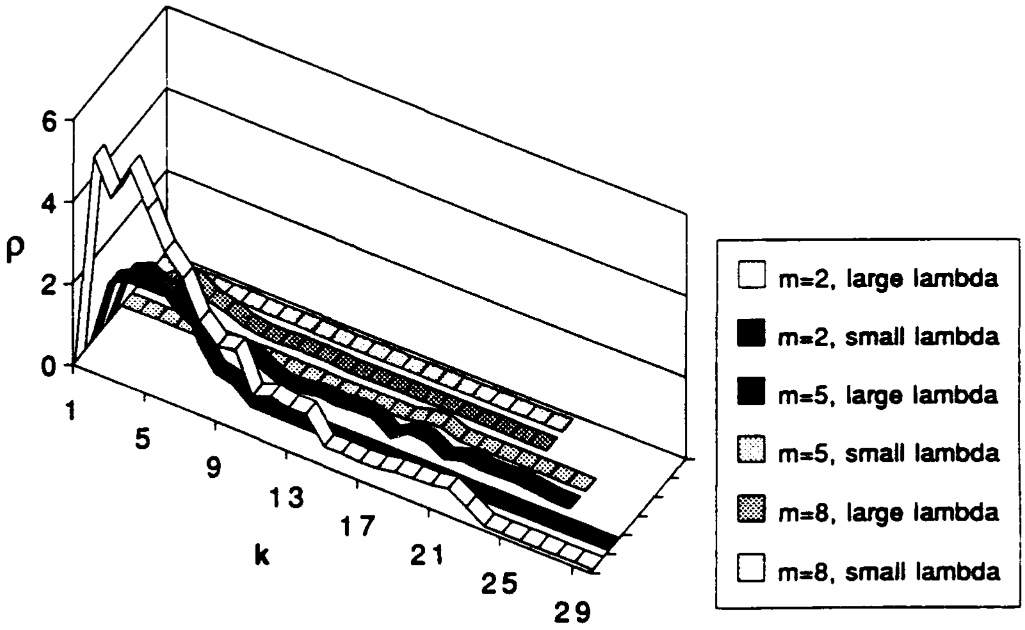
Composite scheme, μ=n



~~fig10b~~

Fig. 71

Composite scheme, μ=16n



# On Reliable Computation with Formal Neurons

Santosh S. Venkatesh and Demetri Psaltis

***Abstract*—We investigate the computing capabilities of formal McCulloch–Pitts neurons when errors are permitted in decisions. Specifically, given a random $m$-set of points $u^1, \cdots, u^m \in \mathbb{R}^K$, a corresponding $m$-set of decisions $d^1, \cdots, d^m \in \{-1, 1\}$, and a fractional error-tolerance $0 \leq \epsilon < 1$, we are interested in the following question: How large can we choose $m$ such that a formal neuron can make assignments $u^\alpha \to d^\alpha$, with no more than $\epsilon m$ errors? We obtain formal results for two protocols for error-tolerance—a *random error protocol* and an *exhaustive error protocol*.**

**In the random error protocol, a random subset of the $m$ points is randomly and independently specified and the associated decisions labeled "don't care." We prove that if $m$ is chosen less than $2n/(1 - 2\epsilon)$, then with high probability, there is a choice of weights for which the expected number of decision errors made by the neuron is no more than $\epsilon m$; if $m$ is chosen larger than $2n/(1 - 2\epsilon)$, then the probability approaches zero that there is a choice of synaptic weights for which the expected number of decision errors made by the neuron is fewer than $\epsilon m$.**

**In the exhaustive error protocol, the total number of decision errors has to lie below $\epsilon m$, but we are allowed to choose the set of decisions that are in error. We show that there is a function $1 \leq \kappa_\epsilon < 505$ such that if $m$ exceeds $2\kappa_\epsilon n/(1 - 2\epsilon)$, then there is, with high probability, no choice of synaptic weights for which a neuron makes fewer than $\epsilon m$ decision errors on the $m$-set of inputs. For small $\epsilon$, the function $\kappa_\epsilon$ is close to 1 so that, informally, we can specify $m$-sets as large as $2n/(1 - 2\epsilon)$ (but not larger) and obtain reliable decisions within the prescribed tolerance for some suitable choice of weights.**

***Index Terms*—Capacity, computation, fault-tolerance, formal neurons, large deviations, reliability.**

## I. Introduction

The formal modeling of biological neurons as linear threshold gates dates to the seminal paper of McCulloch and Pitts [3]. Although the biological plausibility of these models is open to debate, extensive investigations since the work of McCulloch and Pitts have shown that considerable computational power is latent in networks of formal neurons.

In its simplest form, a formal neuron is a computational device that accepts $n$ real inputs and produces a single bit output depending on whether a weighted sum of the inputs exceeds a fixed threshold. If the inputs are constrained to be Boolean variables, then the neuron simply realizes a Boolean function of $n$ variables.

A fundamental counting result (cf. Schläfli [4], Wendel [5], and Cover [6]) helps quantify the computational capability of a neuron: for any $m$ set in Euclidean $n$ space, the result gives a precise count of the number of dichotomies of the set that can be separated by a neuron. Each dichotomy is a collection of $m$ decisions made by the neuron on the $m$ set. Schläfli's theorem, therefore, gives the number of distinct sets of decisions that a neuron can make on a given set of data. The theorem can be used to estimate the maximum number of decisions that can be reliably made by a neuron; this number is linear in $n$, as we will see in the sequel.

In the considerations above, we have tacitly assumed that functions are computed without errors. Although this is the norm in most logical functions implemented on digital computers, computations involving cognitive tasks such as pattern recognition, however, are frequently less exact. It is hence reasonable to wonder whether allowing a formal neuron the latitude to make errors can substantially increase the set of problems that it can handle.[1]

Assume that a set of $m$ decisions are to be made on a randomly specified $m$ set of points in $n$ space and that we allow an error tolerance of $\epsilon m$ decision errors, with $0 \leq \epsilon < 1/2$. We are interested in how large we can choose $m$ such that the neuron makes reliable decisions within the prescribed error tolerance. A superficial consideration of the problem might indicate that substantial gains in computation may be achievable if errors are permitted, as the following analysis indicates. The number of ways that $\epsilon m$ errors can occur in decisions is $\binom{m}{\epsilon m}$ and corresponding to each such specification of the $\epsilon m$ incorrect decisions, there is a set of $(1 - \epsilon)m$ decisions that are required to be correct. Hence, the neuron is only required to realize any one of $\binom{m}{\epsilon m}$ distinct sets of $(1 - \epsilon)m$ decisions reliably. For large $m$, $\binom{m}{\epsilon m} = \Omega(2^{cm})$ for a positive constant $c$ so that there is an exponential number of distinct choices of which the neuron has to implement only one. It hence appears that there may potentially be substantive computational gains to be made if we allow some error tolerance in the decisions. Our main result in this paper, however, indicates that such gains are not actually realized, and the maximum number of decisions that can be made by a neuron under such circumstances remains linear in $n$; specifically, we prove the following results.

- The sequence $2n/(1 - 2\epsilon)$ is a threshold function[2] for the property that there is a choice of synaptic weights for which the neuron makes no more than (essentially) $\epsilon m$ random errors in decisions. In particular, if $m$ is less than $2n/(1 - 2\epsilon)$, then with high probability, there is a choice of weights for the neuron such that the expected number of errors is fewer than $\epsilon m$; if $m$ exceeds $2n/(1 - 2\epsilon)$, then the probability approaches zero that there is a choice of weights for the neuron such that the expected number of errors is fewer than $\epsilon m$.
- There is a function $1 \leq \kappa_\epsilon < 505$ such that if $m$ exceeds $2\kappa_\epsilon n/(1 - 2\epsilon)$, then there is (asymptotically) no choice of synaptic weights for which a neuron makes fewer than $\epsilon m$ decision errors on the $m$ set of inputs.

In the next section, we develop some notation and introduce the notions of $\epsilon$ reliability and capacity function. The main theorems

Manuscript received May 18, 1989; revised January 8, 1991. This work was supported by NSF grant EET-8709198 and Air Force grant AFOSR-89-0523 at the University of Pennsylvania and the Defense Advanced Research Projects Agency at the California Institute of Technology.

S. S. Venkatesh is with the Moore School of Electrical of Engineering, University of Pennsylvania, Philadelphia, PA 19104.

D. Psaltis is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125.

IEEE Log Number 9101447.

[1] Note that we assume that we have complete control over the neuron parameters and that errors creep into the neural output because we are overloading the capacity of the neuron. The notion of reliability of the decisions here is somewhat different from the case where decisions are unreliable because of a lack of control in the specification of the neuron (such as noise in the weights).

[2] The terminology *threshold function*, although standard in the probabilistic method, is a trifle unfortunate in the present context of linear threshold elements. We will replace it by the term *capacity function* in the next section.

0162–8828/92$03.00 © 1992 IEEE

are stated and proved in Section III; two technical results from large deviation probability theory are confined to the Appendix.

Several of the arguments used involve asymptotics, and we briefly sketch our use of notation here. If $\{x_n\}$ and $\{y_n\}$ are any two positive sequences, we say that

(a) $x_n = \Omega(y_n)$ if $x_n/y_n$ is bounded from below
(b) $x_n = O(y_n)$ if $x_n/y_n$ is bounded from above
(c) $x_n \sim y_n$ if $x_n/y_n$ approaches 1 for $n$ large enough
(d) $x_n = o(y_n)$ if $x_n/y_n$ approaches zero as $n \to \infty$.

If $A_n$ is a sequence of events in a probability space, we say that $A_n$ occurs with probability one if $P\{A_n\} \to 1$ as $n \to \infty$. By $\Phi$, we denote the Gaussian distribution function

$$\Phi(x)\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-y^2/2}\,dy.$$

By $b(k; N, p)$, we denote the probability that $N$ Bernoulli trials with probabilities $p$ for success and $(1-p)$ for failure result in $k$ successes and $N-k$ failures, i.e.,

$$b(k;\, N, p) = \binom{N}{k} p^k (1-p)^{N-k}.$$

We also denote by $\mathbb{B}$ the set $\{-1, 1\}$.

## II. Error Tolerance

A formal neuron is a linear threshold gate characterized by a vector of $n$ real weights $w = (w_1, \cdots, w_n)$ and a real threshold $w_0$. The neuron accepts as inputs points $\mathbf{u} \in \mathbf{R}^n$ and produces as output a single bit $d \in \mathbb{B}$, according to the following rule:

$$d = \operatorname{sgn}\left(\sum_{j=1}^{n} w_j u_j - w_0\right) = \begin{cases} 1 & \text{if } \sum w_j u_j \geq w_0 \\ -1 & \text{if } \sum w_j u_j < w_0. \end{cases}$$

We will without loss of generality assume that the threshold $w_0 = 0$.[3]

The neuron hence associates a *decision* or *classification* $(-1$ or $+1)$ with each point in $n$ space. For any given set of points then, the neuron dichotomizes the set of points into two classes—those points mapped to $+1$ and those mapped to $-1$. In the geometric analog the neuron represents a separating hyperplane in $n$ space. We are interested in characterizing the largest set of points for which there exist choices of weights such that almost any set of decisions associated with the specified set of points is realizable.

Let $\mathbf{u}^1, \cdots, \mathbf{u}^m \in \mathbf{R}^n$ be a random $m$ set of points chosen from any joint distribution invariant to reflection of components around the origin and such that every subset of $n$ points is linearly independent with probability one. Let $d^1, \cdots, d^m \in \mathbb{B}$ be a corresponding $m$ set of decisions. For a neuron to make *reliable* decisions, we must find a choice of weights that realize the assignments $\mathbf{u}^\alpha \to d^\alpha$ for each $\alpha = 1, \cdots, m$. Note that we can take all the decisions $d^\alpha$ to be $+1$ without any loss of generality because the vectors $\mathbf{u}^\alpha$ can always be reflected about the origin.

We incorporate error tolerance in decisions by introducing the notion of "*don't-care decisions*." Let $0 \leq \epsilon < 1/2$ be the fraction of errors that we are willing to allow in the decisions (i.e., we tolerate $\epsilon m$ errors in decisions). Let $D^\alpha, \alpha = 1, \cdots, m$ be the outcomes of $m$ identical and independent experiments whose outcomes are subsets of $\{-1, 1\}$, and such that

$$D^\alpha = \begin{cases} \{d^\alpha\} & \text{with probability } 1 - 2\epsilon \\ \{-1, 1\} & \text{with probability } 2\epsilon. \end{cases}$$

If a sample outcome $D^\alpha = \{d^\alpha\}$, then we require that the neuron produces the specified decision $d^\alpha$ as output whenever it receives $\mathbf{u}^\alpha$ as input. If, however, the sample outcome $D^\alpha = \mathbb{B}$, then we associate a don't-care decision with point $\mathbf{u}^\alpha$; the neuron can result in either $-1$ or $1$ as output when $\mathbf{u}^\alpha$ is input. We call $D^\alpha$ the *decision set* associated with decision $d^\alpha$; we say that $D^\alpha$ is *normal* if $D^\alpha = \{d^\alpha\}$ (i.e., the decision has to be accurate), and $D^\alpha$ is exceptional if $D^\alpha = \mathbb{B}$ (i.e., the decision is don't-care). The idea behind defining the decision sets in this fashion is the following. With the $m$ decision sets $D^1, \cdots, D^m$ generated independently according to the above prescription, the expected number of normal decision sets is $(1-2\epsilon)m$, whereas the expected number of exceptional decision sets is $2\epsilon m$. We now forget about those points corresponding to the exceptional decision sets and attempt to find neural weights that will correctly classify the remaining points corresponding to the normal decision sets. If we can successfully do this, then, beside the points corresponding to the normal decision sets, on average, one half of the points corresponding to the exceptional (don't-care) decision sets will serendipitously also turn out to be correctly classified. (Because the weights were chosen without taking the exceptional points into consideration, one half of them, on average, will be correctly classified as the points are chosen from a distribution that is invariant to reflections about the origin.) Thus, the expected number of errors in decision will only be $\epsilon m$. We formalize this notion of a random error protocol in the following:

**Definition 2.1:** Let $w_1, \cdots, w_n \in \mathbf{R}$ be the weights corresponding to a neuron. We say that the neuron makes $\epsilon$-reliable decisions on $\mathbf{u}^1, \cdots, \mathbf{u}^m$ if

$$\operatorname{sgn}\left(\sum_{j=1}^{n} w_j u_j^\alpha\right) \in D^\alpha, \qquad \alpha = 1, \cdots, m.$$

Note that by the Borel strong law, the fraction of don't-care decisions is almost surely $2\epsilon$. Further, because the vectors $\mathbf{u}^\alpha$ are invariant to reflections about the origin, the fraction of actual decision errors that occur for a neuron making $\epsilon$-reliable decisions is almost surely $\epsilon$. The case $\epsilon = 0$ reverts to the case of perfect decisions.

In the random error protocol, we are interested in the following attribute of the $m$ set of points $\mathbf{u}^1, \cdots, \mathbf{u}^m$ and the corresponding decisions:

> EVENT $\mathcal{F}_\epsilon(n, m)$: There is a choice of weights such that the neuron makes $\epsilon$-reliable decisions.

The attribute $\mathcal{F}_\epsilon(n, m)$ deals with the notion of reliable decisions on a random subset of points of expected size $(1-2\epsilon)m$.[4] The average number of errors allowed within this protocol is $\epsilon m$, but it is conceivable, albeit a rare occurrence, that the actual number of errors substantially exceeds $\epsilon m$. In the exhaustive error protocol, however, it is not permitted that the number of errors exceeds $\epsilon m$ substantially. There is no constraint, however, on the choice of which $(1-\epsilon)m$ decisions are to be correctly implemented, and we are

[3] In fact, it is easy to see that the threshold can be accommodated by allowing an additional *constant* input of $-1$ to a zero threshold neuron. Our results will then continue to hold for nonzero thresholds by replacing $n$ by $n+1$.

[4] The method of choosing decision sets advocated in this paper is not sacrosanct, and we could utilize any random strategy for choosing decision sets that yields an expected number of $(1-2\epsilon)m$ normal decision sets. We could, for instance, choose the random subset of normal decision sets from the uniform distribution on all subsets of size $(1-2\epsilon)m$ from the set of $m$ decisions; alternatively, we could replace the independent assignment of don't-cares in this paper by a Markovian strategy. The choice of the binomial distribution for specifying don't-cares in this paper was motivated in part because it is, in a sense, natural—the independent assignment of don't-cares from decision to decision avoids any bias due to prior don't-care assignments—and the fact that the computational complexity of choosing decision sets reduces to an exercise in coin flipping. The results of the next section hold for most reasonable choices of underlying distribution, resulting in an expected number of $(1-2\epsilon)m$ normal decision sets, although the technical details in the proofs can alter slightly.

free to exhaustively check each alternative of making $\epsilon m$ errors and choose the most favorable one. This protocol leads to a consideration of the following attribute of the $m$ set of points $\mathbf{u}^1, \cdots, \mathbf{u}^m$ and the corresponding decisions.

EVENT $\mathcal{G}_\epsilon(n,m)$: There is a choice of weights such that the neuron makes no more than $\epsilon m[1+o(1)]$ decision errors.

The attribute $\mathcal{G}_\epsilon(n,m)$ is somewhat stronger than the attribute $\mathcal{F}_\epsilon(n,m)$; instead of attempting to realize a random subset of decisions, we query whether it is possible to find a choice of weights for the neuron such that *at least one* of the $\binom{m}{\epsilon m}$ subsets of $(1-\epsilon)m$ decisions is reliably made.

**Definition 2.2:** Let $\mathcal{A}(n,m)$ be an attribute of the $m$ set of points $\mathbf{u}^1, \cdots, \mathbf{u}^m$. A sequence $\{C(n)\}_{n=1}^{\infty}$ is a *capacity function* for the attribute $\mathcal{A}(n,m)$ if for $\lambda > 0$ arbitrarily small: 1) $P\{\mathcal{A}(n,m)\} \to 1$ as $n \to \infty$ whenever $m \leq (1-\lambda)C(n)$, and 2) $P\{\mathcal{A}(n,m)\} \to 0$ as $n \to \infty$ whenever $m \geq (1+\lambda)C(n)$.

We say that $C(n)$ is a *lower capacity function* if it satisfies the first condition and that $C(n)$ is an *upper capacity function* if it satisfies the second condition.

The term *threshold function* is more standard in the literature of the probabilistic method when an attribute exhibits such a threshold behavior. Our definition is slightly stronger than is usual.

Capacity functions have been found for a variety of neural network architectures and algorithms [12], [7]–[13]. These investigations into network capacity have hitherto concentrated mainly on capacity functions for perfect decisions with no errors (cf. [9], [10], however, for results on error tolerance in the outerproduct algorithm). In the following, we expand on the results in [12] and show capacity functions for the attributes $\mathcal{F}_\epsilon(n,m)$ and $\mathcal{G}_\epsilon(n,m)$.

## III. Capacity Functions

### A. Error-Free Decisions

We begin by investigating the attribute $\mathcal{F}_0(n,m)$ that there is a neuron that makes reliable decisions $\mathbf{u}^\alpha \mapsto d^\alpha$ for each $\alpha = 1, \cdots, m$. The following fundamental result is due to Schläfli [4]: (Wendel [5] has a more accessible proof of the result; cf. also Cover [6].)

**Lemma 3.1:** The probability that there is a neuron that makes reliable decisions on a random $m$ set of points in Euclidean $n$ space $\mathbf{R}^n$ is given by

$$P\{\mathcal{F}_0(n,m)\} = \sum_{k=0}^{n-1} b(k;\ m-1, 0.5). \tag{1}$$

An application of Lemma A.2 directly yields the following:

**Theorem 3.2:** The sequence $C_0(n) = 2n$ is a capacity function for the attribute $\mathcal{F}_0(n,m)$.[5]

**Proof:** Fix $0 < \lambda < 1$ and let $H(x), 0 \leq x \leq 1$ denote the binary entropy function defined in Lemma A.2. With a choice of $m = \lfloor 2n(1-\lambda) \rfloor$ in (1), we can find $1/2 < c < 1$ such that

$$\begin{aligned} P\{\mathcal{F}_0(n,m)\} &\geq \sum_{k=0}^{c(m-1)} b(k;\ m-1, 0.5) \\ &\geq 1 - 2^{-[1-H(c)](m-1)} \to 1, \qquad n \to \infty \end{aligned}$$

with the second inequality following from Lemma A.2. Similarly, fo: a choice of $m = \lfloor 2n(1+\lambda) \rfloor$, we can find $0 < a < 1/2$ such that

$$\begin{aligned} P\{\mathcal{F}_0(n,m)\} &\leq \sum_{k=0}^{a(m-1)} b(k;\ m-1, 0.5) \\ &\leq 2^{-[1-H(a)](m-1)} \to 0, \qquad n \to \infty \end{aligned}$$

where we have used the dual form of Lemma A.2 (cf. remarks following the lemma). ■

### B. Epsilon Reliability

We now investigate the attribute $\mathcal{F}_\epsilon(n,m)$ that there is a choice of weights for which a neuron makes $\epsilon$-reliable decisions on the $m$ set of points $\mathbf{u}^1, \cdots, \mathbf{u}^m$.

**Theorem 3.3:** The sequence $C_\epsilon(n) = 2n/(1-2\epsilon)$ is a capacity function for the attribute $\mathcal{F}_\epsilon(n,m)$.

**Proof:** Let $0 \leq \epsilon < 1/2$ be the given tolerance. Noting that the decision sets are generated independently of the $m$ set of points, a direct application of Lemma 3.1 yields

$$P\{\mathcal{F}_\epsilon(n,m)\} = \sum_{k=0}^{m} b(k;\ m, 1-2\epsilon) P\{\mathcal{F}_0(n,k)\}. \tag{2}$$

We first claim that $P\{\mathcal{F}_0(n,k)\}$ is a monotone nonincreasing function of $k$ for each positive integer $n$. To show this, consider the difference $P\{\mathcal{F}_0(n,k)\} - P\{\mathcal{F}_0(n,k+1)\}$ for any choice of $k$ and $n$. Using (1) and elementary binomial identities, we have

$$\begin{aligned} P\{\mathcal{F}_0(n,k)\} &- P\{\mathcal{F}_0(n,k+1)\} \\ &= 2^{-k}\left[2\sum_{j=0}^{n-1}\binom{k-1}{j} - \sum_{j=0}^{n-1}\binom{k}{j}\right] \\ &= 2^{-k}\left[\sum_{j=0}^{n-1}\binom{k-1}{j} - \sum_{j=1}^{n-1}\binom{k-1}{j-1}\right] \\ &= 2^{-k}\binom{k-1}{n-1}. \end{aligned}$$

Hence, $P\{\mathcal{F}_0(n,k)\} - P\{\mathcal{F}_0(n,k+1)\} \geq 0$ for any choice of $k$ and $n$, and the claim is proved. Fix parameters $\lambda > 0$ and $1/2 < \nu < 2/3$. Now choose $m = 2n(1-\lambda)/(1-2\epsilon)$ and set

$$v_m = m(1-2\epsilon) + m^\nu\sqrt{2\epsilon(1-2\epsilon)}.$$

Using the monotonicity of $P\{\mathcal{F}_0(n,k)\}$ and (2), we obtain

$$P\{\mathcal{F}_\epsilon(n,m)\} \geq \underbrace{P\{\mathcal{F}_0(n,v_m)\}}_{A} \underbrace{\sum_{k=0}^{v_m} b(k;\ m, 1-2\epsilon).}_{B}$$

As $n \to \infty$, we then have from Theorem 3.2 that

$$A = P\{\mathcal{F}_0(n, \lfloor 2n(1-\lambda)(1+o(1))\rfloor)\} \to 1$$

while an application of Lemma A.1, and the choice $1/2 < \nu < 2/3$, yields

$$B \sim \Phi(m^{\nu-1/2}) = 1 - O\left(e^{-c_1 m^{2\nu-1}}\right)$$

for some positive constant $c_1$. Hence, for every $\lambda > 0$

$$P\{\mathcal{F}_\epsilon(n, \lfloor 2n(1-\lambda)/(1-2\epsilon)\rfloor)\} \to 1, \qquad \text{as } n \to \infty.$$

Now choose $m = 2n(1+\lambda)/(1-2\epsilon)$, and set

$$x_m = m(1-2\epsilon) - m^\nu\sqrt{2\epsilon(1-2\epsilon)}.$$

Again, using the monotonicity of $P\{\mathcal{F}_o(n,k)\}$ in (2), we have

$$P\{\mathcal{F}_\epsilon(n,m)\} \leq \underbrace{\sum_{k=0}^{x_m} b(k;\ m, 1-2\epsilon)}_{C} + \underbrace{P\{\mathcal{F}_0(n,x_m)\}}_{D}.$$

An application of Lemma A.1 and the choice $1/2 < \nu < 2/3$ yields that as $n \to \infty$

$$C \sim \Phi(-m^{\nu-1/2}) = O\left(e^{-c_2 m^{2\nu-1}}\right)$$

[5] cf. also a recent result due to Füredi [16] on random polytopes in the cube.

for some positive constant $c_2$. In addition, Theorem 3.2 yields

$$D = P\{\mathcal{F}_0(n, \lfloor 2n(1+\lambda)(1-o(1))\rfloor)\} \to 0.$$

It hence follows that for every choice of $\lambda > 0$

$$P\{\mathcal{F}_\epsilon(n, \lfloor 2n(1+\lambda)/(1-2\epsilon))\rfloor\} \to 0, \qquad \text{as } n \to \infty.$$

Thus, the sequence $2n/(1-2\epsilon)$ is both a lower and an upper capacity function (hence, a capacity) for the attribute $\mathcal{F}_\epsilon(n,m)$. ■

Let us now consider attribute $\mathcal{G}_\epsilon(n,m)$. The Borel strong law of large numbers yields that the fraction of exceptional decision sets is no more than $2\epsilon[1+o(1)]$ with probability one. For any choice of weights realizing $\epsilon$-reliable decisions on the $m$ set of points $\mathbf{u}^1, \cdots, \mathbf{u}^m$, it follows that the fraction of decision errors will be no more than $\epsilon[1+o(1)]$ with probability one as the $m$ set of points is chosen from a joint distribution invariant to reflections. The proof of the above theorem then directly yields

**Theorem 3.4:** The sequence $\underline{C}_\epsilon(n) = 2n/(1-2\epsilon)$ is a lower capacity function for the attribute $\mathcal{G}_\epsilon(n,m)$.

Now, consider all possible ways of assigning *at most* $\epsilon m[1+o(1)]$ errors in decision to the $m$ set of points. Attribute $\mathcal{G}_\epsilon(n,m)$ is realized if *at least* one of these possibilities is realizable for some appropriate choice of weights. (This clearly relaxes the more stringent requirements of attribute $\mathcal{F}_\epsilon(n,m)$, where we require that a *randomly chosen* one of these possibilities is realizable for an appropriate choice of weights.) Because the number of such possibilities is exponential, we might hope to realize significant gains in capacity for attribute $\mathcal{G}_\epsilon(n,m)$. The following result shows that we can hope to gain by, at best, a constant scale factor but that the linear rate of growth of the capacity function is unaffected.

**Theorem 3.5:** Let $\kappa_\epsilon$ be a function of the error tolerance $\epsilon$ defined by the unique solution of

$$H\left(\frac{1-2\epsilon}{2\kappa_\epsilon}\right) + H(\epsilon) = 1, \qquad 0 \le \epsilon < 1/2 \tag{3}$$

where $H$ is the binary entropy function. Then, the sequence $\overline{C}_\epsilon(n) = 2\kappa_\epsilon n/(1-2\epsilon)$ is an upper capacity function for the attribute $\mathcal{G}_\epsilon(n,m)$.

**Remarks:** The function $\kappa_\epsilon$ is defined as we vary $\epsilon$ in the interval $0 \le \epsilon < 1/2$ and monotonically increases from a value of $+1$ at $\epsilon = 0$ to a value close to 505 as $\epsilon$ approaches 1/2. For small $\epsilon$, it remains close to $+1$ so that the capacity function for the attribute $\mathcal{G}_\epsilon(n,m)$ still behaves like $2n/(1-2\epsilon)$.

**Proof:** Let us consider a tolerance of exactly $\epsilon m$ errors. The proof is not materially altered if the allowed tolerance is $\epsilon m[1+o(1)]$.

Let $P(n,m)$ denote the probability that there is a choice of weights for which a given set of $j$ decisions is made incorrectly, whereas the remaining $m-j$ decisions are made correctly. By Lemma 3.1,[6] we get that for any choice of $j$

$$P(n,m) = \sum_{k=0}^{n-1} b(k;\ m-1, 0.5). \tag{4}$$

Now, the number of ways in which $\epsilon m$ or fewer decision errors can be made is $\sum_{j=0}^{\epsilon m}\binom{m}{j}$; therefore, the union bound gives

$$P\{\mathcal{G}_\epsilon(n,m)\} \le P(n,m)\sum_{j=0}^{\epsilon m}\binom{m}{j}.$$

Let $\lambda > 0$ be fixed but arbitrary, and choose $m = 2\kappa_\epsilon n(1+\lambda)/(1-2\epsilon)$, where $\kappa_\epsilon$ is as defined in (3). Applying Lemma A.2 to (4), we obtain that

$$P(n,m) \le 2^{-(m-1)[1-H\{(1-2\epsilon)/2\kappa_\epsilon(1+\lambda)\}]}$$

while another application of Lemma A.2 gives

$$\sum_{j=0}^{\epsilon m}\binom{m}{j} \le 2^{mH(\epsilon)}$$

for large enough $m$. Hence, for each choice of $0 \le \epsilon < 1/2$ and $\lambda > 0$, there is a choice of $\beta > 0$ such that

$$P\{\mathcal{G}_\epsilon(n,m)\} \le \beta 2^{-m[1-H\{(1-2\epsilon)/2\kappa_\epsilon(1+\lambda)\}-H(\epsilon)]}.$$

The binary entropy function $H(c)$ increases monotonically from a value of 0 at $c=0$ to a value of 1 at $c=1/2$. Hence, with $\kappa_\epsilon$ as in (3), $H\{(1-2\epsilon)/2\kappa_\epsilon(1+\lambda)\} + H(\epsilon) < 1$. Hence, for every choice of $\lambda > 0$ and $m = 2\kappa_\epsilon n(1+\lambda)/(1-2\epsilon)$, there is a choice of $\delta > 0$ such that $P\{\mathcal{G}_\epsilon(n,m)\} < 2^{-\delta m} \to 0$ as $n \to \infty$. ■

## IV. Conclusion

The results proved in this paper demonstrate that a formal neuron has a computational capacity that is linear in $n$ and that this rate of growth of capacity persists even when errors are tolerated in the decisions. A question that arises at this juncture is how this result bears on computations involving networks of formal neurons. In particular, for an associative memory model composed of $n$ densely interconnected formal neurons, the rigorous determination of the maximal storage capacity when errors are permitted in recall is an open question. We analyze this in a subsequent paper [17] (cf. also [2]).

## Appendix
## Large Deviations

We quote two technical lemmas from large deviation probability theory that we will need. Both results concern probabilities in the tails of the binomial distribution. Lemma A.1 provides a good uniform estimate for the cumulative distribution of a sum of $N$ independent (0, 1) random variables valid for deviations from the mean as large as $o(N^{2/3})$ (instead of the $O(\sqrt{N})$ deviations encountered in the usual central limit theorem). The approximation is the strong form of the large deviation central limit theorem [14]. Lemma A.2 is due to Chernoff [15] and estimates probabilities in the extreme tails (deviations of the order of $N$ from the mean) of the binomial distribution.

**Lemma A.1:** Let $0 < p < 1$, and let $\{v_N\}$ be a sequence such that $|v_N - Np| \le K(N) = o(N^{2/3})$. Then

$$\sum_{k=0}^{v_N} b(k; N, p) \sim \Phi\left(\frac{v_N - Np}{\sqrt{Np(1-p)}}\right), \qquad N \to \infty.$$

If, in addition, $v_N - Np = \Omega(N^\nu)$, for some $1/2 < \nu < 2/3$, then

$$\sum_{k=0}^{v_N} b(k; N, p) = 1 - O\left(e^{-\delta N^{2\nu-1}}\right)$$

with $\delta > 0$, which is a constant.

**Lemma A.2:** *Let $0 < p < 1$ be fixed, and let $T_p$ and $H$ be real-valued functions on the closed interval $[0,1]$ defined for $0 \le c \le 1$ by*

$$T_p(c) = -c\log_2 p - (1-c)\log_2(1-p)$$
$$H(c) = -c\log_2 c - (1-c)\log_2(1-c).^{7}$$

[6] Fixing a set of $j$ points $\mathbf{u}^{\alpha_1}, \cdots, \mathbf{u}^{\alpha_j}$ which are incorrectly classified, is equivalent to specifying the corresponding decisions to be $-d^{\alpha_1}, \cdots, -d^{\alpha_j}$; this, however, just yields a different dichotomy of $m$ points in $n$ space, and Schläfli's lemma applies.

[7] We define $H(c) = 0$ when $c = 0$ or $c = 1$.

Then, for every choice of $c \in (p, 1)$ we have

$$\sum_{k=0}^{\lfloor cN \rfloor} b(k; N, p) \geq 1 - 2^{-N[T_p(c) - H(c)]}.$$

**Remarks:** The quantity $H$ is the *binary entropy function.* Note that $T_p(c) \geq H(c)$ for all $c$ (with equality only for $c = p$) so that Chernoff's bound yields an exponentially small probability for the extreme tails. The bound is, in fact, exponentially tight.

For the special case $p = 1/2$, Chernoff's bound yields

$$\sum_{k=0}^{\lfloor cN \rfloor} b(k; N, 0.5) \geq 1 - 2^{-N[1 - H(c)]}$$

for any choice of $1/2 < c \leq 1$. Note also that by the symmetry of the binomial distribution, we have

$$\sum_{k=0}^{\lceil aN \rceil} b(k; N, 0.5) \leq 2^{-N[1 - H(a)]}$$

for any choice of $0 \leq a < 1/2$.

## Acknowledgment

One of the authors would like to thank Prof. J. Komlós for bringing the problem of exhaustive error tolerance to his attention. We would also like to thank the anonymous referees whose suggestions have contributed materially to the clarity of the paper.

## References

[1] S. S. Venkatesh, "Linear maps with point rules: Applications to pattern classification and associative memory," Ph.D. thesis, California Inst. Technol., 1986.

[2] S. S. Venkatesh, "Epsilon capacity of neural networks," in *NeuralNetworks for Computing* (J. Denker, Ed.). New York: AIP, 1986.

[3] W. W. McCulloch, and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[4] L. Schläfli, *Gesammelte Mathematische Abhandlungen I.* Basel, Switzerland: Verlag Birkhäuser, pp. 209–212, 1950.

[5] s

[6] J. G. Wendel, "A problem in geometric probability," *Math Scand.*, vol. 11, pp. 109–111, 1962.

[7] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *Electron. Comput.*, vol. 14, pp. 326–334, 1965.

[8] Y. S. Abu-Mostafa and J. St. Jacques, "Information capacity o. the Hopfield model," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 461–464, 1985.

[9] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 461–482, 1987.

[10] J. Komlós and R. Paturi, "Convergence results in an associative memory model," *Neural Networks*, vol. 1, no. 3, pp. 239–250, 1988.

[11] C. M. Newman, "Memory capacity in neural network models: rigorous lower bounds," *Neural Networks*, vol. 1, no. 3, pp. 233–238, 1988.

[12] P. Baldi and S. S. Venkatesh, "On properties of networks of neuron-like elements," in *Neural Information Processing Systems* (D. Z. Anderson, Ed.). New York: AIP, 1988.

[13] D. Psaltis and S. S. Venkatesh, "Information storage in fully interconnected networks," in *Evolution, Learning and Cognition* (Y. C. Lee, Ed.). New York: World Scientific, 1989.

[14] S. S. Venkatesh and D. Psaltis, "Linear and logarithmic capacities in associative neural networks," *IEEE Trans. Inform. Theory*, vol. 35, pp. 558–568, 1989.

[15] W. Feller, *An Introduction to Probability Theory and its Applications,* vol. I. New York: Wiley, 1968.

[16] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Ann Math. Stat.*, vol. 23, pp. 493–507, 1952.

[17] Z. Füredi, "Random polytopes in the $d$-dimensional cube," *Discrete Comput. Geom.*, vol. 1, pp. 315–319, 1986.

[18] S. S. Venkatesh, "The science of making *erors,*" to be published in *IEEE Trans. Knowledge Data Eng.*, Mar. 1992.

IEEE Transactions on Knowledge
and Data Engineering
vol. 4, no. 2, pp. 135-144
April 1992 [invited paper]

# THE SCIENCE OF MAKING *ERORS*: WHAT ERROR TOLERANCE IMPLIES FOR CAPACITY IN NEURAL NETWORKS[†]

Santosh S. Venkatesh[‡]
Department of Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104
*E-Mail*: venkatesh@ee.upenn.edu

1 July 1991

INDEX TERMS: *Neuron, Neural Networks, Error Tolerance, Capacity, Associative Memory*

**Abstract**

How does an allowed tolerance for error in output affect the computational capability of a neural network and the ability of the network to learn an underlying problem structure? The subject of this communication is the development of formal protocols for handling error tolerance which allow of a precise determination of the computational gains that may be expected. The error protocols are illustrated in the framework of a densely interconnected neural network architecture (with associative memory the putative application), and rigourous calculations of capacity are shown. Explicit capacities are also derived for the case of feedforward neural network configurations.

[†] Presented in part at the *Workshop on Neural Networks for Computing*, Snowbird, Utah, 1986 [1].

[‡] The research reported here was supported by the Air Force Office of Scientific Research under grant AFOSR 89-0523.

# 1 INTRODUCTION

How does an allowed tolerance for error in output affect the computational capability of a neural network and the ability of the network to learn an underlying problem structure? In this paper we attempt to mathematically codify the computational gains that are realised when errors are permitted in the network output. Such error tolerance may occur naturally in applications such as associative memory or pattern classification where we do not insist on accurately classifying every feature; alternatively, error tolerance may be forced upon us* by the inability of the neural architecture under consideration to respond accurately to *every* instance of a specific problem: as in, for example, attempting to classify two non-linearly separable classes of points with a single separating plane (or equivalently, a single McCulloch-Pitts neuron).†

Anecdotal evidence exists for the premise that an allowed error tolerance can have a significant effect on computational capability. Consider, for instance, an associative memory application where it is desired to store memories as attractors in recurrent neural networks whereby a linear number of component errors in any memory are corrected and the memory retrieved. Rigourous investigations by McEliece, *et al* [2] and Komlós and Paturi [3] show that the outer-product algorithm for storing memories in a recurrent network of $n$ neurons stores exactly of the order of $n/\log n$ memories when it is required that each memory be retrieved *exactly* with no component errors. However, in earlier work, Hopfield [4] reports empirical evidence indicating that it may be possible to store a number of memories linear in $n$ with the outer-product algorithm if errors are permitted in the retrieved memories; this was formally verified subsequently by Newman [5] who demonstrated a lower bound linear in $n$ on the memory storage capacity if errors are permitted in retrieval. Thus, an allowed error tolerance effects a substantial improvement in storage capacity from sub-linear in $n$ to at least linear in $n$ for the outer-product algorithm.

Our purpose here is to attempt to quantify the maximal gains in capacity that can accrue for *any* algorithm if errors are allowed in the outputs of a given neural network architecture. In particular, for recurrent neural networks of $n$ neurons (and any choice of algorithm) we settle the issue of whether Newman's linear lower bounds can be substantially improved upon to allow memory storage capacities which increase faster than linearly in $n$ if errors are permitted in the recall of the stored memories.

Let us consider the associative memory application in some more detail. Let $\mathbf{u}^1, \ldots, \mathbf{u}^m$ be a random

*Some men are born into errors, others achieve errors, and some have errors thrust upon 'em.

†In the classical modeling of McCulloch and Pitts, a formal neuron is a linear threshold element which produces a binary output according to the sign of a linear form of the inputs.

selection of $m$ memories drawn from the vertices of the cube $\{-1,1\}^n$. For associative memory it is desired that a random probe differing from a memory in no more than $\rho n$ components (for some choice of $0 \leq \rho < 1/2$) is mapped into the memory. In other words, we require the memories to be ***attractors*** over a Hamming ball of radius $\rho n$.

Consider now a recurrent neural network of $n$ neurons where, at any epoch, the binary $n$-tuple of neural outputs is fed back to constitute the neural inputs for the next epoch. The degrees of freedom in this dynamical system reside in the specification of the weighting factors linking neural outputs to inputs. Each specification of interconnection weights results in the specification of a dynamics characterised by a family of trajectories in a state space of the vertices of the cube $\{-1,1\}^n$. The storage of memories as attractors in such a structure is accomplished if, for a choice of interconnection weights, trajectories originating in a Hamming ball of radius $\rho n$ at the memories ultimately terminate at the memories. As we will see later in Corollary 4.2, it is not possible to store more than a linear number (in $n$) of random memories as attractors in a recurrent neural network of $n$ neurons.

An allowed error tolerance in this context permits a fraction $\epsilon$, $0 \leq \epsilon \leq \rho < 1/2$ of errors in the retrieval of any memory. This situation corresponds to requiring points in the Hamming ball of radius $\rho n$ at a memory to be ultimately mapped into a (smaller) ball of radius $\epsilon n$ at the memory. This is indicated schematically in Figure 1. The capacity question is now to determine the largest allowable rate of growth of the number of memories $m$ with the number of neurons $n$ such that it can be guaranteed with high probability that trajectories originating in a Hamming ball of radius $\rho n$ at any memory are ultimately confined within a Hamming ball of radius $\epsilon n$ at the memory. The following plausibility argument indicates that the flexibility inherent in allowing error tolerance may result in substantial gains in capacity over the error free case.

Let us simplify the problem by considering trajectories originating at a memory, say $\mathbf{u}^\alpha \in \{-1,1\}^n$. If the trajectory is to be confined within an $\epsilon n$ ball at the memory, then clearly a necessary condition is that the *first* synchronous step in the trajectory not lead to a point outside the ball of radius $\epsilon n$ at the memory. However, any transition $\mathbf{u}^\alpha \mapsto \mathbf{v}^\alpha$ where $\mathbf{v}^\alpha$ differs from the memory $\mathbf{u}^\alpha$ in no more than $\epsilon n$ components is an admissible transition (see Figure 2). Thus, for the $m$ memories, there are a total of $\left[\sum_{j=0}^{\epsilon n} \binom{n}{j}\right]^m$ admissible $m$-sets of first synchronous transitions originating at the memories corresponding to the number of different ways of specifying "first transition points" in the $m$ Hamming balls of radius $\epsilon n$ at the memories. Let us now concentrate on the problem of realising an admissible set of $m$ first transitions (one for each memory) within a recurrent neural network structure. This is clearly a necessary prelude to the larger problem of associative memory with error tolerance in the following sense: if $M(\epsilon, n)$ is the largest rate of growth of $m$ with $n$ for which it can be guaranteed with high probability that there exists a set of neural interconnection weights for a recurrent neural network which yields an admissible $m$-set of first synchronous transitions originating at the memories, then $M(\epsilon, n)$ gives an upper bound for the number of memories that can be stored with an error tolerance of $\epsilon n$ components in retrieval.

Our basic question now is the following: For a random selection of $m$ memories, $\mathbf{u}^\alpha \in \{-1,1\}^n$, $\alpha = 1, \dots, m$, and a given (fractional) error tolerance $0 \leq \epsilon < 1/2$, is there any choice of neural interconnection weights for a recurrent neural network which will result in an admissible $m$-set of first synchronous transitions, $\mathbf{u}^\alpha \mapsto \mathbf{v}^\alpha$, $\alpha = 1, \dots, m$, where each $\mathbf{v}^\alpha$ differs from the corresponding memory $\mathbf{u}^\alpha$ in no more than $\epsilon n$ components? Now it is easy to verify using Stirling's formula that the number of admissible $m$-sets of first synchronous transitions from the memories is $\Omega(2^{c(\epsilon)mn})$, where $c(\epsilon)$ is a fixed positive function of $\epsilon$. For a large choice of $m$, any given $m$-set of admissible transitions, $\mathbf{u}^1 \mapsto \mathbf{v}^1, \dots, \mathbf{u}^m \mapsto \mathbf{v}^m$, will have only a small probability of being realised within a recurrent neural network architecture. However, there are an exponential number of possible sets of admissible transitions so that the probability that one or more of these sets of transitions is realisable in a recurrent network may be large even if the probability of realising any individual set of transitions is small. It hence appears that potentially large gains in capacity may be possible if errors are allowed in retrieving memories.

A similar plausibility argument could be made for the improvement in capacity of any neural network architecture if a fraction $\epsilon$ of the network outputs can be in error. For instance, if we are interested in realising a set of desired assignments $\mathbf{u}^\alpha \mapsto f(\mathbf{u}^\alpha)$, $\alpha = 1, \dots, m$ (where $f$ is some underlying function from which random examples are drawn) in a feedforward neural architecture, allowing up to $\epsilon m$ of these assignments to be in error again gives an exponential (in $m$) number of choices for specifying incorrect assignments.

The above specious arguments notwithstanding, the main results of this paper indicate, however, that error tolerance does not buy order of magnitude improvements in capacity over the error free case in neural network architectures: in general, error tolerance results in an improvement in the multiplicative constants, but does not change the rate of growth of capacity.

*Organisation*: In the next section we set up the formal neural model in the framework of a fully-interconnected network architecture for definiteness. We also introduce two protocols for error—a random and an exhaustive error protocol—and define the formal notion of capacity. The definitions here follow those developed in Venkatesh and Psaltis [6] in the analysis of reliability and error tolerance issues for computations with a single neuron. In Section 3 we state some preliminary technical lemmas which are central to the ensuing development. In Section 4 we state and prove the main theorems on the capacity of fully-interconnected, recurrent neural networks when there is a tolerance for output error. Finally, in Section 5 we briefly indicate the extensions of these results to feedforward neural network architectures.

*On Notation*: $\mathbb{B}$ denotes the set $\{-1, 1\}$; the function $\operatorname{sgn} : \mathbb{R} \to \mathbb{B}$ is defined by $\operatorname{sgn} x = x/|x|$ if $x \neq 0$, with the nonce convention $\operatorname{sgn} 0 = 1$; logarithms are to base $e$; for any positive integer $k$, $[k]$ denotes the set $\{1, \ldots, k\}$; $c_1, c_2, c_3, \ldots$ represent absolute positive constants; for any $\mathbf{u} \in \mathbb{B}^n$ and any $r > 0$, $B(\mathbf{u}, r)$ denotes the Hamming ball of radius $r$ at $\mathbf{u}$, i.e., the set of all points in $\mathbb{B}^n$ which differ from $\mathbf{u}$ in $r$ or fewer components; the probability that $N$ Bernoulli trials with probabilities $p$ for success and $1 - p$ for failure result in $k$ successes and $N - k$ failures is denoted by $b(k; N, p)$:

$$b(k; N, p) = \binom{N}{k} p^k (1-p)^{N-k}.$$

We will also have recourse to the following asymptotic order notations. If $\{x_n\}$ and $\{y_n\}$ are positive sequences, we denote: $x_n = O(y_n)$ if there exists $K < \infty$ such that $x_n/y_n \leq K$ for all $n$; $x_n = \Omega(y_n)$ if there exists $L > 0$ such that $x_n/y_n \geq L$ for all $n$; and $x_n = o(y_n)$ if $x_n/y_n \to 0$ as $n \to \infty$.

# 2 ERROR TOLERANCE: PROTOCOLS

## 2.1 Formal Neurons and Networks

A *formal neuron* is a linear threshold element accepting $n$ inputs which produces a binary output according to the sign of a linear form of the inputs. In particular, a neuron is characterised by a vector of $n$ real *weights*, $\mathbf{w} = (w_1 \cdots w_n)$, and, given as input a vector $\mathbf{u} = (u_1 \cdots u_n)$, produces a binary output $v = \operatorname{sgn} \sum_{i=1}^n w_i u_i$.[1] The neuron hence associates a *decision* or *classification*, -1 or +1, with each point in $n$-space. For any given set of points then, the neuron dichotomises the set of points into two classes—those points mapped to +1 and those mapped to -1. In the geometric analogue the neuron represents a separating hyperplane in $n$-space. Let $\mathbf{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ be an $m$-set of points in $n$-space, and, corresponding to each $\mathbf{u}^\alpha \in \mathbf{U}$, let $v^\alpha \in \mathbb{B}$ be a desired decision. The $m$-set of decisions naturally dichotomises $\mathbf{U}$ into two sets $(\mathbf{U}^+, \mathbf{U}^-)$, where $\mathbf{u}^\alpha \in \mathbf{U}^+$ if $v^\alpha = 1$ and $\mathbf{u}^\alpha \in \mathbf{U}^-$ if $v^\alpha = -1$. We say that the dichotomy $(\mathbf{U}^+, \mathbf{U}^-)$ is *separable by a neuron* iff there exists a weight vector $\mathbf{w} \in \mathbb{R}^n$ such that

$$\sum_{i=1}^n w_i u_i^\alpha \begin{cases} \geq 0 & \text{if } \mathbf{u}^\alpha = (u_1^\alpha \cdots u_n^\alpha) \in \mathbf{U}^+, \\ < 0 & \text{if } \mathbf{u}^\alpha = (u_1^\alpha \cdots u_n^\alpha) \in \mathbf{U}^-, \end{cases}$$

i.e., $\operatorname{sgn} \sum_{i=1}^n w_i u_i^\alpha = v^\alpha$ for $\alpha = 1, \ldots, m$.

We will be concerned here with a fully-interconnected, recurrent network of $n$ neurons where the neural outputs at any epoch are fed back to constitute the neural inputs for the next epoch. In particular, for any $i \in [n]$, neuron $i$ is characterised by a set of $n - 1$ weights $\{w_{ij} : j \neq i, j \in [n]\}$, and if the vector

[1] This is the model of McCulloch and Pitts. A real threshold is allowed within the model but is not critical to the present discussion.

$\mathbf{u} = (u_1 \cdots u_n) \in \mathbb{B}^n$ denotes the neural outputs at any epoch, at the next epoch the $i$th neuron then produces a binary output $u_i' = \operatorname{sgn} \sum_{j \neq i} w_{ij} u_j$.[§] The system hence evolves (synchronously) in a state space of vertices of the cube $\mathbb{B}^n$, and is completely characterised by the zero-diagonal matrix of neural interconnection weights $[w_{ij}]$.

Let $\mathbf{u}^1, \ldots, \mathbf{u}^m$ be a random $m$-set of memories chosen independently from the vertices of the cube. In particular, the memory components $\{u_i^\alpha : i \in [n], \alpha \in [m]\}$ are i.i.d. random variables drawn from a sequence of symmetric Bernoulli trials:

$$\mathbf{P}\{u_i^\alpha = 1\} = \mathbf{P}\{u_i^\alpha = -1\} = 1/2, \qquad i \in [n], \quad \alpha \in [m].$$

In an associative memory application, a basic deisderatum would be that all the memories are fixed points—$\mathbf{u}^\alpha \mapsto \mathbf{u}^\alpha$, $\alpha = 1, \ldots, m$—of the network, i.e., that there exists a zero diagonal matrix of weights $[w_{ij}]$ such that

$$\operatorname{sgn} \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} u_j^\alpha = u_i^\alpha, \qquad i \in [n], \quad \alpha \in [m].$$

If now there is an allowed tolerance for error, the fixed point requirement for the memories could be relaxed to allow "admissible" first synchronous transitions of the form $\mathbf{u}^\alpha \mapsto \mathbf{v}^\alpha$. By "admissible" we mean as before that the number of components in which the points $\mathbf{v}^\alpha \in \mathbb{B}^n$ are allowed to differ from the corresponding memories $\mathbf{u}^\alpha$ must be within a given error tolerance. We are interested in estimating the largest allowable rate of growth of $m$ with $n$ for which there exists a zero-diagonal network which realises $m$ admissible synchronous transitions $\mathbf{u}^\alpha \mapsto \mathbf{v}^\alpha$ with high probability as $n$ grows large. In the following we define two formal error protocols which provide different notions of "admissibility."

## 2.2 Random Error Protocol

We begin by defining a protocol which randomly specifies which components of a memory are allowed to be in error by randomly labeling a set of memory components as *don't-cares*. Let $0 \leq \epsilon < 1/2$ be the fraction of errors that we are willing to allow in the retrieval of any memory. For each $i \in [n]$ let $\{D_i^\alpha : \alpha \in [m]\}$ be the outcomes of $m$ identical, and independent experiments whose outcomes are subsets of $\{-1, 1\}$, and such that

$$D_i^\alpha = \begin{cases} \{u_i^\alpha\} & \text{with probability } 1 - 2\epsilon, \\ \{-1, 1\} & \text{with probability } 2\epsilon. \end{cases}$$

[§] To avoid trivial complications, we assume that there is no self feedback from the output of any neuron to its input. This corresponds to setting the weights $w_{ii} \equiv 0$ for $i = 1, \ldots, n$.

If a sample outcome $D_i^\alpha = \{u_i^\alpha\}$, then we require that the $i$th neuron retrieve the $i$th component of memory $\mathbf{u}^\alpha$; if, however, the sample outcome $D_i^\alpha = \mathbb{B}$, then we associate a don't-care decision with point $\mathbf{u}^\alpha$: the neuron can result in either -1 or 1 as output when $\mathbf{u}^\alpha$ is input. We call $D_i^\alpha$ the ***decision set*** associated with memory component $u_i^\alpha$; we say that $D_i^\alpha$ is ***normal*** if $D_i^\alpha = \{u_i^\alpha\}$ (i.e., memory component $u_i^\alpha$ has to be retrieved by the $i$th neuron), and $D_i^\alpha$ is ***exceptional*** if $D_i^\alpha = \mathbb{B}$ (i.e., the decision is don't-care). The idea behind defining the decision sets in this fashion is the following. For each $\alpha \in [m]$, the decision sets $\{D_i^\alpha : i \in [n]\}$ are generated independently according to the above prescription so that the expected number of normal decision sets is $(1-2\epsilon)n$ while the expected number of exceptional decision sets is $2\epsilon n$ for each memory. If now, ignoring components corresponding to exceptional decision sets, we design a zero-diagonal weight matrix to retrieve all components corresponding to the normal decision sets, then on average one-half of the components corresponding to exceptional decision sets will also be retrieved so that the expected number of errors in the retrieval of any memory will be only $\epsilon n$.

**Definition 2.1** For each $i \in [n]$, let $\{w_{ij} : j \neq i\}$ be the set of $n-1$ weights corresponding to the $i$th neuron. We then say that the $i$th neuron makes ***$\epsilon$-reliable decisions*** on the set of memories $\{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$ if

$$\operatorname{sgn} \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} u_j^\alpha \in D_i^\alpha, \qquad \alpha = 1, \ldots, m.$$

If all the neurons make $\epsilon$-reliable decisions, then, by the Borel strong law, the fraction of component errors in the retrieval of any memory is $\epsilon$ almost surely. We are hence interested in the following attribute of the $m$-set of memories.

> EVENT $\mathcal{R}_\epsilon(n,m)$ *[Random Error Protocol with Parameter $\epsilon$]: For each $i \in [n]$, there is a choice of weights for the $i$th neuron such that the neuron makes $\epsilon$-reliable decisions on the set of memories $\{\mathbf{u}^1, \ldots, \mathbf{u}^m\}$.*

The attribute $\mathcal{R}_\epsilon(n,m)$ deals with the notion of random synchronous transitions from the memories by specifying a random choice of (on average $\epsilon n$) component errors for each memory. The computational gains we may expect from this protocol arise from the large number of "typical" transitions that can be specified.

## 2.3 Exhaustive Error Protocol

Consider now a protocol where the number of component errors in a memory in a single synchronous transition is not permitted to exceed $\epsilon n$; however, there is no specification or constraint on which components in a memory are allowed to be in error, and we are free to examine all alternatives of specifying $\epsilon n$ or fewer component errors in each memory in one synchronous step. This protocol leads to a consideration of the following attribute of the $m$-set of memories.

EVENT $\mathcal{E}_\epsilon(n,m)$ *[Exhaustive Error Protocol with Parameter* $\epsilon$*]: For each* $\alpha \in [m]$ *there exists a vertex* $\mathbf{v}^\alpha \in B(\mathbf{u}^\alpha, \epsilon n)$ *such that the set of synchronous transitions* $\{\mathbf{u}^\alpha \mapsto \mathbf{v}^\alpha : \alpha \in [m]\}$ *is realised for some choice of zero-diagonal weight matrix* $[w_{ij}]$ *for a fully-interconnected network of* $n$ *neurons.*

The attribute $\mathcal{E}_\epsilon(n,m)$ is somewhat stronger than the attribute $\mathcal{R}_\epsilon(n,m)$; instead of specifying a random set of essentially $2\epsilon n$ don't-care components for a memory (resulting in essentially $\epsilon n$ component errors), we are now allowed to examine all transitions resulting in $\epsilon n$ or fewer component errors for each memory and choose the most favourable specification of errors. As we saw in the Introduction, this allows us any choice from among an exponentially large number of admissible $m$-sets of first synchronous transitions originating at the memories.

## 2.4 Capacity Functions

The notion of capacity of a fully-interconnected neural network that we espouse is, loosely speaking, the "largest number" of random memories that can be "stored" in the network. The precise meaning we attach to "storage" of a memory depends upon the attribute of interest, such as: all memories are fixed points; almost all memories are attractors over a radius $\rho n$; there are no more than $\epsilon n$ component errors in retrieving any memory. We will be interested in particular in the attributes $\mathcal{R}_\epsilon(n,m)$, the random error protocol with parameter $\epsilon$, and $\mathcal{E}_\epsilon(n,m)$, the exhaustive error protocol with parameter $\epsilon$. The following definition captures the notion of "largest number of memories" as a threshold function of a relevant attribute. The notion is explored in somewhat greater generality in [7].

**Definition 2.2** Let $\mathcal{A}(n,m)$ be an attribute of the $m$-set of memories $\mathbf{u}^1, \ldots, \mathbf{u}^m$. A sequence $\{C(n)\}_{n=1}^{\infty}$ is a *capacity function* for the attribute $\mathcal{A}(n,m)$ if for $\lambda > 0$ arbitrarily small, as $n \to \infty$:

a) $\mathbf{P}\{\mathcal{A}(n,m)\} \to 1$ whenever $m \leq (1-\lambda)C(n)$;

b) $\mathbf{P}\{\mathcal{A}(n,m)\} \to 0$ whenever $m \geq (1+\lambda)C(n)$.

We say that $C(n)$ is a *lower capacity function* if it satisfies the first condition, and that $C(n)$ is an *upper capacity function* if it satisfies the second condition.

Capacity functions have been found for a variety of neural network architectures and algorithms (a survey can be found in Venkatesh [7]). These investigations into network capacity, however, have hitherto concentrated mainly on capacity functions for perfect decisions with no errors (cf. [3, 5], however, for results on error tolerance in the outer-product algorithm). In the following we expand on our results in [1, 6] and show capacity functions for the attributes $\mathcal{R}_\epsilon(n,m)$ and $\mathcal{E}_\epsilon(n,m)$.

## 3 TECHNICAL PRELIMINARIES

Our basic technique is to replace the geometrical notion of trajectories within Hamming balls in $n$-space by calculations involving the tails of binomial distributions. The following is a classical result due to Chernoff [8] which asserts exponential bounds for the binomial tails for linear deviations from the mean.

**Lemma 3.1** *Let* $0 < p < 1$ *be fixed, and let* $T_p$ *and* $H$ *be real-valued functions on the closed interval* $[0,1]$ *defined for* $0 \le c \le 1$ *by*

$$\begin{aligned} T_p(c) &= -c \log p - (1-c)\log(1-p), \\ H(c) &= -c \log c - (1-c)\log(1-c).[¶] \end{aligned}$$

*Then for every choice of* $c \in (p,1]$ *and every integer* $N$ *we have*

$$\sum_{k=0}^{\lfloor cN \rfloor} b(k;N,p) \ge 1 - e^{-N[T_p(c)-H(c)]}.$$

REMARKS: $H$ is the *binary entropy function* which takes values in $[0, \log 2]$. Note that for any choices of $c$ and $p$, Jensen's inequality yields

$$H(c) - T_p(c) = c \log \frac{p}{c} + (1-c)\log\frac{1-p}{1-c} \le \log 1 = 0$$

with equality holding only when $c = p$. Hence $T_p(c) > H(c)$ whenever $c \ne p$. Chernoff's bound hence yields exponentially small probabilities for the extreme tails of the binomial distribution. This bound can be shown to be exponentially tight (see Blake and Darabian [9], for instance).

For the special case $p = 1/2$, Chernoff's bound yields

$$\sum_{k=0}^{\lfloor cN \rfloor} b(k;N,0.5) \ge 1 - e^{-N[\log 2 - H(c)]}$$

for any choice $1/2 < c \le 1$.

[¶] We define $H(c) \equiv 0$ when $c = 0$ or $c = 1$.

For any $m$-set of points $\mathbf{U} \in \mathbb{R}^N$, $|\mathbf{U}| = m$, let $\mathcal{D}(\mathbf{U})$ denote the family of dichotomies of $\mathbf{U}$ that can be separated by a neuron: a dichotomy $(\mathbf{U}^+, \mathbf{U}^-)$ of $\mathbf{U}$ is in $\mathcal{D}(\mathbf{U})$ if and only if there exists a weight vector $\mathbf{w} = (w_1 \cdots w_N) \in \mathbb{R}^N$ such that

$$\sum_{i=1}^{N} w_i u_i \begin{cases} \geq 0 & \text{if } \mathbf{u} = (u_1 \cdots u_N) \in \mathbf{U}^+, \\ < 0 & \text{if } \mathbf{u} = (u_1 \cdots u_N) \in \mathbf{U}^-. \end{cases}$$

The following estimate for the number of dichotomies separable by a neuron was given by Schläfli [10] using an elegant combinatorial argument. (For more accessible proofs of the result see Wendel [11] or Cover [12].)

**Lemma 3.2** *Let $\mathbf{U} \in \mathbb{R}^N$ be an m-set of points. The following estimate holds:*

$$|\mathcal{D}(\mathbf{U})| \leq D(N,m) = 2\sum_{i=0}^{N-1} \binom{m-1}{i}.$$

*Furthermore, the upper bound of $D(N,m)$ is achieved for m-sets of points in general position.*[‖]

A fundamental parameter of interest to us is the probability that a neuron can separate a random dichotomy of a random set of vertices of the cube $\mathbb{B}^N$. Let $\mathbf{u}^1, \ldots, \mathbf{u}^m$ be a randomly drawn set of patterns *from the vertices of the cube* $\mathbb{B}^N$, and let the pattern components $\{u_i^\alpha : i \in [N], \alpha \in [m]\}$ form a sequence of symmetric Bernoulli trials:

$$\mathbf{P}\{u_i^\alpha = -1\} = \mathbf{P}\{u_i^\alpha = +1\} = 1/2, \qquad i \in [N], \quad \alpha \in [m].$$

To each pattern $\mathbf{u}^\alpha$ associate a desired classification $v^\alpha \in \mathbb{B}$ specified independently of $\mathbf{u}^\alpha$. We are interested in the probability $P(N,m)$ that there exists a choice of weight vector $\mathbf{w} \in \mathbb{R}^N$ such that

$$\operatorname{sgn}\left(\sum_{i=1}^{N} w_i u_i^\alpha\right) = v^\alpha, \qquad \alpha = 1, \ldots m.$$

The following asymptotic estimate for $P(N,m)$ was shown by Füredi [13].

**Lemma 3.3** *If $m = O(N)$ as $N \to \infty$ then*

$$P(N,m) = \sum_{j=0}^{N-1} b(j; m-1, 0.5) - O(e^{-c_1 N}).$$

REMARKS: Note that Lemma 3.2 guarantees that $2^{-m}D(N,m)$ is an upper bound for $P(N,m)$. The asymptotic order correction to this estimate in Lemma 3.3 arises because the probability that a random $m$-set of *vertices* from $\mathbf{B}^N$ is in general position rapidly becomes small when $m$ exceeds $N$. The exponentially small order term quoted above is a strengthening of Füredi's original estimate of $O(N^{-1/2})$. The refinement was

[‖] An $m$-set of points in $N$-space is in *general position* iff any subset of $N$ or fewer of the points is linearly independent.

made possible by a new result due to Kahn, Komlós, and Szemerédi [14] which asserts that the probability that a random $N \times N$ $\pm 1$ matrix is singular decreases exponentially fast with $N$. Incorporating this result in Füredi's proof (without any other change) gives the quoted improvement. We will need the stronger form of the result.

A direct application of Chernoff's bound to the above estimate for $P(N, m)$ yields the following

**Lemma 3.4** *For every fixed $\lambda > 0$ we can find absolute positive constants $c_2$ and $c_3$ such that as $N \to \infty$:*

$$\begin{aligned} P(N, 2N(1-\lambda)) &= 1 - O(e^{-c_2 N}), \\ P(N, 2N(1+\lambda)) &= O(e^{-c_3 N}). \end{aligned}$$

This is the well known result that a formal neuron can separate a random dichotomy of up to $2N$ patterns. Using elementary binomial identities it is easy to verify the following "monotone property."

**Lemma 3.5** *If $k = O(n)$ as $N \to \infty$ then*

$$\left| P(N, k) - P(N, k+1) - 2^{-k} \binom{k-1}{N-1} \right| = O(e^{-c_4 N}).$$

In particular, $P(N, k)$ is a monotone non-increasing function of $k$ in the vicinity of $2N$ for large enough $N$. Note that when $k = (2-\delta)N$ or $k = (2+\delta)N$ then Stirling's formula gives $|P(N, k) - P(N, k+1)| = O\left(e^{-c_4' N}\right)$.

# 4 ERROR TOLERANCE: CAPACITIES

## 4.1 Random Error Protocol

We first consider the computational gains that are feasible under a random error protocol with parameter $\epsilon \in [0, 1/2)$. For any fixed $i \in [n]$, let $\{D_i^\alpha : \alpha \in [m]\}$ be the sequence of decision sets corresponding to neuron $i$. Recall that the decision sets are drawn independently according to a sequence of Bernoulli trials, and

$$D_i^\alpha = \begin{cases} \{u_i^\alpha\} & \text{with probability } 1 - 2\epsilon, \\ \{-1, 1\} & \text{with probability } 2\epsilon. \end{cases}$$

**Theorem 4.1** *For any fixed error parameter $0 \le \epsilon < 1/2$, the sequence $\frac{2n}{1-2\epsilon}$ is a capacity function for $\mathcal{R}_\epsilon(n, m)$, the random error protocol with parameter $\epsilon$.*

PROOF: The $i$th neuron makes $\epsilon$-reliable decisions if there is a choice of $n-1$ weights $\{w_{ij} : j \in [n] \setminus \{i\}\}$ such that

$$\operatorname{sgn}\left(\sum_{j \neq i} w_{ij} u_j^\alpha\right) \in D_i^\alpha, \qquad \alpha = 1, \ldots, m.$$

Alternatively, let $A = \{\alpha : D_i^\alpha \text{ is normal}\}$ be the (random) set of indices identifying memories whose $i$th component must be retrieved. The above is then equivalent to requiring that

$$\operatorname{sgn}\left(\sum_{j \neq i} w_{ij} u_j^\alpha\right) = u_i^\alpha, \qquad \alpha \in A.$$

Note that as a consequence of the zero-diagonal nature of the network, the term $u_i^\alpha$ is absent in the sum above. By the independence of the memory components, if $|A| = k$ then the above is equivalent to finding a weight vector in $(n-1)$-space which separates a randomly and independently specified dichotomy of a set of $k$ vertices chosen randomly from $\mathbb{B}^{n-1}$. It is hence clear that $P(n-1, k)$ is the probability that the $i$th neuron makes $\epsilon$-reliable decisions *conditioned* upon there being $k$ normal decision sets and $m-k$ exceptional decision sets. As the distribution of normal and exceptional decision sets is governed by the binomial distribution it follows that the probability $P_\epsilon(n, m)$ that the $i$th neuron makes $\epsilon$-reliable decisions is given by

$$P_\epsilon(n, m) = \sum_{k=0}^{m} b(k; m, 1-2\epsilon)\, P(n-1, k). \tag{1}$$

By Boole's inequality, the probability $1 - \mathbf{P}\{\mathcal{R}_\epsilon(n, m)\}$ that one or more neurons fails to make $\epsilon$-reliable decisions is bounded by

$$1 - \mathbf{P}\{\mathcal{R}_\epsilon(n, m)\} \leq n\,[1 - P_\epsilon(n, m)]\,.$$

Also, the probability $\mathbf{P}\{\mathcal{R}_\epsilon(n, m)\}$ that all the neurons make $\epsilon$-reliable decisions is clearly bounded above by the probability $P_\epsilon(n, m)$ that a single neuron makes $\epsilon$-reliable decisions. Combining this with the above inequality we have the two-sided bounds:

$$1 - n\,[1 - P_\epsilon(n, m)] \leq \mathbf{P}\{\mathcal{R}_\epsilon(n, m)\} \leq P_\epsilon(n, m). \tag{2}$$

Now let $0 < \lambda < 2\epsilon$ be fixed but arbitrary. With the choice

$$m = (1 - \lambda)\frac{2n}{1 - 2\epsilon} \tag{3}$$

we have

$$P_\epsilon(n, m) \geq \sum_{k=0}^{m(1-2\epsilon)(1+\lambda/2)} b(k; m, 1-2\epsilon)\, P(n-1, k) = 1 - O(e^{-c_5 n}) \qquad (n \to \infty).$$

The first inequality is obvious as it arises from the deletion of terms in the sum in (1). This lower bound for $P_\epsilon(n,m)$ can be seen to approach 1 exponentially fast as asserted above by two appeals to Lemma 3.1: with $m$ increasing with $n$ as in (3), $P(n-1,k) = 1 - O\left(e^{-c_5' n}\right)$ for $k$ in the range $0 \leq k \leq m(1-2\epsilon)(1+\lambda/2)$; further, $\sum_{k=0}^{m(1-2\epsilon)(1+\lambda/2)} b(k;m,1-2\epsilon) = 1 - O\left(e^{-c_5'' n}\right)$. It follows from the lower bound of (2) that

$$\mathbf{P}\left\{\mathcal{R}_\epsilon(n,m)\right\} = 1 - O\left(n\,e^{-c_5 n}\right) \qquad (n \to \infty)$$

for $m$ growing as in (3). As $\lambda$ is arbitrary, $2n/(1-2\epsilon)$ is a lower capacity function for $\mathcal{R}_\epsilon(n,m)$.

Now choose $m$ growing with $n$ such that

$$m = (1+\lambda)\frac{2n}{1-2\epsilon}. \tag{4}$$

We then have

$$\begin{aligned} P_\epsilon(n,m) &\leq \sum_{k=0}^{m(1-2\epsilon)(1-\lambda/2)} b(k;m,1-2\epsilon) \;+\; P(n-1, m(1-2\epsilon)(1-\lambda/2)) \;+\; O\left(e^{-c_6' n}\right) \\ &= O\left(e^{-c_6 n}\right) \qquad (n \to \infty). \end{aligned}$$

The first inequality follows from Lemma 3.5 and elementary considerations; the exponential decrease of the upper bound to zero is readily ascertained by applying Lemma 3.1 twice, as before. The upper bound of (2) hence yields

$$\mathbf{P}\left\{\mathcal{R}_\epsilon(n,m)\right\} = O\left(e^{-c_6 n}\right) \qquad (n \to \infty)$$

for $m$ growing as in (4). As $\lambda$ is arbitrary, $2n/(1-2\epsilon)$ is also an upper capacity function, hence a capacity function, for $\mathcal{R}_\epsilon(n,m)$. ∎

The case where each memory is required to be a fixed point of the network corresponds to the choice of error parameter $\epsilon = 0$. The following conclusion is hence immediate:

**Corollary 4.2** *The sequence $2n$ is a capacity function for the attribute $\mathcal{R}_0(n,m)$ that all the memories are fixed points of the network.*

This fixed point capacity of $2n$ was also demonstrated by Venkatesh and Baldi [15] in the analysis of fixed points of higher order neural networks. Recall the classical result restated in Lemma 3.4 that $2n$ is a capacity function for a *single* neuron. (The relevant attribute here is the separation of a random dichotomy of a set of points (memories) in $n$-space by a neuron.) The corollary above asserts that there is no decrease in capacity for a zero-diagonal network of $n$ neurons even though we now require $n$ dichotomies of the set of memories to be *simultaneously* separated. (As seen in the proof of the theorem, neuron $i$, for instance, is required to dichotomise the set of memories according to the set of signs $\{u_i^\alpha : \alpha \in [m]\}$.)

Theorem 4.1 hence asserts that if the fixed point requirement on the memories is relaxed and it is only required that, starting at any memory, a synchronous state transition results in a new state no more than $\epsilon n$ bits away from the memory *on average*, then the capacity increases by a constant multiplicative factor of $1/(1-2\epsilon)$. Note, however, that the rate of increase of the capacity function remains linear in $n$ and is not improved in the random error protocol.

Theorem 4.1 remains true if we are interested in *hetero-associative* maps $\mathbf{u}^\alpha \mapsto \hat{\mathbf{u}}^\alpha$ rather than the *auto-associative* maps $\mathbf{u}^\alpha \mapsto \mathbf{u}^\alpha$ that we have hitherto considered. In particular, for $\alpha = 1, \ldots, m$ let the *associated memories* $\hat{\mathbf{u}}^\alpha$ be chosen independently from $\mathbb{B}^n$ and with components drawn from a sequence of symmetric Bernoulli trials (independent of the memories $\mathbf{u}^\alpha$). The decision sets $D_i^\alpha$ are now specified in natural fashion by a sequence of Bernoulli trials with

$$D_i^\alpha = \begin{cases} \{\hat{u}_i^\alpha\} & \text{with probability } 1-2\epsilon, \\ \{-1,1\} & \text{with probability } 2\epsilon. \end{cases}$$

It is easy to see that the proof of the theorem carries over *in toto* for this hetero-associative case. A more direct proof can be crafted, however, with the observation that the independence of the components of the associated memories $\hat{\mathbf{u}}^\alpha$ yields

$$\mathbf{P}\{\mathcal{R}_\epsilon(n,m)\} = P_\epsilon(n,m)^n.$$

Lemma 3.1 now readily yields that $2n/(1-2\epsilon)$ is both a lower and an upper capacity function for $\mathcal{R}_\epsilon(n,m)$.**

## 4.2 Exhaustive Error Protocol

For $\epsilon$ close to $1/2$ the multiplicative improvement of $1/(1-2\epsilon)$ to the capacity that arises for the random error protocol can become quite large; the gains may nonetheless be perceived as unsatisfactory as there is no improvement in the *rate* of increase of capacity with $n$. The exhaustive error protocol would seem to confer even greater flexibility in the choice of errors—one is allowed in principle to examine every admissible configuration of errors before selecting the most favourable configuration—; as argued heuristically in the Introduction, this might augur well for a large improvement in capacity. We show in this section, however, that while there is a further improvement in the multiplicative constant, the capacity function for the exhaustive error protocol is still linear in $n$.

As a first step let us show that, in accordance with intuition, the exhaustive error protocol attains capacities at least as large as those of the random error protocol.

**We had presented these results without proof for the hetero-associative case in [1]. There we had assumed in addition that the memories $\mathbf{u}^\alpha$ were drawn from an absolutely continuous distribution in Euclidean $n$-space $\mathbb{R}^n$, and not from the vertices of the cube $\mathbf{B}^n$ as is more natural in the recurrent network context. The increased dependency structure in the problem makes the case of auto-association with memories drawn from $\mathbf{B}^n$ somewhat harder technically; the improvement to Füredi's lemma quoted in the text is necessary here.

**Theorem 4.3** *For any fixed error parameter $0 \le \epsilon < 1/2$, the sequence $\frac{2n}{1-2\epsilon}$ is a lower capacity function for $\mathcal{E}_\epsilon(n,m)$, the exhaustive error protocol with parameter $\epsilon$.*

PROOF: If $\epsilon = 0$ there is nothing to prove. Let us hence assume $\epsilon > 0$. Now for any choice $0 \le \epsilon' < \epsilon$, the sequence $2n/(1-2\epsilon')$ is a capacity function for the random error protocol with parameter $\epsilon'$. Invoking Lemma 3.1, within capacity the number of errors in each memory that result in the random error protocol is no more than $(\epsilon' + o(1))n$ with probability approaching one as $n \to \infty$. For any $\epsilon' < \epsilon$ the number of errors in each memory is hence less than $\epsilon n$ with probability one. Consequently, $2n/(1-2\epsilon')$ is a lower capacity function for the exhaustive error protocol with parameter $\epsilon$. As $\epsilon' < \epsilon$ can be chosen arbitrarily close to $\epsilon$, it follows from the definition of capacity that $2n/(1-2\epsilon)$ is a lower capacity for the exhaustive error protocol with parameter $\epsilon$. ∎

**Theorem 4.4** *Let $\kappa_\epsilon$ be a function of the error tolerance $\epsilon$ defined by the unique solution of*

$$H\left(\frac{1-2\epsilon}{2\kappa_\epsilon}\right) + H(\epsilon) = \log 2, \qquad 0 \le \epsilon < \frac{1}{2}, \tag{5}$$

*where $H$ is the binary entropy function. Then the sequence $\frac{2\kappa_\epsilon n}{1-2\epsilon}$ is an upper capacity function for $\mathcal{E}_\epsilon(n,m)$, the exhaustive error protocol with parameter $\epsilon$.*

REMARK: The function $\kappa_\epsilon$ is defined as we vary $\epsilon$ in the interval $0 \le \epsilon < 1/2$, and monotonically increases from a value of $+1$ at $\epsilon = 0$ to a value close to 505 as $\epsilon$ approaches 1/2. For small $\epsilon$ it remains close to $+1$, so that the capacity function for the attribute $\mathcal{E}_\epsilon(n,m)$ still behaves like $2n/(1-2\epsilon)$.

PROOF: Assume that a particular choice of weights for the zero-diagonal network of neurons results in the synchronous transitions: $\mathbf{u}^\alpha \mapsto \mathbf{v}^\alpha$, $\alpha = 1, \ldots, m$. Recall that the $i$th neuron makes a decision error on memory $\mathbf{u}^\alpha$ if $v_i^\alpha \ne u_i^\alpha$, i.e., the $i$th component of memory $\mathbf{u}^\alpha$ is not retrieved. The key observation here is the following: if each $\mathbf{v}^\alpha$ differs from the corresponding memory $\mathbf{u}^\alpha$ in no more than $\epsilon n$ components, then there exists at least one neuron which makes $\epsilon m$ or fewer decision errors. In fact, if there is no neuron which makes $\epsilon m$ or fewer decision errors, then the total number of component errors after one synchronous step summed over all the $m$ memories will exceed $\epsilon mn$ so that there has to be at least one memory $\mathbf{u}^\alpha$ which is mapped to a point $\mathbf{v}^\alpha$ which is at a Hamming distance larger than $\epsilon n$ from $\mathbf{u}^\alpha$. But this contradicts the earlier assumption about the points $\mathbf{v}^\alpha$.

Now, for any selection of values $v_i^\alpha \in \mathbb{B}$, $\alpha = 1, \ldots, m$, the probability that the $i$th neuron can realise the maps $\mathbf{u}^\alpha \mapsto v_i^\alpha$ is exactly $P(n-1, m)$. The number of ways in which $\epsilon m$ or fewer decision errors (i.e., the set $\{\alpha : v_i^\alpha \neq u_i^\alpha\}$) can occur is $\sum_{k=0}^{\epsilon m} \binom{m}{k}$ . Combining Boole's inequality with the observation above, we hence have the upper bound

$$\mathbf{P}\left\{\mathcal{E}_\epsilon(n,m)\right\} \leq n\,P(n-1,m)\sum_{k=0}^{\epsilon m}\left(\begin{array}{c} m \\ k \end{array}\right).$$

Let $\lambda > 0$ be fixed, but arbitrary, and choose $m = 2\kappa_\epsilon n(1+\lambda)/(1-2\epsilon)$, where $\kappa_\epsilon$ is as defined in (5). Two applications of Lemma 3.1 yield

$$P(n-1,m) \leq e^{-(m-1)[\log 2 - H\{(1-2\epsilon)/2\kappa_\epsilon(1+\lambda)\}]},$$

and

$$\sum_{k=0}^{\epsilon m}\left(\begin{array}{c} m \\ k \end{array}\right) \leq e^{mH(\epsilon)} \quad .$$

for large enough $n$. Hence, for each choice of $0 \leq \epsilon < 1/2$ and $\lambda > 0$, there is a choice of $\beta > 0$ such that

$$\mathbf{P}\left\{\mathcal{E}_c(n,m)\right\} \leq \beta n\,e^{-m[\log 2 - H\{(1-2\epsilon)/2\kappa_\epsilon(1+\lambda)\} - H(\epsilon)]}.$$

The binary entropy function $H(c)$ increases monotonically from a value of 0 at $c = 0$ to a value of $\log 2$ at $c = 1/2$. Hence, with $\kappa_\epsilon$ as in (5), $H\{(1-2\epsilon)/2\kappa_\epsilon(1+\lambda)\} + H(\epsilon) < \log 2$. Hence, for every choice of $\lambda > 0$ and $m = 2\kappa_\epsilon n(1+\lambda)/(1-2\epsilon)$ there is a choice of $\delta > 0$ such that $\mathbf{P}\left\{\mathcal{E}_\epsilon(n,m)\right\} < e^{-\delta m} \to 0$ as $n \to \infty$. ∎

Allowing an error tolerance of up to $\epsilon n$ bits in the recall of any memory in an associative memory application corresponds to the requirement that state transitions be confined to the ball of radius $\epsilon n$ at a memory once a transition leads within the ball. The exhaustive error protocol allows any synchronous transition from a memory that does not leave the Hamming ball of radius $\epsilon n$ at the memory. It is clear that this is a necessary condition that must be satisfied if we require confienement of trajectories within balls of radius $\epsilon n$ at the memories. Consequently, Theorem 4.4 implies the following result: *If memory components are drawn from a sequence of symmetric Bernoulli trials, then no algorithm for storing memories in a recurrent neural network can achieve a capacity which increases faster than linearly with $n$; in particular, if a linear number of errors $\epsilon n$ is permitted in the recall of any memory, then* $1010n/(1-2\epsilon)$ *is an upper capacity function for any algorithm.*

# 5 EXTENSIONS

The error protocols that we had defined in Section 2 for fully-interconnected networks can be readily extended to arbitrary network architectures. We briefly derive here certain bounds on the capacity of feedforward neural networks when there is an allowed tolerance to output error.

An *$L$-layer feedforward neural network* is comprised of $L$ ordered subcollections of neurons called *layers* with interconnections specified as follows: for $l = 2, \ldots, L$ the inputs to the $l$th layer are obtained from the outputs of the $(l-1)$th layer. The inputs to the first layer are the network inputs, and the outputs of the $L$th layer are the network outputs. Let $\mathbf{n} = (n_0\, n_1 \cdots n_{l+1})$ denote a vector of positive integers. We use the nonce terminology $\mathbf{n}$*-network* to mean an $(l+1)$-layer feedforward neural network which has $n_0 \equiv n$ inputs and whose $i$th layer contains $n_i$ neurons. For simplicity we will restrict ourselves to the case of a single output neuron, $n_{l+1} = 1$. An $\mathbf{n}$-network then realises a Boolean function of $n_0 = n$ inputs. The error protocols are readily extended to this case.

Let $\mathbf{u}^1, \ldots, \mathbf{u}^m$ be any set of points from Euclidean $n$-space. To each point $\mathbf{u}^\alpha$ we assign a desired classification $v^\alpha \in \mathbb{B}$. We assume that the set of desired classifications $\{v^1, \ldots, v^m\}$ are drawn from a sequence of symmetric Bernoulli trials.

Analogously with the fully-interconnected case, in the random error protocol we independently assign decision sets $D^\alpha$ to each classification $v^\alpha$ with

$$D^\alpha = \begin{cases} \{v^\alpha\} & \text{with probability } 1-2\epsilon, \\ \mathbb{B} & \text{with probability } 2\epsilon. \end{cases}$$

For a particular assignment of weights to the $\mathbf{n}$-network we say that the network makes $\epsilon$-reliable decisions on the set of points $\{\mathbf{u}^\alpha : \alpha \in [m]\}$ if $\mathbf{u}^\alpha \mapsto D^\alpha$ for each $\alpha \in [m]$. The attribute of interest is

EVENT $\mathcal{R}_\epsilon(\mathbf{n}, m)$: *There is a choice of weights such that the* $\mathbf{n}$*-network makes* $\epsilon$*-reliable decisions on the* $m$*-set of points* $\{\mathbf{u}^\alpha : \alpha \in [m]\}$.

A completely analogous development leads to the following attribute for the exhaustive error protocol.

EVENT $\mathcal{E}_\epsilon(\mathbf{n}, m)$: *There is a choice of weights such that the* $\mathbf{n}$*-network makes no more than* $\epsilon m$ *classification errors on the* $m$*-set of points* $\{\mathbf{u}^\alpha : \alpha \in [m]\}$.

The notion of capacity is defined as before as a threshold function of an attribute as the input dimension $n$ becomes large. (Note that we tacitly assume a family of feedforward network architectures with the number of elements in each layer $n_i$ a function of $n$.)

Now let $D(\mathbf{n}, m)$ denote the number of dichotomies of $\{\mathbf{u}^\alpha : \alpha \in [m]\}$ that can be separated by an $\mathbf{n}$-network. The following simple overestimate for $D(\mathbf{n}, m)$ is obtained by applying Lemma 3.2:

$$D(\mathbf{n}, m) \le \prod_{i=0}^{l} D(n_i, m)^{n_{i+1}}.$$

As the classifications are symmetric Bernoulli it follows that the probability $P(\mathbf{n}, m)$ that a random dichotomy can be separated by the network can be bounded by

$$P(\mathbf{n},m) = 2^{-m}D(\mathbf{n},m) \le 2^{-m}\prod_{i=0}^{l} D(n_i,m)^{n_{i+1}} = \exp\left[-m\,\log 2 + \sum_{i=0}^{l} n_{i+1}\,\log D(n_i,m)\right].$$

Using the easy bound $D(n_i,m) < m^{n_i}$ we get

$$P(\mathbf{n},m) < \exp\left[-m\log 2 + \left(\sum_{i=0}^{l} n_i n_{i+1}\right)\log m\right].$$

Define the function $C_1(\mathbf{n})$ by

$$C_1(\mathbf{n})\,\log 2 = \left(\sum_{i=0}^{l} n_i n_{i+1}\right)\log C_1(\mathbf{n}).$$

Note that

$$C_1(\mathbf{n}) = \frac{1}{\log 2}\left(\sum_{i=0}^{l} n_i n_{i+1}\right)\log\left(\sum_{i=0}^{l} n_i n_{i+1}\right)\left[1 + O\left(\frac{\log\log\sum_{i=0}^{l} n_i n_{i+1}}{\log\sum_{i=0}^{l} n_i n_{i+1}}\right)\right] \qquad (n\to\infty).$$

It is clear that for any $\lambda > 0$, if $m \ge (1+\lambda)C_1(\mathbf{n})$ then $P(\mathbf{n},m)\to 0$ as $n\to\infty$. As before, we have

$$\mathbf{P}\{\mathcal{R}_\epsilon(\mathbf{n},m)\} = \sum_{k=0}^{m} b(k;m,1-2\epsilon)\,P(\mathbf{n},k).$$

The same argument in the proof of Theorem 4.3 continues to work, so that we have proved the following

**Theorem 5.1** *The sequence* $\frac{C_1(\mathbf{n})}{1-2\epsilon}$ *is an upper capacity function for the attribute* $\mathcal{R}_\epsilon(\mathbf{n},m)$.

For the exhaustive error protocol, we have likewise

$$\mathbf{P}\{\mathcal{E}_\epsilon(\mathbf{n},m)\} \le P(\mathbf{n},m)\sum_{k=0}^{\epsilon m}\binom{m}{k} < \exp\left[-m\log 2 + \left(\sum_{i=0}^{l} n_i n_{i+1}\right)\log m + m\,H(\epsilon)\right].$$

For a small enough choice of error parameter $\epsilon$ let $C_2(\epsilon;\mathbf{n})$ satisfy

$$C_2(\epsilon;\mathbf{n})[\log 2 - H(\epsilon)] = \left(\sum_{i=0}^{l} n_i n_{i+1}\right)\log C_2(\epsilon;\mathbf{n}).$$

Again we have

$$C_2(\epsilon;\mathbf{n}) = \frac{1}{\log 2 - H(\epsilon)}\left(\sum_{i=0}^{l} n_i n_{i+1}\right)\log\left(\sum_{i=0}^{l} n_i n_{i+1}\right)\left[1 + O\left(\frac{\log\log\sum_{i=0}^{l} n_i n_{i+1}}{\log\sum_{i=0}^{l} n_i n_{i+1}}\right)\right] \qquad (n\to\infty).$$

For any fixed $\lambda > 0$ if $m \ge (1+\lambda)C_2(\epsilon;\mathbf{n})$ then $\mathbf{P}\{\mathcal{E}_\epsilon(\mathbf{n},m)\}\to 0$ as $n\to\infty$. Hence we have the following

**Theorem 5.2** *The sequence* $C_2(\epsilon;\mathbf{n})$ *is an upper capacity function for the attribute* $\mathcal{E}_\epsilon(\mathbf{n},m)$.

The results can be sharpened, for instance, by using the tighter bound $D(n_i,m) < 2m^{n_i-1}/(n_i-1)!$ which is valid if $m > 3n_i$ and $n_i > 3$ instead of the simpler estimate $D(n_i,m) < m^{n_i}$ used here. Unfortunately, good lower bounds are currently unavailable except for the case of one and two layer networks (cf. Venkatesh and Psaltis [6]; Baum [16]).

# 6 CONCLUSIONS

The main contributions of this paper are the development of formal protocols for error tolerance, and the explicit computation of the gains that accrue for neural network capacity under these protocols when errors are permitted in the output. The principal result here is that error tolerance in a network situation results in gains in the multiplicative constants for the capacity, but leaves the rate of growth of capacity as input dimensionality increases unchanged. In particular, if a tolerance of $\epsilon \in [0, 1/2)$ is specified for a fully-interconnected network, then under the random error protocol there is a gain in capacity by a multiplicative factor of $1/(1-2\epsilon)$ over the error free case, while for the exhaustive error protocol the gain is no more than a multiplicative factor of $505/(1-2\epsilon)$. Similar gains accrue for feedforward network configurations. The absence of more startling gains in capacity can be traced to the exponential decay of the relevant probabilities. These protocols are readily applicable in other computational paradigms. Following the analysis here, in a general computational scenario we would expect error tolerance to buy us order of magnitude improvements in computational capability only if the relevant probabilities decay sufficiently slowly.

## References

[1] S. S. Venkatesh, "Epsilon capacity of neural networks," in *Neural Networks for Computing*, (ed. J. Denker). New York: AIP, 1986.

[2] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 461–482, 1987.

[3] J. Komlós and R. Paturi, "Convergence results in an associative memory model," *Neural Networks*, vol. 1, no. 3, pp. 239-250, 1988.

[4] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational properties," *Proc. Nat'l Acad. Sci.*, vol. 79, pp. 2554–2558, 1982.

[5] C. M. Newman, "Memory capacity in neural network models: rigorous lower bounds," *Neural Networks*, vol. 1, no. 3, pp. 223-238, 1988.

[6] S. S. Venkatesh and D. Psaltis, "On reliable computation with formal neurons," *IEEE Trans. Pattern Analysis and Machine Intelligence*, August 1991, to appear.

[7] S. S. Venkatesh, "Computation and learning in the context of neural network capacity," in *Neural Networks for Perception*, (ed. H. Wechsler). New York: Academic Press, 1991.

[8] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Ann. Math. Stat.*, vol. 23, pp. 493–507, 1952.

[9] I. F. Blake and H. Darabian, "Approximations for the probability in the tails of the binomial distribution," *IEEE Trans. Inform. Theory*, vol. IT–33, pp. 426–428, 1987.

[10] L. Schläfli, *Gesammelte Mathematische Abhandlungen I.* Basel, Switzerland: Verlag Birkhäuser, pp. 209–212, 1950.

[11] J. G. Wendel, "A problem in geometric probability," *Math. Scand.*, vol. 11, pp. 109–111, 1962.

[12] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Trans. Elec. Comp.*, vol. EC–14, pp. 326–334, 1965.

[13] Z. Füredi, "Random polytopes in the *d*-dimensional cube," *Discrete Comput. Geom.*, vol. 1, pp. 315–319, 1986.

[14] J. Kahn, J. Komlós, and E. Szemerédi, "On the determinant of random ±1 matrices," preprint.

[15] S. S. Venkatesh and P. Baldi, "Programmed interactions in higher order neural networks: maximal capacity," *Jnl. Compl.*, in press.

[16] E. B. Baum, "On the capabilities of multi-layer perceptrons," *Jnl. Compl.*, vol. 4, pp. 193–215, 1988.

## Figure Captions

**Fig. 1** A schematic demonstrating error correction (attraction) and error tolerance for a memory. Points in the Hamming ball of radius $\rho n$ at the memory lie on trajectories which ultimately are confined in the (smaller) Hamming ball of radius $\epsilon n$ at the memory.

**Fig. 2** A schematic showing a set of "admissible" transitions starting from a memory. Each such transition from a memory must result in a new vertex of the cube $\{-1,1\}^n$ which differs from the memory in no more than $\epsilon n$ components.

Fig. 1

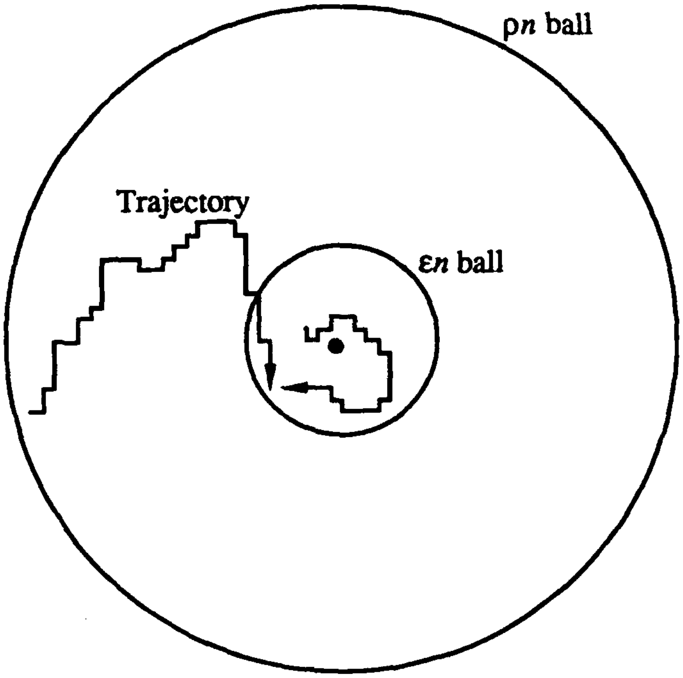
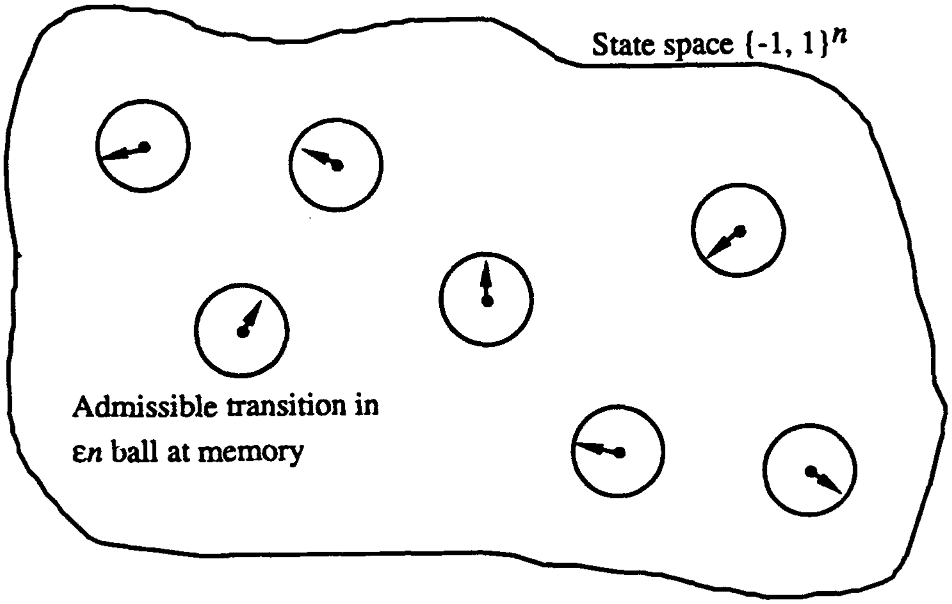


Fig 2


State space $\{-1, 1\}^n$
Admissible transition in
$\varepsilon n$ ball at memory

Advances in Neural Information
Processing Systems 3
(D.S. Touretzky & R. Lippman,eds.)
Morg‑‑n Kaufmann, San Mateo, CA
1991

# Asymptotic slowing down of the nearest-neighbor classifier

**Robert R. Snapp**
CS/EE Department
University of Vermont
Burlington, VT 05405

**Demetri Psaltis**
Electrical Engineering
Caltech 116–81
Pasadena, CA 91125

**Santosh S. Venkatesh**
Electrical Engineering
University of Pennsylvania
Philadelphia, PA 19104

## Abstract

If patterns are drawn from an $n$-dimensional feature space according to a probability distribution that obeys a weak smoothness criterion, we show that the probability that a random input pattern is misclassified by a nearest-neighbor classifier using $M$ random reference patterns asymptotically satisfies

$$P_M(\text{error}) \sim P_\infty(\text{error}) + \frac{a}{M^{2/n}},$$

for sufficiently large values of $M$. Here, $P_\infty(\text{error})$ denotes the probability of error in the infinite sample limit, and is at most twice the error of a Bayes classifier. Although the value of the coefficient $a$ depends upon the underlying probability distributions, the exponent of $M$ is largely distribution free. We thus obtain a concise relation between a classifier's ability to generalize from a finite reference sample and the dimensionality of the feature space, as well as an analytic validation of Bellman's well known "curse of dimensionality."

## 1 INTRODUCTION

One of the primary tasks assigned to neural networks is pattern classification. Common applications include recognition problems dealing with speech, handwritten characters, DNA sequences, military targets, and (in this conference) sexual identity. Two fundamental concepts associated with pattern classification are *generalization* (how well does a classifier respond to input data it has never encountered before?) and *scalability* (how are a classifier's processing and training requirements affected by increasing the number of features that describe the input patterns?).

Despite recent progress, our present understanding of these concepts in the context of neural networks is obstructed by complexities in the functional form of the network and in the classification problems themselves.

In this correspondence we will present analytic results on these issues for the nearest-neighbor classifier. Noted for its algorithmic simplicity and nearly optimal performance in the infinite sample limit, this pattern classifier plays a central role in the field of pattern recognition. Furthermore, because it uses proximity in feature space as a measure of class similarity, its performance on a given classification problem should yield qualitative cues to the performance of a neural network. Indeed, a nearest-neighbor classifier can be readily implemented as a "winner-take-all" neural network.

## 2 THE TASK OF PATTERN CLASSIFICATION

We begin with a formulation of the two-class problem (Duda and Hart, 1973):

> Let the labels $\omega_1$ and $\omega_2$ denote two states of nature, or pattern classes. A pattern belonging to one of these two classes is selected, and a vector of $n$ features, $\mathbf{x}$, that describe the selected pattern is presented to a pattern classifier. The classifier then attempts to guess the selected pattern's class by assigning $\mathbf{x}$ to either $\omega_1$ or $\omega_2$.

As an example, the two class labels might represent the states *benign* and *malignant* as they pertain to the diagnosis of cancer tumors; the feature vector could then be a 1024 x 1024 pixel, real-valued representation of an electron-microscope image. A pattern classifier can thus be viewed as a mapping from an $n$-dimensional feature space to the discrete set $\{\omega_1, \omega_2\}$, and can be specified by demarcating the regions in the $n$-dimensional feature space that correspond to $\omega_1$ and $\omega_2$. We define the decision region $\mathcal{R}_1$ as the set of feature vectors that the pattern classifier assigns to $\omega_1$, with an analogous definition for $\mathcal{R}_2$. A useful figure of merit is the probability that the feature vector of a randomly selected pattern is assigned to the correct class.

### 2.1 THE BAYES CLASSIFIER

If sufficient information is available, it is possible to construct an optimal pattern classifier. Let $P(\omega_1)$ and $P(\omega_2)$ denote the ***prior probabilities*** of the two states of nature. (For our cancer diagnosis problem, the prior probabilities can be estimated by the relative frequency of each type of tumor in a large statistical sample.) Further, let $p(\mathbf{x} \mid \omega_1)$ and $p(\mathbf{x} \mid \omega_2)$ denote the ***class-conditional probability densities*** of the feature vector for the two class problem. The total probability density is now defined by $p(\mathbf{x}) = p(\mathbf{x} \mid \omega_1)P(\omega_1) + p(\mathbf{x} \mid \omega_2)P(\omega_2)$, and gives the unconditional distribution of the feature vector. Where $p(\mathbf{x}) \neq 0$ we can now use Bayes' rule to compute the ***posterior probabilities***:

$$P(\omega_1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_1)P(\omega_1)}{p(\mathbf{x})} \qquad \text{and} \qquad P(\omega_2 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_2)P(\omega_2)}{p(\mathbf{x})}.$$

*The Bayes classifier* assigns an unclassified feature vector $\mathbf{x}$ to the class label having the greatest posterior probability. (If the posterior probabilities happen to be equal, then the class assignment is arbitrary.) With $\mathcal{R}_1$ and $\mathcal{R}_2$ denoting the two decision regions induced by this strategy, the probability of error of the Bayes classifier, $P_B$, is just the probability that $\mathbf{x}$ is drawn from class $\omega_1$ but lies in the Bayes decision region $\mathcal{R}_2$, or conversely, that $\mathbf{x}$ is drawn from class $\omega_2$ but lies in the Bayes decision region $\mathcal{R}_1$:

$$P_B = \int_{\mathcal{R}_2} P(\omega_1 \mid \mathbf{x})\, p(\mathbf{x})\, d^n x + \int_{\mathcal{R}_1} P(\omega_2 \mid \mathbf{x})\, p(\mathbf{x})\, d^n x.$$

The reader may verify that the Bayes classifier minimizes the probability of error.

Unfortunately, it is usually impossible to obtain expressions for the class-conditional densities and prior probabilities in practice. Typically, the available information resides in a set of correctly labeled patterns, which we collectively term a *training* or *reference sample*. Over the last few decades, numerous pattern classification strategies have been developed that attempt to learn the structure of a classification problem from a finite training sample. (The backpropagation algorithm is a recent example.) The underlying hope is that the classifier's performance can be made acceptable with a sufficiently large reference sample. In order to understand how large a sample may be needed, we turn to what is perhaps the simplest learning algorithm of this class.

## 3 THE NEAREST-NEIGHBOR CLASSIFIER

Let $\mathcal{X}_M = \{(\mathbf{x}^{(1)}, \theta^{(1)}), (\mathbf{x}^{(2)}, \theta^{(2)}), \ldots, (\mathbf{x}^{(M)}, \theta^{(M)})\}$ denote a finite reference sample of $M$ feature vectors, $\mathbf{x}^{(i)} \in \mathbf{R}^n$, with corresponding known class assignments, $\theta^{(i)} \in \{\omega_1, \omega_2\}$. The *nearest-neighbor rule* assigns each feature vector $\mathbf{x}$ to class $\omega_1$ or $\omega_2$ as a function of the reference $M$-sample as follows:

- *Identify* $(\mathbf{x}', \theta') \in \mathcal{X}_M$ such that $||\mathbf{x} - \mathbf{x}'|| \leq ||\mathbf{x} - \mathbf{x}^{(i)}||$ for $i$ ranging from 1 through $M$;
- *Assign* $\mathbf{x}$ to class $\theta'$.

Here, $||\mathbf{x} - \mathbf{y}|| = \sqrt{\sum_{j=1}^{n} (x_j - y_j)^2}$ denotes the Euclidean metric in $\mathbf{R}^n$.[1]The nearest-neighbor rule hence classifies each feature vector $\mathbf{x}$ according to the label, $\theta'$, of the closest point, $\mathbf{x}'$, in the reference sample. As an example, we sketch the nearest-neighbor decision regions for a two-dimensional classification problem in Fig. 1.

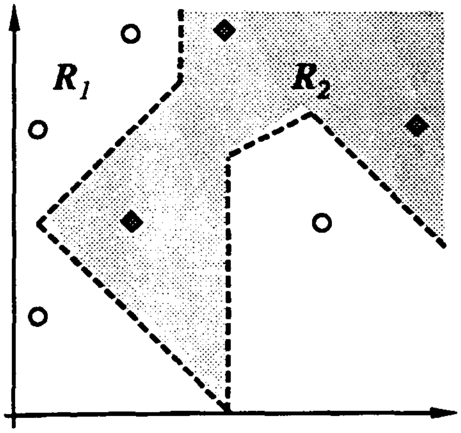


Figure 1: The decision regions induced by a nearest-neighbor classifier with a seven-element reference set in the plane.

[1] Other metrics, such as the more general Minkowski-$r$ metric, are also possible.

It is interesting to consider how the performance of this classifier compares with that of a Bayes classifier. To facilitate this analysis, we assume that the reference patterns are selected from the total probability density $p(\mathbf{x})$ in a statistically independent manner (i.e., the choice of $\mathcal{X}_j$ does not in any way bias the selection of $\mathbf{x}^{(j+1)}$ and $\theta^{(j+1)}$). Furthermore, let $P_M(\text{error})$ denote the probability of error of a nearest-neighbor classifier working with the reference sample $\mathcal{X}_M$, and let $P_\infty(\text{error})$ denote this probability in the infinite sample limit ($M \rightarrow \infty$). We will also let $\mathcal{S}$ denote the volume in feature space over which $p(\mathbf{x})$ is nonzero. The following well known theorem shows that the nearest-neighbor classifier, with an infinite reference sample, is nearly optimal (Cover and Hart, 1967).[2]

**Theorem 1** *For the two-class problem in the infinite sample limit, the probability of error of a nearest-neighbor classifier tends toward the value,*

$$P_\infty(\text{error}) = 2\int_{\mathcal{S}} P(\omega_1 \mid \mathbf{x})P(\omega_2 \mid \mathbf{x})p(\mathbf{x})\, d^n x,$$

*which is furthermore bounded by the two inequalities,*

$$P_B \leq P_\infty(\text{error}) \leq 2P_B(1 - P_B),$$

*where $P_B$ is the probability of error of a Bayes classifier.*

This encouraging result is not so surprising if one considers that, with probability one, about every feature vector $\mathbf{x}$ is centered a ball of radius $\epsilon$ that contains an *infinite* number of reference feature vectors for every $\epsilon > 0$. The annoying factor of two accounts for the event that the nearest neighbor to $\mathbf{x}$ belongs to the class with smaller posterior probability.

## 3.1 THE ASYMPTOTIC CONVERGENCE RATE

In order to satisfactorily address the issues of generalization and scalability for the nearest-neighbor classifier, we need to consider the rate at which the performance of the classifier approaches its infinite sample limit. The following theorem applicable to nearest-neighbor classification in *one-dimensional* feature spaces was shown by Cover (1968).

**Theorem 2** *Let $p(x \mid \omega_1)$ and $p(x \mid \omega_2)$ have uniformly bounded third derivatives and let $p(\mathbf{x})$ be bounded away from zero on $\mathcal{S}$. Then for sufficiently large $M$,*

$$P_M(\text{error}) = P_\infty(\text{error}) + O\left(\frac{1}{M^2}\right).$$

Note that this result is also very encouraging in that an order of magnitude increase in the sample size, decreases the error rate by *two* orders of magnitude.

The following theorem is our main result which extends Cover's theorem to $n$-dimensional feature spaces:

[2] Originally, this theorem was stated for multiclass decision problems; it is here presented for the two class problem only for simplicity.

**Theorem 3** *Let $p(\mathbf{x} \mid \omega_1)$, $p(\mathbf{x} \mid \omega_2)$, and $p(\mathbf{x})$ satisfy the same conditions as in Theorem 2. Then, there exists a scalar $a$ (depending on $n$) such that*

$$P_M(\text{error}) \sim P_\infty(\text{error}) + \frac{a}{M^{2/n}},$$

*where the right-hand side describes the first two terms of an asymptotic expansion in reciprocal powers of $M^{2/n}$. Explicitly,*

$$a = \frac{\Gamma\left(1+\frac{2}{n}\right)\left(\Gamma\left(\frac{n}{2}+1\right)\right)^{2/n}}{n\pi} \sum_{i=1}^{n} \int_S \left( \frac{\beta_i(\mathbf{x}) p_i(\mathbf{x})}{p(\mathbf{x})} + \frac{1}{2}\gamma_{ii}(\mathbf{x}) \right) (p(\mathbf{x}))^{1-2/n} \, d^n x.$$

*where,*

$$\begin{aligned} p_i(\mathbf{x}) &= \frac{\partial p(\mathbf{x})}{\partial x_i} \\ \beta_i(\mathbf{x}) &= P(\omega_1 \mid \mathbf{x}) \frac{\partial P(\omega_2 \mid \mathbf{x})}{\partial x_i} + \frac{\partial P(\omega_1 \mid \mathbf{x})}{\partial x_i} P(\omega_2 \mid \mathbf{x}) \\ \gamma_{ii}(\mathbf{x}) &= P(\omega_1 \mid \mathbf{x}) \frac{\partial^2 P(\omega_2 \mid \mathbf{x})}{\partial x_i^2} + \frac{\partial^2 P(\omega_1 \mid \mathbf{x})}{\partial x_i^2} P(\omega_2 \mid \mathbf{x}). \end{aligned}$$

For $n = 1$ this result agrees with Cover's theorem. With increasing $n$, however, the convergence rate significantly slows down. Note that the constant $a$ depends on the way in which the class-conditional densities overlap. If $a$ is bounded away from zero, then for sufficiently small $\delta > 0$, $P_M(\text{error}) - P_\infty(\text{error}) < \delta$ is satisfied only if $M > (a/\delta)^{n/2}$ so that the sample size required to achieve a given performance criterion is exponential in the dimensionality of the feature space. The above provides a sufficient condition for Bellman's well known "curse of dimensionality" in this context.

It is also interesting to note that one can easily construct classification problems for which $a$ vanishes. (Consider, for example, $p(\mathbf{x} \mid \omega_1) = p(\mathbf{x} \mid \omega_2)$ for all $\mathbf{x}$.) In these cases the higher-order terms in the asymptotic expansion are important.

## 4 A NUMERICAL EXPERIMENT

A conspicuous weakness in the above theorem is the requirement that $p(\mathbf{x})$ be bounded away from zero over $S$. In exchange for a uniformly convergent asymptotic expansion, we have omitted many important probability distributions, including normal distributions. Therefore we numerically estimate the asymptotic behavior of $P_M(\text{error})$ for a problem consisting of two normally distributed classes in $\mathbf{R}^n$:

$$\begin{aligned} p(\mathbf{x} \mid \omega_1) &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left[-\tfrac{1}{2\sigma^2}\left((x_1-\mu)^2 + \textstyle\sum_{j=2}^{n} x_j^2\right)\right], \\ p(\mathbf{x} \mid \omega_2) &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left[-\tfrac{1}{2\sigma^2}\left((x_1+\mu)^2 + \textstyle\sum_{j=2}^{n} x_j^2\right)\right]. \end{aligned}$$

Assuming that $P(\omega_1) = P(\omega_2) = 1/2$, we find

$$P_\infty(\text{error}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\mu^2/2\sigma^2} \int_0^\infty e^{-x^2/2\sigma^2} \operatorname{sech}\left(\frac{\mu x}{\sigma^2}\right) dx.$$

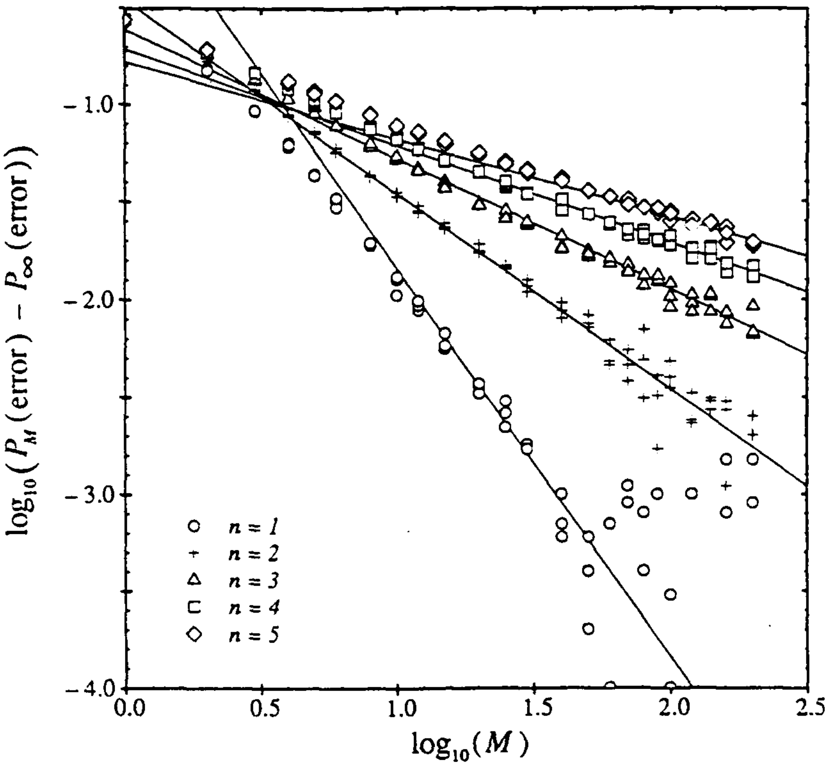


Figure 2: Numerical validation of the nearest-nˈghbor scaling hypothesis for two normally distributed classes in $\mathbf{R}^n$.

For $\mu = \sigma = 1$, $P_\infty$(error) is numerically found to be 0.22480, which is consistent with the Bayes probability of error, $P_B = (1/2)\text{erfc}(1/\sqrt{2}) = 0.15865$. (Note that the expression for $a$ given in Theorem 3 is undefined for these distributions.) For $n$ ranging from 1 to 5, and $M$ ranging from 1 to 200, three estimates of $P_M$(error) were obtained, each as the fraction of "failures" in 160,000 or more Bernoulli trials. Each trial consists of constructing a pseudo-random sample of $M$ reference patterns, followed by a single attempt to correctly classify a random input pattern. These estimates of $P_M$ are represented in Figure 2 by circular markers for $n = 1$, crosses for $n = 2$, etc. The lines in Figure 2 depict the power law

$$P_M(\text{error}) = P_\infty(\text{error}) + bM^{-2/n},$$

where, for each $n$, $b$ is chosen to obtain an appealing fit. The agreement between these lines and data points suggests that the asymptotic scaling hypothesis of Theorem 3 can be extended to a wider class of distributions.

## 5 DISCUSSION

The preceding analysis indicates that the convergence rate of the nearest-neighbor classifier slows down dramatically as the dimensionality of the feature space increases. This rate reduction suggests that proximity in feature space is a less effective measure of class identity in higher dimensional feature spaces. It is also clear that some degree of smoothness in the class-conditional densities is necessary, as well as sufficient, for the asymptotic behavior described by our analysis to occur: in the absence of smoothness conditions, one can construct classification problems for which the nearest-neighbor convergence rate is arbitrarily slow, even in one dimension (Cover, 1968). Fortunately, the most pressing classification problems are typically smooth in that they are constrained by regularities implicit in the laws of nature (Marr, 1982). With additional prior information, the convergence rate may be enhanced by selecting a fewer number of descriptive features.

Because of their smooth input-output response, neural networks appear to use proximity in feature space as a basis for classification. One might, therefore, expect the required sample size to scale exponentially with the dimensionality of the feature space. Recent results from computational learning theory, however, imply that with a sample size proportional to the *capacity*—a combinatorial quantity which is characteristic of the network architecture and which typically grows polynomially in the dimensionality of the feature space—one can in principle identify network parameters (weights) which give (close to) the smallest classification error for the given architecture (Baum and Haussler, 1989). There are two caveats, however. First, the information-theoretic sample complexities predicted by learning theory give no clue as to whether, given a sample of the requisite size, there exist any *algorithms* that can specify the appropriate parameters in a reasonable time frame. Second, and more fundamental, one cannot in general determine whether a particular architecture is intrinsically well suited to a given classification problem. The best performance achievable may be substantially poorer than that of a Bayes classifier. Thus, without sufficient prior information, one must search through the space of all possible network architectures for one that does fit the problem well. This situation now effectively resembles a non-parametric classifier and the analytic results for the sample complexities of the nearest-neighbor classifier should provide at least qualitative indications of the corresponding case for neural networks.

### References

Baum, E. B. and Haussler, D. (1989), "What size net gives valid generalization," *Neural Computation*, **1**, pp. 151–160.

Cover, T. M. (1968), "Rates of convergence of nearest neighbor decision procedures," *Proc. First Annual Hawaii Conference on Systems Theory*, pp. 413–415.

Cover, T. M. and P. E. Hart (1967), "Nearest neighbor pattern classification," *IEEE Trans. Info. Theory*, vol. IT–13, pp. 21–27.

Duda, R. O. and P. E. Hart (1973), *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons.

Marr, D. (1982), *Vision*, San Francisco: W. H. Freeman.

IEEE Transactions on
Information Theory
March, 1993, in press

# ON THE MINIMUM EXPECTED DURATION OF A COIN TOSSING GAME

Inchi Hu[*] Santosh S. Venkatesh[†]

Submitted 10 October 1991
Revised 26 March 1992[‡]

**Abstract**

The following coin tossing game is analysed: A store of $N$ fair coins is given and it is desired to achieve $M$ heads in a round of tosses of the coins. To allow for unfavourable sequences of tails, restarts are permitted at any epoch in the game where, in any restart, all coins are returned to store and tosses are begun anew from tabula rasa. A restart strategy is a prescription which specifies when a restart should be made. It is desired to estimate the minimum expected duration of the game over all restart strategies, and to find an optimal strategy which minimises the expected duration of the game. This simple coin tossing game, proposed by R. L. Rivest, has cryptographic roots and is linked to issues in the factoring of integers.

INDEX TERMS: *Backward Induction, Coin Tossing Game, Cryptography, Integer Factoring, Markov Decision Problem, Optimal Stopping*

## 1 TWENTY QUESTIONS

R. L. Rivest proposed the following problem. An individual has 20 fair coins in his pocket. He takes coins out of his pocket one at a time and tosses them, his objective being to obtain 15 heads. If fewer than 15 heads transpire in any round of 20 tosses, he must return all 20 coins to his pocket and restart the game. He also has the option of restarting the game at any point by ending a round of tosses and returning all 20 coins to his pocket before starting anew. The problem facing our protagonist is to choose an optimal restart strategy which would minimise the expected number of tosses he has to make before achieving his goal of 15 heads.

The general problem where $M$ heads are desired in tosses out of a store of $N$ fair coins, with restarts allowed, has cryptographic roots and, in particular, is related to the problem of choosing an optimal early abort strategy in randomised algorithms for factoring integers.

Consider for instance the problem of factoring an integer $n$. A basic approach to finding a factor of $n$ is to first find integers $k$ and $l$ such that

$$k^2 \equiv l^2 \pmod{n}, \qquad 0 < k, l < n, \qquad k \neq l, \qquad k + l \neq n. \tag{1}$$

In fact, this congruence implies that $n$ is a divisor of $(k^2 - l^2)$ yet $n$ divides neither $(k - l)$ nor $(k + l)$. It follows that $\gcd(n, k - l)$ and $\gcd(n, k + l)$ are proper factors of $n$, and these can be found efficiently, for instance, by Euclid's algorithm (Euclid [1, Book VII, Propositions 1, 2]!).

[‡] Submitted for publication to the *IEEE Transactions on Information Theory*.

[*] Department of Statistics, University of Pennsylvania, Philadelphia, PA 19104

[†] Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104; *electronic mail:* venkatesh@ee.upenn.edu

Now let a number $m < n$ be fixed and let $\pi(m) \equiv N$ denote the number of primes less than $m$. Denote these primes by $p_1, \ldots, p_N$. Now note that it suffices to derive $N+1$ distinct solutions $(k_i, h_{i1}, \ldots, h_{iN})$, $1 \le i \le N+1$ to the congruence

$$k^2 \equiv p_1^{h_1} p_2^{h_2} \cdots p_N^{h_N} \pmod{n}, \tag{2}$$

for then the vectors $(h_{i1}, \ldots, h_{iN})$, $1 \le i \le N+1$ are linearly dependent modulo 2, i.e., there exists $(h'_1, \ldots, h'_N)$ such that

$$(h_{11}, \ldots, h_{1N}) + \cdots + (h_{N+1,1}, \ldots, h_{N+1,N}) = (2h'_1, \ldots, 2h'_N).$$

The integers

$$k = (k_1 \cdots k_{N+1}) \bmod n, \qquad l = (p_1^{h'_1} \cdots p_N^{h'_N}) \bmod n$$

would hence be a solution to the congruence (1).[1]

Dixon's algorithm is a randomised approach to finding solutions to the congruence (2). The algorithm proceeds in a series of rounds prior to each of which a random integer $k$ is generated as a putative solution to (2). During a round each of the $N$ primes $p_i$ is successively tested to see whether ($k^2$ mod $n$) has all its prime factors less than $m$. If all the $N$ primes have been tested and a solution to (2) has not been obtained then a new round is initiated with the generation of a new random integer $k$. Provision is also made for an early abort strategy where a round is ended and a new random integer $k$ generated before all the $N$ primes have been tested. (See Pomerance [2] for more details.)

Checking to see whether a given prime $p$ less than $m$ divides the random integer $k$ during a round corresponds to a coin toss in our problem, with the result a "head" if $p$ divides ($k^2$ mod $n$); the total number of available primes $N$ determines the number of coins available. Success in the coin tossing game corresponds to obtaining a solution to the congruence (2); the number of "heads" needed in a round of "tosses" is the number $M(k) \le N$ of primes $p_i$ for which $h_i \ne 0$ in (2) if in fact the congruence can be satisfied for the given value of $k$. Points of departure from the coin tossing problem are that the probability that any given prime $p$ in our store divides ($k^2$ mod $n$) is $1/p$ (neglecting boundary effects), and this varies from prime to prime. Furthermore, the number $M(k)$ of "heads" needed in a round varies from round to round as the values of $k$ are randomly generated. Roughly speaking, however, most of the primes $p$ are of the order of $m$ and most values of ($k^2$ mod $n$) are of the order of $n$ so that an approximation to the problem in terms of the coin tossing game is to consider a store of $N = \pi(m)$ unfair coins with identical probabilities $1/m$ of a toss resulting in a head and require to find an optimal strategy which minimises the total number of tosses before achieving $M = \log_m n$ heads (see Section 6).

In recent unpublished work, G. F. Bachelis and F. J. Massey [3] have attempted to characterise optimal strategies for the coin tossing problem using elegant techniques from Markov decision theory. Formulating the game as a Markov decision problem, they are led to a consideration of a related optimal stopping problem for a random walk to obtain some general properties of an optimal strategy. They also link the coin tossing problem to a continuous analogue involving an optimal stopping problem for a particle moving under Brownian motion. While explicit closed form solutions for an optimal strategy and the expected minimum duration of the game remain elusive, they obtain asymptotic results on the expected duration of the game for a choice $M = N/2 + O(\sqrt{N})$ ($N \to \infty$) for two suboptimal strategies: restarting only when the number of tails in a round reaches $N - M + 1$, and restarting when the number of tails in a round exceeds the number of heads by a fixed amount.

Our approach to the problem here, in sharp contrast, uses purely elementary techniques. Our main results, contained largely in the next three sections, involve sharp estimates of the minimum expected duration of the game, and a specification of an efficient procedure for finding optimal strategies.

[1] Caveat: unless $k \equiv \pm l$.

In brief, Section 2 is devoted to a formalisation of the problem, and a characterisation of some principal features of an optimal restart strategy. The main result shown here is that the search for an optimal strategy can be confined to a relatively small family of restart strategies that we call parsimonious.

Section 3 contains explicit estimates of the expected duration of the game. In particular, using elementary arguments we obtain an explicit general form for the expected duration of the game under any consistent strategy. We also illustrate the utility of the general solution with calculations for divers strategies. With direct estimates of the minimum expected duration of the game we then show that when $M < N/2$ an optimal strategy (and essentially any sensible strategy) has expected duration $2M[1 + o(1)]$, when $M = N/2$ an optimal strategy has expected duration between $2M$ and $4M$, while for $M > N/2$ an optimal strategy has expected duration increasing exponentially in $M$.

In Section 4 we explicitly prescribe a backward induction algorithm which efficiently generates optimal strategies for the coin tossing problem, and provide a proof of its convergence to an optimal solution. The algorithm exploits the equivalence between the coin tossing game and a related optimal stopping problem using the prior characterisation of the features of an optimal strategy (derived in Section 2) and the general solutions for the expected duration of the game (obtained in Section 3).

In Section 5 we include some numerical computations comparing the expected duration of the game under various strategies with the expected duration of the game under an optimal strategy obtained using the backward induction algorithm. The simulations indicate that a restart strategy introduced here—the *Ballot Strategy*—has a near-optimal character.

We conclude in Section 6 with extensions of our results to coin tossing games where the coin is unfair.

# 2 THE COIN TOSSING GAME

## 2.1 Restart Strategies

Let us begin by formalising the setup of the game. The following three items constitute the game's critical parameters:

- A sequence of symmetric Bernoulli trials $\{X_i\}$ taking values in $\{0, 1\}$ and denoting the results of a sequence of fair coin tosses. A coin toss resulting in a "head" corresponds to $X_i = 1$ and is called a "success".

- A running total of successes $S$ initialised with $S \leftarrow 0$, and the number $n$ of coin tosses in the current round also initialised with the assignment $n \leftarrow 0$.

- A sequence $(f_j, j \geq 1) = (f_1, f_2, \ldots)$ called the ***restart strategy*** where each $f_j$—a ***restart function***—is a randomised Boolean function from $\{0, 1, \ldots, N\} \times \{0, 1, \ldots, M - 1\}$ into $\{0, 1\}$ with
$$f_j(n, m) = \begin{cases} 1 & \text{with probability } \phi_j(n, m), \\ 0 & \text{with probability } 1 - \phi_j(n, m). \end{cases}$$
Here $\phi_j(n, m)$ denotes the probability of deciding to restart (in round $j$) when $m$ successes have been obtained in $n \leq N$ tosses; we also impose the constraint $\phi_j(N, m) \equiv 1$ for $m = 0, \ldots, M - 1$, i.e., a restart is mandated if all $N$ coins are tossed and fewer than $M$ successes obtained.[2]

The game proceeds iteratively as follows, starting at epoch $T = 1$ and initialised with the number of successes in a round $S = 0$, the number of coin tosses in a round $n = 0$, and the number of restarts $K = 0$:

[2] An intuitive approach would be to just consider *stationary, nonrandom* restart strategies specified by $f_j \equiv f$, $j \geq 1$, where $f$ denotes any fixed, deterministic function. It is nontrivial to determine whether or not the increased generality espoused in the setup here buys improvements. See Lemma 2.1 for a resolution.

1. [Accumulation.] Update the running total of successes and the number of coin tosses in the current round of tosses: $S \leftarrow S + X_T$, $n \leftarrow n+1$.

2. [Stop?] Check to see if the number of successes is equal to the desired value: If $S = M$ output the duration of the game $T$ and stop.

3. [Restart?] Check the restart strategy to see if the game should be restarted: If $f_{K+1}(n,S) = 1$ then reset $S \leftarrow 0$ and $n \leftarrow 0$, and increment $K \leftarrow K+1$.

4. [Next epoch.] Increment the epoch by one: $T \leftarrow T+1$. Go back to step 1.

Let $\mathcal{S}$ denote the family of all (random) restart strategies. For a given strategy $(f_j, j \geq 1)$, let $T_{(f_j, j\geq 1)}$ denote the duration of the game. We say that a strategy $(f_j^*, j \geq 1) \in \mathcal{S}$ is *optimal* if $\mathbf{E}\, T_{(f_j^*, j\geq 1)} = \inf_{\mathcal{S}} \mathbf{E}\, T_{(f_j, j\geq 1)}$. Our goal is to find an optimal strategy and to estimate the minimum expected duration of the game

$$D(N,M) = \inf_{\mathcal{S}} \mathbf{E}\, T_{(f_j, j\geq 1)}. \tag{3}$$

The model described here does not include a cost for restarting the game. Restarting costs can, however, be incorporated very simply; see the remark following the proof of Theorem 3.3.

## 2.2 Optimal Strategies

Consider a single round of coin tosses governed by a restart function $f$, i.e., the round of tosses ends either when $M$ successes are obtained or when a restart condition $f(n,m) = 1$ is encountered. Denote by $\alpha_f$ the probability that $M$ heads obtains, $\beta_f = 1 - \alpha_f$ the probability that a restart condition is encountered, and $\tau_f$ the number of coins tossed before the round ends. Now consider a strategy $(f_j, j \geq 1)$. Let the random variable $K$ denote the number of restarts *before the game* finally terminates with $M$ heads. It is clear that $K$ has the distribution

$$\mathbf{P}\{K = k\} = \alpha_{f_{k+1}} \prod_{j=1}^{k} \beta_{f_j}, \qquad k = 0, 1, \ldots. \tag{4}$$

Now conditioned on the event $\{K \geq 1\}$ we have $T_{(f_j, j\geq 1)} = \tau_{f_1} + T_{(f_j, j\geq 2)}$. Further, the events $\{K = 0\}$ and $\{K \geq 1\}$ depend solely on $f_1$, with $\alpha_{f_1} = \mathbf{P}\{K = 0\}$ and $\beta_{f_1} = \mathbf{P}\{K \geq 1\}$. By a simple conditioning argument we then have

$$\begin{aligned} \mathbf{E}\, T_{(f_j, j\geq 1)} &= \mathbf{E}[T_{(f_j, j\geq 1)} | K = 0]\, \mathbf{P}\{K = 0\} + \mathbf{E}[T_{(f_j, j\geq 1)} | K \geq 1]\, \mathbf{P}\{K \geq 1\} \\ &= \mathbf{E}(\tau_{f_1}; K = 0) + [\mathbf{E}(\tau_{f_1} | K \geq 1) + \mathbf{E}(T_{(f_j, j\geq 2)})]\, \mathbf{P}\{K \geq 1\} \\ &= \mathbf{E}(\tau_{f_1}) + \mathbf{E}(T_{(f_j, j\geq 2)})\, \beta_{f_1}. \end{aligned} \tag{5}$$

It follows by induction that

$$\mathbf{E}\, T_{(f_j, j\geq 1)} = \sum_{j=1}^{k} \mathbf{E}(\tau_{f_j}) \prod_{i=1}^{j-1} \beta_{f_i} + \mathbf{E}(T_{(f_j, j\geq k+1)}) \prod_{i=1}^{k} \beta_{f_i} \qquad (k \geq 1). \tag{6}$$

We can now characterise some features of an optimal strategy.

STATIONARITY AND NONRANDOMNESS We say that a strategy $(f_j, j \geq 1) \in \mathcal{S}$ is *stationary* if there exists a (randomised) Boolean function $f : \{0, 1, \ldots, N\} \times \{0, 1, \ldots, M-1\} \to \{0, 1\}$ with

$$f(n,m) = \begin{cases} 1 & \text{with probability } \phi(n,m), \\ 0 & \text{with probability } 1 - \phi(n,m), \end{cases}$$

and such that $f_j = f$ for each $j \geq 1$; we then denote the strategy $(f_j, j \geq 1)$ by $(f)$ and denote the duration of the game $T_{(f_j, j\geq 1)}$ under strategy $(f_j, j \geq 1)$ simply by $T_f$. Now consider any stationary strategy $(f) \in \mathcal{S}$. A direct application of (5) now yields the following fundamental result:

$$\mathbf{E}\,T_f = \frac{\mathbf{E}\,\tau_f}{\alpha_f}, \qquad (f) \in \mathcal{S}, \quad \alpha_f \neq 0. \tag{7}$$

We say that a strategy $(f_j,\, j \geq 1) \in \mathcal{S}$ (governed by restart probabilities $\phi_j(n,m)$) is ***nonrandom*** if $\phi_j(n,m) \in \{0,1\}$ for every choice of $0 \leq n \leq N$, $0 \leq m \leq M-1$, and $j \geq 1$.

**Lemma 2.1** *There exists a stationary, nonrandom optimal strategy.*

REMARK: Equation (7) and Lemma 2.1 can also be demonstrated by setting up the coin tossing problem as a pair of nested Markov decision problems and appealing to results from Markov decision theory (cf. Ross [4], for instance). The prooi given below is elementary.

PROOF: Begin by defining a linear order $\leq$ on $\mathcal{S}$ by $(f_j,\, j \geq 1) \leq (g_j,\, j \geq 1)$ if $\mathbf{E}\,T_{(f_j,\, j\geq 1)} \leq \mathbf{E}\,T_{(g_j,\, j \geq 1)}$. Say that a strategy $(f_j,\, j \geq 1)$ is ***decreasing*** if

$$\forall k \geq 1 : \ (f_{k+j},\, j \geq 1) \leq (f_{k-1+j},\, j \geq 1).$$

The proof of the lemma now proceeds in several steps.

*Claim 1*: If $(f_j,\, j \geq 1) \leq (f_{1+j},\, j \geq 1)$ then $(f_1) \leq (f_j,\, j \geq 1)$.

From (5) and the hypothesis of the claim, it follows that

$$\mathbf{E}\,T_{(f_j,\, j\geq 1)} = \mathbf{E}[\tau_{f_1}] + \mathbf{E}[T_{(f_{1+j},\, j\geq 1)}]\,\beta_{f_1} \geq \mathbf{E}[\tau_{f_1}] + \mathbf{E}[T_{(f_j,\, j\geq 1)}]\,\beta_{f_1}.$$

We can assume $\alpha_{f_1} > 0$ as otherwise $\mathbf{E}\,T_{f_1} = \mathbf{E}\,T_{(f_j,\, j\geq 1)} = \infty$. Using (7) it follows that $\mathbf{E}\,T_{(f_j,\, j \geq 1)} \geq \mathbf{E}(\tau_{f_1})/\alpha_{f_1} = \mathbf{E}\,T_{f_1}$. The claim follows.

*Claim 2*: If $(f_j,\, j \geq 1)$ is not decreasing, then there exists $k \geq 1$ such that $(f_k) \leq (f_j,\, j \geq 1)$.

In fact, let $k$ be the unique integer for which

$$\begin{aligned} (f_{i+j},\, j \geq 1) &\leq (f_{i-1+j},\, j \geq 1), \qquad 1 \leq i \leq k-1, \\ (f_{k-1+j},\, j \geq 1) &\leq (f_{k+j},\, j \geq 1). \end{aligned} \tag{8}$$

(Clearly, $k$ is the smallest integer for which (8) holds.) By transitivity of the linear order $\leq$ it follows that $(f_{k-1+j},\, j \geq 1) \leq (f_j,\, j \geq 1)$, whereas by (8) and Claim 1 we have $(f_k) \leq (f_{k-1+j},\, j \geq 1)$. This proves the claim.

*Claim 3*: $\inf_{(f_j,\, j\geq 1)} \mathbf{E}\,T_{(f_j,\, j\geq 1)} = \inf_{(f)} \mathbf{E}\,T_f$.

By Claim 2, it suffices to show that $\mathbf{E}\,T_{(f_j,\, j\geq 1)} \geq \inf_{(f) \in \mathcal{S}} \mathbf{E}\,T_f$ for any decreasing strategy $(f_j,\, j \geq 1)$. Now, if $\mathbf{E}\,T_{(f_j,\, j\geq 1)} = \infty$, then $(f) \leq (f_j,\, j \geq 1)$ for any choice of $f$. So now suppose that $\mathbf{E}\,T_{(f_j,\, j\geq 1)} < \infty$. As $(f_j,\, j \geq 1)$ is decreasing it follows that the sequence $\{\mathbf{E}\,T_{(f_{k-1+j},\, j\geq 1)},\ k \geq 1\}$ decreases monotonically to a finite limit $T^*$ as $k \to \infty$. Note that by (5) we have

$$\mathbf{E}\,T_{(f_{k-1+j},\, j\geq 1)} = \mathbf{E}[\tau_{f_k}] + \mathbf{E}[T_{(f_{k+j},\, j\geq 1)}]\,\beta_{f_k}.$$

We now assert that the probabilities $\beta_{f_k}$ are bounded away from 1 for large $k$. Indeed, suppose the assertion does not hold. Then, for any $\delta > 0$, we can find arbitrarily large values of $k$ for which $\beta_{f_k} > (1 - \frac{\delta}{T^*})$. Noting that $\mathbf{E}\,\tau_{f_k} \geq 1$ for all $k$, we have the inequality

$$\mathbf{E}\,T_{(f_{k-1+j},\, j\geq 1)} > 1 + \left(1 - \frac{\delta}{T^*}\right)\mathbf{E}[T_{(f_{k+j},\, j\geq 1)}] \geq 1 + \left(1 - \frac{\delta}{T^*}\right)T^* = T^* + (1 - \delta)$$

holding on an unbounded set of values for $k$ for every choice of $\delta > 0$. On the other hand, $\mathbf{E}\,T_{(f_{k-1+j},\,j\geq 1)} \downarrow T^*$ as $k \to \infty$, so that for any $\epsilon > 0$, for large enough values of $k$ we have $T^* \leq \mathbf{E}\,T_{(f_{k-1+j},\,j\geq 1)} < T^* + \epsilon$. Contradiction.

Now fix $\epsilon > 0$ arbitrarily small and (for a suitable choice of $\delta > 0$) select $k(\epsilon)$ so large that for all $k \geq k(\epsilon)$ the following relations hold simultaneously:

$$\beta_{f_k} \leq 1 - \delta,$$
$$T^* \leq \mathbf{E}\,T_{(f_{k+j},\,j\geq 1)} \leq \mathbf{E}\,T_{(f_{k-1+j},\,j\geq 1)} < T^* + \epsilon.$$

By (5) we hence obtain

$$(T^* + \epsilon) - \beta_{f_k} T^* \geq \mathbf{E}\,T_{(f_{k-1+j},\,j\geq 1)} - \beta_{f_k}\,\mathbf{E}\,T_{(f_{k+j},\,j\geq 1)} = \mathbf{E}(\tau_{f_k}), \qquad k \geq k(\epsilon).$$

It follows from (7) that for $k \geq k(\epsilon)$,

$$T^* + \frac{\epsilon}{\delta} \geq T^* + \frac{\epsilon}{\alpha_{f_k}} \geq \frac{\mathbf{E}(\tau_{f_k})}{\alpha_{f_k}} = \mathbf{E}\,T_{f_k} \geq \inf_k \mathbf{E}\,T_{f_k}.$$

As this holds for every $\epsilon > 0$ it follows that

$$T^* \geq \inf_k \mathbf{E}\,T_{f_k} \geq \inf_{(f)\in\mathcal{S}} \mathbf{E}\,T_f.$$

This concludes the proof of the claim.

*Claim 4*: There exists a stationary, nonrandom strategy $(f^*)$ such that $\mathbf{E}\,T_{f^*} = \inf_{(f)\in\mathcal{S}} \mathbf{E}\,T_f$.

Let $f^{(i)}$, $i = 1, \ldots, 2^{NM}$ enumerate all *deterministic functions*

$$f^{(i)} : \{0, 1, \ldots, N\} \times \{0, 1, \ldots, M-1\} \longrightarrow \{0, 1\}$$

with $f^{(i)}(N, m) \equiv 1$ for each $m$ and $i$. Now consider a randomised stationary strategy $(f) \in \mathcal{S}$ (with a corresponding specification of probabilities $\phi(n, m)$). In any round $f$ will then have a sample realisation $f^{(i)}$ with probability $\mathbf{P}\{f = f^{(i)}\} = \prod_{n,m} \phi^{(i)}(n, m) \equiv p^{(i)}$, where[3]

$$\begin{aligned}\phi^{(i)}(n,m) &= f^{(i)}(n,m)\phi(n,m) + (1 - f^{(i)}(n,m))(1 - \phi(n,m)) \\ &= \begin{cases} \phi(n,m) & \text{if } f^{(i)}(n,m) = 1 \\ 1 - \phi(n,m) & \text{if } f^{(i)}(n,m) = 0. \end{cases}\end{aligned}$$

Starting with (7), some reflection now shows that

$$\begin{aligned}\mathbf{E}\,T_f &= \frac{\mathbf{E}\,\tau_f}{\alpha_f} = \frac{\sum_{i=1}^{2^{NM}} \mathbf{E}[\tau_f | f = f^{(i)}]\,\mathbf{P}\{f = f^{(i)}\}}{\sum_{i=1}^{2^{NM}} \alpha_{f^{(i)}}\,\mathbf{P}\{f = f^{(i)}\}} \\ &= \frac{\sum_{i=1}^{2^{NM}} \mathbf{E}[\tau_{f^{(i)}}]\,p^{(i)}}{\sum_{i=1}^{2^{NM}} \alpha_{f^{(i)}}\,p^{(i)}} \geq \min_i \frac{\mathbf{E}[\tau_{f^{(i)}}]}{\alpha_{f^{(i)}}} = \min_i \mathbf{E}\,T_{f^{(i)}}.\end{aligned}$$

As the number of deterministic functions $f^{(i)}$ is finite, there exists $f^* \in \{f^{(1)}, \ldots, f^{(2^{NM})}\}$ for which $\mathbf{E}\,T_{f^*} = \min_i \mathbf{E}\,T_{f^{(i)}}$. Hence $(f^*) \preceq (f)$ for every $(f) \in \mathcal{S}$. This completes the proof of the claim.

Claims 3 and 4 complete the proof of the lemma. ∎

[3] This tacitly assumes that for any choices $\alpha_{n,m} \in \{0, 1\}$, the events $\{f(n,m) = \alpha_{n,m}\}$ are independent over all choices of $(n, m)$. As can be readily seen, the proof works without change even if the independence assumption is relaxed, i.e., the joint distribution of the random variables $f(n, m)$ is not a product distribution.

CONSISTENCY We say that a stationary, nonrandom strategy $(f)$ is *consistent* (or simply, $f$ is consistent) if $f(n,m)$ is increasing in $n$ and decreasing in $m$, i.e.,

$$f(n,m) = 1 \Longrightarrow f(k,l) = 1 \qquad (k \geq n, \quad l \leq m).$$

**Lemma 2.2** *There exists a consistent optimal strategy.*

This is in keeping with intuition; if, for instance, $f(n,m) = 1$ but $f(k,m) = 0$ for some $k > n$ then we would expect the strategy to be suboptimal as increasing the number of tosses in a round while keeping the number of successes fixed at $m$ would not seem to improve the situation. We defer a proof of this assertion till Section 4 where we provide a simple direct proof by considering a related optimal stopping problem.

If $f$ is consistent, we define the *restart boundary* $F(n)$ of $f$ by $F(n) = \max\{m : f(n,m) = 1\}$ if there is any $m$ for which $f(n,m) = 1$; else we set $F(n) = -1$. We will also occasionally refer to $F(n)$ as the restart boundary of the consistent strategy $(f)$. Consistent strategies can be conveniently represented by means of the restart boundary in the $(n,m)$ plane. Some examples are illustrated in Fig. 1. Note that the boundary of consistent strategies is monotone.

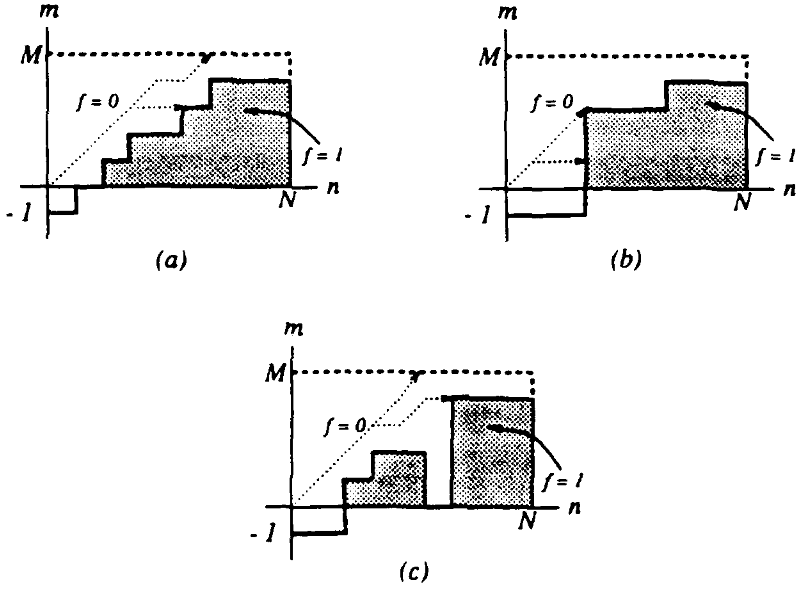


Figure 1: (a) A consistent, parsimonious strategy; (b) a consistent strategy; (c) an inconsistent strategy. The parameter $n$ represents the number of coin tosses in a round, and $m$ denotes the number of heads obtained. The dotted lines show some possible sample paths. The dashed lines indicate the termination boundary ($M$ heads achieved), and the solid lines enclosing the shaded areas denote the restart boundaries. The shaded areas correspond to the restart region.

PARSIMONY We say that a consistent strategy $(f)$ is *parsimonious*[4] (or simply, $f$ is parsimonious) if there are $M$ distinct points of increase on the boundary, i.e., $F(n) \leq F(n+1) \leq F(n) + 1$, $0 \leq n \leq N$. In particular, define the sequence of points $n_m$ by

$$n_m = \min\{n : f(n,m) = 1\}, \qquad m = 0, 1, \ldots, M-1.$$

The points $n_m$ define the points of increase of the boundary $F(n)$. For a parsimonious strategy $(f)$ then, we require that $n_{m+1} > n_m$, $0 \leq m \leq M-2$; in particular, $F(n_m) = m$, $0 \leq m \leq M-1$.

[4] The terminology relates to the size of permitted jumps.

**Lemma 2.3** *There exists a parsimonious optimal strategy, i.e., a consistent optimal strategy* $(f^*)$ *with boundary* $F^*$ *satisfying* $F^*(n) \leq F^*(n+1) \leq F^*(n) + 1,\ 0 \leq n \leq N-1$.

PROOF: Consider a consistent optimal strategy with boundary $F^*$. Let $S_n$ $(n \geq 1)$ denote the number of successes in $n$ tosses in any round following a restart. Assume that $S_i > F^*(i)$, $i \leq n$ and $S_n = F^*(n) + 1$. Thus, the game proceeds without restarting after the $n$th toss in the round. If $F^*(n+1) \geq F^*(n) + 2$ then a restart is forced regardless of the result of the $(n+1)$th toss. But this implies that replacing the boundary point $F^*(n)$ by $F^*(n) + 1$ yields a superior strategy, this contradicting the supposed optimality of $F^*$. ∎

RESTART ON FAILURE What if in any restart the first toss results in a tail? The following assertion maintains that an optimal strategy would restart if a round begins with a failure.

**Lemma 2.4** *There exists a parsimonious optimal strategy whose boundary* $F^*$ *has its first point of increase at* $n_0 = 1$: $F^*(0) = -1,\ F^*(n_0) = F^*(1) = 0$.

PROOF: If a round begins with a failure there are two options: continue the round or restart. Consider *sample paths in the* $(n, m)$ *plane* where $n$ denotes the number of tosses in a round and $m$ denotes the number of successes. Restarting implies starting a sample path anew from $(0,0)$, while continuing the round implies initiating sample paths from $(1,0)$. Now consider the boundary $F$ of any consistent strategy and assume $F(0) = F(1) = -1$. Given that a round begins with a failure, it is clear that the number of sample paths lying strictly above the boundary $F$ is larger if the game is restarted so that replacing the boundary point $F(1) = -1$ by $F(1) = 0$ *yields a superior (or at any rate, not inferior) strategy.* ∎

We encapsulate our findings in the following statement.

**Theorem 2.5** *Let* $\mathcal{P} \subset \mathcal{S}$ *denote the family of parsimonious strategies whose boundaries have their first point of increase at* $n_0 = 1$. *Then*

$$D(N, M) = \inf_{\mathcal{S}} \mathbf{E}\, T_{(f_j, j \geq 1)} = \inf_{\mathcal{P}} \mathbf{E}\, T_f .$$

REMARKS: Note that we are ensured that the probability of obtaining $M$ heads in any round of tosses is non-zero for any parsimonious strategy which has a first point of increase at $n_0 \geq 1$. These two conditions hence eliminate strategies for which success is impossible [see the consistent strategy shown in Fig. 1(b)]. Note also that the restriction to stationary, nonrandom strategies reduces the search problem from an infinity of possible stationary, random strategies to a set of $2^{NM}$ stationary, nonrandom strategies. Restricting attention to consistent strategies further reduces the search space, and the constraints of parsimony with a first point of increase $n_0 = 1$ results in the reduced search space $\mathcal{P}$ of $\binom{N-1}{M-1}$ strategies. We outline a method for efficiently searching the space $\mathcal{P}$ for an optimal strategy in Section 4.

# 3 EXPECTED DURATION

We consider now the problem of estimating the minimum expected duration of the game $D(N, M)$. We begin by showing a strict lower bound for $D(N, M)$, valid for every fixed $M$. Recall that we can restrict ourselves to considering the subfamily of stationary strategies.

**Theorem 3.1** *For every* $M$, $\{D(N, M)\}$ *is a monotone sequence in* $N$ *decreasing to the limit* $D(\infty, M) = 2M$ *as* $N \to \infty$.

PROOF: Fix $M$. Let $\bar{S}(N)$ denote the family of stationary strategies when the number of available coins is $N$. If $N < N'$ it is clear that $\bar{S}(N)$ is contained in $\bar{S}(N')$ in the following sense: for any strategy $(f)$ in $\bar{S}(N)$ there exist strategies $(f')$ in $\bar{S}(N')$ such that the restriction of $f'$ to $\{0, \dots, N\} \times \{0, \dots, M-1\}$ is equal to $f$; in addition, it is easy to see that these strategies will result in the same expected duration as $(f)$ because of the restriction $f(N, m) = 1$. Hence $D(N', M) \leq D(N, M)$.

Now assume that there is an infinite store of coins. It is clear that an optimal strategy is one that never restarts the game. Let $D(\infty, M)$ denote the expected duration of the game under this optimal strategy. Let $\alpha_M(k)$ denote the probability that the $M$th success occurs at the $k$th trial. As a first passage to $M$ involves a prior first passage through $M-1$ we have the convolutional relationship

$$\alpha_M(k) = \sum_{j=1}^{k-1} \alpha_{M-1}(j) \alpha_1(k-j).$$

Consider the generating function $A_M(s) = \sum_{k=0}^{\infty} \alpha_M(k) s^k = [A_1(s)]^M$. As $\alpha_1(0) = 0$ and $\alpha_1(k) = 2^{-k}$, we have $A_1(s) = s/(2-s)$. It follows that $D(\infty, M) = A'_M(1) = 2M$. ∎

Consider any consistent strategy $(f)$. As before, define the sequence of points $n_m$, $0 \leq m \leq M-1$ by

$$n_m = \min\{n : f(n, m) = 1\}.$$

(If $(f)$ is parsimonious, these are just the $M$ point of increase of the boundary.) Now define the sets of pairs

$$\begin{aligned} Q &= \{(n, M) : 0 \leq n \leq N\}, \\ R &= \{(n_m, m) : 0 \leq m \leq M-1\}. \end{aligned}$$

The set $Q$ can be identified as the success termination state (i.e., $M$ heads are achieved in the round), and the set $R$ as the restart state (i.e., the restart boundary is encountered during the round). Let $S_n$ denote the number of heads obtained in a sequence of $n \leq N$ tosses. We momentarily suppress the dependence on $f$ and write

$$\tau = \min\{n : (n, S_n) \in Q \cup R\}$$

for the number of tosses in a round before either the game ends with $M$ successes, or the round of tosses is terminated and the game restarted. Also let

$$p_m = \mathbf{P}\{\tau = n_m, S_\tau = m\} \tag{9}$$

denote the probability that, in any round of tosses, a restart occurs following the $n_m$th toss. (On the $(n, m)$ plane then, this corresponds to a sample path which lies above the boundary for $n < n_m$ and intersects the boundary at the point $(n_m, m)$.) It follows that $\beta = \sum_{m=0}^{M-1} p_m$ is the probability that a restart occurs during any round of tosses, and $\alpha = 1 - \sum_{m=0}^{M-1} p_m$ is the probability that the game terminates with $M$ successes in a given round of tosses.

The following recursive form for the probabilities $p_m$ admits of efficient evaluation.

**Lemma 3.2** *The probabilities $p_m$, $0 \leq m \leq M-1$ satisfy the following recursion:*

$$\begin{aligned} &\text{BASE:} && p_0 = 2^{-n_0}, \\ &\text{RECURSION:} && p_m = \binom{n_m}{m} 2^{-n_m} - \sum_{i=0}^{m-1} \binom{n_m - n_i}{m - i} 2^{-(n_m - n_i)} p_i, \qquad m \geq 1. \end{aligned} \tag{10}$$

PROOF: It is clear that $p_0 = \mathbf{P}\{S_{n_0} = 0\} = 2^{-n_0}$. Now consider $1 \le m \le M-1$. Rewriting $p_m$ in an alternative form, we have

$$\begin{aligned} p_m &= \mathbf{P}\{S_{n_m} = m, \tau > n_{m-1}\} \;=\; \mathbf{P}\{S_{n_m} = m\} - \mathbf{P}\{S_{n_m} = m, \tau \le n_{m-1}\} \\ &= \binom{n_m}{m} 2^{-n_m} - \sum_{i=0}^{m-1} \mathbf{P}\{S_{n_m} = m, \tau = n_i\}. \end{aligned} \tag{11}$$

For $i < m$, the event $\{S_{n_m} = m, \tau = n_i\}$ is contained in the set of sample points $\{S_{n_i} = i\}$; it hence follows that

$$\begin{aligned} \mathbf{P}\{S_{n_m} = m, \tau = n_i\} &= \mathbf{P}\{S_{n_m} = m \mid (\tau, S_\tau) = (n_i, i)\}\, \mathbf{P}\{(\tau, S_\tau) = (n_i, i)\} \\ &= \mathbf{P}\{S_{n_m} - S_{n_i} = m - i\}\, p_i \\ &= \binom{n_m - n_i}{m - i} 2^{-(n_m - n_i)} p_i, \end{aligned}$$

the last but one equation following from the independence of the trials. Substituting in (11) completes the proof. ∎

The probabilities $p_m$ (corresponding to any given consistent strategy $(f)$) turn out to be extraordinarily useful as they allow us to write down an explicit expression for the expected duration of the game under strategy $(f)$.

**Theorem 3.3** *For any consistent strategy $(f)$,*

$$\mathbf{E}\,T_f = 2M + \frac{2\sum_{m=0}^{M-1} m p_m}{1 - \sum_{m=0}^{M-1} p_m}. \tag{12}$$

REMARKS: Note that to compute $\mathbf{E}\,T_f$ it suffices to specify the points of increase $n_m$, $0 \le m \le M-1$. Note also that $\mathbf{E}\,T_f > 2M$ in accordance with Theorem 3.1.

PROOF: An appeal to (7) yields $\mathbf{E}\,T_f = \mathbf{E}(\tau)/\alpha$. Now $S_\tau = \sum_{i=1}^{\tau} X_i$, so that Wald's equation gives us $\mathbf{E}\,S_\tau = (\mathbf{E}\,\tau)(\mathbf{E}\,X_i) = \mathbf{E}(\tau)/2$. We hence have

$$\begin{aligned} \mathbf{E}\,T_f &= \frac{2}{\alpha}\mathbf{E}\,S_\tau \;=\; \frac{2}{\alpha}\left(\mathbf{E}[S_\tau; (\tau, S_\tau) \in Q] + \mathbf{E}[S_\tau; (\tau, S_\tau) \in R]\right) \\ &= \frac{2}{\alpha}\left(M\alpha + \mathbf{E}[S_\tau; (\tau, S_\tau) \in R]\right) \;=\; 2M + \frac{2}{\alpha}\mathbf{E}[S_\tau; (\tau, S_\tau) \in R]. \end{aligned}$$

With $p_m$ defined as in (9), we now have

$$\mathbf{E}[S_\tau; (\tau, S_\tau) \in R] = \sum_{m=0}^{M-1} m\,\mathbf{P}\{(\tau, S_\tau) = (n_m, m)\} = \sum_{m=0}^{M-1} m p_m.$$

Recalling that $\alpha = 1 - \sum_{m=0}^{M-1} p_m$ completes the proof. ∎

REMARK: The model we are concerned with in this paper does not impose a "cost" for restarting. A more general situation where there is a cost $c_k$ associated with $k$ restarts is, however, easily incorporated into the analysis. For instance, if $K$ denotes the number of restarts, let us define the cost $C_f$ of the consistent strategy $(f)$ by

$$\forall k \ge 0 : C_f = c_k\, \mathbf{E}[T_f \mid K = k] \qquad \text{with probability } \alpha\beta^k.$$

Note that $K$ has the geometric distribution (see (4)) so that $\alpha\beta^k$ is just the probability of the event $\{K = k\}$. The expected cost of strategy $(f)$ is hence

$$\mathbf{E}\, C_f = \sum_{k=0}^{\infty} c_k\, \mathbf{E}[T_f; K = k] = \alpha \sum_{k=0}^{\infty} c_k \beta^k\, \mathbf{E}[T_f | K = k].$$

Let us formally define

$$C_1 = \alpha \sum_{k=0}^{\infty} c_k \beta^k, \qquad C_2 = \alpha \sum_{k=0}^{\infty} k c_k \beta^k.$$

Following the previous analysis we can now readily show

$$\mathbf{E}\, C_f = 2C_1 M + \frac{2C_2 \sum_{m=0}^{M-1} m p_m}{\sum_{m=0}^{M-1} p_m}.$$

For the rest of the paper we adopt the ***constant cost model***, $c_k \equiv 1$ $(k \geq 0)$. It is easy to verify that for this model $C_1 = 1$ and $C_2 = \beta/\alpha$ so that $\mathbf{E}\, C_f = \mathbf{E}\, T_f$ agrees with Theorem 3.3. As instances of other cost models, a ***linear cost model*** $c_k = \gamma k$ results in $C_1 = \gamma\beta/\alpha$ and $C_2 = \gamma\beta(1+\beta)/\alpha^2$, and an ***exponential cost model*** $c_k = \lambda\theta^k$ (with $\theta < 1/\beta$) results in $C_1 = \lambda\alpha/(1-\theta\beta)$ and $C_2 = \lambda\theta\alpha\beta/(1-\theta\beta)^2$.

The following examples illustrate the utility of Lemma 3.2 and Theorem 3.3 with explicit calculations of the expected duration of the game for diverse strategies.

EXAMPLE: *Indolent Strategy*

The strategy of indolence $(f_I)$ prescribes that we wait till all $N$ coins are tossed before restarting the game. In particular, the strategy is specified by

$$f_I(n,m) = \begin{cases} 0 & \text{if } 0 \leq n \leq N-1, \\ 1 & \text{if } n = N. \end{cases}$$

The strategy is clearly consistent (but not parsimonious) and we have $n_m = N$, $0 \leq m \leq M-1$ (see Fig. 2). Direct calculation then yields

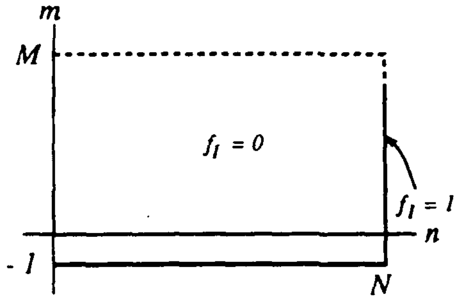


Figure 2: The boundary for the Indolent Strategy.

$$p_m = \mathbf{P}\{S_N = m\} = \binom{N}{m} 2^{-N}, \qquad m = 0, 1, \ldots, M-1. \tag{13}$$

Equation (12) now readily yields

$$\mathbf{E}\, T_I = 2M + \frac{2\sum_{m=0}^{M-1} m \binom{N}{m}}{\sum_{m=M}^{N} \binom{N}{m}}$$

for the expected duration $\mathbf{E}\,T_l$ of the game. ∎

Example: *Hope Springs Eternal*

This strategy restarts the game only when the number of tails in a round of tosses reaches $N - M + 1$, i.e., it is futile to proceed with the round any further. The strategy is parsimonious and its boundary $F_H$ is given by

$$F_H(n) = \begin{cases} -1 & \text{if } 0 \le n \le N - M, \\ n - N + M - 1 & \text{if } N - M + 1 \le n \le N. \end{cases}$$

The points of increase of the boundary are $n_m = m + N - M + 1$, $0 \le m \le M - 1$ (see Fig. 3). We will find it convenient to use the representation $n_m = m + n_0$ with $n_0 = N - M + 1$. We now claim that

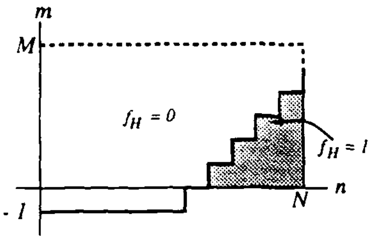


Figure 3: The boundary for Hope Springs Eternal.

$$p_m = \binom{n_0 + m - 1}{m} 2^{-(n_0+m)}, \qquad m = 0, \ldots, M - 1. \tag{14}$$

We prove the result by induction on $m$. It is clear that $p_0 = \mathbf{P}\{S_{n_0} = 0\} = 2^{-n_0}$, so that (14) holds for $m = 0$. Let us now assume that (14) holds for $i \le m - 1$. As $n_m - n_i = m - i$, it follows from (10) and the inductive hypothesis that

$$\begin{aligned} p_m &= \binom{n_0 + m}{m} 2^{-(n_0+m)} - \sum_{i=0}^{m-1} 2^{-(m-i)} p_i \\ &= 2^{-(n_0+m)} \left[ \binom{n_0 + m}{m} - \sum_{i=0}^{m-1} \binom{n_0 + i - 1}{i} \right] \\ &= 2^{-(n_0+m)} \binom{n_0 + m - 1}{m}, \end{aligned}$$

the last step following by repeated application of the binomial identity

$$\binom{x}{r-1} + \binom{x}{r} = \binom{x+1}{r}.$$

This concludes the proof of the claim (14). Equation (12) now yields

$$\mathbf{E}\,T_H = 2M + \frac{2 \sum_{m=0}^{M-1} m 2^{-(n_0+m)} \binom{n_0 + m - 1}{m}}{1 - \sum_{m=0}^{M-1} 2^{-(n_0+m)} \binom{n_0 + m - 1}{m}}$$

for the expected duration $\mathbf{E}\,T_H$ of the game. ∎

EXAMPLE: *Step in Time*

This strategy decrees a restart when the number of tails leads the number of heads by a fixed amount, say $L$. This is again a parsimonious strategy with boundary $F_S$ given by

$$F_S(n) = \begin{cases} -1 & \text{if } 0 \leq n \leq L-1, \\ n - L & \text{if } L \leq n \leq L+M-1, \\ M-1 & \text{if } L+M \leq n \leq N. \end{cases}$$

Hope Springs Eternal is hence a specific case of this strategy with the choice $L = N - M + 1$. The points of increase of the boundary are $n_m = m + L$, $0 \leq m \leq M-1$ (see Fig. 4). It is readily seen that this is equivalent to Hope Springs Eternal with a reduced store of $N' = L + M - 1$ coins. The probabilities $p_m$ for Step in Time can hence be directly inferred from (14):

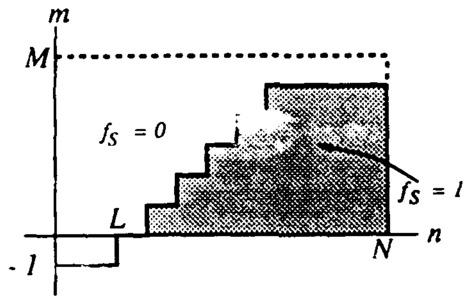


Figure 4: The boundary for Step in Time.

$$p_m = 2^{-(L+m)} \binom{L+m-1}{m}, \qquad m = 0, \ldots, M-1. \tag{15}$$

An appeal to (12) again yields

$$\mathbf{E}\, T_S = 2M + \frac{2\sum_{m=0}^{M-1} m 2^{-(L+m)} \binom{L+m-1}{m}}{1 - \sum_{m=0}^{M-1} 2^{-(L+m)} \binom{L+m-1}{m}}$$

for the expected duration $\mathbf{E}\, T_S$ of the game. ∎

EXAMPLE: *Ballot Strategy*

Consider the parsimonious strategy indicated schematically in Fig. 5. The points of increase of the boundary of the strategy are specified by $n_m = 1 + \lceil mK \rceil$, $0 \leq m \leq M-1$, where $K = (N-1)/(M-1)$. (Note that $n_0 = 1$ and $n_{M-1} = N$.) It is easy to verify that the following inequalities hold:

$$\frac{M-m}{N-n_m} > \frac{M}{N}, \qquad m = 0, \ldots, M-1.$$

Thus, we can interpret the Ballot Strategy as follows: the strategy decrees that a round be restarted iff the ratio of the number of additional heads needed to the number of remaining coins is larger than it was at the start of the game.[5]

For general values of $K$, a closed form for the probabilities $p_m$ is hard to secure, and the general recursion (10) must be appealed to. When $K$ is an integer, however, the following explicit form can be obtained:

$$p_m = \frac{1}{mK+1} \binom{mK+1}{m} 2^{-(mK+1)}, \qquad m = 0, \ldots, M-1. \tag{16}$$

[5] As in the classical ballot theorem (cf. Feller [5], for instance), for a round to continue, we require the initial ratio $M/N$ of heads to coins to lead throughout the round.

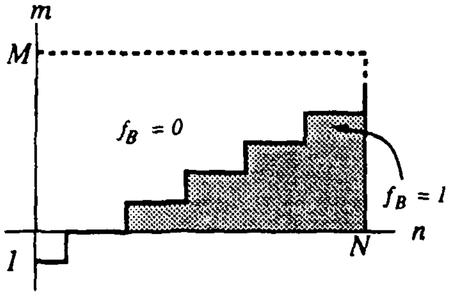


Figure 5: The boundary for the Ballot Strategy.

The proof is again by induction on $m$. As $n_0 = 1$, it is clear that $p_0 = \mathbf{P}\{S_1 = 0\} = 1/2$, so that (16) holds for $m = 0$. Now assume that (16) holds for $i \leq m - 1$. Note the equivalent representation

$$p_i = \frac{1}{i}\binom{iK}{i-1}2^{-(iK+1)}, \qquad i = 1, \ldots, m-1.$$

As $n_m - n_i = (m-i)K$, it follows from (10) and the inductive hypothesis that

$$\begin{aligned} p_m &= 2^{-(mK+1)}\left[\binom{mK+1}{m} - \binom{mK}{m} - \sum_{i=1}^{m-1}\frac{1}{i}\binom{iK}{i-1}\binom{(m-i)K}{m-i}\right] \\ &= 2^{-(mK+1)}\left[\binom{mK}{m-1} - \sum_{i=1}^{m-1}\frac{1}{i}\binom{iK}{i-1}\binom{(m-i)K}{m-i}\right]. \end{aligned}$$

An appeal to the general combinatorial identity [6]

$$\sum_{i=1}^{m}\frac{1}{i}\binom{iK}{i-1}\binom{(m-i)K}{m-i} = \binom{mK}{m-1}$$

completes the proof of (16) for integer $K$. Equation (12) now yields

$$\mathbf{E}\,T_B = 2M + \frac{4\sum_{m=1}^{M-1}\binom{mK}{m-1}2^{-(mK+1)}}{1 - 2\sum_{m=1}^{M-1}\frac{1}{m}\binom{mK}{m-1}2^{-(mK+1)}}$$

for the expected duration $\mathbf{E}\,T_B$ of the game when $K$ is an integer.

The general system of probabilities $\{p_m\}$ (for arbitrary $K$) corresponding to the Ballot Strategy can also be determined from the generating function for which an explicit form can be shown. Define

$$\tau_n = \mathbf{P}\{\tau = n, S_\tau = (n-1)/K\}, \qquad n = 1, 2, \ldots .$$

Note that $\tau_{n_m} = p_m$ for $0 \leq m \leq M-1$. (In general, $\tau_n = 0$ if $n \neq 1 + mK$ for some positive integer $m$.) Thus, $\beta = \sum_{m=0}^{M-1} p_m = \sum_{n=1}^{N}\tau_n$ is the probability of a restart in a round, and $\alpha = 1 - \beta$ is the probability of attaining $M$ successes in a given round. Let

$$\dot{\tau} = \inf\{n : S_n - (n-1)/K \leq 0\}.$$

Clearly, $\dot{\tau}$ has distribution $\{\tau_n\}$. Consider the generating function $G_{\dot{\tau}}(s) = \sum_{n=1}^{\infty}\tau_n s^n$. We can now directly apply a result from Feller [5, page 413] to obtain the expression

$$\begin{aligned} G_{\dot{\tau}}(s) &= 1 - \exp\left[-\sum_{n=1}^{\infty}\frac{s^n}{n}\mathbf{P}\{S_n \geq (n-1)/K\}\right] \\ &= 1 - \exp\left[-\sum_{n=1}^{\infty}\frac{(s/2)^n}{n}\sum_{i \geq (n-1)/K}\binom{n}{i}\right]. \end{aligned}$$

The distribution $\{\tau_n\}$ is thus completely determined, and hence so are the probabilities $\{p_m\}$. ∎

While explicit estimates of the duration of the game under any strategy are thus readily generated from the recursion (10), a careful analysis of the (asymptotic) behaviour of the expected duration of the game under a given strategy requires substantial algebraic effort, even for the simple strategies described here. Nonetheless, sharp results (Theorem 3.6; see also the remarks following the theorem) can be inferred about the *minimum expected duration* $D(N, M)$ of the game by a consideration of the simplest of strategies—the Indolent Strategy.

We begin with a preliminary result due to W. Hoeffding [7].

**Lemma 3.4** *Let $N$ be any positive integer, $\eta \in (0,1)$ and $c \in (0,1)$ any fixed parameters. Then:*

$$\max\left\{\sum_{m\leq(1-c)\eta N}\binom{N}{m}\eta^m(1-\eta)^{N-m},\ \sum_{m\geq(1+c)\eta N}\binom{N}{m}\eta^m(1-\eta)^{N-m}\right\}\leq e^{-2c^2\eta^2N}.$$

This exponential bound for the tail of the binomial is not the sharpest possible, but suffices for our purposes.

**Theorem 3.5** *Let $T_I$ denote the duration of the game under the Indolent Strategy. Then*

$$\frac{1}{N}\,\mathbf{E}\,T_I \leq D(N, M) \leq \mathbf{E}\,T_I.$$

PROOF: The upper bound for $D(N, M)$ is obvious. Now recall that for any consistent strategy $(f)$, we have from (7) that $\mathbf{E}\,T_f(N, M) = \mathbf{E}(\tau_f)/\alpha_f$. Let $\tau_I$ and $\alpha_I$ denote the corresponding values of $\tau_f$ and $\alpha_f$ for the Indolent Strategy. *It is clear that* $\alpha_I = \sup_f \alpha_f$. Also, for any parsimonious strategy $(f) \in \mathcal{P}$, we have $1 \leq \mathbf{E}\,\tau_f \leq N$. Thus, $D(N, M) \geq \inf_{(f)\in\mathcal{P}} \mathbf{E}\,\tau_f / \sup_{(f)\in\mathcal{P}} \alpha_f \geq 1/\alpha_I \geq \mathbf{E}\,\tau_I/N\alpha_I = \mathbf{E}\,T_I/N$. ∎

We can now directly apply Theorem 3.5 to obtain the following result which shows two distinct domains of behaviour for the minimum expected duration of the game.

**Theorem 3.6** *Let $c > 0$ be any fixed parameter and $N$ any positive integer. Then:*

*(a) If $M \leq \frac{1}{2}(1-c)N$, then $2M \leq D(N, M) < 2M\left[1+\frac{e^{-c^2N/2}}{1-e^{-c^2N/2}}\right]$;*

*(b) If $M = N/2$, then $N \leq D(N, N/2) < 2N$;*

*(c) If $M \geq \frac{1}{2}(1+c)N$, then $D(N, M) > \frac{e^{c^2N/2}}{N}$.*

REMARKS: Slightly tighter (if messier) exponents can be obtained using Chernoff's bound for the tail of the binomial instead of the Hoeffding bounds of Lemma 3.4. In particular, we can replace the exponents $c^2/2$ by $\ln 2 - H[(1-c)/2]$ throughout in the above bounds for $D(N, M)$. Here ln denotes a logarithm to base $e$ and $H(x) = -x\ln x - (1-x)\ln(1-x)$ is the binary entropy function in nats. As Chernoff's bound is known to be exponentially tight, this in particular implies the following stronger asymptotic result: *If $M = \frac{1}{2}(1+c)N + o(N)$ for any positive constant $c$, then* $\ln D(N, M) \sim [\ln 2 - H\{(1-c)/2\}]N$ *as* $N \to \infty$ .

Note the abrupt, threshold change in behaviour of the optimal strategy around $M = N/2$ where the minimum expected duration of the game goes from linear to exponential in $M$. The moral of the story is that for $M < N/2$, essentially any strategy (including the Indolent Strategy!) yields performance comparable to the optimal strategy: all strategies in this regime have expected durations $2M + O(e^{-c_1N})$ for a positive constant $c_1$. For $M > N/2$ on the other hand, all strategies have expected duration $\Omega(e^{c_2 N})$ for a positive constant $c_2$.

Proof: Consider the Indolent Strategy. When $M = \frac{1}{2}(1-c)N$ for some fixed choice of $c > 0$, Lemma 3.4 and (13) yield the bound[6]

$$\alpha = 1 - \sum_{m=0}^{M-1} p_m = 1 - 2^{-N} \sum_{m=0}^{M-1} \binom{N}{m} \geq 1 - e^{-c^2 N/2}.$$

A similar application of Lemma 3.4 yields

$$\sum_{m=0}^{M-1} m p_m \leq M \sum_{m=0}^{M-1} p_m = M 2^{-N} \sum_{m=0}^{M-1} \binom{N}{m} \leq M e^{-c^2 N/2}.$$

From (12), we then have

$$\mathbf{E}\, T_I \leq 2M + \frac{2Me^{-c^2 N/2}}{1 - e^{-c^2 N/2}}, \qquad M \leq \frac{1}{2}(1-c)N, \tag{17}$$

the result holding for $M < \frac{1}{2}(1-c)N$ because $\mathbf{E}\, T_I$ decreases monotonically as $M$ decreases for every fixed $N$.

Now consider the case $M = N/2$. From (13) we again have

$$\alpha = 1 - \sum_{m=0}^{M-1} p_m = 1 - 2^{-N} \sum_{m \leq N/2-1} \binom{N}{m} = \frac{1}{2} + O\left(\frac{1}{\sqrt{N}}\right),$$

and

$$\sum_{m=0}^{M-1} m p_m < M 2^{-N} \sum_{m \leq N/2-1} \binom{N}{m} \leq \frac{M}{2} = \frac{N}{4}.$$

Substituting in (12) we have

$$\mathbf{E}\, T_I < N + N \left[1 - O\left(N^{-1/2}\right)\right] \leq 2N, \qquad M = N/2. \tag{18}$$

Finally, when $M = \frac{1}{2}(1+c)N$ for any fixed choice of $c > 0$, Lemma 3.4 and (13) again yield the bound

$$\alpha = \sum_{m \geq (1/2+c)N} \binom{N}{k} 2^{-N} \leq e^{-c^2 N/2}.$$

Also, $\mathbf{E}\, \tau \geq 1$. We then have from (7) that

$$\mathbf{E}\, T_I = \frac{\mathbf{E}\, \tau}{\alpha} \geq e^{c^2 N/2}, \qquad M \geq \frac{1}{2}(1+c)N, \tag{19}$$

the bound holding for $M > \frac{1}{2}(1+c)N$ again by the monotonicity of $\mathbf{E}\, T_I$.

The lower bound on $D(N,M)$ in part (a) of the theorem follows from Theorem 3.1, while the upper bound follows from (17) and the upper bound of Theorem 3.5. The lower bound for $D(N,M)$ in part (b) of the theorem again follows from Theorem 3.1, while the upper bound follows from (18) and the upper bound of Theorem 3.5. Part (c) of the theorem follows from (19) and the lower bound of Theorem 3.5. ∎

[6] We ignore, for the sake of notational economy, the fairly transparent details with regard to rounding to the nearest integer.

When $M \leq N/2$, the expected duration for the Indolent Strategy is (from (17) and (18)) within an additive factor of $N$ of the minimum expected duration of the game. When $M \geq \frac{1}{2}(1+c)N$, the probability of achieving $M$ successes in a round is exponentially small: $\alpha = O(e^{-c^2N/2})$. The expected duration of a round, on the other hand, increases linearly at best: $1 \leq \mathbf{E}\,\tau \leq N$. From (7) and the above proof, it hence follows that the expected duration of the game is dictated predominantly by the factor $1/\alpha$ in this regime. Note that the largest value for $\alpha$ obtains for the Indolent Strategy, so that we obtain the sharpest exponent with $\mathbf{E}\,T_I \geq \frac{N}{2}e^{c^2N/2}$ for this strategy. An optimal strategy can save at best in the $\mathbf{E}\,\tau$ term (bounded between 1 and $N$), but cannot obtain better exponents.

Explicit coefficients can also be obtained for the other strategies we have considered by using (10) and (12). For instance, consider the strategy Hope Springs Eternal. From (14) we can write

$$p_m = 2^{-n_0}\left(\frac{n_0+m-1}{2m}\right)\left(\frac{n_0+m-2}{2(m-1)}\right)\cdots\left(\frac{n_0}{2}\right).$$

Each product term is of the form

$$\frac{n_0+m-k-1}{2(m-k)} = \frac{(n_0-1)+(m-k)}{(m-k)+(m-k)}.$$

If $M \leq \frac{1}{2}(1-c)N$, then $N-M > m$, $0 \leq m \leq M-1$, so that each of the product terms is larger than 1. (Recall $n_0 = N-M+1$.) Hence $p_m$ is monotone increasing. Hence

$$\begin{aligned} \mathbf{E}[S_\tau;(\tau,S_\tau)\in R] &\leq M2^{-(n_0+M-1)}\binom{n_0+M-2}{n_0-1} \\ &= M2^{-N}\binom{N-1}{N-M+2} = O(e^{-c_1N}) \end{aligned}$$

where $c_1$ is a positive constant which can be expressed in terms of the binary entropy function. Further, $\alpha = 1-O(e^{-c_2N})$ for a positive constant $c_2$. Thus, $\mathbf{E}\,T_H = 2M[1+O(e^{-c_1N})]$, as expected. Similarly, using (15) for Step in Time, we can readily obtain $\mathbf{E}\,T_S = 2M[1+O(e^{-c_3N})]$ for a positive constant $c_3$ when $M \leq \frac{(1-c)}{(1+c)}(L-1)$. Bachelis and Massey [3] have done a careful asymptotic analysis of the strategies Hope Springs Eternal and Step in Time for a choice of $M = N/2 + O(\sqrt{N})$.

Similar estimates can also be readily derived for the Ballot Strategy using (16). In particular, the *summands in the sum* $\sum_{m=0}^{M-1} mp_m$ are of the form $\binom{2m}{m-1}2^{-2m}$ which are $\Theta(m^{-1/2})$. If, for instance, $M = N/2$, then $\sum_{m=0}^{M-1} mp_m$ evaluates to $\Theta(\sqrt{N})$. Similarly, the summands in $\beta = \sum_{m=0}^{M-1} p_m$ are of the form $p_m = \frac{1}{2m+1}\binom{2m+1}{m}2^{-(2m+1)} = \Theta(m^{-3/2})$. For $M = N/2$ then, $\beta = 1-\Theta(N^{-1/2})$, so that $\alpha = \Theta(N^{-1/2})$ in this regime. It follows that $\mathbf{E}\,T_B = \Theta(N)$ when $M \leq N/2$. Of course, a more careful analysis of these expressions is needed if exact coefficients are required.

## 4 OPTIMAL STOPPING

The renewal property of our coin tossing game allows us to consider a somewhat simpler optimal stopping problem in order to determine the boundary of an optimal strategy. Recall that our basic problem is to determine a nonrandom Boolean function $f^*$ such that $(f^*)$ is an optimal restart strategy. A heuristic approach towards specifying $f^*$ is as follows: assign a current cost of $n$ if the round is continued after $n$ tosses, and assign a higher restart cost of $d+n$ if there is a restart. The idea then is to choose $f^*$ to minimise the expected cost. (Note that a restart is mandated if $N$ tosses result in fewer than $M$ heads; thus $f^*$ will favour continuing the round for small values of $n$, but will favour restarts when $n$ becomes large and the number of heads is less than $M$.) This related problem is an optimal stopping problem, and as we will see shortly, this will be equivalent to our original coin tossing problem for an appropriate choice of parameter $d$. Bachelis and Massey [3] also consider a similar approach, though the algorithm given here is somewhat more direct, iteratively using the estimates (12) obtained in Theorem 3.3.

Consider a sequence of $n$ tosses of a fair coin stopping on or before the $N$th toss. Let $d > 0$ be some fixed positive real number. Stopping the sequence of tosses at trial $n \in \{0, \ldots, N\}$ results in an allocation of a cost as follows: if the number of heads is less than $M$ the cost is $(d + n)$; if the number of heads is greater than or equal to $M$ the cost is $n$. The optimal stopping problem is to decide on a (randomised) *stopping rule* $f : \{0, \ldots, N\} \times \{0, \ldots, M-1\} \to \{0, 1\}$ which will minimise the cost. More formally, let $f$ denote a stopping rule for the optimal stopping problem, and the random variable $R_f$ denote the cost assigned under $f$. The optimal stopping problem is now to find an optimal stopping rule $f^*$ such that

$$\mathbf{E}\, R_{f^*} = \inf_f \mathbf{E}\, R_f. \tag{20}$$

The technique of backward induction (see, for instance, Chow, Robbins, and Siegmund [8]) can be applied to this optimal stopping problem to generate a nonrandom optimal stopping rule (recall Lemma 2.1). Informally, the procedure asserts that, after the $n$th toss, it is worth continuing the sequence of coin tosses if the conditional expected cost given the results of the first $n$ tosses is smaller than the cost of stopping at the $n$th toss. More precisely, for the optimal stopping rule, we recursively obtain the conditional expected cost $\gamma(n, m)$ corresponding to $n$ tosses with $m$ successes as follows:

$$\begin{array}{ll} \text{BASE:} & \gamma(n, M) = n, \qquad 0 \le n \le N, \\ & \gamma(N, m) = (d + N), \quad 0 \le m \le M - 1, \\ \text{RECURSION:} & \gamma(n, m) = \min\bigl\{(d + n),\ \tfrac{1}{2}[\gamma(n+1, m) + \gamma(n+1, m+1)]\bigr\}. \end{array} \tag{21}$$

The nonrandom optimal stopping rule $f_d^*$ is now determined as follows:

$$f_d^*(n, m) = \begin{cases} 1 & \text{if } \gamma(n, m) = d + n, \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

We define the optimal stopping boundary by

$$F_d^*(n) = \max\{m : \gamma(n, m) = d + n\}. \tag{23}$$

**Lemma 4.1** *The optimal stopping rule $f_d^*$ is consistent for any choice of $d > 0$.*

PROOF: The proof is by backward induction on $n$. The base of the recursion for the expected cost gives

$$\begin{aligned} f_d^*(n, M) &= 0, \qquad 0 \le n \le M, \\ f_d^*(N, m) &= 1, \qquad 0 \le m \le M - 1. \end{aligned}$$

Now by definition, $f_d^*(n, F_d^*(n)) = 1$. We take as inductive hypothesis that $f_d^*(k, l) = 1$ for all $k \ge n$ and $l \le F_d^*(n)$. It is now easy to see from (21) that $f_d^*(n-1, m) = 0$ if $m \ge F_d^*(n) + 1$ and $f_d^*(n-1, m) = 1$ if $m \le F_d^*(n) - 1$. Thus $F_d^*(n-1)$ is either $F_d^*(n)$ or $F_d^*(n) - 1$. In either case it follows that $f_d^*(k, l) = 1$ for all $k \ge n - 1$ and $l \le F_d^*(n-1)$. ∎

The optimal stopping problem described above and the coin tossing game of Section 2 are closely related as shown by the following result.

**Lemma 4.2** *If $d = D(N, M) = \inf_{(f) \in S} \mathbf{E}\, T_f$, then an optimal restart strategy $(f^*)$ for the coin tossing problem (3) determines an optimal stopping rule $f^*$ for the optimal stopping problem (20), and conversely.*

REMARKS: Lemmas 4.1 and 4.2 together complete the proof of Lemma 2.2 so that by Theorem 2.5 it follows that there exists a parsimonious optimal restart strategy. Lemma 4.2 and the proof of Lemma 4.1 hence guarantee that with $d = D(N, M)$, the optimal stopping rule $f_d^*$ determined by (22) is parsimonious.

PROOF: The optimal stopping problem (20) minimises

$$\mathbf{E}\, R_f = \beta_f[d + \mathbf{E}(\tau_f | S_{\tau_f} < M)] + \alpha_f \mathbf{E}(\tau_f | S_{\tau_f} = M) = \beta_f d + \mathbf{E}\, \tau_f .$$

Now select $d = D(N, M)$ and let $(f^*)$ be a stationary, nonrandom optimal strategy for the coin tossing problem (3). It follows from (7) that $D(N, M) = \mathbf{E}\, T_{f^*} = \mathbf{E}(\tau_{f^*})/\alpha_{f^*}$. Let $R^* = \inf_f \mathbf{E}\, R_f$. We claim that $R^* = \mathbf{E}\, R_{f^*} = \mathbf{E}\, T_{f^*}$. Clearly, $R^* \leq \mathbf{E}\, R_{f^*}$. Observe now that with the stopping rule $f^*$, we have

$$\mathbf{E}\, R_{f^*} = \frac{\beta_{f^*}}{\alpha_{f^*}} \mathbf{E}(\tau_{f^*}) + \mathbf{E}(\tau_{f^*}) = \frac{\mathbf{E}(\tau_{f^*})}{\alpha_{f^*}} = \mathbf{E}\, T_{f^*} .$$

Thus, for any $f$,

$$\mathbf{E}\, R_f - \mathbf{E}\, R_{f^*} = \frac{\beta_f}{\alpha_{f^*}} \mathbf{E}(\tau_{f^*}) + \mathbf{E}(\tau_f) - \frac{\mathbf{E}(\tau_{f^*})}{\alpha_{f^*}} = \alpha_f \left[\frac{\mathbf{E}(\tau_f)}{\alpha_f} - \frac{\mathbf{E}(\tau_{f^*})}{\alpha_{f^*}}\right] \geq 0.$$

It follows that $R^* \geq \mathbf{E}\, R_{f^*}$, and hence $R^* = \mathbf{E}\, R_{f^*} = \mathbf{E}\, T_{f^*}$. Thus, if $(f^*)$ is an optimal restart strategy for (3) then $f^*$ is an optimal stopping rule for (20).

To complete the proof, suppose that with $d = D(N, M) = \mathbf{E}\, T_{f^*}$, $\hat{f}$ is an optimal stopping rule for (20): $R^* = \mathbf{E}\, R_{\hat{f}}$. We need to show that $(\hat{f})$ is also an optimal restart strategy for (3). By (7) it suffices to show that $\mathbf{E}(\tau_{\hat{f}})/\alpha_{\hat{f}} = \mathbf{E}(\tau_{f^*})/\alpha_{f^*}$. Now by assumption of optimality of the stopping rule $\hat{f}$ it follows that

$$R^* = \mathbf{E}\, R_{\hat{f}} = \frac{\beta_{\hat{f}}}{\alpha_{f^*}} \mathbf{E}(\tau_{f^*}) + \mathbf{E}(\tau_{\hat{f}}).$$

But $R^* = \mathbf{E}\, T_{f^*} = \mathbf{E}(\tau_{f^*})/\alpha_{f^*}$. It follows that $\mathbf{E}(\tau_{\hat{f}}) = \frac{\alpha_{\hat{f}}}{\alpha_{f^*}} \mathbf{E}(\tau_{f^*})$. ∎

The backward induction (21)–(23) can now be used to iteratively compute the boundary of a parsimonious optimal restart strategy $(f^*)$. The approach followed here can also be derived from a nested Markov decision problem approach (cf. Ross [4]).

**Algorithm I** (*Backward Induction*) Given a number of coins $N$, and the desired number of successes $M$, this algorithm obtains the boundary of an optimal restart strategy.

**I1.** [Initial approximation.] Let $(f_0)$ be any initial strategy, and set $d_0 = \mathbf{E}\, T_{f_0}$. Set $j \leftarrow 0$.

**I2.** [Optimal stopping.] Set $d \leftarrow d_j$ and solve the associated optimal stopping problem using the backward induction (21)–(23) to obtain the optimal stopping boundary $F_d^*$ (corresponding to the stopping rule $f_d^*$).

**I3.** [Check for convergence.] If $n_0 = \min\{n : f_d^*(n, 0) = 1\} = 0$, then output the (optimal) boundary $F_{d_{j-1}}^*$ and stop.

**I4.** [Estimate expected duration.] Set $j \leftarrow j + 1$. Using (9)–(12) set $d_j \leftarrow \mathbf{E}\, T_{f_d^*}(N, M)$.

**I5.** [Iteration.] If $d_j \neq d_{j-1}$, go back to step I2; otherwise, if $d_j = d_{j-1}$, output the (optimal) boundary $F_{d_j}^*$ and terminate the algorithm. ∎

REMARKS: At epoch $j$, the cost $d_j$ for the optimal stopping problem is exactly the expected duration corresponding to the optimal boundary $F^*_{d_{j-1}}$ obtained for the cost $d_{j-1}$. If now we obtain $n_0 = 0$ for the new optimal boundary $F^*_{d_j}$, then this implies the game cannot be started, which in turn implies a cost of $d_j$ for the optimal stopping problem. This, however, is the cost associated with the previous optimal boundary $F^*_{d_{j-1}}$. Thus, encountering $n_0 = 0$ at epoch $j$ in the progress of the algorithm implies that the expected risk $d_j$ is a fixed point of the algorithm, and this expected risk is exactly the expected duration of the coin tossing game for a choice of boundary $F^*_{d_{j-1}}$.

Any convenient value for $d_0$ can be chosen, though the Ballot Strategy yields a particularly good starting point as we will see in the numerical simulations of Section 5.

**Theorem 4.3** *Algorithm I converges to a parsimonious optimal restart strategy* $(f^*)$ *for the coin tossing problem (3).*

PROOF: We first show that the algorithm converges. By step I3 of the algorithm, this clearly happens if $n_0 = 0$ at any point. Assume now that $n_0 \neq 0$ at any point in the progress cf the algorithm. Let $d_j$ denote the value of $d$ at the $j$th iteration (see step I2 of Algorithm I), and let $f^*_{d_j}$ be the solution of the optimal stopping problem with $d = d_j$. Then $d_j = \mathbf{E}(\tau_{f^*_{d_{j-1}}})/\alpha_{f^*_{d_{j-1}}}$. We hence have

$$d_j = \beta_{f^*_{d_{j-1}}} d_j + \mathbf{E}(\tau_{f^*_{d_{j-1}}}) \geq \inf_f [\beta_f d_j + \mathbf{E}(\tau_f)] = \beta_{f^*_{d_j}} d_j + \mathbf{E}(\tau_{f^*_{d_j}}).$$

It follows that $d_j \geq d_{j+1} = \mathbf{E}(\tau_{f^*_{d_j}})/\alpha_{f^*_{d_j}}$. Thus, the sequence $\{d_j\}$ is decreasing, and as there are only finitely many possible values for $d$, the algorithm converges.

Now let $f^*$ be any fixed point of the algorithm. Applying (6) inductively we obtain that for any $r$, and any choices of restart functions $f_i$, $1 \leq i \leq r$, $(f^*) \leq (f_1, \ldots, f_r, f^*, f^*, \ldots)$. Allowing $r \to \infty$ we see that $(f^*)$ is an optimal strategy for the game (3). ∎

# 5 NUMERICAL SIMULATIONS

The Ballot Strategy was observed to have performances comparable to the optimal strategy on simulations over several values of $n$. In Fig. 6 we contrast the expected duration of the game for the Indolent, Hope Springs Eternal, and Ballot Strategies with the minimum expected duration of the game for an optimal strategy generated by Algorithm I using the Indolent Strategy as an initial strategy. (Note that rather large absolute differences in expected duration across the strategies are hidden because of the logarithmic scale of the plots.) Note, that as per parts (a) and (b) of Theorem 3.3, all the strategies are essentially equivalent when $M \leq N/2$, and that it is only in the regime $M > N/2$ of expected exponential duration that it pays to look for an optimal strategy.

Numerically, the strategy Step in Time was found to give results comparable to Hope Springs Eternal for large values of $L$, and substantially poorer results for small values of $L$. We did not attempt to optimise the value of $L$ in light of the performance of the Ballot Strategy. As an aside, the probabilities $p_m$ given by recursion (10) need to be evaluated with some care (especially for large values of $M$) as the backward induction can be numerically sensitive.

In Fig. 7 we have shown plotted the restart boundary of the Ballot Strategy compared with that of an optimal strategy. Note the close correspondence of the boundaries of the two strategies. It would appear that the Ballot Strategy is slightly more conservative in setting the restart boundary than an optimal strategy.

# 6 EXTENSIONS

The results of this paper can be easily generalised to the case where biased coins are tossed. Let $\mathbf{P}\{X_i = 1\} = \eta$, $\mathbf{P}\{X_i = 0\} = 1 - \eta = \nu$. The following generalisation of Lemma 3.2 follows easily:

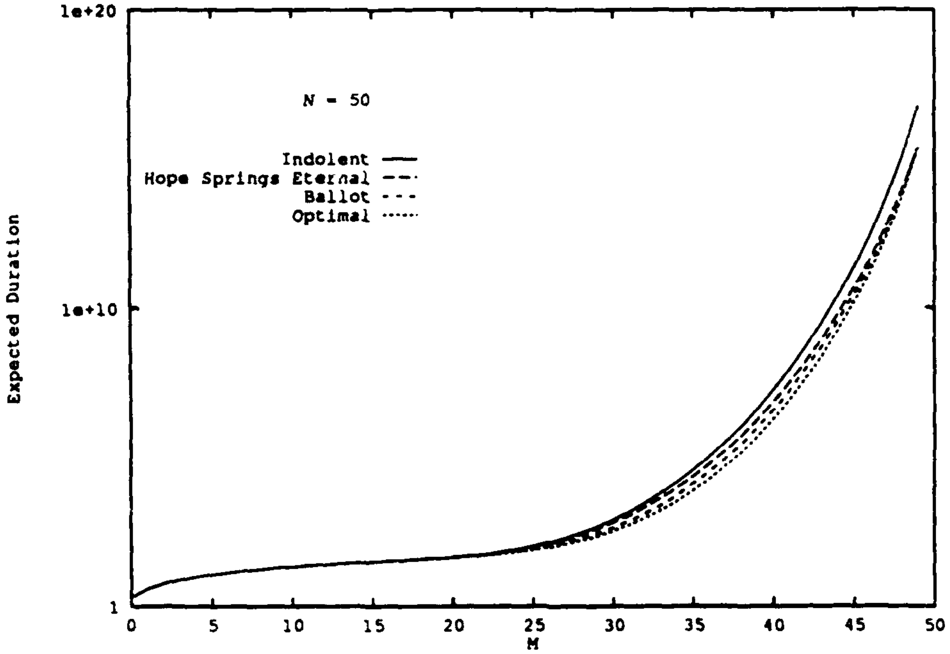


Figure 6: The expected duration of the coin tossing game is plotted (on a logarithmic scale) versus the desired number of successes $M$ for $N = 50$ for the Indolent, Hope Springs Eternal, Ballot, and Optimal Strategies.

**Lemma 6.1** *The probabilities $p_m$, $0 \leq m \leq M-1$ satisfy the following recursion:*

$$\text{BASE:} \qquad p_0 = \nu^{n_0},$$

$$\text{RECURSION:} \quad p_m = \binom{n_m}{m}\eta^m\nu^{n_m-m} - \sum_{i=0}^{m-1}\binom{n_m-n_i}{m-i}\eta^{m-i}\nu^{n_m-n_i-m+i}\,p_i, \qquad m \geq 1.$$

A straightforward generalisation of Theorem 3.3 now yields:

**Theorem 6.2** *Assume $\eta > 0$. Then, for any consistent strategy $f$,*

$$\mathbf{E}\,T_f(N,M) = \frac{M}{\eta} + \frac{\sum_{m=0}^{M-1} m p_m}{\eta\left(1 - \sum_{m=0}^{M-1} p_m\right)}.$$

It is also easily verified that Theorem 3.5 holds unchanged. A more general form of Theorem 3.6 is now readily obtained.

**Theorem 6.3** *Let $c > 0$ be any fixed parameter and $N$ any positive integer. Then:*

*(a) If $M \leq (1-c)\eta N$, then $\frac{M}{\eta} \leq D(N,M) < \frac{M}{\eta}\left[1 + \frac{e^{-2c^2\eta^2N}}{1-e^{-2c^2\eta^2N}}\right]$;*

*(b) If $M \geq (1+c)\eta N$, then $D(N,M) > \frac{1}{N}e^{2c^2\eta^2N}$.*

Note that we can again use Chernoff's bounds to obtain tighter results.

Corresponding to the change in the recursion for $p_m$, the backward induction (21) has a corresponding change with the recursion replaced by

$$\gamma(n,m) = \min\{(d+n),\ \eta\gamma(n+1,m+1) + \nu\gamma(n+1,m)\}.$$

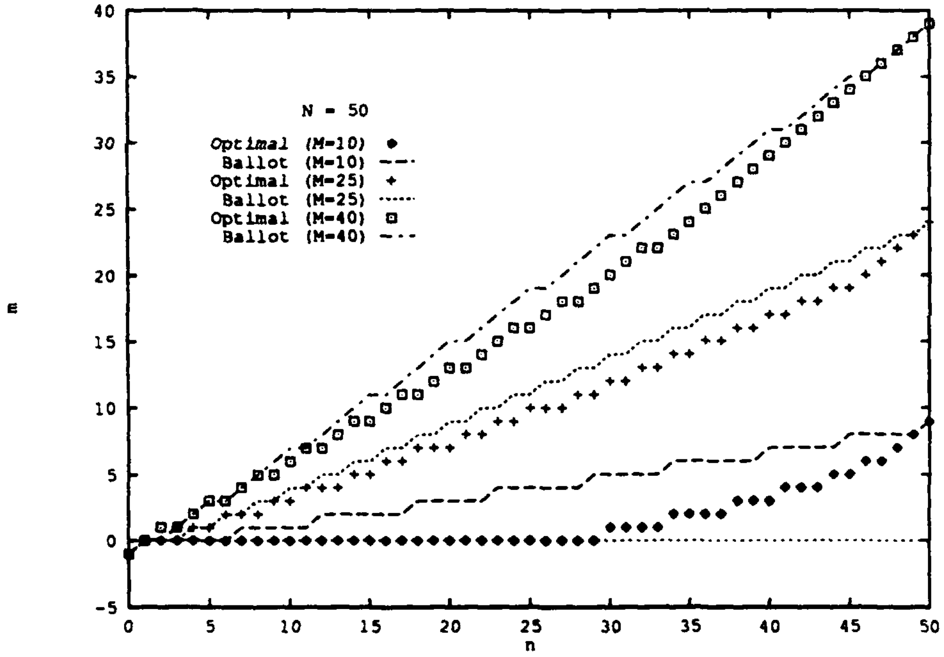


Figure 7: The boundaries of the Ballot Strategy and an optimal strategy for $N = 50$ and choices of $M = 10$, 25, and 40.

Algorithm I continues to work as before.

The problem in this general form with biased coins would seem to have more direct relevance to early abort strategies in randomised algorithms for factoring integers. For instance, going back to the discussion in the Introduction, consider the problem of finding a factor of a (large) integer $n$. Assume that we have selected an integer $m$ of the order of $\ln n$ for Dixon's algorithm. As we saw, a rough approximation to the problem in terms of the coin tossing game is to consider a store of $N = \pi(m)$ unfair coins with identical probabilities $1/m$ of a toss resulting in a head and require to find an optimal strategy which minimises the total number of tosses before achieving $M = \ln n/\ln m$ heads. The classical estimate for the number of primes less than $m$,

$$\pi(m) = \frac{m}{\ln m} + O\left(\frac{m}{(\ln m)^2}\right) \qquad (m \to \infty),$$

shows that we are in the exponential domain for $D(N, M)$, and a ready calculation using Theorem 6.3(b) yields

$$D(N, M) = \Omega\left(\frac{\ln \ln n}{\ln n}\, e^{\kappa \ln n/\ln \ln n}\right)$$

for a positive constant $\kappa$. (Or make the bound exponentially tight by using the sharper binomial tail bounds.) This would suggest the rough estimate $n^{O(1/\ln \ln n)}$ for the minimum (over all early abort strategies) expected number of steps in Dixon's algorithm before one solution to (2) is obtained.

## Acknowledgement

One of the authors (SSV) thanks Ron Rivest for passing the time in Denver Stapleton Airport by airing several open problems, of which the problem addressed in this paper was one, and for subsequently communicating the results of Bachelis and Massey to us. Our thanks also go to Shao Fang who wrote the programs for the simulations in Section 5. We also wish to acknowledge a debt of gratitude to the two anonymous referees whose detailed suggestions have done much to improve the presentation of the results of this paper.

## References

[1] Euclides, *Elements.*

[2] C. Pomerance, "Analysis and comparison of some integer factoring algorithms," in *Computational Methods in Number Theory,* (eds. H. W. Lenstra, Jr. and R. Tijdeman), Mathematical Centre Tracts 154, Mathematisch Centrum, Amsterdam, 1982.

[3] G. F. Bachelis and F. J. Massey III, "A coin tossing problem of R. L. Rivest," preprint. [Communicated to us by R. L. Rivest.]

[4] S. Ross, *Introduction to Stochastic Dynamic Programming.* New York: Academic Press, 1983.

[5] W. Feller, *An Introduction to Probability Theory and its Applications*, vol II. New York: Wiley, 1971.

[6] I. Hu and S. S. Venkatesh, "A combinatorial identity from an urn model for ladder variables," *J. Comb. Theory, Series A*, submitted for publication.

[7] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Stat. Assoc.*, vol. 58, pp. 13–30, 1963.

[8] Y. S. Chow, H. Robbins, and D. Siegmund, *Great Expectations: The Theory of Optimal Stopping.* Boston: Houghton-Mifflin, 1972.