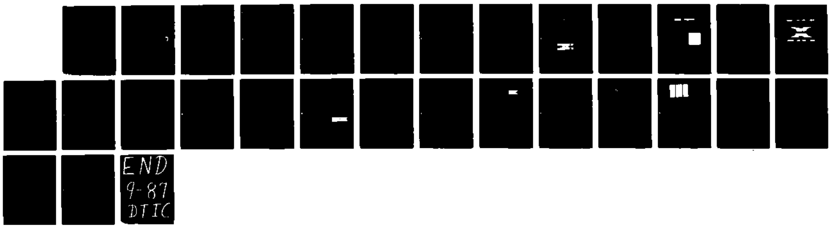
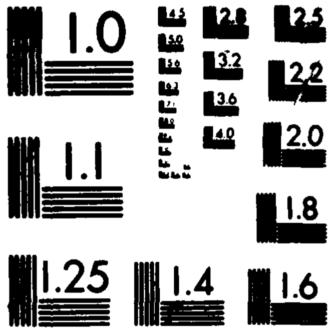
AD-A183 632 DIMENSIONALITY-REDUCTION USING CONNECTIONIST NETWORKS (U) MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB E SAUND JAN 87 AI-M-941 1/1

UNCLASSIFIED N00014-85-K-0124 F/G 12/9 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DTIC FILE COPY (1)

AD-A183 632

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AIM-941	2. GOVT ACCESSION NO	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Dimensionality-Reduction Using Connectionist Networks		5. TYPE OF REPORT & PERIOD COVERED AI-Memo
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Eric Saund		8. CONTRACT OR GRANT NUMBER(s) N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE January, 1987
		13. NUMBER OF PAGES 26
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

DTIC ELECTE AUG 24 1987 D

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

S E

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

dimensionality-reduction, connectionist network, pattern matching, backpropagation, data abstraction, self-organization

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A method is presented for using connectionist networks of simple computing elements to discover a particular type of constraint in multidimensional data. Suppose that some data source provides samples consisting of n-dimensional feature-vectors, but that this data all happens to lie on an m-dimensional surface embedded in the n-dimensional feature space. Then occurrences of data can be more concisely described

(continued on reverse)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (cond.)

by specifying an m-dimensional location on the embedded surface than by reciting all n components of the feature vector. The recoding of data in such a way is known as dimensionality-reduction. This paper describes a method for performing dimensionality-reduction in a wide class of situations for which an assumption of linearity need not be made about the underlying constraint surface. The method takes advantage of self-organizing properties of connectionist networks of simple computing elements. We present a scheme for representing the values of continuous (scalar) variables in subsets of units. The backpropagation weight updating method for training connectionist networks is extended by the use of auxiliary pressure in order to coax hidden units into the prescribed representation for scalar-valued variables.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A. I. Memo 941 January, 1987

Dimensionality-Reduction Using Connectionist Networks

Eric Saund

Abstract

A method is presented for using connectionist networks of simple computing elements to discover a particular type of constraint in multidimensional data. Suppose that some data source provides samples consisting of n-dimensional feature-vectors, but that this data all happens to lie on an m-dimensional surface embedded in the n-dimensional feature space. Then occurrences of data can be more concisely described by specifying an m-dimensional location on the embedded surface than by reciting all n components of the feature vector. The recoding of data in such a way is known as *dimensionality-reduction*. This paper describes a method for performing dimensionality-reduction in a wide class of situations for which an assumption of linearity need not be made about the underlying constraint surface. The method takes advantage of self-organizing properties of connectionist networks of simple computing elements. We present a scheme for representing the values of continuous (scalar) variables in subsets of units. The backpropagation weight updating method for training connectionist networks is extended by the use of auxiliary pressure in order to coax hidden units into the prescribed representation for scalar-valued variables.

This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's Artificial Intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124. The author is supported by a fellowship from the NASA Graduate Student Researchers Program.

1 Introduction

A common objective in Pattern Recognition, Artificial Intelligence, and Machine Vision is to discover good representations in which to describe data. An important step in generating a good representation is to expose constraint and remove redundancy in the description of sensory data. One particular form of constraint that can occur over a collection of data is illustrated in figure 1. Here, some data source generates (x, y) data points. Evidently, the source operates under some constraint because all data points appear to lie on a one-dimensional curve. If one possesses knowledge of this curve then one may describe any given data sample using only a single number, the location along the curve, instead of the two numbers required to spell out the (x, y) coordinates.

This form of data recoding is known as *dimensionality-reduction* [Krishnaiah and Kanal, 1982; Kohonen, 1984]. In general, the problem is to define a computational mapping between locations in an n-dimensional feature space, and locations on an m-dimensional *constraint surface* embedded in the n-dimensional feature space, given a set of samples drawn from the constraint surface. The purpose served by dimensionality-reduction

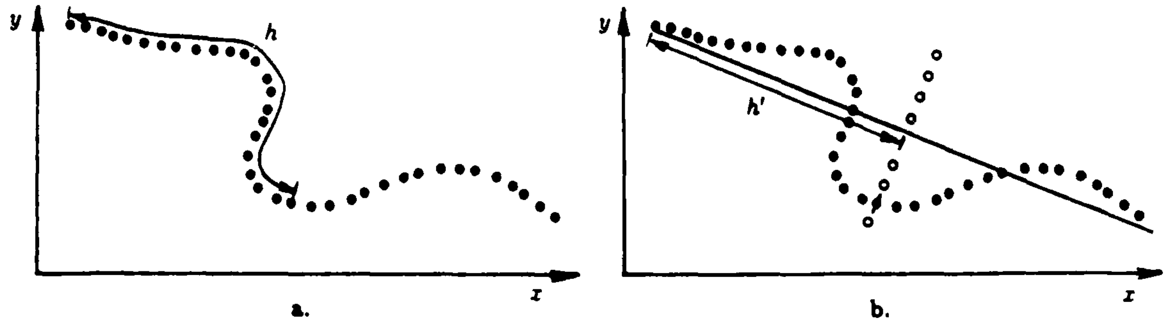


Figure 1. Two-dimensional data samples constrained to lie on a one-dimensional constraint surface. a. Through dimensionality-reduction, the parameter, h, describes data in terms of location on the constraint surface. b. Data on (closed circles) and off (open circles) the constraint surface appears identical under dimensionality-reduction by pure linear transformation.

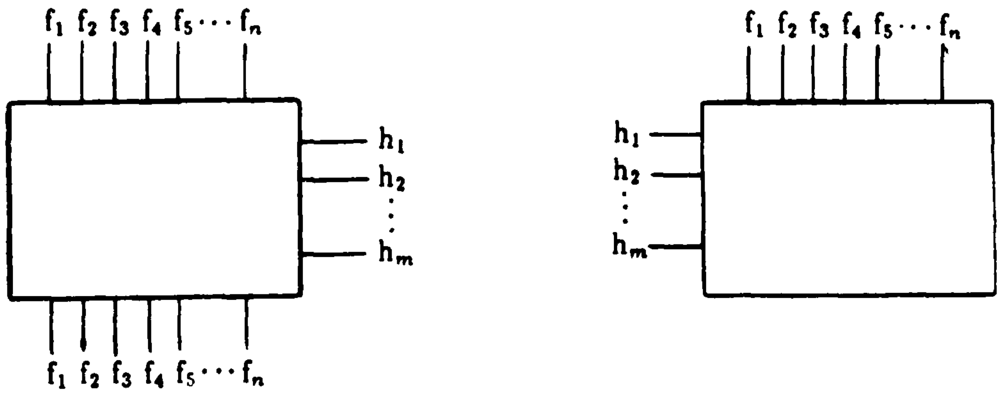
is abstraction and simplification in the description of data. In general, a dimensionality-reduced representation can be expected to make explicit descriptive parameters capturing the natural degrees of variability inherent to a class of data, while it factors out redundancy latent among the original measured features.

Most previous approaches to dimensionality-reduction in pattern recognition assume a linear constraint surface. Then, the techniques of factor analysis, principle components analysis, or equivalently, the Karhunen-Loève expansion [Watanabe, 1965; Tou and Heydorn, 1967; Fukunaga and Koontz, 1970; Kittler and Young, 1973], amount to first, rotating the n-dimensional coordinate system to align with the data, and second, picking out the few dimensions that best account for the variance. Linear dimensionality-reduction methods buy simplicity at a sacrifice of generality. Clearly, they would not work for the constraint surface pictured in figure 1 because this constraint surface doubles back on itself in the embedding space. Furthermore, dimensionality-reduction based on linear approximation to a constraint surface cannot accurately determine whether or not a given, unknown, data sample does or does not lie on the constraint surface, as shown in figure 1b.

This paper presents a more general method for achieving dimensionality-reduction which allows the underlying constraint surface to curve to a considerable degree. The method relies on the self-organizing properties of connectionist networks of simple computing elements [Rumelhart, McClelland, et al, 1986]. The technique differs from that of Kohonen [1984], with which it is compared in the discussion section. An earlier version of this work appears in [Saund, 1986].

2 Black-Box Dimensionality-Reducer

The "black box" depiction of a dimensionality-reducer is presented in figure 2. Each such box contains knowledge of one constraint surface in the feature space. At the bottom of the box enters the description of a data point in terms of an n-dimensional feature vector. Out the top emerge n lines, and out the side, m more, where m is the dimensionality of the constraint surface. Each line can represent a bounded real number; for convenience suppose that the feature coordinates are normalized so that all features



$f_1\ f_2\ f_3\ f_4\ f_5 \cdots f_n$
h_1
h_2
h_m
$f_1\ f_2\ f_3\ f_4\ f_5 \cdots f_n$
$f_1\ f_2\ f_3\ f_4\ f_5 \cdots f_n$
h_1
h_2
h_m

Figure 2. Black-box dimensionality-reducer.

take values between 0 and 1.

The dimensionality-reducer operates as follows. If the numbers coming out the top of the black box match those coming in the bottom, then it is determined that the data point whose feature vector is given does in fact lie on the constraint surface, and its location on the constraint surface may be read on the m lines coming out the side (the dimensionality-reducer implicitly creates a coordinate system for the constraint surface). If the numbers coming out the top do not match the input feature vector, then it is determined that the data point specified at the input does not lie on the constraint surface. In this way a dimensionality-reducer carries out a form of pattern matching between input data and a parameterized "template" defined by the locus of points on the constraint surface.

This black-box dimensionality-reducer may also be used in the opposite direction. That is, an m-dimensional vector specifying a location on the constraint surface may be placed on the side lines as input, and the dimensionality-reducer will then compute, and output at the top, the coordinates of this data point in the n-dimensional feature space.

The contents of the black box dimensionality-reducer described in this paper are a network of simple computing elements.

3 Connectionist Networks

Recently, renewed attention has been directed to "connectionist" networks of simple computing elements. It is believed that new developments in training procedures applied to multilayer networks will overcome some of the well-known objections to classical perceptrons [Rosenblatt, 1962; Minsky and Papert, 1969]. Of interest here are demonstrations by Hinton, et al [1984], and Rumelhart, et al [1985, 1986], that hidden units in multilayer networks are able to attain abstract representations capturing constraint in binary input data.

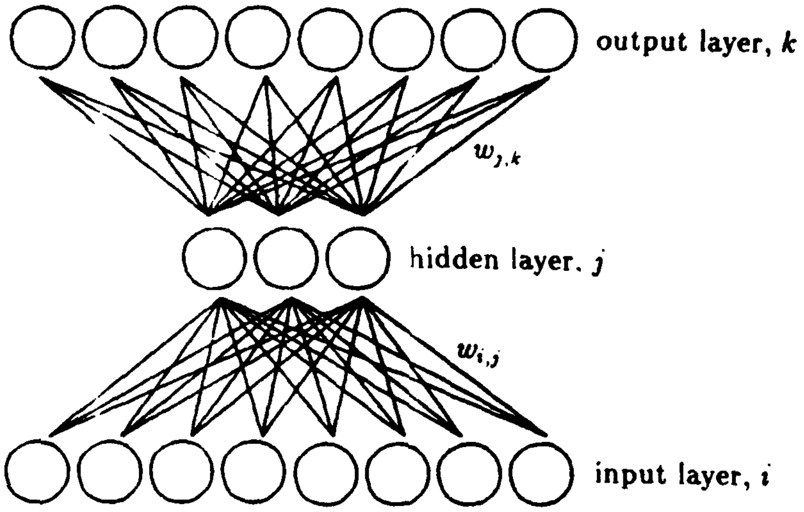
A prototypical example is the "encoder problem" (see figure 3). Here, the activity, a_k, in a layer of eight output units is calculated from the activity, a_j in the middle layer of three units:

$$a_k = f(s_k) = f(\sum_j w_{jk}a_j + b_k), \tag{1}$$

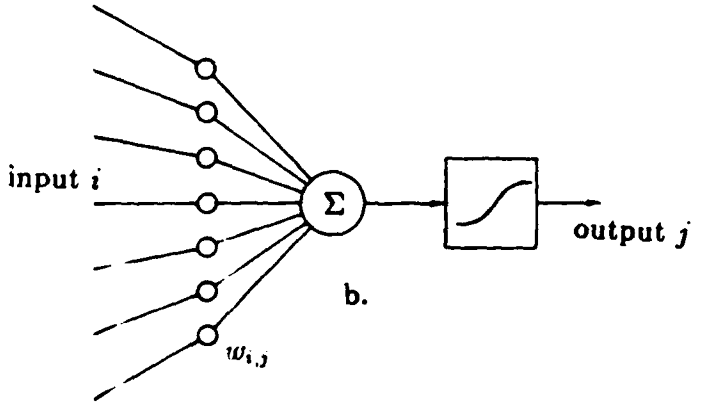
where w_{jk} is the connection weight between the jth middle layer unit and the kth output layer unit. b_k is a bias input; it is omitted from succeeding equations because it can be considered equivalent to the weight, w, from a unit whose output is always 1. The activity in the middle layer is calculated from the activity in the input layer in a similar way. The middle layer units are called "hidden" units because their activity is not directly accessible either at network input or output. Each unit's activity is bounded between 0 and 1 by f, which is typically a sigmoidal nonlinear function, for example,

$$f(x) = \frac{1}{1+e^{-x}}. \tag{2}$$

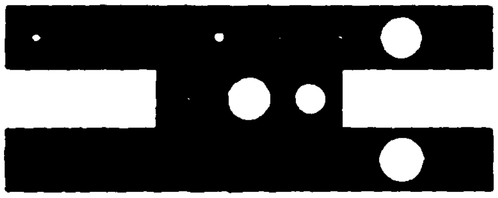
The goal of the encoder problem is to set the weights such that if any single unit in the input layer is set ON (near 1), and the rest set OFF (near 0), the activity will propagate so that only the corresponding output unit will be ON. Because data presented to the input layer is constrained, so that only a subset of all possible patterns of activity in the input layer ever actually appear, the information as to which unit is ON may be propagated to the output layer through a middle layer containing *fewer* than eight units. In particular, the three middle layer units may transmit the information by adopting a binary code. Rumelhart, Hinton, and their colleagues have described a method, called *backpropagation,* for training a



a.



b.



c.

Figure 3. a. Three layer connectionist network. Activity at the input layer drives hidden layer, activity at hidden layer drives output layer. b. Activity at a unit is computed as a semilinear function of the weighted sum of the unit's inputs. c. Activity in an 8-3-8 network trained for the encoder problem. Input is constrained so that only one input unit is ON at a time. Activity at output matches input. The information as to which input is ON is able to be transmitted via the hidden unit layer of only three units. Size of circle represents unit's activity.

network to achieve such behavior without directly specifying the behavior of the hidden layer: repeatedly present the network with sample input and allow activity to propagate to the output layer, observe the error between the desired output and the observed activity at the output layer, and use these errors to update the network weights. This technique is described in more detail below.

Here we describe a method for extending the backpropagation weight updating scheme for connectionist networks in order to perform dimensionality reduction over continuous-valued data.

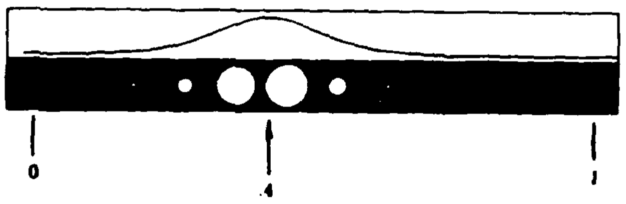
4 Representing Scalar Variables

We must start by endowing networks with the ability to represent not just the binary variables, ON and OFF, but variables continuous over some interval. For convenience let this interval be (0,1).

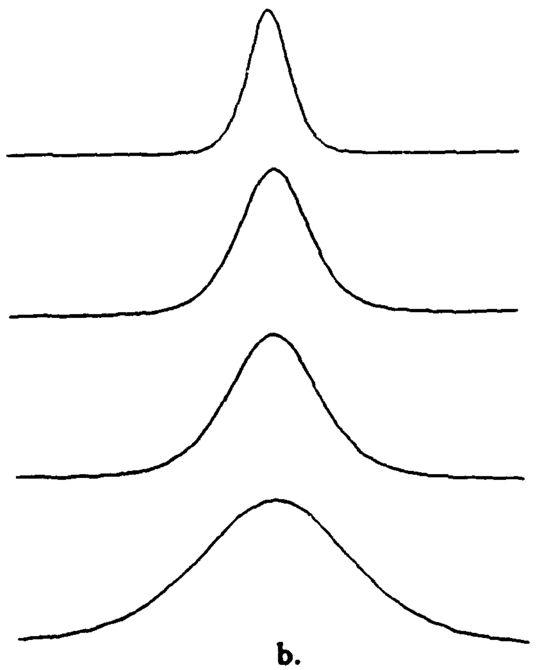
One conceivable way of representing scalars in simple units is via a unit's activity level itself. Since only one weight connects any middle-layer unit to any given output unit, this strategy is clearly inadequate for representing anything but linear relationships between variables. The relationship between x and y in figure 1 is not linear, so the relationship between x and some hidden variable, h, and between y and h must not both be linear. Therefore, we instead use the following scheme.

Each scalar feature value is represented as the pattern of activity over a set of units, called a *scalar-set*, as illustrated in figure 4. The pattern of activity is determined by sampling a unimodal *smearing function*, S_ω, centered at the location along the $(0,1)$ interval corresponding to the intended scalar value. The exact form of the smearing function is not important; here, it happens to be the derivative of the function, f, of equation (2), but other forms, such as the gaussian, may be used. The parameter, ω, controls the width of the smearing effect. This method for encoding scalar values achieves a form of interpolation. Resolution of better than 1 part in 50 may be achieved using as few as eight units.

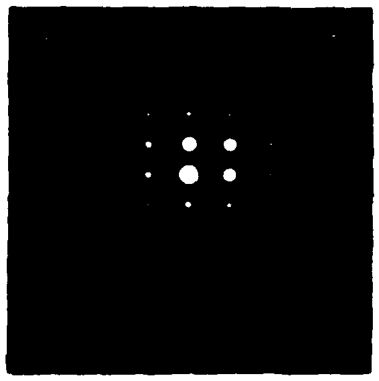
Any set of units whose activity pattern displays the characteristic unimodal form for encoding scalar values is said to exhibit "scalarized" behavior. The number expressed in a scalarized pattern of activity may be decoded as the placement of the smearing function, S, at the location, x,



a.



b.



c.

Figure 4. a Scalar values between 0 and 1 are represented in *scalar-sets* of units whose activity takes a characteristic pattern defined by sampling a unimodal *smearing function*, S_ω. The activity pattern shown in this 12-unit scalar-set represents the number, .4. b. S_ω for several values of the smearing width parameter, ω. c. Scalarized activity in a two-dimensional scalar-set.

within the $(0, 1)$ interval, that minimizes the least-square difference,

$$Q(x) = \sum_i (S_{x-i} - a_i)^2. \tag{3}$$

Two means are available for representing multidimensional vectors in terms of scalar-sets. The most straightforward method is to provide one one-dimensional scalar-set for each scalar component of the vector. However in order to perform dimensionality-reduction it is also necessary to employ an alternative representation in which the scalar-set itself takes the inherent dimensionality of the vector. For example, figure 4c shows a scalarized pattern of activity over a two-dimensional scalar-set.

5 Training the Network

A three-layer connectionist network configured to perform dimensionality-reduction is illustrated in figure 5. A one-dimensional scalar-set is provided at the input layer and at the output layer for each coordinate dimension of the higher-dimensional feature space. The hidden layer is configured as a scalar-set whose dimensionality matches that of the embedded constraint surface. The input, hidden, and output layers of the network correspond to the bottom, side, and top of the black box dimensionality-reducer.

Dimensionality-reducing behavior is achieved by virtue of the link weights between successive layers of the network. These weights are established using the backpropagation training procedure, which furnishes crucial self-organizing properties during the training phase. Training consists of repeated presentation of input activity/desired output activity pairs, where the desired output is defined to be identical to input activity. At each training trial, activity at the input layer is fixed according to the scalarized representation for the coordinates of a data sample expressed in terms of the high-dimensional feature space. Activity propagates through the network, and the output layer activity is compared with that of the input to create an output error vector. This error is used to update link weights within the network in such a way as to reduce the output error.

The backpropagation method for updating weights in order to achieve some desired input/output behavior may be derived by expressing the relationship between a.) a cost for error in output behavior, and b.) the

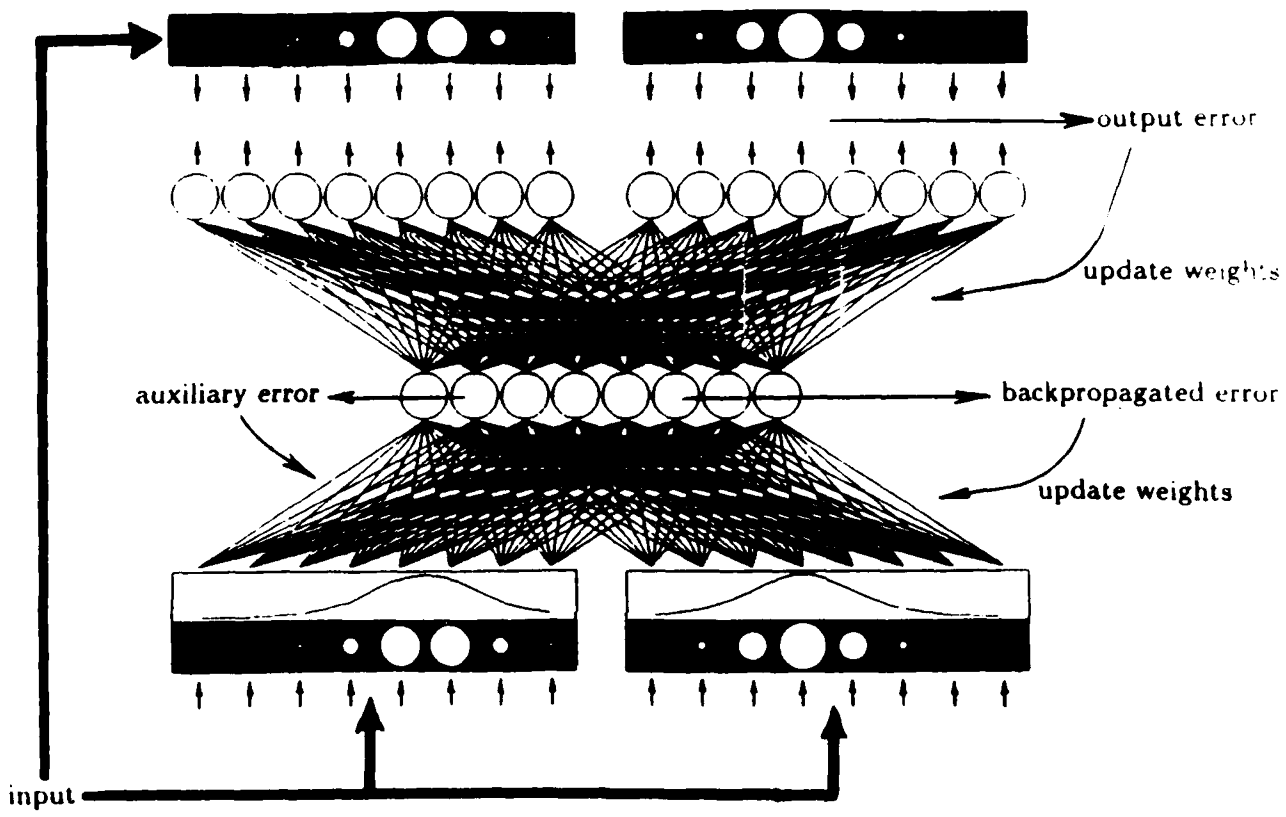


Figure 5. Connectionist dimensionality-reducer. In this case, two scalar-sets are provided at the input and output layers, and the hidden layer is a one-dimensional scalar set, therefore, this network can represent a one-dimensional constraint curve in a two-dimensional feature-space. During training period, errors from desired activity are used to train network to reproduce input activity pattern at output. Auxiliary error pressures hidden layer units into adopting a scalarized representation.

strengths of individual weights in the network. Following Rumelhart, et al [1985], define cost

$$E = \frac{1}{2}\sum_p E_p = \frac{1}{2}\sum_p\sum_k(a_{pk} - t_{pk})^2 = \frac{1}{2}\sum_p\sum_k \delta_{pk}^2 \tag{4}$$

as the cost over all output units, k, of error between output a_k and target output, t_k, summed over all sets of presentations, p, of input data. Weights are adjusted a slight amount at each presentation, p, so as to attempt to reduce E_p. The amount to adjust each weight connecting a middle layer unit and an output unit is proportional to (from (1) and (4))

$$\frac{\partial E_p}{\partial w_{jk}} = \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial s_k}\frac{\partial s_k}{\partial w_{jk}} = \delta_{pk} f'(s_k) a_j. \tag{5}$$

Take

$$\Delta w_{jk} = -\eta \delta_{pk} f'(s_k) a_j \tag{6}$$

as the amount to adjust weight w_{jk} at presentation p. η is a parameter controlling learning rate.

Adjustments of weights between the input and middle layers is proportional to

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial a_j}\frac{\partial a_j}{\partial s_j}\frac{\partial s_j}{\partial w_{ij}} \tag{7}$$

$$= \left(\sum_k \frac{\partial E_p}{\partial a_k}\frac{\partial a_k}{\partial s_k}\frac{\partial s_k}{\partial w_{jk}}\right)\frac{\partial a_j}{\partial s_j}\frac{\partial s_j}{\partial w_{ij}} \tag{8}$$

$$= \sum_k \left(\delta_{pk} f'(s_k) w_{jk}\right) f'(s_j) a_i. \tag{9}$$

Defining

$$\delta_{pj} = \sum_k \delta_{pk} f'(s_k) w_{jk}, \tag{10}$$

we arrive at a recursive formula for propagating error in output back through the network. Take

$$\Delta w_{ij} = -\eta \delta_{pj} f'(s_j) a_i. \tag{11}$$

Essentially, this method for updating weights performs a localized gradient descent in error cost over weight space. By asserting an equivalent hidden layer error vector, δ_{pj}, the backpropagation step amounts to analyzing how each unit at the hidden layer contributes to error observed at the output layer.

In implementation, a "momentum" factor is used to smooth the trajectory in weight space. The actual weight adjustment used at iteration t is given by

$$\Delta w^{update}(t) = \Delta w(t) + \alpha \Delta w(t-1), \tag{12}$$

where α is a parameter controlling the rate of decay of the contribution to Δw^{update} from previous data samples.

6 Auxiliary Error to Constrain Hidden Unit Activity

If all of the data presentations to scalar-sets at the input layer conform to scalarized representations for the scalar components of the data vector, then after a suitable training period the output will come to faithfully mimic the input. Unfortunately, there is no guarantee that the hidden units will adopt the scalarized representation in their transmission of activity from the input layer to the output layer. In general their coding behavior will be unpredictable, depending upon the initial randomized state of the network and the order of data sample presentation.

What is needed is a way to force the scalar-set at the hidden layer into adopting the prescribed scalarized pattern of activity. For this purpose we introduce an auxiliary error term, γ_j, to be added to the error in activity at the hidden layer, δ_j, which is propagated back from the error in activity at the output layer (10). The network weights connecting the input layer and the middle layer are now updated according to

$$\Delta w_{ij} = [\lambda \delta_j + (1-\lambda)\gamma_j] f'(s_j) a_i, \tag{13}$$

where λ $(0 < \lambda < 1)$ trades off the relative contributions of δ and γ. γ must be of a suitable nature to pressure the hidden units into becoming scalarized as the training proceeds. We compute γ_j for the units of the hidden layer scalar-set as follows.

We may view the activity over the set of units in a scalar-set as the transformation, by the smearing function, S, of some underlying "likelihood" distribution, $p(j)$, over values in the interval, $0 < j < 1$. The activity in a scalar-set is the convolution of the likelihood distribution with the smearing function, sampled at every unit. Scalarized activity occurs when the underlying distribution is the Dirac Delta function. The strategy we suggest for adding auxiliary pressure to the scalar-set activity is simply to encourage scalarized behavior: add some factor to sharpen the peaks and lower the valleys of the likelihood distribution, to make it more like the Delta function. A convenient way of doing this is to raise the underlying distribution to some positive power, and normalize so that the total area is unity. In the general case, if this procedure were repeatedly applied to some distribution, one peak would eventually win out over all others. The procedure is summarized by the following equation:

$$\gamma(j) = \left(S * N\left\{\left[S^{-1} * a(j)\right]^{q}\right\}\right) - a(j). \tag{14}$$

The activity in the scalar-set, $a(j)$, is deconvolved with the smoothing function, S, to reveal the underlying likelihood distribution. This is raised to the power, $q > 0$, and then normalized (by N). This new underlying likelihood is now convolved with the smoothing function, S, and γ is taken as the error between this result and the current activity in the scalar-set.

Now, on every training trial the weight updating term, δ_j, pressures hidden units to adopt activities that will reduce the error between input layer activity and output layer activity, while the auxiliary error term, γ_j, pressures hidden units to adopt scalarized activity.

In an actual implementation, $a(j)$ is not a continuous function, but rather consists of the activity in the finite, usually small, number of units in the scalar-set. Therefore the bandwidth available for conveying the underlying likelihood, $p(j)$, is small; sharp peaks in $p(j)$ are not representable because high spatial frequency information cannot be carried in a. An alternative expression for γ, to substitute for (14), has been found acceptable in practice:

$$\gamma_j = N\left[S * a(j)^2\right] - a(j) \tag{15}$$

Thus, we square the activity in each unit, convolve this squared activity in the scalar-set with the smearing function, S, then normalize so that the total activity in the scalar-set sums to a constant. This procedure for generating γ_j approximates the effect of encouraging scalarized patterns of activity in the scalar-set.

7 Sculpting the Cost Landscape

As noted above, the network training procedure carries out gradient descent. Weights are updated at each training presentation so as to reduce the cost, E. This cost is high when activity in the output layer differs from activity in the input layer, and, due to the auxiliary error term, γ, the cost is high when activity in the hidden layer scalar-set does not conform to the characteristic scalarized representation for scalar values. If, as is usually the case, no prior knowledge of constraint operating upon the data source is available, the network is initialized with random values for all weights, and E will be large at the outset.

Simple gradient descent methods commonly suffer from the problem that there may exist local minima in the cost landscape that are far from the global minimum. Once a network falls into a local minimum there is no escape.

The local minimum phenomenon has been reported by Rumelhart, et al [1985], in normal binary-variable connectionist training, where the only pressure to adjust weights comes from error between output and input activity. It should perhaps not then be surprising to encounter local minima in the dimensionality reduction problem, where we impose a cost factor due to non-scalarlike behavior in hidden units, in addition to the normal cost for output activity deviation from input. In effect, what we are doing is adding two cost landscapes to one another. The weight adjustment that reduces one cost component may raise the other. Figure 6 is a simple illustration of one way in which adding two cost landscapes can create local minima.

Two strategies have been proposed for surmounting the local minimum problem. One is simulated annealing in a Boltzmann machine [Kirkpatrick, et al, 1983; Hinton, et al, 1984]. Briefly, simulated annealing allows the training process to adjust weights probabilistically so as to *increase* cost. This allows the procedure to jump out of local minima in cost. Boltzmann machine learning can be slow, and it requires certain conditions on the shape of the cost landscape in order to have a good chance of working. We have not investigated its applicability to the dimensionality-reduction problem.

Another strategy for skirting local minima involves changing the shape of the cost landscape itself as training proceeds. The idea is to introduce a parameter that makes the landscape very smooth at first, so that the

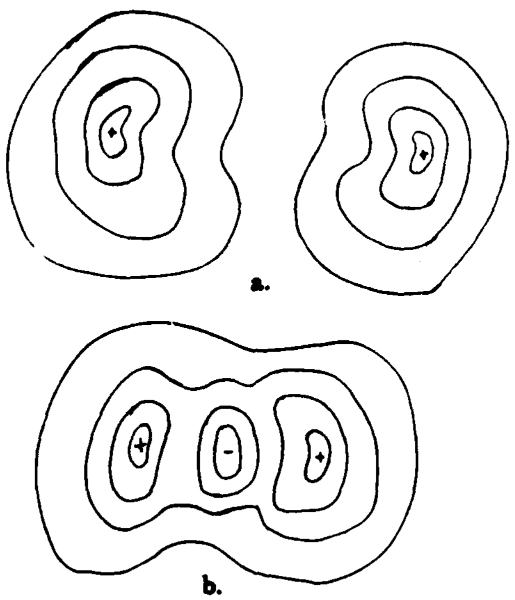


Figure 6. a. Neither of the cost landscapes shown has a local minimum by itself. b. But if they are moved near one another and added, local minima can be created in the resulting landscape.

network may easily converge to the local (and global) minimum. Then, gradually reduce this parameter to slowly change the landscape back into the "bumpy" cost landscape whose minimum defines the network behavior actually desired. A variant on this technique has been used by Hopfield and Tank [1985] to train networks to find good (but not optimal) solutions to the traveling salesman problem (see also [Koch, et al, 1985]).

For the dimensionality reduction problem we take as the cost landscape smoothing parameter the parameter, ω, of the smearing function, S_ω. At the beginning of a training session, the activity in all scalar-sets describing scalar-valued numbers is smeared across virtually all of the units within each scalar-set. Figure 4b illustrates the activity across a scalar-set under a variety of smoothing parameters, ω.

This strategy creates two related effects. First, it reduces the precision to which the data values presented as input activity, and sought by the output error term, are resolved. Thus, local kinks and details of any constraint curve constraining the input data are blurred over more or less, depending upon ω. Second, under smearing with a large ω, auxiliary error on the hidden layers pressures each unit's activity to be not too different from its neighbor's activity. The activity in hidden unit layers is thereby encouraged to organize itself into adopting the scalarized representation.

Training begins with the smearing parameter, ω, set to a large value. The parameter is gradually reduced to its final, highest resolution smearing value according to a training schedule. Typically, a training protocol might involve several thousand data-sampling/weight-updating trials for each of five intermediate values for ω.

8 Performance

The performance of the connectionist dimensionality-reducer on two-dimensional data constrained to lie on a one-dimensional constraint surface is illustrated in figure 7. X's represent locations indicated by output activity computed by the network when the input is drawn from points on the constraint curve; the extent to which X's lie on the curve simply demonstrates that network output conforms to input. Numbers shown are scalar values indicated by scalarized activity in the hidden layer scalar-set for inputs sampled from

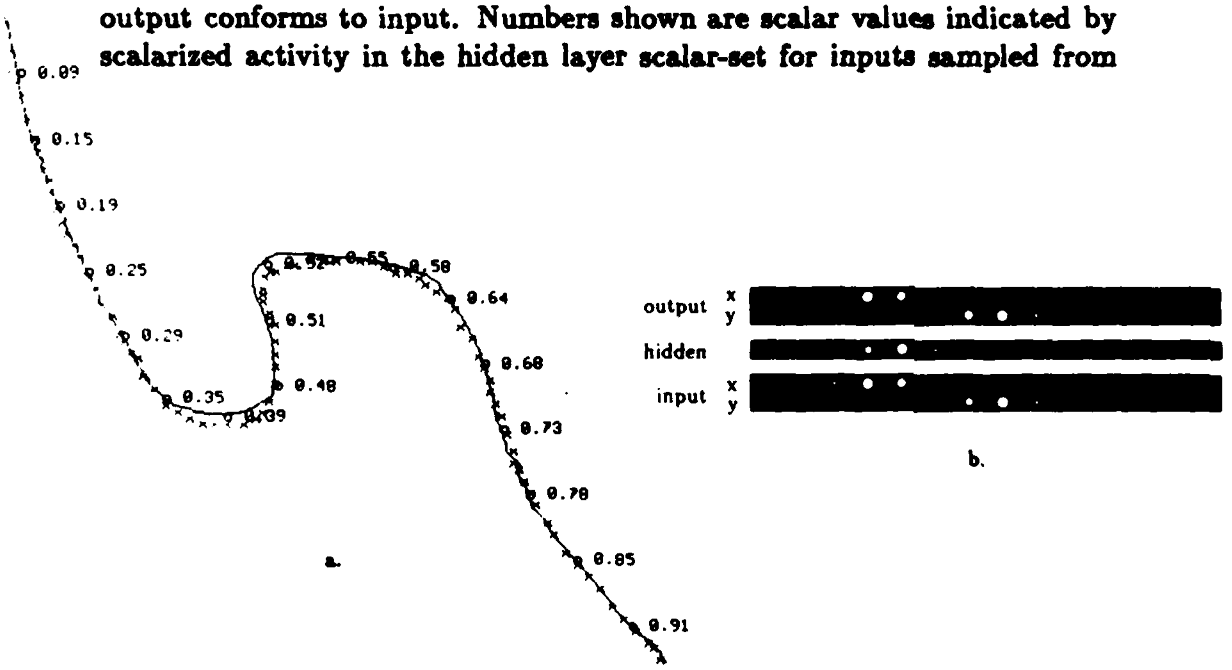


Figure 7. a. One-dimensional constraint in two-dimensional data. X's denote network output when input is taken from the constraint curve shown. Numbers indicate scalar value at hidden layer for points along constraint curve. b. Sample of network activity for one location along constraint curve. Note that output activity matches input activity, and that hidden layer scalar set activity takes a unimodal, scalarized, pattern.

locations along the constraint curve. Due to the self-organizing character of the training procedure, these numbers progress in an orderly fashion from one end of the constraint curve to the other. Figure 7b displays network activity for one data point drawn from the constraint curve. Note that output activity matches input activity, and that the activity of the hidden layer adopts a unimodal, scalarized pattern. Note also that in this case the connectionist dimensionality-reducer captures the constraint surface even though it doubles back on itself in both the x and y dimensions.

Figure 8 illustrates a case in which a connectionist dimensionality-reducer is able to discover a constraint surface given noisy training data. During training, data samples were drawn randomly from the band pictured. Shown slightly offset from this band, numbers indicate hidden scalar-set encoding of locations along the constraint curve found by the dimensionality-reducer (indicated by x's).

Figure 9 shows successful dimensionality-reduction given relatively sparsely sampled data. During training, data samples were drawn, at random, only from the points shown by X's. After training, the dimensionality-reducer nonetheless correctly interprets data at all locations along the length of the curve. Empirical investigation indicates that during training the constraint curve must be sampled no more sparsely than approximately three data points per hidden layer unit.

Some types of constraint surface cannot be discovered by the connectionist dimensionality-reducer. These are curves that double back on themselves radically. Figure 10 illustrates a case where hidden layer activity does not display a scalarized pattern of activity representing an orderly progression of scalar values for successive locations on the constraint curve. The reason for this failure is that points such as p_1 and p_2 in figure 10 appear indistinguishable to the network early in the training procedure when S_ω causes very heavy smearing of their coordinate representations. They are therefore assigned similar encodings in the hidden unit layer. As ω is decreased, later in the training procedure, the network remains stuck in a local minimum of trying to encode both p_1 and p_2 using nearby hidden scalar values, when in fact it turns out that they are on opposite ends of the constraint curve and so should be assigned very different encodings in the hidden layer scalar-set. The cost landscape sculpting strategy does not work when, as the landscape smoothing parameter is decreased, the global

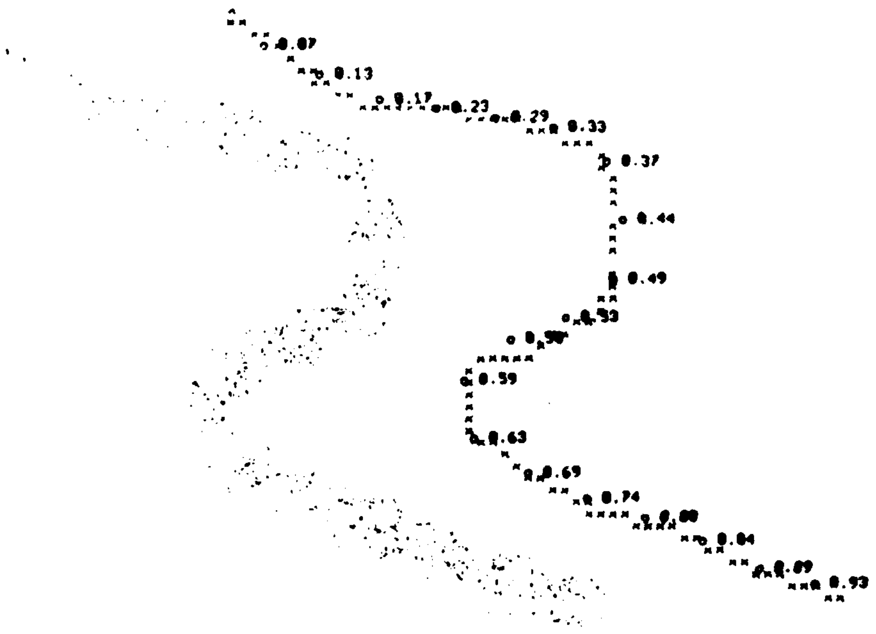


Figure 8. Dimensionality-reduction under noisy training data. For clarity, constraint curve found (denoted by x's) is shown displaced from the band of data samples used in training.

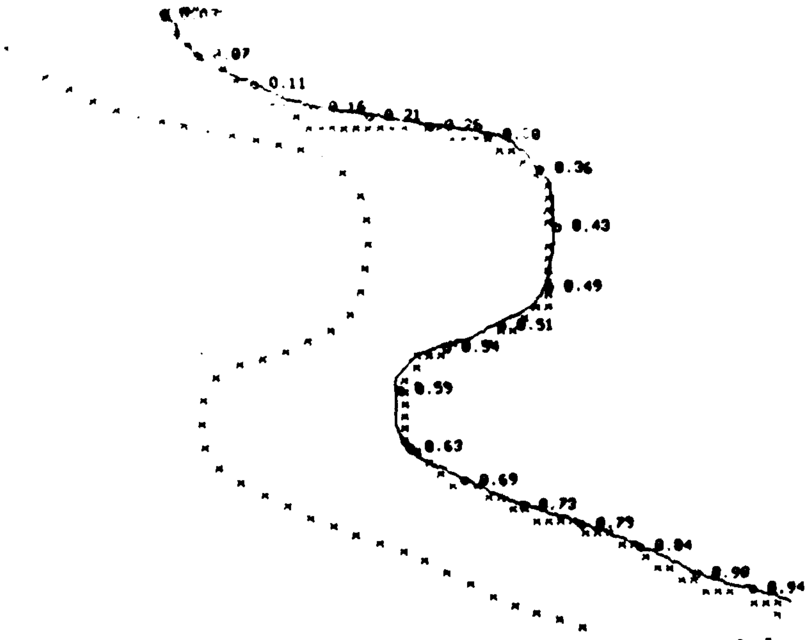


Figure 9. Dimensionality-reduction under sparsely sampled training data. During training, data was sampled from locations shown by X's at left. Each data sample was used repeatedly during training. Constraint curve found is shown displaced from the training data for clarity.

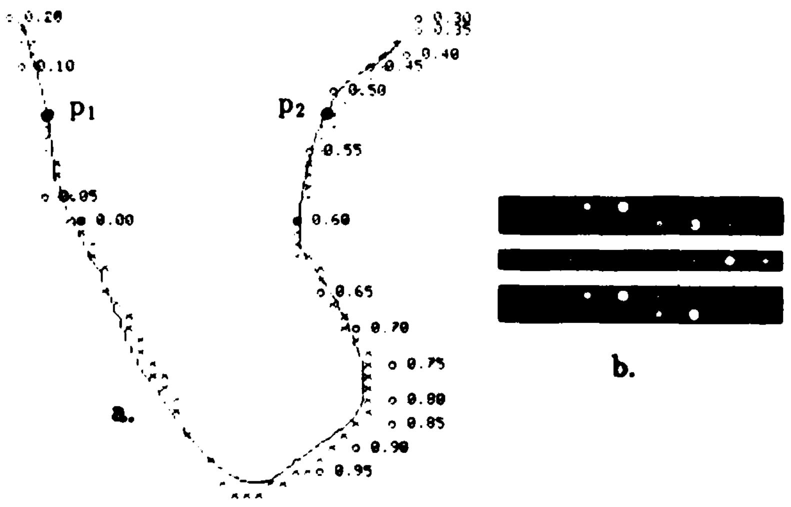


Figure 10. Unsuccessful dimensionality-reduction. When the constraint surface doubles back on itself radically, initial alignment of the network's early rough approximation to the data is not untangled as smearing width parameter ω decreases later in the training protocol. a. Numbers indicating best scalar-value interpretations of hidden layer activity do not progress in an orderly fashion from one end of the constraint curve to the other. b. Sample of network activity, after training, for one location on the constraint curve. Hidden layer does not display scalarised pattern of activity.

minimum in cost suddenly appears in a very different location in weight space from where the previous local minimum had been. Clearly, then, a network cannot be proved to converge to dimensionality-reducing behavior in the general case, which includes pathologically contorted constraint surfaces. However, once the training procedure is completed, it is a straightforward matter to detect whether or not dimensionality-reduction has been achieved, simply by sampling the data source, and determining whether the network maps input activity into (nearly) identical output activity via properly scalarised hidden layer activity.

Prior to training a dimensionality-reducer, it is important to select a dimensionality for the hidden layer to match the inherent dimensionality of the surface constraining the data. The connectionist dimensionality-reducer provides no means for doing this automatically. However, again, it is easy to detect whether the constraint surface is of inadequate dimensionality, because under this condition a network will converge to a state in which it does not correctly map activity at the input layer to (nearly) identical activity at the output layer in terms of a unimodal, scalarised pattern of activity at the hidden layer.

The method expands readily to larger high-dimensional feature spaces simply by adding more scalar-sets at the input and output layers to represent additional scalar components of the feature vector. Figure 11 illustrates the $n = 3, m = 2$ case. Figure 11a is the true underlying constraint surface. Figure 11b represents network output for input data drawn from the constraint surface. Figure 11c illustrates network output when activity corresponding to successive (h_1, h_2) pairs $(0 < h_1 < 1, 0 < h_2 < 1)$ is injected directly into the hidden layer.

However, the tractability of discovering many-dimensional constraint surfaces degrades quickly as the dimensionality of the hidden layer constraint surface increases. The amount of data that must be analyzed in order to establish a constraint surface increases as the power of the surface's dimensionality, and the cost in terms of network links and nodes increases accordingly. To give an idea of computing costs, a training protocol of 2000 trials for each of five values of the smearing parameter, ω, takes approximately 30 minutes for the $n = 3, m = 1$, case, with resolution of eight units per scalar-set, on a Symbolics 3600 lacking floating point hardware, while the $n = 3, m = 2$ case takes three hours.

To illustrate the application of dimensionality-reduction to real data, figure 12 shows a set of bananas that were originally described in terms of six properties crudely measured on the banana shapes, such as the distance between the ends and average curvature of various contour segments. By training a connectionist dimensionality-reducer on these data samples, the bananas were found to lie on a two-dimensional constraint surface in the six-dimensional feature space. The organization of this constraint surface is illustrated in the figure; bananas are placed on a plane according to their respective two-dimensional coordinates. Note that banana shapes are organized on the basis of very subtle differences in their geometrical properties.

Although the reduced dimensionality representation concisely encodes the essential parameters of variability among members of the data class falling on a constraint surface, the lower-dimensional coordinate axes do not necessarily align with interpretations of these parameters preferred by human observers. For example, the horizontal and vertical axes of figure 12 roughly correspond to curvature of the lower part of a banana, and banana size, respectively, however, the dimensionality-reduction training

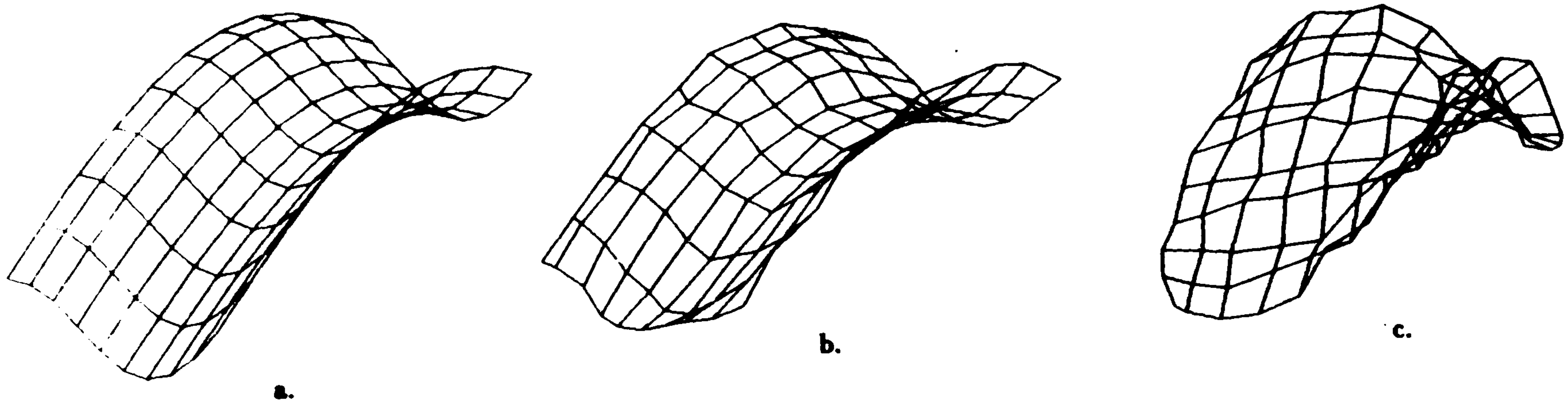
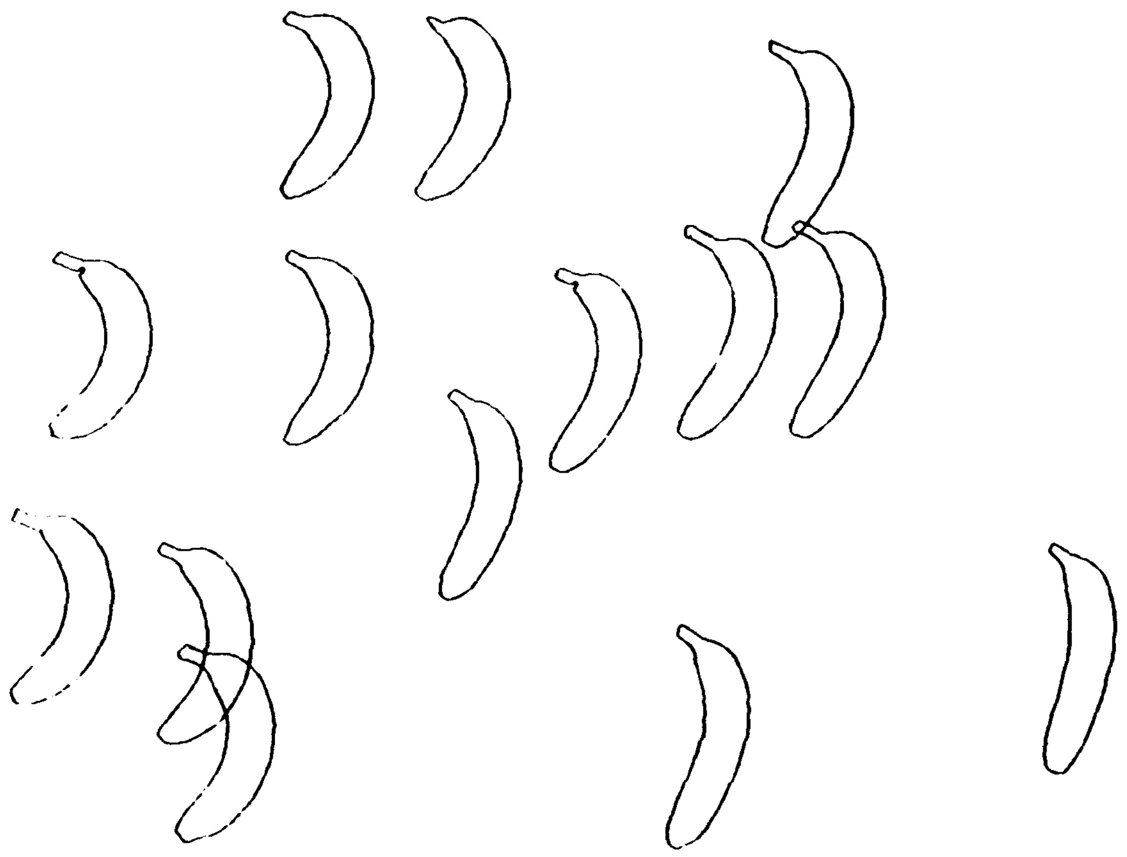
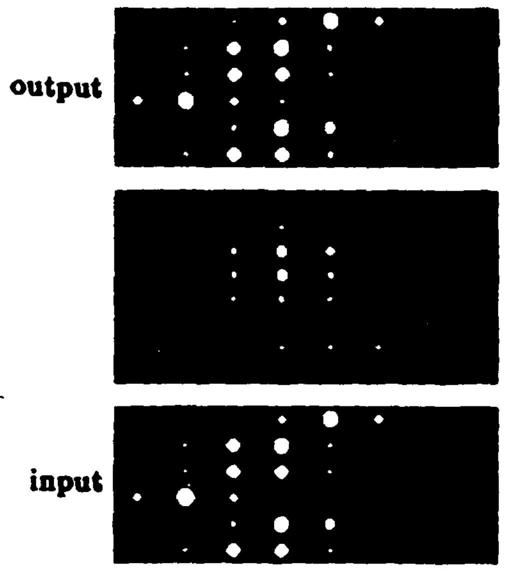


Figure 11. a. Two-dimensional constraint surface in three dimensional feature space. b. Activity at output layer when input layer is fed data from the constraint surface. c. Activity at output layer when hidden layer is injected with scalarized activity centered at regular locations on the scalar set: $(0 < h_1 < 1, 0 < h_2 < 1)$.



a.



b.

Figure 12. a. Banana shapes arranged according to their locations on a two-dimensional constraint surface found by a connectionist dimensionality-reducer. These were originally described in *terms of* six simple features such as distances between the ends and lengths and curvatures of various contour segments. The parameters of variability found to pertain among these bananas are roughly the curvature of the lower part of the banana (left/right), and the overall size of the banana (up/down). b. Activity of the dimensionality-reduction network for one banana. Note that output activity matches input activity, and that hidden layer activity is centered around one location in the two-dimensional scalar-set.

procedure run again on the same banana data might mirror reflect these axes, or rotate them an arbitrary amount in the plane.

9 Discussion

The connectionist dimensionality-reducer is able to capture a wide class of lower-dimensional constraint surfaces embedded in high dimensional feature spaces, even when the constraint surface curves to a considerable degree. The important distinction from previous, linear transformation approaches to dimensionality-reduction is that the connectionist approach enables the system to maintain a great deal of *knowledge* about constraint on the data source reflected in data samples. This knowledge is contained in the weights between units in successive layers of the network. Note that nowhere is the constraint surface described explicitly; its shape remains implicit in the weight connections. In contrast, only a few parameters are available to a linear transformation, which must therefore approximate a complex, curving constraint surface by a linear surface.

Analysis of a dimensionality-reducing network after the completion of training indicates that local regions of the hidden layer scalar-set map to local regions of the constraint surface, in a topology-preserving fashion.

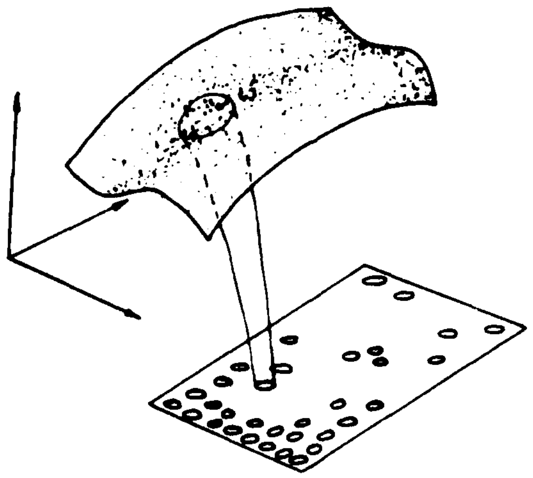


Figure 13. Local regions of a hidden-layer scalar set represent local regions of the embedded constraint surface, in a topology-preserving fashion. For example, regions on a two-dimensional constraint surface are represented by local patches of the hidden layer scalar-set.

For example, a data sample drawn from the constraint surface at the location shown in figure 13 would give rise to scalarized hidden layer activity centered at a corresponding location in the hidden layer scalar-set.

The connectionist dimensionality-reducer described here bears modest commonality with the method of Kohonen [1984]. Kohonen's method, which is based on his theory of the topographic mappings achieved by cells in the brain, also uses a large number of simple computing elements in whose connections are contained knowledge of a constraint surface. However, his method uses a very different type of self-organizing algorithm that confounds the shape of the underlying constraint surface with the probability distribution of data samples over that surface. The present method differs from Kohonen's in the employment of the backpropagation scheme for training multilayer networks, and in the explicit use of the landscape smoothing parameter, ω, to avoid local minima during training. The representation of scalar values in sets of units by use of a smearing function is similar to "value unit" coding described in [Ballard, 1986].

10 Conclusion

We have presented a mechanism for performing dimensionality reduction over data constrained to lie on a lower-dimensional surface embedded in a high-dimensional data feature space. A technique is given for representing in connectionist networks the scalar components of continuous vector-valued data. An auxiliary error pressure is introduced in order to pressure hidden units in the network into adopting this representation for scalar values.

This method has been shown capable of capturing lower-dimensional constraint surfaces which curve to a considerable degree. The network constructed by this method organizes and maintains knowledge of constraint latent in a set of data in order to encode information in a more concise representation than its original description as a high-dimensional feature vector. The connectionist dimensionality-reducer may also be viewed as a means for performing pattern matching to a variable, parameterized, pattern template given by the locus of points comprising the constraint surface in the embedding feature space.

References

Ackley, D., Hinton, G., and Sejnowski, T., [1985], "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, 9, 147-169.

Ballard, D., [1986], "Cortical Connections and Parallel Processing: Structure and Function," *Behavioral and Brain Sciences*, 9, 67-120.

Fukunaga, K., and Koontz, W.,[1970], "Application of the Karhunen-Loève Expansion to Feature Selection and Ordering," *IEEE Transactions on Computers*, C-19, 311-318.

Hinton, G., Sejnowski, T., and Ackley, D., [1984], "Boltzmann Machines: Constraint Satisfaction Networks that Learn," Technical Report CMU-CS-84-119, Carnegie Mellon University.

Hopfield, J., and Tank, D., [1985], "Neural Computation in Optimization Problems," *Biological Cybernetics*, 1985.

Kirkpatrick, S., Gelatt, Sl, and Vecchi, M., [1983], "Optimization by Simulated Annealing," *Science*, 220, 671-680.

Kittler, J., and Young, P., [1973], "A New Application to Feature Selection Based on the Karhunen-Loève Expansion," *Pattern Recognition*, 5, 335-352.

Koch, C., Marroquin, J., and Yuille, A., [1985], "Analog 'Neural' Networks in Early Vision," MIT AI Memo 751, MIT.

Kohonen, T., [1984], *Self-Organization and Associative Memory*, Springer-Verlag, Berlin.

Krishnaiah and Kanal, eds., [1982], *Handbook on Statistics, Vol. 2: Classification, Pattern Recognition, and Reduction of Dimensionality*, North-Holland.

Minsky, M., and Papert, S., [1969], *Perceptrons*, MIT Press, Cambridge, Ma.

Rosenblatt, F., [1962], *Principles of Neurodynamics*, Spartan, New York.

Rumelhart, D., Hinton, G., and Williams, R., [1985], "Learning Internal Representations by Error Propagation," ICS Report 8506, Institute for Cognitive Science, UCSD.

Rumelhart, D., Hinton, G., and Williams, R., [1986], "Learning Internal Representations by Back-propagating Errors," *Nature*, 323:9, 533-536.

Rumelhart, D., McClelland, J, and the PDP Research Group, [1986], *Parallel Distributed Processing: Explorations in the Structure of Cognition*, Bradford Books, Cambridge, Ma.

Saund, E., [1986], "Abstraction and Representation of Continuous Variables in Connectionist Networks," *Proceedings of the Fifth National Conference on Artificial Intelligence*, 638-644.

Tou, J., and Heydorn, R., [1967], "Some Optimal Approaches to Feature Extraction," in Tou, ed., *Computer and Information Sciences II*, Academic Press.

Watanabe, S.,[1965], "Karhunen-Loève Expansion and Factor Analysis," *Trans. 4th Prague Conference on Information Theory.*