AFRL-AFOSR-VA-TR-2015-0355

YIP Bell Inequalities for Complex Networks

Greg Ver Steeg
UNIVERSITY OF SOUTHERN CALIFORNIA LOS ANGELES

10/26/2015
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTC
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 20-10-2015	**2. REPORT TYPE** Final Performance Report	**3. DATES COVERED** *(From - To)* 01/08/2012 - 31/07/2015

4. TITLE AND SUBTITLE Bell Inequalities for Complex Networks	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER FA9550-12-1-0417
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Ver Steeg, Greg L.	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Information Sciences Institute University of Southern California, 4676 Admiralty Way, Ste 1001	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Special Research 875 North Randolph Street, Suite 325 Arlington, VA 22203	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFOSR
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) FA9550-12-1-0417

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION A

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This effort studied new methods to understand the effect of hidden variables affecting complex systems. Bell inequalities are a famous example of a hidden variable test in quantum physics that provides the strongest evidence for that theory. Initial work in this project extended the mathematical formulations of Bell inequalities to design new hidden variable tests that were able to account for confounding effects in complex systems including human social networks. These tests solved an open question about the identifiability of contagion in social network studies. Subsequent work moved beyond identification of hidden variables to develop a new information-theoretic framework capable of reconstructing hidden variables explaining the multivariate dependencies in complex systems. These methods have demonstrated value on diverse problems including human behavior, language, neuroscience, and gene expression.

15. SUBJECT TERMS
information theory, networks, complex systems, bell inequalities, algebraic geometry

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Greg Ver Steeg
U	U	U	UU	30	**19b. TELEPHONE NUMBER** *(Include area code)* 626-840-5901

Reset

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

Final performance report

PI: Greg Ver Steeg

Young Investigator Award
Grant Title: Bell Inequalities for Complex Networks
Grant #: FA9550-12-1-0417
Reporting Period: 1 August 2012 to 31 July 2015

Information Sciences Institute
University of Southern California,
4676 Admiralty Way
Marina del Rey, CA 90292

October 20, 2015

Final Report for "Bell Inequalities for Complex Networks"

Greg Ver Steeg

Abstract

This effort studied new methods to understand the effect of hidden variables affecting complex systems. Bell inequalities are a famous example of a hidden variable test in quantum physics that provides the strongest evidence for that theory. Initial work in this project extended the mathematical formulations of Bell inequalities to design new hidden variable tests that were able to account for confounding effects in complex systems including human social networks. These tests solved an open question about the identifiability of contagion in social network studies. Subsequent work moved beyond identification of hidden variables to develop a new information-theoretic framework capable of reconstructing hidden variables explaining the multivariate dependencies in complex systems. These methods have demonstrated value on diverse problems including human behavior, language, neuroscience, and gene expression.

1 Objectives

The original objectives of this project were the following:

- Objective 1: Develop statistical tests that identify which hidden processes give rise to network dynamics.
- Objective 2: Analyze key models of dynamic processes in networks, e.g., to distinguish whether influence or other hidden factors are responsible for correlations.
- Objective 3: Perform meta-analysis of studies that purport to identify influence in social networks. E.g., the highly publicized result that obesity is contagious in [1] has been criticized [2] and this method could be used to settle the controversy.
- Objective 4: Improve connection between probabilistic graphical models and algebraic geometry leading to more scalable methods.

These objectives were fulfilled in the work [3]. This led to a broadening of the scope of the project to include not just the identification of hidden variables, but also reconstruction. The following objectives became the central focus of the second phase of work.

- Objective 5: Deduce structural relationships resulting from common dependence on hidden variables and reconstruct these hidden variables.

- Objective 6: Generalize to arbitrary complex systems and provide theoretical guarantees to recover hidden variables.

2 Status of effort

This work successfully translated the implications of [4] to leverage ideas from algebraic geometry to construct tests for hidden variables ("Bell inequalities") that were applicable in complex settings like human social networks. The work of Christakis and Fowler suggesting that obesity could be contagious in social networks [1] inspired many papers copying their methodology to support other dubious claims along with many statistical papers criticizing the methodology. This controversy was summarized in [2], leaving an open question as to whether contagion could actually be identified in social networks. Our work [3] settled this open question by providing a constructive test that applied to the study of Christakis and Fowler. We applied this test to the Christakis and Fowler's original obesity study and found that, indeed, various non-contagion confounding effects suggested by [2] and others could be ruled out as the cause of strong correlations in obesity in social networks.

The second phase of this effort led to the development of information-theoretic methods to reconstruct underlying hidden factors responsible for multivariate dependence in data. This line of work [5, 6] has already been successfully applied in several domains including neuroscience [7, 8], analyzing text [9, 10], and gene expression.

3 Accomplishments / New findings

3.1 Identifying hidden variables

Christakis and Fowler's paper suggesting that obesity may spread along social ties [1] has sparked years of discussion about what constitutes evidence of contagion in observational social network studies (see, e.g., this recent review [**?**]). The most general result from the causal modeling perspective shows that latent homophily acts as a confounder for contagion so that uniquely pinpointing the strength of contagion is impossible without additional assumptions [2]. In other words, contagion is non-parametrically unidentifiable. However, if the true goal is to test for the presence of contagion, a lower bound on the strength of contagion is all that is necessary. Our work presented exactly such bounds, derived analogously to Bell inequalities but requiring more sophisticated mathematical methods from algebraic geometry.

Previous tests for contagion in social network studies were vulnerable to the confounding effects of latent homophily (i.e., ties form preferentially between individuals with similar hidden traits). We demonstrated a general method to lower bound the strength of causal effects in observational social network studies, even in the presence of arbitrary, unobserved individual traits. Our tests require no parametric assumptions and each test is associated with an algebraic proof. We demonstrated the effectiveness of our approach by correctly deducing the causal effects for examples previously shown to expose defects in existing methodology. Finally, we applied our methods to the Framingham Heart Study showing that various non-contagion confounding effects suggested by [2] and others could be ruled out as the cause of strong correlations in obesity in social networks [3].

Our method is based on an algebraic geometric approach to deducing the most general possible bounds compatible (or incompatible) with a given hidden variable model. It produces a sequence of bounds on the strength of contagion which converge in some limit to the best possible bounds. In this sense, our method is the best solution to the problem of measuring the strength of contagion that does not involve invoking additional (parametric) assumptions.

We also re-formulated the search for statistical tests as a linear program [3] instead of an SDP[4], leading to the discovery of more powerful tests with less computation. This re-formulation also allows for more flexibility in specifying the null (hidden variable) model. We applied the new tests to FHS data to deduce, for example, that correlations in obesity cannot be explained solely in terms of hidden variable models (without contagion), and correlations in smoking exhibit a non-stationary behavior suggesting common external cause as a necessary factor.

3.2 Recovering hidden variables

Without any prior knowledge, what can be automatically learned from high-dimensional data? If the variables are uncorrelated then the system is not really high-dimensional but should be viewed as a collection of unrelated univariate systems. If correlations exist, however, then some common cause or causes must be responsible for generating them. Without assuming any particular model for these hidden common causes, is it still possible to reconstruct them? We propose an information-theoretic principle, which we refer to as "correlation explanation", that codifies this problem in a model-free, mathematically principled way. Essentially, we are searching for latent factors so that, conditioned on these factors, the correlations in the data are minimized (as measured by multivariate mutual information). In other words, we look for the simplest explanation that accounts for the most correlations in the data. As a bonus, building on this information-based foundation leads naturally to an innovative paradigm for learning hierarchical representations that is more tractable than Bayesian structure learning and provides richer insights than neural network inspired approaches [**?**].

In initial work [5], we introduced a method to learn a hierarchy of successively more abstract representations of complex data based on optimizing an information-theoretic objective. Intuitively, the optimization searches for a set of latent factors that best explain the correlations in the data as measured by multivariate mutual information. The method is unsupervised, requires no model assumptions, and scales linearly with the number of variables which makes it an attractive approach for very high dimensional systems.

We demonstrated that Correlation Explanation (CorEx) automatically discovers meaningful structure for data from diverse sources including personality tests, DNA, and human language. In benchmarks, we showed that CorEx could recover synthetic latent tree models perfectly for problems orders of magnitude larger than standard techniques. Because CorEx scales linearly, it also outperforms methods specially designed for latent tree problems which scale cubically or worse.

In subsequent work [6], we deepened these results by proving rigorous bounds on the quality of the recovered latent factors. In particular, we consider a collection of random variables that each depend only on a set of input variables as a *representation* of the inputs. We presented bounds on how informative a representation is about some input data and extended these bounds to hierarchical representations so that we can quantify the contribution of each layer towards capturing the information in the original data. These results indicated a bottom-up procedure which casts the search for maximally informative hierarchical representations as an optimization problem. The complexity of the resulting optimization is linear in the number of variables. Compared to the results in [5], these results allow several generalizations. First of all, we no longer had to restrict ourselves to latent tree models; arbitrary connectivity between observed variables and learned latent factors is possible. Secondly, we showed in this work how to extend the results from discrete data to continuous data.

One of the drawbacks of the standard deep learning paradigms that is shared by CorEx is the need to specify the number of layers and hidden units in each layer ahead of time. Ideally, we could organically grow a network of hidden units, adding new ones as long as they provided an information advantage and then stopping. In [11], we introduced an approach that does this that we called the information sieve. The sieve is based on a novel hierarchical decomposition of information. Intuitively, data is passed through a series of progressively fine-grained sieves. Each layer of the sieve recovers a single latent factor that is maximally informative about multivariate dependence in the data. The data is transformed after each pass so that the remaining unexplained information trickles down to the next layer. Ultimately, we are left with a set of latent factors explaining all the dependence in the original data and remainder information consisting of independent noise. We presented a practical implementation of this framework for discrete variables. We showed that it could be used for a variety of tasks including (discrete) independent component analysis, lossy and lossless compression, and predicting missing values in data. For discrete independent component analysis, in particular, the sieve appears as a state-of-the-art approach. The best previous approach was exponential in the number of variables [**?**] while the sieve is linear. Although the previous approach provides better theoretical guarantees, the sieve will be more useful in many practical, high-dimensional situations.

4 Personnel supported

This young investigator award was solely used to support a portion of the effort for the PI, Greg Ver Steeg (Research Faculty).

5 Publications

The following publications were supported by this effort:

- Greg Ver Steeg and Aram Galstyan. Statistical tests for contagion in observational social network studies. In Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS), 2013.
- Greg Ver Steeg and Aram Galstyan. Discovering structure in high-dimensional data through correlation explanation. In Advances in Neural Information Processing Systems (NIPS), 2014.
- Greg Ver Steeg and Aram Galstyan. Maximally informative hierarchical representations of high-dimensional data. In Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS), 2015.
- Greg Ver Steeg and Aram Galstyan. The Information Sieve. arXiv:1507.02284, 2015.

Code that resulted from these publications is included in appendices.

6 Interactions/Transitions

I participated in the AFOSR Complex Networks program reviews and I also visited AFRL Rome for a few days to discuss collaborative efforts and gave a talk for the "Distinguished Speaker Seminar Series in Machine Intelligence and Autonomy" in 2014. Besides these Air Force affiliated interactions, I disseminated results of this effort in standard academic venues including presentations at AISTATS in 2013 and 2015 and NIPS in 2014. Other presentations and invited talks include the following.

- Joint Symposium on Neural Computation at USC 2015
- Machine Learning LA at eHarmony 2015
- Information Theory and Applications Workshop 2015
- Santa Fe Institute, workshop on "Statistical Mechanics of Complexity" 2014
- IPAM Mathematics of Social Learning Workshop 2014
- ISI, Natural Language Seminar 2013
- ID Analytics, "Information-Theoretic Tools for Social Media" 2013
- Santa Fe Institute, workshop on "Structure, Statistical Inference and Dynamics in Networks" 2013

- Sante Fe Institute, “Information-Theoretic Tools for Social Media” 2012
- Keynote for “Making Sense of Microposts” workshop at WWW 2012. 2012
- SAP Research, Singapore 2012
- LARC, Singapore Management University seminar. 2012
- UC Irvine, AI-ML seminar. 2012
- Society of Biological Psychiatry, 2015.
- NIPS workshop on Machine Learning in Computational Biology (MLCB), 2014.
- NIPS workshop on Machine Learning and Interpretation in Neuroimaging (MLINI), 2014. *Workshop on Information in Networks*, 2013. *AAAI-13 Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI)*, 2013.
- *Workshop on Information in Networks(WIN)*, 2012.

7 Inventions and patent disclosures

None.

8 Honors/Awards

None.

References

[1] Nicholas A. Christakis and James H. Fowler. The spread of obesity in a large social network over 32 years. *The New England Journal of Medicine*, 357(4):370–379, July 2007.

[2] Cosma R. Shalizi and Andrew C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *arxiv:1004.4704*, 2010.

[3] Greg Ver Steeg and Aram Galstyan. Statistical tests for contagion in observational social network studies. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.

[4] Greg Ver Steeg and Aram Galstyan. A sequence of relaxations constraining hidden variable models. In *Proc. of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, 2011.

[5] Greg Ver Steeg and Aram Galstyan. Discovering structure in high-dimensional data through correlation explanation. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[6] Greg Ver Steeg and Aram Galstyan. Maximally informative hierarchical representations of high-dimensional data. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015. `http://arxiv.org/abs/1410.7404`.

[7] Sarah K. Madsen, Greg Ver Steeg, Adam Mezher, Neda Jahanshad, Talia M. Nir, Xue Hua, Boris A. Gutman, Aram Galstyan, and Paul M. Thompson. Information-theoretic characterization of blood panel predictors for brain atrophy and cognitive decline in the elderly. *IEEE International Symposium on Biomedical Imaging*, 2015.

[8] Madelaine Daianu, Greg Ver Steeg, Adam Mezher, Neda Jahanshad, Talia M. Nir, Xiaoran Yan, Gautam Prasad, Kristina Lerman, Aram Galstyan, and Paul M. Thompson. Information-theoretic clustering of neuroimaging metrics related to cognitive decline in the elderly. In *Proceedings of the MICCAI Workshop on Medical Computer Vision*, 2015.

[9] Nathan Hodas, Greg Ver Steeg, Joshua Harrison, Satish Chikkagoudar, Eric Bell, and Courtney Corley. Disentangling the lexicons of disaster response in twitter. In *The 3rd International Workshop on Social Web for Disaster Management (SWDM'15)*, 2015.

[10] Peixian Chen, Nevin L Zhang, Leonard KM Poon, and Zhourong Chen. Progressive em for latent tree models and hierarchical topic detection. *arXiv preprint arXiv:1508.00973*, 2015.

[11] Greg Ver Steeg and Aram Galstyan. The information sieve. *arXiv preprint arXiv:1507.02284*, 2015.

A Contagion code

This code resulted from [3].

Statistical tests for contagion in observational social network studies

Greg Ver Steeg and Aram Galstyan
Information Sciences Institute, USC

This notebook contains code and examples for the paper of the associated title.

Constructing tests

Definitions

These definitions are used in the rest of the notebook. Note that we always use lexical order for defining binary sequences. A,B=((0,0,0),(0,0,0)),((0,0,0),(0,0,1))...

```
vec[j_,T_]:=IntegerDigits[j,2,T];
vecboth[j_,T_]:={vec[Quotient[j,2^T],T],vec[Mod[j,2^T],T]};
(*Number of transitions of each type*)
f00[a_]:=Sum[(1-a[[i]])(1-a[[i+1]]),{i,Length[a]-1}];
f01[a_]:=Sum[(1-a[[i]])(a[[i+1]]),{i,Length[a]-1}];
f10[a_]:=Sum[(a[[i]])(1-a[[i+1]]),{i,Length[a]-1}];
f11[a_]:=Sum[(a[[i]])(a[[i+1]]),{i,Length[a]-1}];
(*For a non-causal model*)
pseq[a_,pfp_,pfm_,p0_]:= pfp^f01[a] pfm^f10[a] (1-pfp)^f00[a] (1-pfm)^f11[a] p0^a[
(*For a causal model*)
pseqc[a_,b_,pfp_,pfm_,p0_,b0_,b00_,b01_,b10_,b11_]:= pseq[a,pfp,pfm,p0] b0^b[[1]] (1-b0)^
 Which[
 a[[i]]==0 && b[[i]]==0 && b[[i+1]]==1,b00,
 a[[i]]==0 && b[[i]]==0 && b[[i+1]]==0,1-b00,
 a[[i]]==0 && b[[i]]==1 && b[[i+1]]==1,b01,
 a[[i]]==0 && b[[i]]==1 && b[[i+1]]==0,1-b01,
 a[[i]]==1 && b[[i]]==0 && b[[i+1]]==1,b10,
 a[[i]]==1 && b[[i]]==0 && b[[i+1]]==0,1-b10,
 a[[i]]==1 && b[[i]]==1 && b[[i+1]]==1,b11,
 a[[i]]==1 && b[[i]]==1 && b[[i+1]]==0,1-b11
 ],
 {i,Length[b]-1}];
(*For an "instant" causal model*)
pseqi[a_,b_,pfp_,pfm_,p0_,bp0_,bp1_,b00_,b01_,b10_,b11_]:= pseq[a,pfp,pfm,p0] Which[
a[[1]]==0,bp0^b[[1]] (1-bp0)^(1-b[[1]]),
a[[1]]==1,bp1^b[[1]] (1-bp1)^(1-b[[1]])
] Product[
 Which[
 a[[i+1]]==0 && b[[i]]==0 && b[[i+1]]==1,b00,
 a[[i+1]]==0 && b[[i]]==0 && b[[i+1]]==0,1-b00,
 a[[i+1]]==0 && b[[i]]==1 && b[[i+1]]==1,b01,
 a[[i+1]]==0 && b[[i]]==1 && b[[i+1]]==0,1-b01,
 a[[i+1]]==1 && b[[i]]==0 && b[[i+1]]==1,b10,
 a[[i+1]]==1 && b[[i]]==0 && b[[i+1]]==0,1-b10,
 a[[i+1]]==1 && b[[i]]==1 && b[[i+1]]==1,b11,
 a[[i+1]]==1 && b[[i]]==1 && b[[i+1]]==0,1-b11
 ],
 {i,Length[b]-1}];


(*Influence model 1, "Instant" influence*)
phat1[δ_,T_]:= Flatten[Table[2^-T Product[δ Boole[vec[i,T][[t]]==vec[j,T][[t]]]+(1-δ)/2,{t,1,
(*Influence model 2, Delayed influence*)
phat2[δ_,T_]:= Flatten[Table[2^(-T-1) Product[δ Boole[vec[i,T][[t-1]]==vec[j,T][[t]]]+(1-δ)/2,{

(*Given M samples and <c>_obs, the confidence that <c>_real ≥ z = <c>_obs-γ, ASSUMING c i
hoeffding[M_,z_]:=1-Exp[-2 M (z/2)^2];
norm[z_]:=z/Total[z];

id[j_,NN_]:=Table[Boole[i==j],{i,NN}];
(*Convert counts to actual random vectors of which we have M samples*)
rawdata[vector_,counts_]:=Flatten[
 Table[
 Table[vector.id[j,Length[counts]],{k,counts[[j]]}],
 {j,Length[counts]}],
 1];
(*Confidence based on Binomial distribution. Return confidence level rather than p-value*
testbinomial[vector_,counts_,greater_]:=1-N[CDF[BinomialDistribution[Abs[vector].counts,1,
```

FHS data

```
(*Counts of sequences for NR friends, also listed in paper*)
obesity = {5, 2, 2, 0, 4, 3, 2, 2, 1, 3, 3, 3, 3, 1, 2, 4, 4, 1, 6, 1, 3, 4, 5, 3, 2, 0, 8, 0,
   1, 4, 2, 1, 4, 7, 2, 2, 5, 3, 8, 2, 4, 2, 3, 4, 2, 3, 2, 1, 1, 1, 3, 1, 2, 2, 1, 3, 1,
   5, 2, 6, 3, 6, 3, 3, 6, 2, 4, 3, 7, 9, 6, 6, 1, 0, 5, 4, 4, 4, 3, 1, 2, 3, 2, 1, 5, 4,
   3, 5, 2, 7, 4, 1, 2, 3, 0, 1, 1, 2, 3, 6, 4, 5, 5, 4, 1, 7, 3, 1, 0, 6, 0, 0, 0, 0, 5,
   2, 3, 2, 4, 2, 6, 1, 1, 1, 3, 2, 1, 3, 3, 3, 2, 3, 3, 2, 1, 2, 0, 2, 3, 6, 2, 5, 1, 0,
   3, 1, 1, 3, 0, 7, 6, 1, 2, 0, 2, 2, 1, 6, 4, 0, 3, 8, 8, 5, 6, 3, 3, 6, 2, 7, 5, 3,
   5, 3, 5, 2, 3, 2, 4, 5, 2, 2, 4, 2, 4, 3, 4, 5, 1, 2, 2, 3, 3, 1, 2, 3, 3, 0, 2, 5,
   2, 1, 2, 1, 0, 1, 3, 0, 3, 3, 0, 1, 3, 4, 3, 2, 2, 6, 4, 2, 2, 11, 1, 2, 4, 2, 2, 3,
   3, 1, 0, 2, 1, 3, 2, 0, 1, 1, 1, 1, 2, 1, 1, 3, 1, 0, 1, 4, 0, 1, 5, 3, 0, 1, 1, 1};
```

ST Data

Here input data from simulations using Shalizi/Thomas models

```
sthomophily = {106 344, 2067, 2304, 1048, 2159, 954, 1038, 1682, 2189, 1073, 1010,
   1713, 1059, 1761, 1776, 41 313, 2330, 109, 89, 45, 96, 40, 53, 64, 73, 59, 45,
   77, 61, 84, 80, 1890, 2507, 97, 108, 46, 92, 52, 53, 104, 97, 40, 64, 84, 59,
   88, 108, 1967, 1194, 48, 48, 36, 52, 32, 25, 44, 40, 29, 23, 58, 29, 38, 46,
   1238, 2415, 73, 108, 59, 102, 46, 55, 96, 109, 49, 46, 78, 41, 66, 85, 1918,
   1076, 42, 46, 26, 52, 27, 37, 48, 46, 36, 32, 55, 25, 40, 38, 1156, 1083, 59,
   43, 47, 48, 17, 29, 42, 68, 38, 23, 56, 28, 41, 44, 1192, 1961, 75, 80, 42, 78,
   43, 39, 95, 87, 52, 46, 99, 46, 78, 103, 2405, 2359, 99, 103, 53, 83, 45, 55,
   89, 91, 52, 48, 73, 57, 77, 75, 1985, 1207, 52, 42, 24, 32, 25, 34, 46, 56, 31,
   32, 50, 21, 49, 49, 1264, 1199, 42, 39, 27, 43, 27, 27, 53, 44, 30, 23, 62, 27,
   47, 51, 1175, 1926, 88, 94, 56, 72, 55, 46, 114, 77, 55, 49, 107, 48, 91, 94,
   2417, 1256, 48, 52, 30, 51, 25, 29, 42, 46, 28, 20, 48, 26, 36, 51, 1182, 1926,
   80, 86, 45, 69, 43, 41, 86, 60, 51, 34, 99, 45, 93, 84, 2419, 1906, 95, 83, 47,
   79, 53, 58, 92, 73, 54, 41, 109, 50, 108, 98, 2457, 41 840, 1734, 1777, 1014,
   1765, 1028, 996, 2170, 1704, 1125, 1075, 2192, 1095, 2120, 2115, 106 407};
stinfluence = {17 999, 2339, 2289, 1241, 3224, 1127, 1217, 950, 13 987, 2241, 2251,
   1115, 3235, 1146, 1266, 1197, 2350, 2009, 489, 1026, 510, 839, 252, 853, 1671,
   1811, 466, 1041, 479, 762, 257, 1254, 2262, 477, 1479, 849, 610, 275, 850, 774,
   1655, 461, 1431, 872, 564, 262, 828, 1117, 1189, 1054, 896, 2840, 279, 568, 525,
   2527, 834, 1040, 867, 2669, 316, 574, 547, 3302, 3243, 488, 587, 269, 2674, 773,
   1007, 809, 2604, 508, 576, 293, 2892, 769, 1019, 1180, 1161, 869, 260, 543,
   801, 1452, 432, 1550, 792, 813, 253, 557, 830, 1439, 478, 2361, 1146, 248, 854,
   559, 1003, 449, 1929, 1633, 860, 242, 867, 518, 1100, 453, 1872, 2392, 992,
   887, 767, 2509, 734, 1560, 1536, 10 086, 667, 843, 761, 2546, 832, 1689, 1640,
   13 999, 13 947, 1682, 1655, 795, 2551, 723, 903, 688, 9829, 1602, 1525, 759,
   2579, 740, 930, 928, 2222, 1862, 489, 1060, 499, 859, 266, 851, 1565, 1766, 488,
   982, 529, 817, 267, 1241, 2192, 449, 1530, 823, 538, 236, 809, 761, 1539, 439,
   1394, 788, 576, 238, 843, 1130, 1121, 1012, 786, 2671, 276, 537, 538, 2446, 748,
   973, 767, 2478, 274, 536, 482, 3150, 3317, 503, 540, 270, 2775, 812, 1077, 803,
   2580, 515, 579, 262, 3064, 800, 1136, 1242, 1039, 788, 230, 625, 798, 1498, 454,
   1634, 683, 840, 230, 540, 870, 1489, 472, 2375, 1237, 261, 891, 502, 1080, 454,
   1985, 1709, 829, 276, 782, 512, 1179, 478, 2075, 2354, 1280, 1231, 1095, 3342,
   1173, 2179, 2456, 13 982, 946, 1196, 1186, 3095, 1216, 2325, 2473, 18 561};
```

The Basic Optimization Problem

Given some distribution, $\hat{p}$, we want to define an optimization problem that returns $c\,(A, B)$, γ so that $\langle c(A, B)\rangle_{\hat{p}} - \langle c(A, B)\rangle_{\text{nc}} \geq \gamma$ for all non-causal models,nc, and γ is maximized. Eq. 1.7 in the paper.

```
LP1[phat_, T_: 3, setmaxd_: -1, method_: "InteriorPoint"] :=
 Module[{a0, b0, ap, am, bp, bm, c, γ, λ},
  maxd = If[setmaxd < 0, 2 T, setmaxd];
  vars = {a0, b0, ap, am, bp, bm};
  (*An arbitrary operator to take exp. value of*)
  cs = Flatten[Table[c[{i, j}], {i, 0, 2^T - 1}, {j, 0, 2^T - 1}]];
  (*The quantity we ensure is non-negative*)
  exp = -γ + cs.phat - Sum[c[{i, j}] pseq[vec[i, T], ap, am, a0]
      pseq[vec[j, T], bp, bm, b0], {i, 0, 2^T - 1}, {j, 0, 2^T - 1}];

  (*Monomials in the Handelman representation*)
  gs = {a0, b0, 1 - a0, 1 - b0, ap, am, bp, bm, 1 - ap, 1 - am, 1 - bp, 1 - bm};
   (*Powers to use in the Handelman representation*)
  powers = Flatten[Table[{i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11, i12},
     {i1, 0, maxd},
      {i2, 0, maxd - i1},
      {i3, 0, maxd - i1 - i2},
      {i4, 0, maxd - i1 - i2 - i3},
      {i5, 0, maxd - i1 - i2 - i3 - i4},
      {i6, 0, maxd - i1 - i2 - i3 - i4 - i5},
      {i7, 0, maxd - i1 - i2 - i3 - i4 - i5 - i6},
      {i8, 0, maxd - i1 - i2 - i3 - i4 - i5 - i6 - i7},
      {i9, 0, maxd - i1 - i2 - i3 - i4 - i5 - i6 - i7 - i8},
      {i10, 0, maxd - i1 - i2 - i3 - i4 - i5 - i6 - i7 - i8 - i9},
      {i11, 0, maxd - i1 - i2 - i3 - i4 - i5 - i6 - i7 - i8 - i9 - i10},
      {i12, 0, maxd - i1 - i2 - i3 - i4 - i5 - i6 - i7 - i8 - i9 - i10 - i11}], 11];
  powers = Select[powers, Total[#[[1 ;; 4]]] ≤ 2 &];
  (*For a0,1-a0,b0,1-b0, only squared powers*)
  powers = Select[powers, Total[#] ≥ 2 T || Total[#] ≤ 1 &];
  (*It seems this doesn't affect bounds*)

  lambdas = Map[λ[#] &, powers]; (*coefficients in H. Rep.*)
  handelman = FromCoefficientRules[Map[# → λ[#] &, powers], gs];
  (*The polynomial*)
  equalities = Select[Flatten[CoefficientList[exp - handelman, vars]],
    ! MatchQ[#, 0] &];
  (*Equate poly monomials*)

  infostring = "Number of terms in handelman rep, : " <>
    ToString[Length[lambdas]] <> ", for maxd: " <> ToString[maxd] <>
    ", should be <= : " <> ToString[Binomial[maxd + 12, maxd]] <>
    " Total var in LP: " <> ToString[Length[lambdas] + Length[cs] + 1] <>
    " Total number of constraints from equating terms of polys: " <>
    ToString[Length[equalities]];
  temp = PrintTemporary[infostring];

  (*Convert representation of
    constraints for LinearProgramming call into m.x=b*)
  res = CoefficientArrays[equalities, Join[{γ}, lambdas, cs]];
  m = res[[2]];
  b = Map[{#, 0} &, -res[[1]]];
  maxgamma = Join[{-1}, Table[0, {i, Length[lambdas] + Length[cs]}]];
  (*{Lower,Upper bounds} on variables*)
  constraints = Join[Table[{0, Infinity}, {i, Length[lambdas] + 1}],
    Table[{-1, 1}, {i, Length[cs]}]];

  (*Use method "RevisedSimplex" for numerically exact (but slower) answers*)
  sol = LinearProgramming[maxgamma, m, b, constraints, Method → method];

  NotebookDelete[temp];
  (*Return γ,c.phat-γ,c,λ*)
  If[Length[sol] == Length[lambdas] + Length[cs] + 1, {sol[[1]],
    sol[[2 + Length[lambdas] ;;]].phat - sol[[1]], sol[[2 + Length[lambdas] ;;]],
    sol[[2 ;; Length[lambdas] + 1]]}, Print["Could not find a bound"];
   0]
]
```

Defining the equalities

Here we define the set of equality constraints that must be satisfied.

```
equal[vectorpoly_, vars_] :=
 Module[{c, NN = Length[vectorpoly]},
  exp = Sum[c[i] vectorpoly[[i]], {i, NN}];
  equalities = Select[Flatten[CoefficientList[exp, vars]], ! MatchQ[#, 0] &];
  (*Equate poly monomials*)
  NullSpace[CoefficientArrays[equalities, Table[c[i], {i, NN}]][[2]]]
 ]


Print["Equalities satisfied by non-causal models"]
vars = {a0, b0, ap, am, bp, bm};
T = 4;
matrixeq4 =
  equal[Flatten[Table[pseq[vec[i, T], ap, am, a0] pseq[vec[j, T], bp, bm, b0],
     {i, 0, 2^T - 1}, {j, 0, 2^T - 1}], 1], vars];
Print["For T=", T, " there are ", Dimensions[matrixeq4][[1]],
 " linearly independent constraints"]
T = 5;
matrixeq5 =
  equal[Flatten[Table[pseq[vec[i, T], ap, am, a0] pseq[vec[j, T], bp, bm, b0],
     {i, 0, 2^T - 1}, {j, 0, 2^T - 1}], 1], vars];
Print["For T=", T, " there are ", Dimensions[matrixeq5][[1]],
 " linearly independent constraints"]

Print["For causal models"]
varsc = {ap, am, a0, b0, b00, b01, b10, b11};
T = 4;
matrixeq4c =
  equal[Flatten[Table[pseqc[vec[i, T], vec[j, T], ap, am, a0, b0, b00, b01, b10, b11],
     {i, 0, 2^T - 1}, {j, 0, 2^T - 1}], 1], varsc];
Print["For T=", T, " there are ", Dimensions[matrixeq4c][[1]],
 " linearly independent constraints"]
T = 5;
matrixeq5c =
  equal[Flatten[Table[pseqc[vec[i, T], vec[j, T], ap, am, a0, b0, b00, b01, b10, b11],
     {i, 0, 2^T - 1}, {j, 0, 2^T - 1}], 1], varsc];
Print["For T=", T, " there are ", Dimensions[matrixeq5c][[1]],
 " linearly independent constraints"]

Print["For 'instant' causal models"]
varsi = {ap, am, a0, bp0, bp1, b00, b01, b10, b11};
T = 4;
matrixeq4i = equal[
   Flatten[Table[pseqi[vec[i, T], vec[j, T], ap, am, a0, bp0, bp1, b00, b01, b10, b11],
     {i, 0, 2^T - 1}, {j, 0, 2^T - 1}], 1], varsi];
Print["For T=", T, " there are ", Dimensions[matrixeq4i][[1]],
 " linearly independent constraints"]
T = 5;
matrixeq5i = equal[
   Flatten[Table[pseqi[vec[i, T], vec[j, T], ap, am, a0, bp0, bp1, b00, b01, b10, b11],
     {i, 0, 2^T - 1}, {j, 0, 2^T - 1}], 1], varsi];
Print["For T=", T, " there are ", Dimensions[matrixeq5i][[1]],
 " linearly independent constraints"]
```

```
Equalities satisfied by non-causal models

For T=4 there are 60 linearly independent constraints

For T=5 there are 540 linearly independent constraints

For causal models

For T=4 there are 28 linearly independent constraints

For T=5 there are 316 linearly independent constraints

For 'instant' causal models

For T=4 there are 24 linearly independent constraints

For T=5 there are 296 linearly independent constraints
```

Useful equality constraints

Test1 corresponds to c2 in the paper. Test2 corresponds to c1.

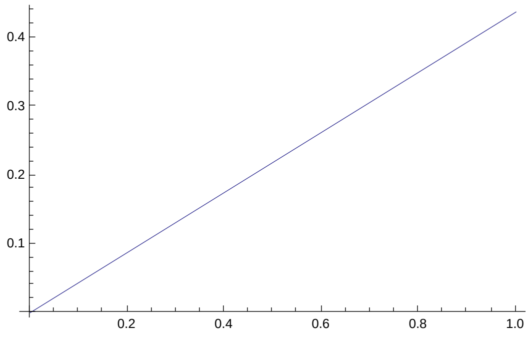
```
(*Using "RevisedSimplex" leads to exact results for the LP, but is slower*)
{ϒ, bound, test1, lambdas} = LP1[phat2[0.1, 4], 4, 0, "RevisedSimplex"];
test1 = Round[test1, 1 / 24]
{ϒ, bound, test2, lambdas} = LP1[phat1[0.1, 4], 4, 0];
test2 = Round[test2, 1]

ptest = Flatten[Table[pseq[vec[i, 4], ap, am, a0] pseq[vec[j, 4], bp, bm, b0],
    {i, 0, 2^4 - 1}, {j, 0, 2^4 - 1}], 1];

(*Verifies equality is satisfied algebraically*)
Print["Check equality test1: ", FullSimplify[test1.ptest]];
Print["Check equality test2: ", FullSimplify[test2.ptest]];
Plot[test1.phat2[δ, 4], {δ, 0, 1}]
Plot[test2.phat1[δ, 4], {δ, 0, 1}]
FullSimplify[test1.phat1[δ, 4]]
FullSimplify[test1.phat2[δ, 4]]
FullSimplify[test2.phat1[δ, 4]]
FullSimplify[test2.phat2[δ, 4]]
```

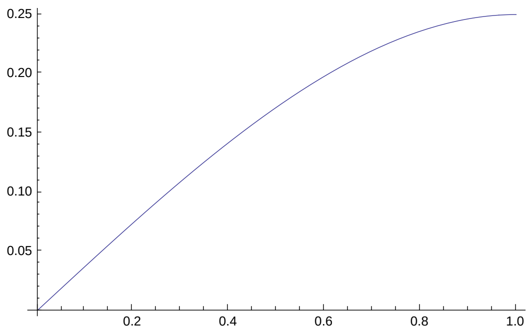
```
{0, 0, 1, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 1, 0, 0, 0, 0, 1, 0, -1, 0, 0, 0, 0, 0, 0,
 1, 0, -1, 0, 0, -1, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, 0, 0,
 -1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, -1, 0, 0, 1, -1, 1, 1, -1, -1, 1, 1, -1, -1, 1,
 -1, 1, -1, 1, 1, 0, 0, 1, 0, -1, 0, 0, 0, 0, 0, 0, 1, 0, -1, 0, 0, 0, 0, 1, 0, -1,
 0, 0, 0, 0, 0, 0, 1, 0, -1, 0, 0, 0, 0, 1, 0, -1, 0, 0, 0, 0, 0, 0, 1, 0, -1, 0, 0,
 0, 0, -1, 0, 1, 0, 0, 0, 0, 0, 0, -1, 0, 1, 0, 0, 0, 0, -1, 0, 1, 0, 0, 0, 0, 0, 0,
 -1, 0, 1, 0, 0, 0, 0, -1, 0, 1, 0, 0, 0, 0, 0, 0, -1, 0, 1, 0, 0, 1, 1, -1, 1, 1,
 1, -1, 1, -1, 1, -1, 1, 1, 1, -1, 1, 0, 0, 1, 0, -1, 0, 0, 0, 0, 0, 0, 1, 0, -1,
 0, 0, -1, -1, 1, -1, -1, -1, 1, -1, 1, -1, 1, -1, -1, -1, 1, -1, 0, 0, 1, 0, -1,
 0, 0, 0, 0, 0, 0, -1, 0, 1, 0, 0, 0, 0, -1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, -1, 0, 0}
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 1, 1, -1, -1, 0, 0, 0, 0, 1, 1, -1, -1, 0, 0, 0, 0, 1, 0, -1, 0, 0, 0, 0, 0, 0,
 1, 0, -1, 0, 0, 0, 0, -1, -1, 1, 1, 0, 0, 0, 0, -1, -1, 1, 1, 0, 0, 0, 0, -1, 0, 1, 0,
 0, 0, 0, 0, 0, -1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, -1, 0, 0, 0, 0, 0, 0, 1, 0,
 -1, 0, 0, 0, 0, 1, 1, -1, -1, 0, 0, 0, 0, 1, 1, -1, -1, 0, 0, 0, 0, -1, 0, 1, 0, 0, 0,
 0, 0, 0, -1, 0, 1, 0, 0, 0, 0, -1, -1, 1, 1, 0, 0, 0, 0, -1, -1, 1, 1, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
Check equality test1: 0
```

```
Check equality test2: 0
```





$$\frac{1}{32}\,\delta\,\left(-7+3\,\delta^2\right)$$

$$\frac{7\,\delta}{16}$$

$$-\frac{1}{8}\,\delta\,\left(-3+\delta^2\right)$$

$$-\frac{3\,\delta}{16}$$

Analyzing data

Results of equality tests for Obesity

```
test2.norm[obesity]
testbinomial[test2, obesity, 0]
test1.norm[obesity]
testbinomial[test1, obesity, 0]
```

$\frac{13}{235}$

```
0.998231
```

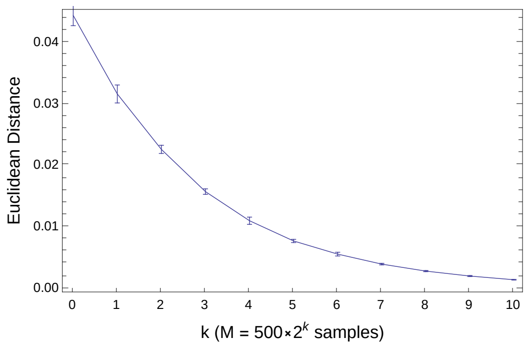
$-\frac{6}{235}$

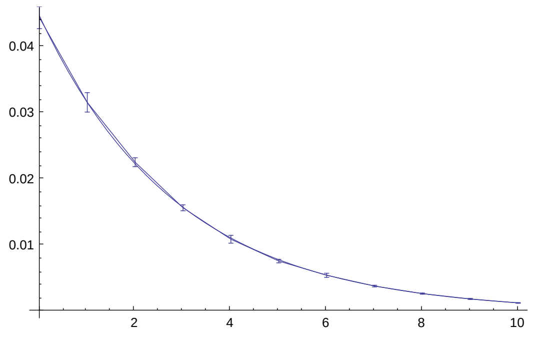
```
0.156967
```

How big are errors from empirical distribution estimation?

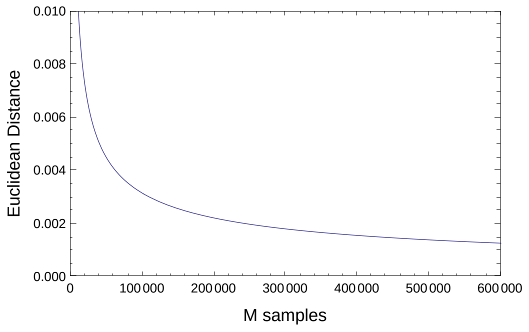
```
(*Draw empirical distributions from a completely mixed distribution*)
Needs["ErrorBarPlots`"]
random[M_] := BinCounts[RandomInteger[255, M], {0, 256, 1}]
stats[M_, x_] := {{M, Mean[x]}, ErrorBar[StandardDeviation[x]]}
ms = Table[k, {k, 0, 10}];
test = Table[ Join[stats[k,
     Table[Norm[norm[random[500 * 2^k]] - Table[1 / 256, {256}]], {i, 10}]]], {k, ms}];
```

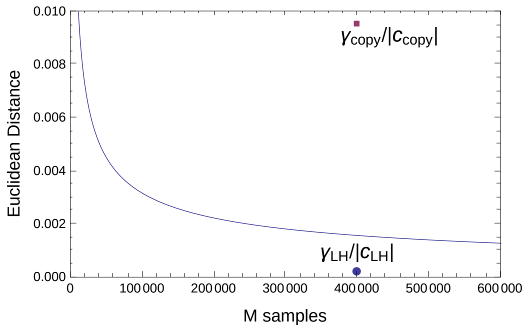
```
ErrorListPlot[test, Joined → True, Frame → True,
 FrameTicks → {{Automatic, Automatic}, {ms, None}},
 FrameLabel → {Style["k (M = 500×2^k samples)", 12, FontFamily → "Arial"],
   Style["Euclidean Distance", 12, FontFamily → "Arial"]}]
Show[Plot[1 / √(500 × 2^k), {k, 0, 10}], %]
```





```
plot2 = Graphics[{FontSize → 14,
    FontFamily → "Helvetica", Text["γcopy/|ccopy|", {445 000, 0.009}]}];
plot3 = Graphics[{FontSize → 14, FontFamily → "Helvetica",
    Text["γLH/|cLH|", {400 000, 0.0009}]}];
plot4 = Graphics[{FontSize → 14, FontFamily → "Helvetica",
    Text["euc(M)", {180 000, 0.0035}]}];
plot6 = Plot[1 / √M, {M, 0, 600 000}, PlotRange → {{0, 600 000}, {0, 0.01}},
  Frame → True, FrameLabel → {Style["M samples", 12, FontFamily → "Arial"],
    Style["Euclidean Distance", 12, FontFamily → "Arial"]}]

plot5 = Show[plot6, ListPlot[{{{400 000, 0.00022}}, {{400 000, 0.0095}}},
   PlotMarkers → Automatic], plot2, plot3]
```





```
Export["deviation.pdf", plot5]
deviation.pdf
```

Check max violation for FHS data with T=4,maxd=8

```
(*For each attribute, link,
calc the violation. Then check confidence that the violation is greater than...
 for no link*)
{ϒ, bound, c, lambdas} = LP1[norm[obesity], 4, 8];
Print["ϒ=", ϒ, " ϒ/|c|=", ϒ / Norm[c], " using M=", Total[obesity]]
```

```
ϒ=0.210877 ϒ/|c|=0.0174146 using M=705
```

Analyze ST Data with both types of test

```
{ϒ, bound, c, lambdas} = LP1[norm[stinfluence], 4, 9];
Print["ϒ=", ϒ, " ϒ/|c|=", ϒ / Norm[c], " using M=", Total[stinfluence]]
```

```
ϒ=0.115694 ϒ/|c|=0.00956559 using M=400 000
```

```
{ϒ, bound, c, lambdas} = LP1[norm[sthomophily], 4, 9];
Print["ϒ=", ϒ, " ϒ/|c|=", ϒ / Norm[c], " using M=", Total[sthomophily]]
```

```
Could not find a bound
```

```
{ϒ, bound, c, lambdas} = LP1[norm[sthomophily], 4, 0];
Print["ϒ=", ϒ, " ϒ/|c|=", ϒ / Norm[c], " using M=", Total[sthomophily]]
```

```
ϒ=0.0023775 ϒ/|c|=0.000226686 using M=400 000
```

```
{ϒ, bound, c, lambdas} = LP1[norm[sthomophily], 4, 8, "RevisedSimplex"];
Print["ϒ=", ϒ, " ϒ/|c|=", ϒ / Norm[c], " using M=", Total[sthomophily]]
```

```
$Aborted
```

```
ϒ=0.0023775 ϒ/|c|=0.000226686 using M=400 000
```

```
Print["Influence, copying model in ST"]
test1.norm[stinfluence] (*deviation*)
test1.stinfluence (*heads+tails*)
Abs[test1].stinfluence (*heads - tails*)
testbinomial[test1, stinfluence, 0] (*confidence using binomial dist*)

test2.norm[stinfluence]
test2.stinfluence
Abs[test2].stinfluence
1 - testbinomial[test2, stinfluence, 0]

Print["Latent homophily model in ST"]
test1.norm[sthomophily]
test1.sthomophily
Abs[test1].sthomophily
testbinomial[test1, sthomophily, 0]

test2.norm[sthomophily]
test2.sthomophily
Abs[test2].sthomophily
testbinomial[test2, sthomophily, 0]
```

```
Influence, copying model in ST
```

$-\frac{387}{50000}$

-3096

```
121 044

0.
```

$\frac{12431}{200000}$

```
24 862

44 726
```

$1.633876631902287 \times 10^{-3181}$

```
Latent homophily model in ST
```

$\frac{27}{80000}$

```
135

39 877

0.7489
```

$\frac{3}{200000}$

```
6

2992

0.536415
```

B Code for Correlation Explanation (CorEx)

This code resulted from [5] and is maintained at `http://www.github.com/gregversteeg/corex`.

```
"""Correlation Explanation

Greg Ver Steeg and Aram Galstyan. "Discovering Structure in
High-Dimensional Data Through Correlation Explanation."
NIPS, 2014. arXiv preprint arXiv:1406.1222.

Code below written by:
Greg Ver Steeg (gregv@isi.edu)
and Gabriel Pereyra

License: GPL2
"""

import numpy as np # Tested with 1.8.0
from scipy.misc import logsumexp # Tested with 0.13.0

class Corex(object):
    """
    Correlation Explanation

    A method to learn a hierarchy of successively more abstract
    representations of complex data that are maximally
    informative about the data. This method is unsupervised,
    requires no assumptions about the data-generating model,
    and scales linearly with the number of variables.

    Greg Ver Steeg and Aram Galstyan. "Discovering Structure in
    High-Dimensional Data Through Correlation Explanation."
    NIPS, 2014. arXiv preprint arXiv:1406.1222.

    Code follows sklearn naming/style (e.g. fit(X) to train)

    Parameters
    ----------
    n_hidden : int, optional, default=2
        Number of hidden units.

    dim_hidden : int, optional, default=2
        Each hidden unit can take dim_hidden discrete values.

    alpha_hyper : tuple, optional
    A tuple of three numbers representing hyper-parameters
    of the algorithm. See NIPS paper for meaning.
    Not extensively tested, but problem-specific tuning
    does not seem necessary.

max_iter : int, optional
    Maximum number of iterations before ending.

batch_size : int, optional
    Number of examples per minibatch. NOT IMPLEMENTED IN THIS VERSION.

n_repeat : int, optional
    Repeat several times and take solution with highest TC.
    NOT IMPLEMENTED IN THIS VERSION. (But a good thing to do.)

verbose : int, optional
    The verbosity level. The default, zero, means silent mode. 1 outputs TC(X;
        Y) as you go
    2 output alpha matrix and MIs as you go.

seed : integer or numpy.RandomState, optional
    A random number generator instance to define the state of the
    random permutations generator. If an integer is given, it fixes the
    seed. Defaults to the global numpy random number generator.

Attributes
----------
labels : array, [n_hidden, n_samples]
    Label for each hidden unit for each sample.

clusters : array, [n_variables]
    Cluster label for each input variable.

p_y_given_x : array, [n_hidden, n_samples, dim_hidden]
    The distribution of latent factors for each sample.

alpha : array-like, shape (n_components,)
    Adjacency matrix between input variables and hidden units. In range [0,1].

mis : array, [n_hidden, n_variables]
    Mutual information between each variable and hidden unit

tcs : array, [n_hidden]
    TC(X_Gj;Y_j) for each hidden unit

tc : float
```

```
    Convenience variable = Sum_j tcs[j]

tc_history : array
    Shows value of TC over the course of learning. Hopefully, it is converging
        .




References
----------

[1]     Greg Ver Steeg and Aram Galstyan. "Discovering Structure in
        High-Dimensional Data Through Correlation Explanation."
        NIPS, 2014. arXiv preprint arXiv:1406.1222.


"""
def __init__(self, n_hidden=2, dim_hidden=2,      # Size of representations
            batch_size=1e6, max_iter=400, n_repeat=1, # Computational limits
            eps=1e-6, alpha_hyper=(0.3, 1., 500.), balance=0., # Parameters
            missing_values=-1, seed=None, verbose=False):

    self.dim_hidden = dim_hidden # Each hidden factor can take dim_hidden
        discrete values
    self.n_hidden = n_hidden # Number of hidden factors to use (Y_1,...Y_m) in
         paper
    self.missing_values = missing_values # Implies the value for this variable
         for this sample is unknown

    self.max_iter = max_iter # Maximum number of updates to run, regardless of
         convergence
    self.batch_size = batch_size # TODO: re-implement running with mini-
        batches
    self.n_repeat = n_repeat # TODO: Run multiple times and take solution with
         largest TC

    self.eps = eps # Change in TC to signal convergence
    self.lam, self.tmin, self.ttc = alpha_hyper # Hyper-parameters for
        updating alpha
    self.balance = balance # 0 implies no balance constraint. Values between 0
         and 1 are valid.

    np.random.seed(seed) # Set for deterministic results
    self.verbose = verbose
    if verbose > 0:
        np.set_printoptions(precision=3, suppress=True, linewidth=200)
        print 'corex, rep size:', n_hidden, dim_hidden
    if verbose > 1:
        np.seterr(all='warn')
    else:
        np.seterr(all='ignore')

def label(self, p_y_given_x):
    """Maximum likelihood labels for some distribution over y's"""
    return np.argmax(p_y_given_x, axis=2).T

@property
def labels(self):
    """Maximum likelihood labels for training data. Can access with self.
        labels (no parens needed)"""
    return self.label(self.p_y_given_x)

@property
def clusters(self):
    """Return cluster labels for variables"""
    return np.argmax(self.alpha[:,:,0],axis=0)

@property
def tc(self):
    """The total correlation explained by all the Y's.
    (Currently correct only for trees, modify for non-trees later.)"""
    return np.sum(self.tcs)

def event_from_sample(self, x):
    """Transform data into event format.
    For each variable, for each possible value of dim_visible it could take (
        an event),
    we return a boolean matrix of True/False if this event occurred in this
        sample, x.
    Parameters:
    x: {array-like}, shape = [n_visible]
    Returns:
    x_event: {array-like}, shape = [n_visible * self.dim_visible]
    """
    x = np.asarray(x)
    n_visible = x.shape[0]
    assert self.n_visible == n_visible, \
        "Incorrect dimensionality for samples to transform."
    return np.ravel(x[:, np.newaxis] == np.tile(np.arange(self.dim_visible), (
        n_visible, 1)))
```

```
def events_from_samples(self, X):
    """Transform data into event format. See event_from_sample docstring."""
    n_samples, n_visible = X.shape
    events_to_transform = np.empty((self.n_events, n_samples), dtype=bool)
    for l, x in enumerate(X):
        events_to_transform[:, l] = self.event_from_sample(x)
    return events_to_transform

def transform(self, X, details=False):
    """
    Label hidden factors for (possibly previously unseen) samples of data.
    Parameters: samples of data, X, shape = [n_samples, n_visible]
    Returns: , shape = [n_samples, n_hidden]
    """
    if X.ndim < 2:
        X = X[np.newaxis, :]
    events_to_transform = self.events_from_samples(X)
    p_y_given_x, log_z = self.calculate_latent(events_to_transform)
    if details:
        return p_y_given_x, log_z
    else:
        return self.label(p_y_given_x)

def fit(self, X, **params):
    """Fit CorEx on the data X.

    Parameters
    ----------

    X: {array-like, sparse matrix}, shape = [n_samples, n_visible]
        Data matrix to be

    Returns
    -------
    self
    """
    self.fit_transform(X)
    return self

def fit_transform(self, X):
    """Fit corex on the data (this used to be ucorex)

    Parameters
    ----------
    X : array-like, shape = [n_samples, n_visible]
        The data.
    Returns
    -------
    Y: array-like, shape = [n_samples, n_hidden]
       Learned values for each latent factor for each sample.
       Y's are sorted so that Y_1 explains most correlation, etc.

    """

    self.initialize_parameters(X)

    X_event = self.events_from_samples(X) # Work with transformed
        representation of data for efficiency

    self.p_x, self.entropy_x = self.data_statistics(X_event)

    for nloop in range(self.max_iter):

        self.update_marginals(X_event, self.p_y_given_x) # Eq. 8

        if self.n_hidden > 1: # Structure learning step
            self.mis = self.calculate_mis(self.log_p_y, self.log_marg)
            self.update_alpha(self.mis, self.tcs) # Eq. 9

        self.p_y_given_x, self.log_z = self.calculate_latent(X_event) # Eq. 7

        self.update_tc(self.log_z) # Calculate TC and record history for
            convergence

        self.print_verbose()
        if self.convergence(): break

    self.sort_and_output()

    return self.labels

def initialize_parameters(self, X):
    """Set up starting state

    Parameters
    ----------
    X : array-like, shape = [n_samples, n_visible]
        The data.

    """
```

```
        self.n_samples, self.n_visible = X.shape
        self.initialize_events(X)
        self.initialize_representation()

    def initialize_events(self, X):
        values_in_data = set(np.unique(X).tolist())-set([self.missing_values])
        self.dim_visible = int(max(values_in_data)) + 1
        if not set(range(self.dim_visible)) == values_in_data:
            print "Warning: Data matrix values should be consecutive integers
                starting with 0,1,..."
        self.n_events = self.n_visible * self.dim_visible

    def initialize_representation(self):
        if self.n_hidden > 1:
            self.alpha = (0.5+0.5*np.random.random((self.n_hidden, self.n_visible,
                1)))
        else:
            self.alpha = np.ones((self.n_hidden, self.n_visible, 1), dtype=float)
        self.tc_history = []
        self.tcs = np.zeros(self.n_hidden)

        log_p_y_given_x_unnorm = -np.log(self.dim_hidden) * (0.5 + np.random.
            random((self.n_hidden, self.n_samples, self.dim_hidden)))
        #log_p_y_given_x_unnorm = -100.*np.random.randint(0,2,(self.n_hidden, self
            .n_samples, self.dim_hidden))
        self.p_y_given_x, self.log_z = self.normalize_latent(
            log_p_y_given_x_unnorm)

    def data_statistics(self, X_event):
        p_x = np.sum(X_event, axis=1).astype(float)
        p_x = p_x.reshape((self.n_visible, self.dim_visible))
        p_x /= np.sum(p_x, axis=1, keepdims=True) # With missing values, each x_i
            may not appear n_samples times
        entropy_x = np.sum(np.where(p_x>0., -p_x * np.log(p_x), 0), axis=1)
        entropy_x = np.where(entropy_x > 0, entropy_x, 1e-10)
        return p_x, entropy_x

    def update_marginals(self, X_event, p_y_given_x):
        self.log_p_y = self.calculate_p_y(p_y_given_x)
        self.log_marg = self.calculate_p_y_xi(X_event, p_y_given_x) - self.log_p_y

    def calculate_p_y(self, p_y_given_x):
        """Estimate log p(y_j) using a tiny bit of Laplace smoothing to avoid
            infinities."""
        pseudo_counts = 0.001 + np.sum(p_y_given_x, axis=1, keepdims=True)
        log_p_y = np.log(pseudo_counts) - np.log(np.sum(pseudo_counts, axis=2,
            keepdims=True))
        return log_p_y

    def calculate_p_y_xi(self, X_event, p_y_given_x):
        """Estimate log p(y_j|x_i) using a tiny bit of Laplace smoothing to avoid
            infinities."""
        pseudo_counts = 0.001 + np.dot(X_event, p_y_given_x).transpose((1,0,2)) #
            n_hidden, n_events, dim_hidden
        log_marg = np.log(pseudo_counts) - np.log(np.sum(pseudo_counts, axis=2,
            keepdims=True))
        return log_marg # May be better to calc log p(x_i|y_j)/p(x_i), as we do in
             Marg_Corex

    def calculate_mis(self, log_p_y, log_marg):
        """Return normalized mutual information"""
        vec = np.exp(log_marg + log_p_y) # p(y_j|x_i)
        smis = np.sum(vec * log_marg, axis=2)
        smis = smis.reshape((self.n_hidden, self.n_visible, self.dim_visible))
        mis = np.sum(smis * self.p_x, axis=2, keepdims=True)
        return mis/self.entropy_x.reshape((1, -1, 1))

    def update_alpha(self, mis, tcs):
        t = (self.tmin + self.ttc * np.abs(tcs)).reshape((self.n_hidden, 1, 1))
        maxmis = np.max(mis, axis=0)
        alphaopt = np.exp(t * (mis - maxmis))
        self.alpha = (1. - self.lam) * self.alpha + self.lam * alphaopt

    def calculate_latent(self, X_event):
        """"Calculate the probability distribution for hidden factors for each
            sample."""
        alpha_rep = np.repeat(self.alpha, self.dim_visible, axis=1)
        log_p_y_given_x_unnorm = (1. - self.balance) * self.log_p_y + np.transpose
            (np.dot(X_event.T, alpha_rep*self.log_marg), (1, 0, 2))

        return self.normalize_latent(log_p_y_given_x_unnorm)

    def normalize_latent(self, log_p_y_given_x_unnorm):
        """Normalize the latent variable distribution

        For each sample in the training set, we estimate a probability
            distribution
        over y_j, each hidden factor. Here we normalize it. (Eq. 7 in paper.)
        This normalization factor is quite useful as described in upcoming work.

        Parameters
```

```
        ----------
        Unnormalized distribution of hidden factors for each training sample.

        Returns
        -------
        p_y_given_x : 3D array, shape (n_hidden, n_samples, dim_hidden)
            p(y_j|x^l), the probability distribution over all hidden factors,
            for data samples l = 1...n_samples
        log_z : 2D array, shape (n_hidden, n_samples)
            Point-wise estimate of total correlation explained by each Y_j for each
                 sample,
            used to estimate overall total correlation.

        """

        log_z = logsumexp(log_p_y_given_x_unnorm, axis=2) # Essential to maintain
            precision.
        log_z = log_z.reshape((self.n_hidden, -1, 1))

        return np.exp(log_p_y_given_x_unnorm - log_z), log_z

    def update_tc(self, log_z):
        self.tcs = np.mean(log_z, axis=1).reshape(-1)
        self.tc_history.append(np.sum(self.tcs))

    def sort_and_output(self):
        order = np.argsort(self.tcs)[::-1] # Order components from strongest TC to
             weakest
        self.tcs = self.tcs[order] # TC for each component
        self.alpha = self.alpha[order] # Connections between X_i and Y_j
        self.p_y_given_x = self.p_y_given_x[order] # Probabilistic labels for each
             sample
        self.log_marg = self.log_marg[order] # Parameters defining the
            representation
        self.log_p_y = self.log_p_y[order] # Parameters defining the
            representation
        self.log_z = self.log_z[order] # -log_z can be interpreted as "surprise"
            for each sample
        if hasattr(self, 'mis'):
            self.mis = self.mis[order]

    def print_verbose(self):
        if self.verbose:
            print self.tcs
        if self.verbose > 1:
            print self.alpha[:,:,0]
        if hasattr(self, "mis"):
            print self.mis[:,:,0]

def convergence(self):
    dist = -np.mean(self.tc_history[-10:-5]) + np.mean(self.tc_history[-5:])
    return np.abs(dist) < self.eps # Check for convergence. dist is nan for
        empty arrays, but that's OK
```

1.

1. Report Type

Final Report

Primary Contact E-mail
Contact email if there is a problem with the report.

gregv@isi.edu

Primary Contact Phone Number
Contact phone number if there is a problem with the report

626-840-5901

Organization / Institution name

Information Sciences Institute, USC

Grant/Contract Title
The full title of the funded effort.

Bell Inequalities for Complex Networks

Grant/Contract Number
AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-12-1-0417

Principal Investigator Name
The full name of the principal investigator on the grant or contract.

Greg Ver Steeg

Program Manager
The AFOSR Program Manager currently assigned to the award

James Lawton

Reporting Period Start Date

08/01/2012

Reporting Period End Date

07/31/2015

Abstract

This effort studied new methods to understand the effect of hidden variables af- fecting complex systems. Bell inequalities are a famous example of a hidden variable test in quantum physics that provides the strongest evidence for that theory. Initial work in this project extended the mathematical formulations of Bell inequalities to design new hidden variable tests that were able to account for confounding effects in complex systems including human social networks. These tests solved an open ques- tion about the identifiability of contagion in social network studies. Subsequent work moved beyond identification of hidden variables to develop a new information-theoretic framework capable of reconstructing hidden variables explaining the multivariate de- pendencies in complex systems. These methods have demonstrated value on diverse problems including human behavior, language, neuroscience, and gene expression.

Distribution Statement
This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement
If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF The maximum file size for an SF298 is 50MB.

sf298_versteeg.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

afosr_versteeg.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

• Greg Ver Steeg and Aram Galstyan. Statistical tests for contagion in observational social network studies. In Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS), 2013.
• Greg Ver Steeg and Aram Galstyan. Discovering structure in high-dimensional data through correlation explanation. In Advances in Neural Information Processing Sys- tems (NIPS), 2014.
• Greg Ver Steeg and Aram Galstyan. Maximally informative hierarchical representa- tions of high-dimensional data. In Proceedings of the Sixteenth International Con- ference on Artificial Intelligence and Statistics (AISTATS), 2015.
• Greg Ver Steeg and Aram Galstyan. The Information Sieve. arXiv:1507.02284, 2015.

Changes in research objectives (if any):

The original goal of the project was to develop tests for the existence of hidden variables affecting complex systems. This evolved in year two to encompass a new approach to reconstructing these hidden factors based on information-theoretic principles.

Change in AFOSR Program Manager, if any:

Original PM was Robert Bonneau. James Lawton is still temporarily administering the complex networks program as far as I know.

Extensions granted or milestones slipped, if any:

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Oct 23, 2015 14:23:32 Success: Email Sent to: gregv@isi.edu