

Semi-Supervised Learning for Electronic Phenotyping in Support of Precision Medicine

by

Yonatan Halpern

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science
New York University
September 2016

David Sontag

Dedication

To Dani, my favourite collaborator.

Acknowledgements

I am incredibly thankful to everyone who made this possible.

Thank you to my advisor, David Sontag, for his boundless energy, continued encouragement, optimism and patience, throughout this entire journey.

To my many teachers from whom I have benefitted tremendously over the years. Kevin Santosuosso, Steve Engels, Rich Zemel, Tim Barfoot, Mike Brudno, and Margaret White. I chose this field of study, because you showed me, all in different ways, how fascinating it can be.

To my co-authors and collaborators, I couldn't have done any of this on my own. I appreciate that you welcomed me into the community of researchers.

To the Emergency Informatics team at Beth Israel Deaconess Medical Center, and particularly Dr Steven Horng, thanks for your wisdom and your support as we try to help doctors; not replace them.

To my fellow students, for their fun and stimulating conversations, school-related and otherwise. I learned a lot from each of you and continue to learn more every day. I am glad we spent this time together: Yacine Jernite, Narges Razavian, Uri Shalit, Rachel Hodos, Rahul Krishnan, Jonathan Tompson, Nathan Silberman, David Eigen, and Ross Goroshin.

To my students and advisees, from whom I learned the most: Maya Rotmensch, Youngduck Choi, Ravi Kiran. It was great working with all of you.

Thanks to the National Research and Engineering Council of Canada (NSERC) for their generous financial support.

Last, but certainly not least, my family. To the Lent family, for all your love, support, and babysitting in the big city of New York. In answer to your repeated questions of “what

are you studying at Columbia these days?” , my answer is: This. And for the last time, I've been at NYU. To my brothers, Zvi and Moshe, for stimulating conversations about nonsense. To my parents, who, with love and pride have always encouraged my education. Now you have the doctor son you always wanted. Well, sort of.

Abstract

Medical informatics plays an important role in precision medicine, delivering the right information to the right person, at the right time. With the introduction and widespread adoption of electronic medical records, in the United States and world-wide, there is now a tremendous amount of health data available for analysis.

Electronic record phenotyping refers to the task of determining, from an electronic medical record entry, a concise descriptor of the patient, comprising of their medical history, current problems, presentation, etc. In inferring such a phenotype descriptor from the record, a computer, in a sense, “understands” the relevant parts of the record. These phenotypes can then be used in downstream applications such as cohort selection for retrospective studies, real-time clinical decision support, contextual displays, intelligent search, and precise alerting mechanisms.

We are faced with three main challenges:

First, the unstructured and incomplete nature of the data recorded in the electronic medical records requires special attention. Relevant information can be missing or written in an obscure way that the computer does not understand.

Second, the scale of the data makes it important to develop efficient methods at all steps of the machine learning pipeline, including data collection and labeling, model learning and inference.

Third, large parts of medicine are well understood by health professionals. How do we combine the expert knowledge of specialists with the statistical insights from the electronic medical record?

Probabilistic graphical models such as Bayesian networks provide a useful abstraction for

quantifying uncertainty and describing complex dependencies in data. Although significant progress has been made over the last decade on approximate inference algorithms and structure learning from complete data, learning models with incomplete data remains one of machine learning's most challenging problems. How can we model the effects of latent variables that are not directly observed?

The first part of the thesis presents two different structural conditions under which learning with latent variables is computationally tractable. The first is the “anchored” condition, where every latent variable has at least one child that is not shared by any other parent. The second is the “singly-coupled” condition, where every latent variable is connected to at least three children that satisfy conditional independence (possibly after transforming the data).

Variables that satisfy these conditions can be specified by an expert without requiring that the entire structure or its parameters be specified, allowing for effective use of human expertise and making room for statistical learning to do some of the heavy lifting. For both the anchored and singly-coupled conditions, practical algorithms are presented.

The second part of the thesis describes real-life applications using the anchored condition for electronic phenotyping. A human-in-the-loop learning system and a functioning emergency informatics system for real-time extraction of important clinical variables are described and evaluated.

The algorithms and discussion presented here were developed for the purpose of improving healthcare, but are much more widely applicable, dealing with the very basic questions of identifiability and learning models with latent variables - a problem that lies at the very heart of the natural and social sciences.

Contents

Dedication	iii
Acknowledgements	iv
Abstract	vi
List of Figures	xi
List of Tables	xxiii
List of Appendices	xxvii
1 Introduction	1
1.1 Building a “smart” EMR system	1
1.2 Machine learning background	6
1.3 Themes	13
1.4 Outline and Contributions	19
2 Phenotype classifiers learned with anchors	21
2.1 Introduction	21
2.2 Anchor and Learn Framework	25
2.3 Evaluation: Phenotype estimation in the emergency department	36
2.4 Results	41
2.5 Discussion	49
2.6 Conclusions	56
2.7 Next steps and open questions	57

3	Semi-supervised learning of joint probabilistic models	61
3.1	Introduction	61
3.2	Anchored factor analysis	64
3.3	Ungrounded latent variables	66
3.4	Semi-supervised training	69
3.5	Evaluation framework	74
3.6	Results	79
3.7	Next steps and open questions	80
4	Learning noisy-or networks with singly-coupled observations	84
4.1	Introduction	84
4.2	Singly-coupled tuples in noisy-or diagnosis networks	87
4.3	Parameter recovery – known structure	93
4.4	Structure learning – discovering latent variables	100
4.5	Experiments	107
4.6	Related work	115
4.7	Conclusions	117
4.8	Next steps and open questions	117
5	Learning a health knowledge graph from electronic medical records	122
5.1	Introduction	122
5.2	Constructing a knowledge graph	126
5.3	Experimental methods	132
5.4	Results	137
5.5	Discussion	141
5.6	Conclusions	147
5.7	Next steps and open questions	148

6	Correlated factors in anchored factor analysis	152
6.1	Introduction	152
6.2	Model definition: Anchored factor analysis	155
6.3	Recovering moments of latent variables	156
6.4	Method of moments learning	160
6.5	Model evaluation	166
6.6	Discussion	174
6.7	Conclusions	177
6.8	Next steps and open questions	178
7	Topic modeling with anchors	181
7.1	Introduction	181
7.2	Anchor words	186
7.3	The anchor words algorithm	187
7.4	Experimental Results	196
7.5	Discussion	205
7.6	Conclusions	209
7.7	Next steps and open questions	209
8	Closing	211
8.1	Looking forward	212
8.2	Open questions	215
8.3	Conclusion	216
	Appendices	217
	Bibliography	303

List of Figures

1.1	Electronic medical record viewer (dashboard) at Beth Israel Deaconess Medical Center.	2
1.2	Example of a context-specific displays possible in a smart EMR. Left: an order set appears for patients who the dashboard recognizes as having “chest pain”. Right: smart autocomplete of chief complaints described implemented in the dashboard as described in Jernite et al. (2013a).	4
1.3	Example of a phenotype extractor learned to identify patients from nursing homes (source: Halpern et al. (2014)) The anchors are a small number of phrases manually specified by a clinical collaborator. The rest of the classifier is learned from medical records. Each word is associated with a weight. Highly indicative words are displayed.	7
1.4	Bayesian networks developed for an ICU alarm system (left) and diagnosis liver diseases (right). Sources: Koller and Friedman (2009); Onisko et al. (1999)	8
1.5	A simplified schematic of the QMR-DT network (Miller et al., 1986; Shwe et al., 1991) structure for medical diagnosis. The full QMR-DT network models the relationships between 570 diseases and 4075 symptoms.	9
1.6	The method of moments attempts to invert the mapping between moments of the distributions (e.g., probability of simultaneous negative results) and the parameters of the model (e.g., a noisy-or parametrization with priors, failures, and leak as described in Section 1.2.3).	13

1.7	Identifiable parameters (X) map to a unique setting of the low orders. Parameters may be non-identifiable (O) if multiple parameter settings lead to the same low order moments.	14
1.8	An example of a manually specified phenotyping algorithm for autism spectrum disorders. Source: Lingren (2013)	15
2.1	A schematic flow of how diverse clinical observations, both structured and unstructured, can be distilled into relevant clinical state variables (phenotypes), and used for applications.	22
2.2	Adding a subtracting anchors: Changes in the area under the receiver operator curve (AUC) as a physician user adds and subtracts anchors to train a classifier for cardiac etiology. The blue solid line shows the performance of the learned classifiers. The green dotted line shows the performance of a naive prediction strategy that simply looks for anchors. Gold standard labels are obtained from questions asked to physicians at disposition time (Section 2.3.4).	35
2.3	A screenshot of the anchor elicitation tool using deidentified patient information. Conditional highlighting emphasizes the presence of previously specified anchors in a note so that the physician can determine quickly whether its usage is as expected.	36
2.4	Comparison of performance of phenotypes learned with 200,000 unlabeled patients using the semi-supervised anchor based method, and phenotypes learned with supervised classification using 5,000 gold standard labels. Error bars indicate 2 * standard error. “Insufficient labels” means that we did not have enough labeled patients to train the fully supervised baseline. “Rules < 80” means that the rules baseline was consistently below 80 so it does not appear in these plots.	44

2.5	Changes to patient records over time. The time of every change to the patient record is recorded (measured in minutes from arrival) and a non-parametric kernel density estimator is used to plot the distribution of times at which changes occur.	45
2.6	Influence and highly changing features for the pneumonia phenotype extractor as a function of time.	46
2.7	Precision-recall curves calculated for each of the phenotypes at triage time, 1 hour after triage, and disposition. Circles indicate operating characteristics achievable by thresholding on the number of anchors present.	47
2.8	Additive change in AUC from baseline for phenotype extraction as a function of the features used. The baseline phenotype extraction uses only features from Age, Sex and Triage vitals and its value is indicated for each phenotype on the y-axis label. Blue bars indicate structured data while red bars indicate free-text data. Hatched lines represent a combination of features. A star is placed below the single feature that has the highest performance. From left to right, the classifiers use: Med Medication history (prior to visit) Pyx Medication dispensing record (during visit) Lab Laboratory values Strct All Structured data (Med + Pyx + Labs) Tri Triage Nursing Text MD Physician Comments Txt All Text (Tri + MD) All All features (Structured + Text).	50

2.9	Trellis view of predictive accuracy based on subsets of a patient’s record being present. Each node represents a setting where only a subset of the data is available (all other data types are dropped synthetically). Directed edges represent the addition of a new data type. Two values are recorded: (left) The AUC for predictions from a “subset classifier” (Section 2.5.4); (right) the AUC for predictions trained on the full feature set. The data types are as follows: Demographics - Age, sex Triage - Age, sex, vitals, triage assessment labs - laboratory results med - medication history (prior to visit) pyx - medication dispensing record (during visit) md - physician comments.	55
3.1	The joint probabilistic model used for clinical tagging is a bipartite graph involving conditions and observations, like QMR-DT (Shwe et al., 1991) with one or more <i>anchor</i> (red outline) for each condition (only one anchor per condition is shown in the illustration). Other observations (black outline) can have multiple parents. The anchors from Halpern et al. (2016) are available on github: https://github.com/clinicalml/clinical-anchors	65
3.2	(Top) Effect of pure likelihood-based optimization on accuracy on a tag. A likelihood optimization procedure is initialized with the parameters of the anchor-based method-of-moments learning algorithm (ADFA-tree; described in Chapter 6). As the likelihood optimization progresses, accuracy at predicting phenotypes degrades. (Bottom) The latent variable associated with the “headache” anchor changes meaning over the course of optimization.	68
3.3	Tagging system within an electronic medical record.	76
4.1	The QMR-DT network is a Bayesian network for medical diagnosis. Latent diseases are connected to observed symptoms through noisy-or edges.	88

4.2	Examples of singly and non-singly coupled triplets. (a,b) are singly coupled. (c,d) are not singly coupled as they violate conditions 1 and 2 respectively.	89
4.3	The distribution of observations from a three view mixture model with a binary hidden state Y can be decomposed into two rank-1 tensors.	90
4.4	Transforming between tensor moments and negative moments	97
4.5	An example network to be learned. In the first step, we identify A, B, C as a singly-coupled triplet, coupled by X and learn its parameters. In the second step, we use the extend step to learn the edges from X to its other children (D, E, F). In the third step, we remove X from the network, using the subtracting off step to remove its influence from low order moments. After subtracting off, Y has a singly coupled triplet (C, D, F) and then we repeat the cycle again.	98
4.6	Two networks with the same moments matrix using parameters: $p_X = 0.2$, $p_X = 0.3$, $p_Z = 0.37$. $f_X = (0.1, 0.2, 0.3)$, $f_Y = (0.6, 0.4, 0.5)$, $f_Z = (0.28, 0.23, 0.33)$. One is singly-coupled, the other is not. High precision values for these parameters are given in Appendix D.9,	101
4.7	Schematic of the unfolded rank test. Samples are used to form a $2 \times 2 \times 2 \times 2$ tensor, which is unfolded to perform the unfolded rank test.	102
4.8	Probability distribution of the third eigenvalue for a quartet with coupling parents, when all parameters are drawn uniformly at random. Blue: two parents, same failure probabilities. Green: three parents, same failure probabilities. Red: two parents, different failure probabilities. Black: three parents, different failure probabilities. This plot illustrates that rank testability only gets easier for random parameters compared to the uniform parameter setting used in the experiments.	104

4.9	Simple network used in synthetic experiments. To learn, first estimate the parameters of Y_0 using the singly-coupled triplet $\{X_0, X_1, X_2\}$, then learn Y_1 whose children are singly-coupled after subtracting off Y_0	108
4.10	(left) Parameter error (L1) for the simple network as a function of number of samples. The uniform strategy estimates 0.5 for every failure probability. (right) comparison of running time between variational learning and method of moments.	109
4.11	(Top row) Synthetic image “sources”. The first and last sources (rhombus and filled square) cannot be learned without subtracting off the other sources. (Bottom row) Samples drawn from the network.	110
4.12	Recovery of the synthetic images using the proposed quartet test method for structure learning and parameter estimation. Each row represents sources learned from a different number of sources. The shaded box underneath each source represents the prior probability that the model assigns to that source.	111
4.13	Recovery of the synthetic images using variational EM with 8 random restarts and allowing for 8, 12, and 16 sources. Each row represents sources learned from a different number of sources. The shaded box underneath each source represents the prior probability that the model assigns to that source. None of the runs in the last row with 8 sources (N=10000) finished in the allotted 1 hour time limit.	112
4.14	Recovery of the synthetic images using variational stochastic gradient ascent with 8 random restarts and allowing for 8, 12, and 16 sources. Each row represents sources learned from a different number of sources. The shaded box underneath each source represents the prior probability that the model assigns to that source. Training was run for 5 minutes.	113

4.15	Mean parameter error (L1) in the failure estimates of the QMR-DT network as a function of the number of synthetic samples to learn from. Each line represents edges learned at a different depth (defined in Section 4.3.5). . . .	115
4.16	(left) Empirical identifiability for $n \times m$ complete bipartite graphs. The values in the cells reflect the order of moments required to obtain identifiability. -1 denotes that model is <i>unidentifiable</i> no matter what order of moments are used. (right) Identifiability in random graphs with 2 parents and 6 children as a function of the edge likelihood, p . The <i>singly-coupled</i> method uses Algorithm 3 to test identifiability.	118
5.1	Workflow of modeling the relationship between diseases and symptoms and knowledge graph construction, for each of our 3 models (naive Bayes, logistic regression and noisy or).	127
5.2	Causal graph of diseases and symptoms. Diseases (Y_1, Y_2) can cause symptoms (only one symptom, X_1 , shown here), and can have a high dimensional set of common parents U (risk factors such as age) which make the diseases non-independent. Diseases and symptoms are observed.	130
5.3	Concept extraction pipeline. Non-negated concepts and ICD-9 diagnosis codes are extracted from Emergency Department electronic medical records. Concepts, codes and concept aliases are mapped to unique IDs, which in turn populate a co-occurrence matrix of size (Concepts) x (Patients).	134

5.4	User interface for evaluation by physicians. The left column shows the list of diseases from which clinicians select a disease to be tagged. When a disease is selected, the upper panel provides a more detailed description of the selected disease. The middle column shows the symptoms that have yet to be tagged and hover over functionality provides a detailed symptom definition. The rightmost column shows the 4 categories into which symptoms can be sorted.	136
5.5	Distribution of number of diseases and symptoms per patient record.	137
5.6	Precision-recall curves for two evaluation frameworks. (a) The precision-recall curve for the automatic evaluation evaluated against the Google health knowledge graph. (b) The precision-recall curve rated according to physicians expert opinion. We see that in both graphs, the relative performance of the models is the same.	142
5.7	Comparison of disease frequency for the ‘adult’ age bracket (40-60 years old). The y-axis shows the number of identified diseases in the emergency department data. The x-axis records the expected frequency of diseases according to the Google health knowledge graph for the ‘adult’ age bracket. The points highlighted demonstrate instances of frequency misalignment due to the differences in populations considered.	146
6.1	Illustration of our algorithm for anchored factor analysis with correlated factors. Observed variables X_0, \dots, X_3 are anchors for latent variables Y_0, \dots, Y_3 . First, we recover the low-order moments of the latent variables by solving an optimization problem involving moments of the observed anchors (Section 6.3). Second, we use these moments to learn a Bayesian network describing the joint distribution of the latent variables (Section 6.4.1). Finally, we learn the conditional distributions for the rest of the observations (Section 6.4.2). . . .	154

6.2	<p>Y_i has a direct effect on the distribution of X_j as well as indirect effects that pass through its two neighbors. Correction factors are introduced to cancel the indirect effects through each neighbor, leaving only the direct effect which is modeled with the parameter $f_{i,j}$.</p>	163
6.3	<p>The learned tree structured latent variable model describing the distribution of 23 conditions in the emergency department, learned using marginal polytope constraints. Green lines represent positive correlations and red lines represent negative correlations. The section in the box shows small subgraphs of a more complex structure learned allowing for two parents per variable.</p>	169
6.4	<p>Average accuracy on the last-tag prediction task for 5K held-out instances for Emergency and Stack Overflow datasets. Anchored Factor Analysis (AFA) models are learned using a tree structured model for the latent variables and an independent model. Dotted lines are oracle results from the same model family.</p>	173
6.5	<p>(Left) Per document held-out likelihood of learned tree structured models on Stack Overflow using held-out sets of 10K documents, each marker represents a different random train/test sample for the data. Solid lines represent the average of 8 runs. Different lines represent successively tight outer bounds to the marginal polytope constraints. (Right) When the Stack Overflow data is replaced by synthetic data drawn from a learned model, the difference between the constraints is much less pronounced.</p>	175
7.1	<p>The generative model used in topic modeling.</p>	183

7.2	The rows of Q are vectors in V -dimensions, and their convex hull is a K -simplex whose vertices are anchor rows. Here $Q_i = \frac{1}{2}Q_{\pi(k)} + \frac{1}{2}Q_{\pi(k')}$ and this implies that the posterior distribution $P(z_1 = * w_1 = i)$ assigns $\frac{1}{2}$ to $z_1 = k$ and $\frac{1}{2}$ to $z_1 = k'$ and zero to every other topic.	188
7.3	The first three steps of FindAnchors consist of finding a starting point furthest from the origin, finding the furthest point from the initial point, and finding the furthest point from the line defined by the first two points.	192
7.4	Training time on synthetic NIPS documents.	199
7.5	ℓ_1 error for learning semi-synthetic LDA models with $K = 100$ topics (top : based on NY Times, bottom : based on NIPS abstracts). The horizontal lines indicate the ℓ_1 error of K uniform distributions.	200
7.6	When we add artificial anchor words before generating synthetic documents, ℓ_1 error goes to zero for Recover and close to zero for RecoverL2	201
7.7	ℓ_1 error increases as we increase topic correlation (top : $\rho = 0.05$, bottom : $\rho = 0.1$). Based on the NY Times semi-synthetic model with 100 topics. . . .	202
7.8	Held-out probability (per token) is similar for RecoverL2 and Gibbs sampling. RecoverL2 has better coherence, but fewer unique terms in the top $N = 20$ words than Gibbs. (Up is better for all three metrics.)	204
B.1	A screenshot of the anchor elicitation tool using deidentified patient information. Conditional highlighting emphasizes the presence of previously specified anchors in a note so that the physician can determine quickly whether its usage is as expected.	222

C.1	Improvement of the likelihood-based objective (left) and heldout tag (right) as optimization progresses for different values of L2 regularization (weight decay) in the recognition model. The model chosen by the model selection procedure (Section 3.4.4) is marked with a large red dot. The first 50 epochs are a burn-in period where only the recognition model changes.	233
D.1	Two networks with the same moments matrix using parameters: $p_X = 0.2$, $p_X = 0.3$, $p_Z = 0.37$. $f_X = (0.1, 0.2, 0.3)$, $f_Y = (0.6, 0.4, 0.5)$, $f_Z = (0.28, 0.23, 0.33)$. One is singly-coupled, the other is not.	253
D.2	A network that satisfies the expansion property but has no singly-coupled triplets.	256
D.3	A network that is learnable with singly-coupled triplets but does not satisfy the expansion property.	256
E.1	The causal graph of diseases and symptoms. Diseases (Y_1, Y_2) can cause symptoms (only one symptom, X_1 , shown here), and can have a high dimensional set of common parents U (risk factors such as age) which make the diseases non-independent. Diseases and symptoms are observed.	257
E.2	The causal graph from Figure E.1 after intervening (do), setting $Y_1 = y_1$. . .	258
F.1	Tree model learned for Emergency data. Red and green edges represent positive and negative correlations between the variables, respectively.	278
F.2	Tree model learned for Stack Overflow data. Red and green edges represent positive and negative correlations between the variables, respectively.	279
F.3	Bounded in-degree model (≤ 2) learned for Emergency data. Red and green edges represent positive and negative correlations between the variables, respectively.	280

F.4	Bounded in-degree model (≤ 2) learned for Stack Overflow data. Red and green edges represent positive and negative correlations between the variables, respectively.	281
G.1	Illustration of the Algorithm	283
G.2	Proof of Lemma 15, after projecting to the orthogonal subspace of $\text{span}(S)$	287
G.3	Results for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$	298
G.4	Results for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$, with a synthetic anchor word added to each topic.	298
G.5	Results for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$, with moderate correlation between topics.	299
G.6	Results for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$, with stronger correlation between topics.	299
G.7	Results for a semi-synthetic model generated from a model trained on NIPS papers with $K = 100$. For $D \in \{2000, 6000, 8000\}$, Recover produces log probabilities of $-\infty$ for some held-out documents.	300

List of Tables

2.1	A selection of the 42 phenotypes built as part of this ongoing project. Each phenotype is defined by its anchors, which can be specified as ICD9 codes, medications (history or dispensed), or free text. When a large number of anchors are specified, only a selection are shown. For display, medications are grouped by extended therapeutic class (ETC).	28
2.2	Features used to build binary patient description vectors	37
2.3	Phenotype variables used for evaluation. The text in the Disposition Question column was shown to physicians at the end of patient disposition. The parenthetical text was shown if physicians selected a click-through option for additional information. N gives the number of labels collected while Pos gives the fraction of positively labeled cases.	38
2.4	Top 20 weighted terms in the classifiers for three of the learned phenotypes. These classifiers are learned using medical records as they appear at time of disposition from the emergency department.	43
2.5	Precision and recall of the anchors used for training (includes diagnosis codes).	45
3.1	The full list of phenotypes included in the model. Acute conditions relate to the patient’s current condition. History refers to the patient’s history.	77
3.2	Features extracted . Billing codes were extracted to perform the evaluation, but were not used to create the patient feature vector.	78

3.3	Results for last-tag prediction. Performance measures are Accuracy, Top-5 (correct tag within the top 5) and MRR (mean reciprocal rank). Noisy-or init uses the model with the θ_0 parameters. Noisy-or final shows the result after likelihood optimization.	80
3.4	Highly weighted (low failure probability) words learned by the noisy-or model after likelihood optimization. Words marked neg: are within a negation scope. Some shortforms are present in the text (ed: emergency department, sob: shortness of breath, loc: loss of consciousness, s/p: status post).	80
4.1	Learnability of the QMR-DT network. Diseases and edges learned at lower depth are learned with higher precision.	114
5.1	Top edge suggestions by models for a randomly chosen disease (Middle Ear Infection). The number of shown edges corresponds to the number of edges in the Google health knowledge graph. For logistic regression, naive Bayes, and noisy or, the suggestions are ordered by their relative importance score. For the Google health knowledge graph, the edges are sorted according to two broad buckets of edge frequency that are provided in the graph. The stars associated with each edge represent the expected frequency for which “disease A causes symptom B” as rated by physicians. [*** = always happens, ** = sometimes happens, * = rarely happens, = never happens]	139

5.2	Top edge suggestions by models for the disease “Gallstones”. The number of shown edges corresponds to the number of edges in the Google health knowledge graph. For logistic regression, naive Bayes, and noisy or, the suggestions are ordered by their relative importance score. For the Google health knowledge graph, the edges are sorted according to two broad buckets of edge frequency that are provided in the graph. The stars associated with each edge represent the expected frequency for which “disease A causes symptom B” as rated by physicians. [*** = always happens, ** = sometimes happens, * = rarely happens, = never happens]	140
5.3	Subset of the knowledge graph learned using the noisy or model. For each disease we show the full list of edges along with their corresponding importance score in parentheses. Symptoms are ordered according to importance scores.	143
6.1	Complexity of learning different model classes. After performing the moment-transformations (Section 6.3), the complexity of learning the models with latent variables is no harder than learning with fully observed moments.	165
6.2	Top weighted words for factors in the Stack Overflow and Emergency. Words marked <i>neg:</i> are within a negation scope.	170
7.1	Example topic pairs from NY Times (closest ℓ_1), anchor words in bold. The UCI NY Times corpus includes named-entity annotations, indicated by the <i>zzz</i> prefix. All 100 topics are shown in Appendix G.3.1.	204
D.1	Two settings where the determinant of the moments matrix is non zero Left: $Det(\theta_1) = 5.33 \times 10^{-7}$. Right: $Det(\theta_2) = 4.95 \times 10^{-7}$	250
D.2	Results of a simulation experiment. 10,000 samples were drawn and tested whether they are consistent with a distribution from singly-coupled triplets.	255

F.1 Learned medical concepts. For each concept we display the top 10 weighted factors and one supplied anchor. 270

F.2 Learned concepts from Stack Overflow. For each concept we display the top weighted factors and one supplied anchor. Note that in the Stack Overflow dataset we use a simple rule to provide a single anchor for every latent variable.273

G.1 Example topic pairs from NY Times sorted by ℓ_1 distance, anchor words in bold.294

G.2 Example topic pairs from NY Times sorted by ℓ_1 distance, anchor words in bold.295

G.3 Example topic pairs from NY Times sorted by ℓ_1 distance, anchor words in bold.297

List of Appendices

Appendix to Introduction	218
Appendix to Electronic Phenotyping with Anchors	220
Appendix to Clinical Tagging with Joint Probabilistic Models	226
Appendix to Parameter Learning in Diagnosis Networks	235
Appendix to Learning a Health Knowledge Graph from Electronic Medical Records	257
Appendix to Anchored Factor Analysis	259
Appendix to Topic modeling with anchors	282

Chapter 1

Introduction

1.1 Building a “smart” EMR system

With the introduction and widespread adoption of electronic medical records (EMR), in the United States and world-wide, there is now a tremendous amount of health data available for both retrospective and real-time analysis. The challenge now is to make all of this information useful, delivering the right information to the right person, at the right time. Electronic medical records can serve as both a source of knowledge discovery, information delivery, and action. Observational studies now routinely use information extracted from EMR to define cohorts and measure outcomes (some review articles that document this trend are Dean et al. (2009); Lau et al. (2011); Lin et al. (2013)) Information is delivered and acted upon through EMR dashboards. Figure 1.1 shows the custom built emergency department dashboard at Beth Israel Deaconess Medical Center in Boston, MA. Through this interface, physicians enter and retrieve data about the patient, order labs and medications, and receive alerts and notifications.

The “smart” electronic medical record (EMR) system for precision healthcare is not simply a mechanism to store and retrieve information, rather it is an active participant in

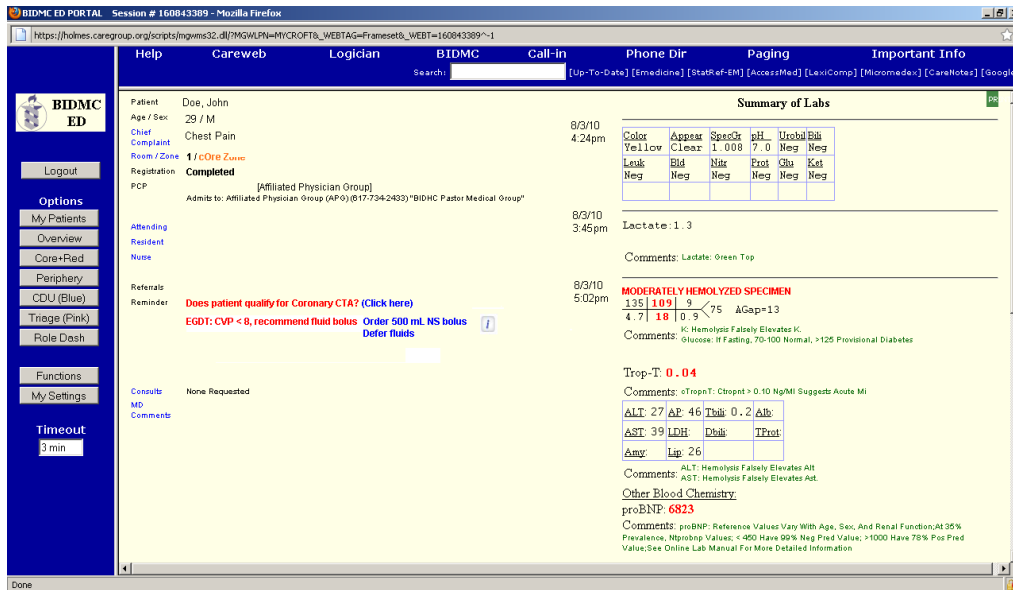


Figure 1.1: Electronic medical record viewer (dashboard) at Beth Israel Deaconess Medical Center.

the patient's care; understanding the patient's condition and needs in order to facilitate care. The EMR system does not seek to replace the physician, or even serve as a standalone informatics consult service (for the drawbacks of this "Greek Oracle" paradigm, see Miller and Masarie (1990)). Rather, it takes its inputs from the physician, and plays the role of the helpful assistant, anticipating future needs. Even this assistant role requires a significant understanding of the data entered into the medical records.

1.1.1 Potential applications

Some examples of tasks that we would expect a smart EMR to be able to help with are listed here.

- **Clinical decision support:** Best-practice guidelines abound in medicine, based on an increasing body of evidence-based-medicine. This body of literature is growing faster than individual practitioners, even specialists, can keep track of (Glasziou and Haynes,

2005; Berner and Moss, 2005). An EMR should be able to easily identify patients that fit guideline classifications and point physicians to relevant documentation.

- **Contextual displays:** It should be easy to find tests and medications that are relevant for a given patient. Standard dialog-based navigations can impose a significant cognitive load and require periods of acclimatization. The contextual displays shown in Figure 1.2 are examples of panels that make it easy to access standardized order sets or document chief complaints.
- **Patient surveillance:** The medical record system receives inputs from multiple care providers and automated inputs from devices. As such, it is the best source to flag conditions that could indicate danger to a patient. In addition, as the number of monitoring devices increases, it becomes impossible to manually monitor them at all times, especially in resources constrained settings such as emergency departments.
- **Personalized alerting:** The EMR system can pull from multiple data sources to build alerting conditions that take the context of the patient into account (e.g., historical baseline values, medications, and procedures). This broad view of the patient can be used as input for more personalized alerts, potentially suppressing alerts for values that would cause alerts in a standard patient but are “normal” for this patient (e.g., patients with chronic hypotension routinely trigger false alarms due to their low blood pressure). Conversely, it can alert for values that are within the population’s “normal” range, but indicate a severe condition for this particular patient.
- **Search and relevant history discovery:** It can be difficult to wade through years of patient records to find relevant history. In many cases, an in-depth exploration of the patient’s medical history is simply not feasible for time-constrained physicians. However, a document or procedure from years ago may significantly impact a physician’s

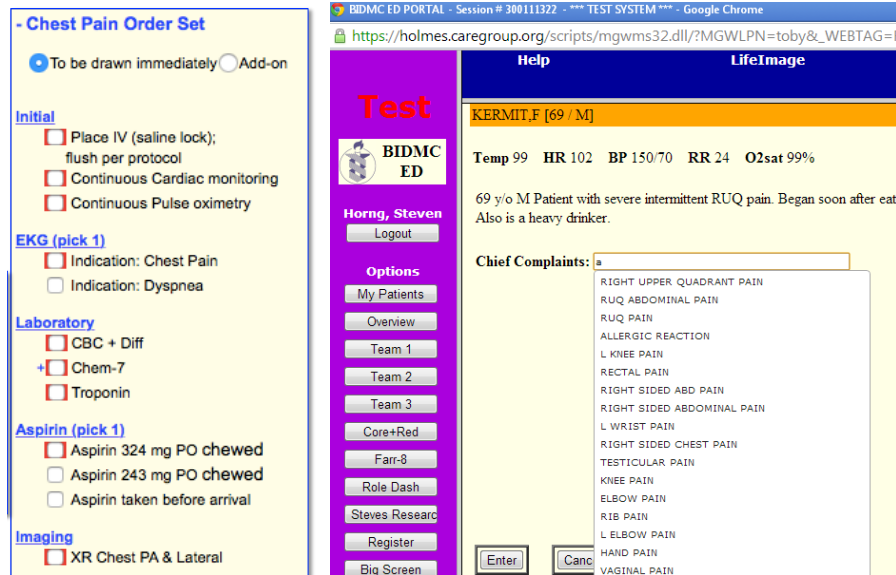


Figure 1.2: Example of a context-specific displays possible in a smart EMR. Left: an order set appears for patients who the dashboard recognizes as having “chest pain”. Right: smart autocomplete of chief complaints described implemented in the dashboard as described in Jernite et al. (2013a).

understanding of a patient’s current presentation. The ability to easily browse a patient’s record for relevant past events could significantly reduce missed or delayed diagnoses.

- **Cohort selection for real-time or retrospective studies:** We would like to be able to query the medical record system with flexible cohort definitions in order to facilitate quality assurance and observational studies. For example, Longhurst et al. (2014) describes a system to allow physicians to run personalized observational studies “on-the-fly” in order to make maximally informed decisions.

1.1.2 EMR Phenotyping – a useful building block

As a building block towards the smart applications described in the previous section, we focus on a task known as electronic medical record (EMR) phenotyping. EMR phenotyping is the task of determining, from an electronic medical record entry, a concise descriptor of

the patient, comprising of their medical history, current problems, presentation, etc. In this work, the EMR phenotype consists completely of the answers to yes/no questions (e.g., Does the patient have a stroke, Is the patient currently anticoagulated? Is the patient in septic shock?). In inferring such a phenotype descriptor from the record, a computer, in a sense, “understands” the medical record. These phenotypes can then be used in downstream applications listed above.

Earlier work on electronic record phenotyping focused on manual specification of consensus-based rules (e.g., Newton et al., 2013; Mo et al., 2015; Kho et al., 2011; Overby et al., 2013; Conway et al., 2011). These methods suffer from issues of portability, due to different coding and recording standards at different sites (Liu et al., 2012). Structured data such as problem lists are often incomplete (Wright et al., 2012; Gandhi et al., 2011) and thus by themselves not reliable for making important clinical decisions. While the development and adoption of common data models, such as those used by OMOP (Stang et al., 2012) or PCORnet (Fleurence et al., 2014), can contribute to developing portable rule-based specifications, they are necessarily built for generalization, preferring to use structured data elements, and avoiding the use of data elements that are not generally available across sites. Figure 1.8 shows an example of a consensus-built phenotype definition for autism spectrum disorders.

Manual rules are difficult to apply to free-text fields, since the number of different ways of saying the same thing can potentially be very large (Friedman and Elhadad, 2014). While some work has begun to incorporate the output of natural language processing algorithms as part of phenotype definitions (Liao et al., 2015), this is not yet the standard approach. In developing manual rules, we are also limited in the number of variables and cross-interactions that can be considered at once.

Statistical learning methods from the machine learning community have the potential to build much more robust, site-specific, and personalized phenotype algorithms that can

simultaneously consider data elements from a patient’s entire electronic health record. Figure 1.3 shows an example of a learned phenotyping algorithm to find patients who have come from a nursing home. These rules can be learned on site-specific data, so that they can understand new (or deprecated) data inputs that are not widely used; and be tailored to particular demographics (the common conditions in an old age home differ from those in a nursery, and from Alaska to Puerto Rico).

1.1.3 Difficulty of obtaining gold-standard labels

A difficulty that often arises in training statistical models is that they require labels of cases and controls with respect to conditions of interest. This task usually often requires manual labeling through chart review, which requires trained physicians and is extremely labor intensive. Structured fields in the medical record, like diagnosis codes, or problem lists are often incomplete or incorrect, and thus cannot be fully trusted (Birman-Deych et al., 2005; Aronsky et al., 2005; Tieder et al., 2011; Cipparone et al., 2015; Wright et al., 2012; Gandhi et al., 2011; O’malley et al., 2005).

1.2 Machine learning background

We now turn to some of the tools and concepts that will be useful throughout the thesis.

1.2.1 Probabilistic Graphical Models

Probabilistic graphical models (Koller and Friedman, 2009) such as Bayesian networks provide a useful abstraction for quantifying uncertainty and describing complex dependencies in data. They belong to both the world of probability (random variables, distributions, and statistical complexity), and computer science (graph algorithms, computational complexity).

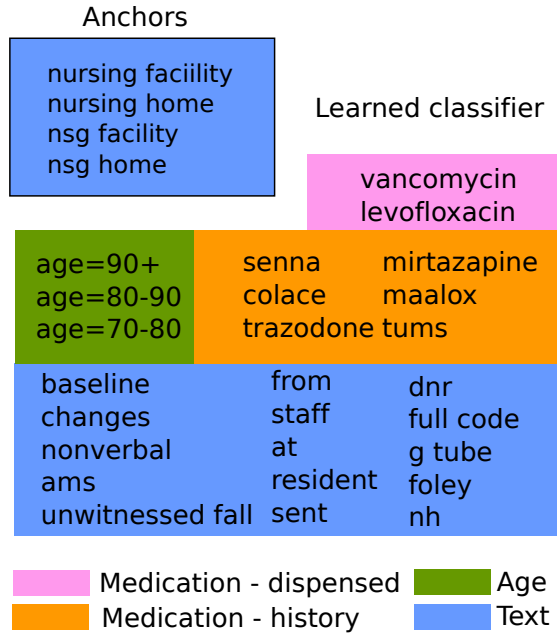


Figure 1.3: Example of a phenotype extractor learned to identify patients from nursing homes (source: Halpern et al. (2014)) The anchors are a small number of phrases manually specified by a clinical collaborator. The rest of the classifier is learned from medical records. Each word is associated with a weight. Highly indicative words are displayed.

In a Bayesian network, conditional independences between variables are encoded in a directed graph. A distribution over variables $P(X_1, \dots, X_n)$ can be described by a graph \mathcal{G} with nodes corresponding to each of the variables if the distribution *factorizes* according to the structure of the graph:

$$P(X_1, \dots, X_n) = \prod_i P(X_i | \text{Pa}(X_i; \mathcal{G})), \quad (1.1)$$

where $\text{Pa}(X_i; \mathcal{G})$ are the parents of X_i in the directed graph \mathcal{G} . A distribution can then be described by a graph structure \mathcal{G} and a set of conditional probabilities $P(X_i | \text{Pa}(X_i))$. For graphs with low in-degree or parametrized conditional probabilities, this representation can be much more compact than the original joint distribution $P(X_1, \dots, X_n)$. Figure 1.4 shows some examples of Bayesian networks that have been used in healthcare settings.

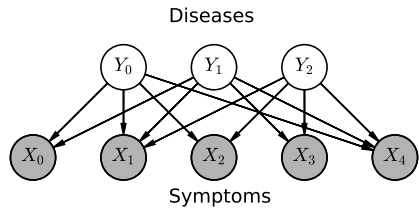


Figure 1.5: A simplified schematic of the QMR-DT network (Miller et al., 1986; Shwe et al., 1991) structure for medical diagnosis. The full QMR-DT network models the relationships between 570 diseases and 4075 symptoms.

1.2.3 Bipartite noisy-or Bayesian networks

The QMR-DT network (Miller et al., 1986; Shwe et al., 1991) (Figure 1.5) is a Bayesian network describing binary random variables corresponding to diseases and findings. The diseases are considered to be *unobserved* variables, and findings can either be “present” or “absent” (“unknown” findings can also be handled within this framework by treating them as unobserved as well). The model is bipartite, with edges going from diseases to symptoms. An edge is present from a disease Y_i to a symptom X_j if the disease can cause that symptom, and the conditional distribution of each symptom is modeled by a noisy-or gate:

$$P(X_j = 0 | Y = \{y_1, \dots, y_n\}) = (1 - l_j) \prod_{i=1}^n f_{i,j}^{y_i}. \quad (1.3)$$

For computational efficiency of inference, $P(Y)$ is also generally assumed to be a factorized distribution of Bernoulli random variables:

$$P(Y) = \prod_{i=1}^n \pi_i^{y_i} (1 - \pi_i)^{1-y_i}. \quad (1.4)$$

Chapter 6 explores other options beyond this fully-factorized setting.

The full joint distribution is then a product of the conditional distributions of the

symptoms:

$$P(X, Y) = P(Y) \prod_{j=1}^m P(X_j | Y). \quad (1.5)$$

Inference in the model corresponds to determining, from the observed symptoms, the most likely diseases that the patient has.

The generative model of the noisy-or can be described as follows:

First, the parents Y_1, \dots, Y_n , turn on following some distribution (independent Bernoulli or otherwise). Then, every parent Y_i which has a value 1, attempts to “turn on” each of its children, X_j and fails with a failure probability $f_{i,j}$ which is a parameter for every edge in the graphical model. Finally, an extra noise process attempts to “turn on” each variable X_j and fails with probability $1 - l_j$. We then observe X_j as “off” if all of the attempts to turn it on fail, otherwise it is seen to be on.

Noisy-or networks make the following simplifying assumptions:

- Only the presence of a disease has the ability to make a symptom appear, not its absence.
- The failure of a symptom to manifest depends only on the presence or absence of diseases. Symptoms do not cause, or inhibit each other.
- Diseases and symptoms are binary. No notion of severity is captured in these models (though extensions do exist).

The QMR-DT network and its parameters were elicited from experts and was estimated to take approximately 20 person-years of researcher time to build the entire network. The ability to learn the network and its parameters from data would be very useful to ever build something of a similar magnitude or larger.

Diseases in the QMR-DT network are *latent variables*. They are never observed directly, only inferred based on constellations of symptoms.

1.2.4 Worst-case computational complexity and hardness

The most common way of fitting a model to data is to seek the model that maximizes the probability of the observed data under the model, called maximum likelihood estimation (MLE). However, for many interesting models with latent variables, learning and inference can be shown to be computationally intractable in the worst case. For example, topic modeling with Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is NP-hard for both maximum likelihood estimation of its parameters (i.e., learning) (Arora et al., 2012b) and inference (Sontag and Roy, 2011). Mixtures of Gaussians and Hidden Markov models are both NP-hard to learn (though inference in both models is tractable). The medical diagnosis task, which corresponds to inferring the likely diseases for a patient after observing only their symptoms, is #P-hard (Cooper, 1987) in noisy-or networks like QMR-DT, and maximum likelihood estimation of parameters in these models is conjectured to be NP-hard as well. The difficulty comes from the fact that even evaluating the likelihood objective requires *marginalizing* over the unknown variables (Equation 1.6 shows this marginalization procedure for binary latent variables). In the QMR-DT network, this would consist of summing over all possible states of a binary vector of length 470 (i.e., 2^{470} possible combinations).

$$\log P(X) = \log \sum_{y \in \{0,1\}^n} P(X, y). \quad (1.6)$$

In addition, for the noisy or Bayesian networks described in Section 1.2.3 (and many other latent variable models as well), the likelihood objective is not concave in its parameters, making it difficult to optimize with local search methods, even if the objective could be computed tractably.

1.2.5 Method of moments learning and identifiability

An alternative method to maximum likelihood learning, called the method of moments, seeks to find a model that best matches certain low-order moments of the data distribution. Figure 1.6 shows a schematic of this process, mapping from low order moments to parameters. The original application of the method dates back to Pearson’s study of Naples crabs (Pearson, 1894) in which he computed the roots of a sixth order polynomial to fit a mixture of two Gaussians.

While less asymptotically statistically efficient than Fisher’s maximum likelihood estimation technique (Fisher, 1922), the method of moments can yield more computationally efficient learning algorithms. This computational efficiency comes from a number of factors. First, unlike many popular likelihood-based methods like Expectation Maximization (EM) (Dempster et al., 1977), or variational EM (Neal and Hinton, 1998), these methods avoid costly inference subroutines. They also tend to be highly parallelizable and only read through the data once to collect sufficient statistics, making them attractive for streaming or large data. With the availability of large data sets, the option to build computationally efficient tools at the expense of statistical efficiency is attractive.

Using method-of-moments to recover the original parameters of the data generating distribution presupposes some notion of identifiability (Figure 1.7). That is, that the parameters of the distribution are recoverable from moments of the distribution. Proving identifiability, and constructing mappings between moments and the parameters of a generating distribution for latent variable models has only been accomplished for a few distinct families of models (Allman et al., 2009; Anandkumar et al., 2013c). The method of moments algorithms presented in the thesis (Chapters 2,4,6,7) build explicit mappings between low order moments and parameters of the model to show identifiability, and turn these explicit constructions into useable learning algorithms. The use of low-order moments is intentional to maintain

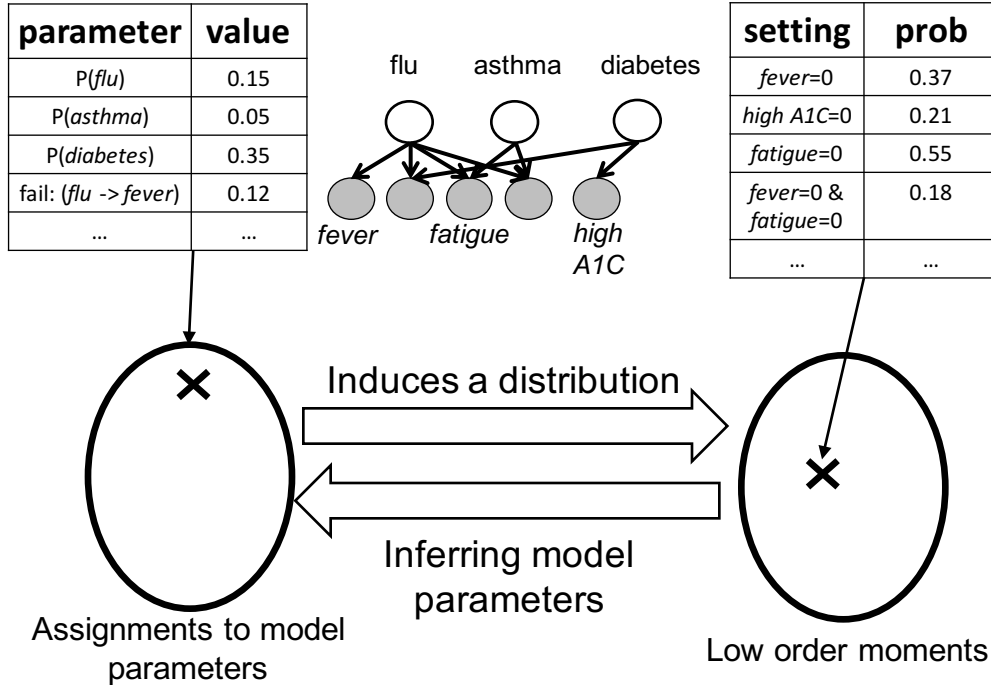


Figure 1.6: The method of moments attempts to invert the mapping between moments of the distributions (e.g., probability of simultaneous negative results) and the parameters of the model (e.g., a noisy-or parametrization with priors, failures, and leak as described in Section 1.2.3).

both computational efficiency, since high order moments are computationally expensive to compute, and statistical efficiency, since estimating higher order moments accurately requires more data.

1.3 Themes

A number of themes recur throughout the thesis. To avoid excessive repetition, we present a general discussion of them here and point back to them where relevant.

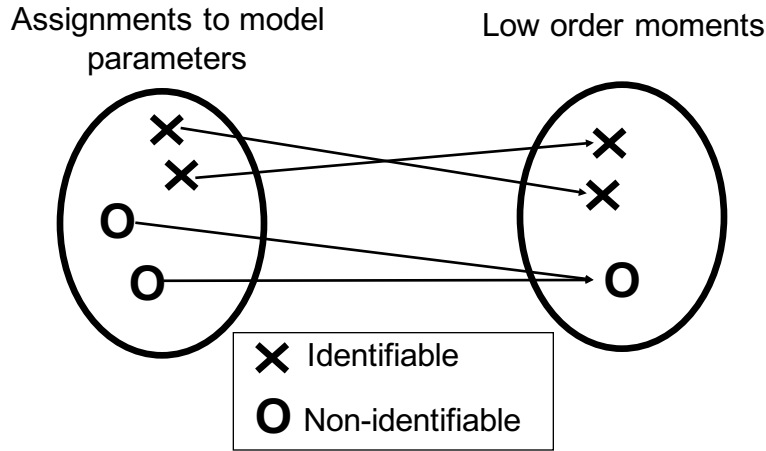


Figure 1.7: Identifiable parameters (X) map to a unique setting of the low orders. Parameters may be non-identifiable (O) if multiple parameter settings lead to the same low order moments.

1.3.1 Learning from data and expert input

Even the earliest computerized decision support systems recognized the value in learning from data (Leaper et al., 1972). With the new-found availability of large collections of electronic medical records, the possibility of learning directly from data is extremely appealing. However, the data can be modeled in many different ways and expert input is often required to guide the models to learn the “right” thing. Alternatively, input from an expert can significantly reduce the complexity of the learning task, requiring fewer samples or less compute time to learn.

Bayesian networks make a clean distinction between the structure of a network, which is often specified through domain knowledge, and the parameters of a network, which are often learned from data. However, when modeling arbitrary features from a medical record (e.g., unigrams and bigrams from text), it is often very difficult for an expert to specify the complete structure of the Bayesian network by hand, and structure learning is useful.

Manual labeling of instances (i.e., supervised learning) is also a way for experts to provide guidance to a learning algorithm, but obtaining these labels can be expensive, and an

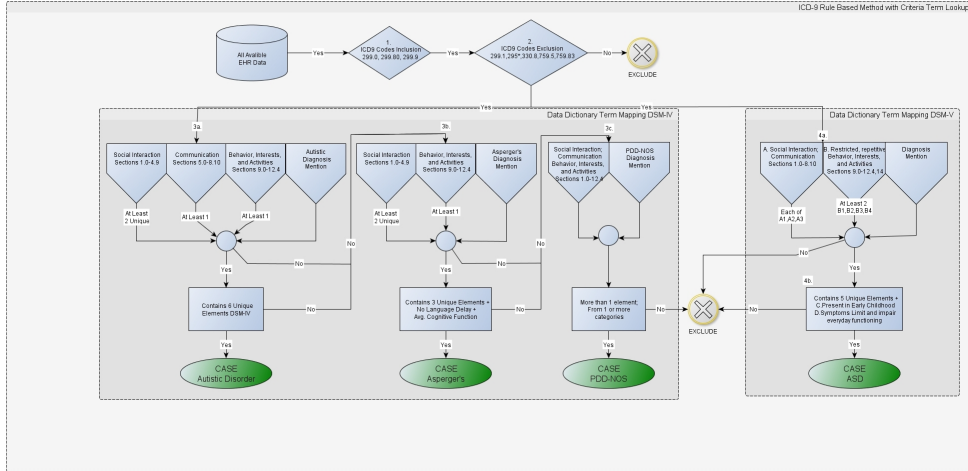


Figure 1.8: An example of a manually specified phenotyping algorithm for autism spectrum disorders. Source: Lingren (2013)

inefficient use of expert’s time.

1.3.2 Features that make learning tractable

Many of the formal results on computational intractability of maximum likelihood rely on constructions with some amount of worst-case data. When data is known to be drawn from distributions that have certain features that make learning easy, the solutions to method-of-moments algorithms can be shown to coincide with the output of maximum likelihood estimation (both are consistent estimators), providing a computationally tractable solution. Two examples of these features that are described in this thesis are *anchors* and *singly-coupled quartets*. By assuming the presence of a structural condition linking one or more observations to each latent variable, it becomes possible to identify the effect of the latent variable, which effectively unlocks the parameters of the entire distribution. Interestingly, this effect seems to only occur when viewing aggregate statistics as in the method-of-moments algorithms. Incremental likelihood optimizations such as EM do not have an obvious way to exploit these distributional assumptions, and can still get stuck in local optima even when the underlying

distribution satisfies these assumptions.

1.3.3 Robustness to model misspecification

With real data, it is rarely true that the true data distribution lies in a parametrized model family. Ideally, under model-misspecification, we would like that the learned model still represents the data as best as possible. With maximum likelihood estimation, the learned model is a projection of the true distribution onto the model family using KL-divergence as a measure of difference. With method-of-moments, the robustness to model-misspecification can depend heavily on the particular algorithm. Constrained optimization or projection techniques have been used to at least ensure that the learned model has feasible parameters (Shaban et al., 2015; Cohen et al., 2013; Duchi et al., 2008; Knol and ten Berge, 1989).

1.3.4 Assessing utility of a phenotyping system

In this section, we elaborate on some possible use cases of an EMR phenotyping system to allow for a grounded discussion of the performance characteristics of the learned classifiers, and to motivate the evaluation frameworks of later chapters. A full assessment of utility would require a clinical trial with the phenotype estimates driving some health IT application and the ultimate goal being improved outcomes for patients. In this work, as a first step, we evaluate the phenotype estimates themselves for potential use cases described below. Other potential use cases and metrics that are not directly relevant to the evaluations in this thesis are found in Appendix A.1.

Each of the settings below can be implemented in both real-time or retrospective analyses. The main difference between these two frameworks is the amount of information that is available to make decisions.

1.3.4.1 Closed loop systems: Extremely high PPV

The use case with the highest bar for operating characteristics are uses where the computer is given the power to take implement an intervention (e.g., change dosing or machine settings) based on information in the EMR. This type of control requires the most trust in the computer system’s understanding of the situation. In many situations, the computer’s action could have negative consequences and thus a high positive predictive value (PPV) is required to have confidence that the computer’s actions are justified under the circumstances. This type of use-case is very rare in healthcare today. Closed loop ventilators (Clavieras et al., 2013; Arnal et al., 2012) and drug-delivery systems (Janda et al., 2011; Hemmerling and Charabti, 2009) are currently undergoing clinical trials, but have not achieved widespread adoption.

1.3.4.2 Interruptive alert triggering: High PPV

One use of phenotype variables is to interrupt physician’s regular work to encourage them to pay attention to a particular detail or fact. These alerts are useful is there is potentially an important piece of information that if overlooked could lead to bad outcomes. Examples of these warnings currently implemented in many hospitals include warnings for shock, allergies, or drug interactions. These alerts are widely recognized as important, and there is a real need to implement them effectively, however, they are often plagued with implementation issues such as alert fatigue, a well-documented phenomenon that alerts with low positive-predictive value tend to be ignored over time. This phenomenon is particularly present in intensive care units (ICUs), emergency departments (ED), and surgical settings, since the number of potential life-threatening conditions are high, and alarms (both true and false) are prevalent.

Alert fatigue is directly related to the positive predictive value (PPV) of a classifier, and thus the PPV of a proposed classifier is an important quantity to be assessed. Classifiers for use in interruptive alert triggering can be compared along the axis of recall at high precision,

setting thresholds so that precision is fixed at 0.75 or 0.9 and asking how many positive cases can be detected by the classifier.

1.3.4.3 Contextual displays: Moderate PPV or top K suggestions

A second type of alert, that is less interruptive than the alerts described above, is a conditional display, or passive alert. These alerts appear on a physician screen, sometimes as highlighting or under sidebars with headings such as “reminders” or “suggested links”. They can be used to encourage enrollment of patients in relevant clinical pathways, which have been shown to reduce variations in care and improve outcomes by promoting best practices (Panella et al., 2003); provide helpful information at the point of care, such as suggested treatments or dosing for particular conditions; or simply reduce the number of clicks required to pull up a relevant screen.

Contextual displays incur less risk of alert fatigue, since they do not interrupt the physician’s work-flow, but they still run the risk of being ignored if they do not meet a requisite threshold of containing some amount of useful information near the top of the list (usually sized to contain 3-5 items). Including a “snippet” or other representation of why the reminder triggered along the reminder, allows the physician to quickly evaluate whether the display is relevant or not, and quickly act to accept or reject the suggestion.

1.3.4.4 Comparing between methodologies: Area under the ROC or PR curve

Area under the receiver operator characteristic curve (AUC) or precision-recall curve give a holistic view of a classifier operating over a wide range of characteristics. This can be useful when comparing between two methodologies to show the advantage of one over the other, without reference to any particular set of operating characteristics. In settings with high class imbalance, high values for AUC may not indicate clinical utility, and area under the precision-recall curve can be more informative (Saito and Rehmsmeier, 2015).

1.4 Outline and Contributions

Chapters 2 and 3 address the problem of learning models for electronic medical record phenotyping without gold standard labels (on the difficulty of obtaining gold-standard labels, see Section 1.1.3).

Chapter 2 presents the Anchor and Learn framework to learn discriminative classifiers to classify patients as having or not having a particular phenotype, without requiring extensive manual labeling of cases and controls. We demonstrate the feasibility of this approach by building 42 publicly available clinical phenotypes, and training each one on a dataset of 200,000 emergency department records. The methodology is validated on 8 clinical state variables in an emergency department setting, comparing the phenotype extractors learned with anchors against prospectively gathered gold-standard labels. We demonstrate that the classifiers perform compare favorably with models learned with fully supervised data or manual rules. All but one of the learned classifiers are sufficiently reliable for some form of clinical use.

Chapter 3 presents a likelihood-based approach using anchors to learn factor-analysis models of phenotypes and general features extracted from the EMR. The likelihood objective has the advantage of being robust to model misspecification as discussed in Section 1.3.3. These models can be used to infer the values of unknown phenotypes and determine the most likely phenotype to explain a set of features. We describe the ungrounded latent variable problem which occurs when learning with anchors instead of true labels and describe a semi-supervised objective which is effective at mitigating the problem. We experimentally validate the learned algorithms against relevant baselines including the independent classifiers described in Chapter 2 on a clinically relevant inference task and show that a joint model performs best at ranking most-likely applicable phenotypes, performing nearly as well as oracle models learned with true labels.

Chapters 4 and 5 address theoretical problems related to EMR phenotyping which is the polynomial time learnability of noisy-or factor analysis models used in Chapter 3.

Chapter 4 gives a set of *sufficient* structural conditions for polynomial learnability based on singly-coupled tuples and develop a provably correct polynomial-time method-of-moments algorithm based on these sufficient conditions. We show empirically that the sufficient conditions for learnability almost hold in the case of the QMR-DT network structure, meaning that the vast majority of its parameters *can* be estimated consistently, leaving a small number of parameters unknown. Compared to variational EM, the method-of-moments algorithm is both faster for large datasets, and provides consistent parameter estimates in settings where variational EM does not. We show experimentally that stochastic variational inference (the method introduced in Chapter 3) is also effective and fast for parameter recovery.

Chapter 5 shows polynomial time learnability for the *anchored* setting, and develops a different polynomial time method of moments algorithm for anchored factor analysis. In the anchored setting, we can relax the assumption of factor independence that appears in all of the previous chapters, learning models where the distribution of the factors is a tree structured Bayesian network in polynomial time. Models where the distribution of the factors is a general Bayesian network can also be recovered, but not provably efficiently.

Chapter 6 assesses the feasibility of using EMR data to learn causal models of diseases and symptoms as the basis for a knowledge base.

Chapter 7 shows the potential of the concepts described in the previous chapters to apply broadly in data science, using the anchor assumption to learn topic models. It presents a provable method of moments algorithm that is also practical to run on real datasets, learning models that compete with state of the art approximate likelihood approaches and run in a fraction of the time.

Chapter 8 outlines broad possible directions for future research.

Chapter 2

Phenotype classifiers learned with anchors

Acknowledgments This work was joint with Youngduck Choi, Abdul Tlimat, Steve Horng, and David Sontag. Parts of the work were previously published in: “Using Anchors to Estimate Clinical State without Labeled Data”. Halpern, Choi, Horng and Sontag. AMIA 2014 and “Electronic medical record phenotyping using the anchor and learn framework” Halpern, Hong, Choi and Sontag. JAMIA 2016.

2.1 Introduction

Electronic medical record phenotyping is the task of extracting simple facts about a patient from their electronic medical record, which are suitable to use as input for downstream health-IT applications (see Section 1.1.2 for a more complete introduction to EMR phenotyping). These facts serve as a knowledge representation of the individual patient (Figure 2.1), distilling the entire patient narrative into a form suitable as input for clinical decision support, bringing personalized evidence-based risk assessments and treatment recommendations to the bedside.

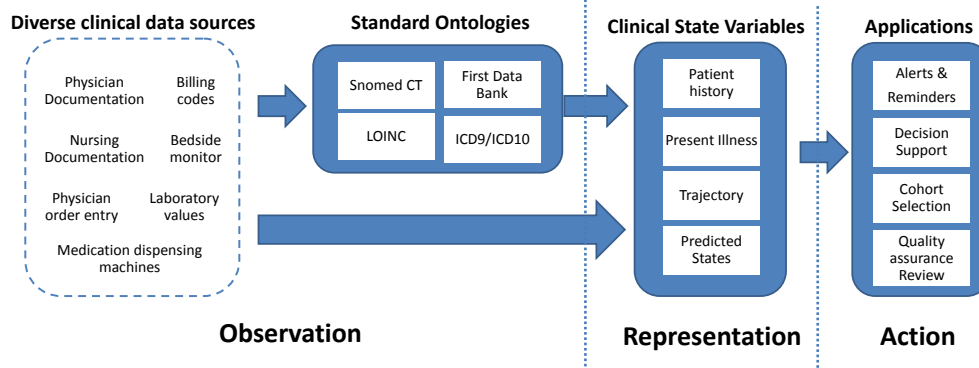


Figure 2.1: A schematic flow of how diverse clinical observations, both structured and unstructured, can be distilled into relevant clinical state variables (phenotypes), and used for applications.

The scalability of all parts of this procedure is of utmost importance. Existing approaches tend to use manual rule-specification or supervised machine learning techniques. Unfortunately, neither of these approaches scale to a large number of phenotypes nor to large numbers of institutions.

Manually specified rules can be difficult to build based on consensus of expert physicians for a number of reasons. First, actionable data in EMRs is often found in unstructured free-text notes, for which it can be difficult to specify manual rules, due to the natural variability in free-text.

For supervised machine learning, manual chart review remains the standard approach to derive gold-standard labels for training, but it is not feasible at scale. Hundreds to thousands of phenotypes are relevant for implementing existing guidelines and we expect that number to grow in the future. Data collection differs from hospital-to-hospital, and it is expensive to label enough instances to train fully supervised classifiers for new site.

Intuition tells us that a perfect labeling is probably not necessary to learn a reasonable classifier and a scalable method of labeling could be achieved if we were satisfied with imperfect labels. However, it does not tell us how to *design* a noisy labeling system to minimize the impact on the learned classifier. Agarwal et al. (2014) analyze the phenotype problem under

a noisy-label framework where label-flipping has a fixed probability. However, they assume from the start that the framework holds, and give no practical guidelines on how to build a noisy-labeling that satisfies these properties.

In this work, we analyze a different setting, which we believe is easier to achieve. We propose learning with “anchors”, a semi-supervised method, where experts specify, for each phenotype variable, at least one observable variable that is uniquely linked to it (more details in Section 2.2), that can act as an anchor (a type of partially reliable label).

2.1.1 Contributions

This chapter contains the following contributions:

- We formally define the noisy-labeling scheme based on *anchors*, and link it with previous results from machine learning on learning with noisy-labels.
- We demonstrate the feasibility of this approach by building 42 clinical phenotypes, and training each one on a dataset of 200,000 emergency department records. The 42 phenotype definitions are publicly available, allowing others institutions to use them to train on their own datasets. We also encourage others to reproduce our anchor-specification pipeline and contribute phenotype definitions which are useful at their institutions back to the broader community. The larger goal is to build a publicly available phenotype library, upon which clinical decision support and research applications can be built.
- The methodology is validated on 8 clinical state variables in an emergency department setting, comparing the phenotype extractors learned with anchors against prospectively gathered gold-standard labels. We demonstrate that the classifiers compare favorably with models learned with fully supervised data or manual rules. All but one of the learned classifiers are sufficiently reliable for some form of clinical use.

- The classifiers are described qualitatively to show face validity and we describe the data that was most useful to us in obtaining the recorded performance at our site.

2.1.2 Related work

Phenotypes based on data in the electronic medical record have been used to identify adverse drug events (Liu et al., 2013), perform genome-wide association studies (Kullo et al., 2010; Crosslin et al., 2012; Denny et al., 2010; Kullo et al., 2011; Denny et al., 2011; Kho et al., 2012), and for other large-scale health research initiatives (Jensen et al., 2012; Wilke et al., 2011; Richesson et al., 2013, 2014; Shivade et al., 2014). While there has been considerable success in sharing community-built phenotypes for research purposes, (e.g., the PheKB knowledge base (Newton et al., 2013)) there has been less work on building phenotypes for activating clinical decision support in real-time. Phenotypes intended for retrospective studies often rely heavily on ICD9 and CPT codes, which would typically not be available in time to be useful for clinical decision support. Recent work also includes input from free text either in the form of simple queries (Conway et al., 2011), or using more advanced natural language processing (Liao et al., 2015).

Phenotypes in PheKB are developed manually through a rigorous process, requiring multiple iterations and eventual physician consensus. The final definitions achieve high concordance with clinical gold standards, but they are time consuming to build, requiring many hours of expert time (Newton et al., 2013). In contrast to the manually derived rules for electronic phenotyping, statistical methods, drawing on established machine learning techniques, have been used to estimate phenotypes based on inputs from the EMR. Previous work has shown success in predicting individual phenotype variables from raw data (e.g. smoking status (Liu et al., 2012; McCormick et al., 2008), rheumatoid arthritis (Carroll et al., 2011), colorectal cancer (Xu et al., 2011)), but the methodology for developing these

predictors invariably uses manually labeled cases and controls derived from chart review. As such, these efforts are limited in scope, focusing on one or two phenotype variables at a time. In addition, the learned classifiers are institution-specific and often do not generalize well without modifications (Liu et al., 2012) or local retraining (Liu et al., 2012).

Methodologically, the closest system to our current work uses Silver standard training sets (Agarwal et al., 2014, 2016), with partially reliable labels, within a machine learning pipeline which learns to estimate phenotype variables. While our phenotype specification framework and training methodology in this chapter is similar, our phenotype library and evaluation focus on phenotypes relevant for real-time clinical decision support, as opposed to retrospective comparative effectiveness studies.

2.2 Anchor and Learn Framework

We use Y to denote the true phenotype variable, or label, X to denote features, and A to denote the anchor feature. \tilde{X} denotes all of the features *except* the anchor. We use uppercase to denote the name of the random variable (e.g., $P(X)$), and lowercase to denote the value of the variable (e.g., $P(Y = 1|X = x; w, b) = \sigma(w \cdot x + b)$).

2.2.1 Choosing phenotypes

Before choosing anchors, the first task is to choose which phenotypes (also called “tags”) are going to be modeled. In the medical setting, these could be a set of medical conditions which we want to be able to detect in real time (e.g., fall risk, nursing home, insulin-dependent diabetes), or it a particular cohort to be studied retrospectively (e.g., patients with drug induced liver injury). More broadly, this approach could be applied to non-healthcare settings as well. For example, it has been applied to questions on the Stack Overflow site, a collaboratively edited question-answer site for computer programmers. The “phenotypes”

were user-provided tags such as *arrays* or *c++*, and the features were drawn from the text of the questions (See Section 6.5). It has also been applied to restaurant reviews where the phenotypes are descriptors such of the cuisine, decor, price range, etc. A phenotype can be any descriptive question that can be phrased as a binary classification task that we would expect to be able to answer from the data.

2.2.2 Feature extraction

We start with a high dimension feature vector corresponding to each patient instance. A patient instance can represent a single visit, or multiple visits collapsed over a length of time. In this example, we will consider a simple high-dimensional binary bag-of-words scheme where each feature can either be “present” or “absent”.

Continuous features such as lab test values may have a non-linear relationship with the phenotype variables. Anchors can also be used to learn the optimal bin boundaries to convert continuous variables to binary indicators. We follow the optimal binning procedure of Fayyad and Irani (1993) using a decision tree to predict the presence or absence of the anchor from a single continuous variable. The leaves of the decision tree are then used to bin the continuous value into binary indicators. This leads to a different set of bin boundaries for each phenotype prediction problem, as the boundaries are learned specifically to be meaningful for the individual extraction task.

2.2.3 Choosing anchors

The task of choosing anchors is where the practitioner encodes their knowledge of the problem. In general, anchors cannot be specified automatically, and require input from domain experts. In Section 2.2.8, we describe a scheme for generating *suggestions* for anchors, after a single anchor has been specified. But these are only suggestions, based on a simple

heuristic. A domain expert must evaluate whether or not the suggested anchor makes sense. Some examples of the phenotypes and anchors chosen by a clinical collaborator for use in Beth Israel Deaconess Medical Center’s emergency department are shown in Table 2.1.

2.2.3.1 Anchor definition

For each phenotype, we then choose one or more anchors to act as surrogate labels. Anchors are simply features that satisfy the two anchor conditions:

- **High Positive Predictive Value:** If the anchor is present, that should be highly indicative that the phenotype is positive. Formally: $P(Y = 1|A = 1) \approx 1$ or $P(A = 1|Y = 0) \approx 0$.
- **Conditional independence:** The anchor should *only depend* on the true phenotype. Conditioned on knowing the true phenotype, the presence or absence of the anchor should be a random variable, independent of all other features. Formally: $A \perp \tilde{X}|Y$ or $P(A|Y, \tilde{X}) = P(A|Y)$.

The conditional independence condition is the condition that unlikely to hold perfectly on real data, and does not have a clear test. Even so, in section 2.2.7, we will describe some checks and heuristics for making sure an anchor is good.

2.2.3.2 More than one anchor

More than one anchor can be specified per condition. The simplest way of handling multiple anchors is to combine them with an OR operator (i.e., create a new feature called `combined-anchor`, which is on if *any* of the anchors are present). More generally, features can be combined with any set of logical operations (e.g., AND, OR, NOT) to create a final anchor. Another method is to learn multiple classifiers, one for each anchor, and then to combine their results through voting.





Phenotype	Data Source	Anchors	
Anticoagulated (history)	ICD9 codes	790.92 ABNORMAL COAGULATION PROFILE	
		E934.2 ADV EFF ANTICOAGULANTS	
	Medication history	V58.61 LONG TERM USE ANTIGOAGULANT	
		Anticoagulants – Coumarin	
		Thrombin Inhibitor - Selective Direct & Reversible	
		Factor IX Preparations	
Medical Text	ffp		
Diabetes (history)	ICD9 codes	250:DIABETES MELLITUS	
	Medication history	Diabetic Therapy	
Liver (history)	ICD9 codes	571 CHRONIC LIVER DISEASE AND CIRRHOSIS	
	ICD9 codes	572.2 HEPATIC ENCEPHALOPATHY	
	Medical Text	cirrhosis	
	Medical Text	esld	
	Medical Text	hcv	
Allergic reaction (acute)	ICD9 codes	995.3 - ALLERGY, UNSPECIFIED	
	Medical Text	allergic reaction	
	Medical Text	allergic rxn	
Cholecystitis (acute)	ICD9 codes	574 CHOLELITHIASIS	
	ICD9 codes	575.0 ACUTE CHOLECYSTITIS	
Deep vein thrombosis (acute)	ICD9 codes	453.40 ACUTE VENOUS EMBOLISM AND THROMBOSIS OF UNSPECIFIED DEEP VESSELS OF LOWER EXTREMITY	
	ICD9 codes	453.41 ACUTE VENOUS EMBOLISM AND THROMBOSIS OF DEEP VESSELS OF PROXIMAL LOWER EXTREMITY	
Employee exposure (acute)	Medical Text	employee exposure	
	Medical Text	needlestick	
Epistaxis (acute)	ICD9 codes	E920.5 HYPODERMIC NEEDLE	
	ICD9 codes	784.7 EPISTAXIS	
Laceration (acute)	Medical Text	lac	
	Medical Text	laceration	
Suicidal ideation (acute)	ICD9 codes	V62.84 SUICIDAL IDEATION	
	Medical Text	si	
	Medical Text	suicidal ideation	
Legend:			
	Medication dispensing record		Medication history
	ICD9 codes		Medical Text

Table 2.1: A selection of the 42 phenotypes built as part of this ongoing project. Each phenotype is defined by its anchors, which can be specified as ICD9 codes, medications (history or dispensed), or free text. When a large number of anchors are specified, only a selection are shown. For display, medications are grouped by extended therapeutic class (ETC).

If the two anchors are conditionally independent, that also implies the presence of a singly-coupled triplet that can be used to learn the noise rates of the anchors or other parts of the model (see Chapter 4). This is the setting described in Steinhardt and Liang (2016) for estimating the performance of the learned classifier as well. However, in practice, multiple conditionally independent anchors can be difficult to specify for a particular domain.

2.2.4 Learning a model

We describe a method for learning phenotype estimators based on the learning procedure outlined in Elkan and Noto (2008) for learning with noisy labels (a derivation is found in Section 2.2.5). We train models with L2 regularized logistic regression (this can be replaced with another calibrated classifier like random forests). A logistic regression model seeks a weight vector w and bias b to minimize:

$$\operatorname{argmin}_w \|w\|^2 + \sum_n \ell(\sigma(wx^{(n)} + b), y^{(n)}), \quad (2.1)$$

where $\tilde{x}^{(n)}$ is the feature vector associated with the n th training example, $y^{(n)}$ is the corresponding label, σ is the sigmoid function, and ℓ is the log-loss.

In the anchor and learn framework, we substitute the anchor(s), A for the true labels, which are unavailable. We also modify the feature matrix X , to remove the columns corresponding to the anchors, yielding a censored matrix \tilde{X} . Thus the final optimization problem is:

$$\operatorname{argmin}_w \|w\|^2 + \sum_n \ell(\sigma(w\tilde{x}^{(n)} + b), a^{(n)}), \quad (2.2)$$

2.2.4.1 Learning a calibration factor

A calibration factor is estimated on a *heldout* set out anchored patients \mathcal{P} as:

$$C = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \sigma(w\tilde{x}^{(p)} + b). \quad (2.3)$$

This calibration factor is the average likelihood of belonging to the positive class assigned by the learned model to anchored patients, which are known to be positive because of the high positive predictive value condition. It is an estimator of $P(A = 1|Y = 1)$, the likelihood of the anchor turning on, if the true label is on.

2.2.4.2 Test-time procedure

The test time procedure (Equation 2.4) is broken into two sub-cases, depending on whether an anchor is present or not. If the anchor is present, the prediction is 1, because of the high positive predictive value condition. If the anchor is absent, we fall back on the learned classifier, scaled by the calibration coefficient.

$$P(Y = 1|x) = \begin{cases} 1 & \text{Anchor} = 1 \\ \frac{(1-C)}{C} \exp(wx + b) & \text{Anchor} = 0. \end{cases} \quad (2.4)$$

To obtain a *ranking* classifier which does not output calibrated probabilities, it is sufficient to use $wx + b$ when Anchor=0, since the scaling factor of $(1 - C)/C$ is a positive constant that applies to all patients equally and the exp function is monotonic.

2.2.5 Derivation

The theory behind our use of anchors in learning is based on Elkan and Noto (2008). Here we present a short derivation to justify the anchor conditions presented in Section 2.2.3.1,

and the relationship between classifiers trained on the anchors $P(A|\tilde{X})$ and classifiers trained on the true labels $P(Y|X)$.

$$P(A = 1|\tilde{X}) = P(A = 1|Y = 1, \tilde{X})P(Y = 1|\tilde{X}) \quad (2.5)$$

$$\begin{aligned} &+ P(A = 1|Y = 0, \tilde{X})P(Y = 0|\tilde{X}) \\ &= P(A = 1|Y = 1, \tilde{X})P(Y = 1|\tilde{X}) \end{aligned} \quad (2.6)$$

$$= P(A = 1|Y = 1)P(Y = 1|\tilde{X}) \quad (2.7)$$

$$= CP(Y = 1|\tilde{X}) \quad (2.8)$$

The first equation simply introduces a latent variable Y and marginalizes over its value. The second equation uses the assumption of high positive predictive value ($P(A = 1|Y = 0) \approx 0$) to remove the second term in the sum. The third equation uses the conditional independence assumption to remove \tilde{X} from the conditioning, yielding a constant correction term $C \equiv P(A = 1|Y = 1)$ such that $\frac{1}{C}P(A = 1|\tilde{X})$ is equal to $P(Y = 1|\tilde{X})$.

To incorporate conditioning on A into the predictor (i.e., transform from $P(Y = 1|\tilde{X})$ to $P(Y = 1|X)$), we break the prediction into two cases:

$$P(Y = 1|X) = \begin{cases} 1 & A=1 \\ \frac{(1-C) P(A=1|\tilde{X})}{C P(A=0|\tilde{X})} & \text{otherwise,} \end{cases}$$

where $P(A|\tilde{X})$ is the output of the classifier trained in Section 2.2.4.

The first case comes from the high positive predictive value assumption. The second case comes from the following derivation:

$$\begin{aligned}
P(Y = 1|A = 0, \tilde{X}) &= \frac{P(A = 0|Y = 1, \tilde{X})P(Y = 1|\tilde{X})}{P(A = 0|\tilde{X})} \\
&= \frac{P(A = 0|Y = 1)P(Y = 1|\tilde{X})}{P(A = 0|\tilde{X})} \\
&= \frac{(1 - C) P(A = 1|\tilde{X})}{C P(A = 0|\tilde{X})}.
\end{aligned}$$

The first line of the above derivation comes from applying Bayes rule while maintaining the conditioning on \tilde{X} . The second line comes from the conditional independence assumption. The third line substitutes $1 - C \equiv P(A = 0|Y = 1)$ and $\frac{1}{C}P(A = 1|\tilde{X}) = P(Y = 1|\tilde{X})$ (from Equation 2.8). Previous published works (Halpern et al., 2014, 2016) use a slightly different form of Equation 2.4, which does not explicitly account for the observation $A = 0$ in the second case. This procedure does not change the ranking performance of the classifiers, but does not take advantage of extra information available from observing A at test time.

The correction term C can be estimated by finding a representative population of positive cases, \mathcal{P} , and taking the average prediction of the model (Elkan and Noto, 2008).

$$P(A = 1|Y = 1) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} P(A = 1|\tilde{x}^{(p)}). \quad (2.9)$$

The anchor assumptions allow us to use a set of examples with positive anchors as a representative set of positively labeled patients.

Many other frameworks for learning with noisy labels exist (e.g., Natarajan et al. (2013); Platanios et al. (2014); Reed et al. (2014); Sukhbaatar et al. (2014)), and could each be useful for different settings. Cotraining also has many similarities to the anchor framework (Blum and Mitchell, 1998), and indeed training with anchors can be seen as one step of a co-training algorithm where one view is the anchor, and the other view is all other features.

2.2.6 Implementation Details

We implement the learning in python using scikit-learn and scipy.sparse libraries. X is a csr-matrix. Every time a new anchor is added or subtracted, we add or subtract a column to a mask matrix M which is implemented as a dok-matrix. These columns are cached when X is generated and can be retrieved quickly rather than iterating through the X matrix which is not in a format suitable for column slicing. When bigrams are specified as an anchor, their component unigrams are also masked-out for instances that have the bigram. Learning is then done on the $X - M$ matrix as the masked feature vector, and the logical OR of M 's columns as the “label” vector.

2.2.7 Assessing a model

How do we know if we've learned a good model? Should we add more anchors? Should we remove one of the anchors that was previously added?

We can assess whether an anchor truly has high positive predictive value by filtering for patients with the anchor and manually assessing whether they are truly positive cases. For example, without inspection, aspirin could mistakenly be taken for an anchor for a patient having a cardiac etiology to their emergency department visit. It is standard clinical practice to give aspirin to a patient when a cardiac etiology is suspected. However, aspirin is also given in patients with a suspected stroke. By reviewing a list of recently anchored patients, a user can quickly determine how specific an anchor is for the phenotype of interest.

To assess the conditional independence assumption, we look at highly weighted features. If a feature is highly weighted, it is highly indicative of the anchor, but this may not necessarily be because of the underlying disease. It may also be because the conditional independence assumption is being broken. Are these highly weighted words often collocated with the anchor? If so, the correlation may come from linguistic patterns (e.g., n-grams) rather than

being caused purely by the presence of the disease of interest. In this case, we can mitigate the violation of conditional independence by censoring words in a fixed window around the anchor. Do they have another reason to be correlated with the anchor? For example, if the anchor is a code for a dispensed drug, the feature associated with the name of the drug itself appearing in a note may be highly weighted. These two are not conditionally independent. The drug name can be added as an anchor as well, or can be removed as a feature before relearning.

Both of the anchor conditions will be violated to some extent in real data. We can also qualitatively assess the learned model by using it to rank patients (that were held-out from learning) and look at patients with high rank, but no anchor. These patients are representative of the patients that the classifier thinks are positive cases. Do they represent the expected phenotype or has the algorithm learned a different-but-related phenotype, or a mix of multiple phenotypes?

2.2.8 Anchor suggestions

We incorporate a second form of feedback to the user in the form of “suggested anchors”, by learning a second logistic regression classifier to predict the anchor, this time with high L1 regularization. This heuristic yields a small number of features that are highly associated with the anchor, and may have the requisite positive predictive and conditional independence properties. Whether to include a suggested anchor (e.g., by creating a union of anchors) is the choice of the modeler. It may help, by detecting more positive cases to learn from and at test time, but it can also dilute the positive predictive value and conditional independence conditions. Figure 2.2 shows an example trace of a user adding and subtracting anchors, and the corresponding changes in performance of a classifier learned using the anchors.

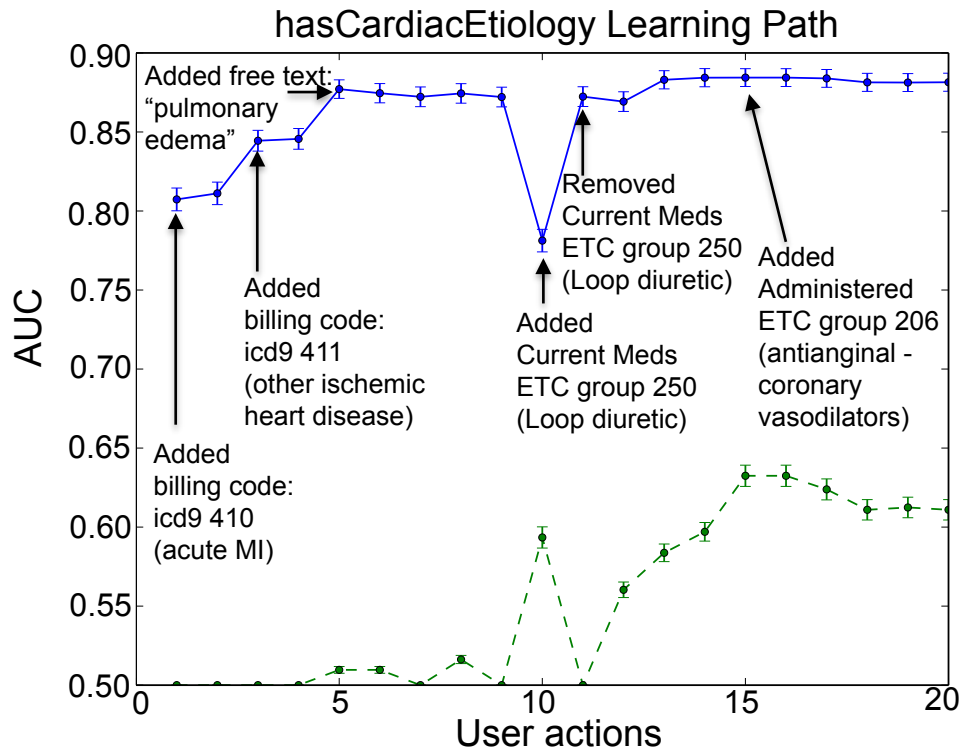


Figure 2.2: Adding a subtracting anchors: Changes in the area under the receiver operator curve (AUC) as a physician user adds and subtracts anchors to train a classifier for cardiac etiology. The blue solid line shows the performance of the learned classifiers. The green dotted line shows the performance of a naive prediction strategy that simply looks for anchors. Gold standard labels are obtained from questions asked to physicians at disposition time (Section 2.3.4).

2.2.9 Code available

Code is available to review records and learn anchored phenotype models at <http://clinicalml.org>. Figure 2.3 shows a screenshot of the original anchor learning tool. A more modern, web-based version is under development and will be released on the same site.

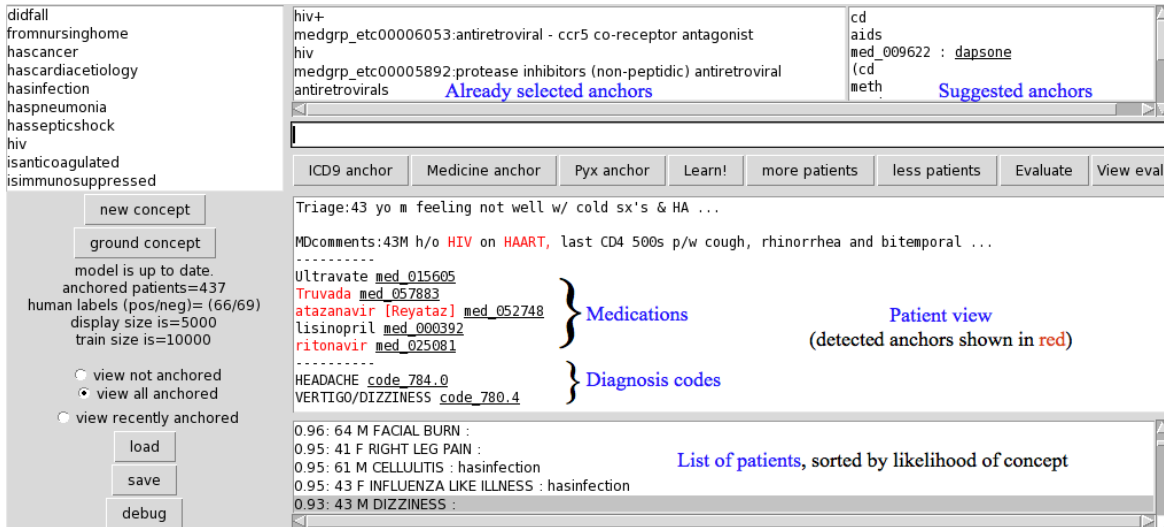


Figure 2.3: A screenshot of the anchor elicitation tool using deidentified patient information. Conditional highlighting emphasizes the presence of previously specified anchors in a note so that the physician can determine quickly whether its usage is as expected.

2.3 Evaluation: Phenotype estimation in the emergency department

2.3.1 Study Design

We conducted a retrospective observational study to build and test a collection of clinical state variable predictors. The study was approved by the institutional review board at Beth Israel Deaconess Medical Center in Boston, MA.

2.3.2 Setting and Selection of Participants

The study was performed in a 55,000 visits/year trauma center and tertiary academic teaching hospital. All consecutive ED patients between 2008 and 2013 were included. Each record represents a single patient visit. No patients were excluded, leading to a total of 273,174 records of emergency department patient visits.

	Representation	Dimension
Age	Binned by decade	11
Sex	M/F	2
Medication History	Indicators by Medication Generic Sequence Number (GSN)	1,947
Medication Dispensing Record		279
Triage Vitals	Binned by decision tree	77
Lab results		2,805
Triage Assessment	Binary bag-of-words	7,073
MD Comments		8,909

Table 2.2: Features used to build binary patient description vectors

2.3.3 Data Collection and Preparation

As input for classification tasks, we build a patient feature vector with binary features by concatenating 8 smaller sparse feature vectors built from the data sources described in Table 2.2.

Medication history refers to the medications the patient was taking prior to the ED visit as documented on the patient’s medication reconciliation form. Medication Dispensing Record is documented on the hospital Omnicell and Pyxis Medication dispensing systems. Triage assessment refers to the free text nursing assessment documented by the nurse at triage. Generic sequence numbers (GSN) are associated with medications using the First Data Bank Drug database. MD Comments refers to the free text scratch space used to track a patient’s course that is updated in real-time. All of these data elements were recorded electronically at the same time that the data was collected.

The free text fields, Triage Assessment and MD Comments, were preprocessed with simple bigram and negation detection before being represented as a binary bag-of-words. A more detailed description of the text processing is available in Appendix B.2.2. Features that appear in fewer than 50 patient records are discarded, leaving a final concatenated feature vector with 21,103 dimensions.

Phenotype	Disposition Question	N	Pos
Cardiac - acute	In the workup of this patient, was a cardiac etiology suspected?	17,258	0.068
Infection – acute	Do you think this patient has an infection? (Suspected or proven viral, fungal, protozoal or bacterial infection)	62,589	0.213
Pneumonia – acute	Do you think this patient has pneumonia?	9,934	0.073
Septic Shock - acute	Is the patient in septic shock?	6,867	0.020
Nursing home – history	Is the patient from a nursing home or similar facility? (Interpret as if you would be giving broad spectrum antibiotics)	36,256	0.045
Anticoagulated - history	Prior to this visit, was the patient on anticoagulation? (Excluding antiplatelet agents like aspirin or Plavix)	1,082	0.047
Cancer - history	Does the patient have an active malignancy? (Malignancy not in remission; and recent enough to change clinical thinking)	4,091	0.042
Immunosuppressed - history	Is the patient currently immunocompromised?	12,857	0.040

Table 2.3: Phenotype variables used for evaluation. The text in the Disposition Question column was shown to physicians at the end of patient disposition. The parenthetical text was shown if physicians selected a click-through option for additional information. N gives the number of labels collected while Pos gives the fraction of positively labeled cases.

2.3.4 Gold-standard Phenotype Labels

We evaluate the phenotype learning framework using 8 phenotype variables relevant in the emergency department. For evaluation we prospectively collected gold-standard labels. Physicians were prompted upon patient disposition to provide gold-standard responses to questions from a rotating pool of research questions used in the emergency department. The phenotypes for which gold-standard labels were collected are listed in Table 2.3 They include both acute conditions such as whether a cardiac etiology is suspected for this patient visit, and historical phenotypes such as whether a patient is immunosuppressed. Responses were recorded on a Likert scale from 1-5. We take 4 and 5 to be positive and everything else to be negative.

2.3.5 Anchor elicitation and phenotype choices

A single emergency physician specified anchors for phenotypes using the custom interactive anchor elicitation tool described in Section 2.2.9.

The initial 42 phenotypes selected for this pilot study were chosen because of their relevance in the emergency department. Broadly, they fall into six different categories and use cases. Phenotypes for which gold-standard labels are available are italicized.

Change clinical thinking: Some phenotypes change baselines and normal values for lab results and vital signs. Others put the patient at heightened risk for complications. In both cases, they are important to the clinician’s thinking and can lead to errors in diagnosis if missed. These phenotypes include: *active malignancy*, *immunosuppressed*, *nursing home*, alcoholic, liver damage.

Avoiding medical errors: Certain populations require special attention to avoid causing them further harm during their hospital stay. These phenotypes include: *anti-coagulated*, *immunosuppressed*, deep vein thrombosis, HIV positive, syncope, and suicidal ideation.

Emergent conditions: Some conditions require immediate attention, and the faster they are noticed and treated, the better. These phenotypes include: *septic shock*, allergic reaction, intracranial hemorrhage, cerebrovascular accident, severe sepsis, small bowel obstruction, motor vehicle accident, bicycle accident, epistaxis, gastrointestinal bleed.

Diagnostic support: Some phenotypes are common, but require attention to rule out dangerous etiologies. Examples of these phenotypes are: abdominal pain, headache, and back pain.

Require specialist attention: Some conditions require the attention of a specialist outside of the emergency department. These phenotypes can be used to proactively alert staff to schedule consults early. These phenotypes include: *cardiac etiology*, psychiatry, sexual

assault.

Clinical pathways / Standardized order sets: Our clinical collaborators are in the process of building standardized treatment paths or order sets for a wide range of phenotypes. These phenotypes include: *infection*, hematuria, chest pain, cellulitis, pneumonia, urinary tract infection, congestive heart failure, asthma/COPD, diabetes, gastroenteritis, cholecystitis, pancreatitis, ankle fracture, employee exposure, kidney stone, and laceration.

2.3.6 Phenotype Evaluation

Area under the receiver-operator curve (AUC) and precision-recall curves were evaluated using the prospectively gathered gold-standard labels. When evaluating the supervised method, 10-fold cross-validation was performed to allow for testing on the full set of gold-standard labeled patients. Standard errors in AUC for anchor-based learning are evaluated using 100 bootstrap samples from the test set. Standard errors in AUC in the supervised method are calculated across the folds of the 10-fold cross-validation.

2.3.7 Real-time Setting

To evaluate the effectiveness of phenotype prediction in a real-time setting, we performed a retrospective analysis of patient records applying our phenotyping algorithms to snapshots of the patient records as they appeared at 0, 30, 60, 120, 180, and 360 minutes after arrival to the emergency department, as well as at the time of disposition from the emergency department. We compare phenotype extraction using classifiers learned using the “Anchor and Learn” framework (Section 2.2) with 200,000 patients against fully supervised classifiers trained using 5000 patients labeled using the gold-standard data. Within the Anchor and Learn framework, patients with a positive anchor all receive an equal score of 1. We first break ties by counting the number of distinct anchors present in the patient’s record, then by

the output of the learned classifier. Both the anchor-based classifiers and the gold-standard classifiers are trained for each time step independently, using only the data available up to that time, yielding 7 different classifiers for each method. In the discussion (Section 2.5.4), we discuss a different approach to dealing with data changes over time, that involves learning a different classifier for each possible pattern of data “missingness”.

AUC gives a sense of how the classifiers perform over a range of thresholds and is useful to compare different methods of learning the classifiers. To determine utility for real-life usage, we also compute precision-recall curves for the anchor-based classifiers.

2.3.8 Performance Breakdown by Data Type

To better understand the contributions of different data types in the EMR, we trained classifiers using only subsets of the EMR data types. In all cases, we allow all of the classifiers to use age, sex, and triage vitals, and then measure performance using AUC at disposition time with classifiers that additionally use medication history, medication dispensing record, lab results, triage text, and MD comments. We also look at classifiers that use all structured data (medication history + medication dispensing record + labs) and all free text data (triage text + MD comments), and finally compare to the classifiers that use all of the above mentioned data types.

2.4 Results

2.4.1 Building a Phenotype Library

We built phenotypes of immediate relevance in the emergency department (Section 2.3.5). Each phenotype is defined by a small number of anchors (Table 2.1), which are used to learn logistic regression classifiers as described in Section 2.2. Highly weighted terms learned by the

classifiers are shown in Table 2.4. Building each phenotype took approximately 10 minutes of physician time spent using the anchor elicitation interface. The full list of phenotypes and their definitions is available on github: <https://github.com/clinicalml/clinical-anchors>.

Table 2.5 shows the precision and recall of the anchors for the eight phenotypes that were evaluated. Unlike the real-time evaluations in the remainder of this section, these results show the characteristics of the anchors retrospectively, as they were used for training. As such, these results include diagnosis codes, which cannot be used for real-time phenotype estimation. The precision values are all less than one, partly because it is difficult to specify anchors with perfect positive predictive value, but also because the gold standard clinical evaluation occasionally missed positive cases.

2.4.2 Phenotype Evaluation in Real-time Setting

For all of but one of the phenotypes (nursing home), the Anchor & Learn framework outperforms supervised training on a set of manually collected gold-standard labels and a naive strategy of simply searching for the anchors.

Figure 2.4 shows a comparison between the three methods for learning phenotypes as a function of time. Some conditions are easier to detect than others, with highly acute conditions like pneumonia and septic shock reaching AUC values above 0.95. For 5 out of 8 phenotypes, the manual rules method has an AUC below AUC=0.8 from arrival to disposition.

Changes to a patient’s EMR happen multiple times over the course of a patient visit and different pieces of information become available at different times. Medication reconciliation usually happens in the first 30 minutes of a visit and lab results tend to become available between 1.5 hours to 2 hours after patient arrival. However, there are a significant number of updates to these fields that occur after that peak time. MD comments and dispensed

Phenotype	Data source	Observed Feature	Weight
Diabetes (history)		dm	2.97
		Blood glucose testing	2.92
		dm2	2.23
		GLUCOSE (>266.5)	2.1
		MetFORMIN (Glucophage)	1.98
		iddm	1.87
		GLUCOSE (198.5-266.5)	1.8
		dmii	1.72
		diabetes	1.56
		Fingerstick lancets	1.47
		diabetic	1.42
		Blood glucose testing	1.25
		diabetic	1.22
		hypoglycemia	1.22
		iddm	1.19
		bs	1.16
		Insulin HumaLOG	1.16
		GLUCOSE (175.5-198.5)	1.13
		Tricor	1.1
		dm1	1.1
Employee exposure		needle	1.9
		Pain(<0.05)	1.47
		LaMIVudine-Zidovudine (Combivir)	1.41
		or	1.36
		stuck	1.13
		exposure	1.06
		neg:bleeding	1
		washed	0.98
		went	0.96
		Temp (98.98-99.21)	0.95
		cath	0.94
		ept	0.93
		glove	0.91
		dirty	0.81
		sq	0.8
		thumb	0.77
		patient	0.77
		needle	0.73
		Heart Rate (61.5-66.5)	0.72
	Allergic reaction		id
		DiphenhydRAMINE	1.43
		benadryl	1.13
		MethylPREDNISolone Sodium Succ	1.09
		DiphenhydRAMINE	1.05
		Famotidine	0.89
		benadryl	0.88
		neg:hives	0.86
		throat	0.79
		PredniSONE	0.73
		itching	0.72
		neg:sob	0.71
		swelling	0.7
		neg:rash	0.66
		Famotidine (PO)	0.63
		iv	0.63
		allergy	0.58
	feeling	0.52	
	ate	0.52	
	hives	0.51	
	rash	0.51	

Legend:	Color/Pattern	Source
	Yellow diagonal lines	Triage assessment
	Green	MD comments
	Pink vertical lines	Medication History
	Purple vertical lines	Medication Dispensing Record
	Dark green	Triage vitals
	Blue horizontal lines	Lab results

Table 2.4: Top 20 weighted terms in the classifiers for three of the learned phenotypes. These classifiers are learned using medical records as they appear at time of disposition from the emergency department.

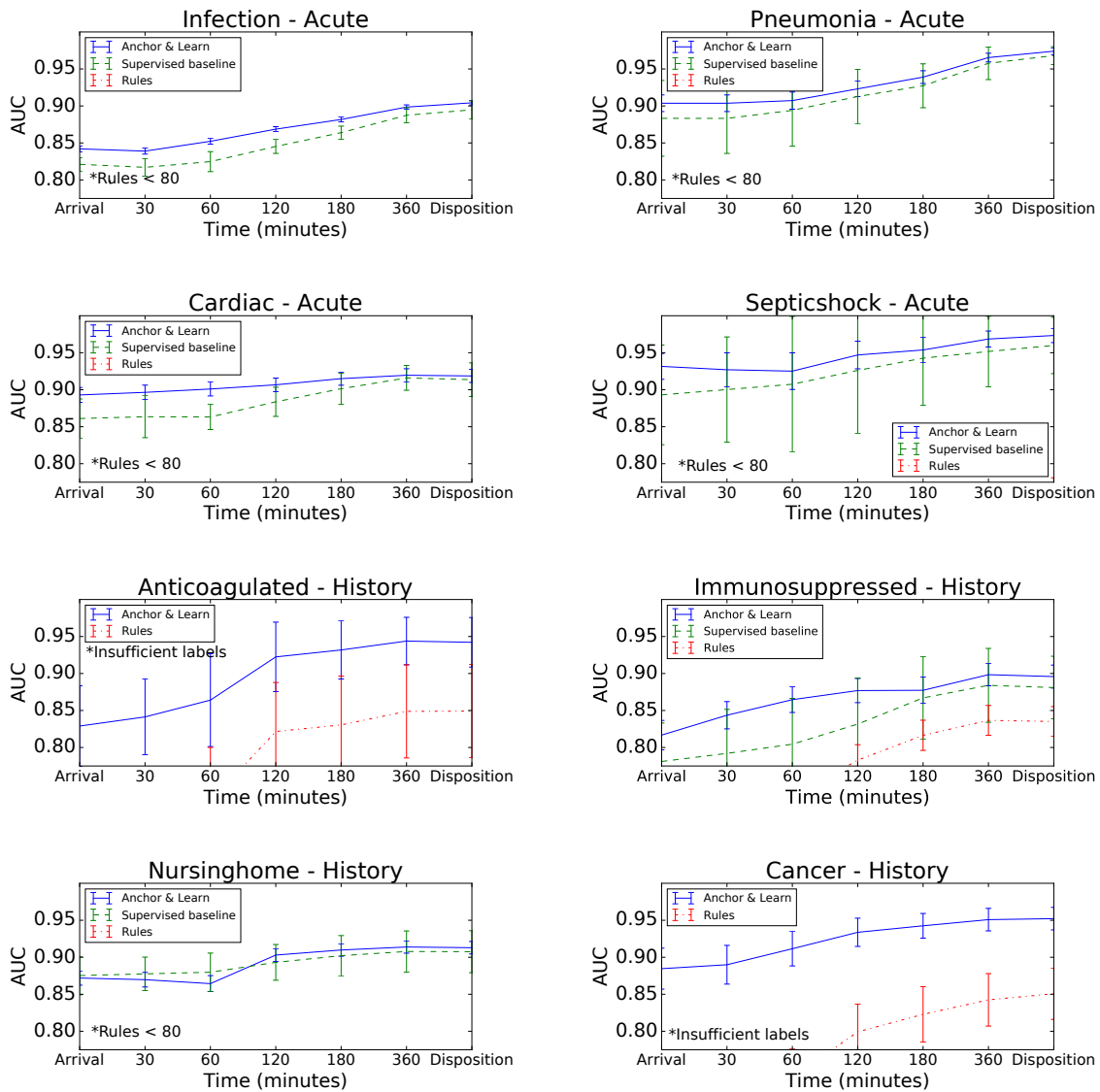


Figure 2.4: Comparison of performance of phenotypes learned with 200,000 unlabeled patients using the semi-supervised anchor based method, and phenotypes learned with supervised classification using 5,000 gold standard labels. Error bars indicate $2 * \text{standard error}$. “Insufficient labels” means that we did not have enough labeled patients to train the fully supervised baseline. “Rules < 80” means that the rules baseline was consistently below 80 so it does not appear in these plots.

Table 2.5: Precision and recall of the anchors used for training (includes diagnosis codes).

Phenotype	Precision	Recall
Cardiac - Acute	0.55	0.36
Infection - Acute	0.70	0.74
Pneumonia - Acute	0.86	0.67
Septic Shock - Acute	0.53	0.5
Nursing Home - History	0.68	0.22
Anticoagulated - History	0.54	0.73
Cancer - History	0.41	0.78
Immunosuppressed - History	0.22	0.77

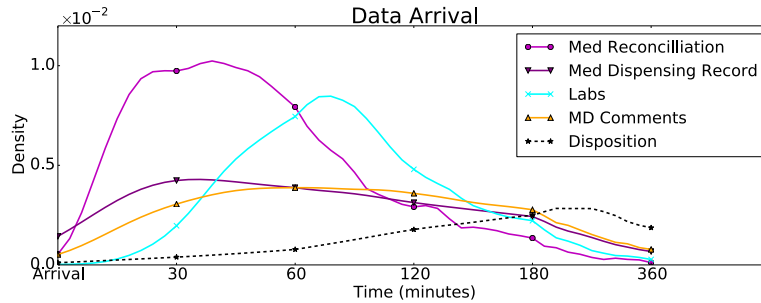


Figure 2.5: Changes to patient records over time. The time of every change to the patient record is recorded (measured in minutes from arrival) and a non-parametric kernel density estimator is used to plot the distribution of times at which changes occur.

medications are constantly being updated. The median visit is about 5 hours in length. Figure 2.5 shows the distribution of when changes occur in the EMR, accumulated over 20,000 patient visits.

At the beginning of the patient’s visit, phenotype decisions are dominated by the triage time information from age, vitals and triage note. As time progresses, MD comments, labs and dispensed medications become more important in determining the patient’s phenotype. Figure 2.6 shows features picked up by the learned classifiers as time progresses, using the pneumonia phenotype as an example. The stacked bars show the relative influence of each data type on classification (see Appendix B.2.5 for details of influence measure). For the pneumonia phenotype, the least important factor is the medication history; for other phenotypes, such as anticoagulation, it is much more prominent. The text on Figure 2.6

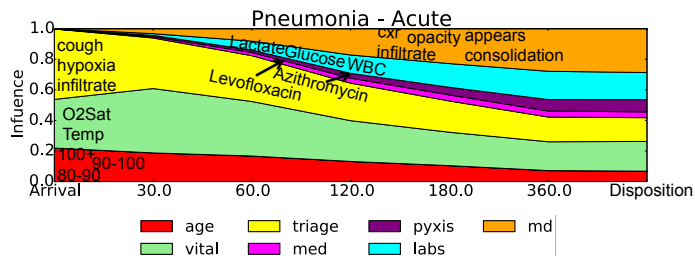


Figure 2.6: Influence and highly changing features for the pneumonia phenotype extractor as a function of time.

shows features whose weights have significantly increased, with the position on the x-axis indicating approximately when they start becoming important for prediction.

2.4.3 Precision-recall operating characteristics

Figure 2.7 shows the precision-recall curves calculated for the classifiers learned at triage time, 1 hour after triage, and disposition. We report recall values at PPV (positive predictive value) of 0.75 and 0.5 because these are reasonable PPV values for alerts and conditional displays respectively. The precision-recall of the manual rules system (i.e., ordering by number of distinct anchors present in the record) is denoted by a circle.

At triage time, acute conditions such as general infection and pneumonia can be detected at PPV=0.75 with a recall of 0.32 and 0.11 respectively. At PPV=0.5, we can achieve recall of: Pneumonia 0.48; cardiac etiology 0.47; septic shock 0.20. Simple heuristics like counting distinct anchors do not work well on their own in this setting, since at triage time, the anchors, which generally consist of structured data such as medications or diagnosis codes, are not available.

After one hour, general infection with a PPV=0.75 has a recall of 0.35 (compare to rules: PPV=0.71, recall=0.05). Using PPV=0.5 for the other acute phenotypes, the recall values rise to: Pneumonia 0.55; cardiac etiology 0.48; septic shock 0.35.

At disposition time, medication and text anchors are often present in the patient note,

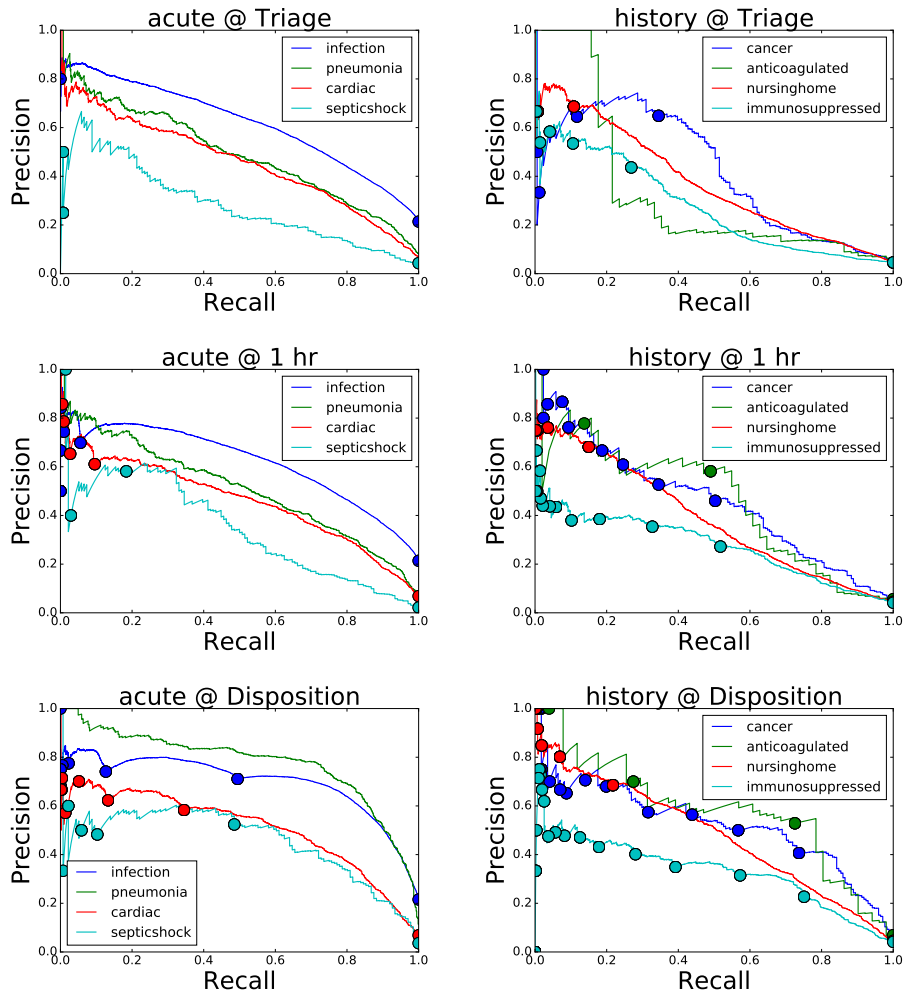


Figure 2.7: Precision-recall curves calculated for each of the phenotypes at triage time, 1 hour after triage, and disposition. Circles indicate operating characteristics achievable by thresholding on the number of anchors present.

making the rules-based classifiers fairly effective. One notable exception is the pneumonia phenotype which is based exclusively on anchors from diagnosis codes, which are not populated in the patient record until later. Even so, the learned classifiers can be used to increase recall in the general infection phenotype at disposition time from 0.5 to 0.7 without reducing precision below 0.7. Similarly, for cardiac etiology, the classifiers can be used to move the operating characteristic from (PPV=0.58, Recall=0.34) to (PPV=0.5, Recall=0.67).

After one hour, historical conditions become relevant for decision making. With a positive predictive value of 0.5, recall values are: nursing home 0.37 (compare to rules: PPV=0.68, recall=0.15), cancer 0.70 (compare to rules: PPV=0.5, recall=0.5), anticoagulated 0.57 (not that different from rules: PPV=0.58, recall=0.5). The immunosuppressed phenotype has very low recall with precision of 0.5 (recall=0.04) and therefore would likely not be put into use without further improvement. One factor which likely contributed to the poor performance of the immunosuppressed classifier was the low positive predictive value of its anchors (Table 2.5). This may have been due to a difference between the intent of the clinician who specified the anchors to include anyone who was at risk of being immunosuppressed, and the understanding of the labeling physicians who were asked at disposition time “is the patient currently immunocompromised”, which could have been interpreted more restrictively.

2.4.4 Performance Breakdown by Data Type

For phenotypes based on patient history (Immunosuppressed, Nursing home, Anticoagulated, and Cancer), medication history is the most important structured data type, and structured data is more important than free text for all but the cancer phenotype.

For phenotypes that represent acute problems (Infection, Pneumonia, Cardiac etiology, and Septic shock), the medication dispensing record is the most useful data type among the structured records, and free text tends to be more informative than structured data. Septic

shock is an exception, where the medication dispensing record is more informative than the free text.

For all phenotypes, combining free text and structured data was more informative than either of the two on its own.

Figure 2.8 shows change in AUC from baseline as a function of the data types used for classification. The baseline uses only age, sex and vital signs.

2.5 Discussion

2.5.1 Utility of the learned classifiers

Based on the uses discussed in Section 1.3.4, we can use precision-recall curves (Figure 2.7) to assess the potential utility of each of the models for alerts, reminders and displays. The gold-standard label collection process for these experiments does not provide a complete labeling for any patient (see Section 2.3.4), so we do not consider inter-phenotype ranking measures like top-5 phenotype suggestions. These are evaluated using a different set of gold-standard labels in Chapter 3.

The infection and pneumonia classifiers have reasonable recall at PPV=0.75 at triage time (0.32 and 0.11 respectively), and improve as the visit progresses, suggesting that these could be useful for non-interruptive reminders and alerts without too much fear of alarm fatigue.

Cardiac etiology and septic shock are acute conditions. With a PPV=0.5 at triage time, we can use them for contextual display changes, such as displaying standardized order sets on the side-bar to make navigation easier, but since they are incorrect almost half the time, actively bringing alerts to the clinician's attention would not be recommended. Similarly, the phenotypes related to patient history such as anticoagulated, nursing home, and active malignancy could be used after an hour in the ED, when much of the decision-making and

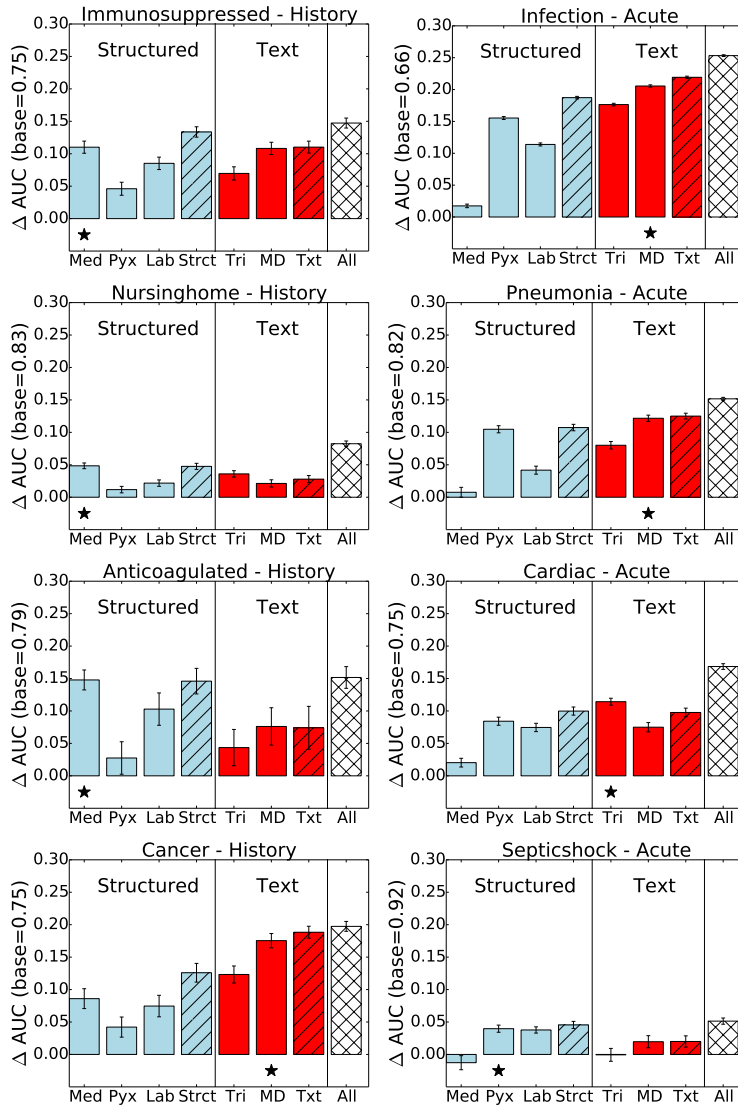


Figure 2.8: Additive change in AUC from baseline for phenotype extraction as a function of the features used. The baseline phenotype extraction uses only features from Age, Sex and Triage vitals and its value is indicated for each phenotype on the y-axis label. Blue bars indicate structured data while red bars indicate free-text data. Hatched lines represent a combination of features. A star is placed below the single feature that has the highest performance. From left to right, the classifiers use:

- Med Medication history (prior to visit)
- Pyx Medication dispensing record (during visit)
- Lab Laboratory values
- Strct All Structured data (Med + Pyx + Labs)
- Tri Triage Nursing Text
- MD Physician Comments
- Txt All Text (Tri + MD)
- All All features (Structured + Text).

planning for the patient occurs.

Compared to simply searching for the anchors themselves, the learned classifiers provide value in the early parts of the patient visit when anchors may not yet be present in the patient record (this is especially relevant when the anchors come from diagnosis codes which are generally not filled in until after the patient leaves the emergency department).

This assessment of overall utility, even at the single site under consideration, is limited by two factors. First, these analyses are retrospective, and more experimentation with live implementations would be required to understand whether these alerts and display changes actually provide value to clinicians, either by reducing missed conditions or streamlining workflows, and whether they lead to improved outcomes. The binary evaluation does not take into account the possibility that a phenotype may represent a *relevant* consideration in clinical decision making, even though it is eventually ruled out.

A second limitation is that for each patient, the gold-standard labels are obtained from a single physician at disposition time and different physicians contributed labels over time. While we believe this method is more reliable than using administrative codes such as ICD9 diagnosis codes, we do not have a measure of inter-rater agreement to determine the accuracy of the labels. A measure of inter-rater agreement and modeling individual effects of the labeling physician, could help determine the accuracies of the reported values of PPV, which we suspect are actually higher than reported here.

2.5.2 Effectively using unstructured data

The classifiers presented in Table 4 use both structured and unstructured data to determine whether the patient has the phenotype, and generalize beyond the initial anchors input by the physician in Table 3. For example, the classifiers learn to look for appropriate medications used to treat for allergic reaction (e.g. steroids like prednisone and methylprednisolone,

and antihistamines like diphenhydramine and famotidine). They also naturally pick up on variations of free text and statistical synonyms without having to specify this information manually. For example, in the classifier for diabetes, we see dm, dm2, dmii, or in the classifier for cholecystitis (not shown in Table 4) we see both surg and surgery.

The classifier for employee exposure makes heavy use of textual terms, each of which are not strongly indicative on their own, but often appear together in the narrative used to determine the risk of HIV and hepatitis transmission after an employee exposure, including the location (thumb), circumstance (operating room, cath), mechanism of injury (needle), barriers (glove), and decontamination (washed).

Important structured data tends to be repeated in the MD comments, so using only free text, without structured data beyond demographics and triage vitals, tends to perform well. One important exception to that trend is determining whether a patient is anticoagulated, which represents an important piece of background information regarding the patient, but may not be pertinent to the patient's current illness. Nursing home is better detected from the triage note, as it is often included in the triage assessment and then dropped in the MD comments if it is deemed irrelevant for the patient's current problem.

2.5.3 Temporal aspects of the classifiers

The plots in Figure 2.4 show that classification becomes more accurate as the patient visit progresses, which makes sense since more informative features become available. The more general phenotypes like infection and cardiac etiology show the least improvement over the course of the visit, as they are often clear from the patient's initial complaint and presentation. In fact, we find that the single most important data type in determining cardiac etiology is the free text written at triage. More specific diagnoses, like pneumonia, become increasingly easy to determine as the visit continues. The progression of classification performance generally

mirrors when significant data items become available. For example, determining whether a patient is on anticoagulation therapy improves dramatically 30-60 minutes after triage, corresponding to the times when medication history and lab results become available, as seen in Figure 2.5.

The gaps between our method and supervised training in Figure 2.4 are larger towards the beginning of the visit when there is less information available. By learning weights in a statistical classifier and using a large amount of data, we allow for evidence to accumulate (for example swelling is indicative of allergic reaction, but can occur for many reasons), making a continuous-valued prediction based on the accumulated evidence rather than making decisions based on individual words or phrases in the note. This advantage is more pronounced towards the beginning of the patient’s visit, when there are fewer obvious cues to pick up on. Clinical decision support is most useful early in a patient’s emergency department course, when timely interventions can change clinical trajectories and before critical decisions are made. The performance improvement between our method and supervised training is therefore critical to our intended use case of real-time clinical decision support.

2.5.4 Alternative methods of handling missing data

In Section 2.3.7 we describe how we use different classifiers, each trained for a different time step to handle the temporal aspect of how data becomes available in a patient record. This method could potentially perform poorly for patients who follow a different time course than the standard patient, either having their data populated more rapidly or more slowly.

Another possible method is to train classifiers on subsets of the data, for example, only triage information and medications, but not labs. That way, depending on the missingness pattern of the patient record, we can use the appropriate classifier to estimate phenotype variables. Figure 2.9 shows how classifiers trained specifically on subsets of the data perform

better than classifiers trained on all the data when data is artificially dropped from the feature vector to simulate the scenario where data is not yet available. For example, for the nursing home phenotype, a subset classifier trained only on triage note information achieves an AUC of 0.884, while a classifier trained on all of the data only has an AUC of 0.829 when only the triage note is available. The different paths through the trellis are different possible courses that a patient could take; some patients have their labs populated before their medications, and others vice versa.

One drawback to this method is that it assumes that each data type is filled in all at once. In fact, some laboratory results are much faster than others (e.g., arterial blood gas can be measured at the bedside, while bacterial cultures can take days to grow). Similarly, the MD comments field is a free text space where physicians continuously update the patient's course. This can be edited multiple times in a visit, so the notion of being missing or present does not fully apply (before a physician has seen the patient at all, it does make sense to consider it missing).

2.5.5 Statistical models – interpretability and failure modes

The statistical model sometimes puts high weight on features that might not at first be intuitive. For example, the negation of hives is indicative of allergic reaction. Although this is counter-intuitive from a clinical perspective, this does make sense from statistical perspective because physicians are only likely to document the absence of certain findings when it is pertinent to a particular condition. In medicine, these terms are known as pertinent negatives and often matter as much as pertinent positives.

Nursing home is well detected by structured data, particularly by the patient's medication history (see Figure 2.8) since patients from nursing homes are more likely to have very long and detailed medication lists. This is partly because nursing homes, and other assisted

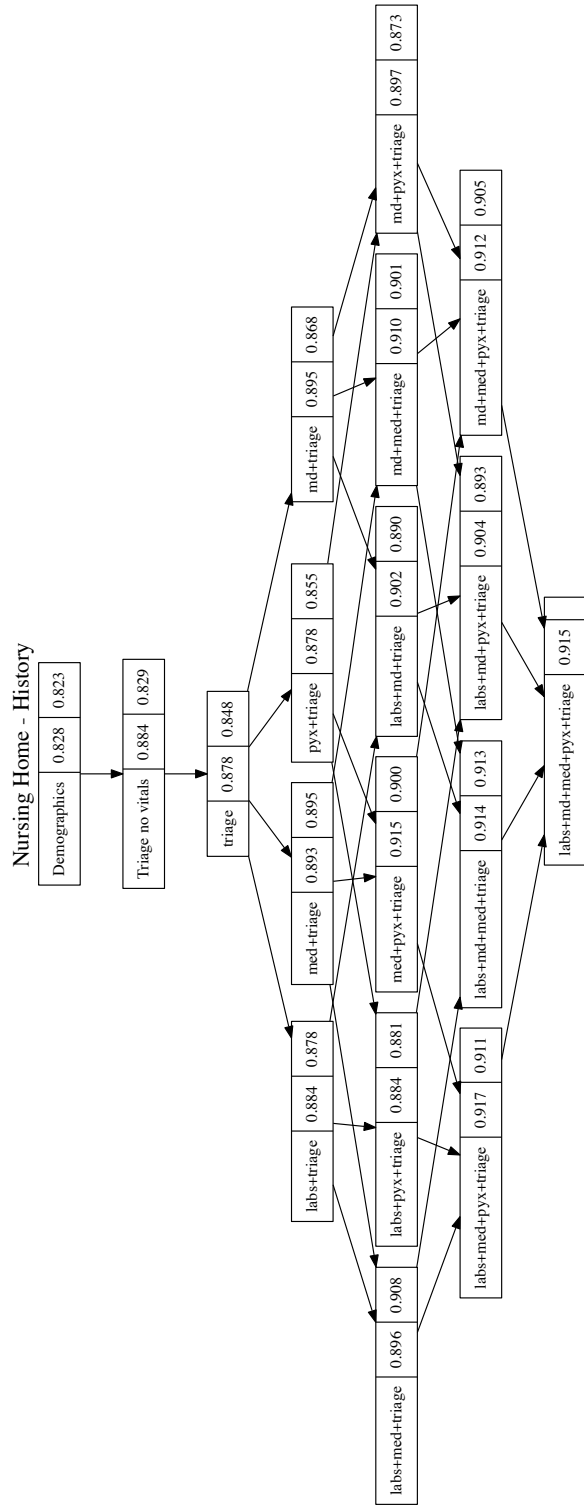


Figure 2.9: Trellis view of predictive accuracy based on subsets of a patient’s record being present. Each node represents a setting where only a subset of the data is available (all other data types are dropped synthetically). Directed edges represent the addition of a new data type. Two values are recorded: (left) The AUC for predictions from a “subset classifier” (Section 2.5.4); (right) the AUC for predictions trained on the full feature set. The data types are as follows:

- Demographics - Age, sex
- Triage - Age, sex, vitals, triage assessment
- labs - laboratory results
- med - medication history (prior to visit)
- pyx - medication dispensing record (during visit)
- md - physician comments.

care settings keep their residents' complete medication lists on file, and send their file along when the patient is brought to the hospital. This is another example of a cue picked up by statistical learning, but which would be difficult to specify a priori as a manual rule.

While these examples of counter-intuitive or unexpected features contribute to the strong performance of the statistical classifiers over manual rules, they can also lead to odd failure modes. For example, neglecting to document a *negative* symptom may lead to a missed phenotype, or over-documentation of irrelevant symptoms may confuse the classifiers and cause them to falsely report a phenotype that is not correct. Alternatively, a nursing home that does not export detailed medication lists, may not be recognized.

2.6 Conclusions

Every patient has a unique history and presentation that must be considered in providing treatment. Currently, that information is captured in the electronic medical record in a form that is difficult to use in applications such as clinical decision support. As our collective understanding of medicine becomes more precise, we would like to represent all of the information in the EMR, including both structured and unstructured data in a fine-grained manner that can be used to provide personalized recommendations and clinical decision support.

We demonstrate a scalable method of building data-driven phenotypes with a small amount of manual input from domain experts in the form of anchor variables that can be shared widely between institutions. The phenotypes are then implemented as classifiers that can be statistically learned from large amounts of clinical data in each institution. We show that phenotypes learned in this way are comparable to phenotypes learned with manually identified cases and controls for use in a real-time setting, and allow us to easily scale our collection of phenotypes, building an initial library of 42 phenotypes with help from a single

clinical collaborator as a proof of concept. Of the eight phenotypes evaluated against a gold-standard, two predictors (general infection and pneumonia) show promise for real-time alerting, and five more (Cardiac etiology, septic shock, anticoagulated, nursing home, and active malignancy), show promise for conditional displays.

2.7 Next steps and open questions

In this work, we only consider data from a single hospital, and even though we were able to specify 42 phenotypes, we were only able to quantitatively evaluate 8 of them. The quantitative assessment relied on questions asked to physicians at disposition time, and have not been compared to a gold-standard manual chart review to determine their reliability. The classifier performance was evaluated retrospectively and disconnected from a particular health IT application. A natural next step would be to evaluate the real life impact of these applications on clinical care.

Data came from a single hospital emergency department, and testing portability of phenotype definitions is a clear next step. In our framework, phenotypes are defined only by anchor variables and then classifiers are learned on each institution’s data independently. We expect this method will allow each institution to learn classifiers that are appropriate to their patient population and local linguistic features.

We compare to simple rule-based classifiers (i.e., searching for the anchor), and show that the anchor-and-learn framework is more effective (Figures 2.4 and 2.7). We do not explicitly compare to other manually defined phenotypes built with community consensus and repetitive refinement such as those in eMerge (Newton et al., 2013) or OMOP (<http://omop.org/HOI>). Two phenotypes (type II diabetes and myocardial infarction) learned with noisy-labels were compared against previously validated rule-based definitions in Agarwal et al. (2016), and showed comparable performance, but further comparisons are warranted. Many of the

validated rule-based definitions use diagnosis codes as part of the definition, which is useful for retrospective cohort selection, but is not available in real-time. Anchors can be useful for quickly developing new phenotypes or adapting them to incorporate new settings where more or fewer data types are available. When a well-established rule-based definition is available, should we train new classifiers or fall back on the established rules? Do established rules from eMerge and OMOP make good anchors?

We show how to learn phenotypes using a small amount of input from domain experts in the form of anchor variables. However, as these phenotypes are put to use driving IT applications, they can be automatically refined through usage, either explicitly by correcting predictions made by the algorithm, or by taking actions like enrolling a patient in a care pathway or using a standardized order set that implies agreement or disagreement with the model’s predictions.

Emergency department time scales are relatively short; other settings, such as intensive care units or longitudinal studies have much longer time scales. In this work, we assumed the patient’s phenotype stayed constant over the course of the visit (the median ED visit is shorter than 5 hours from triage to disposition). Incorporating the evolution of the patient’s phenotype over time requires more attention. Naively, each time-step (e.g., hour, visit) can be modeled as an independent phenotype prediction problem, but it is not obvious how to provide labels or label surrogates such as anchors for each time step. Which data become “stale” or irrelevant and at what time scales? How does one reliably determine quantities such as time of onset, or time of recovery? Henry et al. (2015) use a definition of time of onset in training a classifier to detect sepsis and septic shock, but many phenotypes do not have standard definitions for their time of onset.

Systematically understanding the failure modes of these classifiers is an important step to designing safe systems. How can these models be queried and tested to determine the edge cases in which they fail? Challenging physicians to “Beat the machine” (Attenberg et al.,

2015) by designing challenging queries or finding difficult patients is one route to discovering these edge cases. A related question relates to the human factors aspects of these systems. While the phenotype classifiers are still imperfect, but behave well on common presentations, what are appropriate displays and interfaces that allow the EMR system to help the clinicians perform their jobs? How do we build trust in the system to the point where it is useful without allowing users to become overly reliant on these systems which may miss the rare or atypical presentations?

Using highly expressive classifiers such as deep neural networks allows the models to “overfit” to the anchors, taking advantage of violations in the conditional independence assumption to predict the anchor directly, possibly doing a poor job at predicting the true phenotype label. Simple classifiers have less capacity to take advantage of these violations, but are limited in their expressive power. How do we use high capacity classifiers, while protecting against this overfitting phenomenon? Multiple anchors can be used to regularize the learned classifier. For example, we could train classifiers on each anchor while enforcing that the classifier’s parameters or predictions be close to each other.

What are the minimal requirements to be a good anchor? Conditional independence is a strong assumption; are there weaker assumptions that can achieve the same goal? One avenue for exploration is a comparison to co-training (Blum and Mitchell, 1998), which was initially formulated as requiring independent classifiers, but that requirement was later weakened in Balcan et al. (2004). Learning with anchors can be seen as a co-training setting where the two views are the anchor and all other features. This view could be fruitful in analyzing the anchor learning setting in a different light.

How can anchors be used to create application-specific patient similarity metrics or latent representations? A global similarity metric is difficult to create or even define, because for each query, one can ask “similar with respect to *what?*” A large collection of phenotypes, which a physician manually chooses to include or exclude from the similarity metric, could

be a latent space on which to project patients. Unlike fully unsupervised similarity metrics, the latent space would be interpretable since each dimension corresponds directly to a known phenotype. More generally, the ability to quickly build interpretable latent spaces could be a useful cross-disciplinary data science tool.

Chapter 3

Semi-supervised learning of joint probabilistic models

Acknowledgments This work was joint with Steve Horng and David Sontag. This work was previously published in “Clinical Tagging with Joint Probabilistic Models” Halpern, Horng and Sontag. Machine Learning for Healthcare 2016.

3.1 Introduction

Clinical decision support systems aim to relay clinically relevant information while the patient is being treated. The relevant information can vary and can include recommendations of standardized pathways of care (Panella et al., 2003), evidence-based guidelines, and warnings about allergies and other contraindicated medications. The most effective systems are those that can *understand* the patient’s electronic medical record as it is being populated. By harnessing information that is entered as part of the clinician’s regular workflow, these systems do not add additional work or cognitive burden and can seamlessly integrate into clinical care.

While the field of machine learning has shown tremendous success learning to recognize patterns from large collections of labeled examples, one issue that arises repeatedly when applying machine learning to medical applications is the difficulty and cost of obtaining accurate labels for training. In this work, we focus on the so-called “anchored” setting, where gold-standard labels are difficult to obtain, but noisy versions of these labels can be easily extracted from clinical text using simple rules (Halpern et al., 2014; Agarwal et al., 2016). These rules (called anchors) are then used as surrogate labels in standard machine learning pipelines, with appropriate adjustments to account for noise (Natarajan et al., 2013; Elkan and Noto, 2008). Anchors need to be specified manually by experts, but are much easier than labeling large numbers of patients with manual chart abstraction.

Previous work with anchors (Halpern et al., 2016) (described in Chapter 2) showed that they can be used to build a large number of individual classifiers to identify a range of clinical conditions, but did not address the joint modeling of these conditions. Without a joint generative model, we implicitly assume when applying the phenotype classifiers that phenotypes are conditionally independent given the patient feature descriptor, and that the feature descriptor was always complete. Both of these assumptions are unrealistic. First, patient records are filled in over time (as discussed in the previous chapter) and thus certain fields may be missing at the time of inference. Second, confirming one phenotype may make a second much less likely, as in the case of two competing hypotheses to explain an abnormal finding.

Using anchors instead of the true labels means leaving the true labels as unobserved or *latent* variables. This introduces modeling complexities (for a more detailed introduction, see Section 1.2.4), *how do we model variables that are never observed in the data?* Approaches that maximize likelihood of the data (marginalizing over the latent variables) are computationally expensive to perform exactly, and may suffer from “ungrounded” latent variables, where the latent variable is introduced to model a particular phenotype, but models something

completely different.

In this work, we present a method of training joint probabilistic models with anchors using an approximation (variational lower bound) to maximum likelihood optimization. A semi-supervised term is added to the objective to ground the latent variables and ensure they model the phenotypes for which they are introduced.

Experimentally, we show that the semi-supervised term is necessary and effective to avoid the ungrounded latent variable problem. We evaluate joint models against individual classifiers in a clinical condition tagging task, specifically in answering the question “what else might the patient have?”. This type of query is useful to combat search satisfaction errors, as described later on. It also serves as an example of a general inference query that could be posed to an interactive system that allows for phenotypes to be confirmed or rejected during use.

3.1.1 Contributions

This chapter contains the following contributions:

- We generalize the definition of anchors from Chapter 2 to be a purely structural condition.
- We introduce a factor-analysis approach to model phenotypes and observed features, along with a learning algorithm that optimizes a variational lower-bound to the likelihood.
- We describe the ungrounded latent variable problem which occurs when learning with anchors instead of true labels and describe a semi-supervised objective which is effective in mitigating the problem.
- We experimentally validate the learned algorithms against relevant baselines including

the independent classifiers described in Chapter 2 on a clinically relevant inference task and show that a joint model performs best at ranking most-likely applicable phenotypes, performing nearly as well as oracle models learned with true labels. Whereas Chapter 2 focused on evaluating each phenotype predictor in isolation, here we explicitly evaluate the phenotypes against each other.

3.2 Anchored factor analysis

We model conditions and observations as a bipartite Bayesian network. In the following sections, we will describe the structure of the model and methods for learning its parameters. Throughout, we will follow the convention that random variables are denoted by uppercase letters (e.g., Y_i) and their values indicated by lowercase variables ($y_i \in \{0, 1\}$).

3.2.1 Anchor assumption

We assume the anchors are corrupted versions of the true labels and that the corruption process obeys a conditional independence constraint: The state of the anchor depends only on the true label. Specifically, conditioned on the true label, it is independent of all other observations. The anchors must contain some information about the true label. Let A_i be the anchor for Y_i , we assume that $P(A_i|Y_i = 0) \neq P(A_i|Y_i = 1)$.

Learning independent classifiers using anchors (Halpern et al., 2014, 2016) (Described in Chapter 2) additionally required an assumption of high positive predictive value (the corruption process does not produce false positive cases), which we do not require here. Instead we will require that that the class-conditional noise rates of the corruption process are known.

This is a strict generalization of the earlier setting, since Elkan and Noto (2008) show how to estimate the corruption rates under the assumption of high positive predictive value.

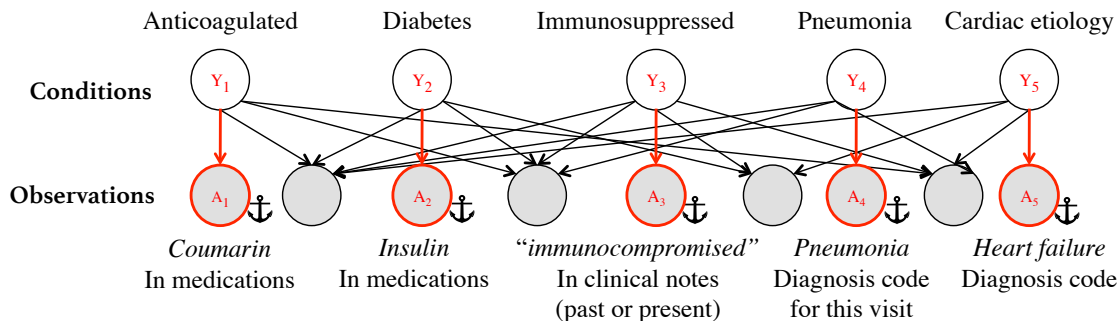


Figure 3.1: The joint probabilistic model used for clinical tagging is a bipartite graph involving conditions and observations, like QMR-DT (Shwe et al., 1991) with one or more *anchor* (red outline) for each condition (only one anchor per condition is shown in the illustration). Other observations (black outline) can have multiple parents. The anchors from Halpern et al. (2016) are available on github: <https://github.com/clinicalml/clinical-anchors>

3.2.2 Model structure and parametrization

We use a graphical model patterned after the historical QMR-DT network, originally designed for medical diagnosis (Shwe et al., 1991) (see Figure 3.1; more on the QMR-DT network is found in Section 1.2.3). The Bayesian network consists entirely of binary random variables, which are partitioned into diseases (or phenotypes) (Y_1, \dots, Y_n) and observations (X_1, \dots, X_m) . The model is bipartite with directed edges from diseases to observations.

The diseases are assumed to be marginally independent with individual prior probabilities, denoted as π_i :

$$P(Y = \{y_1, \dots, y_n\}) = \prod_{i=1}^n \pi_i^{y_i} (1 - \pi_i)^{1-y_i}. \quad (3.1)$$

The conditional probabilities of the observations (given the state of the diseases), are parametrized with a “noisy-or” distribution (Equation 3.2) (Shwe et al., 1991; Pearl, 1988):

$$P(X_j = 0 | Y = \{y_1, \dots, y_n\}) = (1 - l_j) \prod_{i=1}^n f_{i,j}^{y_i}, \quad (3.2)$$

where the parameters $f_{i,j}$ are referred to as *failure* probabilities and l_j is the *leak* probability.

The network can be viewed as a generative model. For each new patient, each disease Y_i

independently turns on with probability π_i . Each disease which is on tries to turn on each of its children, X_j , but fails with probability $f_{i,j}$. An additional “noise” parent is always on and fails to turn on its children with probability $(1 - l_j)$.

Rather than naively treat the anchors (which are unreliable labels) as telling us which diseases are present, we treat the conditions as latent variables and treat the anchors as observations. The anchor assumption places a structural constraint on the graphical model. Specifically, each anchor A_i must have only one parent which is Y_i .

Treating the conditions as latent variables makes learning the parameters of the model computationally difficult. When the variables are all observed, maximum likelihood estimation of the parameters is a concave optimization problem that can be solved efficiently. However, when the conditions are unobserved, the problem is no longer concave and optimization procedures can get stuck in local minima, even when the anchor assumption holds true.

A second problem is the *ungrounded latent variable* problem in which a latent variable, introduced specifically to model one phenotype, takes on a completely new meaning in order to increase the likelihood objective.

In the following sections, we describe the ungrounded latent variable problem in more detail, and then describe a semi-supervised likelihood-based objective and an effective initialization that allow us to learn models with good correspondence between the latent variables and the desired phenotype variables. Inference in these models can then be used for joint phenotype estimation.

3.3 Ungrounded latent variables

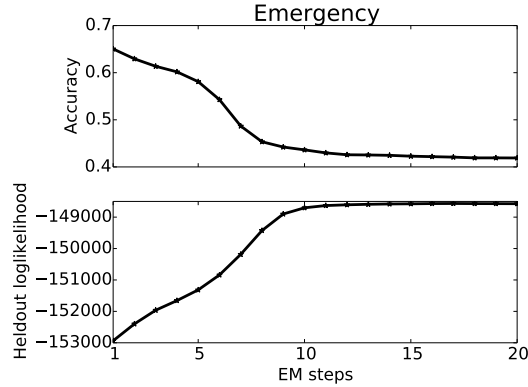
In unsupervised learning of latent variable models, we have little control over the interpretation of a particular latent variable, and interpretations are generally assigned in a post-hoc manner to each latent variable. In this chapter, anchors are used to ground each

latent variable to a certain meaning, particularly by constraining the space of models so that each anchor only has a single parent, its associated latent variable.

In this section, we show experimentally that maximizing the likelihood of the observations directly leads to poor performance on an inference task (the last-tag prediction described later on in Section 3.5.1). This maximization is a non-concave problem due to the latent variables, and marginalizing the latent variables to compute the likelihood is computationally difficult. However, we can use a generalized EM procedure (Dempster et al., 1977) (more detail in Appendix C.9).

We initialize the parameters using the Anchored Discrete Factor Analysis Tree (ADFA-Tree) method (described in Chapter 6), and run EM to determine whether additional steps of likelihood optimization can improve beyond the solutions found with ADFA-Tree. The particular initialization is unimportant here, only that it is a reasonable initialization. The somewhat surprising result is that as the likelihood of the data *increases*, the ability of the model to predict the last applicable phenotype for a patient *decreases* (See figure 3.2. Data likelihood for the figure is estimated with importance sampling. The proposal distribution is a product distribution using marginals inferred with Gibbs sampling.)

Examining the learned model reveals one cause of this mismatch between the likelihood objective and the phenotype prediction task. One of the latent variables, which was introduced to model the “headache” phenotype, takes on a completely different meaning as optimization progresses. We do not believe this is a feature of the particular learning algorithm (similar results have been obtained using gradient based optimization of a variational bound instead of EM), but rather a feature of the likelihood objective itself. The likelihood objective is improved by using the latent variable associated with the headache anchor to model negation scopes, an output of our feature processing pipeline, which is not associated with any particular phenotype. Even though the model “pays” the price of not being able to model the headache anchor well, the loss from that is offset by the gains of modeling the



Before optimization	anchor:headache, complains of, head, nausea, today, denies, acetaminophen, neck pain, neg:changes, neg:vision
After 9 EM steps	anchor:headache, neg:imaging, neg:ed, neg:course, neg:labs, neg:consults, neg:initial vitals, neg:trigger, neg:chest pain, neg:interventions

Figure 3.2: (Top) Effect of pure likelihood-based optimization on accuracy on a tag. A likelihood optimization procedure is initialized with the parameters of the anchor-based method-of-moments learning algorithm (ADFA-tree; described in Chapter 6). As the likelihood optimization progresses, accuracy at predicting phenotypes degrades. (Bottom) The latent variable associated with the “headache” anchor changes meaning over the course of optimization.

negation scopes.

We call this problem the *ungrounded latent variable problem*. Simply adding anchors is not sufficient to ensure that latent variables take on the meaning intended by the modeler. This problem could possibly be avoided by adding new latent variables with no anchors and hoping that they model all of the extraneous factors, leaving the anchored latent variables to model their intended phenotypes, but there is no guarantee that the optimization would proceed in that direction.

3.4 Semi-supervised training

3.4.1 Semi-supervised objective

Exact inference in the QMR-DT model is known to be NP-hard, even if there is an anchor for every condition. The typical approach maximizing likelihood in the presence of latent variables, Expectation Maximization (EM), is computationally challenging because at each step one has to use an approximate inference algorithm such as Markov chain Monte Carlo to approximate the necessary expectations.

Instead, we follow Mnih and Gregor (2014) in formulating a variational lower bound on the likelihood function using a recognition model. We start with the standard Evidence Lower Bound (ELBO), which holds true for *any* distribution, $q(Y|X)$:

$$\mathcal{L}(\theta, q) \equiv E_{y \sim q} [\log P(X, y; \theta) - \log q(y|X)] \leq \log P(X). \quad (3.3)$$

In mean field variational inference, q is chosen to be a fully factorized distribution. In this work, we restrict q to be the output of a parametrized model, with parameters ϕ . The parametrized distribution $q(y|X; \phi)$ is referred to as a *recognition* model and its function is to perform approximate inference in the network. The bound is tightest as $q(Y|X)$ approaches $P(Y|X; \theta)$, that is, as it approximates inference in the generative model. As learning proceeds, the recognition model learns to *compile* inference.

In our work we use a simple recognition model that performs logistic regression to approximate the posterior of each condition independently:

$$q(y|x; \phi) = \prod_i \left(y_i \sigma(\phi_i \cdot x) + (1 - y_i)(1 - \sigma(\phi_i \cdot x)) \right), \quad (3.4)$$

where σ is the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$ and x is padded with a 1 to allow for a bias term.

In contrast with mean-field inference, where q is optimized for every data point separately, here we train a single model, allowing us to amortize the cost of inference over many data points. Methods to take gradients with respect to θ and ϕ and optimize with stochastic gradient ascent are described in detail in Mnih and Gregor (2014). Our hyperparameter settings are described in Appendix C.2. Using this algorithm enabled us to learn orders of magnitude faster than EM, with comparable results in terms of likelihood objective obtained.

However, we found that without adding an additional term to the objective, the anchor constraints (i.e., each anchor only has a single, specified parent) were not sufficient to make sure that the latent variables took on their intended meanings. Specifically, we observed that as the held-out likelihood of the observations improved, the predictive quality of the models got worse (measured using the heldout tag prediction task of Section 3.5.1). Upon inspecting the model, we found that drift occurred: the latent variables took on new meanings and lost their original grounding.

Inspired by recent work on semi-supervised learning with deep generative models (Kingma et al., 2014), our solution is to add a supervised term to the objective that encourages the recognition model to additionally predict the presence or absence of the anchors, ensuring that the meaning of the latent variable is tightly tied to the anchor. The prediction cannot use the anchors themselves, so we form a new censored vector, \tilde{x} , which is a copy of x but has the values of the anchors set to a constant 0. We also introduce an additional bias term ϕ'_0 to allow the prediction of the anchors and the labels to differ from each other. The supervised term has the form:

$$R(\phi, \phi'_0) = -\ell(\sigma(\phi \cdot \tilde{x} + \phi'_0), a), \quad (3.5)$$

where $\ell(\cdot, \cdot)$ is log loss and a is the vector of anchors. The final objective is thus:

$$\text{maximize } \mathcal{L}(\theta, \phi) + \lambda R(\phi, \phi'_0), \quad (3.6)$$

where $\lambda > 0$ is a hyperparameter specifying the trade-off between these two terms in the objective, and although we wrote Eq. 3.6 for a single data point, the actual objective we optimize is the sum of this over all the data points. A more detailed version of the objective is found in Appendix C.3.

At test time, we discard the recognition model, q_ϕ , which was used to train the parameters of the generative bipartite Bayesian network, but cannot support queries with conditioning on some of the clinical variables, and use the joint probabilistic model, $P(X, Y; \theta)$, for inference. Inference in the joint model does not have an efficient closed form solution, but can be approximated with Gibbs sampling.

The supervised term is similar to the “Anchor and Learn” objective described in the previous chapter (Section 2.2), as it is trained to predict the anchor given all other features. The additional bias term, ϕ'_0 , serves as a calibration term which is learned simultaneously. An important difference is that here these classifiers are being used in the *recognition network*, to perform approximate inference, but algorithm’s output is the *generative model* described in Section 3.2.

3.4.2 Model initialization

In order to initialize the model, we use the anchors to get a rough estimate of the failure probabilities for each of the observations. If we observed the latent clinical conditions (the Y variables), we could use a simple moments-based estimator using empirical counts to estimate the failure probabilities (Equation 3.7).

$$\hat{f}_{i,j} = \frac{\hat{P}(X_j = 0 | Y_i = 1)}{\hat{P}(X_j = 0 | Y_i = 0)}. \quad (3.7)$$

The estimator $\hat{f}_{i,j}$ is then clipped to lie between $[0,1]$. The consistency of the method is not affected by this clipping, since if sufficient data were drawn from the model, the

estimator would naturally lie in that range and clipping would not be necessary. Once all the failure probabilities are estimated, the leak probabilities can be estimated to account for the difference between the true observed counts and those predicted by the model (Appendix C.4).

Since the clinical conditions are generally unobserved, we estimate these conditional probabilities using empirical counts assuming that the anchors (which are noisy versions of the labels) are Y , and then perform a *denoising* step to estimate the conditional probabilities as though the true labels were observed. Specifically, in Section 3.2.1 we assumed that the label corruption process was independent of all other observed variables. This leads to the following equation:

$$P(X_j|A_i) = P(Y_i = 1|A_i)P(X_j|Y_i = 1) + P(Y_i = 0|A_i)P(X_j|Y_i = 0) \quad (3.8)$$

The left-hand side of this equation is a quantity that only involves observed variables (X_j, A_i) and can be estimated from empirical counts. The right-hand side uses the noise rates of the corruption process $P(Y_i|A_i)$ and the conditional probabilities that we care about, $P(X_j|Y_i)$. If we assume that the noise rates of the corruption process are known, then we can form four independent linear equations with four unknowns and solve the following matrix equation:

$$\vec{P}(X_j|A_i) = R\vec{P}(X_j|Y_i), \quad (3.9)$$

where $\vec{P}(X_j|A_i)$ is a column vector with four entries, one for each setting of (X_j, A_i) in $\{0, 1\}^2$. R is a 4×4 matrix encoding the noise rates of the corruption process. Explicit constructions of these terms are given in Appendix C.5.

We could simply invert the noise matrix R to solve $\vec{P}(X_j|Y_i) = R^{-1}\vec{P}(X_j|A_i)$, however, it would not be guaranteed that the solution would give a valid probability (i.e., non-negative and sum-to-one conditions) for $\vec{P}(X_j|Y_i)$. Instead, we explicitly solve the optimization

problem with simplex constraints to minimize a KL-divergence measure between a proposed distribution $\vec{P}(X_j|Y_i)$ and the denoised version of the empirical counts $P(X_j|A_i)$:

$$\vec{P}(X_j|Y_i) = \operatorname{argmin}_{\vec{p} \in \Delta} D_{\text{KL}} \left(\vec{P}(X_j|A_i) \parallel R\vec{p} \right) \quad (3.10)$$

The optimization is convex and we solve it with exponentiated gradient descent (Kivinen and Warmuth, 1995). The *cleaned* distribution, $\vec{P}(X_j|Y_i)$, obtained from solving Equation 3.10 is then substituted into the failure probability estimator in Equation 3.7 to obtain estimates of the failure and leak probabilities (regarding the leak probabilities, see Appendix C.4). This whole procedure can be shown to be a *consistent* estimator, meaning that if the model assumptions hold (i.e., of conditional independence), this will converge to the true probabilities as the amount of data goes to infinity.

A more advanced version of this initialization is described in Chapter 6. In this chapter, we also do not use the additional constraints to provide robust recovery described in Section 6.3.2, though they could be applied directly in this setting. The model used in this chapter assumes a product-distribution for the prior probability of the phenotypes (i.e., independent Bernoulli variables). As such, there is no need for the subtracting off or conditioning procedures described in Section 6.4.2.

The model for $P(Y)$ could be made more expressive. The optimization procedure described in Section 3.4.1 makes no assumption of the independence of latent variables, so there is no technical barrier to applying it to a model with a structured model for $P(Y)$ (e.g., the tree structured Bayesian networks described in Chapter 6, an auto-regressive prior (Larochelle and Murray, 2011), etc.), though finding a good local optimum may be more difficult with more parameters.

Algorithm 1 Parameter estimation algorithm

- 1: (Precondition) Identify anchors
 - 2: Obtain cleaned moment estimates using anchors (Eq. 3.10)
 - 3: Initialize θ_0 using method of moments (Eq. 3.7).
 - 4: Initialize ϕ_0 randomly
 - 5: NVIL optimization (Mnih and Gregor, 2014) of Eq. 3.6.
 - 6: Discard ϕ and use joint model parametrized by θ .
-

3.4.3 Complete algorithm

The full parameter estimation algorithm is summarized in Algorithm 1.

3.4.4 Model selection

We do not assume that we have any ground truth labels to do model selection, so we use a stopping criteria based on heldout *anchor* prediction. We hold out 1000 patients from the train set as a validate set to determine the stopping criteria. For each patient in the validate set, we censor one positive anchor that appears in the record and all of the negative anchors, and perform inference with the joint model $P(X, Y; \theta)$ to determine which anchor is missing. Inference for the final anchor is performed with Gibbs sampling to obtain marginals for the tags, and then the likelihood of each anchor is computed as a function of the marginal likelihood of its parent tag (details in Appendix C.6). This model selection criterion mimics the heldout-tag prediction task described in Section 3.5.1 used in the evaluation, but uses anchors instead of the true labels. Other stopping criteria using anchors could be developed depending on the intended use.

3.5 Evaluation framework

In this section, we describe the evaluation task and experimental framework used to assess the joint modeling of phenotypes.

3.5.1 Heldout phenotype prediction task

We test the ability of our model to perform inference with conditioning by presenting it with a heldout phenotype prediction task. Conditioning occurs naturally in clinical practice when the physician is given the ability to confirm or reject phenotypes suggested by the EMR system. A phenotype “tagging” system is currently implemented at Beth Israel Deaconess Medical Center’s emergency department’s electronic medical record system. Physicians, during their ordinary clinical workflow, are presented with possible phenotypes for each patient as part of their dashboard view (see figure 3.3). Accepting or rejecting a phenotype updates recommendations through inference. Accepted phenotypes initiate changes to the physician’s display, for example, displaying standardized order sets, enrollment in clinical pathways, or other preprogrammed responses.

In the held out phenotype prediction task, the model is presented with a patient record and all but one of the phenotypes that apply for that record. The task is to predict the final phenotype that applies for this patient. We record the accuracy of each model (i.e., the proportion of times it chooses the correct phenotype to fill in), top-5 performance (i.e., proportion of times the correct phenotype appears in the top 5 predictions) and the mean-reciprocal rank of the correct prediction (MRR). Since this task is choosing the best last phenotype, we do not need to perform approximate inference. Instead we evaluate the likelihood of the data with each possible final phenotype and perform the normalization, which corresponds to exact inference (Appendix C.7).

The held out phenotype prediction task is clinically relevant in that can be used to combat a recognized cognitive bias known as “search satisfaction error” (Groopman and Prichard, 2007). This cognitive bias, a failing of the Occam’s razor heuristic, is the tendency to overlook additional conditions once a single unifying diagnosis is found. This is particularly dangerous when a more serious diagnosis is overlooked because a unifying diagnosis was discovered first.

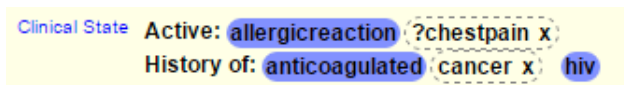


Figure 3.3: Tagging system within an electronic medical record.

For example, a patient with signs of a severe infection may be diagnosed with urinary tract infection and treated with antibiotics, while missing a second diagnosis of pneumonia. The delay in treatment of the pneumonia could be potentially dangerous to the patient. Another example is a patient with a kidney stone, whose co-existing urinary tract infection was missed. Although kidney stones generally resolve on their own, patients who also have a concurrent urinary tract infection require immediate intervention. Clinical decision support systems can mitigate this cognitive bias by suggesting additional diagnoses that may explain a set of symptoms. We simulate this problem by randomly removing a phenotype, and trying to recover it using the model. This could correspond to a situation where the physician has confirmed one diagnosis and the model attempts to suggest a likely second diagnosis that would go along with the first. Since the model performs inference, this could potentially be very different from the next most likely phenotype if the model received no confirmation of the first phenotype.

3.5.2 Phenotypes

We chose to model 23 phenotypes relevant to the emergency department. These are a subset of the phenotypes modeled in Halpern et al. (2016), chosen to have anchors which include both ICD9 billing codes and another form of observation (either free text or medication). The anchors are taken directly¹ from Halpern et al. (2016). To simulate the anchor setting while still having labels for evaluation, we use the ICD9 codes to represent an unobserved “ground truth” for the purposes of this study and other data from the EMR (i.e., medications

¹Available on github: <https://github.com/clinicalml/clinical-anchors>

Phenotypes		
abdominal pain acute	alcohol acute	allergic reaction acute
asthma-copd acute	back pain acute	cellulitis acute
cva acute	epistaxis acute	fall acute
gi bleed acute	headache acute	hematuria acute
intracranial hemorrhage acute	kidney stone acute	vehicle collision acute
pneumonia acute	severe sepsis acute	sexual assault acute
suicidal ideation acute	syncope acute	uti acute
liver history	hiv history	

Table 3.1: The full list of phenotypes included in the model. Acute conditions relate to the patient’s current condition. History refers to the patient’s history.

and free text entries) are used for observations. While ICD9 codes are generally unreliable for establishing gold-standard phenotypes (e.g., Cipparone et al., 2015; Tieder et al., 2011; Birman-Deych et al., 2005; Aronsky et al., 2005), we consider them reliable enough to assess relative performance of different methods which are trained using anchors. Table 3.1 gives the full list of phenotypes that were modeled.

3.5.3 Cohort selection

The study was performed in a 55,000-visit/year level 1 trauma center and tertiary academic teaching hospital. All consecutive emergency department (ED) patients between 2008 and 2013 were included. Records were de-identified and personal health information was removed before beginning the analysis. Each record represents a single patient visit, leading to a total of 273,174 records of emergency department patient visits. The study was approved by the hospital’s institutional review board.

We focus on patients with at least two of the modeled phenotypes so that it is possible to identify one condition and ask “what else might the patient have?” After filtering for patients with at least two of the modeled conditions, we were left with 16,268 patients. Of these patients, 11,000 were designated for training and 5,000 for testing. The final 268 patients

Field	Representation
Age	Binned to nearest decade
Sex	M / F
Chief Complaint	Free text
Triage Assessment	Free text
MD comments	Free text
Medication history	GSN codes
Dispensed medications	GSN codes
Billing codes	ICD9 codes

Table 3.2: Features extracted . Billing codes were extracted to perform the evaluation, but were not used to create the patient feature vector.

were not used.

3.5.4 Data extraction and feature selection

For each visit, we extracted data from the fields listed in Table 3.2 to build a binary bag-of-words representation for every patient. Full details of the free-text processing pipeline including negation and bigram detection can be found in Appendix C.1. For each condition, we create an anchor token which appears if *any* of the condition’s anchors appears in the note. Terms that appear in more than 50% of the patient records are removed as stopwords, and the most common 1000 terms are kept. Any anchors that were filtered out in this step are added back in, yielding a final feature vector with 1003 binary indicators.

3.5.5 Baselines

We compare against 3 different baselines. Since our proposed method assumes access to the correct failure and leak parameters for the anchors, we provide that information to all of the baselines for fairness.

1. **Naive labels** – treats the anchors as true labels and learns a noisy-or model with maximum likelihood estimation. In the final model, we “edit” the failure and leak

parameters of the anchors to set them to the correct values.

2. **Noise tolerant classifiers** – We use the method presented in Natarajan et al. (2013) to learn noise-tolerant classifiers (explicitly providing the algorithm with the noise rates). In experiments we found that this method was more effective than the method of Elkan and Noto (2008), so we present this baseline to compare to the individual classifiers learned in Halpern et al. (2014). The method of Natarajan et al. (2013) does not explicitly describe how to predict when anchors are observed in the record. In this case, we simply predict the noise rate of the anchor, which we found to be more effective than ignoring the special status of the anchor.
3. **Oracle MLE** – An upper bound on performance which uses the true labels to learn a noisy-or model with maximum likelihood estimation. This is impractical in practice, but gives us a sense of how close to optimal we are performing using our noisy labels.

3.6 Results

Table 3.3 presents our method compared with the baselines presented in Section 3.5.5. ‘Noisy-or init’ refers to the moments-based estimator described in Section 3.4.2, whereas ‘Noisy-or final’ refers to the results after the semi-supervised learning algorithm described in Section 3.4.1. The results here use a λ parameter of 10. Other details of the hyperparameters are in Appendix C.2.

The noisy-or model significantly outperforms the noise-tolerant classifiers and the naive labeling baselines. Our performance comes close to the optimal maximum likelihood performance, suggesting that even though we don’t use the true labels in training, we are still able to recover a model which is similar to the one we would learn if we had access to the true labels. The method of moments initialization is helpful. Using random initialization, we do

Model	Accuracy	Top 5	MRR
Noise tolerant classifiers	0.54	0.86	0.67
Naive Labels	0.61	0.85	0.71
Noisy-or init	0.64	0.91	0.76
Noisy-or final	0.68	0.92	0.79
Noisy-or oracle MLE	0.71	0.93	0.81

Table 3.3: Results for last-tag prediction. Performance measures are Accuracy, Top-5 (correct tag within the top 5) and MRR (mean reciprocal rank). Noisy-or init uses the model with the θ_0 parameters. Noisy-or final shows the result after likelihood optimization.

Tag	Top weighted terms
abdominal pain	pain, Ondansetron, nausea, days
alcohol	male, sober, ed, admits, found, denies
asthma	albuterol sulfate, sob, Methylprednisolone, cough
fall	s/p fall, fall, fell, neg:loc, neg:head
hematuria	infection, male, urology, urine, flank pain
HIV+	male, Truvada, cd4, age:40-50, Ritonavir
collision	car, neg:loc, age:20-30, hit, neck, driver

Table 3.4: Highly weighted (low failure probability) words learned by the noisy-or model after likelihood optimization. Words marked neg: are within a negation scope. Some shortforms are present in the text (ed: emergency department, sob: shortness of breath, loc: loss of consciousness, s/p: status post).

not beat the naive labels baseline. Table 3.4 shows the highly weighted words learned by our model. All of the noisy-or models learn similar sets of highly weighted words, the main differences between the models are in the exact settings of the parameters which affect the inference procedure to choose the correct last tag.

3.7 Next steps and open questions

There are a number of limitations to this study. First, we used the ED ICD-9-CM discharge diagnoses which may have misclassified patients. Patients may have been suspected of having one diagnosis in the ED and ultimately may have had an alternative diagnosis. As such, we can only assess relative performance of the various models, but cannot draw

conclusions about absolute accuracy.

This study occurred at a single institution with a custom built ED information system. These results might not generalize to other systems that may not be modified to support complete electronic capture of clinical data and customized decision support. While we internally validated the results, external validation is warranted. It will be interesting to discover whether the same algorithm may be applied to another institution, or whether reliable machine learning requires first training on local clinical data.

Our held-out tag prediction task is a synthetic task intended to simulate real clinical scenarios where some, but not all conditions are known about a patient. The evaluation is performed using retrospective data. Further study would be needed to confirm these results in real clinical practice.

We depend on estimates of the parameters of the corruption process to perform the method-of-moments initialization in Section 3.4.2. This is a potential weakness of the algorithm, though we are careful to consider baselines which can make use of the same information. In this work we do not address how those parameters could be obtained and rely on oracle estimates of these parameters. However, we expect that even by labeling a small number of examples we could obtain good enough estimates of these parameters to serve for the initialization. This question appears again in Chapter 6 and is discussed in more detail in Section 8.1.1.

Diseases are not truly statistically independent of one another, despite our modeling them as such in this chapter. As mentioned at the end of Section 3.4.2, the semi-supervised objective applies equally to arbitrary prior distributions of the phenotypes. More elaborate method-of-moment estimation techniques, such as those described in Chapter 6 can be used together with anchors to learn the joint distribution of the conditions, even when they are never observed in the data.

The feature set used in the experiments is fairly small (1000 features) and was chosen

with a naive feature selection process. The result is that some rarer but highly informative words were likely excluded. The feature set could be scaled up. Including more features may require regularization in the generative network to ensure that rare features do not have undue influence on inference tasks. Another factor for consideration would be whether violations of the factor analysis assumptions (observations are conditionally independent given the phenotypes) would accumulate and degrade the inference results as the feature set grows larger.

The semi-supervised objective is a heuristic, meant to approximate $P(anchor|\tilde{x})$ using the variational distribution. Are there ways of optimizing this term more directly?

In our experiments we found that randomly initialized models did not outperform the independent classifiers baseline while those initialized with method-of-moments achieved the best performance (results not shown). Under what conditions can we learn effective models starting from random initialization? What is the role of method-of-moments or other initialization schemes in learning with recognition models?

In this work we use a very simple recognition model: independent logistic regression classifiers. Inference could be better approximated using a multilayer or deep neural network, though the variance in the gradients would need to be controlled. Would a better approximation of inference improve the learned models? More generally, can deep recognition models be a useful tool for learning *shallow* graphical models (e.g., factor analysis or topic models) using a recognition model for approximate inference?

It is important to note that the oracle results described in Section 3.5.5 are oracle results for models *within the model family of the anchored factor analysis models*. Independent logistic regression classifiers (one for each phenotype), actually perform better at the held-out tag prediction task when provided with oracle access to the true values for the tags while training. Oracle logistic regression classifiers have accuracy of 0.78 for the held out (compare with 0.71 for oracle result in Table 3.3). Other, more expressive, fully supervised classifiers or

generative models could potentially do even better. Thus, the comparison with oracle results is to show what we can learn with anchors instead of fully labeled data within a single model family, not to give an absolute upper limit on the possible performance on the task, which could be higher.

Chapter 4

Learning noisy-or networks with singly-coupled observations

Acknowledgments: The work on learning with unknown structure was done jointly with Yacine Jernite and David Sontag. Thanks to Rong Ge, Ankur Moitra and Sanjeev Arora for helpful discussions. Thanks to Tomas Singliar for providing his code for variational learning of noisy-or. The QMR Knowledge Base is provided by University of Pittsburgh through the efforts of Frances Connell, Randolph A. Miller, and Gregory F. Cooper. Parts of this chapter were previously published in: “Unsupervised learning of noisy-or Bayesian networks. Halpern & Sontag. UAI 2013; “Discovering hidden variables in noisy-or networks using quartet tests” Jernite, Halpern & Sontag. NIPS 2013.

4.1 Introduction

Chapter 3 used *anchors* to learn a joint generative model patterned after the historical Quick Medical Reference - Decision Theoretic (QMR-DT) model (Miller et al., 1986; Shwe et al., 1991). In this chapter we address the theoretical question of the *polynomial learnability*

of this family of Bayesian networks (bipartite noisy-or factor analysis models), using a different structural condition known as *singly-coupled tuples*.

We approach this analysis disconnected from any particular EMR phenotyping task, though we note that this family of probabilistic models is interesting because it has been used historically for medical diagnosis, which is closely related to phenotyping, and was used in Chapter 3 in a practical EMR phenotyping system.

This analysis is complementary to the anchor-based approach. Some networks do not have anchors for every latent variable, but *do* have singly-coupled tuples and vice versa. Unlike anchors, which need to be identified by experts, and have their noise rates determined through an auxiliary process, singly-coupled tuples do not *require* manual specification on the part of experts, though expert knowledge can be used to improve both the computational and statistical efficiency of the learning procedure.

The original QMR-DT network was designed to model the diagnostic reasoning relevant in the practice of internal medicine, included the relationships between 570 diseases and 4075 symptoms. Its parameters and structure were specified by hand in a labor intensive process that was estimated to take over 20 years of researcher time, eliciting probabilities from experts and trusted textual sources such as journal articles (Shwe et al., 1991).

A natural question is whether it is *possible* to learn this type of network directly from clinical data. The desirability of learning models for diagnosis directly from data was already demonstrated in early work on computerized diagnosis models (Leaper et al., 1972), but obtaining large-scale collections of medical records in a format suitable for computerized analysis was not feasible until the recent widespread adoption of electronic medical records.

A second barrier to learning directly from data is the completeness of the medical record, and particularly in the diagnoses. When diagnoses are completely observed, learning the parameters of the network through maximum likelihood estimation is a concave optimization problem which can be solved efficiently using standard gradient-based methods (Appendix D.2).

However, many hospital visits are resolved without a conclusive list of diagnoses. The diagnosis codes or problem list items recorded in the electronic medical record should not be assumed to be complete, or fully reliable (Wright et al., 2012; Gandhi et al., 2011; O’malley et al., 2005; Birman-Deych et al., 2005; Tieder et al., 2011). When the diagnoses are not observed (latent variable setting), maximum likelihood estimation of the parameters is a non-concave optimization and is conjectured to be worst-case NP-hard. This transition from computational tractability of the maximum likelihood problem to worst-case intractability when latent variables are introduced is a common phenomenon which is discussed in the Introduction (Section 1.2.4).

This chapter addresses the following theoretical question regarding the QMR-DT network and similarly structured diagnosis networks: Can the parameters and structure of a diagnosis network be consistently estimated (to arbitrary accuracy) in polynomial time from observations of symptoms, but not diseases?

The main tool used in this chapter to prove polynomial learnability is the method-of-moments, pioneered by Pearson (Pearson, 1894). This method is discussed in more detail in Section 1.2.5. This method works with aggregate statistics of the dataset, and avoids performing (approximate) inference on individual data instances as is common in likelihood-based methods (e.g., variational EM (Šingliar and Hauskrecht, 2006) or the stochastic variational gradient ascent method described in Chapter 3).

The analyses presented in this chapter rely on the structural features of individual networks (i.e., the presence of singly-coupled tuples). For this purpose, the historical QMR-DT network will serve as an example of a real-life network that we would like to be able to learn. Unlike previous chapters which analyzed algorithms on real datasets, the main contributions of this chapter are theoretical and the experiments are performed on simulated datasets based on the real QMR-DT network.

4.1.1 Contributions

The chapter is divided into two sub-sections. The first addresses the question of parameter learning when structure is known (e.g., learning the parameters of the QMR-DT network after the structure had been specified). The second additionally addresses the question of structure learning. For each setting, we make the following contributions:

- We give a set of *sufficient* structural conditions for polynomial learnability based on singly-coupled tuples.
- We develop a provably correct polynomial-time method-of-moments algorithm based on these sufficient conditions.
- We show empirically that the sufficient conditions for learnability almost hold in the case of the QMR-DT network structure, meaning that the vast majority of its parameters *can* be estimated consistently, leaving a small number of parameters unknown.
- We compare against a variational EM approach, and show that the method-of-moments algorithm is both faster for large datasets, and provides consistent parameter estimates in settings where variational EM does not.
- We introduce a stochastic variational inference algorithm that is also shown to be very fast and effective for learning similar models.

4.2 Singly-coupled tuples in noisy-or diagnosis networks

The analyses in this chapter apply to noisy-or diagnosis networks with singly-coupled tuples. We first define these terms and some notation, and identify a key property of singly-coupled triplets. We will use the terminology of diseases and findings from the QMR-DT

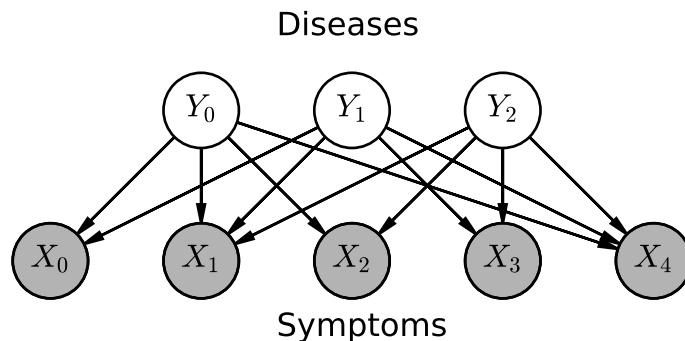


Figure 4.1: The QMR-DT network is a Bayesian network for medical diagnosis. Latent diseases are connected to observed symptoms through noisy-or edges.

network, but these models could be more generally applied to EMR phenotyping or more general factor analysis.

4.2.1 Noisy-or diagnosis networks

The probabilistic graphical model for noisy-or diagnosis network is defined as follows:

\mathcal{G} is a bipartite graph between diseases Y_1, \dots, Y_m and observed findings X_1, \dots, X_n . Conditional probabilities in the graph are parametrized as “noisy-or” gates (Pearl, 1988; Henrion, 1989):

$$P(X_j = 0|Y) = (1 - l_j) \prod_{i \in Pa(X_j)} f_{i,j}^{y_i}, \quad (4.1)$$

where the parameters $f_{i,j}$ are referred to as *failure* probabilities and l_j are *leak* probabilities. More information about the noisy-or conditional probability distribution can be found in the Introduction (Section 1.2.3). Diseases are marginally independent Bernoulli random variables:

$$P(Y = \{y_1, \dots, y_m\}) = \prod_i \pi_i^{y_i} (1 - \pi_i)^{1-y_i}. \quad (4.2)$$

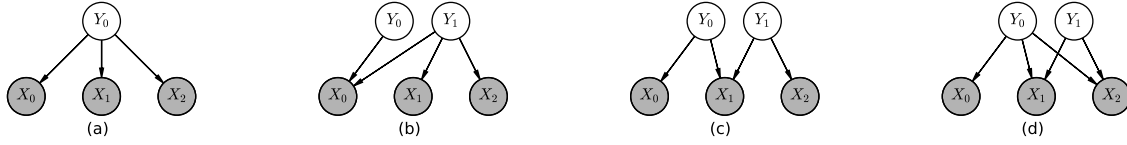


Figure 4.2: Examples of singly and non-singly coupled triplets. (a,b) are singly coupled. (c,d) are not singly coupled as they violate conditions 1 and 2 respectively.

4.2.2 Singly-coupled tuples

The main structural property that plays a key role in our learning algorithms is *singly-coupled tuples*.

Definition 1. Singly-coupled tuple: A k -tuple of findings (X_1, \dots, X_k) is singly-coupled in a bipartite noisy-or network \mathcal{G} if the following conditions hold:

1. There is one parent that is common to all k findings.
2. There is no second parent that is shared by any pair of findings.

Examples of singly and non-singly coupled triplets are shown in Figure 4.2. Unlike anchors which are linked by an edge to a single-latent variable, singly coupled tuples are *d-separated* (Pearl, 1988) by a single latent variable. Both structural properties have the effect that they link an unknown latent variable (e.g., a disease) to observed variables in a way that allows us to uncover properties of the latent variable from the observed variables.

4.2.3 Parameter recovery with singly-coupled triplets

Singly-coupled triplets are interesting because of their connection to three-view mixture models and canonical tensor decompositions. We begin by introducing the canonical tensor decomposition and how it can be used to solve for parameters in the three-view mixture

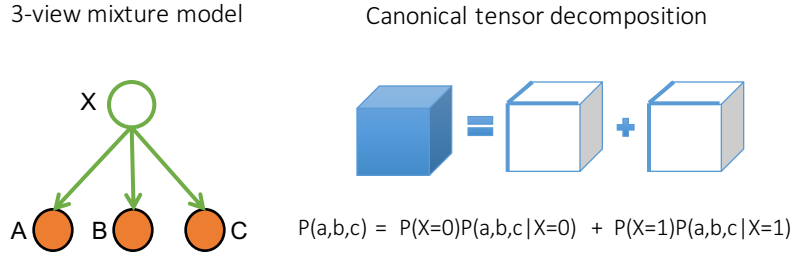


Figure 4.3: The distribution of observations from a three view mixture model with a binary hidden state Y can be decomposed into two rank-1 tensors.

model. Finally, we will show a simple reduction from parameter learning in mixture models to singly-coupled triplets.

4.2.3.1 Canonical tensor decomposition and its relationship with mixture models

The canonical tensor decomposition, similar to the SVD for matrices, attempts to write a tensor as a linear combination of rank-1 tensors. For example, for a 3-dimensional tensor T , the canonical decomposition is:

$$T = \sum_{i=1}^K \lambda_i u_i \otimes v_i \otimes w_i, \quad (4.3)$$

where u, v, w are vectors. The minimal K for which this decomposition is possible is known as the tensor-rank T .

Mixture models have a similar form (Figure 4.3). For example, a three-view mixture model with discrete variables A, B, C and with a hidden class variable Y , taking K states, is written as:

$$P(A, B, C) = \sum_{i=1}^K P(Y = i) P(A|Y = i) \otimes P(B|Y = i) \otimes P(C|Y = i). \quad (4.4)$$

We see then that solving the canonical decomposition with simplex constraints (i.e., finding $\lambda, u, v, w \in \Delta$ for a given T) corresponds to finding the components of a mixture model. In general, finding the minimal rank canonical decomposition is computationally intractable, and is not necessarily unique, but in settings where a $2 \times 2 \times 2$ tensor is known to be a joint distribution corresponding to a 2-state mixture model (i.e., A, B, C, Y are all binary random variables), a unique decomposition exists and can be calculated efficiently (Lazarsfeld, 1950; Berge, 1991). The algorithm is presented in Algorithm 2.

4.2.3.2 Mixture models components and noisy-or parameters

From recovered mixture model components, it is easy to estimate the noisy-or parameters of the model. Equations 4.5-4.8 show how to do that.

$$\pi_i = \gamma \tag{4.5}$$

$$f_{Y,A} = \frac{P(A = 0|Y = 1)}{P(A = 0|Y = 0)} \tag{4.6}$$

$$f_{Y,B} = \frac{P(B = 0|Y = 1)}{P(B = 0|Y = 0)} \tag{4.7}$$

$$f_{Y,C} = \frac{P(C = 0|Y = 1)}{P(C = 0|Y = 0)}. \tag{4.8}$$

4.2.3.3 Singly-coupled triplets and mixture models

The final step in learning the parameters of singly-coupled triplets is to notice that singly-coupled triplets are *marginally* mixture models, where the hidden state variable is the coupling latent variable. The parameters associated with the other latent variables can simply be treated as part of the *leak* variable and thus do not change the estimate of the failure probabilities. Said differently, for any singly-coupled triplet, there exists an equivalent

Algorithm 2 Binary Tensor Decomposition

Input: Tensor T of size $2 \times 2 \times 2$.

Output: Two rank-1 tensors such that $\gamma T_1 + (1 - \gamma)T_2 = T$.

- 1: Matrix $X_1 = T_{(0, \cdot, \cdot)}$
- 2: Matrix $X_2 = T_{(1, \cdot, \cdot)}$
- 3: $Y_2 = X_2 X_1^{-1}$
- 4: $\lambda_1, \lambda_2 = \text{roots}(\lambda^2 - \text{Tr}(Y_2)\lambda + \text{Det}(Y_2))$
- 5: $\vec{u}_1 \vec{v}_1^T = (\lambda_1 - \lambda_2)^{-1}(X_2 - \lambda_2 X_1)$
- 6: $\vec{u}_2 \vec{v}_2^T = -(\lambda_1 - \lambda_2)^{-1}(X_2 - \lambda_1 X_1)$
- 7: Decompose* $\vec{u}_1 \vec{v}_1^T, \vec{u}_2 \vec{v}_2^T$ into $\vec{u}_1, \vec{u}_2, \vec{v}_1, \vec{v}_2$.
- 8: $\vec{l}_1 = (1 \ \lambda_1)^T, \vec{l}_2 = (1 \ \lambda_2)^T$
- 9: $T_1 = \vec{u}_1 \otimes \vec{v}_1 \otimes \vec{l}_1, T_2 = \vec{u}_2 \otimes \vec{v}_2 \otimes \vec{l}_2$
- 10: **if** $T_1(0, 0, 0) > T_2(0, 0, 0)$ **then**
- 11: Swap T_1, T_2
- 12: **end if**
- 13: $\gamma = \sum_{i,j,k} T_1$
- 14: normalize $T_1 = \frac{T_1}{|T_1|}, T_2 = \frac{T_2}{|T_2|}$

*To decompose the 2×2 matrix $\vec{u}\vec{v}^T$ into vectors \vec{u} and \vec{v} , set \vec{v}^T to the top row and $\vec{u}^T = \begin{pmatrix} 1 & \frac{(\vec{u}\vec{v}^T)_{(2,2)}}{(\vec{u}\vec{v}^T)_{(1,2)}} \end{pmatrix}$.

-Notation $T = \vec{u} \otimes \vec{v} \otimes \vec{l}$ means that $T_{(i,j,k)} = u_i v_j l_k$. $|T| = \sum_{i,j,k} |T_{i,j,k}|$.

mixture model, where all of the parents other than the coupling parent have been marginalized out. The failure probabilities associated with that mixture model are the same as the failure probabilities associated with the original singly-coupled triplet.

4.2.3.4 Triplet moments are necessary

It is natural to ask whether a similar unique decomposition could be achieved by looking at pairs of observations rather than triplets. It turns out that pairwise moments are insufficient for *identifiability*, that is, there are multiple parameter settings that give rise to indistinguishable pairwise moments. This insufficiency of pairwise moments has been noted already in Chang (1996) and appears in detail with an example in Appendix G of Anandkumar et al. (2012c). Without identifiability, we cannot hope to consistently recover parameters from pairwise moments alone.

4.3 Parameter recovery – known structure

The previous section gives us the tools to learn the noisy-or parameters associated with variables in singly-coupled triplets. However, not all of the edges in a network are part of singly-coupled triplets. For example, the QMR-DT network has 45,470 edges, but only 34,972 of the edges can be learned directly from singly-coupled triplets without additional steps.

In this section, we outline three main *steps* that can be performed when learning the parameters of a diagnosis network, and show how to combine them to learn almost all of the parameters in the QMR-DT network. Combined, these steps can be used to give a constructive proof of both identifiability and polynomial time learnability.

4.3.1 Step 1: Parameter learning from singly-coupled triplets

As described in the previous section (Section 4.2.2), we can learn the failure parameters and prior probabilities associated with edges that are part of a singly-coupled triplet.

4.3.2 Step 2: Extending a parent

Once a singly-coupled triplet is found, it can be used to learn *all* the parameters related to the coupling parent. If two children (A, B) are a singly-coupled pair¹ (with parent Y_i), and the prior probability of their parent is known, then we can determine the failure probability between the parent and *any* other observation X_j . Computing the conditional pointwise mutual information (CPMI) between A, B conditioned on X_j allows us to find the failure probability of $f_{i,j}$. The extending step thus allows us to learn parameters of *all* of the children of Y_i after finding one singly-coupled triplet. The intuition is as follows: conditioning on $X_j = 0$ *lowers* the likelihood of $Y_i = 1$, which in turn changes the mutual information between

¹if (A, B, C) is a singly-coupled triplet with parent Y_i then (A, B) is a singly coupled pair with the same parent.

A and B . The exact amount of change depends on the strength of the edges between Y_i and A, B and X_j . Since the parameters relating Y_i to A, B have been previously estimated, we can solve for the relationship with X_j .

First we define the pointwise mutual information (PMI) and then its conditional form:

$$\text{CPMI}(A, B) = \frac{P(\bar{A}, \bar{B})}{P(\bar{A})P(\bar{B})}. \quad (4.9)$$

$$\text{CPMI}(A, B|X_j) = \frac{P(\bar{A}, \bar{B}|\bar{X}_j)}{P(\bar{A}|\bar{X}_j)P(\bar{B}|\bar{X}_j)}. \quad (4.10)$$

Equations 4.11 and 4.12 show how conditional pointwise information can be used to determine $\pi_{i|\bar{X}} = P(Y_i = 1|X_j = 0)$.

$$\text{CPMI}(A, B|X_j) = \frac{(1 - \pi_{i|\bar{X}_j} + \pi_{i|\bar{X}_j}f_{i,A}f_{i,B})}{(1 - \pi_{i|\bar{X}_j} + \pi_{i|\bar{X}_j}f_{i,A})(1 - \pi_{i|\bar{X}_j} + \pi_{i|\bar{X}_j}f_{i,B})} \quad (4.11)$$

All the parameters in this equation other than $\pi_{i|\bar{X}_j}$ have already been estimated since A, B belong to a singly-coupled tuple. We can then solve for $\pi_{i|\bar{X}_j}$ as a root of a quadratic equation (Derived in Appendix D.1.2).

$$\pi_{i|\bar{X}_j} = \text{Roots}(f(x)) \quad (4.12)$$

where $f(x)$ has the form:

$$\begin{aligned} f(x) &= \text{CPMI}(A, B|X)(1 - f_{i,A})(1 - f_{i,B})x^2 \\ &\quad + (\text{CPMI}(A, B|X)(f_{i,A} + f_{i,B} - 2) - (f_{i,A}f_{i,B} - 1))x \\ &\quad + \text{CPMI}(A, B|X) - 1 \end{aligned}$$

There remains ambiguity over *which* root to pick, as $f(x)$ can potentially have two real roots. However, we show that one of the roots is always above 0.5 (see Appendix D.1.4). As long as we assume that the true priors are below 0.5, the true value remains identifiable. The final step is to use $\pi_{i|\bar{X}_j}$ to estimate $f_{i,j}$ (Derived in Appendix D.1.3).

$$f_{i,j} = \frac{(1 - \pi_i)}{\pi_i} \frac{\pi_{i|\bar{X}_j}}{(1 - \pi_{i|\bar{X}_j})} \quad (4.13)$$

4.3.3 Step 3: Subtracting-off parents

Once all of the parameters associated with a latent variable Y_i are estimated (i.e., its prior probability π_i , and the failure probabilities to all of its children, $f_{i,\cdot}$, Y_i can be effectively *removed* from the network, potentially uncovering more singly-coupled triplets. The subtracting-off step is based on the observation that the moments of observations in a noisy-or network have a form in which each latent parent's contribution can be isolated. After transforming the moments to that form, it is possible to subtract off the influence of parents whose parameters are known, leaving the observations *as though* that parent was not part of the network at all.

In order to understand the mechanisms of subtracting off, it is important to distinguish between two different representations of a distribution, which we call the *tensor moment* and the *negative moments*.

Definition 2. (*Tensor moment*) The tensor moment of observations \mathcal{X} is the expectation of the outer product of indicator vectors:

$$E_x \left[\otimes_{j \in \mathcal{X}} e_{x_j} \right],$$

where $e_0 = [1, 0]$ and $e_1 = [0, 1]$.

Definition 3. (*Negative moments*) The negative moments of observations \mathcal{X} is the expectation that subsets of \mathcal{X} are observed to be off simultaneously.

$$\{E_x [\mathbf{1}[\mathcal{S} = 0]]\}_{\forall \mathcal{S} \subset \mathcal{X}}.$$

For a tuple \mathcal{X} with three variables, the tensor moment is a $2 \times 2 \times 2$ tensor, while the negative moments are a set of 7 values in the unit interval.

The subtracting step operates on *negative* moments. Let \mathcal{X} be a set of observed variables and let $\bar{\mathcal{X}}$ denote the event where *all* members of \mathcal{X} are observed to be off (i.e., $\mathcal{X} = 0$). Equation 4.14 gives the probability in terms of the model parameters Heckerman (1990).

$$P(\bar{\mathcal{X}}) = \prod_i \left(1 - \pi_i + \pi_i \prod_{j \in \mathcal{X}} f_{i,j} \right) \quad (4.14)$$

We call each of the parents' terms its *influence* and use the notation $I(Y_i \rightarrow \mathcal{X})$ to denote the influence of Y_i on observations \mathcal{X} .

$$I(Y_i \rightarrow \mathcal{X}) = \left(1 - \pi_i + \pi_i \prod_{j \in \mathcal{X}} f_{i,j} \right) \quad (4.15)$$

Notably, for a parent Y_i whose prior π_i and whose failure probabilities $f_{i,\cdot}$ have previously been estimated, its influence can simply be divided out of Equation 4.14.

The subtracting-off step operates on negative moments, but the decomposition to recover

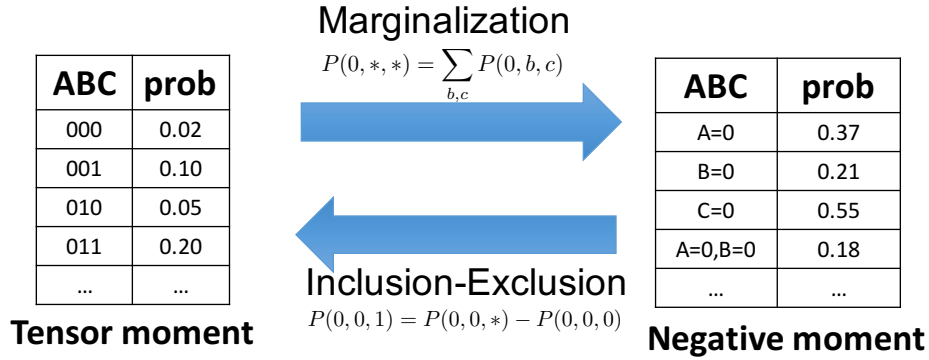


Figure 4.4: Transforming between tensor moments and negative moments

parameters from singly-coupled triplets requires tensor moments. This is not problematic since a tensor moment can be transformed to negative moments through marginalization and negative moments can be transformed to a tensor moment through the inclusion-exclusion formula. Since these moments are bounded in size (tuples in this chapter do not exceed quartets) none of these transformations are computationally expensive. To perform a subtracting-off step, we transform the tensor moment to a negative moment, subtract-off the relevant parents, and then transform back to tensor moment form (Figure 4.4).

4.3.4 Step 4: Repeat

Subtracting off a latent variable can uncover new singly-coupled triplets. We can then repeat Steps 1-3 in a loop until there are no more singly-coupled triplets to learn (Figure 4.5). At this point the algorithm terminates. There may still be some unlearned parameters, but the parameters that are learned in this process are consistently estimated. Algorithm 3 presents the complete learning procedure for learning with known structure. If the algorithm succeeds in learning all of the parameters of a network, then its parameters are learnable in polynomial time. Note whether a network is fully learnable or not can be determined from the network *structure* (by running Algorithm 3 without performing the parameter learning

Learning with known structure

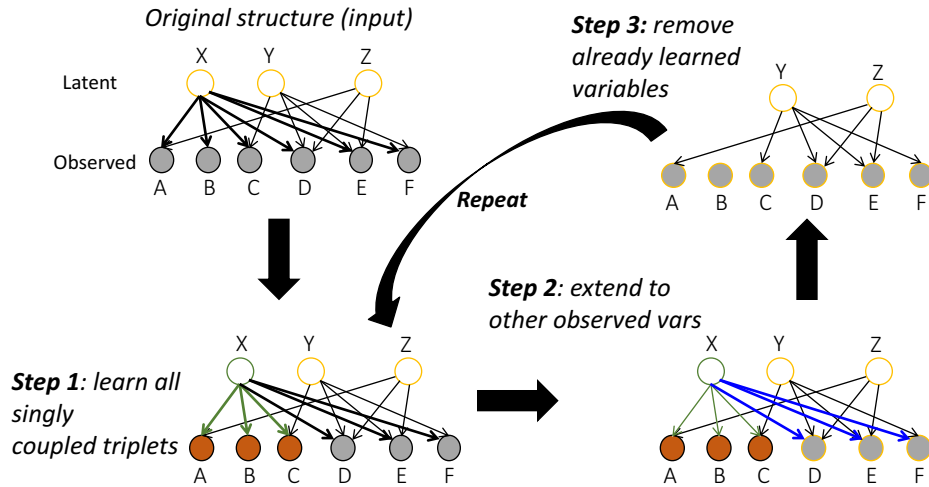


Figure 4.5: An example network to be learned. In the first step, we identify A, B, C as a singly-coupled triplet, coupled by X and learn its parameters. In the second step, we use the extend step to learn the edges from X to its other children (D, E, F). In the third step, we remove X from the network, using the subtracting off step to remove its influence from low order moments. After subtracting off, Y has a singly coupled triplet (C, D, F) and then we repeat the cycle again.

step).

4.3.5 Algorithmic analysis

In this section we show that the termination of Algorithm 3 with no unlearned parameters after a fixed number of iterations implies polynomial time learnability. The number of iterations before termination is the *depth* of the network.

4.3.5.1 Computational complexity

We can run the algorithm on the graph structure first to determine which moments need to be obtained from the data.

The outer loop of Algorithm 3 can be run at most n times, and each time it may have to

Algorithm 3 Known structure

Input: Graph structure \mathcal{G} , samples from the graph \mathcal{S}

```
1: while There are parents that can be learned do
2:   for  $Y_i \in \text{Conditions}$  do
3:     if  $Y_i$  has a singly-coupled triplet,  $t$  then
4:       Find blocking parents of  $t$  with respect to  $Y_i$ , and subtract them off.
5:       Learn parameters of  $Y_i \rightarrow t$ 
6:       Extend  $Y_i$ 
7:       Schedule  $Y_i$  for removal
8:     end if
9:   end for
10:  Remove all learned parents from  $\mathcal{G}$ 
11: end while
```

*Blocking parents of a triplet with respect to Y_i are parents (other than Y_i) that appear in the original graph and link to at least two members of the triplet.

search over $O(m^3)$ triplets to find a singly-coupled triplet. Extending the parent could involve iterating over m more children. Once a parent is learned, its influence can be subtracted off from each triplet containing its children, potentially adding another m^3 operations.

The complexity of this part is then: $O(nm^3)$.

Let N be the number of samples required to reach the desired accuracy. Reading the data requires at most $O(Nmn)$ updates to triplet moment counters since every edge requires only one singly-coupled triplet to estimate its parameters.

Thus, the final computational complexity is $O(nm^3 + Nmn)$.

4.3.5.2 Statistical complexity

Finally, we show that the number of samples required to learn the parameters of the network to arbitrary accuracy scales polynomially. Theorem 1 formalizes the consistency and finite sample complexity of the learning procedure as a function of the depth and bounds on the parameters and marginals of the model.

Theorem 1 (Learning with known structure). *Suppose a network with n latent parents and m*

observed variables is triplet-learnable at depth d . Let M_0 be the minimum marginal probability of an observation being off and let $\pi_{max} \leq 0.5, \pi_{min}$ be bounds on the prior probabilities, f_{max}, f_{min} be bounds on the failures and l_{max} be an upper bound on the leak parameters. The additive error on any of the parameters, ϵ is bounded with probability $1 - \delta$ by:

$$\epsilon(N) = O \left(\left(\frac{nM_0^6 \sqrt{\ln(\frac{2m}{\delta})}}{f_{min}^{18} (1 - f_{max})^6 l_{max}^{28} \pi_{min}^{13} \sqrt{N}} \frac{1}{\sqrt{N}} \right)^{2d} \right) \quad (4.16)$$

The proof to Theorem 1 is found in Appendix D.5. Algorithm 3 is a polynomial time algorithm that can achieve arbitrary accuracy in recovering the parameters of the network, thus it forms a constructive proof of polynomial time learnability for noisy-or networks with fixed depth.

4.4 Structure learning – discovering latent variables

Section 4.3 assumes that the complete structure of the Bayesian network is available and that all that remains to be learned are the parameters. This may be a reasonable setting for diagnosis networks where the latent variables are diseases and the observations are symptoms, since the relationships between diseases and symptoms (i.e., can disease A cause symptom B) are fairly well known. However, for more general EMR phenotyping tasks, where the phenotypes may be more broadly defined and the observations can be general features derived from unstructured fields of the EMR (e.g., unigrams or bigrams), it is much harder to specify the full graph structure connecting phenotypes and features.

In order to apply the algorithm described in Section 4.3 to the more general setting where the structure is not known in advance, the key step is to identify singly-coupled observations directly from data. We describe two methods of identifying singly-coupled observations from quartets, and use them to build a structure learning algorithm. The definitions of triplet

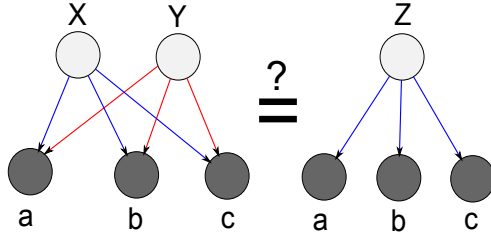


Figure 4.6: Two networks with the same moments matrix using parameters: $p_X = 0.2$, $p_Y = 0.3$, $p_Z = 0.37$. $f_X = (0.1, 0.2, 0.3)$, $f_Y = (0.6, 0.4, 0.5)$, $f_Z = (0.28, 0.23, 0.33)$. One is singly-coupled, the other is not. High precision values for these parameters are given in Appendix D.9,

learnability, and depth of a network from Section 4.3 transfer directly to the quartet setting as well. As in the previous section, the algorithms are presented as though perfect data is available (infinite data drawn from a distribution within the model family). The statistical complexity presented in Section 4.4.2 accounts for the noise introduced by observing a finite amount of data. Proofs of the correctness of the tests and the theorems in this section can be found in Appendix D.6 .

4.4.1 Identifying singly-coupled tuples

Although the parameters of a singly-coupled triplet can be identified from its data distribution, a non-intuitive result is that deciding *whether* a triplet is singly-coupled from data is not possible. Figure 4.6 gives an example of two networks that cannot be distinguished and Appendix D.9 gives more detailed information. One way to understand this non-identifiability is through the result of Kruskal (1989) that even though the maximal rank of a $2 \times 2 \times 2$ tensor is 3, the set of rank-2 tensors has non-zero volume, and in fact *most* $2 \times 2 \times 2$ tensors have a rank-2 decomposition.

While singly-coupled *triplets* are not identifiable from data, it turns out that singly-coupled *quartets* are. Two methods are available to determine whether a quartet is singly coupled: a rank test, and a coherence test.

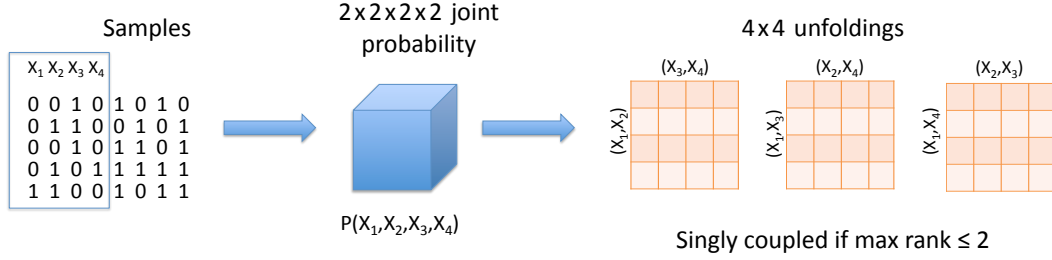


Figure 4.7: Schematic of the unfolded rank test. Samples are used to form a $2 \times 2 \times 2 \times 2$ tensor, which is unfolded to perform the unfolded rank test.

4.4.1.1 Rank test

The rank test consists of two sub-tests.

The first sub-test, compares pointwise information ($\text{PMI}(A, B)$ see Equation 4.9) and pointwise mutual information ($\text{CPMI}(A, B, C)$ see Equation 4.9) for all ordered triplets (A, B, C) (line 2 of Algorithm 4). If the CPMI is lower than the PMI for all ordered triplets (A, B, C) , there is at least one joint parent that is common to all of the members of the quartet.

The second, *unfolded rank test* checks whether the unfolded rank is less than or equal to 2. If it is, the quartet has no more than one parent (line 6 of Algorithm 4). This process is illustrated in Figure 4.7.

A positive result from both sub-tests implies that the quartet is singly-coupled and a negative result implies that it is not, giving a reliable method for testing whether a quartet is singly-coupled. Lemma 1 gives the formal properties of the rank test and its pseudocode is presented in Algorithm 4.

Lemma 1 (Rank test). *For an ϵ -rank-testable model, let (A, B, C, D) be a quartet of observed variables. The following two conditions hold iff (A, B, C, D) is a singly-coupled quartet:*

1. *The third eigenvalue of all 4×4 unfoldings of the $2 \times 2 \times 2 \times 2$ tensor moment is less than ϵ*
2. *For all ordered triplets $\{A', B', C'\} \subset \{A, B, C, D\}$, $\text{CPMI}(A', B' | C') \leq \text{PMI}(A', B')$.*

Algorithm 4 Rank Test

Input: quartet (a, b, c, d) , thresholds τ_e, τ_r
for triplet (x, y, z) in (a, b, c, d) **do**
 Test $\text{CPMI}(x, y) \leq \text{PMI}(x, y|z) - \tau_e$
end for
 $T_{a,b,c,d}$ = fourth order tensor moment of (a, b, c, d)
for unfolded matrices M_i of $T_{a,b,c,d}$ **do**
 Test third eigenvalue of $M_i \leq \tau_r$
end for
if all tests pass **then**
 Return True
else
 Return False
end if

The unfolded rank test allows us to determine whether a candidate quartet is singly coupled by testing that the third eigenvalues of the unfolded matrices are strictly greater than 0. However, for the algorithm to be practical with finite data, we need a slightly stronger formalization of the property, which we call ϵ -rank-testability:

Definition 4. *We say that a model is ϵ -rank-testable if for any quartet $\{a, b, c, d\}$ that share a parent Y_i and are not singly-coupled, the unfolded rank test gives a value greater than ϵ .*

Rank testability holds for all but a very small number of parameter settings. For randomly drawn parameters, with probability one there is an ϵ such that the model is ϵ -rank-testable (since the determinant of the unfolded matrices is a polynomial function of the parameters of the model, its roots form a zero measure set). This remains true even if all of the failure probabilities are constrained to be equal.

Even though rank testability holds with probability 1, that does not give much information on the more practical notion of ϵ -rank. Figure 4.8, however provides an approximation of the probability density function of the third eigenvalue of the moments matrix of a noisy-or network whose parameters are drawn from a uniform distribution.

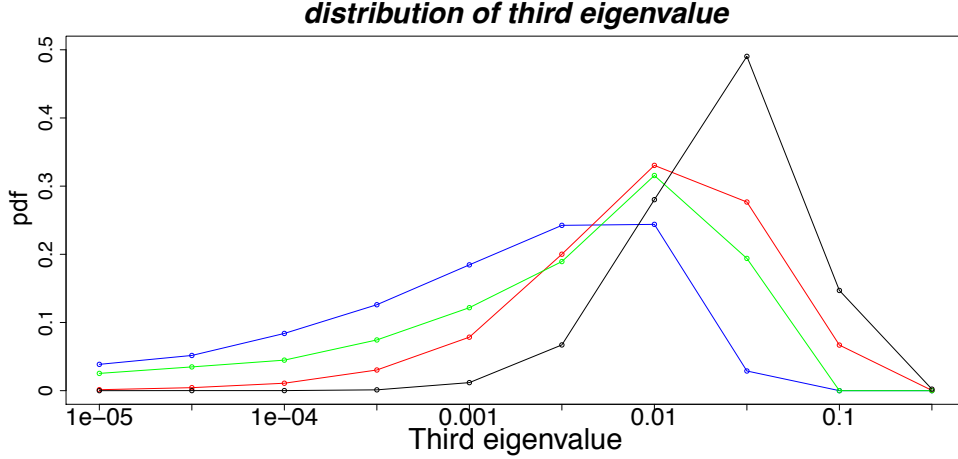


Figure 4.8: Probability distribution of the third eigenvalue for a quartet with coupling parents, when all parameters are drawn uniformly at random. **Blue:** two parents, same failure probabilities. **Green:** three parents, same failure probabilities. **Red:** two parents, different failure probabilities. **Black:** three parents, different failure probabilities. This plot illustrates that rank testability only gets easier for random parameters compared to the uniform parameter setting used in the experiments.

4.4.1.2 Coherence test

The coherence test checks whether the fourth-order tensor moment could have come from a singly-coupled quartet by breaking the quartet down into triplets and learning the parameters as though the triplets are singly coupled. The parameters from each triplet are then combined to build a model that contains all four children, and checks whether the observed fourth-order tensor moment matches the moment predicted by the model to within a tight tolerance ϵ . Similar to the rank test, this test returns true if and only if the quartet is singly coupled. Lemma 2 gives the formal properties of the coherence test and its pseudocode is presented in Algorithm 5.

Lemma 2 (Coherence test). *For an ϵ -rank-testable model and a quartet of observed variables, (A, B, C, D) . Let T_R represent the reconstructed fourth-order tensor moment and T the*

Algorithm 5 Coherence Test

Input: quartet (a, b, c, d) , threshold τ
for T in $\{T_{a,b,c}, T_{a,b,d}, T_{a,c,d}, T_{b,c,d}\}$ **do**
 Estimate failures, priors and noise from T
end for
 $\theta =$ Average multiple estimates for each parameter
 $T_\theta = p(a, b, c, d; \theta)$
 $T_{a,b,c,d} =$ fourth order tensor moment of (a, b, c, d)
return $|T_{a,b,c,d} - T_\theta|_\infty \leq \tau$

empirical fourth order tensor moment, we have:

$$\|T_R - T\|_\infty > \left(\frac{\epsilon}{8}\right)^4 \text{ iff } (A, B, C, D) \text{ are singly coupled.} \quad (4.17)$$

This can be proved using a result on eigenvalue perturbation from Elsner (1985) for an unfolding of the moments' tensor.

4.4.2 Algorithm and analysis

The complete structure learning algorithm is similar to the algorithm developed in the previous section for the known-structure setting (Algorithm 3), but replaces the structural test for singly-coupled triplets with an empirical test for singly coupled quartets.

Computational complexity Algorithm 6 runs in $O(n^2m^5 + Nm^4)$. This is markedly higher than the $O(nm^3 + Nmn)$ complexity of the known-structure setting, particularly because every fourth order tensor moment is formed, requiring $O(Nm^4)$ data accesses. This complexity can be reduced in practice by only considering quartets whose constituent triplets have high conditional pointwise mutual information (i.e., pass the second sub-test of the rank test in Section 4.4.1.1), though the worst-case analysis is unchanged.

Algorithm 6 Structure learning

```
1: Parents =  $\emptyset$ 
2: while There are parents that can be learned do
3:   for quartet  $q$  in  $\mathcal{X}$  do
4:      $T$  = fourth order tensor moment
5:     Subtract off learned parents from  $T$ 
6:     if  $q$  is singly-coupled* then
7:       Learn parameters  $Y_{\text{new}} \rightarrow q$ 
8:       Extend  $Y_{\text{new}}$ 
9:       Parents = Parents  $\cup Y_{\text{new}}$ 
10:    end if
11:  end for
12: end while
```

*Testing whether a quartet is singly-coupled can be done with the Rank test (Alg. 4) or the Coherence test (Alg. 5) on the tensor moment T .

Statistical complexity Theorem 2 formalizes the statistical complexity of the structure learning algorithm described in this section (Proof in Appendix D.6).

Theorem 2 (Learning with unknown structure). *Suppose a network with n latent variables and m observed variables is quartet-learnable at depth d and is ζ -rank-testable. Let M_0 be the minimum marginal probability of an observation being off; $\pi_{\max} \leq 0.5, \pi_{\min}$ be bounds on the prior probabilities; f_{\max} , and f_{\min} be bounds on the failures and l_{\max} be an upper bound on the leak probabilities. Its structure can be learned with probability $(1 - \delta)$ with N_S samples, where:*

$$N_S = O\left(\left(\frac{nM_0^6}{f_{\min}^{18}(1-f_{\max})^6 l_{\max}^{28} p_{\min}^{13}}\right)^{2d} \times \max\left(\frac{1}{\zeta^8}, \frac{1}{l_{\max}^8 p_{\min}^2 (1-f_{\max})^8}\right) \ln\left(\frac{2m}{\delta}\right)\right) \quad (4.18)$$

Corollary 1 (Parameter error). *For the network described in Theorem 2 and N samples where $N \geq N_S$, the parameter error in the learned network is bounded with high probability*

by:

$$\epsilon(N) = O \left(\left(\frac{nM_0^6 \sqrt{\ln(\frac{2m}{\delta})}}{f_{min}^{18} (1 - f_{max})^6 l_{max}^{28} p_{min}^{13} \sqrt{N}} \right)^{2d} \right) \quad (4.19)$$

Corollary 1 is a direct application of Theorem 1 with the number of samples required to learn the structure from Theorem 2.

4.5 Experiments

In this section, we compare the method of moments algorithms for learning with known-structure and unknown structure against the variational baselines on synthetic data. We analyze the structure of the QMR-DT network to determine whether it is learnable with triplets and quartets, and show empirically that many of its parameters can be accurately recovered with a reasonable amount of data.

4.5.1 Parameter recovery in simple networks

We test the method described in Section 4.3 at recovering the parameters of a simple network. Samples are drawn from 64 different networks with the structure shown in Figure 4.9, the simplest structure which requires the subtracting-off step described in Section 4.3.3. Each network instance has failure and prior probabilities drawn uniformly at random between $[0.2, 0.8]$. Leak probabilities are all set to $l_j = 0.01$. We compare against exact and variational EM. Figure 4.10 shows the L1 parameter error of the different methods after convergence and a timing comparison between our method-of-moments algorithm and the variational EM algorithms. Exact EM requires exact inference which is intractable for all but the simplest networks.

Even in the simple network setting, variational EM is not a consistent estimator. Even with a large number of samples, its L1 error in parameter estimation does not go below a

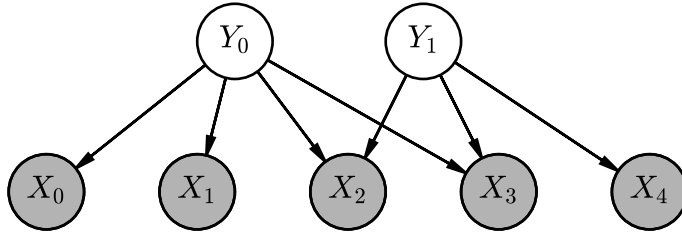


Figure 4.9: Simple network used in synthetic experiments. To learn, first estimate the parameters of Y_0 using the singly-coupled triplet $\{X_0, X_1, X_2\}$, then learn Y_1 whose children are singly-coupled after subtracting off Y_0 .

naive strategy of estimating 0.5 for every parameter. Variational EM performs variational inference as an inner-loop of learning, making it scale slowly with the size of the dataset. The method of moments algorithm reads in the whole dataset once, but then works on aggregate statistics, making the dependence on the size of the dataset less pronounced.

4.5.2 Learning with unknown structure – synthetic images

We test the method described in Section 4.4 at recovering the latent structure of a dataset on a synthetic image dataset from Šingliar and Hauskrecht (2006). Each pixel in an 8×8 image patch is an observed binary variable. Each latent variable connects to the pixels in a different pattern. Figure 4.11 shows the patterns that were used and representative samples from the model. Each source was given a 0.25 prior probability, all edges were given a failure probability of 0.2 and a leak probability of 0.01. The structure learning task is to recover the sources from their noisy combinations.

The resulting network is quartet learnable, and requires subtracting-off in order to learn the first and final sources in Figure 4.11.

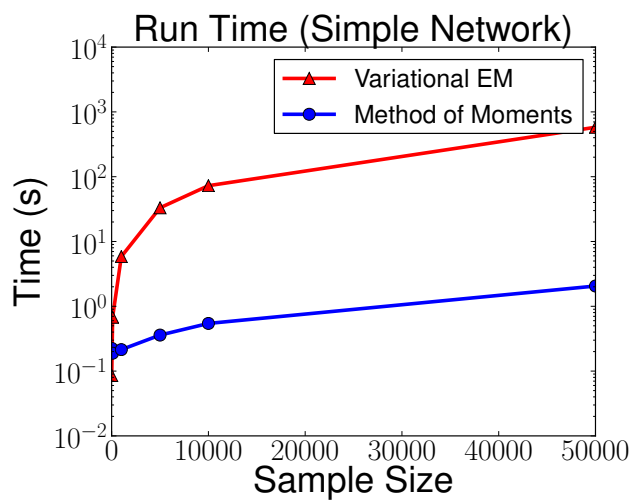
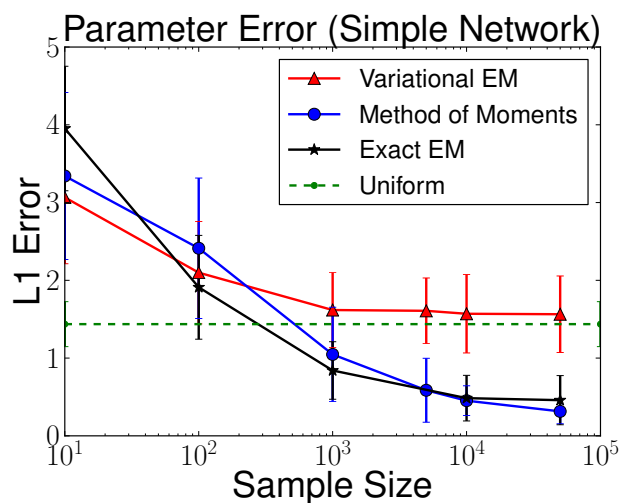


Figure 4.10: (left) Parameter error (L1) for the simple network as a function of number of samples. The uniform strategy estimates 0.5 for every failure probability. (right) comparison of running time between variational learning and method of moments.



Figure 4.11: (Top row) Synthetic image “sources”. The first and last sources (rhombus and filled square) cannot be learned without subtracting off the other sources. (Bottom row) Samples drawn from the network.

Since evaluating the likelihood of noisy-or networks is computationally intractable, we use the error in pairwise mutual information (Equation 4.9) to determine hyperparameters (i.e. rank test or coherence test and the values of the thresholds).

Figure 4.12 shows the recovered sources as a function of the number of samples. For comparison, we also show (Figure 4.13) the sources learned by variational EM (Šingliar and Hauskrecht, 2006) and variational stochastic gradient ascent with a recognition model from Chapter 3 (Figure 4.14). The stochastic gradient ascent method is used here without the semi-supervised objective described in Chapter 3 as that objective is specific to the anchored setting. Both of the baseline models require that the number of sources be set in advance, so we learn separate models with 8, 12, and 16 sources.

With 1000 samples, the method of moments is able to perfectly recover the structure of the sources in approximately 4 minutes using 8 processes on AMD-based Dell R815 machine with 64 cores and 256GB RAM. The 6 sources that are learned without subtracting off are already discovered with 500 samples. The variational EM approach is not able to perfectly recover the sources, even with 10000 samples and 8 random restarts, giving each restart one hour to run. The stochastic variational gradient ascent approach is surprisingly effective, learning to recover the sources with only 200 samples and 8 random restarts, giving each restart only 5 minutes to run. Providing the stochastic variational gradient ascent method with extra sources during learning (the third setting in Figure 4.14) is essential to its good performance, allowing the model to avoid poor local optima; the final learned model uses the correct number of sources (additional sources have priors very close to 0 by the end of the

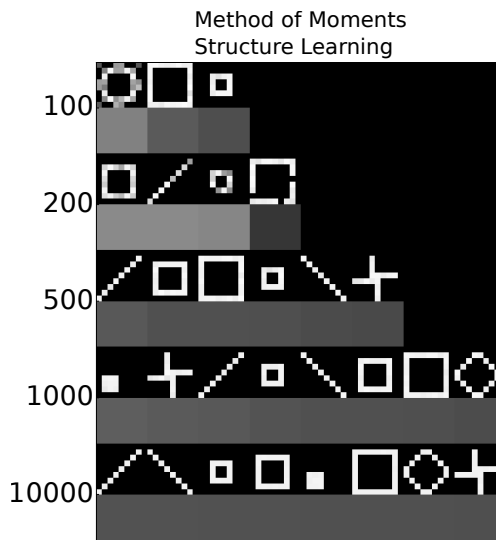


Figure 4.12: Recovery of the synthetic images using the proposed quartet test method for structure learning and parameter estimation. Each row represents sources learned from a different number of sources. The shaded box underneath each source represents the prior probability that the model assigns to that source.

optimization procedure).

4.5.3 Learnability of QMR-DT

In this section we assess how much of the QMR-DT network could we learn in the parameter learning setting where the structure is known, and the structure learning setting where it is not known in advance. Table 4.1 describes the triplet and quartet learnability of the QMR-DT network using the algorithms from Section 4.3 and 4.4.

Approximately 95% of the parameters in the known structure setting and 87% in the unknown structure setting can be learned without any subtracting off at all. The surprising result is that *almost* all of the QMR-DT network can be learned using singly coupled triplets or quartets, suggesting that the sufficient conditions for learnability described in Sections 4.3 and 4.4 include a non-trivial set of networks. The 122 edges that cannot be learned in the known structure setting (marked as depth= ∞ in Table 4.1) belong to two diseases in the

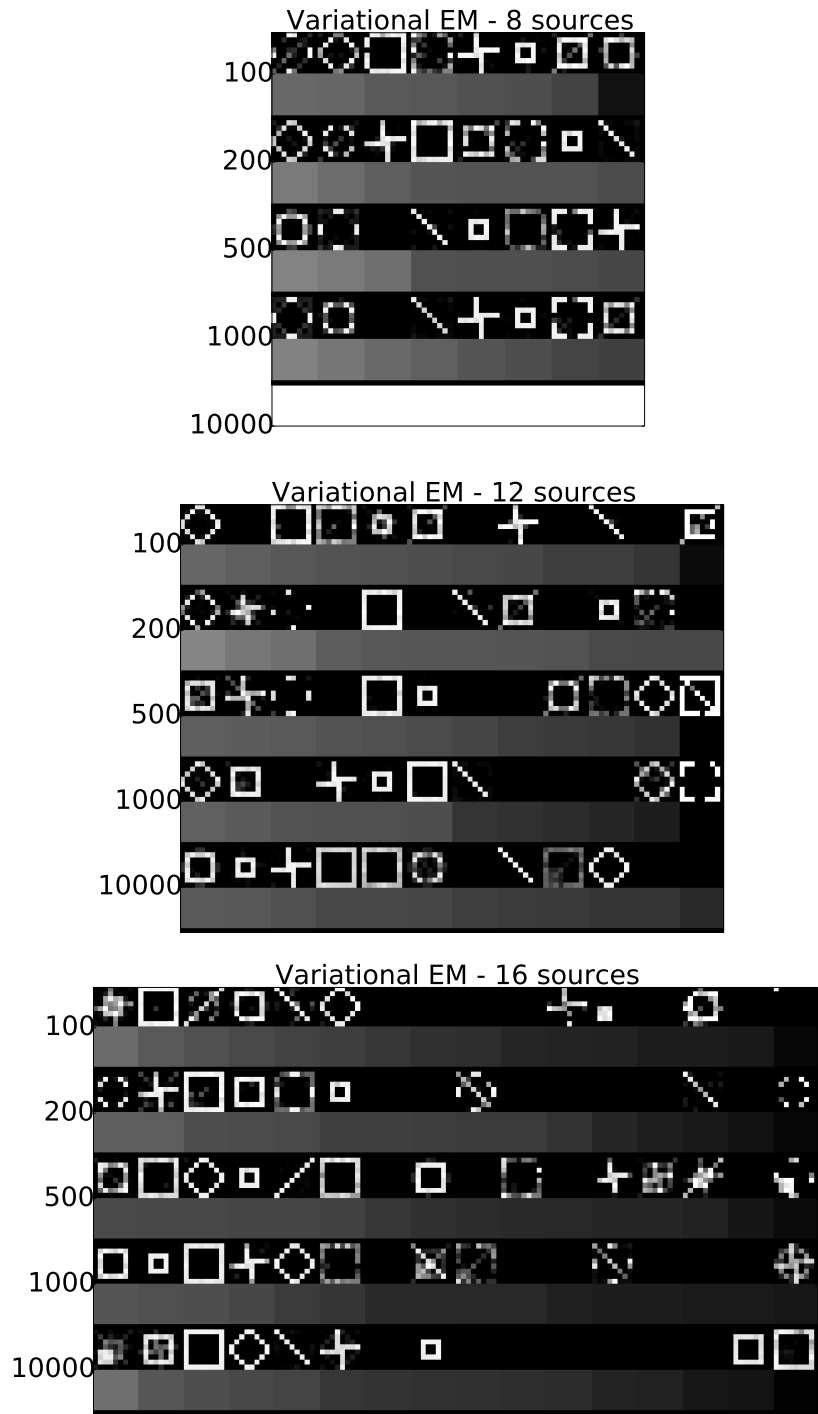


Figure 4.13: Recovery of the synthetic images using variational EM with 8 random restarts and allowing for 8, 12, and 16 sources. Each row represents sources learned from a different number of sources. The shaded box underneath each source represents the prior probability that the model assigns to that source. None of the runs in the last row with 8 sources ($N=10000$) finished in the allotted 1 hour time limit.

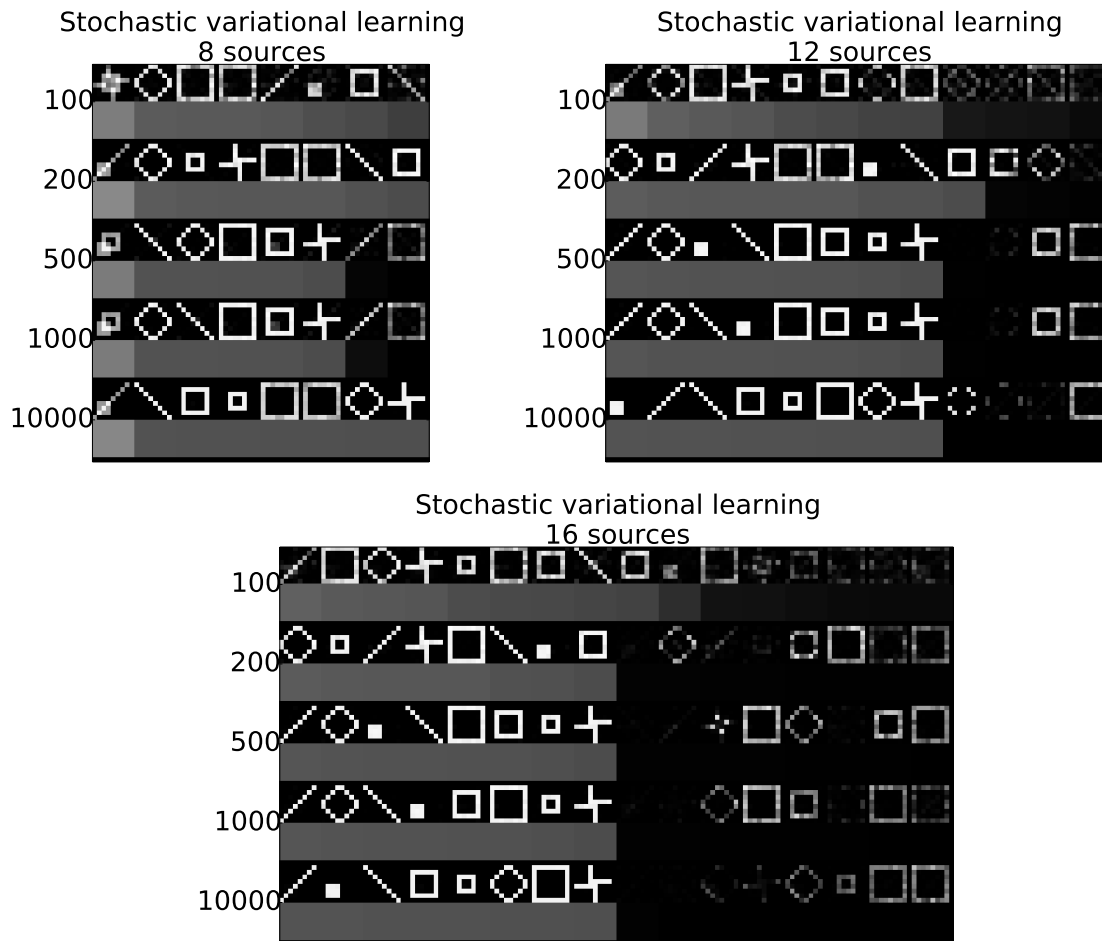


Figure 4.14: Recovery of the synthetic images using variational stochastic gradient ascent with 8 random restarts and allowing for 8, 12, and 16 sources. Each row represents sources learned from a different number of sources. The shaded box underneath each source represents the prior probability that the model assigns to that source. Training was run for 5 minutes.

depth	triplets (known structure)		quartets (unknown structure)	
	diseases processed	edges learned	diseases discovered	edges learned
0	527	43,139	469	39,522
1	39	2,109	82	4,875
2	2	100	13	789
3	0	0	2	86
inf	2	122	4	198

Table 4.1: Learnability of the QMR-DT network. Diseases and edges learned at lower depth are learned with higher precision.

QMR-DT network, each with 62 children symptoms, that so densely overlap that 61 of the children are shared between the two diseases (DX134 and DX472 of the anonymized QMR-DT network). In the unknown structure setting, where quartets are required instead of triplets, all but four of the diseases can be discovered and their parameters learned.

4.5.4 Empirical statistical complexity

To show the empirical statistical complexity of the learning procedure, we plot the error in the failure probabilities estimated in the known structure setting, as a function of the number of samples, and stratified by the depth at which the parameters are learned. Samples are drawn from the anonymized QMR-DT network using the parameter settings described in Morris (2001). Figure 4.15 shows the results.

The sample complexity bounds are quite loose as most of the parameters (i.e., those learned without subtracting off) can be estimated to within 0.05 of the correct failure probability using 1 million synthetic samples. This is a reasonable scale for a large collection of medical records. For comparison, the emergency department dataset from Beth Israel Deaconess Medical Center (Chapters 2 and 3) has at least 200,000 patient visits collected over five years. The edges learned at a higher depth do show a loss in accuracy, as expected from the effects of the subtracting off step, but they become increasingly more accurate as the amount

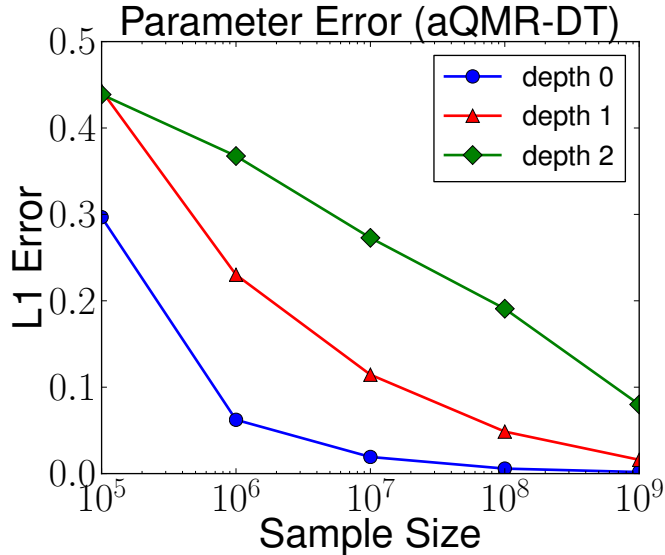


Figure 4.15: Mean parameter error (L1) in the failure estimates of the QMR-DT network as a function of the number of synthetic samples to learn from. Each line represents edges learned at a different depth (defined in Section 4.3.5).

of data increases.

4.6 Related work

Tensor decompositions have been used for proving global identifiability and learning in latent variable models (Lazarsfeld, 1950; Chang, 1996; Allman et al., 2009; Anandkumar et al., 2014b). The orthogonal tensor decompositions that are used to recover parameters in a variety of latent variable models where the low order moments have a multi-linear form (e.g., Anandkumar et al., 2012b,c, 2013a,b, 2014a,b) do not apply to noisy-or networks. Instead of using a single decomposition to determine all of the parameters at once, we find small parts of the network whose parameters can be learned through tensor decomposition and build up a collection of parameters until we cannot learn anymore.

For binary data, tetrachoric correlations can be used to reduce the learning problem to the traditional linear-Gaussian setting (Christofferson, 1975). Martin and VanLehn (1995)

greedily introduce latent factors that cover the dependencies on which they are most certain as a method to learn noisy-or factor analysis models. Kearns and Mansour (1998) describe a family of noisy-or networks that are learnable from data in polynomial time, but are limited to networks where the prior probabilities of each latent variable are equal, the in-degree of every observation is bounded by a constant, and the failure probabilities come from a finite set that can be enumerated. Exact inference in noisy-or networks is NP-hard (Cooper, 1987). Šingliar and Hauskrecht (2006) uses noisy-or networks for exploratory factor analysis using the variational approximation to inference from Jaakkola and Jordan (1999) within EM, but it does not have consistency guarantees and is susceptible to finding poor local optima. This approach is used as a baseline in Section 4.5. Wood et al. (2006) and Courville et al. (2009) use Markov chain Monte Carlo approaches to parameter estimation, but these methods do not provide consistency guarantees for parameter recovery. In contrast, the method-of-moments approach is both consistent and globally optimizable.

Previous work has used structural signatures similar to the singly-coupled quartet setting for structure learning of Bayesian networks. Elidan et al. (2001) uses semi-cliques to infer the presence of latent variables in a heuristic manner. Silva et al. (2006) gives quartet conditions under which the presence of a latent variable can be determined in linear models. Pearl and Tarsi (1986) and Ishteva et al. (2013) use similar fourth order tests to learn latent tree structures. Chaganty and Liang (2014) use singly-coupled triplets to recover parameters in more general graphs, but do not address the question of structure learning.

Steinhardt and Liang (2016) use the singly-coupled triplet assumption for unsupervised *risk estimation* to determine the performance characteristics of a classifier on a new distribution where the singly-coupled condition is assumed to hold.

4.7 Conclusions

We develop new polynomial time algorithms with provable guarantees for parameter and structure learning in noisy-or diagnosis networks that depend on singly-coupled triplets or quartets. We demonstrate the speed and accuracy of the parameter and structure estimates on synthetic data. The new method outperforms variational EM and is comparable to the stochastic variational gradient ascent method described in Chapter 3 for parameters that can be learned without using the subtracting off step. The subtracting off step severely reduces the statistical efficiency of the method, but real-life networks like the QMR-DT network (Shwe et al., 1991) only require a small number of subtraction steps to learn almost all of the parameters of the model.

4.8 Next steps and open questions

The method-of-moments approach is exciting in that it can provide provable polynomial time algorithms with provable guarantees, though many open questions remain for consideration.

4.8.1 Necessary conditions for learnability

The learning algorithms presented in this chapter present *sufficient* conditions for learnability, but do not address the polynomial time learnability of networks that do not have singly-coupled triplets (or quartets).

One avenue of exploration is understanding the *identifiability* of networks that are not triplet or quartet learnable. If a network is not identifiable, then it is impossible to recover its parameters uniquely from data. If it is identifiable, then there is hope that a polynomial time recovery algorithm exists. Testing global identifiability is difficult, but Hsu et al. (2012)

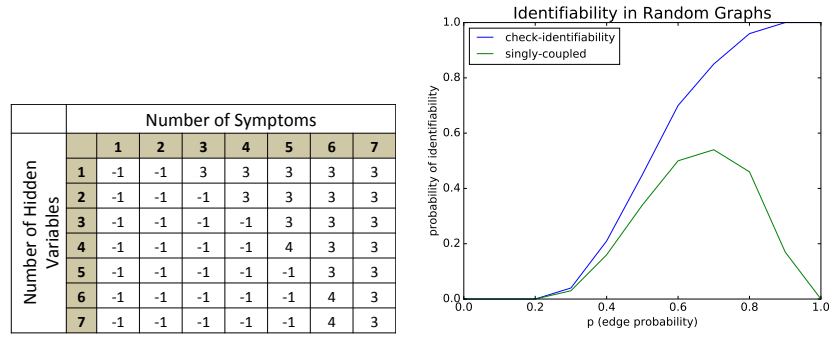


Figure 4.16: (left) Empirical identifiability for $n \times m$ complete bipartite graphs. The values in the cells reflect the order of moments required to obtain identifiability. -1 denotes that model is *unidentifiable* no matter what order of moments are used. (right) Identifiability in random graphs with 2 parents and 6 children as a function of the edge likelihood, p . The *singly-coupled* method uses Algorithm 3 to test identifiability.

describes the Check-Identifiability routine to check the weaker *local* identifiability of a network given its structure. We next present some results using that routine to show that there is still a gap between what we know how to learn and what we know is non-identifiable:

Fully-connect bipartite graphs Figure 4.16 shows the order of moments required for local identifiability on fully connected $n \times m$ bipartite graphs. Although there are no singly-coupled tuples in a fully connected graph, many of them are locally identifiable from low-order moments, suggesting that other efficient learning algorithms could exist.

Random graphs Figure 4.16 shows the results of the Check-Identifiability routine, and tests learnability with singly-coupled triplets on 2×6 bipartite random graphs where each edge is present or absent with likelihood p . The Check-Identifiability routine finds local identifiability even in dense graphs, whereas the singly-coupled triplets works only in mid-range levels of sparsity, as singly-coupled triplets do not occur if the graph is too sparse or too dense.

Can polynomial-time learning algorithms (e.g., up to label switching) be formulated for all models which are identifiable?

4.8.2 Similar analyses of other parametrizations

Noisy-or is similar to single-layer sigmoid belief networks, with a small difference in the normalization. This difference makes the subtracting-off step difficult, but the rest of the analysis can be applied to single-layer sigmoid networks as well. Extending to deep networks is more challenging. Similar learning approaches has been applied to latent tree models (Pearl and Tarsi, 1986; Anandkumar et al., 2011; Ishteva et al., 2013) but finding ways to apply to deep networks with higher connectivity is an open problem.

For linear Gaussian models, finding singly coupled quartets from data (Silva et al., 2006), learning with singly-coupled triplets and subtracting-off (from the covariance matrix) all apply. The extending step is more difficult to formulate with continuous variables.

4.8.3 Learnability for likelihood-based approaches

Can the singly-coupled conditions tell us anything about the learnability of models using likelihood-based optimization? Are models that have singly-coupled tuples inherently “easy” to learn, even for likelihood-based methods? We find that variational stochastic gradient ascent with a recognition model is able to perfectly recover the structure and parameters in the synthetic image dataset (Section 4.5.2), but variational EM is not. Both approaches optimize a non-concave objective, but stochastic gradient ascent has more ability to get out of shallow local optima. Can we characterize the local optima of the objective when singly-coupled tuples are present to show that these networks are easier to learn?

4.8.4 Subtracting off and dependence on depth

The subtracting-off procedure introduces parameter error that compounds exponentially in the depth. Is there another way of estimating these parameters while controlling the parameter error that comes from the subtracting off step? Similar to the previous question,

we can ask if this dependence on depth fundamental is and whether the networks that require many subtracting-off steps are hard to learn? Or, is this specific to the particular algorithm that we chose?

4.8.5 Synthesizing multiple parameter estimates

Each failure probability may be estimatable from multiple singly-coupled triplets. What is the best way to combine those parameter estimates? Naive averaging and bootstrapping confidence intervals give a small improvement over random choice, but there may be better ways of performing the tensor decompositions jointly to obtain better parameter estimates all around.

4.8.6 Robustness to model error and applications to real data

The presentation of this chapter relies on the assumption that the data is drawn from the model family (or very close to it, so that the error due to misspecification is close to statistical error). What happens when the data is drawn from real distributions which are more complex than the parametric form? Ensuring robustness of method-of-moments algorithms is a general open question. Recent works have considered constrained optimization and projection as a general technique for increasing robustness (e.g., Shaban et al., 2015; Cohen et al., 2013; Duchi et al., 2008; Lee et al., 2015; Halpern et al., 2015; Arora et al., 2013). In this work, each tensor decomposition is solved disjointly and it is not obvious where and how to apply meaningful constraints or projections.

4.8.6.1 Discovering new phenotypes

Most of the work in this thesis assumes that there is a pre-existing collection of phenotypes of interest. This makes sense in some specialties (e.g., cardiology) where the set of phenotypes

are well understood . In other specialties (e.g., psychiatry), there is still active debate about the definitions of disorder phenotypes. For infectious diseases, there is always the potential that a new disease could arise through mutation or changing environmental factors.

The structure learning algorithm in this chapter raises the possibility of discovering *new* phenotypes. This could be done by first *subtracting off* the effects of already known phenotypes, and then learning from the residual distribution. The ability to discover new phenotypes could contribute towards our understanding of medicine, and serve as an early detection system for disease outbreaks. More broadly, the structure learning algorithm could be a new data science tool for performing factor analysis with discrete data.

Chapter 5

Learning a health knowledge graph from electronic medical records

Acknowledgments This work was joint with Maya Rotmensch, Abdul Tlimat, Steve Horng, and David Sontag. Thanks to Google for providing their health knowledge graph for the evaluation.

5.1 Introduction

Automated tools to support medical diagnostic reasoning are used by patients seeking information about their symptoms (Gann, 2011; Paparrizos et al., 2016; Tang and Ng, 2006; White and Horvitz, 2009), as well as by clinicians when faced with a difficult case or to avoid prematurely focusing on a small number of potential diagnoses (Groopman and Prichard, 2007). Considerable effort has been put into building diagnostic reasoning systems and encoding relevant information into accurate knowledge bases (Barnett et al., 1987; Bisson et al., 2014; Lally et al., 2014; Ramnarayan et al., 2004; Shwe et al., 1991). These models showed significant success in improving didactic practices (Miller and Masarie Jr, 1989; Warner

et al., 1988), assisting with diagnosis (Barnett et al., 1987; Bisson et al., 2014; Ramnarayan et al., 2004; Van Melle, 1978; Miller, 1994), and at times even outperforming experienced doctors (De Dombal et al., 1972).

Previous chapters of this thesis focused on building models using general features from electronic medical records including free text features, medications and lab results. In those chapters, the inference task was emphasized, and the learned features were *associated* with medical conditions, but not necessarily caused by them. For example, the models in Chapter 3 (Table 3.4) list the following features as relevant: “albuterol sulfate” for asthma (a treatment); “sober” for alcoholism (relevant history); “urology” for hematuria (specialty). While these associated features are helpful for inference in the setting where they are learned (i.e., in hospital with trained professionals taking history and writing notes), they are not able to provide information in more abstract settings like self triage from a list of symptoms or providing information about a disease.

In this chapter, we present a scalable method for learning knowledge bases from electronic medical record (EMR) data. We address the question: “Can EMRs serve as a source for discovering generalizable medical knowledge?” by attempting to uncover scientifically meaningful, causal relationships between diseases and symptoms. The discovered relationships are formally assessed by comparing them with the opinions of domain experts in both a specialist (emergency department) and generalist setting.

Historically, the knowledge bases used by diagnostic reasoning systems were specified manually, requiring tremendous amounts of expert time and effort. For example, it was estimated that about twenty person-years were spent building the Internist-1/QMR knowledge base for internal medicine (Shwe et al., 1991). However, the manual specification made these models extremely brittle and difficult to adapt to new diseases or clinical settings. Automatic compilation of a graph relating diseases to the symptoms that they cause has the potential to significantly speed up the development of such diagnosis tools. Moreover, such graphs would

provide value in and of themselves. For example, given that general-purpose web-search engines are among the most commonly consulted sources for medical information (White and Horvitz, 2009; Hider et al., 2009), health panels such as those provided by Google using their health knowledge graph (Pinchin, 2016; Ramaswami, 2016) have a tremendous potential for impact.

Previous work considered the use of natural language processing to find relationships between diseases and symptoms from unstructured or semi-structured data. For example, IBMs WatsonPaths and the symptom checker Isabel made use of medical textbooks, journals, and trusted web content (Lally et al., 2014; Ramnarayan et al., 2004). However, the electronic medical record, which has become increasingly prevalent in the United States and worldwide (Charles et al., 2013), is still underutilized.

EMR data is more difficult to interpret than textbooks, journals, and web content for four main reasons: First, the text of physician and nursing notes is less formal than that of traditional textbooks, making it difficult to consistently identify disease and symptom mentions. Second, textbooks and journals often present simplified cases that relay only the most typical symptoms for didactic purposes. In contrast, EMRs present real patients with all of the comorbidities, confounding factors, and nuances that make them individuals. Third, unlike textbooks that state the relationships between diseases and symptoms in a declarative manner, the associations between diseases and symptoms in the EMR are statistical, making it is easy to confuse correlation with causation. Finally, how observations are recorded in the EMR is filtered through the decision-making process of highly trained nurses and physicians. Information deemed irrelevant may be omitted or not pursued, leading to information missing not at random (Weiskopf et al., 2013).

Despite EMR data being more difficult to work with for the reasons described above, it has the advantage of being closer to the actual practice of medicine than the idealized and curated information presented in textbooks and journals. For example, learning from EMRs provides

the opportunity to discover new relationships that were previously unrecognized. Additionally, we can learn specialized graphs with different granularity for different specialties or settings by simply learning models from the records of patients from that setting. Finally, learning a causal graph relating diseases to symptoms from EMRs is the first step toward learning models that perform diagnostic inference directly from the real data that is continuously being generated from the healthcare system.

5.1.1 Related work

In recent work, Finlayson et al. (2014) quantify the relatedness of 1 million concepts by computing their co-occurrence in free-text notes in the EMR, releasing a graph of medicine. Sondhi et al. (2012) measure the distance between mentions of two concepts within a clinical note for determination of edge-strength in the resulting graph. Goodwin and Harabagiu (2013) use natural language processing to incorporate the belief state of the physician for assertions in the medical record, which is complementary to and could be used together with our approach. Importantly, whereas the aforementioned works consider purely associative relations between medical concepts and evaluate on auxiliary tasks such as information retrieval, our focus is on deriving a causal graph relating diseases to symptoms and we directly evaluate the learned relationships against the opinions of domain experts.

5.1.2 Contributions

The contributions of this chapter are as follows:

- We present simple methods for automatically deriving a causal graph relating diseases to the symptoms that they cause from EMR data.
- We apply the method to a corpus of emergency department records comprising over 270,000 patient visits and evaluate the graph structure directly, comparing it with

the Google health knowledge graph and through manual review by physicians. We conclude that many of the relationships learned from the electronic medical record are scientifically valid.

- With respect to generalization from one domain to the other, we document some of the challenges that arise in translating between the acute care (emergency department) and generalist (Google health knowledge graph) settings.

5.2 Constructing a knowledge graph

Learning the knowledge graph consists of three main steps. First, positive disease and symptom mentions were extracted from structured data and unstructured text (detailed in ‘Data collection and preparation’). Second, statistical models relating diseases and symptoms were learned. Third, the learned statistical models were translated into knowledge graphs. The overall procedure is summarized in Figure 5.1.

5.2.1 Parameter estimation

We considered three statistical models: logistic regression (LR), naive Bayes (NB) and a Bayesian network modeling diseases and symptoms with noisy or gates (noisy or). Logistic regression, which is widely used for binary classification, was chosen as an example of a well-established machine learning classifier with interpretable parameters that is frequently used for modeling binary outcomes (Hastie et al., 2009). Naive Bayes was chosen as it provides a baseline of what can be inferred from simple pairwise co-occurrences (Murphy, 2012). Noisy or was chosen as an example of a probabilistic model that jointly models diseases and symptoms; similar models have successfully been used in medical diagnosis applications (Shwe et al., 1991; Miller et al., 1982).

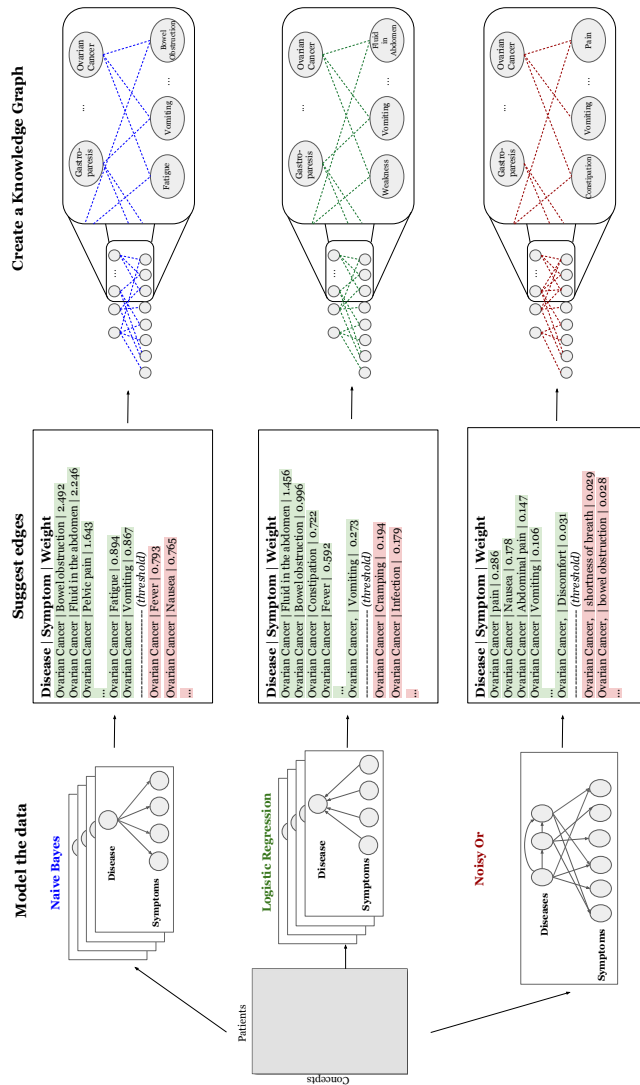


Figure 5.1: Workflow of modeling the relationship between diseases and symptoms and knowledge graph construction, for each of our 3 models (naive Bayes, logistic regression and noisy or).

Parameters for all three models were learned using maximum likelihood estimation. For logistic regression and naive Bayes, a model was learned separately for each disease. For noisy or, all the parameters were learned jointly. L1 regularization was used for logistic regression both to prevent overfitting and to encourage sparsity, desirable because we expect most diseases to cause a small number of symptoms (Razavian et al., 2015). Laplacian smoothing was used to prevent overfitting for naive Bayes. For both, the values of the hyper-parameters were chosen separately for each disease via a 3-fold cross-validation.

5.2.2 From statistical models to knowledge graphs

For each model, we construct an importance measure to determine whether an edge should be included between symptom and disease. The importance measures denote each model’s relative confidence that an edge exists between a pair of nodes. We then sort symptoms for each disease by the importance measure.

Logistic regression: The importance measure for logistic regression was taken to be:

$$\text{Impt}_{\text{LR}} = \max\{b_{i,j}, 0\}, \tag{5.1}$$

where $b_{i,j}$ is the weight associated with symptom j in the logistic regression model fit to predict disease i . In other words, if the appearance of a symptom made a disease more likely, then we believed that a corresponding edge exists in the graph. Note that in our setting it is sensible to compare weights between features since all of our variables are binary.

Naive Bayes: The importance measure for naive Bayes was taken to be:

$$\text{Impt}_{\text{NB}} = \log P(X_j = 1|Y_i = 1) - \log P(X_j = 1|Y_i = 0), \tag{5.2}$$

where X_j is the binary variable denoting the presence of symptom j and Y_i is the binary

variable denoting the presence of disease i . In words, if the appearance of disease makes the observation of symptom more likely, we have higher confidence that an edge exists.

For both naive Bayes and logistic regression, we enforced a minimum of 5 co-occurrences for any disease-symptom pair for any suggested edge as a de-noising measure.

Noisy or: The noisy or conditional distribution is parametrized by failure probabilities (for more information on noisy or conditional distributions, see Section 1.2.3). Let n be the number of diseases and m be the number of symptoms. Each disease Y_i that is present turns on each of its children symptoms indexed by X_j , but fails with probability $f_{i,j}$. A leak probability l_j represents the probability of a symptom being on even if all diseases are off. Thus the probability of a symptom being present is:

$$P(X_j = 1|y_1, \dots, y_n) = 1 - (1 - l_j) \prod_{i=1}^n f_{i,j}^{y_i}. \quad (5.3)$$

We took the importance measure to be:

$$\text{Impt}_{\text{NO}} = 1 - f_{i,j}. \quad (5.4)$$

The higher the failure probability, the less likely it is for the symptom to be a child of the disease. Importantly, by optimizing the conditional probability, we make no assumptions about the prior distribution of diseases $P(Y_1, \dots, Y_n)$.

5.2.3 A causal interpretation of the learned models

Pearl introduced the notion of the *do* conditioning operator to formalize causal reasoning. Unlike the standard conditioning operator, the *do* operator sets a variable to a particular value in a way that does not affect its causal parents (for more information, see Pearl (2009)). If we assume the graphical structure denoted in Figure 5.2, then the *do* operator can be

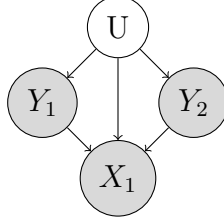


Figure 5.2: Causal graph of diseases and symptoms. Diseases (Y_1, Y_2) can cause symptoms (only one symptom, X_1 , shown here), and can have a high dimensional set of common parents U (risk factors such as age) which make the diseases non-independent. Diseases and symptoms are observed.

related to the conditioning operator according to Equation 5.5 (derived in Appendix E.1):

$$P(X_1 | \text{do}(Y_1 = y_1)) = E_{y_2 \sim P(Y_2)} P(X_1 | y_1, y_2) \quad (5.5)$$

The failure probability of noisy or (used in the importance measure, Equation 5.4) can be interpreted as a ratio of *do* operators. It tells us how the likelihood of a symptom being off changes as a disease is turned from on to off. Let Y_{-1} denote all of the diseases other than Y_1 .

$$\begin{aligned}
 \frac{P(X_j = 0 | \text{do}(Y_1 = 1))}{P(X_j = 0 | \text{do}(Y_1 = 0))} &= \frac{E_{Y_{-1}} P(X_j = 0 | Y_1 = 1, Y_{-1})}{E_{Y_{-1}} P(X_j = 0 | Y_1 = 0, Y_{-1})} \\
 &= \frac{E_{Y_{-1}} f_1 \prod_{i=2}^n f_{i,j}^{y_i}}{E_{Y_{-1}} 1 \prod_{i=2}^n f_{i,j}^{y_i}} \\
 &= \frac{f_1 E_{Y_{-1}} \prod_{i=2}^n f_{i,j}^{y_i}}{1 E_{Y_{-1}} \prod_{i=2}^n f_{i,j}^{y_i}} \\
 &= f_{1,j}.
 \end{aligned}$$

The first line comes from Equation 5.5, and the second line substitutes the noisy or parametrization. The third and fourth line are algebraic manipulations. The derivation assumes that data is consistent with a noisy or model and that the noisy-or model can be

consistently estimated by a maximum likelihood estimator (i.e., no unobserved diseases), but *crucially makes no assumptions about $P(Y)$* .

In contrast, the naive Bayes estimator can only be made to correspond to a *do* operator if all of the diseases are independent.

$$\begin{aligned}
\frac{P(X_j | \text{do}(Y_1 = 1))}{P(X_j | \text{do}(Y_1 = 0))} &= \frac{E_{y_{-1} \sim P(Y_{-1})} P(X_j | Y_1 = 1, y_{-1})}{E_{y_{-1} \sim P(Y_{-1})} P(X_j | Y_1 = 0, y_{-1})} \\
&= \frac{\sum_{y_{-1}} P(y_{-1}) P(X_j, Y_1 = 1, y_{-1}) / P(Y_1 = 1, y_{-1})}{\sum_{y_{-1}} P(y_{-1}) P(X_j, Y_1 = 0, y_{-1}) / P(Y_1 = 0, y_{-1})} \\
&= \frac{\sum_{y_{-1}} P(y_{-1}) P(X_j, Y_1 = 1, y_{-1}) / P(Y_1 = 1) P(y_{-1})}{\sum_{y_{-1}} P(y_{-1}) P(X_j, Y_1 = 0, y_{-1}) / P(Y_1 = 0) P(y_{-1})} \\
&= \frac{\sum_{y_{-1}} P(X_j, y_{-1} | Y_1 = 1)}{\sum_{y_{-1}} P(X_j, y_{-1} | Y_1 = 0)} \\
&= \frac{P(X_j | Y_1 = 1)}{P(X_j | Y_1 = 0)}
\end{aligned}$$

Where the first equality comes from Equation 5.5 and the second line expands out the conditioning operator and the expectation. The third line uses the independence assumption. The next lines follow algebraically. The result, $\frac{P(X_j | Y_1 = 1)}{P(X_j | Y_1 = 0)}$, is the value used in the importance measure for naive Bayes (Equation 5.2) using $X_j = 1$.

The objective of logistic regression does not relate to a *do* operation on a disease.

Even though the failure probability in a noisy or model is related to a causal quantity, it may not be the best quantity for characterizing the relationship between a disease and a symptom because it relates to the change in likelihood of a disease being *absent* due to an intervention (turning “on” a disease), and not the change in likelihood of a disease being *present*. The naive Bayes model works for either value of X_j , but makes the assumption that diseases are independent (as mentioned above).

5.3 Experimental methods

In this section, we describe a study undertaken to assess the ability of each of the models described in the previous section to extract meaningful causal relationships between diseases and symptoms from electronic medical records.

5.3.1 Study design

We conducted a retrospective observational study using previously collected data from electronic medical records to construct a knowledge graph relating symptoms to diseases. We evaluated our candidate knowledge graphs against a gold-standard knowledge graph provided by Google (Google health knowledge graph) and the expert opinion of physicians. The study was approved by the hospital’s institutional review board.

5.3.2 Setting and selection of participants

The study was performed using data from a 55 000-visit/year trauma center and tertiary academic teaching hospital. All consecutive emergency department (ED) patients between 2008 and 2013 were included. Each record represents a single patient visit. No patients were excluded, leading to a total of 273 174 records of emergency department patient visits.

5.3.3 Data collection and preparation

5.3.3.1 Concept extraction from electronic medical record

We extracted positive mentions of diseases and symptoms (concepts) from structured and unstructured data. Structured data consisted of ICD-9 (International Classification of Diseases) diagnosis codes. Unstructured data consisted of chief complaint, Triage Assessment, Nursing Notes, and MD comments. Triage Assessment refers to the free-text nursing

assessment documented at triage. Medical Doctor (MD) Comments and Nursing notes refer to the free-text scratch space used by physicians and nurses respectively to track a patient’s course. Free text fields were de-identified using PhysioNet’s deid software package (Neamatullah et al., 2008; Goldberger et al., 2000).

The set of diseases and symptoms considered were chosen from the Google health knowledge graph (described below) to establish a basis for later comparison. We used string-matching to search for concepts via their common names, aliases or acronyms, where aliases and acronyms were obtained both from the Google health knowledge graph as well as from the Unified Medical Language System (UMLS) for diseases where the mapping was known. Similarly, if a link to an ICD-9 code was provided, we searched for that code in the record’s structured administrative data. A modified version of NegEx was used to find negation scopes in the clinical text (Chapman et al., 2001; Jernite et al., 2013a). Mentions that occur within a negation scope were not counted. Figure 5.3 illustrates the data extraction and processing pipeline.

5.3.3.2 Google health knowledge graph

We used a subset of Google’s health knowledge graph as of August 2015 for which we had sufficient support in our data. We defined sufficient support for a disease as having at least 100 positive mentions and for a symptom as having at least 10 positive mentions. This resulted in 156 diseases and 491 symptoms. The graph is comprised of medical concepts (diseases and symptoms) as nodes and disease-symptom relations as edges.

A small number of concepts in the Google health knowledge graph are classified as both a disease and a symptom (e.g., ‘Type II diabetes’ is a disease, but also a symptom of ‘Polycystic Ovarian Cancer’). In these cases, we designated these concepts as diseases only.

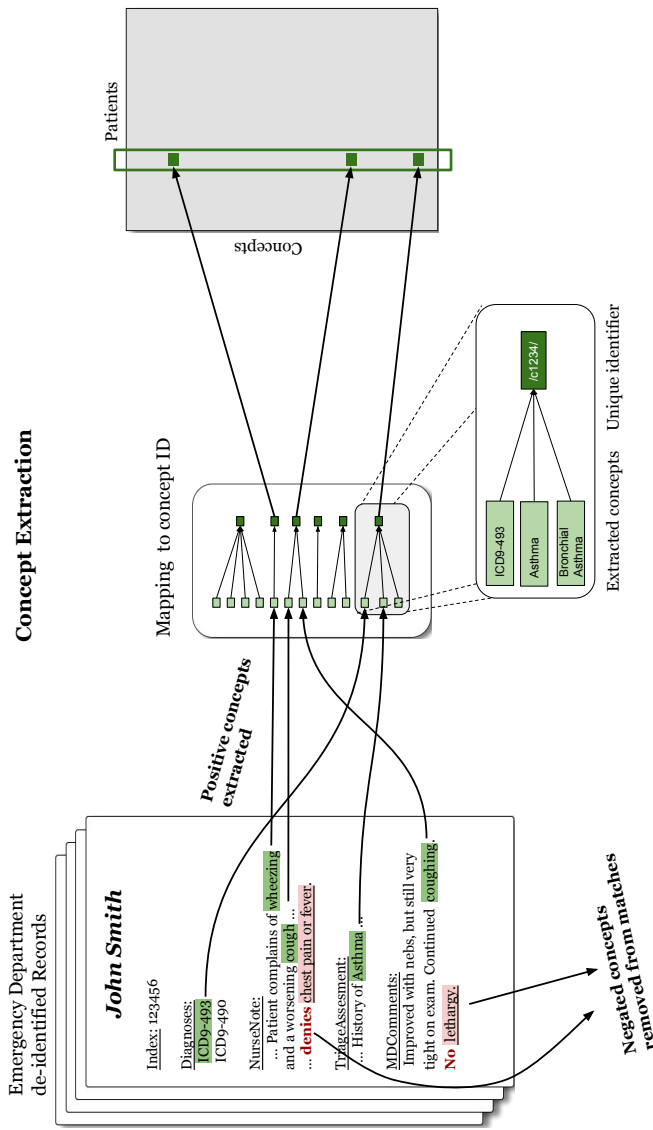


Figure 5.3: Concept extraction pipeline. Non-negated concepts and ICD-9 diagnosis codes are extracted from Emergency Department electronic medical records. Concepts, codes and concept aliases are mapped to unique IDs, which in turn populate a co-occurrence matrix of size (Concepts) x (Patients).

5.3.4 Evaluation and analysis

We compare our learned graphs against two different gold standards using precision-recall graphs to evaluate quality.

5.3.4.1 Automatic evaluation: Comparison to the Google health knowledge graph

In the automatic evaluation, we assess the performance of our models against a binary target (either the suggested edge is present in the Google health knowledge graph or it is not). While we believe the edges present in the knowledge graph are correct, we do not assume that they are exhaustive. Therefore, the automatic evaluation allows us to efficiently compare between the models considered, but likely underestimates precision, marking edges as false-positives even though they are actually correct, just missing from the Google graph. The symptom ‘pain’ was removed from this evaluation because it was overly general and inconsistently used in the Google graph.

5.3.4.2 Manual evaluation: Evaluation by physicians

To address the issue of the Google graph incompleteness, we also evaluate the graphs against physicians’ expert opinion. Given that the set of potential disease-symptom edges is large, it is impractical to ask evaluators to label all possible edges. Therefore, we use a procedure in which the top N results from each model are pooled together and rated by clinical evaluators. The edges outside the pooled results are considered irrelevant. This method, termed ‘pooling’, is frequently used in information retrieval settings (Jones, 1975). For a given disease, the top two models from the automated evaluation, as well as the industry-provided graph contributed to the pool of edges to be evaluated by the physicians. Each model suggests up to $K+10$ symptoms, where K is the number of symptoms present in

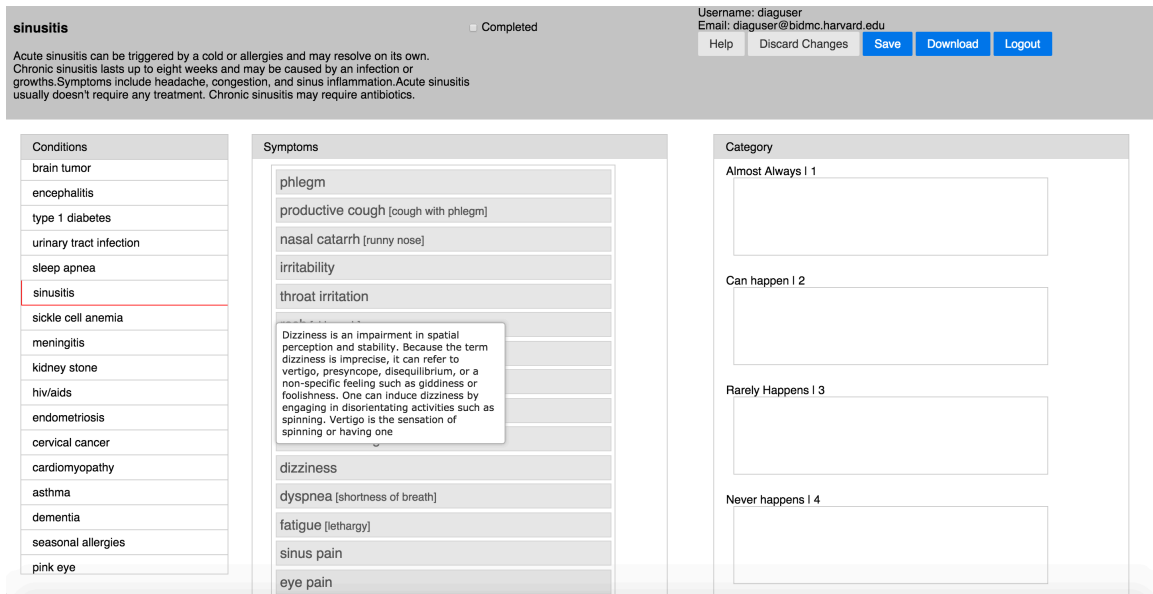


Figure 5.4: User interface for evaluation by physicians. The left column shows the list of diseases from which clinicians select a disease to be tagged. When a disease is selected, the upper panel provides a more detailed description of the selected disease. The middle column shows the symptoms that have yet to be tagged and hover over functionality provides a detailed symptom definition. The rightmost column shows the 4 categories into which symptoms can be sorted.

the knowledge graph for that disease. The 10 additional symptoms are introduced to allow for the possibility of the true graph being denser than the Google health knowledge graph.

To evaluate the graphs, physicians rated the suggested edges according to the statement “disease A causes symptom B” using the 4-point scale: ‘always happens’, ‘sometimes happens’, ‘rarely happens’ or ‘never happens’. A user interface was built to facilitate easy rating (Figure 5.4). For the evaluation itself, physician responses were binarized by grouping results from the ‘always’, ‘sometimes’ and ‘rarely’ categories into the positive category, leaving the ‘never’ tag to be negative.

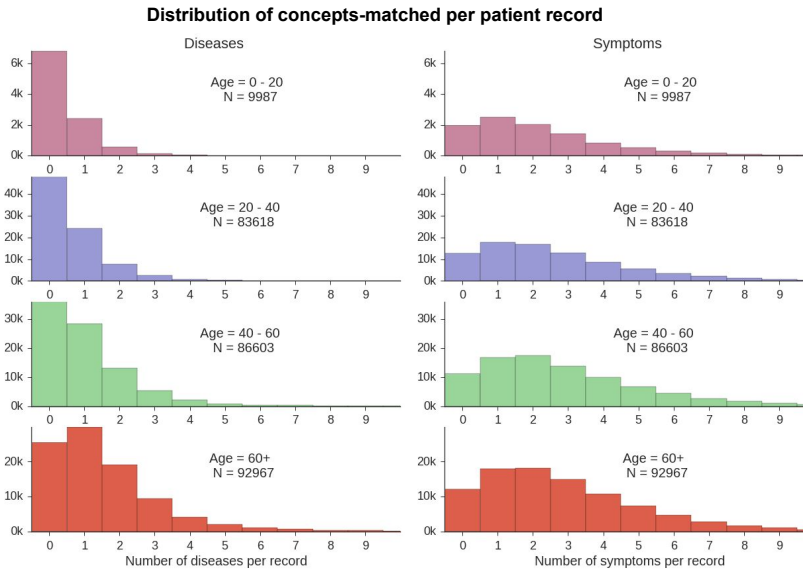


Figure 5.5: Distribution of number of diseases and symptoms per patient record.

5.3.4.3 Statistical methodology

While the pooling method is overly optimistic in its evaluation of recall, it provides a reliable measure of precision and the set of omitted elements does not unjustly bias one method over another. To determine whether the differences in model precision were statistically significant, we use a Wilcoxon signed rank test (Zobel, 1998).

To determine inter-rater agreement, 15 diseases were rated by a second physician. We use the Spearman rho correlation to measure inter-rater agreement and calculate confidence intervals using bootstrapping.

5.4 Results

Figure 5.5 shows the distribution of the number of identified diseases and symptoms across medical records. We observe that the distributions are positively skewed across age groups and that a substantial fraction of older patients (40+) have two or more diseases identified per record.

After concepts were extracted, we constructed a knowledge graph from each model by either choosing an importance threshold or by allowing each model to suggest a constant number of edges per diseases. Table 5.1 shows the top symptoms suggested by each model for Middle Ear Infection. The table shows that logistic regression is not well calibrated for the task of constructing a knowledge graph, suggesting three irrelevant symptoms in its top ten suggestions. In this example, noisy or and naive Bayes both perform similarly well. Table 5.2 shows the top symptoms for Gallstones.

The same trend is observed when evaluating precision-recall graphs across all available diseases. Figure 5.6(a) shows the Precision-Recall curve resulting from the automatic evaluation. Here too logistic regression falls short in performance. Figure 5.6(b) shows the Precision-Recall results for the clinical evaluation. The computed inter-rater agreement measure (mean = 0.7448, std = 0.0297) shows considerable agreement between evaluators. Additionally, we observe that both noisy or and naive Bayes have lower recall and higher precision in the clinical evaluation than suggested by the automatic evaluation. The lower recall exhibited by our models, coupled with the observation that both models surpass the recall of the Google graph in the clinical evaluation, suggests that the Google graph is not exhaustive.

The same trend is observed when evaluating precision-recall graphs across all available diseases. Figure 5.6(a) shows the Precision-Recall curve resulting from the automatic evaluation. Here too logistic regression falls short in performance. For instance, for a recall of 0.5, noisy or, naive Bayes and logistic regression achieve a precision of 0.23, 0.18 and 0.13, respectively. Figure 4(b) shows the Precision-Recall results for the clinical evaluation. The computed inter-rater agreement measure (mean = 0.74, std = 0.03) shows considerable agreement between evaluators. Additionally, we observe that both noisy or and naive Bayes have lower recall and higher precision in the clinical evaluation than suggested by the automatic evaluation. For a recall of 0.5, noisy or and naive Bayes achieve a precision of

Top edge suggestions for 'Middle Ear Infection'									
Ranking (importance score)	Logistic regression model		Naive Bayes model		Noisy or model	Frequency (Google health knowledge graph buckets)	Google health knowledge graph		
1	Ear pain	***	Inflammation of ear	***	Ear pain	***	<i>Always</i>	Inflammation of ear	***
2	Teeth chattering		Ear pain	***	Inflammation of ear	***	<i>Frequent</i>	Ringing in the ears	**
3	Red face	*	Exudate	***	Sore throat	**	<i>Frequent</i>	Headache	**
4	Inflammation of ear	***	Ache	***	Coughing	*	<i>Frequent</i>	Nausea	*
5	Itchy eyes	**	Nasal congestion	*	Fever	**	<i>Frequent</i>	Crying	**
6	Irritability	**	Sore throat	**	Nasal congestion	*	<i>Frequent</i>	Fever	**
7	Anger	*	Runny nose	*	Pain	***	<i>Frequent</i>	Nasal congestion	*
8	Red rashes		Coughing	*	Ache	***	<i>Frequent</i>	Ear pain	***
9	Sleepiness	**	Sensitivity to light	*	Chills	**	<i>Frequent</i>	Loss of appetite	**
10	Facial paralysis		Fever	**	Headache	**	<i>Frequent</i>	Vertigo	*

Table 5.1: Top edge suggestions by models for a randomly chosen disease (Middle Ear Infection). The number of shown edges corresponds to the number of edges in the Google health knowledge graph. For logistic regression, naive Bayes, and noisy or, the suggestions are ordered by their relative importance score. For the Google health knowledge graph, the edges are sorted according to two broad buckets of edge frequency that are provided in the graph. The stars associated with each edge represent the expected frequency for which “disease A causes symptom B” as rated by physicians. [*** = always happens, ** = sometimes happens, * = rarely happens, = never happens]

Top edge suggestions for 'Gallstones'							
Ranking (importance score)	Logistic regression Model	Naive Bayes Model	Noisy or Model	Frequency (Google health knowledge graph buckets)	Google health knowledge graph		
1	Abdominal cramping from Gallstones ***	Abdominal cramping from Gallstones ***	Pain ***	<i>Frequent</i>	Back pain **		
2	Pain in upper-right abdomen ***	Pain in upper-right abdomen ***	Nausea ***	<i>Frequent</i>	Pain between shoulder blades		
3	Yellow skin and eyes **	Upper abdominal pain ***	Abdominal pain ***	<i>Frequent</i>	Severe pain ***		
4	Pain ***	Dark urine *	Pain in upper abdomen ***	<i>Frequent</i>	Mild pain **		
5	Pain in upper abdomen ***	Yellow skin and eyes **	Vomiting ***	<i>Frequent</i>	Night pain		
6	Dark urine *	Pain in upper abdomen ***	Chills *	<i>Frequent</i>	Abdominal discomfort ***		
7	Upper abdominal pain ***	Intermittent abdominal pain ***	Tenderness ***	<i>Frequent</i>	Nausea ***		
8	Dry skin	Belching	Abdominal cramping from Gallstones ***	<i>Frequent</i>	Side pain *		
9	Sleepiness *	Discomfort in upper abdomen ***	Yellow skin and eyes **	<i>Frequent</i>	Pain in upper-right abdomen ***		
10	Abdominal pain ***	Abdominal pain ***	Pain in upper-right abdomen ***	<i>Frequent</i>	Flatulence		
11	Restless legs syndrome	Intermittent pain ***	Diarrhea *	<i>Frequent</i>	Indigestion *		
12	Side pain *	Swollen veins in the lower esophagus	Fever **	<i>Frequent</i>	Vomiting ***		
13	Regurgitation	Fluid in the abdomen	Flank pain *	<i>Frequent</i>	Abdominal cramping from Gallstones ***		

Table 5.2: Top edge suggestions by models for the disease “Gallstones”. The number of shown edges corresponds to the number of edges in the Google health knowledge graph. For logistic regression, naive Bayes, and noisy or, the suggestions are ordered by their relative importance score. For the Google health knowledge graph, the edges are sorted according to two broad buckets of edge frequency that are provided in the graph. The stars associated with each edge represent the expected frequency for which “disease A causes symptom B” as rated by physicians. [*** = always happens, ** = sometimes happens, * = rarely happens, = never happens]

0.85 and 0.8, respectively. The lower recall exhibited by our models (Figure 5.6(b)), coupled with the observation that both models surpass the recall of the Google graph in the clinical evaluation, suggests that the Google graph may be incomplete.

In both evaluation frameworks presented in Figure 5.6, noisy or outperforms naive Bayes. The Wilcoxon signed rank test determined that the differences in precision were statistically significant for both evaluation frameworks ($p \leq 0.01$).

Table 5.3 shows a subset of the knowledge graph constructed by noisy or, our best performing model. The number of edges included was chosen to match the number presented to clinical evaluators.

5.5 Discussion

5.5.1 Differences between the evaluation methods

There were a number of differences between the edges suggested by the learned models and marked correct by the clinical evaluators and those contained in the Google health knowledge graph. The Google health knowledge graph was designed to provide useful information to web-users, which explains some of the differences between it and the emergency department setting where the data was collected.

5.5.1.1 Google health knowledge graph not exhaustive

The Google health knowledge graph omits ‘Exudate’, ‘Ache’ and ‘Sore throat’ for the disease ‘Middle Ear Infection’, despite them being labeled as relevant by both clinical evaluators (Table 5.1). Similarly, the symptoms ‘Tenderness’ and ‘Intermittent pain’ are not listed in the Google graph’s symptoms for the disease ‘Gallstones’ (Table 5.2). These symptoms were suggested by our learning algorithms, illustrating the potential for an EMR

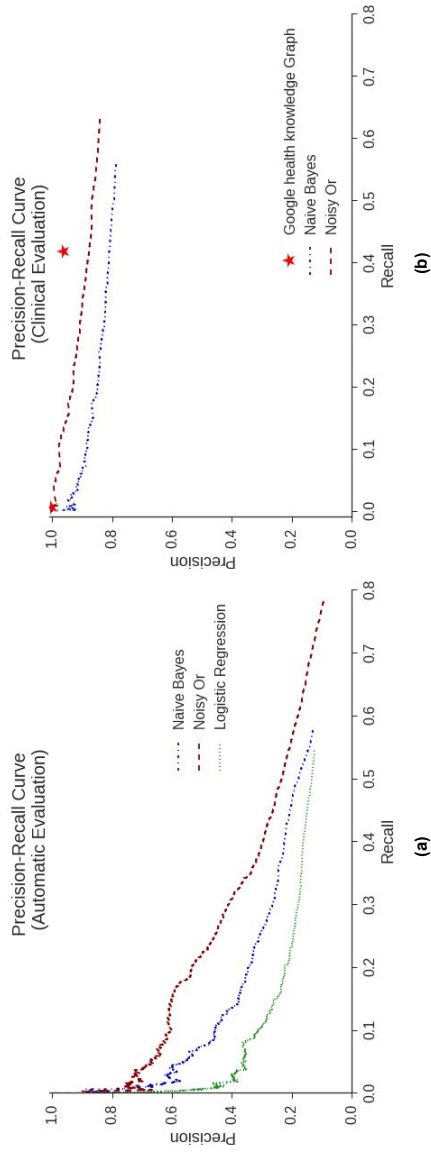


Figure 5.6: Precision-recall curves for two evaluation frameworks. (a) The precision-recall curve for the automatic evaluation evaluated against the Google health knowledge graph. (b) The precision-recall curve rated according to physicians expert opinion. We see that in both graphs, the relative performance of the models is the same.

Examples of Edge Suggestions for Noisy or	
Diseases	Suggested edges
Aphasia	problems with coordination (0.318), weakness (0.181), confusion (0.106), mental confusion (0.088), slurred speech (0.074), numbness (0.071), headache (0.049), seizures (0.045), weakness of one side of the body (0.042), difficulty speaking (0.034), blurred vision (0.018), malnutrition (0.017)
Appendicitis	pain (0.881), nausea (0.401), abdominal pain (0.361), tenderness (0.163), chills (0.152), diarrhea (0.124), vomiting (0.118), fever (0.096), loss of appetite (0.068), lower abdominal pain (0.040), cramping (0.037), constipation (0.036), discomfort (0.033), cyst (0.030), pain in right lower abdomen (0.029), sharp pain (0.023), pain during urination (0.022), pain in upper abdomen (0.020), pelvic pain (0.017), flank pain (0.016), vaginal discharge (0.013), abdominal discomfort (0.013), dull pain (0.012), infection (0.011)
Bed bug bite	skin rash (0.329), itching (0.173), anxiety (0.048), infection (0.029), sadness (0.026), depression (0.026), red spots (0.018), skin irritation (0.018), sweating (0.016), eye pain (0.015), lesion (0.012), substance abuse (0.011), hallucination (0.009), swollen feet (0.009), skin lesion (0.009), brief visual or sensory abnormality (0.009)
Bell's palsy	numbness (0.308), weakness (0.198), headache (0.134), facial paralysis (0.071), ear pain (0.052), slurred speech (0.051), paralysis (0.046), facial pain (0.040), neck pain (0.038), facial swelling (0.037), tongue numbness (0.031), asymmetry (0.026), blurred vision (0.024), drooping of upper eyelid (0.020), lesion (0.019), malnutrition (0.019), difficulty swallowing (0.018), double vision (0.016)
Carpal tunnel syndrome	numbness (0.175), pain (0.167), hand pain (0.094), weakness (0.083), arm pain (0.071), wrist pain (0.060), swelling (0.054), hand numbness (0.041), redness (0.030), pins and needles (0.024), shoulder pain (0.024), vertigo (0.020), hand swelling (0.016), neck pain (0.016), infection (0.014), depression (0.011), sadness (0.011), anxiety (0.011), chronic back pain (0.010), back pain (0.010), malnutrition (0.010), severe pain (0.008), unsteadiness (0.008), dry skin (0.008)
Ectopic pregnancy	pain (0.537), bleeding (0.204), vaginal bleeding (0.181), abdominal pain (0.167), cramping (0.155), spotting (0.154), nausea (0.104), cyst (0.067), tenderness (0.055), lower abdominal pain (0.048), pelvic pain (0.040), diarrhea (0.031), vaginal discharge (0.023), discomfort (0.020), vomiting (0.016), back pain (0.015), vaginal pain (0.014), lightheadedness (0.011)
Kidney stone	pain (0.608), flank pain (0.495), nausea (0.232), blood in urine (0.141), pain during urination (0.084), vomiting (0.083), chills (0.067), abdominal pain (0.065), back pain (0.050), tenderness (0.040), discomfort (0.019), groin pain (0.018), severe pain (0.013), fever (0.012), testicle pain (0.011), frequent urge to urinate (0.011), lower abdominal pain (0.011), dark urine (0.011), urinary retention (0.011), sharp pain (0.010), cyst (0.010), pain in lower abdomen (0.010), diarrhea (0.009), constipation (0.008), infection (0.007), pelvic pain (0.007), side pain (0.004), dull pain (0.004)
Retinal detachment	vision loss (0.125), blurred vision (0.065), headache (0.057), neck pain (0.041), eye pain (0.039), dehydration (0.024), difficulty walking (0.023), itching (0.020), discomfort (0.018), unequal pupils (0.017), watery diarrhea (0.015), bone loss (0.015), partial loss of vision (0.014), ear pain (0.013), fast heart rate (0.012), slow bodily movement (0.009), low oxygen in the body (0.009), vision disorder (0.009), elevated alkaline phosphatase (0.009), seeing spots (0.009), abnormality walking (0.009), malnutrition (0.009)

Table 5.3: Subset of the knowledge graph learned using the noisy or model. For each disease we show the full list of edges along with their corresponding importance score in parentheses. Symptoms are ordered according to importance scores.

data-driven approach to uncover relevant symptoms.

5.5.1.2 Patient appropriate language

Another class of differences between the Google health knowledge graph and the physician evaluation involves use of precise language. For example, the Google’s graph contains an edge from ‘Gallstones’ to ‘Pain between shoulder blades’. While this is technically not the precise location of gallstone pain, it is a description that a patient may use.

5.5.1.3 Severity Misalignment

A third class of differences between the Google graph and those learned by our models and approved by the clinical evaluators are edges that express a heightened severity. For instance, for the disease ‘Gallstones’ (Table 5.2), the clinical collaborators approved ‘Abdominal Pain’, while the Google graph contains ‘Abdominal Discomfort’. Similarly, our models suggest ‘Diarrhea’ in place of the more mild ‘Indigestion’.

Our model’s selection of more severe presentations of edges suggests that the graph is organically tailored towards the emergency department data setting.

Another indication of the differences in severity between the two settings appears in the expected frequency of diseases, as listed in the Google health knowledge graph for the ‘adult’ age bracket, compared to the observed count of diseases for that age bracket as found in our data (Figure 5.7). Both ‘Multiple Sclerosis’ and ‘Crohn’s Disease’ appear very frequently in the emergency department data even though they are listed as ‘Rare’ in the Google health knowledge graph. Conversely, ‘Vaginitis’, ‘Plantar Wart’ and ‘Nail Fungus’ appear very infrequently in the emergency department data, even though they are as listed as ‘Very Frequent’ according to the Google health knowledge graph. This selection bias towards higher acuity conditions and presentations leads to structural differences between our constructed graphs and the Google health knowledge graph, and suggests that there may not be a single

graph that generalizes across all settings. Our methodology provides a way of automatically adapting a knowledge graph by training on patient records from the relevant setting.

5.5.2 Analysis of model errors

We look the most common symptoms that are wrongly suggested by each model to determine if there are certain characteristic errors that the learning algorithms make.

5.5.2.1 Noisy or: general symptoms

The noisy or model tends to rank general symptoms highly, such as ‘Pain’, ‘Weakness’, ‘Lethargy’, ‘Sadness’ and ‘Infection’. For example, in Table 5.2 we see that for disease ‘Gallstones’, noisy or suggests ‘Pain’, ‘Nausea’ and ‘Abdominal Pain’ before the more specific symptom ‘Abdominal Cramping from Gallstones’. While these edge suggestions are not necessarily incorrect, they are substantially less informative than their more specific counterparts. This trend is not shared by the naive Bayes and logistic regression models.

5.5.2.2 Naive Bayes & logistic regression: Correlated symptoms

Both naive Bayes and logistic regression wrongly suggest ‘Bone Loss’, ‘Lethargy’ and ‘Confusion’ as likely symptoms for diseases that are common in elderly patients. For example, ‘Bone Loss’ is weighted highly for diseases such as ‘Shingles’ and ‘Hiatal Hernia’. ‘Lethargy’ is weighted highly for ‘Thyroid cancer’, ‘Myasthenia Gravis’ and ‘Neutropenia’. These incorrect edges are likely being suggested because old age is a confounding factor that correlates a number of disease with each other. The problem of disambiguating correlation and causation is partly avoided using the noisy or model, which has a causal interpretation (Section 5.2.3). This is particularly relevant since many patients have multiple diseases simultaneously (Figure 5.5).

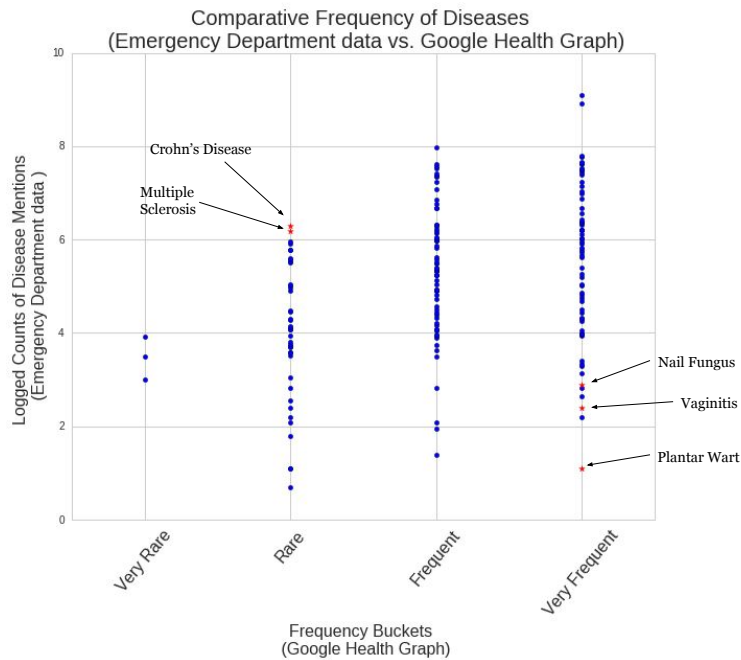


Figure 5.7: Comparison of disease frequency for the ‘adult’ age bracket (40-60 years old). The y-axis shows the number of identified diseases in the emergency department data. The x-axis records the expected frequency of diseases according to the Google health knowledge graph for the ‘adult’ age bracket. The points highlighted demonstrate instances of frequency misalignment due to the differences in populations considered.

5.6 Conclusions

We find that it is possible to construct a high quality health knowledge graph directly from electronic medical records. The high precision displayed by the noisy or model with a precision of 0.85 for the max recall of 0.6 suggests that a two-step process would be well suited for the construction of the knowledge graph, in which a clinician reviews and rejects some of the edges suggested by the model. Using the results of the clinical evaluation, we can infer that if a filtering step were added to the pipeline, for a recall of 60%, physicians would have to discard fewer than 2 out of 10 suggested edges. This “clean up” phase is also used in other approaches for constructing knowledge bases; since text mining and natural language processing are typically imperfect (Ford et al., 2016; Ramaswami, 2016; Ferrucci and Brown, 2011).

This method of automatically constructing knowledge graphs allows us to create graphs from EMRs in any number of domains quickly and without any prior knowledge. We believe that the most promising avenues for future research include incorporating more elaborate concept extraction algorithms into our pipeline and experimenting with other methods of measuring causal effects that do not assume a parametric form.

In addition to creating new knowledge graphs, such automated algorithms can be used to augment and maintain existing knowledge graphs. For example, they can be run regularly on current EMR data with existing knowledge graphs to suggest new edges over time that were not previously known. They can also be used to calibrate a knowledge base created for one setting to an entirely different setting.

5.7 Next steps and open questions

5.7.1 Improved concept extraction

The precision and recall obtained by our constructed knowledge graphs show that reasonable results can be obtained even with a rudimentary concept extraction pipeline. Nonetheless, because of the simplicity of the pipeline, at times we do not have coverage for concepts, despite them being present in the emergency department data. 34% of the symptoms associated with the subset of diseases that came from the Google health knowledge graph did not reach the required threshold of 10 positive mentions and were dropped due to insufficient support. One example is the symptom ‘Bull’s Eye Rash’ for disease ‘Lyme Disease’. Because of the varying ways in which the symptom is recorded and punctuated (for example: “bullseye”, “bullseye rash”, “bull eye”, “bull’s eye”, etc.), we record it fewer than 10 times. A more elaborate concept extraction pipeline would increase our coverage and improve the subsequent graph.

While our pipeline does not require any prior knowledge of the target area of application, it does require a base set of concepts to evaluate as potential nodes in the graph. For evaluation purposes, we used the concepts from the Google health knowledge graph. For alternate uses, any set of concepts, such as UMLS, would be appropriate. Nonetheless, it is important to recognize that our task was made simpler by working with a set of concepts clearly delineated into ‘diseases’ and ‘symptoms’ and for which every symptom is relevant to at least one disease.

5.7.2 Relaxing modeling assumptions

Neither noisy or nor our baseline models allow for edges to exist between the symptom nodes. It may be reasonable to allow symptoms to cause other symptoms as well, creating a softer classification into symptoms and diseases. An example where a softer classification

might be useful is in the previously mentioned ‘Type II diabetes’, which is a symptom of ‘Polycystic Ovarian Cancer’, but may itself cause other symptoms such as ‘Fatigue’ or ‘Frequent Urination’. Future research might benefit from investigating models that do not assume symptom conditional independence in order to capture this complexity.

All the models we have applied to the problem of knowledge graph construction are parametric and therefore restricted by their parametric form (e.g., noisy or conditional distributions). It might be useful to look into models that are not constrained by this form, particularly in order to have a closer match with the causal interpretation presented in Section 5.2.3.

5.7.3 Extending/refining existing graphs

In Section 5.5.1, we note some of the differences between the acute care and general web-search settings. Does having a good model of one setting make it easier to learn in the other setting? So far, we have only used the Google health knowledge graph for evaluation, but it could also be used as a seed for learning a model for the acute care setting. We could use it as a prior on the graph structure, to help learn the relevant relationships for diseases and symptoms that are rare in our data.

Alternatively, if seeded with an incomplete graph from experts, we could try to fill in the relevant edges using matrix completion methods (Koren et al., 2009) using the medical records as side information.

5.7.4 Combining declarative and statistical knowledge sources

How can we fruitfully combine information from textbooks, journals and web-content with data from electronic medical records or web search logs? One direction would be to use the trusted data sources to establish the *structure* of the model and then use the data to

learn parameters. Building a shared vocabulary of concepts to map between how concepts are expressed in textbooks and in medical records would be a first important step, and existing medical vocabularies would be useful for this, but are not complete. Another approach would be to jointly learn models for both knowledge sources with a shared latent space (Tucker, 1958).

5.7.5 Applying the learned models for inference

In this work, we only evaluate the *structure* of the derived knowledge base. A natural next step would be to evaluate the learned knowledge base for symptom checking (Semigran et al., 2015), clinical decision support, or information retrieval. For structure learning, a surprising result of this work was that treating the noisy concept extractions as ground truth label extractions, rather than introducing latent variables for the true underlying condition of the patient, was good enough to learn reasonable models. For the inference task, we have seen in Chapter 3 that explicitly modeling anchors, rather than treating them naively as true labels does lead to improved performance on the held out tag inference task (Table 3.3). Determining when it is worthwhile to introduce latent variables, with all the difficulty in optimization involved, and when it is better to ignore the noise in the labels, is still an open question.

Do causal models make more sensible errors in diagnosis? One of the drawbacks of the statistical models discussed in Section 2.5.5 was the possibility of mistakes that are very different from those that a human would make. This relates broadly to the question of interpretability in machine learning models (Lipton, 2016) and formulating performance metrics that go beyond measuring prediction accuracy. Comparing the *mistakes* made by a model that explicitly models the causal relations between diseases and symptoms to more general prediction models could be a useful step towards building safer and more trustworthy

machine learning models for healthcare.

5.7.6 Beyond diseases and symptoms

We studied symptoms and diseases whose causal relations are fairly well understood. A possible extension would be to use the noisy or model to find side effects of medications or procedures. In some cases, it may be possible to separate the symptoms of the disease itself and effects of the treatment temporally, but it may not always be possible, especially for chronic, progressive conditions. Other methods, such as comparing similar patients who have opted for different treatment paths, would be required to separate those effects.

A similar modeling framework could be more broadly useful outside of medicine, for example, modeling mechanical failures in aviation, automobile or industry settings from case reports.

Chapter 6

Correlated factors in anchored factor analysis

Acknowledgments This work was joint with Steve Horng and David Sontag.

6.1 Introduction

The factor analysis models considered in Chapters 3 and 4 make the simplifying assumption that the factors were marginally independent. Without further assumptions, allowing dependencies between the factors naturally introduces non-identifiabilities into the modeling process, making it difficult to interpret the learned parameters. In this chapter, we show that the anchor assumption is sufficient to learn factor analysis models with arbitrarily complex prior distributions of the factors.

Much of this work depends on material previously presented in Chapter 3 with adjustments to allow for dependencies between the factors. We will link to the material there, and reproduce it as necessary for completeness in this chapter.

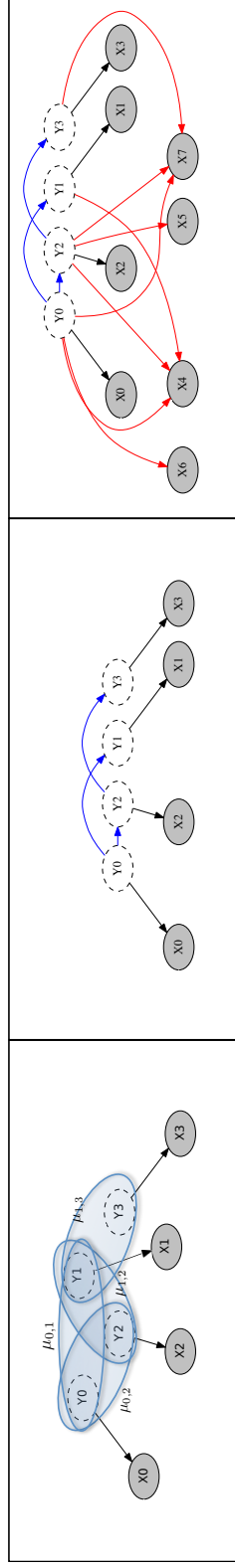
Estimating models involving latent variables is computationally challenging (See Sec-

tion 1.2.4). However, another source of computational complexity arises when we model the distribution of the factors themselves with a Bayesian network. Even with fully-observed data, structure learning in Bayesian networks is NP-hard (Chickering, 1996). Unlike Chapter 4, where we were able to give a provable algorithm with polynomial run time, in this chapter we reduce the problem of learning the structure of the latent variables to that of structure learning with fully observed data, which, despite being NP-hard in the worst case, is a well-studied problem for which practical algorithms exist for many real-world problems.

The algorithm consists of three main parts: First, we use anchors to recover moments involving the latent variables. Then, we use the recovered moments to learn a Bayesian network to describe the distribution of the latent variables. Finally, we learn conditional distributions of the observations given the latent variables. The algorithm is graphically displayed in Figure 6.1.

6.1.1 Contributions

1. We present an algorithm for recovery of moments involving latent variables using anchors, and derive a method-of-moments algorithm for noisy-or factor analysis using the recovered moments.
2. The presented algorithm is a constructive proof of identifiability, even for models with arbitrarily complex latent distributions. When the latent distribution is a tree-structured Bayesian network, we show how to learn the model in polynomial time.
3. We experimentally validate the method of moments algorithm against relevant baselines (including the independent classifiers described in Chapter 2) on a clinically relevant inference task, and demonstrate that modeling the correlations between phenotypes improves predictions.



Recover low-order marginals

Recover latent structure

Recover factor loadings

Figure 6.1: Illustration of our algorithm for anchored factor analysis with correlated factors. Observed variables X_0, \dots, X_3 are anchors for latent variables Y_0, \dots, Y_3 . First, we recover the low-order moments of the latent variables by solving an optimization problem involving moments of the observed anchors (Section 6.3). Second, we use these moments to learn a Bayesian network describing the joint distribution of the latent variables (Section 6.4.1). Finally, we learn the conditional distributions for the rest of the observations (Section 6.4.2).

4. We show that the modeling procedure can be applied beyond the healthcare setting by applying it to modeling tags in a collection of user-contributed programming questions (stack overflow dataset). A simple heuristic is sufficient to specify reasonable anchors in this setting.
5. We develop a robust version of the moment recovery algorithm, based on constrained optimization using marginal polytope constraints or relaxations thereof. We show empirically that the robust recovery procedure is more effective at recovering the distribution of the tags in the stack overflow dataset.

6.2 Model definition: Anchored factor analysis

We study the *factor analysis* setting (see Section 1.2.2), where factors and observations are modeled as a Bayesian network, with directed edges from the factors to the observations, and potentially between factors as well. In this section, we describe the model under consideration. The following two sections present a method for structure and parameter learning for this model.

We use a model similar to that used in Chapter 3. The Bayesian network consists entirely of binary random variables, which are partitioned into factors (Y_1, \dots, Y_n) and observations (X_1, \dots, X_m) .

Unlike the presentation in Chapter 3, the factors are *not* assumed to be marginally independent with individual prior probabilities. Instead, we parametrize them with an arbitrary Bayesian network $\mathcal{G}(Y)$:

$$P(Y) = \prod_{i=1}^n P(y_i | Pa(Y_i; \mathcal{G}(Y))), \quad (6.1)$$

where $Pa(Y_i; \mathcal{G}(Y))$ refers to the parents of Y_i in the graph $\mathcal{G}(Y)$. The complexity of

$P(Y)$ can be controlled by constraining the complexity (e.g., maximum in degree) of $\mathcal{G}(Y)$.

The probabilities of the observations conditioned on the factors are parametrized with a “noisy-or” distribution (Equation 6.2) (Shwe et al., 1991; Pearl, 1988):

$$P(X_j = 0|Y = \{y_1, \dots, y_m\}) = (1 - l_j) \prod_{i=1}^m f_{i,j}^{y_i}, \quad (6.2)$$

where the parameters $f_{i,j}$ are referred to as *failure* probabilities and l_j is the *leak* probability (see Section 1.2.3).

The model is *anchored* in that every factor Y_i , has an associated anchor observation $A_i \in X$. As in Section 3.2.1, we assume the anchors depend only on their respective parent factors (structurally, anchors only have a single parent). When conditioning on its parent, an anchor becomes independent of all other observations (by d-separation). Additionally, we assume that the edge between a latent variable and its anchor is not vacuous. Particularly, A_i is not independent of Y_i . The right panel of Figure 6.1 shows an example model. X_0, \dots, X_3 are anchors for the latent variables Y_0, \dots, Y_3 .

6.3 Recovering moments of latent variables

In this section, we present a method of recovering moments of latent variables using moments of anchors, which are fully observed. The presentation here is similar to that in Section 3.4.2, though slightly more general to allow for the more general setting in this chapter.

6.3.1 Naive moment recovery

An anchor is independent of all other observed variables when conditioning on its parent (Section 6.2). This implies a particular relationship between the moments involving latent

variables and the moments involving their anchors. For a set of latent variables, $Z \subset Y$, and their associated anchors, A_Z , the relationship described in Equation 6.3 holds:

$$P(A_Z) = \sum_z P(Z = z) \prod_{i=1}^{|Z|} P(A_i|z_i). \quad (6.3)$$

The left-hand side of this equation is a quantity that only involves observed variables and can be estimated from empirical counts. The right-hand side uses the conditional distributions of the anchors, $P(A_i|Z_i)$, and the moments of the latent variables, $P(Z = z)$. When the conditional distributions of the anchors are known (discussed in Section 8.1.1), we can solve the following matrix equation:

$$\mu_{A_Z} = R_Z \mu_Z, \quad (6.4)$$

where μ_{A_Z} is a vectorized form of $P(A_Z)$, with $2^{|Z|}$ entries, one for each setting of A_Z in $\{0, 1\}^{|Z|}$. R_Z is a $2^{|Z|} \times 2^{|Z|}$ matrix encoding the conditional distributions of the anchors (Appendix F.1 describes the construction of the R_Z matrix) and μ_Z is the vectorized form of $P(Z)$.

We could simply invert the noise matrix R_Z to solve for μ_Z as in Equation 6.5 (R_Z is invertible, see Appendix F.1), however without infinite data, it would not be guaranteed that the solution for μ_Z would be a valid probability distribution (i.e., non-negative and sum-to-one).

$$\mu_Z = R_Z^{-1} \mu_{A_Z} \quad (6.5)$$

Instead, we explicitly solve the constrained optimization over the simplex to minimize a divergence measure between a proposed distribution $\vec{P}(X_j|Y_i)$ and the denoised version of the empirical counts $P(X_j|A_i)$ (Equation 6.6):

$$\mu_Z = \operatorname{argmin}_{\vec{p} \in \Delta} D(\mu_{A_Z} || R_Z \vec{p}). \quad (6.6)$$

Chaganty and Liang (2014) show the consistency of this estimator using both L2 distance and KL divergence for D . The consistency of the estimator means that the marginals recovered from empirical estimates, $\hat{\mu}_Z$ will converge to the true values, μ_Z , assuming that the anchor assumption is correct.

6.3.2 Robust moment recovery

The method for moment recovery, described above, recovers each marginal distribution (i.e., for each set of variables Z) independently. When learning models, we often want to recover many marginal distributions for use within a learning algorithm. Consider, for example, the case of learning a tree-structured distribution on the latent variables. The Chow-Liu algorithm (Chow and Liu, 1968) is an efficient algorithm for maximum likelihood learning in tree structured Bayesian networks, which requires pairwise marginals for every pair of variables Y_i, Y_j as input. We could solve $\binom{n}{2}$ independent optimization problems of the form given in Eq. 6.6, resulting in a set of estimates, $\hat{\mu}_{ij}$, for input to the algorithm. However, we would not be taking full advantage of the joint nature of the problem.

Our key insight is that the true pairwise marginal vector, $\mu \equiv \{\mu_{ij} : i, j \in Y\}$, consists of marginalizations of a single distribution, $P(Y)$. As such, there are additional constraints that it must satisfy. For example, using the notation $\mu_{ij}(y_i, y_j)$ to mean the marginal distribution of Y_i, Y_j evaluated at the setting (y_i, y_j) , the *local consistency* constraints must hold:

$$\sum_{y_i} \mu_{ij}(y_i, y_j) = \sum_{y_k} \mu_{jk}(y_j, y_k) \quad \forall i, j, k \in Y \text{ and } y_j. \quad (6.7)$$

More generally, μ must lie in the *marginal polytope*, \mathcal{M} , consisting of the space of all possible marginal distribution vectors that can arise from any distribution (Wainwright and Jordan, 2008). The local consistency constraints form an outer bound to the marginal polytope (called the local consistency polytope) since there exist vectors μ which satisfy the local consistency constraints and are not in the marginal polytope, but not vice versa.

Optimizing over the marginal polytope is NP-hard, but its outer bounds have been studied extensively in the context of inference in probabilistic graphical models. Maximum a posteriori (MAP) inference corresponds to optimizing a linear objective over the marginal polytope, and computing the log-partition function can be shown to be equivalent to optimizing a non-linear objective over the marginal polytope (Wainwright and Jordan, 2008). Approximate inference algorithms optimize over outer or inner bounds to the marginal polytope. Belief propagation and tree-reweighted sum product optimize over the local consistency constraints. Other bounds, such as the cycle relaxation (Sontag and Jaakkola, 2007) or a semidefinite outer bound (Torr, 2003; Peng et al., 2012) have also been used for approximate inference.

Solving for moment recovery jointly with consistency constraints, we obtain the following optimization problem for robust recovery of the true moments of the latent variables from observations of the anchors:

$$\mu = \operatorname{argmin}_{\mu' \in \mathcal{P}} \sum_{Z \subseteq Y: |Z| \leq K} D_{KL}(\mu_{A_Z} || R_Z \mu'_Z), \quad (6.8)$$

where K is the size of the moments needed within the structure learning algorithm (e.g., $K = 2$ for a tree-structured distribution (Chow and Liu, 1968)) and \mathcal{P} denotes the marginal polytope or a relaxation thereof. Note that Equation 6.8 is very similar to its non-robust counterpart, Equation 6.6. The main difference is in the constraints under which the optimization problem is solved. In fact, the approach of solving multiple independent moment recovery problems using Equation 6.6, described when introducing the robust recovery algorithm, corresponds

to an outer bound on the marginal polytope constraints (consisting only of non-negative and sum-to-one constraints), looser than the polytope formed by the local consistency constraints of Equation 6.7.

We use the conditional gradient, or Frank-Wolfe (Frank and Wolfe, 1956), method to minimize (6.8). Frank-Wolfe solves this convex optimization problem by repeatedly solving linear programs over \mathcal{P} . When \mathcal{P} corresponds to the local consistency constraints or the cycle relaxation (Sontag and Jaakkola, 2007), these linear programs can be solved using off-the-shelf linear programming solvers (e.g., Gurobi Optimization, 2014). Alternatively, when there are sufficiently few variables, one can optimize over the marginal polytope itself. For this, we use the observation of Belanger et al. (2013) that optimizing a linear function over the marginal polytope can be performed by solving an integer linear program with local consistency constraints. In the experiments section we show that constrained optimization improves the robustness of the moment-recovery step compared to unconstrained optimization and that using increasingly tight approximations of the marginal polytope within the conditional gradient procedure yields increasingly improved results.

Constrained optimization has been used previously to improve the robustness of method-of-moments results (Shaban et al., 2015). Our work differs in that the constrained space naturally coincides with the marginal polytope, which allows us to leverage the Frank-Wolfe algorithm for interior-point optimization and relaxations of the marginal polytope that have been studied in the context of variational inference.

6.4 Method of moments learning

In this section, we describe a method of moments algorithm that takes the recovered marginals from Section 6.3 as inputs. We take a two step approach. First, the distribution $P(Y)$ is estimated, then it is used to estimate conditionals $P(X|Y)$. The final joint model,

$P(X, Y)$ is simply a product of the two distributions.

6.4.1 Learning $P(Y)$

Structure learning background: Approaches for Bayesian network structure learning typically follow two basic strategies: they either search over structures \mathcal{G} that maximize the likelihood of the observed data (score-based methods), or they test for conditional independencies and use these to constrain the space of possible structures. A popular scoring function is the BIC score (Lam and Bacchus, 1994; Heckerman et al., 1995):

$$\text{BIC}(\mathcal{G}) = \sum_{i=1}^m N \hat{I}(Y_i; Y_{Pa(i)}) - N \hat{H}(Y_i) - \log(N) 2^{|Pa(i)|},$$

where N is the number of samples and \hat{I}, \hat{H} are the empirical mutual information and entropy respectively. $Pa(i)$ denotes the parents of node i in graph \mathcal{G} . The last term is a complexity penalty that biases toward structures with fewer parameters. Once the optimal graph structure is determined, the conditional probabilities, θ , that parametrize the Bayesian network are estimated from the empirical counts for a maximum likelihood estimate.

BIC is known to be *consistent*, meaning that if the data is drawn from a distribution which has the same conditional independencies as a graph \mathcal{G}^* , once there is sufficient data optimizing the BIC score will recover \mathcal{G}^* (up to Markov equivalency). In general, finding a maximum scoring Bayesian network structure is NP-hard (Chickering, 1996). However, approaches such as integer linear programming (Jaakkola et al., 2010; Cussens and Bartlett, 2013), branch-and-bound (Fan et al., 2014), and greedy hill-climbing (Teyssier and Koller, 2005) can be remarkably effective at finding globally optimal or near-optimal solutions. Restricting the search to tree-structures, we can find the highest-scoring network efficiently using a maximum spanning tree algorithm (Chow and Liu, 1968).

A different approach is to use conditional independence tests. Under the assumption that

every variable has at most k neighbors, these can be used to give polynomial time algorithms for structure learning (Pearl and Verma, 1991; Spirtes et al., 2001), which are also provably consistent.

Our algorithm: We utilize the fact that both score-based and conditional independence based structure learning algorithms can be run using derived statistics rather than the raw data. Suppose $P(Y)$ is a Bayesian network with maximum in-degree of k . Then, to estimate the mutual information \hat{I} and entropy \hat{H} needed to evaluate the BIC score for all possible such graphs, we only need to estimate all moments of size at most $k + 1$. For example, to search over tree-structured Bayesian networks (Chow and Liu, 1968), we would need to estimate $P(Y_i, Y_j)$ for every i, j as mentioned in Section 6.3.2.

Theorem 3 formalizes the consistency properties of our algorithm.

Theorem 3. *Let $\mathcal{G}(Y)$ be the graph structure of a Bayesian network describing the probability distribution, $P(Y)$, of the latent variables in an anchored discrete factor analysis model. Using the low-order moments recovered in Equation 6.6 in a structure learning algorithm which is consistent for fully-observed data, is a consistent structural estimator. That is, as the number of samples, $N \rightarrow \infty$, the structure recovered, $\mathcal{G}'(Y)$, is Markov equivalent to $\mathcal{G}(Y)$.*

The consistency of the algorithm follows from the consistency of the moment recovery process, which was shown by Chaganty and Liang (2014), combined with a consistent structural estimator.

6.4.2 Learning factor loadings, $P(X|Y)$

In the previous section, we showed how to learn a Bayesian network for $P(Y)$. Now we turn to estimating the conditional distributions in the model, $P(X|Y)$. We use $\mathcal{G}(Y)$ to denote the Bayesian network structure.

To properly estimate a failure probability, $f_{i,j}$, from low order moments, we need to

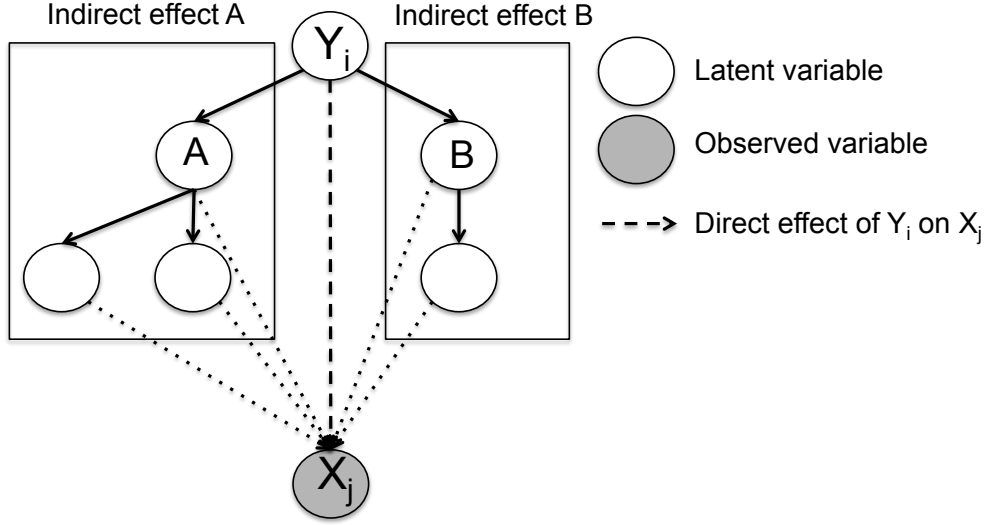


Figure 6.2: Y_i has a direct effect on the distribution of X_j as well as indirect effects that pass through its two neighbors. Correction factors are introduced to cancel the indirect effects through each neighbor, leaving only the direct effect which is modeled with the parameter $f_{i,j}$.

separate the direct effect of Y_i in turning on X_j and all other indirect effects. An example of an indirect effect is that conditioning on $Y_i = 1$ changes the likelihood of another latent variable $Y_{i'}$ being on, which in turn affects X_j . Figure 6.2 shows direct and indirect effects.

Estimation using Markov blanket conditioning: The first method of separating direct and indirect effects uses the entire Markov blanket of each latent variable. Denote the Markov blanket of Y_i in $\mathcal{G}(Y)$ as $B_i \subset Y$. For any setting of $B_i = b$, the following is a consistent estimator of $f_{i,j}$:

$$\hat{f}_{i,j}^{blanket} = \frac{\hat{P}(X_j = 0 | Y_i = 1, b)}{\hat{P}(X_j = 0 | Y_i = 0, b)}, \quad (6.9)$$

as shown in Appendix F.2. The simplest application of this estimator is in models where the latent variables are assumed to be independent, where the Markov blanket is empty and the estimator is simply:

$$\hat{f}_{i,j}^{direct} = \frac{\hat{P}(X_j = 0 | Y_i = 1)}{\hat{P}(X_j = 0 | Y_i = 0)}. \quad (6.10)$$

This is the setting described in Chapters 3 and 4.

Unfortunately, for general graphs, in order to estimate these moments, we have to form moments that are of the same order as the Markov blanket of each latent variable which can potentially be quite large, even in simple tree models, giving poor computational and statistical efficiency.

Improved estimation for tree-structured models: When $\mathcal{G}(Y)$ is a tree, it is possible to carefully construct correction factors that correct for indirect effects while only requiring conditioning on two latent variables at a time. Without loss of generality, the tree can be seen as rooted at any variable Y_i . Let \mathcal{N}_i denote the children of Y_i in $\mathcal{G}(Y)$. When calculating $f_{i,j}^{tree}$, we introduce one correction factor, $c_{i,j,k}$, for each child Y_k . The correction factors $c_{i,j,k}$ have the form:

$$c_{i,j,k} = \frac{\sum_{y_k \in \{0,1\}} \hat{P}(y_k | Y_i = 1) \hat{P}(X_j = 0 | Y_i = 0, y_k)}{\hat{P}(X_j = 0 | Y_i = 0)}. \quad (6.11)$$

See Appendix F.3. Each factor represents the effect of the entire subtree rooted at Y_k on X_j . This procedure is depicted in Figure 6.2, where the subtrees to be subtracted off are labeled A and B.

Correcting for all of the neighbors gives the following estimator:

$$\hat{f}_{i,j}^{tree} = \left(\prod_{k \in \mathcal{N}_i} \frac{1}{c_{i,j,k}} \right) \frac{\hat{P}(X_j = 0 | Y_i = 1)}{\hat{P}(X_j = 0 | Y_i = 0)}, \quad (6.12)$$

where the required conditional probabilities can be estimated from the low-order moments recovered with the methods in Section 6.3. An intuitive explanation is that the numerator of Equation 6.11 estimates the effect of Y_k as though Y_i is on, but leaves all other neighbors to behave as though Y_i is off. The other neighbors are canceled by the denominator and only the effect that flows through Y_k remains. Once this effect is isolated, it can be canceled out in Equation 6.12. This canceling procedure is different from conditioning (Equation 6.9) and

$\mathcal{G}(Y)$	Complexity of learning $\mathcal{G}(Y)$	Factor loadings
independent	None	\hat{f}^{direct} (Eq. 6.10)
tree	Chow-Liu: $O(n^2)$	\hat{f}^{tree} (Eq. 6.12)
degree-K	Indep tests: $O(n^2 \sum_{i=1}^K \binom{n}{i} 2^K)$	$\hat{f}^{blanket}$ (Eq. 6.9)
indegree-K	Score-based: NP-hard worst case	$\hat{f}^{blanket}$ (Eq. 6.9)

Table 6.1: Complexity of learning different model classes. After performing the moment-transformations (Section 6.3), the complexity of learning the models with latent variables is no harder than learning with fully observed moments.

the two can be used in conjunction to learn parameters in more complicated graphs.

For the correction factors to be defined, we require that it is possible to condition on $(y_i = 0, y_k = \{0, 1\})$, so we require that for all pairs of latent variables, Y_i, Y_k , $P(Y_k|Y_i)$ is not deterministic.

Once all other parameters are learned, the leak parameters can be estimated to correct the marginal probabilities of the observations (Appendix F.4).

6.4.3 Putting it all together

The full learned model is a product of the latent distribution $P(Y)$, which is described by an arbitrary Bayesian network $G(Y; \theta)$ and the factor loadings which are described by noisy-or link functions. The computational complexity of learning the model depends on the choice of constraints for the Bayesian network, $G(Y)$. Table 6.1 outlines the different classes of models that can be learned and the associated computational complexities. The robust recovery procedure (Section 6.3.2) introduces additional computational complexity, but it is not included in the table since even the naive recovery procedure provides consistency. Despite the high computational complexity for some of the settings outlined in Table 6.1, the models are all *identifiable* from moments that are bounded in size by the largest Markov blanket plus two.

6.5 Model evaluation

We perform an empirical study of our learning algorithm using two real-world multi-tag datasets, testing its ability to model observations using correlated factors.

In all of our experiments, we provide the algorithms and baselines with the empirical conditional probabilities of the anchors. Other methods of estimating these values exist (discussed in Section 8.1.1), but here we use the ground truth values for these noise rates in order to focus on the errors that arise from modeling error and finite data samples. Even though we constrain the moments to be valid using relaxations of the marginal polytope constraints, the learned parameters (using Equations 6.9, 6.10 or 6.12) may be out of valid ranges. We clip failures to be within $[0,0.99]$ and the leak parameters to lie between $[0.01, 1]$. Details on the optimization procedures and parameters used can be found in Appendix F.6.

6.5.1 Emergency Dataset

The emergency department dataset was previously described in Chapter 3. It consists of a corpus of medical records, collected over 5 years in the emergency department of a Level I trauma center and tertiary teaching hospital in a major city. Features consist of binary indicators from processed medical text and medication records. Patients are filtered to exclude patients with fewer than two of the specified conditions, leaving 16,268 patients. Details of the processing and a selection of the physician-specified anchors can be found in Section 3.5.1.

6.5.2 Stack Overflow Dataset

To demonstrate the broad usefulness of this method beyond the clinical phenotyping setting and to provide a reproducible basis for comparisons with future work, we also evaluate our methods on a large publicly available dataset using questions from Stack Overflow, a

popular collaboratively edited question and answer site for programmers¹. We model the user provided tags for each question with latent variables.

We use the 50 most popular tags from the dataset. The dataset initially contains 5 million questions. After filtering for questions that contain at least two from the 50 most popular tags, we are left with 695,170 questions. Models are trained on 500,000 questions and tested on a heldout set of 5000 questions. The remaining questions are left unused for future model development and testing.

The observed vocabulary consists of the 1000 most common tokens in the questions, using a different vocabulary for the question header and body. Each question is described by a binary bag-of-words feature vector, denoting whether or not each word in the vocabulary occurs in the question text.

A simple heuristic rule is used to define anchor variables: for each tag, the text of the tag appearing in the header of the question is used as an anchor. For example, for the tag *unix*, the anchor is a binary feature indicating whether the header of the text contains the word “unix”.

6.5.3 Qualitative evaluation – Face validity of models

Prior distribution of tags $P(Y)$: In this section we evaluate the quality of the learned representations of the latent variables, $P(Y)$. This task is interesting in its own right in settings where we care about understanding the interactions of the latent variables themselves for the purposes of knowledge discovery and understanding. Figure 6.3 shows a tree-structured graphical model learned to represent the distribution of the 23 latent variables in the Emergency dataset as well as small sub-graphs from a more complex model.

In the tree-structured model, highly correlated conditions are linked by an edge. For

¹<http://blog.stackoverflow.com/category/cc-wiki-dump/>. This data was also used in a Kaggle competition: <https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction>

example, asthma and allergic reactions, or alcohol and suicidal ideation. This is significant, since the model learning procedure learns only using anchors and without access to the underlying conditions. The insert shows subgraphs from a model learned with two parents per variable. This allows for more complex structures. For example: being HIV positive makes the patient more at risk for developing infections, such as cellulitis and pneumonia. Either one of these is capable of causing septic shock in a patient. The v-structure between cellulitis, septic shock, and pneumonia expresses the *explaining away* relationship: knowing that a patient has septic shock raises the likelihood of cellulitis or pneumonia. Once one of those possible parents is discovered (e.g., the patient is known to have cellulitis), the septic shock is explained away and the the probability of having pneumonia is reduced. In the second example relationship, both asthma and urinary tract infections are associated with allergic reactions (asthma and allergic reactions are closely related and many allergic reactions in hospital occur in response to antibiotics administered for infections), but asthma and urinary tract infections are negatively correlated with each other since either one is sufficient to explain away the patient’s visit to the emergency department.

Complete tree structured models for both datasets, as well as graphs with maximum in degree of two are found in Appendices F.7.1 and F.7.2.

Factor loadings $P(X|Y)$: Table 6.2 shows a selection of learned factor loadings on the Stack Overflow dataset, learned using the tree-structured estimation procedure described in Section 6.4.2. The rest of the learned factors for both Stack Overflow and the Emergency datasets are found in the Appendix F.7.3. The highly weighted terms for each of the tags make sense, for example, the mysql tag has observations related to database queries and the xml tag relates to document parsing.

The factors learned for the Emergency dataset are very similar to those learned in Chapter 3 (Table 3.4). One notable difference is the relationship between the factor “hematuria” and the observation of the word infection. In the models learned in Chapter 3, infection is a highly

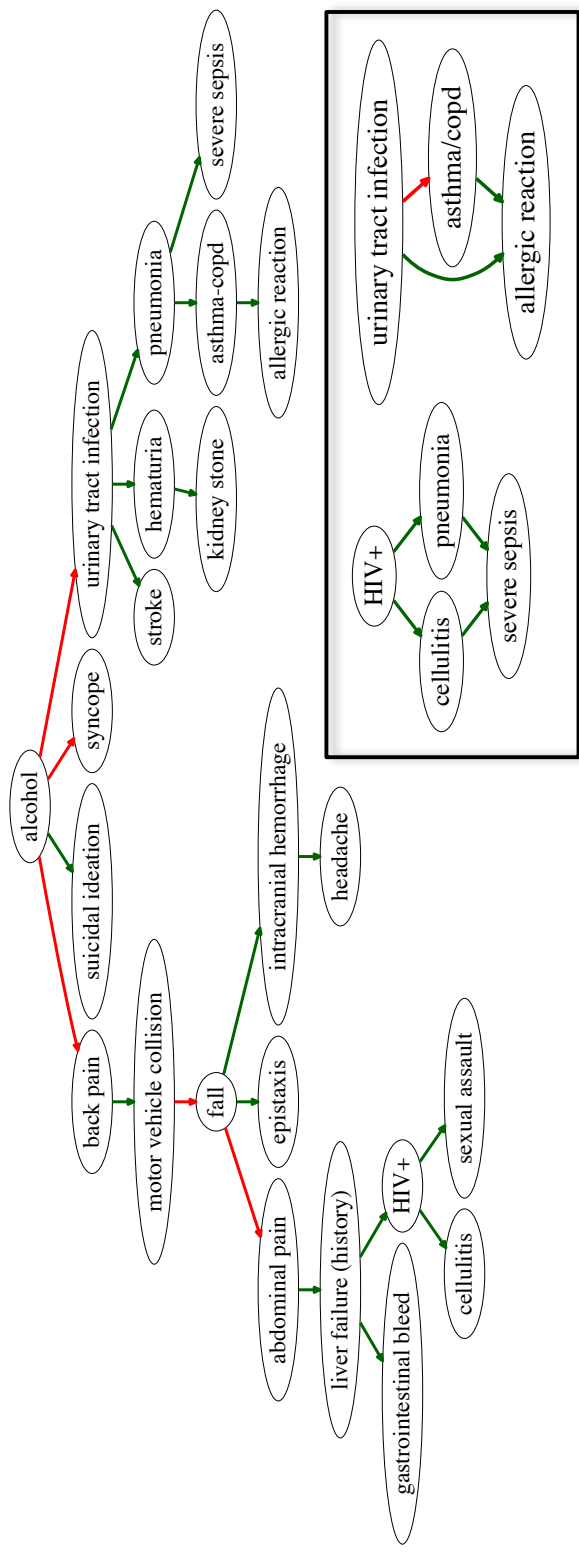


Figure 6.3: The learned tree structured latent variable model describing the distribution of 23 conditions in the emergency department, learned using marginal polytope constraints. Green lines represent positive correlations and red lines represent negative correlations. The section in the box shows small subgraphs of a more complex structure learned allowing for two parents per variable.

Stack Overflow	
Tag	Top weighted terms
osx	osx, i'm, running, i've, install, installed, os, code
image	image, code, size, upload, html, save, picture, width
mysql	mysql, query, rows, row, id, 1, tables, join
xml	xml, parse, document, read, data, string
linux	linux, running, command, machine, system, server
query	query, table, a, result, results, queries, tables, return
regex	regex, match, expression, regular, pattern, i'm

Emergency	
Tag	Top weighted terms
abdominal pain	pain, Ondansetron, nausea, neg:fevers
alcohol	male, sober, admits, found, drink
asthma	albuterol sulfate, sob, Methylprednisolone
stroke	age:80-90, admit, patient, head, ekg:
hematuria	male, urine, urology, blood, foley
HIV+	male, Truvada, cd4, age:40-50, Ritonavir
collision	car, neg:loc, age:20-30, hit, neck

Table 6.2: Top weighted words for factors in the Stack Overflow and Emergency. Words marked *neg:* are within a negation scope.

weighted for hematuria. In this chapter, where we model the correlations between factors instead of assuming independence, we see instead a positive association between the factors hematuria and urinary tract infection (Figure 6.3). Urinary tract infection is then highly associated with “infection” (not shown) and hematuria’s association is explained through its association with the urinary tract infection factor.

6.5.4 Quantitative evaluation – Held-out tag prediction

We test the ability of our learned models to perform inference by presenting it with the held out tag prediction task, previously described in Section 3.5.1. For this task, we use learn a tree-structure for $\mathcal{G}(Y)$ using the naive moment estimation, as we did not find that robust moment estimation helped in this task.

6.5.4.1 Baselines

We compare to the following relevant baselines and oracle results.

Noise tolerant discriminative training: We compare to the noise-tolerant learning procedure of Natarajan et al. (2013) using independent binary classifiers, trained using logistic regression with reweighted samples as described in their paper. This is a more recent and general algorithm than that of Elkan and Noto (2008) described in Section 2.2 (does not assume high positive predictive value of the anchors). To ensure a fair comparison with our learning algorithm, which is provided with the anchor noise rates, we also provide the baselines with the exact values for the noise rates. Since the learning approach of Natarajan et al. (2013) is not designed to use the noisy labels (anchors) at prediction time, we consider two variants: one ignores the anchors at test time, the other predicts only according to the noise rates of the anchors (ignoring all other observations), if the anchor is present. We report the best of the two baselines.

Imputation: We also compare to an imputation-based learning method, which learns a maximum likelihood model from the same model family as our learned model (using a tree distribution for the latent variables) from the fully imputed data. We use a single sample from the independent binary baseline classifiers (Natarajan et al., 2013) to impute the values of the latent variables.

Oracle comparisons: We compare to two different oracle implementations to deconvolve different sources of error. The first, *Oracle MoM*, uses a method-of-moments approach to learning the joint model (as described in Sections 6.4.1 and 6.4.2), but uses oracle access to the marginal distributions that include latent variables (i.e. does not incur any error when recovering these distributions using the method described in Section 6.3). The second oracle method *Oracle ML*, uses a maximum likelihood approach to learning the joint models with full access to the latent variables for every instance (i.e., learning as though all data is fully

observed). Comparing the gap between oracle ML and oracle MoM shows the loss in model quality that comes from choosing a method-of-moments approach over a maximum likelihood approach.

6.5.4.2 Results

Figure 6.4 shows the improvement over baseline on the heldout tag prediction task for the stack overflow and emergency datasets. We observe that learning the factor analysis model does indeed help with prediction in both tasks and we believe that this is because the independent classifiers in the baseline cannot take advantage of the correlation structure between latent variables. In the Stack Overflow corpus, we see that there is a clear advantage to learning a structured representation of the latent variables as the performance of the tree structured discrete factor analysis model outperforms the oracle bounds of a model with independent variables. In both datasets, the oracle results show that using method-of-moments for parameter learning is not a big source of disadvantage compared to maximum likelihood estimation. The gap between the method of moments and maximum likelihood oracle results are smaller in the tree structured models than in the independent models.

6.5.5 Robustness to model error

In practice, the anchors that we specify are never perfect anchors, i.e., they don't fully satisfy the conditional independence conditions. In this section, we test whether enforcing marginal or local polytope constraints (see Section 6.3.2) during the moment recovery process provides moment estimates that are more robust to imperfect anchors, improving the overall quality of the learned models.

We evaluate on the stack overflow dataset, which is a real world dataset in which we expect to have some anchors which violate conditional independence. We also generate a

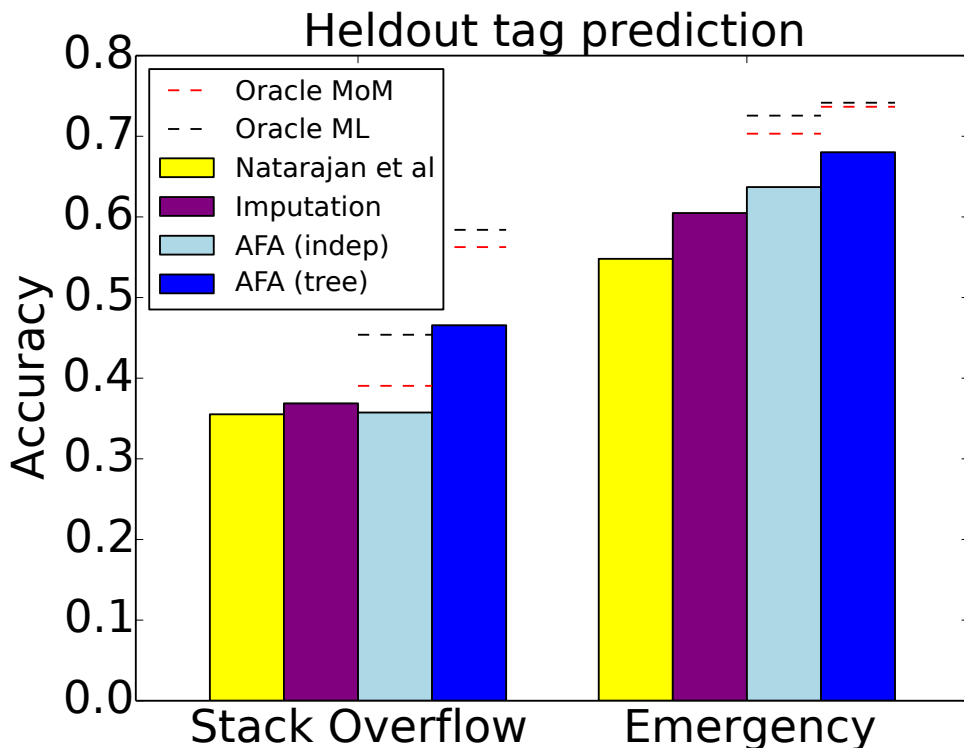


Figure 6.4: Average accuracy on the last-tag prediction task for 5K held-out instances for Emergency and Stack Overflow datasets. Anchored Factor Analysis (AFA) models are learned using a tree structured model for the latent variables and an independent model. Dotted lines are oracle results from the same model family.

synthetic corpus patterned after the stack overflow dataset by first training a model on the stack overflow dataset using method of moments (enforcing that the learned model has anchors), and then generating data from that model.

The weakest constraint we consider is using independent simplex constraints (naive moment recovery). In addition, we evaluate models learned with moments recovered using local consistency constraints, described in Equation 6.7 and the marginal polytope constraints described in Section 6.3.2. We use the moments recovered using each outer bound on the marginal polytope to learn a tree structured Bayesian network, and then evaluate the likelihood of the true tags from a held out set of data under the learned model.

Figure 6.5 shows the held-out likelihood of the tags according to tree models learned from moments recovered with increasingly tight approximations to the marginal polytope constraints on the Stack Overflow corpora. The tighter constraints learn higher quality models. This effect persists even in the large sample regime, suggesting that the residual gaps between the methods are due to sensitivity to model error. The oracle held-out likelihood on the real data set (using a model learned with oracle access to the true tags at training time) is -7.9 nats and on the synthetic dataset, -8.05 nats. Additionally, for the synthetic dataset, and the gap between models learned with tighter or looser approximations to the marginal polytope disappears.

6.5.5.1 Computational cost of robustness

Solving with tighter constraints does increase running time. For example, learning a model for Stack Overflow with 50 latent variables requires 30 seconds using the simplex constraints (trivially parallelized over 16 processes), 519 seconds for the local polytope and 6,073 seconds for the marginal polytope. Unlike EM-based procedures for learning models with latent variables which iterate over training samples in an inner loop, the running time of these method-of-moments approaches depends on the number of samples only when loading the data for the first time.

6.6 Discussion

6.6.1 Interpreting oracle results

As mentioned in section 3.7, it is important to note that the oracle results in this chapter are oracle results for models *within the model family of the anchored factor analysis models*. Independent logistic regression classifiers (one for each latent factor), actually perform better

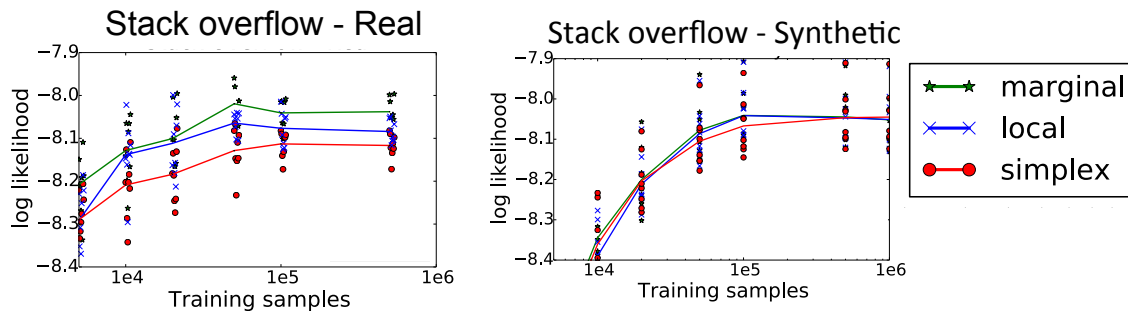


Figure 6.5: (Left) Per document held-out likelihood of learned tree structured models on Stack Overflow using held-out sets of 10K documents, each marker represents a different random train/test sample for the data. Solid lines represent the average of 8 runs. Different lines represent successively tight outer bounds to the marginal polytope constraints. (Right) When the Stack Overflow data is replaced by synthetic data drawn from a learned model, the difference between the constraints is much less pronounced.

at the held-out tag prediction task when provided with oracle access to the true values for the tags while training. Oracle logistic regression classifiers have accuracy of 0.78 for the emergency dataset (compare with 0.74 for oracle ML tree structured AFA) and 0.61 for stack overflow (compare to 0.58 with oracle ML tree structured AFA), and other, more expressive, fully supervised classifiers or generative models could potentially do even better. Thus, the comparison with oracle results is to show what we can learn with anchors instead of fully labeled data within a single model family, not to give an absolute upper limit on the possible performance on the task, which could be higher.

6.6.2 Advantage of using structured latent variables

Using structured latent variables can be useful in at least two different ways.

First, introducing the structure on the latent variables can a boost in performance on the held out tag prediction task, as the model can use the conditioning on some of the tags to help infer the last tag. For the emergency dataset, we see that this gain is small, possibly because after conditioning on the observations, the marginal information gain from conditioning on

the known tags is fairly small.

A second advantage to introducing the latent structure is that it can reduce the amount of misspecification. This is particularly relevant for the method-of-moments learning algorithm, which assumes the model family from the outset, and thus can be sensitive to model misspecification. In Figure 6.4, the gap between the method of moments oracle result and the maximum likelihood oracle result is reduced when the tree structured distribution for the latent variables is introduced. Thus, adding structured latent variables can then be useful for learning with method of moments.

Using a more complex graph structure for the latent variables (e.g., in degree ≤ 2), introduces more error in the moment recovery step (requires recovery of moments of 3 latent variables rather than 2), and thus did not provide performance improvements in our experiments. Balancing between introducing more expressive models to reduce model misspecification, and the corresponding increases in sample complexity is currently performed empirically.

6.6.3 Method of moments provides grounding

Anchors can be introduced to ensure that a latent variable reliably represents a real-world concept of interest. In our experiments, we found that this anchoring was ineffective within a likelihood-based learning framework without introducing additional terms to the objective (the “ungrounded anchor” problem described in Section 3.3). This problem was not observed when learning was completely within the method of moments framework. In these experiments, we use the same emergency dataset as that used in Section 3.3. The method of moments approach, which estimates every parameter using the anchors, provides a measure of grounding in a way that the likelihood approach does not.

6.6.4 Trading robustness and computational complexity

Improperly specified anchors can adversely affect a method-of-moments algorithm. We see this in the result that none of the models learned on real data in Figure 6.5 achieve the oracle value for held out likelihood at -7.9 nats whereas in the synthetic setting (where all of the anchors perfectly obey the conditional independence assumption), all of the different methods achieve the oracle value of -8.05 nats. Enforcing marginal polytope constraints can increase robustness to model error at the expense of adding computational complexity (Section 6.5.5.1). This computational complexity can be controlled by using outer bounds on the marginal polytope, trading off robustness with running time. Interestingly, we don't see improvements in the low-sample regime. Statistical complexity in the synthetic setting does not seem to be improved by constrained optimization.

6.7 Conclusions

Learning interpretable models with latent variables is a difficult task. In this chapter we present a fast and expressive method that allows us to learn models with complex interactions between the latent variables. The user specified anchors provide identifiability to the model and ground the latent variables providing interpretability, as shown in Section 6.5.3. On two different real-world datasets, we show that our method is able to find correlations between latent variables that are useful for inferring a held out tag, and outperform competitive baseline procedures at this task (Section 6.5.4). Enforcing marginal polytope constraints is useful for improving robustness to model error (Section 6.5.5), a technique we believe can be more widely applied. Anchors have an interesting property that they make the structure and parameter estimation with latent variables as easy as learning with fully observed data for method-of-moments algorithms that require low order moments. In contrast, likelihood-based learning remains equally hard, even with anchors, motivating approximations such as the

variational approximation used in Chapter 3.

6.8 Next steps and open questions

In this section we outline potential directions for future research.

6.8.1 Conditional distributions of anchors

Similar to the work in Chapter 3, a clear next step would be experimenting with methods of estimating the conditional distributions of the anchors, and determining the sensitivity of all of the steps of the algorithm presented here to that additional source of noise.

6.8.2 Likelihood objective

A natural next step would be to apply a refinement procedure based on the semi-supervised objective described in Chapter 3. As we note in that chapter, the semi-supervised objective can be applied to models with complex distributions over the latent factors.

6.8.3 Scientific dataset exploration

The method described in this chapter is exciting because it can be used to discover meaningful interactions between latent variables. So far we have applied it to two real-world tasks for evaluation, but we have not applied it more broadly with the express purpose of *learning something* about the underlying datasets. A natural application would be in modeling diseases and symptoms as in Chapter 5 and using the correlated diseases to understand comorbidity structures. So far, a limiting factor has been the requirement of determining the conditional distributions of the anchors in real datasets, which has held us back from

applying this method immediately in that setting, using the condition term extractions as anchors for the conditions themselves.

6.8.4 Subtracting off and conditioning

In Section 6.4.2, we note that the parameter estimation procedure of Equation 6.9 (which uses conditioning) and the cancellation procedure of Equation 6.12 are different and can be used together to learn more complicated graphs. Each method introduces error. Conditioning segments the dataset, reducing the statistical efficiency of learning; cancellation relies on the accurate estimates of the parameters from a previous step (similar to the subtracting off procedure in Section 4.3.3). Conditioning can be used to isolate sub-graphs that are locally tree-like, and then allow for the use of the cancellation procedure in those graphs. Given a graph structure for $P(Y)$, determining how to use the two methods together optimally is an open question.

The cancellation procedure of Equation 6.12 is limited to the setting of tree structured latent variables and noisy-or conditional distributions. General methods for estimating direct effects (such as Pearl’s back door procedure (Pearl, 2009)), could be useful for building canceling procedures that do not require tree structures, or that work for distributions beyond noisy-or.

6.8.5 Understanding robust moment recovery

A number of method of moments algorithms use constrained optimization or projection to improve robustness of the method and avoid returning solutions that do not correspond to valid models (Shaban et al., 2015; Cohen et al., 2013; Duchi et al., 2008; Lee et al., 2015). However, a theoretical characterization of when this works is still missing. Are there certain types of errors that are better corrected by applying constraints? For example, is there a

difference between a setting where a small number of anchors violate the anchor definition grossly compared to all of the anchors violating slightly, and which setting would benefit more from constrained optimization? As mentioned in Section 6.6.4, error due to finite samples (statistical noise) does not seem to be improved by constraints in the synthetic setting, but more investigation is warranted.

Chapter 7

Topic modeling with anchors

Acknowledgments This work was joint with Sanjeev Arora, Rong Ge, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, Michael Zhu. Additional thanks to Anantha Ravi Kiran for followup work. This work was previously published as “A Practical Algorithm for Topic Modeling with Provable Guarantees” Arora, Ge, Halpern, Mimno, Moitra, Sontag, Wu and Zhu. ICML 2013.

7.1 Introduction

This chapter describes a method-of-moments approach to learning topic models using an “anchor” assumption, very similar to the assumptions that appear in Chapters 2, 3 and 6. While not directly related to electronic medical records or the phenotyping task, the purpose of this chapter is to show that the anchor assumption is broadly useful in learning latent variable models, and to highlight some of the characteristics of a method-of-moments algorithm that successfully learns from real data.

In the remainder of this section, we give some background on the probabilistic topic modeling task and list the contributions of the chapter.

7.1.1 Probabilistic topic models

Probabilistic topic modeling is a popular method that learns thematic structure from large document collections without human supervision (Blei, 2012). Text has never been more important to how we communicate, or more easily available. But the massive text streams that are available today far outstrip anyone’s ability to read. Topic models are a vital tool for helping to navigate big text data, if we can train them quickly and reliably.

In probabilistic topic modeling we represent each document in a collection as a combination of words from various *topics*. Topics are shared across all documents, but each document has its own unique proportions specifying how much of each topic is present in that document. For example, a news article about legislation relating to retirement accounts might combine a topic related to *politics* and a topic related to *personal finance*. Other articles might exhibit other combinations, such as *politics* and *military conflict*, or *personal finance* and *computer software*.

The topics themselves are probability distributions over words. Like topics across documents, each word in the vocabulary is shared across all topics, but each topic has its own unique proportion of these words. A word like *account* may appear with high probability in several topics: it could refer to a financial product (a bank account) or a story (a fictional account). Less ambiguous terms, like *401k* (a US retirement account), might have significant probability only in one topic.

To extract topics from a large collection of documents, we posit a generative model that describes an imaginary process by which topics could generate documents (see Figure 7.1). We can then work backwards to find the topics that are most likely to have produced a specific set of documents. Each of the K topics consists of a different categorical distribution over words, A_1, \dots, A_K . To create each document, its topic proportions W_d are drawn from a distribution τ . Then, for each token i in the document, a topic $z_i \in \{1, \dots, K\}$ is sampled

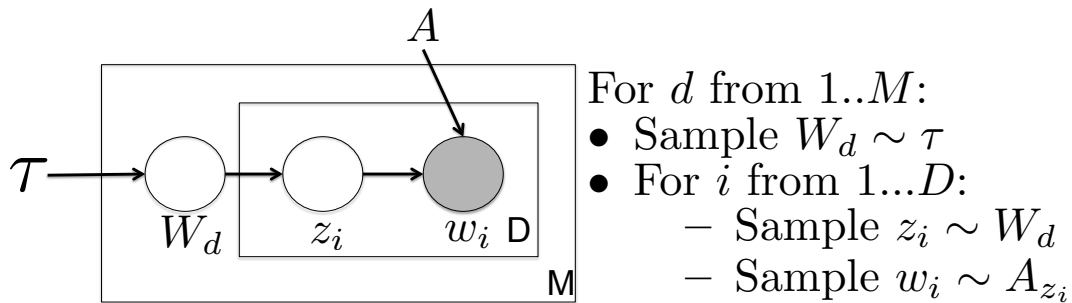


Figure 7.1: The generative model used in topic modeling.

from W_d , the topic-proportions of that document. A word is then chosen randomly from that topic’s distribution over words, $w_i \sim A_{z_i}$. This formulation is very general and includes the most widely used probabilistic topic model, Latent Dirichlet Allocation (LDA) (Blei et al., 2003), where τ is a Dirichlet distribution, as well as subsequent extensions such as Correlated Topic Models (Blei and Lafferty, 2007), where τ is a logistic Normal distribution, or Pachinko Allocation (Li and McCallum, 2007) where τ is a Bayesian network.

Even though this generative process is an unrealistic account of how documents are really created, it retains the important qualities of the model. Documents can be about many topics, and topics are shared between different documents in the corpus. This probabilistic approach has been used to uncover meaningful topics that have proven to be useful summarizations in a broad range of applications.

7.1.2 Challenges of learning probabilistic topic models

The challenges faced in learning probabilistic topic models are similar to those faced when learning noisy-or factor analysis models (Section 1.2.4).

The traditional method of learning probabilistic topic models is by *maximum likelihood estimation*, where we seek a set of K topics, $\{A_1, \dots, A_K\}$, that maximize the likelihood of the entire collection being generated by the procedure described above. This *learning* problem

of recovering the broad topics present in the collection is distinct from determining which topics are present in each document, often referred to as the *inference* task.

Maximum likelihood estimation is an extremely difficult optimization problem because the likelihood objective is non-convex, with many local maxima. Optimizing non-convex functions is notoriously difficult, and standard local-search based techniques like Expectation-Maximization (Dempster et al., 1977) or gradient ascent can only guarantee convergence to a local maximum, which may be much worse in terms of the objective value than the global optimum of the objective function. To make matters worse, even evaluating the likelihood function is intractable due to the presence of latent variables, namely the topic proportions of each document, W_d , and the topic assignments of each word, z_i . To evaluate the likelihood of a single document requires integrating over all possible topic-proportions, a high-dimensional integral with no closed form, as well as summing over an exponential number of possible topic assignments for the words in the document. Inference in even the simplest of topic models, LDA, is known to be NP-hard (Sontag and Roy, 2011).

Previous work in topic modeling solve approximate versions of the maximum likelihood problem. For example, the variational-EM approach (Blei et al., 2003; Hoffman et al., 2013) maximizes an objective that lower bounds the likelihood objective, but cannot guarantee that the solution is close to the optimum of the likelihood objective itself. The Markov Chain Monte Carlo (MCMC) approach (Griffiths and Steyvers, 2004) uses Markov chains constructed to generate samples from the posterior distribution of the parameters conditioned on the observed collection of documents, but suffers from well-known drawbacks: It is difficult to assess convergence, and the resulting samples are not guaranteed to be close to the maximum likelihood solution. These approximations to the maximum likelihood objective are, in a sense, necessary, since recent work has shown that even for the simple LDA model, finding the maximum likelihood solution is NP-hard, even if there are only two topics (Arora et al., 2012b).

These approximations tend to be slow in practice, because they contain an inner loop in which they perform an approximation to inference, determining which topics are likely present in each document in the collection. Thus, we seek a principled new approach that can circumvent both the hardness of maximum likelihood learning and inference.

Previous work on learning topic models with anchors by Arora et al. (2012b) is theoretically important in that it shows polynomial recoverability, but it is not practical. Its run-time is polynomial, but prohibitively large. Worse, as we show in the Experiments section, it is sensitive to violations of the modeling assumptions, learning poor quality topics when run on real-world data collections.

7.1.3 Contributions

The contribution of this chapter are as follows:

- We present a faster and more robust version of the provable method of moments algorithm for learning anchored topic models with arbitrary topic distributions originally presented in (Arora et al., 2012b).
- We show that models fit to real-world data sets nearly satisfy the anchor assumption. When run on data generated from these models, our algorithm is robust to these small violations of the anchor assumption, performing parameter recovery as well as a state of the art approach.
- We demonstrate that the new algorithm competes with state of the art approximate likelihood approaches on real data while running in a fraction of the time.

7.2 Anchor words

The correctness of our algorithm relies on an assumption that topics are *separable*, that is, that they can be reliably distinguished from one another via *anchor words*.

Anchor words, in the context of topic models, are specialized words that are specific to a single topic. For example, if the word *401k* occurs in a document then it is a strong indicator that the document is at least partially about *personal finance*. Natural language contains many of these unambiguous words, and thus the anchor word assumption is reasonable in many text-processing applications. Formally, a word is an *anchor* for topic k , if it has non-zero probability of being generated by topic k , and zero probability of being generated by any other topic. This is similar to the anchor definition from Chapter 2, applied to topic models.

If each topic in the topic model has at least one anchor word, we can provably recover the parameters of the model by solving an inverse problem constrained by the pairwise word co-occurrence counts, which are second order moments of the distribution.

Rather than optimizing generative likelihood, we consider the problem of finding topics that match the second-order moments, i.e. pairwise co-occurrence counts of words observed in the collection. For a vocabulary of size V , the expectation of the pairwise co-occurrence counts can be written as a $V \times V$ matrix:

$$Q = A\mathbb{E}_\tau[WW^T]A^T, \quad (7.1)$$

where A is a $V \times K$ matrix whose columns are topic distributions, and W is a $K \times M$ stochastic matrix whose columns are the topic proportions, W_d , for each document in the collection. W is unknown and stochastically generated: we can never expect to exactly recover it. However, if every topic has at least one anchor word, then Equation 7.1 is a *separable* nonnegative matrix factorization problem. Donoho and Stodden (2003) initially

showed that the separability criteria was sufficient to guarantee a unique solution for A and $R = \mathbb{E}_\tau[WW^T]$. Arora et al. (2012a) gave a polynomial time recovery algorithm. Thus, the anchor-word assumption gives a fruitful direction to explore for polynomial time topic-model learning.

7.3 The anchor words algorithm

7.3.1 From probability to geometry

Separable topic models have various important probabilistic and geometric properties. These properties will form the foundation for our algorithm for identifying anchor words and then using those anchor words to recover the parameters of the topic model. We will work with simple statistics measuring how often various pairs of words co-occur in a document.

We begin with a matrix Q whose entries represent co-occurrence *probabilities*: Consider a pair of words drawn randomly (without replacement) from a document. The words are denoted as $(w_1, w_2) \in [V]^2$ and their respective latent topic assignments are denoted as $(z_1, z_2) \in [K]^2$. Q is of dimension $V \times V$ and each element $Q_{i,j}$ is equal to $P(w_1 = i, w_2 = j)$. If we renormalize the rows of Q so that the rows sum to one, we obtain a matrix \bar{Q} containing conditional probabilities, so that $\bar{Q}_{i,j} = P(w_2 = j | w_1 = i)$. It is important to emphasize that we can estimate the entries of both Q and \bar{Q} from our data, but in our analysis we will need to bound how the error in our estimates contributes to errors in our inferred topic model. We will defer this complication for the time being, and assume that our estimates are exact.

It is useful to consider this data geometrically. We can view the rows of \bar{Q} as points in a simplex embedded in a V -dimensional space, where each dimension corresponds to a word in the vocabulary. In particular, each point is nonnegative and its entries sum to one. We will call a row of \bar{Q} an *anchor row* if it corresponds to an anchor word. A simplified illustration of

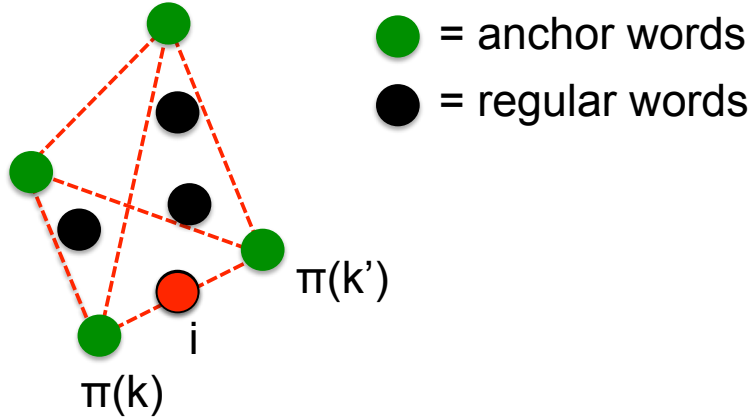


Figure 7.2: The rows of Q are vectors in V -dimensions, and their convex hull is a K -simplex whose vertices are anchor rows. Here $Q_i = \frac{1}{2}Q_{\pi(k)} + \frac{1}{2}Q_{\pi(k')}$ and this implies that the posterior distribution $P(z_1 = * | w_1 = i)$ assigns $\frac{1}{2}$ to $z_1 = k$ and $\frac{1}{2}$ to $z_1 = k'$ and zero to every other topic.

anchor and non-anchor points is shown in Figure 7.2. The key insight behind our algorithm is:

Lemma 3. *Each row of \bar{Q} is a convex combination of the anchor rows.*

Thus we can say much more about the rows of \bar{Q} . Not only do they belong to the simplex in V -dimensions, but their convex hull itself is a K -simplex where each vertex is an anchor row. This is the geometric property we will exploit when we design a simple, greedy algorithm for identifying the anchor words. Our proof of this lemma will be through elementary manipulations on various conditional probabilities.

Let us consider the following simplified setting. We will use the notation $\pi(k)$ to denote an anchor word for topic k . Then the definition of an anchor word gives us:

$$P(z_1 = k' | w_1 = \pi(k)) = \begin{cases} 1 & k' = k \\ 0 & \text{else} \end{cases}$$

This follows because when an anchor word is observed, there is only one topic that could

have generated it! Moreover let \bar{Q}_i denote the i th row of \bar{Q} . Then $\bar{Q}_i = P(w_2 = * | w_1 = i)$. It follows that

$$\bar{Q}_{\pi(k)} = P(w_2 = * | w_1 = \pi(k)) = P(w_2 = * | z_1 = k)$$

And finally we can write

$$\begin{aligned} \bar{Q}_i &= \sum_{k'} P(w_2 = * | w_1 = i, z_1 = k') P(z_1 = k' | w_1 = i) \\ &= \sum_{k'} P(w_2 = * | w_1 = \pi(k')) P(z_1 = k' | w_1 = i) \end{aligned}$$

This formula explicitly represents \bar{Q}_i as a convex combination of the anchor rows, but moreover we see that the convex combination is given by the conditional probabilities $P(z_1 = k' | w_1 = i)$ of which topic generated word $w_1 = i$. Thus our strategy is to find the anchor rows, and then solve a low-dimensional convex program to represent every non-anchor row as a convex combination of the anchor rows to find $P(z_1 = k' | w_1 = i)$. From there, we can use Bayes' rule to compute $P(w_1 = i | z_1 = k')$ which are exactly the parameters (except for the hyperparameters) of our topic model.

7.3.2 Finding anchors

A simple, greedy algorithm called `FindAnchors` (Algorithm 7) provably finds the first K anchor words. More precisely, given the matrix Q defined in the previous subsection, it finds the anchor rows. A graphical illustration of the algorithm is shown in Figure 7.3. However what is important about this algorithm is its behavior in the presence of noise, when we are given an estimate of Q instead. In that setting, it can be shown that `FindAnchors` recovers *near anchor words* — i.e. words whose row in Q is close in ℓ_1 distance to some anchor word. We need these latter types of guarantees to quantify how much data we need to get estimates that are provable close to the true parameters of the topic model.

Algorithm 7 FindAnchors

Input: \bar{Q} **Output:** Anchor indices Π

- 1: $\Pi = \emptyset$
- 2: **for** $k = 1$ TO K **do**
- 3: $\Pi = \Pi \cup \operatorname{argmax}_i \operatorname{dist}(\bar{Q}_i, \operatorname{span}(\{\bar{Q}_j\}_{j \in \Pi}))$
- 4: **end for**
- 5: **for** $k = 1$ TO K **do**
- 6: $\pi(k) = \operatorname{argmax}_i \operatorname{dist}(\bar{Q}_i, \operatorname{span}(\{\bar{Q}_j\}_{j \in \Pi \setminus \pi(k)}))$
- 7: **end for**
- 8: Return Π

Notation: $\operatorname{span}(S)$ denotes the subspace spanned by the points in the set S . The Euclidean distance between a point, x , and a subspace, $\operatorname{span}(S)$, denoted as $\operatorname{dist}(x, \operatorname{span}(S))$ is $\min_{s \in \operatorname{span}(S)} \|x - s\|$. When $S = \emptyset$, $\operatorname{span}(S)$ contains only the 0 vector.

The algorithm starts with a set that contains the point farthest from the origin. Then it iteratively add points that maximize distance from the span of the previously collected points. This procedure can also be seen as iteratively growing the simplex, adding vertices that greedily maximize the enclosed volume. While the general problem of choosing K rows of a matrix \bar{Q} to maximize the enclosed volume is NP-hard, it becomes easy when the points are known to lie in a simplex and the vertices of the simplex are themselves among the input points.

For the purpose of improving the noise tolerance, we add a second “clean up” stage that iteratively removes each vertex and adds back the point farthest from the span of the remaining vertices. While this additional round of cleanup has been previously suggested as a heuristic to improve quality, here we show that it also improves the theoretical guarantees of the algorithm.

Finally, the running time of this algorithm can be further improved by using random projection. Randomly projecting a collection of vectors in high dimensions onto a random low-dimensional subspace is well-known to approximately preserve the pairwise distance

Algorithm 8 Recover-Topics (L2)

Input: Word co-occurrence matrix Q , Anchor indices $\Pi = \{\pi(1), \dots, \pi(K)\}$, Word probabilities p_w .

Output: Topic model parameters: A, R

```
1:  $\bar{Q} =$  row normalized  $Q$ 
2: for  $i = 1, \dots, V$  do
3:   Solve  $C_i = \operatorname{argmin}_{C_i \in \Delta^{K-1}} \|\bar{Q}_i - C_i^T \bar{Q}_\Pi\|^2$ 
4: end for
5:  $C = [C_1, \dots, C_V]^T$ 
6:  $A' = \operatorname{diag}(p_w)C$ 
7:  $A =$  column normalized  $A'$ 
8:  $R = A^+QA^{T+}$ 
9:
10: return  $A, R$ 
```

A^+ denotes the pseudoinverse of A .

between each pair of vectors. And since our algorithm iteratively finds the farthest point from a subspace, its behavior is preserved after a random projection. This refinement of the algorithm allows it to work with low-dimensional points, and improves its efficiency. The final running time is $\tilde{O}(V^2 + VK/\epsilon^2)$.

7.3.3 Topic recovery

Here we give an algorithm called `Recover-Topics` to provably recover the word-topic distribution when given the anchor words. The algorithm exploits the probabilistic and geometric properties of separable topic models, which we outlined earlier. Recall that every row of \bar{Q} can be written as a convex combination of the anchor rows. Moreover, the mixing weights are exactly the probabilities $P(z_1|w_1)$.

The algorithm solves the convex optimization problem of finding a vector of coefficients, $C_i \in \Delta^{K-1}$, that minimize $\operatorname{dist}(\bar{Q}_i, C_i^T \bar{Q}_\Pi)$, where \bar{Q}_Π denotes the matrix containing only the anchor rows of \bar{Q} . This is a minimization problem that can be solved effectively using the Exponentiated Gradient algorithm (Kivinen and Warmuth, 1995). The resulting matrix

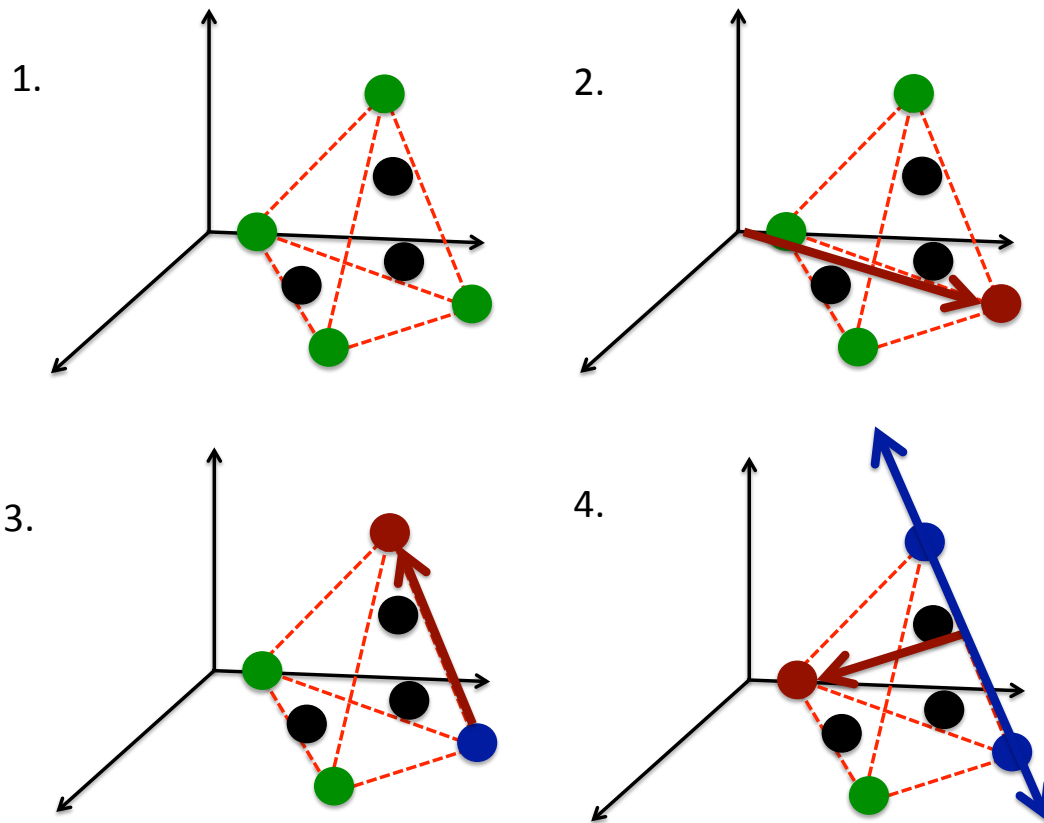


Figure 7.3: The first three steps of `FindAnchors` consist of finding a starting point furthest from the origin, finding the furthest point from the initial point, and finding the furthest point from the line defined by the first two points.

$C = [C_1, \dots, C_V]^T$, can be interpreted as containing conditional probabilities of the topic assignments conditioned on an observed word, $C_{i,j} = P(z_1 = j | w_1 = i)$.

Recovering the mixing weights is not quite enough, but in conjunction with an estimate of $P(w_1 = i)$, which can be obtained from simple word counts, we can recover the elements of the A matrix through Bayes' rule (this calculation is performed in matrix form on lines 6

and 7 of `Recover-Topics`):

$$P(w_1 = i | z_1 = k) = \frac{P(z_1 = k | w_1 = i) P(w_1 = i)}{\sum_{i'} P(z_1 = k | w_1 = i') p(w_1 = i')}.$$

Once the A matrix is recovered, it yields enough information to recover the topic-topic covariance matrix $R = \mathbb{E}[W^T W]$ along with it. Recall from Equation 7.1 that the second-order co-occurrence probability matrix Q , can be written as: $Q = A R A^T$. Thus we can recover R by computing $R = A^+ Q A^{T+}$, where A^+ denotes the pseudoinverse of A .

In the special case of the LDA model, we can additionally recover the Dirichlet hyperparameters. Recall that in implementing Bayes' rule, we compute for $k \in [K]$, the denominator $\sum_{i'} P(z_1 = k | w_1 = i') p(w_1 = i') = P(z_k)$ (this is done implicitly when normalizing the columns of A' in Algorithm 8), which gives us, up to a constant scaling, the Dirichlet hyperparameters. This scaling constant can be recovered from the R matrix as described in Arora et al. (2012b), but in practice we find it better to choose this single parameter using a grid search to maximize the likelihood of the data.

Different variants of this algorithm consider different distance functions $\text{dist}(\bar{Q}_i, C_i^T \bar{Q}_\Pi)$ to quantify the reconstruction loss. Using the squared Euclidean distance has the attractive property that the minimization on line 3 of `Recover-Topics` can be “kernelized”, making the running time of each iteration of exponentiated gradient independent of the vocabulary size, V .

The objective can be re-written in kernelized form as:

$$\|\bar{Q}_i - C_i^T \bar{Q}_\Pi\|^2 = C_i^T (\bar{Q}_\Pi \bar{Q}_\Pi^T) C_i - 2C_i (\bar{Q}_\Pi \bar{Q}_i^T) + \|\bar{Q}_i\|^2,$$

where $\bar{Q}_\Pi \bar{Q}_\Pi^T$ is $K \times K$ and can be computed once and used for all words, and $\bar{Q}_\Pi \bar{Q}_i^T$ is $K \times 1$ and can be computed once prior to running the exponentiated gradient algorithm for word i

and $\|\bar{Q}_i\|^2$ is constant with respect to the optimization over C_i and can be ignored.

Solving the minimization problem with a tolerance of ϵ^2 requires $K \log K/\epsilon^2$ iterations of the Exponentiated Gradient (Kivinen and Warmuth, 1995) algorithm. The running time of `Recover-Topics` is $\tilde{O}(V^2K + VK^3/\epsilon^2)$ and the for-loop which constitutes the main computational bottleneck can be trivially parallelized.

7.3.4 Theoretical guarantees

Here we state rigorous guarantees on the sample complexity and running time of our algorithm. When we are given a finite set of samples, our empirical statistics — which we denote by \tilde{Q} — will be a good, but imperfect approximation to \bar{Q} . In order to bound how many samples we need to obtain some target accuracy in recovering the true parameters of the topic model, we need to track the various sources of error through our algorithm.

Moreover, we need that certain parameters are bounded in reasonable ranges to guarantee that the inverse problem we are trying to solve is well-posed. Recall that the existence of anchors implies that we are trying to solve a *separable* non-negative matrix factorization problem. We characterize the separability of the problem as follows:

Definition 5. *The word-topic matrix A is p -separable for $p > 0$ if for each topic k , there is some word i such that $A_{i,k} \geq p$ and $A_{i,k'} = 0$ for $k' \neq k$.*

If p is too small, then anchor words rarely occur and cannot help us in learning the parameters of our model. We will require a lower bound on p , and the sample complexity of our algorithm will depend on $1/p$. The running time will depend on $1/p$ as well, since we have to read in enough data points to see each anchor at least once. We will also need a second measure γ that we will use to denote the smallest singular value of R . When γ is too small, the problem of recovering A and R from $Q = ARA^T$ becomes unstable. Note that this measure also implies that a topic should not appear with a low probability since for any

topic i , it can be shown that $\gamma \leq P(z = i)$.

When the problem is well-behaved with respect to these two measures, our algorithm achieves the following guarantee:

Theorem 4. *There is a polynomial time algorithm that learns the parameters of a topic model if the number of documents is at least*

$$M = \max \left\{ O \left(\frac{\log V \cdot K^6}{\epsilon^2 p^6 \gamma^6 D} \right), O \left(\frac{\log K \cdot K^4}{\gamma^4} \right) \right\},$$

where p and γ are the two non-degeneracy measures defined above and $D \geq 2$ is the length of the shortest document. The algorithm learns the word-topic matrix A and the topic-topic covariance matrix R up to additive error ϵ .

To prove this theorem, we show that the `FindAnchors` algorithm successfully recovers near-anchor words, and the `Recover-Topics` algorithm accurately estimates the desired parameters given near-anchor words.

Before stating the guarantee for `FindAnchors` algorithm, we first introduce the following notion of α -covering.

Definition 6. *Consider a simplex P , with vertices $\{v_1, \dots, v_K\}$. A point x is said to α -cover a vertex v_k if whenever x is written as a convex-combination of the vertices, $x = \sum_{k'=1}^K c_{k'} v_{k'}$, then $c_k \geq 1 - \alpha$. A set of points α -covers the vertices if each vertex is α -covered by a point in the set.*

Clearly, we would like the anchor points to be α -covered by the set of near-anchors found by `FindAnchors` algorithm.

Let δ be the largest perturbation between the rows of \bar{Q} and \tilde{Q} , $\max_i \|\bar{Q}_i - \tilde{Q}_i\| \leq \delta$. Lemma 4 connects the indices found by `FindAnchors` and the true anchors.

Lemma 4. *If $\delta < (\gamma p)^3/20K$, `FindAnchors` acting on the perturbed matrix \tilde{Q} , will output indices $\tilde{\Pi}$ such that $\{\bar{Q}_{\tilde{\pi}(1)}, \dots, \bar{Q}_{\tilde{\pi}(K)}\}$ $O(\delta/\gamma p)$ -covers the true anchor rows $\{\bar{Q}_{\pi(1)}, \dots, \bar{Q}_{\pi(K)}\}$.*

Next we show the `Recover-Topics` algorithm is robust to perturbations in the vertices and the internal points, making it possible to bound the error in the reconstruction coefficients in Lemma 5.

Lemma 5. *When `Recover-Topics` is provided with the perturbed matrix \tilde{Q} and perturbed anchors which $O(\delta/\gamma p)$ -cover the true anchors, the element-wise error on the returned matrix A is less than $O(\delta K/\gamma^3 p^2)$.*

Combining these two lemmas, and standard concentration bounds for the empirical correlation matrix \hat{Q} , we get the guarantees in the main Theorem 4.

7.4 Experimental Results

The proposed method in this chapter, anchor finding followed by convex optimization for topic recovery, is both faster than standard probabilistic approaches and more robust to violations of model assumptions than previous provable approaches. We compare two parameter recovery methods and a standard, probabilistically motivated algorithm. The first method is a simple matrix inversion presented in Arora et al. (2012b), which we call `Recover`. This inversion method is theoretically optimal, but fails in practice. The second is the constrained recovery method using a squared ℓ_2 loss, which we call `RecoverL2` as shorthand for `Recover-Topics (L2)`. As a comparison, we also consider a state-of-the-art Gibbs sampling implementation (McCallum, 2002). We would like an algorithm to be fast, accurate, and robust to noisy data. We find that the anchor-based algorithm is substantially faster than the standard algorithm, especially for large corpora. To evaluate accuracy we test

the algorithms on semi-synthetic data (with known topic distributions) and real documents. In addition, we measure the effect of different sources of error and model mismatch.

7.4.1 Methodology

We train models on two synthetic data sets to evaluate performance when model assumptions are correct, and on real documents to evaluate real-world performance. To ensure that synthetic documents resemble the dimensionality and sparsity characteristics of real data, we generate *semi-synthetic* corpora. For each real corpus, we train a model using Gibbs sampling and then generate new documents using the parameters of that model (these parameters are *not* guaranteed to be separable; we found that about 80% of topics fitted by Gibbs sampling had anchor words).

We use two real-world data sets, a large corpus of **New York Times** articles (295k documents, vocabulary size 15k, mean document length 298) and a small corpus of **NIPS** abstracts (1100 documents, vocabulary size 2500, mean length 68). Vocabularies were pruned with document frequency cutoffs. We generate semi-synthetic corpora of various sizes from models trained with $K = 100$ from NY Times and NIPS, with document lengths set to 300 and 70, respectively, and with document-topic distributions drawn from a Dirichlet with symmetric hyperparameters 0.03.

For the first stage of the algorithm, anchor word recovery, we use the **FindAnchors** algorithm in all cases. The original linear programming-based anchor word finding method presented with **Recover** in (Arora et al., 2012b) is too slow to be comparable. For Gibbs sampling we obtain the word-topic distributions by averaging over 10 saved states, each separated by 100 iterations, after 1000 burn-in iterations.

We use a variety of metrics to evaluate the learned models. For the semi-synthetic corpora, we compute the **reconstruction error** between the true word-topic distributions

and the learned distributions. In particular, given a learned matrix \hat{A} and the true matrix A , we use bipartite matching to align topics, and then evaluate the ℓ_1 distance between each pair of topics. When true parameters are not available, a standard evaluation for topic models is to compute **held-out probability**, the probability of previously unseen documents under the learned model. This computation is intractable in general, but there are reliable approximations (Wallach et al., 2009; Buntine, 2009).

Topic models are useful because they provide interpretable latent dimensions. We can evaluate the **semantic quality** of individual topics using a metric called *Coherence* (Mimno et al., 2011). This metric has been shown to correlate well with human judgments of topic quality. If we perfectly reconstruct topics, all the high-probability words in a topic should co-occur frequently, otherwise, the model may be mixing unrelated concepts. Given a set of words \mathcal{W} , coherence is

$$Coherence(\mathcal{W}) = \sum_{w_1, w_2 \in \mathcal{W}} \log \frac{D(w_1, w_2) + \epsilon}{D(w_2)}, \quad (7.2)$$

where $D(w)$ and $D(w_1, w_2)$ are the number of documents with at least one instance of w , and of w_1 and w_2 , respectively. We set $\epsilon = 0.01$ to avoid taking the log of zero for words that never co-occur (Stevens et al., 2012). Coherence measures the quality of individual topics, but does not measure redundancy, so we measure **inter-topic similarity**. For each topic, we gather the set of the N most probable words. We then count how many of those words do not appear in any other topic's set of N most probable words. For these experiments we use $N = 20$. Some overlap is expected due to semantic ambiguity, but lower numbers of unique words indicate less useful models.

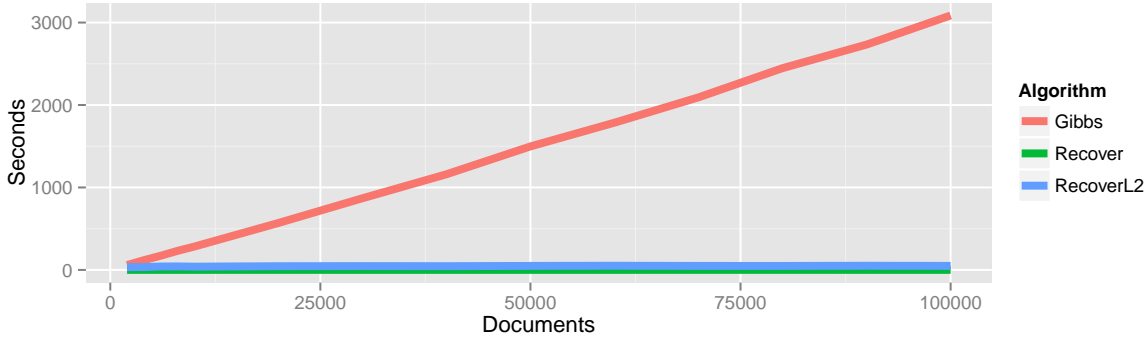


Figure 7.4: Training time on synthetic NIPS documents.

7.4.2 Efficiency

Both the `Recover` and `RecoverL2` algorithms, in Python, are faster than a heavily optimized Gibbs sampling implementation in Java (Yao et al., 2009). Fig. 7.4 shows the time to train models on synthetic corpora on a single machine. Gibbs sampling is linear in the corpus size. `RecoverL2` is also linear ($\rho = 0.79$), but only varies from 33 to 50 seconds. Estimating Q is linear, but takes only 7 seconds for the largest corpus. `FindAnchors` takes less than 6 seconds for all corpora.

7.4.3 Semi-synthetic documents

The new algorithms have good ℓ_1 reconstruction error on semi-synthetic documents, especially for larger corpora. Results for semi-synthetic corpora drawn from topics trained on NY Times articles are shown in Fig. 7.5 (top) for corpus sizes ranging from 50k to 2M synthetic documents. In addition, we report results for the `Recover` and `RecoverL2` algorithms on “infinite data,” that is, the true Q matrix from the model used to generate the documents. Error bars show variation between topics. `Recover` performs poorly in all but the noiseless, infinite data setting. Gibbs sampling has the lowest ℓ_1 on smaller corpora. However, for the larger corpora the new `RecoverL2` algorithm have the lowest ℓ_1 error and

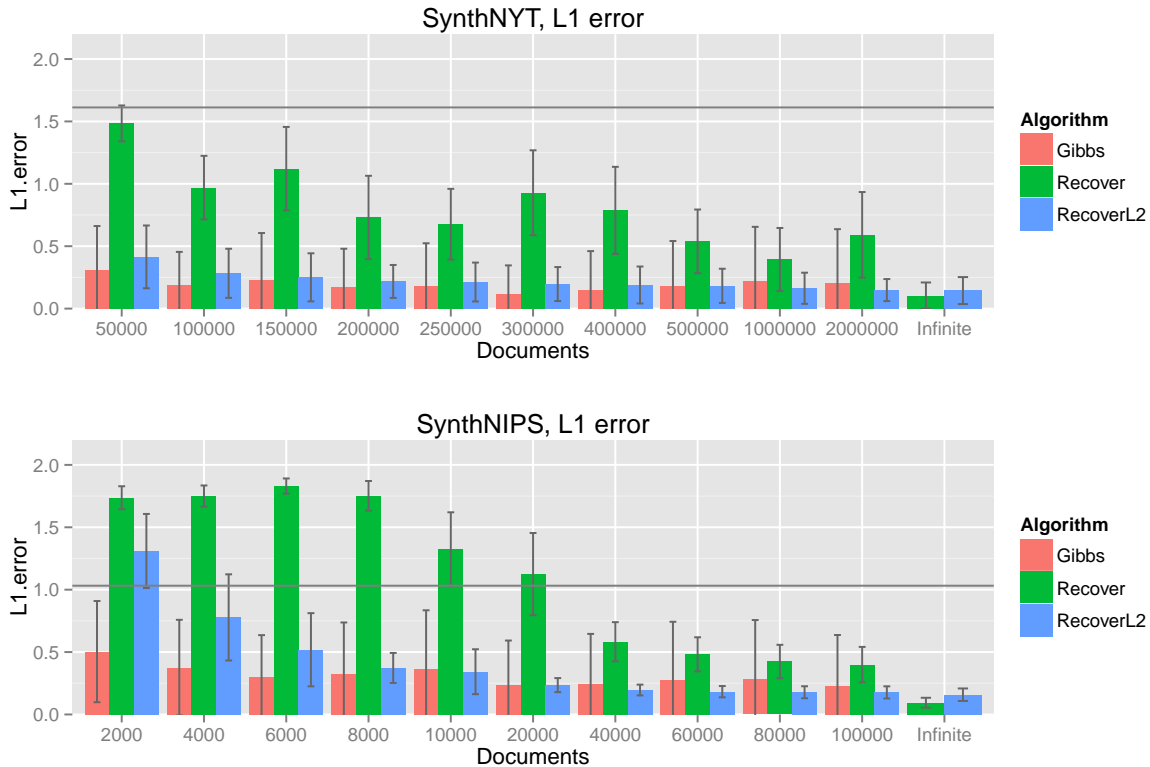


Figure 7.5: ℓ_1 error for learning semi-synthetic LDA models with $K = 100$ topics (**top**: based on NY Times, **bottom**: based on NIPS abstracts). The horizontal lines indicate the ℓ_1 error of K uniform distributions.

smaller variance (running sampling longer may reduce MCMC error further). Results for semi-synthetic corpora drawn from NIPS topics are shown in Fig. 7.5 (bottom), and are similar.

Effect of separability. Notice that in Fig. 7.5, **Recover** does not achieve zero ℓ_1 error even with noiseless “infinite” data. Here we show that this is due to lack of separability, and that the new recovery algorithms are more robust to violations of the separability assumption. In our semi-synthetic corpora, documents are generated from an LDA model, but the topic-word distributions are learned from data and may not satisfy the anchor words assumption. We now add a synthetic anchor word to each topic that is, by construction,

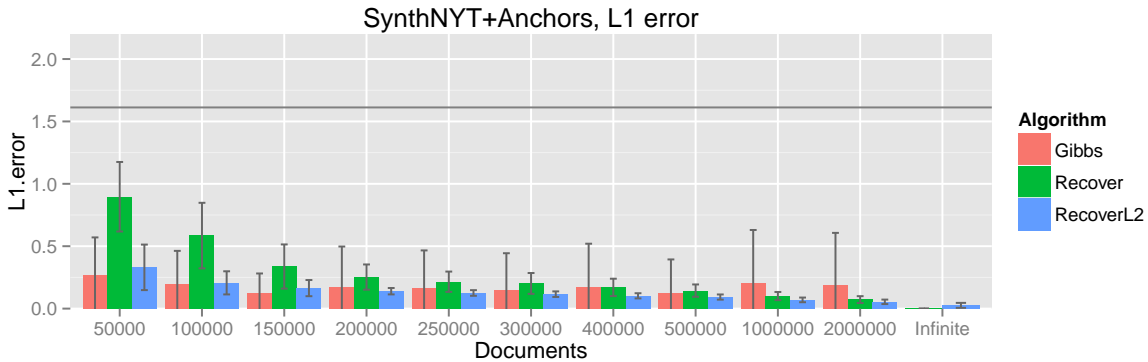


Figure 7.6: When we add artificial anchor words before generating synthetic documents, ℓ_1 error goes to zero for `Recover` and close to zero for `RecoverL2`.

unique to that topic. We assign the synthetic anchor word a probability equal to the most probable word in the original topic. This causes the distribution to sum to greater than 1.0, so we renormalize. Results are shown in Fig. 7.6. The ℓ_1 error goes to zero for `Recover`, and close to zero for `RecoverL2` (not zero because we do not solve to perfect optimality).

Effect of correlation. The theoretical guarantees of the new algorithms apply even if topics are correlated. To test the empirical performance in the presence of correlation, we generated new synthetic corpora from the same $K = 100$ model trained on NY Times articles. Instead of a symmetric Dirichlet distribution, we use a logistic Normal distribution with a block-structured covariance matrix. We partition topics into 10 groups. For each pair of topics in a group, we add a non-zero off-diagonal element (ρ) to the covariance matrix. This block structure is not necessarily realistic, but shows the effect of correlation. Results for $\rho = 0.05$ and 0.1 are shown in Fig. 7.7. `Recover` performs much worse with correlated topics than with LDA-generated corpora (c.f. Fig. 7.5). The other three algorithms, especially Gibbs sampling, are more robust to correlation. Performance consistently degrades as correlation increases. For the recovery algorithms this is due to a decrease in γ , the condition number of the R matrix. With infinite data, ℓ_1 error is equal to the ℓ_1 error in the uncorrelated

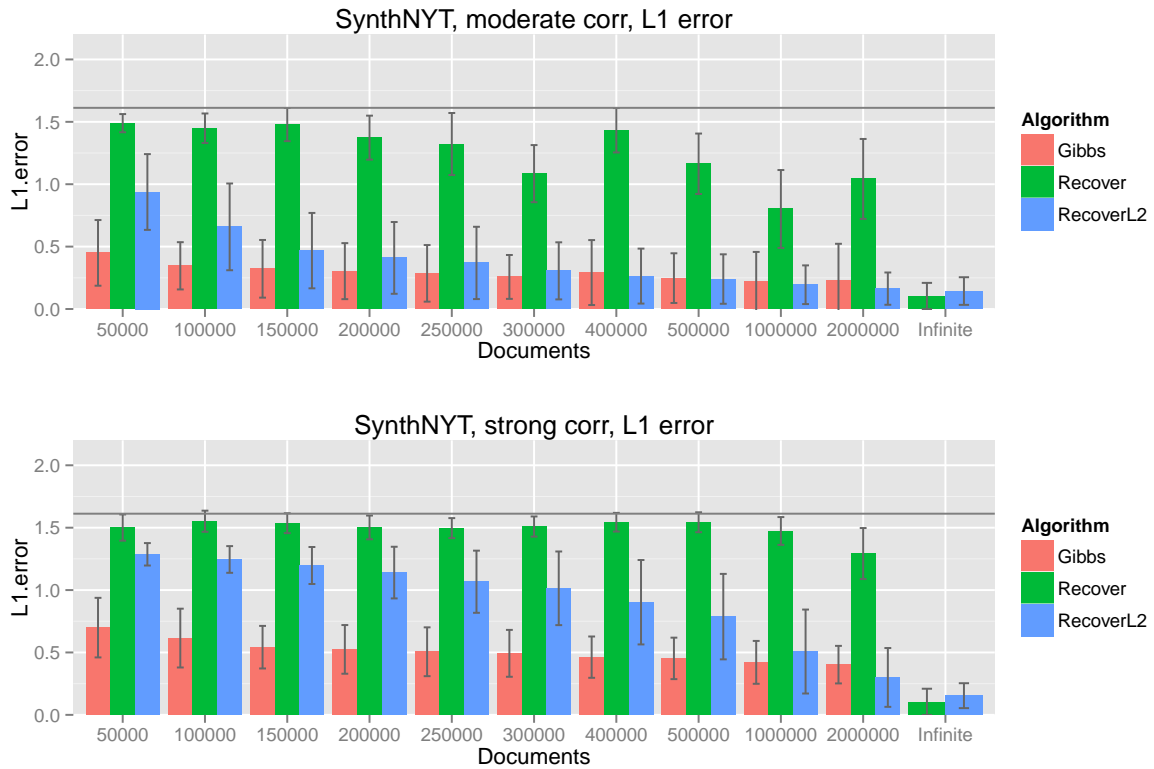


Figure 7.7: ℓ_1 error increases as we increase topic correlation (**top**: $\rho = 0.05$, **bottom**: $\rho = 0.1$). Based on the NY Times semi-synthetic model with 100 topics.

synthetic corpus (non-zero because of violations of the separability assumption).

7.4.4 Real documents

The new algorithms produce comparable quantitative and qualitative results on real data. Fig. 7.8 shows three metrics for both corpora. Error bars show the distribution of log probabilities across held-out *documents* (top panel) and coherence and unique words across *topics* (center and bottom panels). Held-out sets are 230 documents for NIPS and 59k for NY Times. For the small NIPS corpus we average over 5 non-overlapping train/test splits. The matrix inversion step in **Recover** fails for the NIPS corpus so we modify the procedure to use pseudoinverse. This modification is described in Appendix G.4.2. In both corpora, **Recover** produces noticeably worse held-out log probability per token than the other algorithms. Gibbs sampling produces the best average held-out probability ($p < 0.0001$ under a paired t -test), but the difference is within the range of variability between documents. We tried several methods for estimating hyperparameters, but the observed differences did not change the relative performance of algorithms. Gibbs sampling has worse coherence than the other algorithms, but produces more unique words per topic. These patterns are consistent with semi-synthetic results for similarly sized corpora (Appendix G.3).

For each NY Times topic learned by **RecoverL2** we find the closest Gibbs topic by ℓ_1 distance. The closest, median, and farthest topic pairs are shown in Table 7.1. We observe that when there is a difference, recover-based topics tend to have more specific words (*Anaheim Angels* vs. *pitch*).

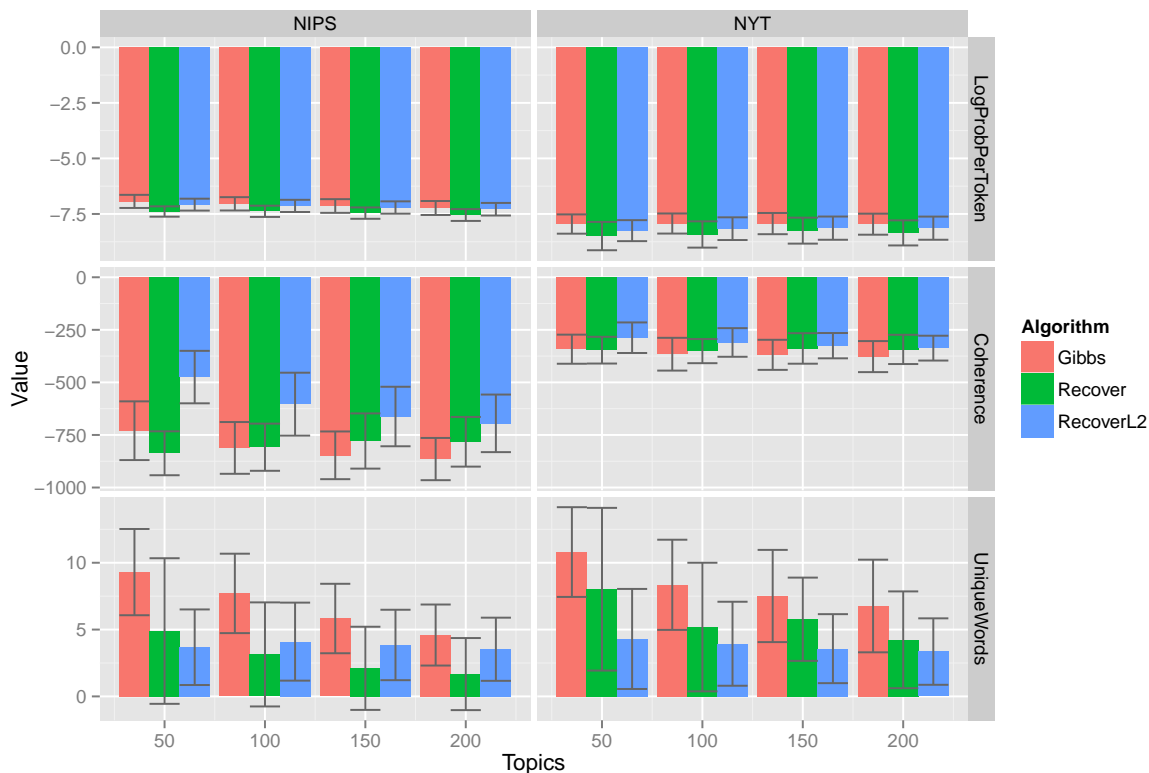


Figure 7.8: Held-out probability (per token) is similar for RecoverL2 and Gibbs sampling. RecoverL2 has better coherence, but fewer unique terms in the top $N = 20$ words than Gibbs. (Up is better for all three metrics.)

Table 7.1: Example topic pairs from NY Times (closest ℓ_1), anchor words in bold. The UCI NY Times corpus includes named-entity annotations, indicated by the zzz prefix. All 100 topics are shown in Appendix G.3.1.

RecoverL2	run inning game hit season zzz_anaheim_angel
Gibbs	run inning hit game ball pitch
RecoverL2	father family zzz_elian boy court zzz_miami
Gibbs	zzz_cuba zzz_miami cuban zzz_elian boy protest
RecoverL2	file sport read internet email zzz_los_angeles
Gibbs	web site com www mail zzz_internet

7.5 Discussion

7.5.1 Computational efficiency

Computational efficiency is of utmost importance for learning and using topic models from big data. The run-time of the anchor-word learning algorithm is nearly independent of the number of documents on which it is run.

With sampling and EM approaches, more data means more work. In our setting, the number of samples only shows up because you have to read in the data, but aside from that dependence, more samples do not increase run time. This confirms with intuition, more samples should be a *good thing*, it should not make the problem harder!

Many aspects of the anchor-word learning algorithm can also be trivially parallelized. Construction of the input matrix Q requires summing over separate matrices derived from each individual document, so data need not be co-located on one server, and data can be imported in parallel. Anchor finding using the Gram-Schmidt process is sequential, but within each iteration, the projection step for each row of \bar{Q} can be trivially parallelized. Finally, the recovery step consists of a separate constrained linear regression for each non-anchor word, which again can be trivially parallelized.

Since this initial work, exciting advances have been made in stochastic variational inference for learning topic models (Hoffman et al., 2013; Mnih and Gregor, 2014) which can also learn very effectively and quickly on large data sets. Thus, the speed comparisons should be understood to be against a weaker baseline than current state of the art.

7.5.2 Related work

Similar problems have been studied extensively under the name of hyperspectral unmixing, where the separability or anchored assumption is called the “pure pixel” assumption. With

this assumption, the VCA (Nascimento and Dias, 2004) algorithm is very similar to the algorithm we propose here, without the cleanup phase. The N-FINDR (Gomez et al., 2007) algorithm tries to greedily maximize the volume of a simplex by doing iterations like the clean-up phase in our algorithm. Recently, such ideas have also been applied to the problem of non-negative matrix factorization (Thurau et al., 2010; Kumar et al., 2012). Our approach differs from Kumar et al. (2012) in that we use slightly different geometric interpretations (simplex vs. cones), and because our heuristic is (exactly) equivalent to greedily maximizing volume.

Anchor words are not the only assumption that allow for polynomial time learning of topic models. Anandkumar et al. (2012a) also present a provable algorithm for topic modeling based on third-order moments and tensor decomposition that does not require anchor words, but, unlike the algorithm of Arora et al., assumes that topics are not correlated. Although standard topic models like LDA (Blei et al., 2003) assume that topic proportions in a document are uncorrelated, there is strong evidence that topics are dependent (Blei and Lafferty, 2007; Li and McCallum, 2007): *economics* and *politics* are more likely to co-occur than *economics* and *cooking*.

Since its publication, the **AnchorWords** algorithm has been extended in a number of directions. Roberts et al. (2014) find that using an anchor-based model as an initialization for a probabilistic algorithm reduces variability and improves model fit. The success of the algorithm depends on finding high quality anchor words efficiently, but finding good anchors requires finding an optimal low-dimensional convex hull in a high-dimensional space. Ding et al. (2014) present an anchor finding algorithm that can be parallelized across multiple servers.

Unfortunately, finding anchor words that differ from the greedy Gram-Schmidt orthogonalization has proven difficult, but several promising approaches have been developed. Zhou et al. (2014) find anchors by repeated random projections of the word cooccurrence matrix into the

positive quadrant of a plane. For each projection, it is then simple to find the two extreme vectors. Words whose projection is often an extreme vector are likely to be good anchor words. Lee and Mimno (2014) replace random projections with a single heavy-tailed t -SNE projection that does not preserve pairwise ℓ_2 distances, but preserves local distances, allowing points to spread out in the projected space. When used with two- or three-dimensional t -SNE projections, this space is both visualizable and a convex hull with vertices corresponding to anchor words can be found analytically. Lee et al. (2015) projects the cooccurrence matrix to be low-rank and doubly non-negative (both are features of the cooccurrence matrix generated under ideal circumstances) and show how that learns more robust topics from small and noisy data.

There has also been some work on improving the topic recovery step. Nguyen et al. (2014) adds regularization to smooth the estimated topic-word distributions, resulting in improved interpretability.

7.5.2.1 Topic models and noisy-or factor analysis

Topic modeling bears some similarity to noisy-or factor analysis models, which appear in many other chapters of this thesis, with the main difference being in the type of data being modeled. The observed variables in a topic model are *counts* and the latent variables are probability vectors (i.e., non-zero and sum-to-one), rather than binary. The interpretations of the latent space are different as well. The latent variables in a topic model represent the *proportion* of a document that is related to a specific topic, whereas the variables in a noisy-or factor analysis model refer to *whether* a factor is active or not.

In Chapter 6 we address the setting where the latent variables in a non-independent. The latent variables in a topic model cannot be independent, since they must sum-to-one, but the LDA model uses a Dirichlet prior which is *close* to independent (samples from a Dirichlet can be generated by sampling independent gamma random variables and normalizing). An

interesting feature of the algorithm described in this chapter is that it can provably recover the word-topic matrix no matter how complex the prior distribution of the topics is, but it does not provide an explicit construction of the distribution $\mathbb{W}_d \sim \tau$. Rather, it provides only the pairwise moments of τ in the form of the R matrix. In contrast, the algorithm in Chapter 6 explicitly recovers a Bayesian network to describe the prior distribution of the factors.

Unlike the anchored noisy-or factor analysis work, where anchors are specified by domain experts, in the topic modeling setting, we are able to *find* anchors directly from the data by searching for extreme points in word-word co-occurrence space (Section 7.3.2). A similar geometric characterization of anchors in the noisy-or setting has not yet been found. Topic modeling bears some similarity to noisy-or factor analysis models, which appear in many other chapters of this thesis, with the main difference being in the type of data being modeled. The observed variables in a topic model are *counts* and the latent variables are probability vectors (i.e., non-zero and sum-to-one), rather than binary. The interpretations of the latent space are different as well. The latent variables in a topic model represent the *proportion* of a document that is related to a specific topic, whereas the variables in a noisy-or factor analysis model refer to *whether* a factor is active or not.

In Chapter 6 we address the setting where the latent variables are non-independent. The latent variables in a topic model cannot be independent, since they must sum-to-one, but the LDA model uses a Dirichlet prior which is *close* to independent (samples from a Dirichlet can be generated by sampling independent gamma random variables and normalizing). An interesting feature of the algorithm described in this chapter is that it can provably recover the word-topic matrix no matter how complex the prior distribution of the topics is, but it does not provide an explicit construction of the distribution $\mathbb{W}_d \sim \tau$. Rather it provides only the pairwise moments of τ in the form of the R matrix. In contrast, the algorithm in Chapter 6 explicitly recovers a Bayesian network to describe the prior distribution of the

factors.

Unlike the anchored noisy-or factor analysis work, where anchors are specified by domain experts, in the topic modeling setting, we are able to *find* anchors directly from the data by searching for extreme points in word-word co-occurrence space (Section 7.3.2). A similar geometric characterization of anchors in the noisy-or setting has not been developed.

7.6 Conclusions

We present a new algorithm for topic modeling, inspired by (Arora et al., 2012b), which is efficient and simple to implement yet maintains provable guarantees. The running time of this algorithm is effectively independent of the size of the corpus. Empirical results suggest that the sample complexity of the algorithm is somewhat greater than MCMC but it provides comparable results in a fraction of the time.

7.7 Next steps and open questions

In this section, we outline some possible next steps and open questions.

As discussed in the related work section, new methods of finding anchors could be useful in building interpretable topics. The current algorithm requires thresholds to ensure that rare words are not included as anchors (e.g., excluding words that do not appear in more than 100 documents from being anchors), even though they are likely to lie at extreme points due to their high variance. Adding a notion of certainty in the “location” of an anchor could naturally alleviate this problem. Another approach to finding anchors interprets the Gram-Schmidt algorithm as greedily maximizing the enclosed volume. Rather than maximizing the enclosed volume, we could attempt to minimize the distance from all other words to the convex hull of the anchors.

A general problem with method-of-moments algorithms is how to constrain their solutions to lie in the feasible space. In this work, we apply constraints to ensure that the A matrix is feasible, leaving the R matrix unconstrained, and leaving open the possibility that there may be no matrix R such that $ARA^T = Q$. This decision stemmed from an intuitive determination that the A matrix was somehow the more *important* part of the topic modeling process. When tradeoffs between obtaining valid parameters and violating moment constraints apply, is there a principled way of choosing between them?

Recovering an explicit representation of the topic distribution τ is important to be able to sample from the model. For latent Dirichlet allocation, there is a method to derive the Dirichlet parameters from the R matrix, but for general correlated topic models we do not have such a method. Fitting an expressive distribution (e.g., mixture of Dirichlets or deep generative models (Salakhutdinov, 2009)) to maximize likelihood or a moment matching objective while holding the word-topic distributions (i.e., the A matrix) fixed could be an interesting direction for further work.

The continued development of method of moments algorithms that are useful for general data-science tasks can allow us to effectively analyze large datasets that are being created at unprecedented scale.

Chapter 8

Closing

In this section, we return to the central theme of electronic medical record phenotyping, outline the contributions made in this dissertation, and the open questions that remain. As motivated in the introduction, health IT systems play an important role in making personalized medicine a reality, from providing reliable data for analyses and knowledge discovery to bringing relevant evidence-based guidelines to the bedside. A particularly difficult problem when applying machine learning to this task is finding gold standard labels to train classifiers. In this work, we explored semi-supervised methods based on structural features of the data distribution, which remove the need for manual labeling.

In this work, we worked closely with physicians from Beth Israel Deaconess Medical Center and focused on real-time phenotype prediction for use in the emergency department. Other work has focused on using anchors for research and knowledge discovery. For example, Agarwal et al. (2016) evaluate a method similar to that described in Chapter 2 for cohort selection in retrospective observational studies.

8.1 Looking forward

We envision a future where a large collection of phenotypes are widely available within EMR systems, forming a middleware layer upon which smart health IT applications can be built. Many EMR systems already handle manual rules involving simple queries and branching logic. For example: *If the patient has a listed allergy within their structured allergy list, then mark the allergen with a warning on all order sets.* We would like to see phenotype definitions widely available to be used in a similar way: *If the stroke phenotype is positive with probability above 80%, display stroke guidelines on the physician dashboard.*

Some short and long term steps towards this goal are listed here.

8.1.1 Determining conditional distributions of anchors

In both Chapters 3 and 6, we assume that the conditional distributions are known. We do so to highlight other aspects of the algorithm, assuming that it will be possible to obtain reasonable estimates for these values. In order to deploy these joint models in practice, we need to be able to estimate the conditional distributions. A number of approaches are possible. Singly coupled triplets (described in Chapter 4) or more general tensor decomposition methods can be used to estimate conditional probabilities of the anchors (Anandkumar et al., 2012c; Halpern and Sontag, 2013; Steinhardt and Liang, 2016). Platanios et al. (2014) presents a method of estimating conditional probabilities or noise rates based on learning the parameters of a simple graphical model. For anchors with high positive predictive value, the calibration step of Elkan and Noto (2008) (described in Chapter 2.2.4.1) provides an estimate of the conditional distribution. Scott et al. (2013) use *maximal source separation* to determine mixture proportions and label noise rates in a noisy label setting. A clear next step is to determine whether any of these methods provide sufficiently good estimates for the conditional distributions of the anchors to build models that outperform independent

classifiers. Another approach would be to use a small amount of manual labeling only for the purpose of determining the conditional observations. This could be made easier by building good user interfaces (possibly integrated into the physician’s workflow), and investigating non-uniform sampling techniques to reduce the number of queries required (Sawade et al., 2010).

8.1.2 Generalizability

We have publicly released a collection of anchors for 42 different phenotypes¹. A natural next step would be to partner with other institutions and determine whether the anchors that we have specified can be used to train phenotype estimators for use at their institutions. This may require further linking to publicly available standardized ontologies (we use First Databank’s Enhanced Therapeutic Classification to group medications) to make sure that the anchors can be found at their institution as well.

Another important dimension for generalizability is testing whether the initial anchors specified for the emergency department also work for intensive care units, floor, outpatient, or psychiatric settings. Anchors based on interventions may be specific to a particular setting (e.g., some interventions can only take place in intensive care units), but we would expect that the text-based and medication history anchors should transfer across settings.

8.1.3 Clinical evaluations

As a middleware level, the utility of the phenotypes is ultimately measured within a particular health IT application in clinical terms such as reduction of preventable errors, improved compliance with clinical guidelines, and most importantly, improved patient outcomes. An initial clinical trial could include a single phenotype that is linked to an emergency

¹Available on github: <https://github.com/clinicalml/clinical-anchors>

department clinical pathway (for example, hematuria or pneumonia). We can compare rates of correct enrollment and implementation of the pathway using reminders that trigger based on simple rules versus a reminder that triggers using simple logic from a phenotype estimator. A next step would be to use phenotype estimators to suggest enrollment in *all* of the clinical pathways within an institution and monitor outcomes.

Moving beyond clinical pathways, a longitudinal electronic medical record that has every patient encounter annotated with relevant phenotype variables would be easier to search and summarize, improving communication between healthcare providers and with patients.

As a long term goal, entire hospital IT systems could use phenotypes as a representation for all of their functions, including estimating staffing needs, scheduling consults, auto-populating records, displaying clinical decision support, managing followup care, monitoring population health, and supporting research projects.

8.1.4 Continuous learning and evaluation

The algorithms presented in this dissertation are batch algorithms, run on a large collection of historical data to build a single model for deployment. Ideally, the model would be able to learn continuously, based on new patient records, but also on explicit feedback and usage patterns. In addition to improving predictions, this feedback could be useful to monitor a system in continuous use in order to understand whether it makes systematic errors and taking steps (e.g., adding or removing anchors) to correct that. Feedback could also be actively solicited. Manual chart review is a laborious process, but if the computer system can ask a small number of relevant questions in real time as part of a physician’s regular workflow, we expect that it would be easier to obtain high quality labeled examples. This was the guiding principle behind the “disposition question” framework used to collect gold standard labels in Chapter 2, though that system asked its questions on a fixed rotation,

rather than choosing more smartly.

Another avenue for improvement through continued interaction is in the *inference* task. By asking a small number of questions, the computer could reduce uncertainty in its estimates of phenotype variables for an individual patient, in order to facilitate more accurate personalized care.

8.2 Open questions

8.2.1 Improved machine learning approaches

How should we learn models with latent variables? The method of moments offers a class of objectives that provide consistent parameter estimation and are more computationally tractable than the likelihood objective in the worst case. However, there may be other practical algorithms based on likelihood or approximations for both learning and inference in phenotyping models. Chapter 3 gives one approach to likelihood optimization, based on recent work by Mnih and Gregor (2014) on building deep generative models. We use a simple recognition model (independent logistic regression). Using a simple recognition model limits the possible tightness of the variational lower bound. However, adding complexity to the recognition model (e.g., using a multilayer neural network) could increase the non-concavity of the objective, making it harder to optimize in practice. In addition, controlling the variance in the gradients within the stochastic gradient ascent procedure is challenging. While Mnih and Gregor (2014) offer some tricks that worked in our setting, the problem is not fully solved. A surprising result in Section 4.5.2 is that noisy-or networks trained with stochastic variational inference using a recognition network succeeds in recovering the sources in the synthetic image data set (when provided with enough extra sources). Are there conditions under which we can prove that stochastic variational inference will succeed at parameter

recovery? How do these conditions relate to the structural conditions that make for tractable methods of moments algorithms (e.g., anchors or singly coupled quartets).

Within method of moments algorithms, there is still an open question of which moments are the most relevant and how to design algorithms that are robust to model misspecification.

8.2.2 Human computer interaction

Electronic phenotypes are another form of information, derived and synthesized from the patient's medical record. How is this information best presented to the physician to be maximally useful at the point of care? Is it important to be able to present the system's *reasoning* behind its phenotype estimations? What are effective ways of presenting a computer's thought process and what other factors are important to build trust in the system?

How do smart EMR systems integrate into medical culture? Physicians spend an increasingly large part of their work hours on documentation and other non-clinical related tasks. An EMR that makes workflows faster and more integrated could help reduce the amount of time spent on documentation and make more time for patient care, and would likely be welcomed. However, physicians rightly understand the high stakes involved in their jobs, and take the life and well-being of their patients very seriously. As such, they would likely be very hesitant to put too much trust in the outputs of a computer system that they do not fully understand. How do these factors balance with each other? And what are the factors for successful institutional adoption of smart EMR systems?

8.3 Conclusion

Electronic medical record phenotyping has the potential to transform health IT systems, making them smarter and more effective. We have made progress on building scalable machine

learning algorithms that can learn to estimate patient phenotypes in acute care settings, even in the absence of gold standard phenotyping labels. Next steps include testing the generalization of these methods in other institutions and settings, and their overall impact on patient outcomes. We look forward to a future where electronic medical record systems are smarter, providing physicians with relevant information and options to provide personalized care to their patients.

Appendix A

Appendix to Introduction

A.1 Assessing utility for more use cases

A.1.1 Search functionality: top K suggestions

An application with similar operating requirements to contextual displays are record-search capabilities, which could be used within a single patient record (searching over patient history) or across patients. Here the relevant metric would be determining whether relevant patients or notes appear near or at the top of the list (top K suggestions).

A.1.2 Screening tools: Sensitivity or Specificity

Screening tools can be designed to operate with high recall or specificity. High specificity screens are desired when a highly accurate test exists, but is expensive, invasive, or otherwise resource intensive to apply. In this setting, we may require a particular recall threshold (to avoid positive cases from being missed), and then evaluate classifiers based on their specificity. If there is a limited number of accurate tests that can be performed per day, then the specificity is fixed and classifiers can be compared by their sensitivity.

A.1.3 Scheduling and staffing: Calibration / weighted accuracy

Phenotypes can be used for prospective scheduling or staffing, for example of nursing staff, floor or ICU beds, or specialty consult services. For this use, it is less important to predict the need correctly for each individual patient, but rather to determine the need over an aggregation of an entire patient population (may be a small setting, such as a single emergency department or physician practice, or over a large population health setting). False positives may be more or less costly than false negatives, and thus each implementation would have to assess relevant weightings to determine a meaningful weighted accuracy.

A.1.4 Personalized treatment recommendations and decision making: Calibration

In order to make informed decisions and act on them, it is not only important for classifiers to have high accuracy, they also have to be properly calibrated. For example, if a decision-making tool reports a 30% chance of complication from a particular surgery, the physician and patient need to be able to trust the reported value in order to make a properly informed decision.

A.1.5 Scientific research: fixed specificity

For scientific research (e.g., comparative efficacy studies, genome wide association studies (GWAS), adverse effect reporting, drug repurposing), medical records are often mined for associations which are then followed up with controlled experiments for confirmation. This is similar to the screening tools setting with a fixed budget for followup, and we would like to increase the number of true positives or recall in this setting.

Appendix B

Appendix to Electronic Phenotyping with Anchors

B.1 Anchor Explorer – a human in the loop learning system

In practice, specifying anchors can be challenging. In order to specify anchors, one must have sufficient domain knowledge to evaluate whether a variable fits the definition of an anchor. To ease the process of eliciting anchors from domain experts, we built an interactive interface to allow domain experts to specify anchor variables and visualize the resulting model learned with those anchors. Figure B.1 shows a screenshot of the tool being used to specify anchors to identify HIV positive patients.

The interface is a general tool for specifying anchors and viewing the learned classifier. A user can add latent variables and specify anchors for them. After adding an anchor, the user can, in real time, update the learned model and view a ranked list of patients at the bottom of the screen. The ranking is generated according to the predicted likelihood of the

latent variable being positive according to the model built with the current set of anchors. For each patient, a short summary is presented for easy viewing, and selected patients can be viewed in more detail in the middle pane. Patients can be filtered according to three different criteria: view only patients with anchors (to judge whether the anchors are catching the correct subset of patients), view patients that have the most recently added anchor (to judge the incremental effect) and view patients without anchors (looking at a ranked list of these patients provides an idea of how well the learning algorithm has *generalized* beyond simply looking for patients that have the anchors).

After learning a model, the tool additionally *suggests* new anchors by showing the observations ranked by weights of a linear classifier learned with a penalty on the L1 norm of the weight vector. The L1 penalty encourages the learned classifier to use a minimal number of variables, effectively selecting highly informative observations. The user then uses clinical judgment to decide whether or not each suggestion would make a good anchor, e.g. by including the new observation and seeing the incremental effect on the ranking, or by viewing the newly anchored patients. The result is a simplified active learning workflow with a human-in-the-loop.

The ranking and filtering mechanisms provide feedback to the user, giving information about whether the model is being built reasonably or not. Users can sometimes specify anchors that are highly correlated for a specific latent variable, but not very specific. For example, at first glance, aspirin could mistakenly be taken for an anchor for a patient having a cardiac etiology to their ED visit. It is standard clinical practice to give aspirin to a patient when a cardiac etiology is suspected. However, aspirin is also given in patients with a suspected stroke. By reviewing a list of recently anchored patients, a user can quickly determine how specific an anchor is for a latent variable.

Anchors can be specified as words or phrases, and they are interpreted as queries on the free text portions of the medical record. Additionally, the interface allows for incorporation

of anchors according to standardized hierarchical ontologies. For example, medications are grouped by families according to First Databank's Enhanced Therapeutic Classification (ETC) hierarchy, and diagnosis codes are grouped according to the ICD9 hierarchy. For these hierarchical structures, including a parent as an anchor automatically adds all of its children as well.

In addition to specifying anchors, the tool is useful for performing fast interactive cohort selection, allowing the user to quickly learn classifiers to find members of a target population using the anchor approach. The learned classifiers can be exported as well for use in real-time decision applications. The tool is freely available for download at <http://clinicalml.org/>.

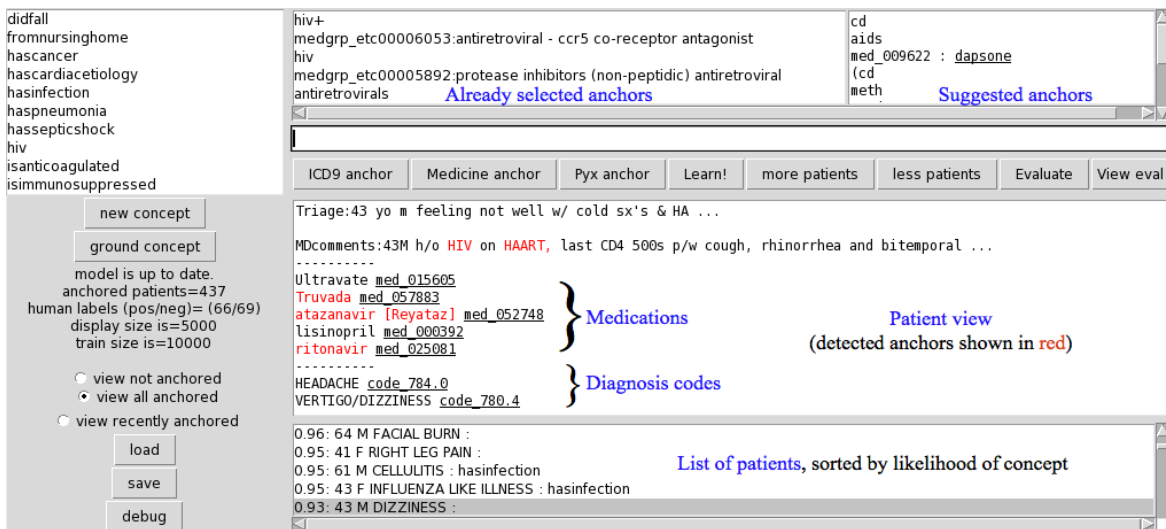


Figure B.1: A screenshot of the anchor elicitation tool using deidentified patient information. Conditional highlighting emphasizes the presence of previously specified anchors in a note so that the physician can determine quickly whether its usage is as expected.

B.2 Detailed methods

B.2.1 Building a phenotype library

The interactive tool uses a subset of the patients to make response times short after adding each anchor, in which the classifiers must be re-learned. For this step, 20,000 patients, selected randomly from the full patient population, were used. The final classifiers were then trained offline, using 200,000 patients from the population, chosen randomly from the full patient population after excluding patients for whom gold standard labels were available.

B.2.2 Text preprocessing

Deidentified free text was preprocessed using a modified version of NegEx (Chapman et al., 2001; Jernite et al., 2013a) and negated words were replaced by a new token (i.e. if the token “fever” was within the scope of a negation, it was transformed to a new token, “negfever”). A second step of preprocessing collected 1,500 significant bigrams and appended them to the text (i.e. the phrase “chest pain” was augmented to be “chest pain chest_pain” with a new token “chest_pain” representing the bigram).

The conditional independence assumption is difficult to guarantee in natural language data because language creates correlations between words and terms that are stronger than can be explained by underlying clinical conditions. To partially correct for this when learning with anchors, we censor a window of 3 words before and after an anchor to ensure that we are not simply learning local sentence structure rather than the underlying phenotypes. When the anchor is a bigram, this has the effect of ensuring that the individual components of the bigram (e.g. chest and pain for chest_pain) do not get undo weight.

B.2.3 Machine Learning

Logistic regression classifiers were trained using Scikit-learn (Pedregosa et al., 2011) using L2 regularization and 4-fold cross validation to choose appropriate regularization constants. The constants for C were chosen from $(10^{-6}, 10^{-5}, \dots, 10^5)$. Intercept scaling was set to 100. Supervised classifiers were evaluated with 4-fold cross-validation, thus the labeled examples were divided into 4 equal parts. A supervised classifier was trained using 5000 examples from one part and then tested on the remaining 3/4ths of the labeled data. This was repeated for all four folds and we report the min and max values. Error bars on AUC values in were computed with bootstrap sampling from the test set.

B.2.4 Phenotype Evaluation

For the anchor-based approach, ties between patients who both have anchors present were broken as follows: (a) patients with more anchors are ranked above patients with fewer anchors, (b) if both patients have the same number of anchors, the patient with the higher score given by the logistic regression classifier is ranked higher.

B.2.5 Measuring Influence

To illustrate how the classifiers change over time, we calculate a relative influence measure as follows: For every patient and every data type, we first compute an unnormalized influence score by taking the sum of weights associated with positive observations in that data type. The influence of a data type on a patient prediction is then computed by taking the absolute values of the unnormalized influence scores and dividing by the total, so that all of the values are non-negative and sum to one. The influence of a data type on predictions in an entire patient population is computed as the average influence of the data type on predictions for each of the patients. In addition to influence, at each time step, we identify the positive

features which have the largest gain in weight since the previous time step.

Appendix C

Appendix to Clinical Tagging with Joint Probabilistic Models

C.1 Text processing

We apply negation detection to the free-text section using “negex” rules (Chapman et al., 2001) with some manual adaptations appropriate for Emergency department notes (Jernite et al., 2013a), and replace common bigrams with a single token (e.g. “chest pain” is replaced by “chest_pain”. the patient record). Negated terms are then added as additional vocabulary words.

The following words are indicators of the beginning of a negation: `no`, `not`, `denies`, `without`, `non`, `unable`.

The token `-` is treated as a special negation word whose scope only includes the word that follows it.

The following tokens stop a negation: `.` ; `[` `-` `newline` `+` `but` `and` `pt` `except` `reports` `alert` `complains` `has` `states` `secondary` `per` `did` `aox3`.

C.2 Parameter settings for learning with Neural Variational Inference and Learning (Mnih and Gregor, 2014)

We use the signal-centering and normalization described in the paper as well as input-dependent baselines. The input dependent baselines use a two-layer neural network with 100 hidden units and tanh activations. Learning rate is set to 0.0001. 10 samples are used to estimate each gradient. When θ is initialized with method of moments, we have 50 epochs of “burn-in” where θ is held fixed and only ϕ is optimized. The learning rate of θ is set to 1/5th of that of ϕ . π parameters and the failure probabilities of the anchors are initialized using the estimated noise rates and never optimized. The code is implemented in Torch and RMSprop is used for optimization. Failure probabilities in θ are mapped using a sigmoid function (i.e., $f_{i,j} = \sigma(\theta_{i,j})$) to allow for continuous optimization over an unbounded space. We experimented with different values of L2 regularization (weight decay) in the recognition model, in the range $\{0, 0.1, 0.01, 0.001\}$ and chose the value that gave the best heldout-anchor performance. Parameters of ϕ are initialized uniformly at random between $[-0.1, 0.1]$.

C.3 Semi-supervised objective details

In this appendix, we expand the objective which was written in compressed notation in Section 3.4.1.

The parameters to be optimized are:

- $\theta \in [0, 1]^{n \times m + n + m}$: Parameters of the generative model, consisting of $n \times m$ failure probabilities ($f_{i,j}$), n prior probabilities (π_i), and m leak probabilities (l_j).
- $\phi \in \mathbb{R}^{n \times (m+1)}$: Parameters of n independent logistic regression models, one for each tag.

- $\phi'_0 \in \mathbb{R}^n$: Additional bias terms that are introduced to account for the difference between the predictions of the recognition model, which predicts the tags, and the desired semi-supervised objective which predicts the *anchors*.

For the a single data point, x is the binary feature vector. In practice, we center the inputs to the q model (the P model cannot support centering since the generative model is defined for binary variables) and pad with a single 1 to allow for a bias term. \bar{x} is a centered and padded copy of x . \tilde{x} is a copy of \bar{x} , with the values of the anchors set to 0. a is a vector containing the binary values of the anchors from x .

The likelihood of the generative model is then:

$$P(x, y; \theta) \equiv \left(\prod_{i=1}^n \pi_i^{y_i} (1 - \pi_i)^{1-y_i} \right) \prod_{j=1}^m \left(x_j \left(1 - (1 - l_j) \prod_i f_{i,j}^{y_i} \right) + (1 - x_j)(1 - l_j) \prod_{i=1}^n f_{i,j}^{y_i} \right) \quad (\text{C.1})$$

The recognition model consists of n independent logistic regression models.

$$q(y|\bar{x}; \phi) = \prod_{i=1}^n (y_i \sigma(\phi_i \cdot \bar{x}) + (1 - y_i)(1 - \sigma(\phi_i \cdot \bar{x}))), \quad (\text{C.2})$$

where σ is the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$.

Let $\lambda > 0$ be a hyperparameter specifying the trade-off between the lower bound on the likelihood and the semi-supervised objective term. The final objective, for a collection of N patient records, indexed by p , is to maximize:

$$\sum_{p=1}^N \left(E_{y \sim q(y|\bar{x}^{(p)})} [\log P(x^{(p)}, y; \theta) - \log q(y|\bar{x}^{(p)})] + \lambda \sum_{i=1}^n \left[a_i^{(p)} \log \sigma(\phi_i \cdot \tilde{x}^{(p)} + \phi'_{0i}) + (1 - a_i^{(p)}) \log(1 - \sigma(\phi_i \cdot \tilde{x}^{(p)} + \phi'_{0i})) \right] \right). \quad (\text{C.3})$$

C.4 Leak probabilities

Leak probabilities are calculated to account for the difference between the actual observed counts and those predicted by the model. If all of the failure probabilities are known, then the marginal probability of an observation predicted by the model can be calculated with the Quickscore equation (Heckerman, 1990).

$$P(X_j = 0) = (1 - l_j) \prod_{i=1}^n (1 - \pi_i + \pi_i f_{i,j}). \quad (\text{C.4})$$

Thus, we can solve:

$$\hat{l}_j = 1 - \frac{\hat{P}(X_j = 0)}{\prod_{i=1}^n (1 - \pi_i + \pi_i \hat{f}_{i,j})}. \quad (\text{C.5})$$

This assumes that we have estimates for π_i , but that is no different from assuming that we have estimates of the noise rates $P(Y_i|A_i)$, since $\pi_i = \sum_{a_i} P(a_i)P(Y_i = 1|a_i)$, where $P(a_i)$ can be estimated from counts.

C.5 Explicit matrix version of Equation 3.9

$$\vec{P}(X_j|A_i) = \begin{bmatrix} P(X_j = 0|A_i = 0) \\ P(X_j = 0|A_i = 1) \\ P(X_j = 1|A_i = 0) \\ P(X_j = 1|A_i = 1) \end{bmatrix} \quad (\text{C.6})$$

$$R = \begin{bmatrix} P(A_i = 0|Y_i = 0) & P(A_i = 0|Y_i = 1) & 0 & 0 \\ P(A_i = 1|Y_i = 0) & P(A_i = 1|Y_i = 1) & 0 & 0 \\ 0 & 0 & P(A_i = 0|Y_i = 0) & P(A_i = 0|Y_i = 1) \\ 0 & 0 & P(A_i = 1|Y_i = 0) & P(A_i = 1|Y_i = 1) \end{bmatrix} \quad (\text{C.7})$$

$$\vec{P}(X_j|Y_i) = \begin{bmatrix} P(X_j = 0|Y_i = 0) \\ P(X_j = 0|Y_i = 1) \\ P(X_j = 1|Y_i = 0) \\ P(X_j = 1|Y_i = 1) \end{bmatrix}. \quad (\text{C.8})$$

C.6 Held out anchor inference

To perform the held out anchor inference (described in Section 3.4.4), we use the fact that conditioned on some feature vector X' (in this case, some of the anchors are held out so

it is not the full vector X), we have:

$$\begin{aligned} P(A_i|X') &= \sum_{y_i \in \{0,1\}} P(y_i|X')P(A_i|Y_i = y_i, X') \\ &= \sum_{y_i \in \{0,1\}} P(y_i|X')P(A_i|Y_i = y_i). \end{aligned}$$

We assume that the corruption rate, $P(A_i|Y_i)$, is known, and estimate $P(y_i|X')$ with Gibbs sampling.

C.7 Held out tag inference

Exact inference is possible in the held out tag prediction task. For a single data point, we have a feature vector X . Assume without loss of generality that the first k tags known to be positive, $y_1 = \dots = y_k = 1$ and the final $n - k$ tags are unknown. Let U refer to the unknown tags.

Of the unknown tags, we know that by the design of the task, only one is on and the rest are off. We can condition on the sum of all of the Y variables being $k + 1$, and we know that U has to be an indicator vector (i.e., one index on and the rest off).

We would like to calculate the likelihood that $U_i = 1$ (i.e., the i th unknown tag is on). Let U_{-i} be all of the unknown tags other than U_i . The likelihood is calculated as follows:

$$\begin{aligned}
P(U_i = 1 | X, Y_{1:k} = 1, \sum y = k + 1) &= P(U_i = 1, U_{-1} = 0 | X, Y_{1:k} = 1, \sum y = k + 1) \\
&= \frac{P(X, U_i = 1, U_{-1} = 0, Y_{1:k} = 1, \sum y = k + 1)}{P(X, Y_{1:k} = 1 | \sum y = k + 1)} \\
&= \frac{P(X, U_i = 1, U_{-1} = 0, Y_{1:k} = 1, \sum y = k + 1)}{\sum_{u \in |U|} P(X, Y_{1:k} = 1, u | \sum y = k + 1)}.
\end{aligned}$$

The first equality uses the fact that U is an indicator vector. The second equality is from the definition of conditioning. The third line marginalizes over unknown values of u in the denominator. Note that the final line uses only complete likelihoods, so each term can be calculated efficiently using Equation C.1. The sum in the denominator is over the set of indicator vectors of size $n - k$, so it can be computed efficiently as well.

C.8 Optimization course

In Section 3.4.1 we discuss how the likelihood objective is not aligned with the held out tag prediction task. Figure C.1 shows that the semi-supervised objective remedies this situation, displaying how the likelihood objective and heldout tag predictions improve from the initialization baseline as optimization on the semi-supervised objective is run.

C.9 Monte Carlo EM

When running EM, we optimize the variational lower bound on the data likelihood:

$$\log P(X; \theta) \geq \mathcal{L}(q, \theta) = E_{y \sim q} [\log P(x, y; \theta) - \log P(y)]$$

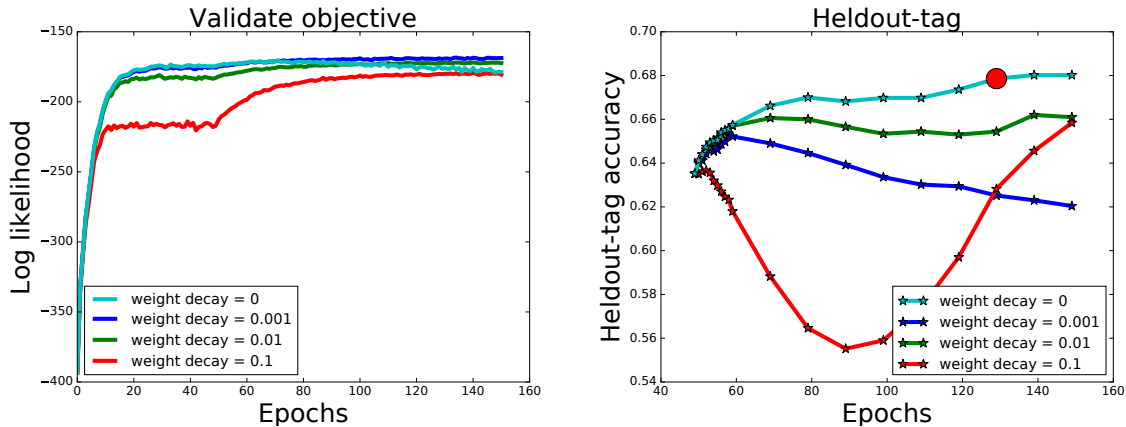


Figure C.1: Improvement of the likelihood-based objective (left) and heldout tag (right) as optimization progresses for different values of L2 regularization (weight decay) in the recognition model. The model chosen by the model selection procedure (Section 3.4.4) is marked with a large red dot. The first 50 epochs are a burn-in period where only the recognition model changes.

The EM algorithm optimizes this bound with a coordinate ascent, alternating between E-steps which improve q and M-steps which improve θ . Usually both the E-step and M-steps are maximization steps, but incomplete M-steps which only improve θ in every M-step also leads to monotonic improvement of $\log(P(X; \theta))$ with every step. In this section, we describe a variant of Monte Carlo EM which is useful for this model using Gibbs sampling to approximate the E-step and a custom M-step which is guaranteed to improve the variational lower bound at every step. We hold the distribution $P(\mathcal{Y})$ fixed and only optimize the failure and leak probabilities. For these purposes, the leak probabilities can be treated as simply failure probabilities of an extra latent variable whose value is always 1.

C.9.0.1 Outer E-step

For the E-step, we use the Gibbs sampling procedure described in 4.3 of Wang et al. (2014).

Algorithm 9 Alternative generative model with auxiliary A variables.

$Y \sim P(Y)$
for j in $1..m$ **do**
 $A_j \sim P(A_j = k|Y)$
 $X_j = A_j \leq n$
end for
 $P(A_j = k|Y) = \begin{cases} (1 - f_{k,j}) \prod_{i=0}^{k-1} f_{i,j}^{y_i} & k \leq n \\ 1 - \prod_{i=0}^{k-1} f_{i,j}^{y_i} & k = n + 1 \end{cases}$

C.9.0.2 Outer M-step

The M-step consists of a coordinate step guaranteed to improve θ for a fixed q . $P(x, y; \theta)$ is a fully observed model, but optimizing θ has no closed form. Instead we introduce m *auxiliary variables* $A \in [0, n + 1]^m$, and adopt the generative model described in Algorithm 9 which equivalently describes the fully-observed noisy-or model (i.e. $\sum_a P(X, Y, a; \theta) = P_{\text{noisy or}}(X, Y; \theta)$). In this new expanded model, we can perform a single E-M step with respect to the latent variable A in closed form.

inner E-step:

$$P(A_j = k | X_j, Y) = \begin{cases} 1 & X_j = 0 \cap k = n + 1 \\ 0 & X_j = 0 \cap k \neq n + 1 \\ 0 & X_j = 1 \cap k = n + 1 \\ 0 & X_j = 1 \cap k \leq n \cap Y_k = 0 \\ \propto \prod_{i=0}^{k-1} f_{i,j}^{y_i} (1 - f_{k,j}) & X_j = 1 \cap k \leq n \cap Y_k = 1 \end{cases} \quad (\text{C.9})$$

inner M-step:

$$f_{i,j} = 1 - \frac{\text{count}(A_j = i)}{\text{count}(A_j \leq i)} \quad (\text{C.10})$$

Appendix D

Appendix to Parameter Learning in Diagnosis Networks

D.1 Derivations

D.1.1 Noisy-or parameters from tensor decompositions

Here we derive the estimators of the noisy-or parameters used in Equations 4.5-4.8.

$$\begin{aligned}\frac{P(a = 0|Y_i = 1)}{P(a = 0|Y_i = 0)} &= \frac{f_{i,a} \prod_{k \in -i} (1 - p_k + p_k f_{k,a})}{\prod_{k \in -i} (1 - p_k + p_k f_{k,a})} \\ &= f_{i,a}.\end{aligned}$$

Similarly for b,c.

D.1.2 Estimating $\pi_{i|\bar{X}_j}$ as the root of a quadratic equation

For notational simplicity, we rewrite Equation 4.11 with the following name substitutions:

$$Z \equiv \text{CPMI}(A, B|X_j), \quad x \equiv \pi_{i|\bar{X}_j}.$$

$$\begin{aligned}
Z &= \frac{(1 - x + xf_{i,A}f_{i,B})}{(1 - x + xf_{i,A})(1 - x + xf_{i,B})} \\
0 &= \frac{(1 - x + xf_{i,A}f_{i,B})}{(1 - x + xf_{i,A})(1 - x + xf_{i,B})} - Z \\
0 &= (1 - x + xf_{i,A}f_{i,B}) - Z(1 - x + xf_{i,A})(1 - x + xf_{i,B}) \\
0 &= (1 - x + xf_{i,A}f_{i,B}) - Z(1 - 2x + x^2 + x^2f_{i,A}f_{i,B} + xf_{i,B} - x^2f_{i,B} + xf_{i,A} - x^2f_{i,A}) \\
0 &= -Z(f_{i,A}f_{i,B} - f_{i,A} - f_{i,B} + 1)x^2 + (-1 + f_{i,A}f_{i,B}) - Z(-2 + f_{i,B} + f_{i,A})x + 1 - Z \\
0 &= Z(1 - f_{i,A})(1 - f_{i,B})x^2 + (Z(-2 + f_{i,B} + f_{i,A}) - (f_{i,A}f_{i,B} - 1))x + (Z - 1)
\end{aligned}$$

Solving for the roots of the right hand side gives the value of $\pi_{i|\bar{X}_j}$.

D.1.3 Estimating $f_{i,j}$ when $\pi_{i|\bar{X}_j}$ is known

We start with the definition of $\pi_{i|\bar{X}_j}$:

$$\pi_{i|\bar{X}_j} = P(Y_i = 1 | X_j = 0) \tag{D.1}$$

$$\pi_{i|\bar{X}_j} = \frac{P(Y_i = 1, X_j = 0)}{P(X_j = 0)} \tag{D.2}$$

$$\pi_{i|\bar{X}_j} = \frac{\pi f_{i,j} \prod_{i' \neq i} (1 - \pi'_i + \pi'_i f_{i',j})}{\prod_{i'} (1 - \pi'_i + \pi'_i f_{i',j})} \tag{D.3}$$

$$\pi_{i|\bar{X}_j} = \frac{\pi f_{i,j}}{(1 - \pi_i + \pi_i f_{i,j})}. \tag{D.4}$$

Rearranging, we get:

$$f_{i,j} = \frac{(1 - \pi_i)}{\pi_i} \frac{\pi_{i|\bar{X}_j}}{(1 - \pi_{i|\bar{X}_j})},$$

as in Equation 4.13.

D.1.4 Identifiability of $\pi_{i|\bar{X}}$

Recall, we have $\pi_{i|\bar{X}}$ as the roots of an quadratic:

$$\pi_{i|\bar{X}} = \text{Roots}(f(x)) \tag{D.5}$$

where $f(x)$ has the form:

$$\begin{aligned} f(x) &= \text{CPMI}(A, B|X)(1 - f_{i,A})(1 - f_{i,B})x^2 \\ &\quad + (\text{CPMI}(A, B|X)(f_{i,A} + f_{i,B} - 2) - (f_{i,A}f_{i,B} - 1))x \\ &\quad + \text{CPMI}(A, B|X) - 1 \end{aligned}$$

Substituting for CPMI using Equation 4.11, performing some tedious algebraic simplification, and solving for the critical point of the function, we get:

$$f'(x) = 0 \rightarrow x = \frac{1(1 - \pi_{i|\bar{X}}^2(1 - f_{i,A}f_{i,B}))}{2(1 - \pi_{i|\bar{X}}(1 - f_{i,A}f_{i,B}))} \geq 0.5 \tag{D.6}$$

Where the \geq comes from the fact that $\pi_{i|\bar{X}}, f_{i,A}, f_{i,B}$ all lie in $[0, 1]$. Since f is quadratic, its roots must lie on either side of its critical point, thus one of the roots is at or above 0.5.

The last step is to show that $\pi_i \leq 0.5$ implies that $\pi_{i|\bar{X}} \leq 0.5$.

We show that $\pi_{i|\bar{X}} \leq \pi_i$,

$$\begin{aligned}
\pi_{i|\bar{X}} = P(Y_i = 1|X = 0) &= \frac{P(X = 0|Y_i = 1)P(Y = 1)}{P(X = 0)} \\
&= \frac{P(X = 0|Y_i = 1)}{P(X = 0)}\pi_i \\
&= \frac{f_i \prod_{i' \neq i} (1 - \pi_{i'} + \pi_{i'} f_{i'})}{\prod_{i'} (1 - \pi_{i'} + \pi_{i'} f_{i'})} \pi_i \\
&= \frac{f_i}{1 - \pi_i + \pi_i f_i} \pi_i \leq \pi_i,
\end{aligned}$$

which completes the proof.

D.2 Parameter estimation in noisy networks – fully observed setting

Proposition 1. *Maximum likelihood estimation of noisy-or parameters is concave in the log of the failure probabilities in the fully observed setting.*

Proof. Let $\theta_{i,j} = \log f_{i,j}$, we will do the optimization in the θ parameters. $f_{i,j}$ is constrained to lie in $[0, 1]$, therefore $\theta_{i,j} \in (-\infty, 0)$.

The optimization has the form:

$$\operatorname{argmax}_{\theta \leq 0} f(\theta) \tag{D.7}$$

We now show that $f(\theta)$ is jointly-concave in θ .

$$\begin{aligned}
f(\theta) &= \sum_n \log P(X_j^{(n)} | Y^{(n)}; \theta) \\
&= \sum_n \mathbf{1}[X_j = 0] \log \prod_i f_{i,j}^{y_i} + \mathbf{1}[X_j = 1] \log(1 - \prod_i f_{i,j}^{y_i}) \\
&= \sum_n \mathbf{1}[X_j = 0] \sum_i y_i \theta_{i,j} + \mathbf{1}[X_j = 1] \log(1 - \exp \sum_i y_i \theta_{i,j})
\end{aligned}$$

The first term is linear in θ_i , and therefore concave.

The second term takes a bit more work:

$$g_n(\theta) = \mathbf{1}[X_j = 1] \log(1 - \exp \sum_i y_i \theta_{i,j})$$

1. $(1 - \exp \sum_i y_i \theta_{i,j})$ is concave (since \exp is convex) and non-negative (since $\theta \leq 0$).
2. Every concave function that is nonnegative on its domain is log-concave.
3. Therefore $\mathbf{1}[X_j = 1] \log(1 - \exp \sum_i y_i \theta_{i,j})$ is concave.

The sum of two concave functions is concave. Therefore $f(\theta)$ is concave. □

D.3 Hardness results

D.3.1 Exact inference in quartet-learnable networks NP-hard

Quartet-learnable networks The networks that can be learned in polynomial time using the algorithm from Jernite et al. (2013b) must have failure probabilities bounded away from 0 and 1 by a constant and singly-coupled quartets for every latent variable.

Reduction from #2SATX #2SATX (also referred to as #MONOTONE-2SAT) is a variant of 2SAT variant with clauses $C = \{c_1 \dots c_n\}$ and variables $V = \{v_1 \dots v_m\}$. In #2SATX, every clause c_j is a disjunction of 2 non-negated variables from V . The decision problem 2SATX is trivial. Since none of the variables are negated, the problem is trivially satisfiable by setting all of the variables to True. However, the #2SATX problem asks to count the number of satisfying assignments and is known to be #P hard (Provan and Ball, 1983).

It has previously been shown that #2SATX can be reduced to exact inference in a bipartite noisy-or network (B2NOI) with failure probabilities of 0.

(#2SATX \subset B2NOI (Cooper, 1987)) Construct a bipartite graph with one latent parent for every variable in V and one observation for every clause in C . Place a directed edge from v_i to c_j if variable i appears in clause j . Set all the prior probabilities to 0.5 and all failure probabilities to 0. The number of satisfying assignments can be computed as $2^m P(C_1 = 1, C_2 = 1, \dots C_n = 1)$.

Here we show that exact inference in a quartet-learnable bipartite noisy-or network (B2QNOI) is similarly hard.

(#2SATX \subset B2QNOI) Construct a bipartite graph with one latent parent for every variable in V and one observation for every clause in C . Place a directed edge from v_i to c_j if variable i appears in clause j . Additionally, add 3 new children observations for every variable in V to ensure that the network is strongly quartet learnable. These variables will not be included in the inference query, but they ensure that the network satisfies the quartet learnable property¹. Set all the prior probabilities to 1/2.

In a satisfying assignment, some of the clauses may be double-satisfied (both parents on) or singly-satisfied (one parent on). Given a satisfying assignment with k singly-satisfied clauses and $n - k$ double-satisfied clauses, if the failure probabilities in the noisy-or network

¹If you want to make sure that all “observed” variables appear in the inference query, you can set the failures for the three anchor variables to be 1/2 and the set the prior probabilities to 8/9. After conditioning on the anchor variables being off, the prior probabilities become 1/2 and the rest of the problem is unchanged.

are all equal to f , then the probability that all of the observed variables corresponding to the clauses are “on” is:

$$P(C_1 = 1, C_2 = 1, \dots, C_n = 1 | \# \text{ singly satisfied} = k) = (1 - f)^k (1 - f^2)^{n-k} \quad (\text{D.8})$$

Let α_k be the number of satisfying assignments with k singly-satisfied clauses.

$$P(C_1 = 1, C_2 = 1, \dots, C_n = 1; f) = 2^{-m} \sum_{k=0}^n \alpha_k (1 - f)^k (1 - f^2)^{n-k} \quad (\text{D.9})$$

We can evaluate $P(C_1 = 1, C_2 = 1, \dots, C_n = 1; f)$ at $n + 1$ different values of f and solve for $\alpha_0 \dots \alpha_n$ with matrix inversion. The sum $\sum_{k=0}^n \alpha_k$ is the number of satisfying assignments, which completes the reduction. This does not require that f be close to 0 or 1 and in fact applies even if we restrict ourselves to networks where all the priors are the same and all the failures are drawn from a finite set as the networks that are learnable by the method of Kearns and Mansour (1998).

D.4 Preliminary results

Throughout the sample complexity proofs, we will need to be able to bound the error on a fraction and on the roots of a polynomial given the uncertainty on their terms. Lemmas 6 and 7 respectively give such bounds.

Lemma 6. *Let $\tilde{a} \in [a - \eta, a + \eta]$ and $\tilde{b} \in [b - \epsilon, b + \epsilon]$. If $\beta < b - \epsilon$ and $a + \eta < A$, then $\frac{\tilde{a}}{\tilde{b}} \in [\frac{a}{b} - (\frac{A}{\beta^2}\epsilon + \frac{1}{\beta}\eta), \frac{a}{b} + (\frac{A}{\beta^2}\epsilon + \frac{1}{\beta}\eta)]$*

Lemma 7. *Let $P(X) = aX^2 + bX + c$ and $\Delta = b^2 - 4ac$. Let (x_1, x_2) be the roots of P , and suppose $(\tilde{a}, \tilde{b}, \tilde{c})$ are estimates of (a, b, c) with an error bounded by ϵ . Suppose $\exists d, 0 < d < \Delta$ and $\exists k > 0, -k < (\tilde{x}_1, \tilde{x}_2) < k$. Then the error on (x_1, x_2) is bounded by*

$$|(x_1, x_2) - (\tilde{x}_1, \tilde{x}_2)| \leq \frac{2(1+k+k^2)}{\sqrt{d}}\epsilon.$$

Proof of Lemma 6. We have,

$$\begin{aligned} \frac{a}{b+\epsilon} - \frac{\eta}{b-\epsilon} &\leq \frac{\tilde{a}}{\tilde{b}} \leq \frac{a}{b-\epsilon} + \frac{\eta}{b-\epsilon} \\ \frac{a}{b} - \frac{a\epsilon}{b(b+\epsilon)} - \frac{\eta}{b-\epsilon} &\leq \frac{\tilde{a}}{\tilde{b}} \leq \frac{a}{b} + \frac{a\epsilon}{b(b-\epsilon)} + \frac{\eta}{b-\epsilon} \\ \frac{a}{b} - \left(\frac{a\epsilon}{b(b-\epsilon)} + \frac{\eta}{b-\epsilon}\right) &\leq \frac{\tilde{a}}{\tilde{b}} \leq \frac{a}{b} + \left(\frac{a\epsilon}{b(b-\epsilon)} + \frac{\eta}{b-\epsilon}\right) \\ \frac{a}{b} - \left(\frac{A\epsilon}{\beta^2} + \frac{\eta}{\beta}\right) &\leq \frac{\tilde{a}}{\tilde{b}} \leq \frac{a}{b} + \left(\frac{A\epsilon}{\beta^s} + \frac{\eta}{\beta}\right) \end{aligned}$$

Proof of Lemma 7. Let $K = 1 + k + k^2$, then $P(\tilde{x}_1) - K\epsilon < \tilde{P}(\tilde{x}_1) = 0 < P(\tilde{x}_1) + K\epsilon$, hence the estimated \tilde{x}_1 is between the roots of $P - K\epsilon$ and $P + K\epsilon$. Moreover, since $\sqrt{1+x} \in (1, 1+x)$:

$$\begin{aligned} \frac{b + \sqrt{b^2 - 4a(c + K\epsilon)}}{2a} &= \frac{b + \sqrt{\Delta} \sqrt{1 - \frac{4aK}{\Delta}\epsilon}}{2a} \geq \frac{b + \sqrt{\Delta}}{2a} - \frac{2K}{\sqrt{\Delta}}\epsilon \\ \frac{b + \sqrt{b^2 - 4a(c - K\epsilon)}}{2a} &= \frac{b + \sqrt{\Delta} \sqrt{1 + \frac{4aK}{\Delta}\epsilon}}{2a} \leq \frac{b + \sqrt{\Delta}}{2a} + \frac{2K}{\sqrt{\Delta}}\epsilon \end{aligned}$$

Hence $x_1 - \frac{2K}{\sqrt{\Delta}}\epsilon \leq \tilde{x}_1 \leq x_1 + \frac{2K}{\sqrt{\Delta}}\epsilon$. Similarly, $x_2 - \frac{2K}{\sqrt{\Delta}}\epsilon \leq \tilde{x}_2 \leq x_2 + \frac{2K}{\sqrt{\Delta}}\epsilon$.

D.5 Sample complexity of learning with known structure

The parameter learning algorithm failure probabilities using tensor decomposition and extending steps. Hence we need to know the sensitivity of the algorithm to errors in either of these steps.

Lemma 8. (*Parameter error from decomposition*) If the error on the moments is bounded by ϵ , then the error on the parameters we obtain from the tensor decomposition of third order moments is bounded by $\frac{5000 \times 1050 \epsilon}{f_{\min}^{18} p_{\min}^{12} l_{\max}^{24} (1 - f_{\max})^3}$.

Proof is in Section D.7.1.

Lemma 9. (*Parameter error from extending step*) If the moments and parameters are known to within an error of ϵ , the error on the failure probability we obtain from the extending step is bounded by $\frac{11648 \epsilon}{(1 - f_{\max})^3 n_{\min}^4 p_{\min}}$.

Proof is in Section D.7.2.

Before introducing the subtraction step, Lemma 10 gives a bound on the parameter error.

Lemma 10. (*Parameter error at depth 0*) After N samples, the additive error on any of the parameters at depth 0 $\epsilon_0(N)$ is bounded with probability $1 - \delta$ by:

$$\epsilon_0(N) \leq \frac{5000 \times 1050 \times 11648 \times \sqrt{\ln\left(\frac{2m}{\delta}\right)}}{f_{\min}^{18} (1 - f_{\max})^6 l_{\max}^{28} p_{\min}^{13}} \frac{1}{\sqrt{N}}. \quad (\text{D.10})$$

Proof is in Section D.7.3.

The subtraction step introduces a multiplicative error with every subtraction. Let M_0 be a lower bound on the likelihood of an observation being off.

Lemma 11. (*Error propagation from subtraction*) If the additive error on the negative moments of an observed quartet \mathcal{C} and on the parameters of the l latent variables (X_1, \dots, X_l) , which we want to remove from \mathcal{C} is bounded by ϵ , then the error on the subtracted-off moments is bounded by $M_0^6 \times 6(l + 2)\epsilon$.

Proof is in Section D.7.4.

The final sample complexity of the parameter learning procedure is then given by Theorem 1.

Theorem 1 (Learning with known structure). *Suppose a network with n latent parents and m observed variables is triplet-learnable at depth d . Let M_0 be the minimum marginal probability of an observation being off and let $\pi_{max} \leq 0.5, \pi_{min}$ be bounds on the prior probabilities, f_{max}, f_{min} be bounds on the failures and l_{max} be an upper bound on the leak parameters. The additive error on any of the parameters, ϵ is bounded with probability $1 - \delta$ by:*

$$\epsilon(N) = O \left(\left(\frac{nM_0^6 \sqrt{\ln(\frac{2m}{\delta})}}{f_{min}^{18} (1 - f_{max})^6 l_{max}^{28} \pi_{min}^{13} \sqrt{N}} \frac{1}{\sqrt{N}} \right)^{2d} \right) \quad (4.16)$$

The proof comes from applying Lemmas D.7.3 and D.7.4 to bound the multiplicative error propagation which each round of the algorithm.

D.6 Sample complexity of learning with unknown structure

Learning with unknown structure adds the additional complexity of ensuring that the quartet tests are all correct. Lemma 12 described the conditions under which the rank test succeeds, and Lemma 13 gives the conditions under which the extending step succeeds.

Lemma 12. *(Rank test) If the model is (ζ) -rank-testable and the estimation error on the fourth order moments of (a, b, c, d) is bounded by $\epsilon \leq \zeta^4/2^{16}$, then the magnitude of the third eigenvalue of every 4×4 unfolding of joint distribution is smaller than $\zeta/2$ if and only if (a, b, c, d) is singly coupled.*

Proof in Section D.7.5.

Lemma 13. (*Extending test*) If the maximum estimation error ϵ on all of the moments up to third order is such that $1040\epsilon < l_{max}^4 p_{min}(1 - f_{max}^4)$, (a, b, c) share a parent if and only if:

$$PMI(a, b) - CPMI(a, b|x) \geq \frac{p_{min}(1 - f_{max}^3)(1 - f_{max}^2)}{40} \quad (\text{D.11})$$

Proof in Section D.7.6.

Lemma 14 gives the conditions under which the structure learning procedure succeeds for the subnetwork learnable at depth 0.

Lemma 14. (*Structure learning at depth 0*) If a network with m observed variables is strongly quartet-learnable and ζ -rank-testable, then its structure can be learned with probability $(1 - \delta)$ with polynomial number of samples N_S :

$$N_S = 3 \max\left(\frac{4 \times 2^{32}}{\zeta^8}, \frac{3 \times 1040^2}{l_{max}^8 p_{min}^2 (1 - f_{max})^8}\right) \ln\left(\frac{2m}{\delta}\right)$$

Proof in Section D.7.7.

Theorem 2 (Learning with unknown structure). *Suppose a network with n latent variables and m observed variables is quartet-learnable at depth d and is ζ -rank-testable. Let M_0 be the minimum marginal probability of an observation being off; $\pi_{max} \leq 0.5$, π_{min} be bounds on the prior probabilities; f_{max} , and f_{min} be bounds on the failures and l_{max} be an upper bound on the leak probabilities. Its structure can be learned with probability $(1 - \delta)$ with N_S samples, where:*

$$N_S = O\left(\left(\frac{nM_0^6}{f_{min}^{18}(1 - f_{max})^6 l_{max}^{28} p_{min}^{13}}\right)^{2d} \times \max\left(\frac{1}{\zeta^8}, \frac{1}{l_{max}^8 p_{min}^2 (1 - f_{max})^8}\right) \ln\left(\frac{2m}{\delta}\right)\right) \quad (4.18)$$

Proof comes from applying the error propagation bound introduced by the subtracting step (Lemma 11) to the accuracy required for correct structure learning given by Lemma 14.

D.7 Proofs

D.7.1 Proof of Lemma 8

We seek a bound on the errors of $f_{X,u}$ and p_X for a triplet (u, v, w) singly coupled by latent variable X as a function of ϵ . Assume that matrices $X_1 = P(v, w, u = 0)$ and $X_2 = P(v, w, u = 1)$ are known to some element-wise error ϵ . Let $Y_2 = X_2 X_1^{-1}$ and let (λ_1, λ_2) be the eigenvalues of Y_2 .

From Algorithm 2 (and Equations 4.5-4.8) we have that:

$$f_{X,u} = \frac{1 + \lambda_1}{1 + \lambda_2}.$$

We first bound the error on X_1^{-1} . If $X_1 = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, then we have $X_1^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$. Moreover, $(ad - bc) = p_X(1 - p_X)f_{X,u}(1 - f_{X,v})(1 - f_{X,w})n_1^2 n_2 n_3 \geq f_{min}^3 p_{min}^2 l_{max}^4$, so if $\epsilon < \frac{f_{min}^3 p_{min}^2 l_{max}^4}{2}$, we get using lemma 6 with $A = 1, \beta = \frac{f_{min}^3 p_{min}^2 l_{max}^4}{2}$ that the element-wise error on X_1^{-1} is at most $\frac{25\epsilon}{f_{min}^6 p_{min}^4 l_{max}^8}$. Hence the error on the terms of Y_2 is bounded by $\frac{6 \times 25\epsilon}{f_{min}^6 p_{min}^4 l_{max}^8}$.

(λ_1, λ_2) are the roots of the polynomial $P(X) = X^2 - Tr(Y_2)X + Det(Y_2)$ and $\Delta = |\lambda_1 - \lambda_2|^2 > (1 - f_{max})^2$. Moreover, $Det(Y_2) = \lambda_1 \lambda_2 > 0$, so:

$$-\frac{2}{f_{min}^3 p_{min}^2 l_{max}^4} < -\frac{2}{ad - bc} < -|Tr(Y_2)| < (\lambda_1, \lambda_2) < |Tr(Y_2)| < \frac{2}{ad - bc} < \frac{2}{f_{min}^3 p_{min}^2 l_{max}^4}$$

Hence, according to Lemma 7, the error on the eigenvalues of Y_2 is bounded by $\frac{1050\epsilon}{f_{min}^{12} p_{min}^8 l_{max}^{16} (1 - f_{max})}$.

Furthermore, we have $n_1 = \frac{\lambda_1}{1+\lambda_1}$, hence $1 + \lambda_1 = \frac{1}{1-n} > 1$, hence if $\eta = \frac{1050\epsilon}{f_{min}^{12} p_{min}^8 l_{max}^{16} (1-f_{max})} < \frac{1}{2}$, the error on $f_{X,u}$ is bounded by:

$$\frac{\eta}{1 + \lambda_1 - \eta} + \frac{(1 + \lambda_2)\eta}{(1 + \lambda_1)(1 + \lambda_1 - \eta)} \leq 2(1 + f_{max})\eta = \frac{2100(1 + f_{max})\epsilon}{f_{min}^{12} p_{min}^8 l_{max}^{16} (1 - f_{max})}$$

Let us now bound the error on the parameter p_X . According to Equation 4.5, we have:

$$p_X = \frac{1 + \lambda_2}{\lambda_2 - \lambda_1} \times \mathbf{1}^T (X_2 - \lambda_1 X_1) \mathbf{1} .$$

We showed that the error on λ_1 and λ_2 is bounded by η . Moreover, for $\epsilon \leq \frac{f_{min}^{12} p_{min}^8 l_{max}^{16} (1-f_{max})}{8400}$, we can apply lemma 6 with $A = \frac{3}{f_{min}^3 p_{min}^2 l_{max}^4}$ and $\beta = \frac{1-f_{max}}{4}$ to bound the error on $\frac{1+\lambda_2}{\lambda_2-\lambda_1}$ by $e_1 = \frac{100}{f_{min}^3 p_{min}^2 l_{max}^4 (1-f_{max})^2}$. Since the error on the individual terms of X_1 and X_2 is bounded by $\epsilon \ll \eta$, we can also coarsely bound the error on $\mathbf{1}^T (X_2 - \lambda_1 X_1) \mathbf{1}$ by $e_2 = 16 \times 2 \times \eta$.

Given our upper bound on (λ_1, λ_2) and lower bound on $\lambda_2 - \lambda_1$, we also have $\frac{1+\lambda_2}{\lambda_2-\lambda_1} \leq m_1 = \frac{12}{f_{min}^3 p_{min}^2 l_{max}^4 (1-f_{max})}$, and $\mathbf{1}^T (X_2 - \lambda_1 X_1) \mathbf{1} \leq m_2 = \frac{48}{f_{min}^3 p_{min}^2 l_{max}^4}$. Hence, the error on p_X is bounded by:

$$e_1 \times m_2 + e_2 \times m_1 + e_1 \times e_2 \leq \frac{5000 \times 1050\epsilon}{f_{min}^{18} p_{min}^{12} l_{max}^{24} (1 - f_{max})^3}.$$

D.7.2 Proof of Lemma 9

As stated in the proof of Lemma 13, if the moments are known with error $\epsilon < \frac{n_{min}^2}{4}$, R is known to within $\frac{52\epsilon}{l_{max}^4}$. The error on the coefficients of $Q(X)$ is then bounded by $7 \times \frac{52\epsilon}{l_{max}^4}$. Moreover, we know that one root is smaller than $\frac{1}{2}$ and that the other is bigger than $\frac{1}{1+\sqrt{f_{A,a}f_{A,b}}}$, hence $\frac{\sqrt{\Delta}}{R(f_{A,a}-1)(f_{A,b}-1)} > \frac{1}{1+\sqrt{f_{A,a}f_{A,b}}} - \frac{1}{2} > (1 - f_{max})$, which implies that $\Delta > (1 - f_{max})^6 R_{min}^2 > (1 - f_{max})^6$. We can then use lemma 7 with $k = 1$ to show that the error on the roots of $Q(X)$ is bounded by $\frac{7 \times 4 \times 52\epsilon}{(1 - f_{max})^3 l_{max}^4}$. Additionally assuming $\epsilon < p_{min}/2$ and using lemma 6 with $A = 1/2, \beta = 1/2$, the error on $f_{A,x}$ is then bounded by $\frac{1-p_A}{p_A} \frac{2 \times 4 \times 1456\epsilon}{(1-f_{max})^3 l_{max}^4} < \frac{11648\epsilon}{(1-f_{max})^3 l_{max}^4 p_{min}}$.

D.7.3 Proof of Lemma 10

Using a Chernoff bound we have that the error in the empirical moments obtained from N samples is less than $\sqrt{\ln(\frac{2m}{\delta})} \frac{1}{\sqrt{N}}$ with probability $1 - \delta$. We combine the multiplicative factors on the error introduced by the tensor decomposition (Lemma 8) and during the extension step (Lemma 9) to achieve the stated result.

D.7.4 Proof of Lemma 11

We have $p_{min} < p_X < \frac{1}{2}$ and $f_{min} < f_c \forall c \in \mathcal{C}$, so for $\epsilon < \frac{p_{min} f_{min}^4}{6}$, $(1 - \tilde{p}_X + \tilde{p}_X \prod_{c \in \mathcal{C}} \tilde{f}_{X,c}) > (1 - p_X + p_X \prod_{c \in \mathcal{C}} f_{X,c}) - 6\epsilon > \frac{1}{2}$. Moreover, if the error on $(1 - p_X + p_X \prod_{c \in \mathcal{C}} f_{X,c})$ is bounded by 6ϵ , then the error on $\prod_{j=1}^l (1 - p_{X_j} + p_{X_j} \prod_{c \in \mathcal{C}} f_{X_j,c})$ is bounded by $6(l+1)\epsilon$ for ϵ small enough ($\epsilon < \frac{1}{3l^2(1+\epsilon)^l}$), and $\prod_{j=1}^l (1 - \tilde{p}_{X_j} + \tilde{p}_{X_j} \prod_{c \in \mathcal{C}} \tilde{f}_{X_j,c}) > M_0^l |\mathcal{C}|$. Since the negative moment $N_{\mathcal{C}} < 1$, using Lemma 6, we then get that the error on the adjusted moment, $\tilde{N}'_{\mathcal{C}}$, is bounded by $M_0^2 |\mathcal{C}| \times 6(l+2)\epsilon$. l is bounded by the total number of parents, n , and $|\mathcal{C}|$ is bounded by 3.

D.7.5 Proof of Lemma 12

Proof. Let M be the 4×4 matrix representing the joint distribution of aggregated variables (a, b) and (c, d) . Let \mathcal{S} be the set of parents shared between two or more of those, and $\forall X \in \mathcal{S}, \forall u \in \{a, b, c, d\}$, let $f_{X,u}$ be the failure probability of the edge from X to u , and let n_u be the probability that u is **not** activated by parents outside of \mathcal{S} .

$\forall S \subset \mathcal{S}, \forall u \in \{a, b, c, d\}$, let $e_{S,u} = n_u \prod_{X \in S} f_{X,u}$. $e_{S,u}$ is the marginal probability of u being off given that nodes in S are on and nodes in $\mathcal{S} \setminus S$ are off. We then have:

$$M = \sum_{S \subset \mathcal{S}} \left(\prod_{X \in S} p_X \prod_{Y \in \mathcal{S} \setminus S} (1 - p_Y) \right) q_S r_S^T$$

With:

$$U_S = \begin{pmatrix} e_{S,a}e_{S,b} \\ e_{S,a}(1 - e_{S,b}) \\ (1 - e_{S,a})e_{S,b} \\ (1 - e_{S,a})(1 - e_{S,b}) \end{pmatrix}, V_S = \begin{pmatrix} e_{S,c}e_{S,d} \\ e_{S,c}(1 - e_{S,d}) \\ (1 - e_{S,c})e_{S,d} \\ (1 - e_{S,c})(1 - e_{S,d}) \end{pmatrix}$$

In particular, this means that if $\{a, b, c, d\}$ only share one parent, the rank of M is at most two (sum of two rank one matrices).

Conversely, if $|\mathcal{S}| > 1$, M is the sum of at least 4 rank 1 matrices, and its elements are polynomial expressions in the parameters of the model. The determinant itself is then a polynomial function of the parameters of the model $P(n_u, p_X, f_{X,u}; \forall u \in \{a, b, c, d\}, X \in \mathcal{S})$. Hence, if $P \not\equiv 0$, the set of its roots has measure 0. Table D.1 gives two examples of parameter settings showing that $P \not\equiv 0$. For other structures, notice that these can also serve as examples by setting $p_U = 0$ for additional parents.

U	p_U	$f_{U,a}$	$f_{U,b}$	$f_{U,c}$	$f_{U,d}$	U	p_U	$f_{U,a}$	$f_{U,b}$	$f_{U,c}$	$f_{U,d}$
X	0.2	0.2	0.4	0.6	1	X	0.2	0.2	0.4	0.6	0.8
Y	0.3	1	0.2	0.4	0.6	Y	0.3	1	0.2	0.4	1

Table D.1: Two settings where the determinant of the moments matrix is non zero **Left:** $Det(\theta_1) = 5.33 \times 10^{-7}$. **Right:** $Det(\theta_2) = 4.95 \times 10^{-7}$

□

D.7.6 Proof of Lemma 13

Proof. Indeed, we have:

$$\frac{P(\bar{a}, \bar{b})}{P(\bar{a})P(\bar{b})} - \frac{P(\bar{a}, \bar{b}|\bar{x})}{P(\bar{a}|\bar{x})P(\bar{b}|\bar{x})} = g_{f_{A,a}, f_{A,b}}(p_A) - g_{f_{A,a}, f_{A,b}}(p_{A|\bar{x}})$$

Moreover:

$$\begin{aligned} p_A - p_{A|\bar{x}} &= p_A \left(1 - \frac{f_{A,x}}{1 - p_A + p_A f_{A,x}} \right) \\ &\geq \frac{1}{2} p_{\min} (1 - f_{\max}) \end{aligned}$$

And, for $p \leq \frac{1}{2}$:

$$\begin{aligned} \frac{\partial g_{f_{A,a}, f_{A,b}}(p)}{\partial p} &= \frac{(1 - f_{A,a})(1 - f_{A,b})((1 - f_{A,a}f_{A,b})p^2 - 2p + 1)}{(1 - p(1 - f_{A,a}))^2(1 - p(1 - f_{A,b}))^2} \\ &\geq \frac{(1 - f_{\max})^2(1 - f_{\max}^2)}{8} \end{aligned}$$

Hence:

$$\frac{P(\bar{a}, \bar{b})}{P(\bar{a})P(\bar{b})} - \frac{P(\bar{a}, \bar{b}|\bar{x})}{P(\bar{a}|\bar{x})P(\bar{b}|\bar{x})} > \frac{p_{\min}(1 - f_{\max})^3(1 - f_{\max}^2)}{8}.$$

Suppose the estimation error on all of the probabilities is bounded by $\epsilon < \frac{l_{max}^2}{4}$, then the error on $P(\bar{a})P(\bar{b})$ is bounded by 3ϵ . We have $P(\bar{a})P(\bar{b}) > l_{max}^2$ and $P(\bar{a}, \bar{b}) \leq 1$, thus according to lemma 6 using $A = 1, \beta = n_{min}^2/4$, the estimation error on $\frac{P(\bar{a}, \bar{b})}{P(\bar{a})P(\bar{b})} - \frac{P(\bar{a}, \bar{b}|\bar{x})}{P(\bar{a}|\bar{x})P(\bar{b}|\bar{x})}$ is bounded by $2 \times (\frac{48}{l_{max}^4}\epsilon + \frac{4}{l_{max}^2}) \leq \frac{104}{l_{max}^4}\epsilon = \eta$.

Hence, if $\epsilon \leq \frac{l_{max}^4 p_{min}(1-f_{max})^3(1-f_{max}^2)}{104 \times 10}$, $\eta \leq \frac{p_{min}(1-f_{max})^3(1-f_{max}^2)}{10}$, which proves the lemma. \square

D.7.7 Proof of Lemma 14

Proof. According to Lemmas 1 and 13, we need to bound the error on fourth-order moments by η_4 and on the third-order moments by η_3 respectively. Using a Chernoff bound we get that after N samples:

- if $N \geq \frac{3}{\eta_4} \ln(\frac{2m^4}{\delta})$, then the error on a fourth-order moment is smaller than η_4 with probability $(1 - \frac{\delta}{m^4})$, hence the maximum error on the fourth-order moments is bounded by η_4 with probability at least $(1 - \delta)$.
- Similarly, if $N \geq \frac{3}{\eta_3} \ln(\frac{2m^3}{\delta})$, then the maximum error on the third-order moments is bounded by η_3 with probability at least $(1 - \delta)$.

Taking the maximum of these numbers of samples, we find that with probability at least $(1 - \delta)$, all our structure tests are correct. \square

D.8 Empirically testing local identifiability

Section 4.3 describes an constructive proof of identifiability for a limited family of networks. However, it leaves open the possibility that there are networks outside of that family that are still identifiable and some may even be identifiable from low-order moments.

An empirical test for identifiability is described in Hsu et al. (2012). The **Check-identifiability** routine works by constructing the Jacobian matrix $J_{i,j} = \delta f_j / \delta \theta_i$ where f_j is the j th observed moment and θ_i is the i th parameter of the model and evaluating it at a random parameter setting θ_0 . If the rank of the matrix is equal to the number of parameters, then the model is locally identifiable for all but a zero-measure set of parameter settings. This local identifiability test can be used to rule out identifiability of certain structures since the lack of local identifiability implies the lack of global identifiability as well. A stronger result that holds if the moments can be expressed as polynomial expressions of the parameters, as they can in the noisy-or setting (Equation 4.14). In that case, local identifiability implies that the number of observationally equivalent models is finite.

Theorem 5 formally describes the properties of the check-identifiability routine.

Theorem 5. (*Check-Identifiability (Hsu et al., 2012)*) *Assume the parameter space Θ is a non-empty open connected subset of $[0 - 1]^n$; and the observed moments $\mu : \mathbb{R}^n \rightarrow \mathbb{R}^m$, with respect to the observation function ϕ is a polynomial map. Then with probability 1, Check-identifiability returns “yes” iff the model family is locally identifiable from ϕ . Moreover, if it returns “yes”, then there exists $\mathcal{E} \subset \Theta$ of measure 0 such that the model family with parameter space $\Theta \setminus \mathcal{E}$ is identifiable from ϕ .*

The proof is found in appendix A of Hsu et al. (2012).

D.9 Non identifiability of singly-coupled triplets

Finding the latent structure of a network is an especially interesting problem. Not all network structures are identifiable. Indeed, by applying the tensor decomposition method to a triplet that shares two parents, we can often find one parent that would explain the moments just as well. Figure D.1 gives an example of such a network. The parameters are the following:

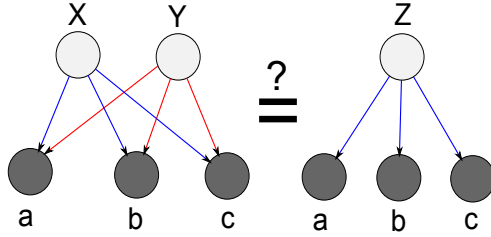


Figure D.1: Two networks with the same moments matrix using parameters: $p_X = 0.2$, $p_Y = 0.3$, $p_Z = 0.37$. $f_X = (0.1, 0.2, 0.3)$, $f_Y = (0.6, 0.4, 0.5)$, $f_Z = (0.28, 0.23, 0.33)$. One is singly-coupled, the other is not.

original values

```
p = [0.20000000000000001, 0.29999999999999999]
f = [[0.10000000000000001, 0.20000000000000001, 0.29999999999999999],
      [0.59999999999999998, 0.40000000000000002, 0.5]]
n = [0.94999999999999996, 0.94999999999999996, 0.94999999999999996]
```

new

```
p = 0.369002801906
f = [0.27471379503828641, 0.22833778992716067, 0.3253928941874199]
n = [0.93603297899373428, 0.91486319364631141, 0.92461657113192608]
```

```
original_D [[[ 0.50557963  0.05459129]
              [ 0.06907822  0.05627086]]
             [[ 0.05137117  0.04281791]
              [ 0.06842098  0.15186994]]]
```

```
new_D [[[ 0.50557963  0.05459129]
```

```
[ 0.06907822  0.05627086]]
```

```
[[ 0.05137117  0.04281791]
```

```
[ 0.06842098  0.15186994]]]
```

```
difference = 3.46944695195e-17
```

D.10 Simulation experiments

Just how confusable are singly-coupled triplets with other networks? The following two simulation experiments explore this question. A $2 \times 2 \times 2$ probability tensor can be tested to see whether it is consistent with a singly-coupled triplet by decomposing the triplet as in Algorithm 2 and extracting the parameters with Equations 4.5-4.8. If the parameters lie in the unit interval then the distribution is consistent with a singly-coupled triplet.

Dirichlet samples: $2 \times 2 \times 2$ tensors are drawn i.i.d from a Dirichlet distribution with $\alpha = 1$ (i.e., uniformly from the simplex) and tested to see whether they are consistent with a distribution induced by singly-coupled triplets.

Noisy-or network samples: Parameters of noisy-or networks are drawn uniformly from the unit interval for networks with 3 children and 1-5 parents and the resulting distribution is tested to see whether it is consistent with a distribution induced by singly-coupled triplets.

The results of the simulation are recorded in Table D.2. The implication is that even though general distributions are not often consistent with singly-coupled triplets, distributions drawn from noisy-or networks with any number of parents are. This is not surprising. Variables drawn from a noisy-or network cannot be negatively correlated with each other, so no distribution with negatively correlated variables can be perfectly consistent with a noisy-or

Table D.2: Results of a simulation experiment. 10,000 samples were drawn and tested whether they are consistent with a distribution from singly-coupled triplets.

Experiment	proportion valid
Dirichlet	0.022
Noisy-or 1	1
Noisy-or 2	0.973
Noisy-or 3	0.9803
Noisy-or 4	0.9897
Noisy-or 5	0.995

network. However, once samples are drawn from the family of noisy-or networks, it is very easy to confuse a non-singly coupled triplet with one that is.

D.11 Learning with Singly-coupled triplets and the Expansion property of Anandkumar et al. (2013c)

It is natural to compare the graph expansion property for identifiability in Anandkumar et al. (2013c) with the graph properties that make a network learnable from singly-coupled triplets. It turns out that neither property implies the other.

D.11.1 Expansion property

The expansion property for bipartite graphs $\mathcal{G}(\mathcal{V}_1, \mathcal{V}_2)$ with edges from \mathcal{V}_1 to \mathcal{V}_2 is as follows:

Definition 7. (*Expansion (Anandkumar et al., 2013c)*) For any subset $S \subset \mathcal{V}_1$ with $|S| \geq 2$, we have $|N(S)| \geq |S| + d_{max}$ where $N(S)$ are the neighbors of S , and d_{max} is the maximal degree of any node in \mathcal{V}_1 .

An additional requirement mentioned in Anandkumar et al. (2013c) is that there are at least $3\times$ more observed children than parents.

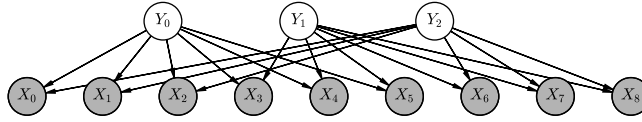


Figure D.2: A network that satisfies the expansion property but has no singly-coupled triplets.

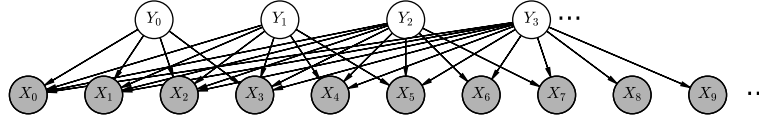


Figure D.3: A network that is learnable with singly-coupled triplets but does not satisfy the expansion property.

D.11.2 Expansion property does not imply learnable with singly-coupled triplets

The graph in Figure D.2 satisfies the expansion property but has no singly coupled triplets. In this network, each parent connects to 6 children, so $d_{max} = 6$. Any two parents connect to 9 children, so they satisfy $N(S) = 9 \geq 2 + d_{max} = 8$. All 3 parents connect to 9 children, so they satisfy $N(S) = 9 \geq 3 + d_{max} = 9$. There are 3 parents and 9 children, so the ratio of parents-to-children satisfies the assumptions of Anandkumar et al. (2013c).

D.11.3 Singly-coupled triplets does not imply Expansion

The graph in Figure D.3 is learnable with singly-coupled triplets but does not satisfy the expansion property since the last latent variable Y_n has degree m (i.e., connects to all of the observations). Since $d_{max} = m$ it is impossible to satisfy the expansion property, but it is learnable as described in Section 4.3.3.

Appendix E

Appendix to Learning a Health Knowledge Graph from Electronic Medical Records

E.1 Derivation of equation 5.5

Figure E.1 shows the causal structure assumed in Equation 5.5. The *do* operator performs “surgery” in the graphical model (Figure E.2), removing the edge from U to Y_1 and setting $Y_1 = y_1$. That leads to the following derivation for Equation 5.5:

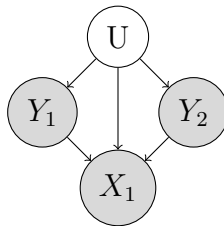


Figure E.1: The causal graph of diseases and symptoms. Diseases (Y_1, Y_2) can cause symptoms (only one symptom, X_1 , shown here), and can have a high dimensional set of common parents U (risk factors such as age) which make the diseases non-independent. Diseases and symptoms are observed.

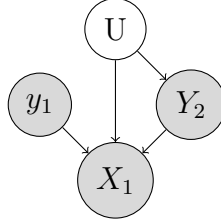


Figure E.2: The causal graph from Figure E.1 after intervening (do), setting $Y_1 = y_1$.

$$\begin{aligned}
 P(X_1 | \text{do}(Y_1 = y_1)) &= \sum_{u, y_2} P(u, y_2, X_1 | y_1) \\
 &= \sum_{y_2} P(y_2, X_1 | y_1) \\
 &= \sum_{y_2} P(y_2 | y_1) P(X_1 | y_1, y_2) \\
 &= \sum_{y_2} P(y_2) P(X_1 | y_1, y_2) \\
 &= E_{y_2 \sim P(Y_2)} P(X_1 | y_1, y_2).
 \end{aligned}$$

The equality $P(Y_2 | Y_1) = P(Y_2)$ (moving from the third to fourth line of the derivation) comes from the independence statements encoded in the graph after applying the do operator (Figure E.2).

Appendix F

Appendix to Anchored Factor Analysis

F.1 R matrix is full rank

In this section we show that the matrix R is invertible by showing that its component blocks R_Z are all invertible.

R is block-diagonal, so its determinant is equal to the product of the determinants of the blocks and will be non-zero as long as all of the blocks have non-zero determinants.

Each block, R_Z is a Kronecker product of conditional distributions: $R_Z = \otimes_{k=1}^{|Z|} P(A_k|Z_k)$, so its determinant will be non-zero as long as each of the 2×2 matrices, $P(A_k|Z_k)$, have non-zero determinants.

$P(A_k|Z_k)$ has a non-zero determinant as long as $P(A_k|Y_k = 0) \neq P(A_k|Y_k = 1)$ which is assumed in the definition of anchored latent variable models since $A_k \not\perp Y_k$.

Thus, the determinant of the Kronecker product is also non-zero, and the R matrix is full rank.

$$\begin{aligned}
P(W_1, W_2, X_j) &= P(Y_i = 0)P(W_1|Y_i = 0)P(W_2|Y_i = 0)P(X_j|Y_i = 0) \\
&\quad + P(Y_i = 1)P(W_1|Y_i = 1)P(W_2|Y_i = 1)P(X_j|Y_i = 1) \tag{F.1}
\end{aligned}$$

This decomposition can be computed efficiently (Berge, 1991), and the conditional probabilities can be recovered.

F.2 Estimating failures with Markov blanket conditioning

The Markov Blanket estimator from Section 6.4.2 follows from the noisy-or parametrization of the model. Here we show that the estimator from Equation 6.9 is indeed a consistent estimator. We start with the estimator:

$$f_{i,j}^{blanket} = \frac{P(X_j = 0|Y_i = 1, B = b)}{P(X_j = 0|Y_i = 0, B = b)},$$

Breaking the numerator and denominator of the RHS. Let $\mathcal{C} = B \cup Y_i$ represent the conditioned variables and $\mathcal{U} = \mathcal{Y} \setminus \mathcal{C}$ be the unconditioned variables. For the numerator, we have:

$$\begin{aligned}
P(X_j = 0|Y_i = 1, B = b) &= (1 - l_j) f_{i,j} \prod_{i' \in B} f_{i',j}^{y_{i'}} \left(\sum_{y_{\mathcal{U}}} P(y_{\mathcal{U}}|Y_i = 1, B = b) \prod_{k \in \mathcal{U}} f_{k,j}^{y_k} \right) \\
&= f_{i,j} \prod_{i' \in B} f_{i',j}^{y_{i'}} \left(\sum_{y_{\mathcal{U}}} P(y_{\mathcal{U}}|B = b) \prod_{k \in \mathcal{U}} f_{k,j}^{y_k} \right). \tag{F.2}
\end{aligned}$$

Similarly, for the denominator:

$$\begin{aligned}
P(X_j = 0|Y_i = 0, B = b) &= (1 - l_j) \prod_{i' \in B} f_{i',j}^{y_{i'}} \left(\sum_{y_{\mathcal{U}}} P(y_{\mathcal{U}}|Y_i = 0, B = b) \prod_{k \in \mathcal{U}} f_{k,j}^{y_k} \right) \\
&= \prod_{i' \in B} f_{i',j}^{y_{i'}} \left(\sum_{y_{\mathcal{U}}} P(y_{\mathcal{U}}|B = b) \prod_{k \in \mathcal{U}} f_{k,j}^{y_k} \right). \tag{F.3}
\end{aligned}$$

The second lines follow from the Markov blanket property. Thus, the ratio is equal to $f_{i,j}$. As the number of samples approaches infinity, the empirical estimates, $\hat{P}(X_j = 0|Y_i = 0, B = b)$ and $\hat{P}(X_j = 0|Y_i = 1, B = b)$, respectively approach the values of the true probabilities, and thus the estimator is consistent, provided that, $P(X_j = 0|Y_i = 0, B = b) > 0$.

F.3 Estimating correction coefficients serially in trees

In this section we derive the correction terms for estimating the failure probabilities $f_{i,j}$ in tree models described in Section 6.4.2 (Equation 6.12). While the directionality of the tree is not important, for the derivation it is easier notationally to consider a rooted tree (so that we can refer to specific subtrees), and without loss of generality, when estimating the failure term $f_{i,j}$, we can consider the tree as rooted at Y_i .

We introduce the notation $P_{\mathcal{T}(Y_i)}(e)$ to denote the likelihood of an event e in the graphical model where all nodes but Y_i and its non-descendants are removed (i.e., all that remains is the subtree rooted at Y_i), and where we exclude the leak probabilities (so that they are not double counted).

Conditioning on the variable Y_i taking the value y_i , the likelihood of $X_j = 0$ can be shown

to be of the form:

$$\begin{aligned}
P(X_j = 0|y_i) &= (1 - l_j)P_{\mathcal{T}(Y_i)}(X_j = 0|y_i) \\
&= (1 - l_j)f_{i,j}^{y_i} \prod_{k \in \text{child}(Y_i)} \sum_{y_k} P(y_k|y_i)P_{\mathcal{T}(Y_k)}(X_j = 0|y_k).
\end{aligned} \tag{F.4}$$

Correction terms are defined as

$$c_{i,j,k} = \frac{\sum_{y_k} P(y_k|y_i^{(1)})P_{\mathcal{T}(Y_k)}(X_j = 0|y_k)}{\sum_{y_k} P(y_k|y_i^{(0)})P_{\mathcal{T}(Y_k)}(X_j = 0|y_k)}. \tag{F.5}$$

Using Equation F.4, it is easy to see that:

$$\frac{P(X_j = 0|y_i^{(1)})}{P(X_j = 0|y_i^{(0)})} \prod_{k \in \text{child}(Y_i)} \frac{1}{c_{i,j,k}} = f_{i,j}. \tag{F.6}$$

It remains to be shown that $c_{i,j,k}$ can be estimated from low order moments as in Equation 6.11, i.e.

$$c_{i,j,k} = \frac{\sum_{y_k} P(y_k|y_i^{(1)})P(x_j^{(0)}|y_i^{(0)}, y_k)}{P(x_j^{(0)}|y_i^{(0)})}. \tag{F.7}$$

We can expand $P(x_j^{(0)}|y_i^{(0)}, y_k)$ as follows:

$$\begin{aligned}
& P(x_j^{(0)} | y_i^{(0)}, y_k) \\
&= (1 - l_j) f_{k,j}^{y_k} \prod_{k' \in \text{child}(Y_i) \setminus k} \sum_{y_{k'}} P(y_{k'} | y_i^{(0)}, y_k) P_{\mathcal{T}(Y_{k'})}(X_j = 0 | y_{k'}) \\
&\times \prod_{k' \in \text{child}(Y_k)} \sum_{y_{k'}} P(y_{k'} | y_i^{(0)}, y_k) P_{\mathcal{T}(Y_{k'})}(X_j = 0 | y_{k'}) \\
&= (1 - l_j) f_{k,j}^{y_k} \prod_{k' \in \text{child}(Y_i) \setminus k} \sum_{y_{k'}} P(y_{k'} | y_i^{(0)}) P_{\mathcal{T}(Y_{k'})}(X_j = 0 | y_{k'}) \\
&\times \prod_{k' \in \text{child}(Y_k)} \sum_{y_{k'}} P(y_{k'} | y_k) P_{\mathcal{T}(Y_{k'})}(X_j = 0 | y_{k'}), \tag{F.8}
\end{aligned}$$

where the second equality comes from the conditional independence properties that conditioning on (Y_i, Y_k) makes children of Y_k independent of Y_i and children of Y_i (other than Y_k) independent of Y_k .

We now substitute Eq. F.8 into Eq. F.7 and treat the numerator (num) and denominator (denom) separately:

$$\begin{aligned}
\text{num} &= (1 - l_j) \sum_{y_k} P(y_k | y_i^{(1)}) f_{k,j}^{y_k} \prod_{k' \in \text{child}(Y_k)} \sum_{y_{k'}} P(y_{k'} | y_k) P_{\mathcal{T}(Y_{k'})}(X_j = 0 | y_{k'}) \\
&\times \prod_{k' \in \text{child}(Y_i) \setminus k} \sum_{y_{k'}} P(y_{k'} | y_i^{(0)}) P_{\mathcal{T}(Y_{k'})}(X_j = 0 | y_{k'}^{(0)}) \tag{F.9}
\end{aligned}$$

The product over children of Y_i does not depend on y_k and can be pulled out of the sum.

$$\begin{aligned}
\text{num} &= (1 - l_j) \prod_{k' \in \text{child}(Y_i) \setminus k} \sum_{y_{k'}} P(y_{k'} | y_i^{(0)}) P_{\mathcal{T}(Y_{i'})}(X_j = 0 | y_i^{(0)}) \\
&\times \sum_{y_k} P(y_k | y_i^{(1)}) f_{k,j}^{y_k} \prod_{k' \in \text{child}(Y_k)} \sum_{y_{k'}} P(y_{k'} | y_k) P_{\mathcal{T}(Y_{k'})}(X_j = 0 | y_{k'}) \\
&= (1 - l_j) \prod_{k' \in \text{child}(Y_i) \setminus k} \sum_{y_{k'}} P(y_{k'} | y_i^{(0)}) P_{\mathcal{T}(Y_{i'})}(X_j = 0 | y_i^{(0)}) \\
&\times \sum_{y_k} P(y_k | y_i^{(1)}) P_{\mathcal{T}(Y_k)}(X_j = 0 | y_k) \tag{F.10}
\end{aligned}$$

Expanding the denominator using Equation F.4:

$$\text{denom} = P(x_j^{(0)} | y_i^{(0)}) = (1 - l_j) \prod_{k' \in \text{child}(Y_i)} \sum_{y_{k'}} P(y_{k'} | y_i^{(0)}) P_{\mathcal{T}(Y_{k'})}(X_j = 0 | y_{k'}) \tag{F.11}$$

Canceling the leak term and the product over all children of Y_i except for Y_k from both the numerator and denominator, we are left with:

$$\frac{\text{num}}{\text{denom}} = \frac{\sum_{y_k} P(y_k | y_i^{(1)}) P_{\mathcal{T}(Y_k)}(X_j = 0 | y_k)}{\sum_{y_k} P(y_k | y_i^{(0)}) P_{\mathcal{T}(Y_k)}(X_j = 0 | y_k)}$$

as desired.

F.4 Estimating leak parameters

Leak parameters can be estimated by comparing the empirical estimate of $\hat{P}(x_j^{(0)})$ to the one predicted by the model without leak, denoted as $P_{-l_j}(x_j^{(0)})$.

$$l_j = 1 - \frac{P_{-l_j}(x_j^{(0)})}{\hat{P}(x_j^{(0)})}. \tag{F.12}$$

If the latent variables are independent, the Quickscore algorithm (Heckerman, 1990) gives an efficient method of computing marginal probabilities:

$$P_{-l_j}(x_j^{(0)}) = \prod_i \left(P(y_i^{(0)}) + P(y_i^{(1)})f_{i,j} \right). \quad (\text{F.13})$$

For trees, this value can be computed efficiently using belief propagation and in more complex models it can be estimated by forward sampling.

F.5 Dataset preparation

In this section we provide additional details about the preparation of the two real-world datasets used in the experimental results.

Emergency: The emergency dataset consists of the following fields from patients’ medical records: Current medications (medication reconciliation) and Administered medications (pyxis records) mapped to GSN (generic sequence number) codes; Free text concatenation of `chief complaint`, `triage assessment` and `MD comments`; age binned by decade; sex; and administrative ICD9 billing codes (used to establish ground truth but not visible during learning or testing). We apply negation detection to the free-text section using “negex” rules (Chapman et al., 2001) with some manual adaptations appropriate for Emergency department notes (Jernite et al., 2013a), and replace common bigrams with a single token (e.g. “chest pain” is replaced by “chest_pain”). We reduce the dataset from 273,174 patients to 16,268 by filtering all patients that have fewer than 2 of the manually specified conditions. We filter words to remove those that appear in more than 50% of the dataset and take the 1000 most common words after that filtering. Table F.1 lists the concepts that are used and a selection of their anchors specified by a physician research collaborator. In the feature vector, anchors are replaced by a single feature which represents a union of the anchors (i.e. whether *any* of

the anchors appear in the patient record).

Stack Overflow: Questions were initially filtered to remove all questions that do not have at least two of the 200 most popular tags. We filter words to remove those that appear in more than 50% of the dataset and take the 1000 most common words after that filtering. Tag names that contain multiple words are treated as N-grams that are replaced by a single token in the text.

F.6 Detailed methods

F.6.1 Regularization

We found it useful to introduce an additional regularization parameter to Equation 6.8 that encourages the recovered marginals to be close to independent unless there is strong evidence otherwise.

$$\mu_{\mathcal{Y}}^* = \operatorname{argmin}_{\mu \in \mathcal{M}} D_{KL}(\mu_{\mathcal{A}}, R\mu) + \lambda D_{KL}(\mu_{indep}, \mu), \quad (\text{F.14})$$

Where μ_{indep} is a marginal vector constructed using the single-variable marginals in an independent distribution.

For recovery of marginals under local consistency and marginal polytope constraints, we use the conditional gradient algorithm, as discussed in section 6.3.2, using a tolerance of 0.005 for the duality gap (as described in Jaggi (2013)) as a stopping criterion for Stack Overflow and 0.01 for medical records. When using simplex constraints, we use the more appropriate Exponentiated Gradient algorithm (Kivinen and Warmuth, 1995) for each marginal independently. We use a regularization parameter of $\lambda = 0.01$ to learn $P(\mathcal{Y})$ and $\lambda = 0.1$ to learn $P(\mathcal{X}|\mathcal{Y})$. The moments required to learn $P(\mathcal{X}|\mathcal{Y})$ are recovered using only simplex constraints.

Algorithm 10 Moment recovery (cond. gradient)

Minimize $f(\mu) = D_{KL}(\mu_A, R\mu)$ s.t. $\mu \in \mathcal{M}$

- 1: initialize $\mu^0 \in \mathcal{M}$ (e.g. uniform)
 - 2: **for** $k = 0, 1, \dots, M$ **do**
 - 3: $s \leftarrow \operatorname{argmin}_{s' \in \mathcal{M}} \langle s', \nabla f(\mu^k) \rangle$
 - 4: Compute search direction: $d \leftarrow s - \mu^k$
 - 5: Determine stepsize, $\gamma \in [0, 1]$
 - 6: Move in descent direction: $\mu^{k+1} \leftarrow \mu^k + \gamma d^k$
 - 7: **end for**
-

Linear programs in the conditional gradient algorithm are solved using Gurobi (Gurobi Optimization, 2014) and integer linear programs are solved using Toulbar2 (Sanchez et al., 2008). For structure learning we use the gobnilp package (Cussens and Bartlett, 2013) with a BIC score objective, though we note that any exact or approximate structure learner that takes a decomposable score as input could be used equally well.

F.6.2 Conditional gradient algorithm for moment recovery

Algorithm 10 describes the conditional gradient (Frank and Wolfe, 1956) algorithm that was used for moment recovery. Line 3 minimizes a linear objective over a compact convex set. In our setting, this is the marginal polytope or its relaxations. To minimize a linear objective over a compact convex set, it suffices to search over the vertices of the set. For the marginal polytope, these correspond to the integral vertices of the local consistency polytope. Thus, this step can be solved as an integer linear program with local consistency constraints.

We use a “fully corrective” variant of the conditional gradient procedure. If the minimization of line 3 returns a vertex that has previously been used, we perform an additional step, moving to the point that minimizes the objective over convex combinations of all previously seen vertices.

F.6.3 Monte Carlo EM

When running EM, we optimize the variational lower bound on the data likelihood:

$$\log P(X; \theta) \geq \mathcal{L}(q, \theta) = E_{y \sim q} [\log P(x, y; \theta) - \log P(y)]$$

The EM algorithm optimizes this bound with a coordinate ascent, alternating between E-steps which improve q and M-steps which improve θ . Usually both the E-step and M-steps are maximization steps, but incomplete M-steps which only improve θ in every M-step also leads to monotonic improvement of $\log(P(X; \theta))$ with every step. In this section, we describe a variant of Monte Carlo EM which is useful for this model using Gibbs sampling to approximate the E-step and a custom M-step which is guaranteed to improve the variational lower bound at every step. We hold the distribution $P(\mathcal{Y})$ fixed and only optimize the failure and leak probabilities. For these purposes, the leak probabilities can be treated as simply failure probabilities of an extra latent variable whose value is always 1.

F.6.3.1 Outer E-step

For the E-step, we use the Gibbs sampling procedure described in 4.3 of Wang et al. (2014).

F.6.3.2 Outer M-step

The M-step consists of a coordinate step guaranteed to improve θ for a fixed q . $P(x, y; \theta)$ is a fully observed model, but optimizing θ has no closed form. Instead we introduce m *auxiliary variables* $A \in [0, n + 1]^m$, and adopt the generative model described in Algorithm 11 which equivalently describes the fully-observed noisy-or model (i.e. $\sum_a P(X, Y, a; \theta) = P_{\text{noisy or}}(X, Y; \theta)$). In this new expanded model, we can perform a single E-M step with respect to the latent variable A in closed form.

Algorithm 11 Alternative generative model with auxiliary A variables.

```

 $Y \sim P(Y)$ 
for  $j$  in 1.. $m$  do
   $A_j \sim P(A_j = k|Y)$ 
   $X_j = A_j \leq n$ 
end for
 $P(A_j = k|Y) = \begin{cases} (1 - f_{k,j}) \prod_{i=0}^{k-1} f_{i,j}^{y_i} & k \leq n \\ 1 - \prod_{i=0}^{k-1} f_{i,j}^{y_i} & k = n + 1 \end{cases}$ 

```

inner E-step:

$$P(A_j = k|X_j, Y) = \begin{cases} 1 & X_j = 0 \cap k = n + 1 \\ 0 & X_j = 0 \cap k \neq n + 1 \\ 0 & X_j = 1 \cap k = n + 1 \\ 0 & X_j = 1 \cap k \leq n \cap Y_k = 0 \\ \propto \prod_{i=0}^{k-1} f_{i,j}^{y_i} (1 - f_{k,j}) & X_j = 1 \cap k \leq n \cap Y_k = 1 \end{cases} \quad (\text{F.15})$$

inner M-step:

$$f_{i,j} = 1 - \frac{\text{count}(A_j = i)}{\text{count}(A_j \leq i)} \quad (\text{F.16})$$

F.7 Learned models

F.7.1 Tree models

Figures F.1 and F.2 show models learned using second order recovered moments to learn maximal scoring tree structured models. Tables F.1 and F.2 show the factor loadings learned for these tree-structured models.

F.7.2 Bounded in-degree models

Figures F.3 and F.4 show models learned using third order recovered moments to learn maximal scoring graphs with bounded in-degree of two.

F.7.3 Factor loadings

Table F.1: Learned medical concepts. For each concept we display the top 10 weighted factors and one supplied anchor.

Latent variable name	Top weights	anchors
abdominal pain	pain, dispensed:ondansetron, nausea, neg:fevers, dispensed:morphine sulfate, vomiting, days, dispensed:hydromorphone (dilaudid), neg:vomiting	abdominal pain
alcohol	sex:m, sober, admits, found, drink, dispensed:thiamine, dispensed:folic acid, dispensed:diazepam, dispensed:multivitamins, dispensed:multivitamins	etoh
allergic reaction	disposition, initial vitals, pending, consults, interventions, trigger, imaging, ed, diagnosis, pain	allergy
asthma-copd	med-history:albuterol sulfate, sob, dispensed:methylprednisolone sodium succ, cough, nebs, med-history:spiriva with handihaler, home, days, dyspnea	asthma

back pain	pain, neg:or, neg:pain, back, denies, lower back, dispensed:oxycodone-acetaminophen, neg:of, low back, neg:bowel	back pain
cellulitis	ed, admit, swelling, imaging, diagnosis, days, consults, iv, leg, interventions	cellulitis
stroke	age 80, admit, patient, head, ekg, weakness, ed, ct head, neuro, htn	stroke
epistaxis	nose, bleeding, blood, neg:bleeding, ekg, neg:with, ed, bleed, consults, today	epistaxis
fall	pain, denies, neg:pain, p fall, neg:or, ed, imaging, consults, interventions, neg:loc	fell down
gastrointestinal bleed	blood, dispensed:pantoprazole sodium, hct, gi, stool, admit, rectal, today, dispensed:lidocaine jelly 2% (urojet),	gi bleed
headache	head, nausea, today, neg:or, denies, neg:pain, days, dispensed:acetaminophen, pain, neck pain	headache
hematuria	sex:m, urine, urology, blood, foley, neg:pain, days, neg:fevers, dysuria, bladder	hematuria
hiv+	sex:m, med-history:truvada, cd, age 40, med-history:ritonavir, days, pcp, cough, fever, denies	hiv
intracranial hemorrhage	osh, ct, fall, age 80, found, head ct, transfer, sex:m, repeat, small	ich

kidney stone	pain, flank pain, dispensed:ondansetron, dispensed:ketorolac, stone, nausea, urology, ct, dispensed:morphine sulfate	kidney stone
liver failure (history)	sex:m, liver, age 50, med-history:lactulose, admit, ed, med-history:folic acid, med-history:furosemide, med-history:multivitamin, med-history:lasix	cirrhosis
motor vehicle collision	car, neg:loc, age 20, hit, neck, neg:head, driver, mph, neg:airbag, front	mvc
pneumonia	cxr, admit, sex:m, cough, ed, fever, diagnosis, ekg, med-history:icine, admit med-history:icine	pna
severe sepsis	dispensed:fentanyl citrate, found, icu, osh, vanc, age 80, imaging, consults, interventions, lactate	severe sepsis
sexual assault	patient, dispensed:ondansetron odt, evaluation, ceftriaxone, man, eval, age 30, home, plan, flagyl	sane nurse
suicidal ideation	depression, neg:hi, denies, states, neg:si, plan, dispensed:lorazepam, eval, age 40, section	si
syncope	ekg, neg:pain, fall, head, fell, denies, neg:cp, ed, neg:of, loc	syncope
urinary tract infection	ed, imaging, consults, interventions, diagnosis, ua, cipro, admit, pending, disposition	uti

Table F.2: Learned concepts from Stack Overflow. For each concept we display the top weighted factors and one supplied anchor. Note that in the Stack Overflow dataset we use a simple rule to provide a single anchor for every latent variable.

Latent variable name	Top weights	anchors
osx	osx, i'm, running, i've, install, installed, os, code, managed, existing	osx
ruby-on-rails-3	parsing, executed, web-application, don't, numbers, developed, resources, dynamic, named, rest	ruby-on-rails-3
image	image, code, size, upload, html, save, picture, width, page, display	image
mysql	mysql, query, rows, row, id, 1, tables, join, insert, column	mysql
web-services	web, service, web-services, client, java, services, things, sharepoint, don't, true	web-services
objective-c	objective-c, i'm, code, don't, xcode, dynamic, syntax, including, follow, trouble	objective-c
linux	linux, running, command, machine, system, server, directory, install, php, servers	linux
query	query, table, a, result, results, queries, tables, return, returns, select	query
regex	regex, match, expression, regular, pattern, i'm, replace, characters, html, extract	regex
php	php, server, array, send, data, i'm, database, output, website, user	php

java	java, string, program, client, xml, read, code, existing, send, environment	java
ruby-on-rails	ruby-on-rails, rails, i'm, database, user, controller, page, ruby, model, code	ruby-on-rails
asp.net	asp.net, asp.net-mvc-3, controller, site, website, database, control, ajax, jquery, action	asp.net
sql-server	sql-server, query, tsql, sql, sql-server-2005, server, stored, procedure, rows, columns	sql-server
forms	form, forms, page, code, data, html, jquery, user, button, submit	forms
python	python, i'm, a, program, string, list, module, output, don't, write	python
json	json, string, response, data, php, parse, code, format, ajax, javascript	json
html	html, tags, tag, page, website, code, links, show, webpage, text	html
iphone	iphone, app, view, device, sqlite, api, video,uitableview, images, ios5	iphone
performance	performance, time, data, question, takes, slow, faster, run, seconds, running	performance
android	android, code, xml, java, activity, device, app, phone, screen, android-layout	android
multithreading	thread, multithreading, code, threads, application, method, data, main, run, managed	multithreading

xcode	xcode, project, build, app, 4, version, i've, running, debug, target	xcode
css	css, page, css3, jquery, style, javascript, chrome, works, width, browser	css
jquery	jquery, html, works, plugin, javascript, jquery-ui, css, jquery-ajax, elements, link	jquery
string	string, a, strings, array, characters, output, convert, json, character, split	string
c#	c#, function, code, methods, event, excel, click, written, program, call	c
javascript	javascript, jquery, js, script, works, ajax, php, button, browser, code	javascript
ios	ios, iphone, app, ios5, device, i'm, ipad, 5, user, screen	ios
linq	linq, linq-to-sql, code, class, c#, error, xml, expression, property, collection	linq
xml	xml, parse, document, read, data, string, node, output, format, tag	xml
ajax	ajax, code, javascript, jquery-ajax, works, call, response, user, request, html	ajax
facebook	facebook, users, login, api, app, sdk, post, php, id, link	facebook
html5	html5, html, browser, chrome, css, support, image, browsers, works, android	html5

sql	sql, table, database, query, data, tsql, tables, statement, column, sql-server	sql
wpf	wpf, control, window, property, binding, a, bind, controls, ui, items	wpf
asp.net-mvc	code, existing, browsers, query, faster, issues, row, flash, environment, program	asp.net-mvc
ruby	ruby, i'm, rails, running, string, install, installed, array, feed, executed	ruby
ipad	ipad, works, fine, screen, ios, page, device, problem, safari, app	ipad
c	c, program, a, write, c+, library, compile, string, array, functions	c
database	database, a, table, user, db, code, sql, data, store, created	database
arrays	arrays, i'm, 2, data, dynamic, results, saved, row, 0, finally	arrays
vb.net	vb.net, string, project, work, database, code, developed, results, rest, existing	vb.net
.net	.net, framework, i've, windows, executed, managed, valid, don't, developed, existing	.net
eclipse	eclipse, project, build, installed, plugin, debug, running, folder, version, projects	eclipse
c++	c++, c+, code, i'm, managed, find, access, application, implementation, writing	c++

windows	windows, windows-7, a, running, winapi, c#, machine, run, application, window	windows
winforms	winforms, control, user, i'm, windows, a, .net, controls, forms, don't	winforms
cocoa-touch	a, i'm, cocoa-touch, code, dynamic, results, follow, row, send, environment	cocoa-touch
sql-server-2008	sql-server-2008, sql-server, sql, query, tsql, stored, 1, procedure, developed, resources	sql-server-2008

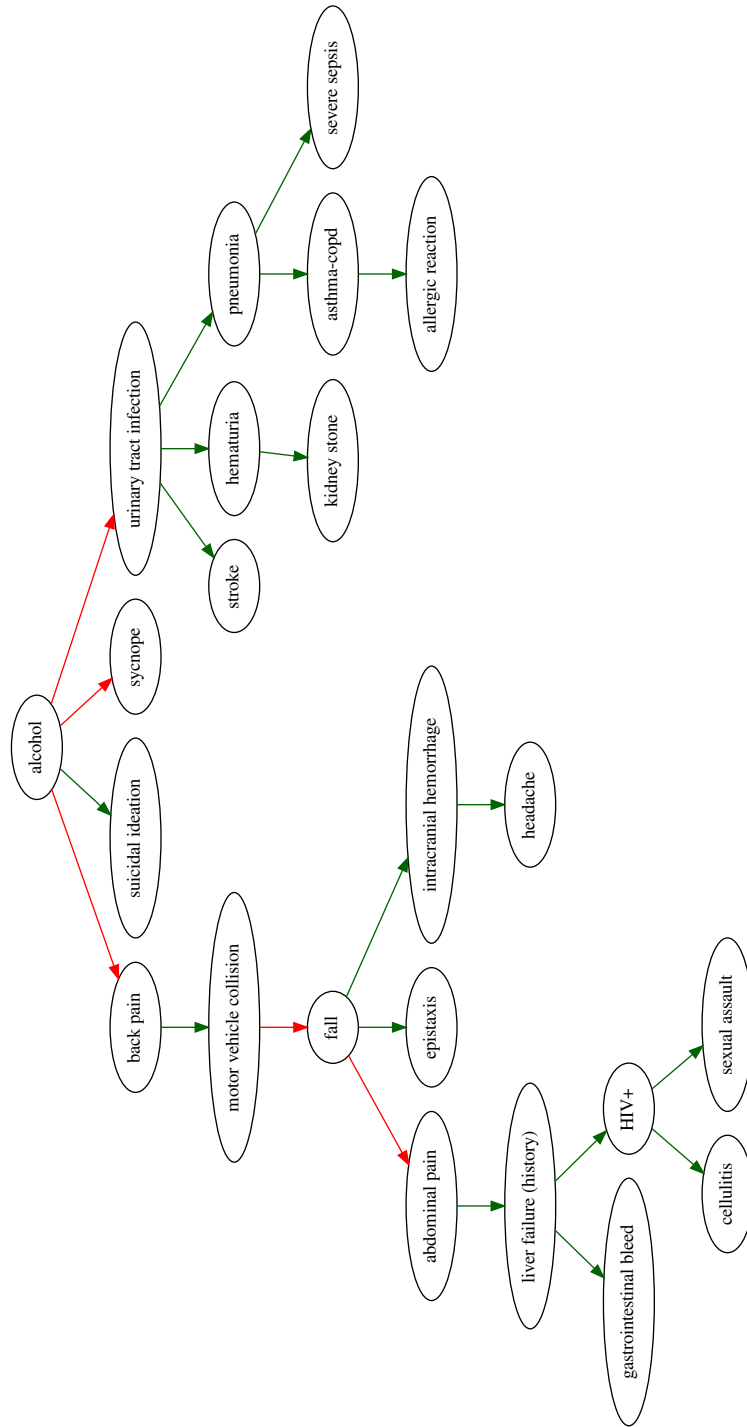


Figure F.1: Tree model learned for Emergency data. Red and green edges represent positive and negative correlations between the variables, respectively.

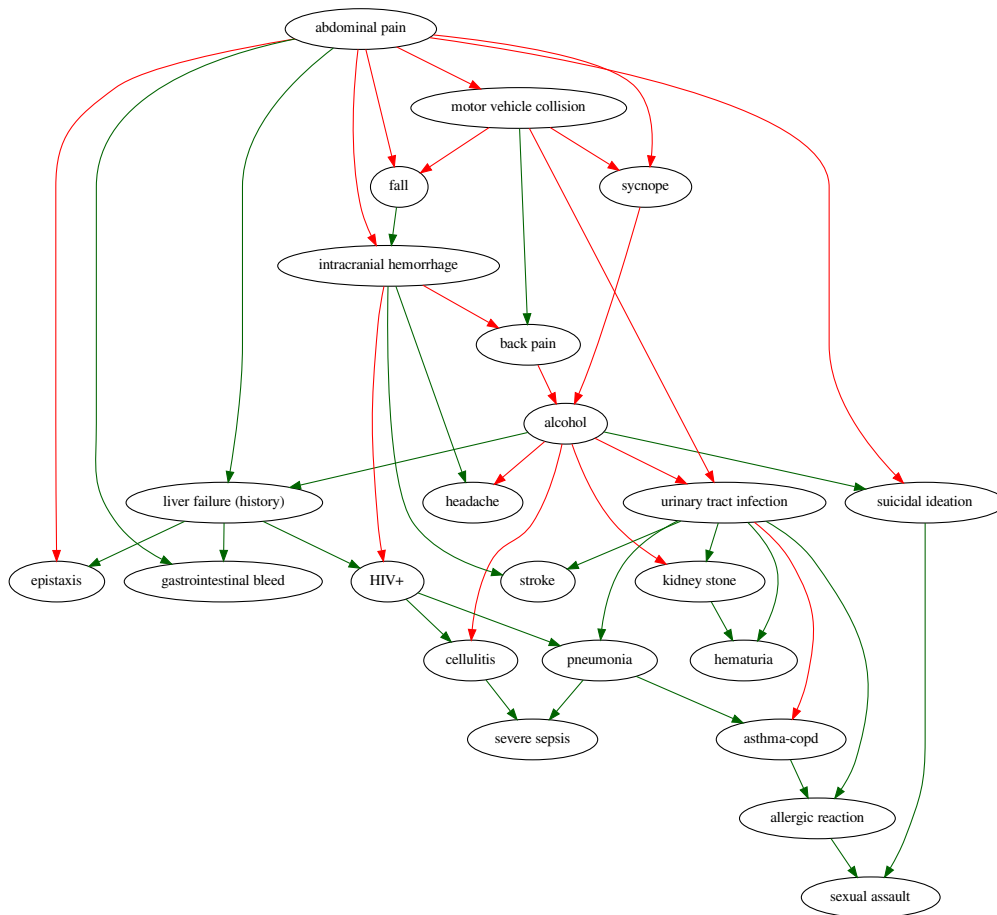


Figure F.3: Bounded in-degree model (≤ 2) learned for Emergency data. Red and green edges represent positive and negative correlations between the variables, respectively.

Appendix G

Appendix to Topic modeling with anchors

G.1 Proof for anchor-words finding algorithm

Recall that the correctness of the algorithm depends on the following Lemma:

Lemma 15. *The point d_j found by the algorithm must be $\delta = O(\epsilon/\gamma^2)$ close to some vertex v_i . In particular, the corresponding a_j $O(\epsilon/\gamma^2)$ -covers v_i .*

In order to prove this Lemma, we first show that even if previously found vertices are only δ close to some vertices, there is still another vertex that is far from the span of previously found vertices.

Lemma 16. *Suppose all previously found vertices are $O(\epsilon/\gamma^2)$ close to distinct vertices, there is a vertex v_i whose distance from $\text{span}(S)$ is at least $\gamma/2$.*

In order to prove Lemma 16, we use a volume argument. First we show that the volume of a robust simplex cannot change by too much when the vertices are perturbed.

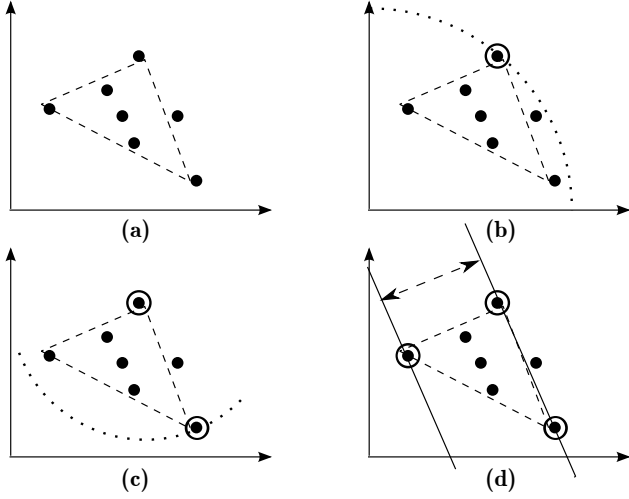


Figure G.1: Illustration of the Algorithm

Lemma 17. *Suppose $\{v_1, v_2, \dots, v_K\}$ are the vertices of a γ -robust simplex S . Let S' be a simplex with vertices $\{v'_1, v'_2, \dots, v'_K\}$, each of the vertices v'_i is a perturbation of v_i and $\|v'_i - v_i\|_2 \leq \delta$. When $10\sqrt{K}\delta < \gamma$ the volume of the two simplices satisfy*

$$\text{vol}(S)(1 - 2\delta/\gamma)^{K-1} \leq \text{vol}(S') \leq \text{vol}(S)(1 + 4\delta/\gamma)^{K-1}.$$

Proof. As the volume of a simplex is proportional to the determinant of a matrix whose columns are the edges of the simplex, we first show the following perturbation bound for determinant.

Claim 1. *Let A, E be $K \times K$ matrices, the smallest eigenvalue of A is at least γ , the Frobenius norm $\|E\|_F \leq \sqrt{K}\delta$, when $\gamma > 5\sqrt{K}\delta$ we have*

$$\det(A + E)/\det(A) \geq (1 - \delta/\gamma)^K.$$

Proof. Since $\det(AB) = \det(A)\det(B)$, we can multiply both A and $A + E$ by A^{-1} . Hence $\det(A + E)/\det(A) = \det(I + A^{-1}E)$.

The Frobenius norm of $A^{-1}E$ is bounded by

$$\|A^{-1}E\|_F \leq \|A^{-1}\|_2 \|E\|_F \leq \sqrt{K}\delta/\gamma.$$

Let the eigenvalues of $A^{-1}E$ be $\lambda_1, \lambda_2, \dots, \lambda_K$, then by definition of Frobenius Norm $\sum_{i=1}^K \lambda_i^2 \leq \|A^{-1}E\|_F^2 \leq K\delta^2/\gamma^2$.

The eigenvalues of $I + A^{-1}E$ are just $1 + \lambda_1, 1 + \lambda_2, \dots, 1 + \lambda_K$, and the determinant $\det(I + A^{-1}E) = \prod_{i=1}^K (1 + \lambda_i)$. Hence it suffices to show

$$\min \prod_{i=1}^K (1 + \lambda_i) \geq (1 - \delta/\gamma)^K \text{ when } \sum_{i=1}^K \lambda_i^2 \leq K\delta^2/\gamma^2.$$

To do this we apply Lagrangian method and show the minimum is only obtained when all λ_i 's are equal. The optimal value must be obtained at a local optimum of

$$\prod_{i=1}^K (1 + \lambda_i) + C \sum_{i=1}^K \lambda_i^2.$$

Taking partial derivatives with respect to λ_i 's, we get the equations $-\lambda_i(1 + \lambda_i) = -\prod_{i=1}^K (1 + \lambda_i)/2C$ (here using $\sqrt{K}\delta/\gamma$ is small so $1 + \lambda_i > 1/2 > 0$). The right hand side is a constant, so each λ_i must be one of the two solutions of this equation. However, only one of the solution is larger than $1/2$, therefore all the λ_i 's are equal.

□

For the lower bound, we can project the perturbed subspace to the $K - 1$ dimensional space. Such a projection cannot increase the volume and the perturbation distances only get smaller. Therefore we can apply the claim directly, the columns of A are just $v_{i+1} - v_1$ for $i = 1, 2, \dots, K - 1$; columns of E are just $v'_{i+1} - v_{i+1} - (v'_1 - v_1)$. The smallest eigenvalue of A is at least γ because the polytope is γ robust, which is equivalent to saying after orthogonalization each column still has length at least γ . The Frobenius norm of E is at

most $2\sqrt{K-1}\delta$. We get the lower bound directly by applying the claim.

For the upper bound, swap the two sets S and S' and use the argument for the lower bound. The only thing we need to show is that the smallest eigenvalue of the matrix generated by points in S' is still at least $\gamma/2$. This follows from Wedin's Theorem Wedin (1972) and the fact that $\|E\| \leq \|E\|_F \leq \sqrt{K}\delta \leq \gamma/2$. \square

Now we are ready to prove Lemma 16.

Proof. The first case is for the first step of the algorithm, when we try to find the farthest point to the origin. Here essentially $S = \{\vec{0}\}$. For any two vertices v_1, v_2 , since the simplex is γ robust, the distance between v_1 and v_2 is at least γ . Which means $\text{dis}(\vec{0}, v_1) + \text{dis}(\vec{0}, v_2) \geq \gamma$, one of them must be at least $\gamma/2$.

For the later steps, recall that S contains vertices of a perturbed simplex. Let S' be the set of original vertices corresponding to the perturbed vertices in S . Let v be any vertex in $\{v_1, v_2, \dots, v_K\}$ which is not in S . Now we know the distance between v and S is equal to $\text{vol}(S \cup \{v\}) / (|S| - 1)\text{vol}(S)$. On the other hand, we know $\text{vol}(S' \cup \{v\}) / (|S'| - 1)\text{vol}(S') \geq \gamma$. Using Lemma 17 to bound the ratio between the two pairs $\text{vol}(S)/\text{vol}(S')$ and $\text{vol}(S \cup \{v\})/\text{vol}(S' \cup \{v\})$, we get

$$\text{dis}(v, S) \geq (1 - 4\epsilon'/\gamma)^{2|S|-2}\gamma > \gamma/2$$

when $\gamma > 20K\epsilon'$.

\square

Lemma 15 is based on the following observation: in a simplex the point with largest ℓ_2 is always a vertex. Even if two vertices have the same norm if they are not close to each other the vertices on the edge connecting them will have significantly lower norm.

Proof. (Lemma 15)

Since d_j is the point found by the algorithm, let us consider the point a_j before perturbation. The point a_j is inside the simplex, therefore we can write a_j as a convex combination of the vertices:

$$a_j = \sum_{t=1}^K c_t v_t$$

Let v_t be the vertex with largest coefficient c_t . Let Δ be the largest distance from some vertex to the space spanned by points in S ($\Delta = \max_l \text{dis}(v_l, \text{span}(S))$). By Lemma 16 we know $\Delta > \gamma/2$. Also notice that we are not assuming $\text{dis}(v_t, \text{span}(S)) = \Delta$.

Now we rewrite a_j as $c_t v_t + (1 - c_t)w$, where w is a vector in the convex hull of vertices other than v_t .

Observe that a_j must be far from $\text{span}(S)$, because d_j is the farthest point found by the algorithm. Indeed

$$\begin{aligned} \text{dis}(a_j, \text{span}(S)) &\geq \text{dis}(d_j, \text{span}(S)) - \epsilon \\ &\geq \text{dis}(v_l, \text{span}(S)) - 2\epsilon \geq \Delta - 2\epsilon. \end{aligned}$$

The second inequality is because there must be some point d_l that correspond to the farthest vertex v_l and have $\text{dis}(d_l, \text{span}(S)) \geq \Delta - \epsilon$. Thus as d_j is the farthest point $\text{dis}(d_j, \text{span}(S)) \geq \text{dis}(d_l, \text{span}(S)) \geq \Delta - \epsilon$.

The point a_j is on the segment connecting v_t and w , the distance between a_j and $\text{span}(S)$ is not much smaller than that of v_t and w . Following the intuition in ℓ_2 norm when v_t and w are far we would expect a_j to be very close to either v_t or w . Since $c_t \geq 1/K$ it cannot be really close to w , so it must be really close to v_t . We formalize this intuition by the following calculation (see Figure G.2):

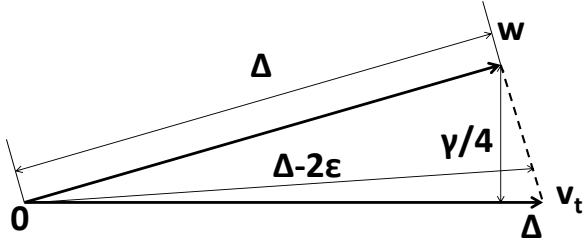


Figure G.2: Proof of Lemma 15, after projecting to the orthogonal subspace of $\text{span}(S)$.

Project everything to the orthogonal subspace of $\text{span}(S)$ (points in $\text{span}(S)$ are now at the origin). After projection distance to $\text{span}(S)$ is just the ℓ_2 norm of a vector. Without loss of generality we assume $\|v_t\|_2 = \|w\|_2 = \Delta$ because these two have length at most Δ , and extending these two vectors to have length Δ can only increase the length of d_j .

The point v_t must be far from w by applying Lemma 16: consider the set of vertices $V' = \{v_i : v_i \text{ does not correspond to any point in } S \text{ and } i \neq t\}$. The set $V' \cup S$ satisfy the assumptions in Lemma 16 so there must be one vertex that is far from $\text{span}(V' \cup S)$, and it can only be v_t . Therefore even after projecting to orthogonal subspace of $\text{span}(S)$, v_t is still far from any convex combination of V' . The vertices that are not in V' all have very small norm after projecting to orthogonal subspace (at most δ_0) so we know the distance of v_t and w is at least $\gamma/2 - \delta_0 > \gamma/4$.

Now the problem becomes a two dimensional calculation. When c_t is fixed the length of a_j is strictly increasing when the distance of v_t and w decrease, so we assume the distance is $\gamma/4$. Simple calculation (using essentially just Pythagorean theorem) shows

$$c_t(1 - c_t) \leq \frac{\epsilon}{\Delta - \sqrt{\Delta^2 - \gamma^2/16}}.$$

The right hand side is largest when $\Delta = 2$ (since the vectors are in unit ball) and the maximum value is $O(\epsilon/\gamma^2)$. When this value is smaller than $1/K$, we must have

$1 - c_t \leq O(\epsilon/\gamma^2)$. Thus $c_t \geq 1 - O(\epsilon/\gamma^2)$ and $\delta \leq (1 - c_t) + \epsilon \leq O(\epsilon/\gamma^2)$. \square

The cleanup phase tries to find the farthest point to a subset of $K - 1$ vertices, and use that point as the K -th vertex. This will improve the result because when we have $K - 1$ points close to $K - 1$ vertices, only one of the vertices can be far from their span. Therefore the farthest point must be close to the only remaining vertex. Another way of viewing this is that the algorithm is trying to greedily maximize the volume of the simplex, which makes sense because the larger the volume is, the more words/documents the final LDA model can explain.

The following lemma makes the intuitions rigorous and shows how cleanup improves the guarantee of Lemma 15.

Lemma 18. *Suppose $|S| = K - 1$ and each point in S is $\delta = O(\epsilon/\gamma^2) < \gamma/20K$ close to distinct vertices v_i 's, the farthest point found by the algorithm is d_j , then the corresponding a_j $O(\epsilon/\gamma)$ -covers the remaining vertex.*

Proof. We still look at the original point a_j and express it as $\sum_{t=1}^K c_t v_t$. Without loss of generality let v_1 be the vertex that does not correspond to anything in S . By Lemma 16 v_1 is $\gamma/2$ far from $\text{span}(S)$. On the other hand all other vertices are at least $\gamma/20r$ close to $\text{span}(S)$. We know the distance $\text{dis}(a_j, \text{span}(S)) \geq \text{dis}(v_1, \text{span}(S)) - 2\epsilon$, this cannot be true unless $c_1 \geq 1 - O(\epsilon/\gamma)$. \square

These lemmas directly lead to the following theorem:

Theorem 6. *FastAnchorWords algorithm runs in time $\tilde{O}(V^2 + VK/\epsilon^2)$ and outputs a subset of $\{d_1, \dots, d_V\}$ of size K that $O(\epsilon/\gamma)$ -covers the vertices provided that $20K\epsilon/\gamma^2 < \gamma$.*

Proof. In the first phase of the algorithm, do induction using Lemma 15. When $20K\epsilon/\gamma^2 < \gamma$ Lemma 15 shows that we find a set of points that $O(\epsilon/\gamma^2)$ -covers the vertices. Now Lemma 18 shows after cleanup phase the points are refined to $O(\epsilon/\gamma)$ -cover the vertices. \square

G.2 Proof for Nonnegative Recover Procedure

In order to show RecoverL2 learns the parameters even when the rows of \bar{Q} are perturbed, we need the following lemma that shows when columns of \bar{Q} are close to the expectation, the posteriors c computed by the algorithm is also close to the true value.

Lemma 19. *For a γ robust simplex S with vertices $\{v_1, v_2, \dots, v_K\}$, let v be a point in the simplex that can be represented as a convex combination $v = \sum_{i=1}^K c_i v_i$. If the vertices of S are perturbed to $S' = \{\dots, v'_i, \dots\}$ where $\|v'_i - v_i\| \leq \delta_1$ and v is perturbed to v' where $\|v - v'\| \leq \delta_2$. Let v^* be the point in $\text{conv}\{S'\}$ that is closest to v' , and $v^* = \sum_{i=1}^K c'_i v_i$, when $10\sqrt{K}\delta_1 \leq \gamma$ for all $i \in [K]$ $|c_i - c'_i| \leq 4(\delta_1 + \delta_2)/\gamma$.*

Proof. Consider the point $u = \sum_{i=1}^K c_i v'_i$, by triangle inequality: $\|u - v\| \leq \sum_{i=1}^K c_i \|v_i - v'_i\| \leq \delta_1$. Hence $\|u - v'\| \leq \|u - v\| + \|v - v'\| \leq \delta_1 + \delta_2$, and u is in S' . The point v^* is the point in $\text{conv}\{S'\}$ that is closest to v' , so $\|v^* - v'\| \leq \delta_1 + \delta_2$ and $\|v^* - u\| \leq 2(\delta_1 + \delta_2)$.

Then we need to show when a point (u) moves a small distance, its representation also changes by a small amount. Intuitively this is true because S is γ robust. By Lemma 16 when $10\sqrt{K}\delta_1 < \gamma$, the simplex S' is also $\gamma/2$ robust. For any i , let $Proj_i(v^*)$ and $Proj_i(u)$ be the projections of v^* and u in the orthogonal subspace of $\text{span}(S' \setminus v'_i)$, then

$$\begin{aligned} |c_i - c'_i| &= \|Proj_i(v^*) - Proj_i(u)\| / \text{dis}(v_i, \text{span}(S' \setminus v'_i)) \\ &\leq 4(\delta_1 + \delta_2)/\gamma \end{aligned}$$

and this completes the proof. □

With this lemma it is not hard to show that RecoverL2 has polynomial sample complexity.

Theorem 7. *When the number of documents M is at least*

$$\max\{O(aK^3 \log V/D(\gamma p)^6 \epsilon), O((aK)^3 \log V/D\epsilon^3(\gamma p)^4)\}$$

our algorithm using the conjunction of *FastAnchorWords* and *RecoverL2* learns the A matrix with entry-wise error at most ϵ .

Proof. (sketch) We can assume without loss of generality that each word occurs with probability at least $\epsilon/4aK$ and furthermore that if M is at least $50 \log V/D\epsilon_Q^2$ then the empirical matrix \tilde{Q} is entry-wise within an additive ϵ_Q to the true $Q = \frac{1}{M} \sum_{d=1}^M AW_dW_d^T A^T$ see Arora et al. (2012b) for the details. Also, the K anchor rows of \tilde{Q} form a simplex that is γp robust.

The error in each column of \tilde{Q} can be at most $\delta_2 = \epsilon_Q \sqrt{4aK/\epsilon}$. By Theorem 6 when $20K\delta_2/(\gamma p)^2 < \gamma p$ (which is satisfied when $M = O(aK^3 \log V/D(\gamma p)^6 \epsilon)$), the anchor words found are $\delta_1 = O(\delta_2/(\gamma p))$ close to the true anchor words. Hence by Lemma 19 every entry of C has error at most $O(\delta_2/(\gamma p)^2)$.

With such number of documents, all the word probabilities $p(w = i)$ are estimated more accurately than the entries of $C_{i,j}$, so we omit their perturbations here for simplicity. When we apply the Bayes rule, we know $A_{i,k} = C_{i,k}p(w = i)/p(z = k)$, where $p(z = k)$ is α_k which is lower bounded by $1/aK$. The numerator and denominator are all related to entries of C with positive coefficients sum up to at most 1. Therefore the errors δ_{num} and δ_{denom} are at most the error of a single entry of C , which is bounded by $O(\delta_2/(\gamma p)^2)$. Applying Taylor's Expansion to $(p(z = k, w = i) + \delta_{num})/(\alpha_k + \delta_{denom})$, the error on entries of A is at most $O(aK\delta_2/(\gamma p)^2)$. When $\epsilon_Q \leq O((\gamma p)^2 \epsilon^{1.5}/(aK)^{1.5})$, we have $O(aK\delta_2/(\gamma p)^2) \leq \epsilon$, and get the desired accuracy of A . The number of document required is $M = O((aK)^3 \log V/D\epsilon^3(\gamma p)^4)$.

The sample complexity for R can then be bounded using matrix perturbation theory. \square

For *RecoverKL*, we observe that the dimension and minimum values of v_i 's are all bounded by polynomials of ϵ , a , r (see Section 3.5 Reducing Dictionary Size of Arora et al. (2012b)). In this case, when distance δ is small enough, we know the KL-divergence is both upper and lowerbounded by some polynomial factor times ℓ_2 norm squared.

Lemma 20. *When all values in the vectors $\{v'_i\}$ are at least $l = \epsilon^2/20a^2r^2$, if u is one of v'_i , and v is in the convex hull of perturbed vertices $\{v'_1, v'_2, \dots, v'_K\}$, $\|u - v\| \leq \epsilon^2/100a^2r^2$, then $D_{KL}(u||v) \leq 2\|u - v\|^2/l$.*

Proof. Let $s_i = u_i - v_i$, apply Taylor's expansion on $\log(v_i + s_i)/v_i$, we know in the range of parameters $s_i + s_i^2/2v_i \leq \log(v_i + s_i)/v_i \leq s_i + 2s_i^2/v_i$.

Adding this up, using the fact $\sum s_i = \sum u_i - \sum v_i = 0$, we know the KL-divergence is bounded by

$$D_{KL}(u||v) \leq 2 \sum s_i^2/v_i \leq 2\|u - v\|^2/l.$$

□

On the other hand, by Pinsker's inequality, we know $D_{KL}(u||v) \geq 2|u - v|_1^2 \geq 2\|u - v\|^2$.

Using these two bounds we can easily prove a replacement for Lemma 19.

Lemma 21. *For a γ robust simplex S with vertices $\{v_1, v_2, \dots, v_K\}$, let v be a point in the simplex that can be represented as a convex combination $v = \sum_{i=1}^K c_i v_i$. If the vertices of S are perturbed to $S' = \{\dots, v'_i, \dots\}$ where $\|v'_i - v_i\| \leq \delta_1$ and v is perturbed to v' where $\|v - v'\| \leq \delta_2$. Further assume all entries of v' and v'_i are at least $l = \epsilon^2/20a^2r^2$. Let v^{KL} be the point in $\text{conv}\{S'\}$ that has smallest $D_{KL}(v'||v^{KL})$, and $v^{KL} = \sum_{i=1}^K c'_i v_i$, when $10\sqrt{K}\delta_1 \leq \gamma$, $(\delta_1 + \delta_2) < l/5$, for all $i \in [K]$ $|c_i - c'_i| \leq 4(\delta_1 + \delta_2)/\gamma\sqrt{l}$.*

Proof. Let v^* be the closest point (in ℓ_2 distance) of v' in $\text{conv}\{S'\}$. By proof of Lemma 19 we know $\|v^* - v'\| \leq \delta_1 + \delta_2$. Hence by Lemma 20 $D_{KL}(v'||v^*) \leq 2(\delta_1 + \delta_2)^2/l$.

Since v^{KL} is the point with smallest divergence, we know in particular $D_{KL}(v'||v^{KL}) \leq 2(\delta_1 + \delta_2)^2/l$. On the other hand, by Pinsker's inequality $D_{KL}(v'||v^{KL}) \geq 2\|v' - v^{KL}\|^2$, therefore we know $\|v' - v^{KL}\| \leq (\delta_1 + \delta_2)/\sqrt{l}$.

Now we follow the proof of Lemma 19 and define $u = \sum_{i=1}^K c_i v'_i$, then we know $\|u - v^{KL}\| \leq \|u - v'\| + \|v' - v^{KL}\| \leq 2(\delta_1 + \delta_2)/\sqrt{l}$, and similar to Lemma 19 we know $|c_i - c'_i| \leq 4(\delta_1 + \delta_2)/\gamma\sqrt{l}$. \square

We can simply replace Lemma 19 with this Lemma and get provable guarantee of RecoverKL. However, the argument here is not tight (in particular it gives worse bound than ℓ_2).

G.3 Empirical results

This section contains plots for ℓ_1 , held-out probability, coherence, and uniqueness for all semi-synthetic data sets. Up is better for all metrics except ℓ_1 error. The advantage of the non-negative recovery methods over the original Recover method on the real data is consistent with the results observed on the semi-synthetic data. For example, one can compare the mean log likelihood on real NY Times data from Figure 5 of the main paper (100 topics; 236k docs) with the semi-synthetic NY Times data shown in Figure 3 of the supplementary materials (100 topics; 250k docs). The values for the real data are [Recover: -8.42, RecoverL2: -8.16, Gibbs -7.93] and for semi-synthetic are [Recover: -8.23, RecoverL2: -8.08, Gibbs: -8.076].

G.3.1 Sample Topics

Tables G.1, G.2, and G.3 show 100 topics trained on real NY Times articles using the RecoverL2 algorithm. Each topic is followed by the most similar topic (measured by ℓ_1 distance) from a model trained on the same documents with Gibbs sampling. When the anchor word is among the top six words by probability it is highlighted in bold. Note that the anchor word is frequently not the most prominent word.

RecoverL2	run inning game hit season zzz_anaheim_angel
Gibbs	run inning hit game ball pitch
RecoverL2	king goal game team games season
Gibbs	point game team play season games
RecoverL2	yard game play season team touchdown
Gibbs	yard game season team play quarterback
RecoverL2	point game team season games play
Gibbs	point game team play season games
RecoverL2	zzz_laker point zzz_kobe_bryant zzz_o_neal game team
Gibbs	point game team play season games
RecoverL2	point game team season player zzz_clipper
Gibbs	point game team season play zzz_usc
RecoverL2	ballot election court votes vote zzz_al_gore
Gibbs	election ballot zzz_florida zzz_al_gore votes vote
RecoverL2	game zzz_usc team play point season
Gibbs	point game team season play zzz_usc
RecoverL2	company billion companies percent million stock
Gibbs	company million percent billion analyst deal
RecoverL2	car race team season driver point
Gibbs	race car driver racing zzz_nascar team
RecoverL2	zzz_dodger season run inning right game
Gibbs	season team baseball game player yankees
RecoverL2	palestinian zzz_israeli zzz_israel official attack zzz_palestinian
Gibbs	palestinian zzz_israeli zzz_israel attack zzz_palestinian zzz_yasser_arafat
RecoverL2	zzz_tiger_wood shot round player par play
Gibbs	zzz_tiger_wood shot golf tour round player
RecoverL2	percent stock market companies fund quarter
Gibbs	percent economy market stock economic growth
RecoverL2	zzz_al_gore zzz_bill_bradley campaign president zzz_george_bush vice
Gibbs	zzz_al_gore zzz_george_bush campaign presidential republican zzz_john_mccain
RecoverL2	zzz_george_bush zzz_john_mccain campaign republican zzz_republican
Gibbs	voter zzz_al_gore zzz_george_bush campaign presidential republican zzz_john_mccain
RecoverL2	net team season point player zzz_jason_kidd
Gibbs	point game team play season games
RecoverL2	yankees run team season inning hit
Gibbs	season team baseball game player yankees
RecoverL2	zzz_al_gore zzz_george_bush percent president campaign zzz_bush
Gibbs	zzz_al_gore zzz_george_bush campaign presidential republican zzz_john_mccain
RecoverL2	zzz_enron company firm zzz_arthur_andersen companies lawyer
Gibbs	zzz_enron company firm accounting zzz_arthur_andersen financial
RecoverL2	team play game yard season player
Gibbs	yard game season team play quarterback
RecoverL2	film movie show director play character
Gibbs	film movie character play minutes hour
RecoverL2	zzz_taliban zzz_afghanistan official zzz_u_s government military
Gibbs	zzz_taliban zzz_afghanistan zzz_pakistan afghan zzz_india government
RecoverL2	palestinian zzz_israel israeli peace zzz_yasser_arafat leader

Gibbs	palestinian zzz_israel peace israeli zzz_yasser_arafat leader
RecoverL2	point team game shot play zzz_celtic
Gibbs	point game team play season games
RecoverL2	zzz_bush zzz_mccain campaign republican tax zzz_republican
Gibbs	zzz_al_gore zzz_george_bush campaign presidential republican zzz_john_mccain
RecoverL2	zzz_met run team game hit season
Gibbs	season team baseball game player yankees
RecoverL2	team game season play games win
Gibbs	team coach game player season football
RecoverL2	government war zzz_slobodan_milosevic official court president
Gibbs	government war country rebel leader military
RecoverL2	game set player zzz_pete_sampras play won
Gibbs	player game match team soccer play
RecoverL2	zzz_al_gore campaign zzz_bradley president democratic zzz_clinton
Gibbs	zzz_al_gore zzz_george_bush campaign presidential republican zzz_john_mccain
RecoverL2	team zzz_knicks player season point play
Gibbs	point game team play season games
RecoverL2	com web www information sport question
Gibbs	palm beach com statesman daily american

Table G.1: Example topic pairs from NY Times sorted by ℓ_1 distance, anchor words in bold.

RecoverL2	season team game coach play school
Gibbs	team coach game player season football
RecoverL2	air shower rain wind storm front
Gibbs	water fish weather storm wind air
RecoverL2	book film beginitalic enditalic look movie
Gibbs	film movie character play minutes hour
RecoverL2	zzz_al_gore campaign election zzz_george_bush zzz_florida president
Gibbs	zzz_al_gore zzz_george_bush campaign presidential republican zzz_john_mccain
RecoverL2	race won horse zzz_kentucky_derby win winner
Gibbs	horse race horses winner won zzz_kentucky_derby
RecoverL2	company companies zzz_at percent business stock
Gibbs	company companies business industry firm market
RecoverL2	company million companies percent business customer
Gibbs	company companies business industry firm market
RecoverL2	team coach season player jet job
Gibbs	team player million season contract agent
RecoverL2	season team game play player zzz_cowboy
Gibbs	yard game season team play quarterback
RecoverL2	zzz_pakistan zzz_india official group attack zzz_united_states
Gibbs	zzz_taliban zzz_afghanistan zzz_pakistan afghan zzz_india government
RecoverL2	show network night television zzz_nbc program
Gibbs	film movie character play minutes hour
RecoverL2	com information question zzz_eastern commentary daily
Gibbs	com question information zzz_eastern daily commentary
RecoverL2	power plant company percent million energy
Gibbs	oil power energy gas prices plant

RecoverL2	cell stem research zzz_bush human patient
Gibbs	cell research human scientist stem genes
RecoverL2	zzz_governor_bush zzz_al_gore campaign tax president plan
Gibbs	zzz_al_gore zzz_george_bush campaign presidential republican zzz_john_mccain
RecoverL2	cup minutes add tablespoon water oil
Gibbs	cup minutes add tablespoon teaspoon oil
RecoverL2	family home book right com children
Gibbs	film movie character play minutes hour
RecoverL2	zzz_china chinese zzz_united_states zzz_taiwan official government
Gibbs	zzz_china chinese zzz_beijing zzz_taiwan government official
RecoverL2	death court law case lawyer zzz_texas
Gibbs	trial death prison case lawyer prosecutor
RecoverL2	company percent million sales business companies
Gibbs	company companies business industry firm market
RecoverL2	dog jump show quick brown fox
Gibbs	film movie character play minutes hour
RecoverL2	shark play team attack water game
Gibbs	film movie character play minutes hour
RecoverL2	anthrax official mail letter worker attack
Gibbs	anthrax official letter mail nuclear chemical
RecoverL2	president zzz_clinton zzz_white_house zzz_bush official zzz_bill_clinton
Gibbs	zzz_bush zzz_george_bush president administration zzz_white_house zzz_dick_cheney
RecoverL2	father family zzz_elian boy court zzz_miami
Gibbs	zzz_cuba zzz_miami cuban zzz_elian boy protest
RecoverL2	oil prices percent million market zzz_united_states
Gibbs	oil power energy gas prices plant
RecoverL2	zzz_microsoft company computer system window software
Gibbs	zzz_microsoft company companies cable zzz_at zzz_internet
RecoverL2	government election zzz_mexico political zzz_vicente_fox president
Gibbs	election political campaign zzz_party democratic voter
RecoverL2	fight zzz_mike_tyson round right million champion
Gibbs	fight zzz_mike_tyson ring fighter champion round
RecoverL2	right law president zzz_george_bush zzz_senate zzz_john_ashcroft
Gibbs	election political campaign zzz_party democratic voter
RecoverL2	com home look found show www
Gibbs	film movie character play minutes hour
RecoverL2	car driver race zzz_dale_earnhardt racing zzz_nascar
Gibbs	night hour room hand told morning
RecoverL2	book women family called author woman
Gibbs	film movie character play minutes hour

Table G.2: Example topic pairs from NY Times sorted by ℓ_1 distance, anchor words in bold.

RecoverL2	tax bill zzz_senate billion plan zzz_bush
Gibbs	bill zzz_senate zzz_congress zzz_house legislation zzz_white_house
RecoverL2	company francisco san com food home
Gibbs	palm beach com statesman daily american
RecoverL2	team player season game zzz_john_rocker right

Gibbs	season team baseball game player yankees
RecoverL2 Gibbs	zzz.bush official zzz.united.states zzz.u.s president zzz.north.korea zzz.united.states weapon zzz.iraq nuclear zzz.russia zzz.bush
RecoverL2 Gibbs	zzz.russian zzz.russia official military war attack government war country rebel leader military
RecoverL2 Gibbs	wine wines percent zzz.new.york com show film movie character play minutes hour
RecoverL2 Gibbs	police zzz.ray.lewis player team case told police officer gun crime shooting shot
RecoverL2 Gibbs	government group political tax leader money government war country rebel leader military
RecoverL2 Gibbs	percent company million airline flight deal flight airport passenger airline security airlines
RecoverL2 Gibbs	book ages children school boy web book author writer word writing read
RecoverL2 Gibbs	corp group president energy company member palm beach com statesman daily american
RecoverL2 Gibbs	team tour zzz.lance.armstrong won race win zzz.olympic games medal gold team sport
RecoverL2 Gibbs	priest church official abuse bishop sexual church religious priest zzz.god religion bishop
RecoverL2 Gibbs	human drug company companies million scientist scientist light science planet called space
RecoverL2 Gibbs	music zzz.napster company song com web palm beach com statesman daily american
RecoverL2 Gibbs	death government case federal official zzz.timothy.mcveigh trial death prison case lawyer prosecutor
RecoverL2 Gibbs	million shares offering public company initial company million percent billion analyst deal
RecoverL2 Gibbs	buy panelist thought flavor product ounces food restaurant chef dinner eat meal
RecoverL2 Gibbs	school student program teacher public children school student teacher children test education
RecoverL2 Gibbs	security official government airport federal bill flight airport passenger airline security airlines
RecoverL2 Gibbs	company member credit card money mean zzz.enron company firm accounting zzz.arthur.andersen financial
RecoverL2 Gibbs	million percent bond tax debt bill million program billion money government federal
RecoverL2 Gibbs	million company zzz.new.york business art percent art artist painting museum show collection
RecoverL2 Gibbs	percent million number official group black palm beach com statesman daily american
RecoverL2 Gibbs	company tires million car zzz.ford percent company companies business industry firm market
RecoverL2 Gibbs	article zzz.new.york misstated company percent com palm beach com statesman daily american
RecoverL2 Gibbs	company million percent companies government official company companies business industry firm market
RecoverL2 Gibbs	official million train car system plan million program billion money government federal
RecoverL2	test student school look percent system

Gibbs	patient doctor cancer medical hospital surgery
RecoverL2	con una mas dice las anos
Gibbs	fax syndicate article com information con
RecoverL2	por con una mas millones como
Gibbs	fax syndicate article com information con
RecoverL2	las como zzz.latin.trade articulo telefono fax
Gibbs	fax syndicate article com information con
RecoverL2	los con articulos telefono representantes zzz.america.latina
Gibbs	fax syndicate article com information con
RecoverL2	file sport read internet email zzz.los.angeles
Gibbs	web site com www mail zzz.internet

Table G.3: Example topic pairs from NY Times sorted by ℓ_1 distance, anchor words in bold.

G.4 Algorithmic details

G.4.1 Generating Q matrix

For each document, let H_d be the vector in \mathbb{R}^V such that the i -th entry is the number of times word i appears in document d , n_d be the length of the document and W_d be the topic vector chosen according to Dirichlet distribution when the documents are generated. Conditioned on W_d 's, our algorithms require the expectation of Q to be $\frac{1}{M} \sum_{d=1}^M AW_dW_d^T A^T$.

In order to achieve this, similar to Anandkumar et al. (2012a), let the normalized vector $\tilde{H}_d = \frac{H_d}{\sqrt{n_d(n_d-1)}}$ and diagonal matrix $\hat{H}_d = \frac{\text{Diag}(H_d)}{n_d(n_d-1)}$. Compute the matrix

$$\tilde{H}_d \tilde{H}_d^T - \hat{H}_d = \frac{1}{n_d(n_d-1)} \sum_{i \neq j, i, j \in [n_d]} e_{z_{d,i}} e_{z_{d,j}}^T.$$

Here $z_{d,i}$ is the i -th word of document d , and $e_i \in \mathbb{R}^V$ is the basis vector. From the generative model, the expectation of all terms $e_{z_{d,i}} e_{z_{d,j}}^T$ are equal to $AW_dW_d^T A^T$, hence by linearity of expectation we know $\mathbf{E}[\tilde{H}_d \tilde{H}_d^T - \hat{H}_d] = AW_dW_d^T A^T$.

If we collect all the column vectors \tilde{H}_d to form a large sparse matrix \tilde{H} , and compute the sum of all \hat{H}_d to get the diagonal matrix \hat{H} , we know $Q = \tilde{H} \tilde{H}^T - \hat{H}$ has the desired

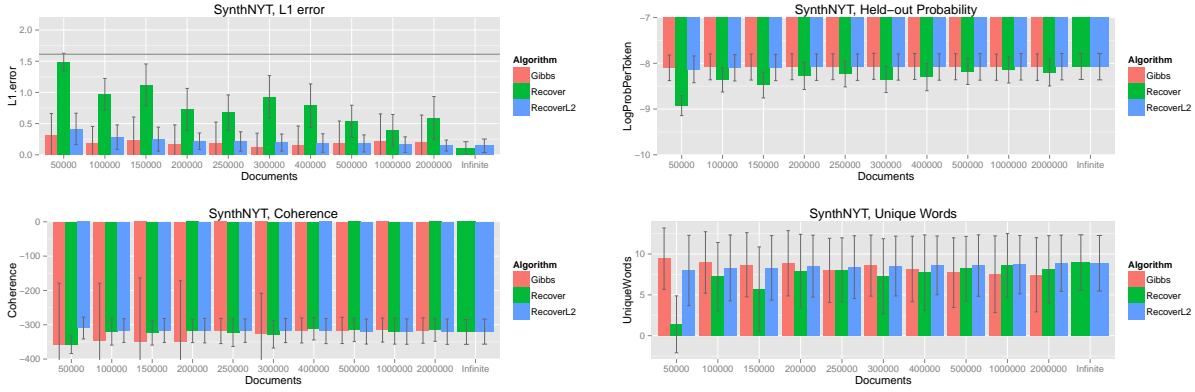


Figure G.3: Results for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$.

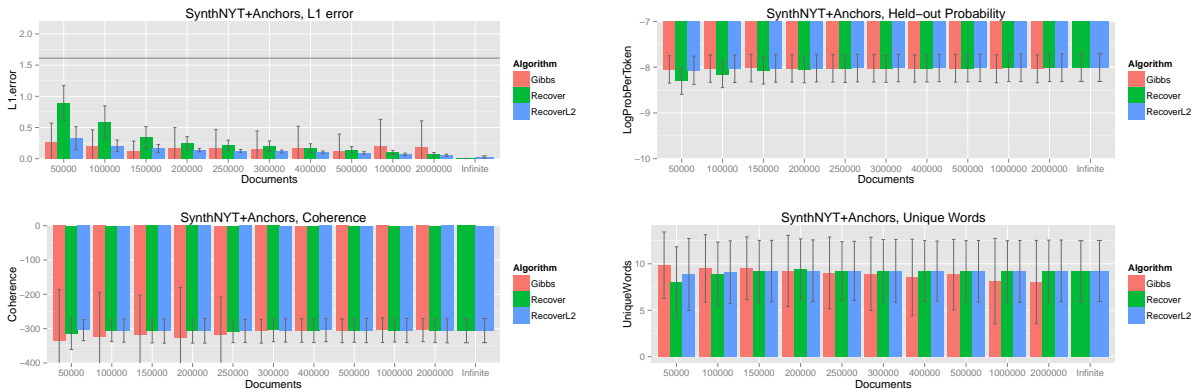


Figure G.4: Results for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$, with a synthetic anchor word added to each topic.

expectation. The running time of this step is $O(MD^2)$ where D^2 is the expectation of the length of the document squared.

G.4.2 Applying recover to small datasets

The original Recover algorithm from Arora et al. (2012b) can fail on small datasets if the $Q_{S,S}$ matrix which holds the anchor-anchor co-occurrence counts is rank deficient due to sparsity. When Recover fails, we use a modified version of the algorithm, solving for \vec{z}

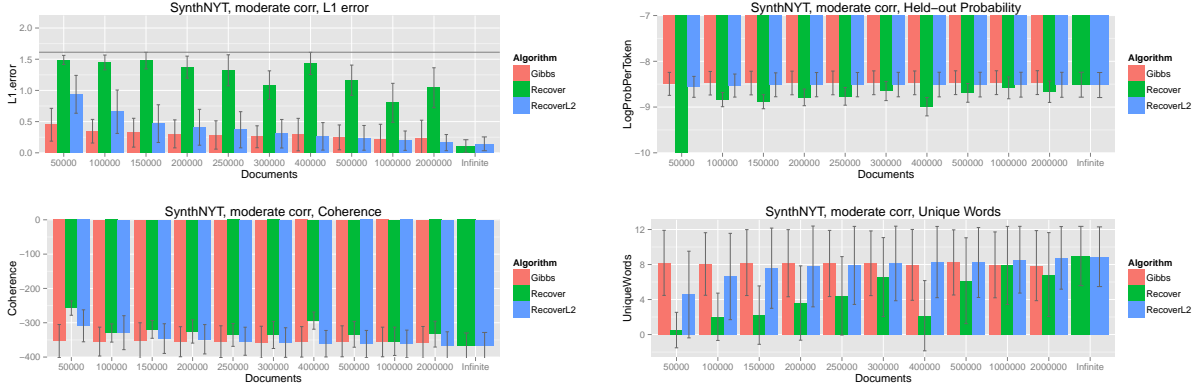


Figure G.5: Results for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$, with moderate correlation between topics.

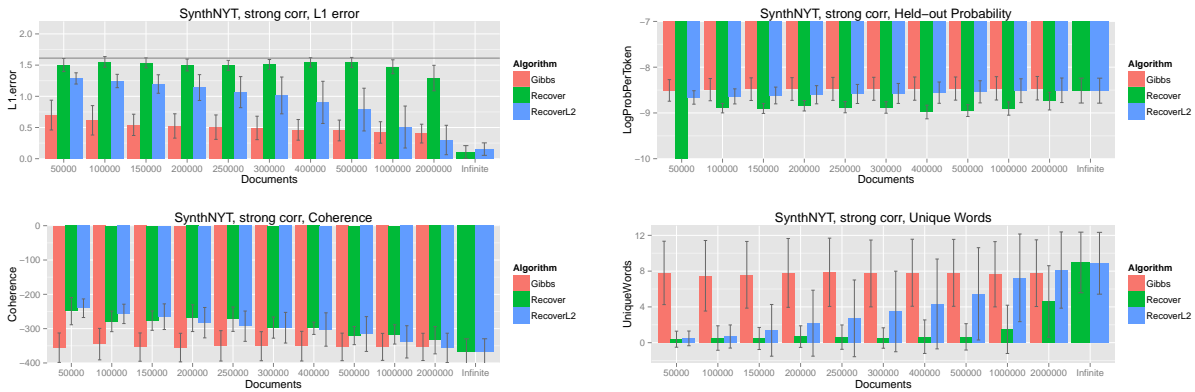


Figure G.6: Results for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$, with stronger correlation between topics.

by finding a least squares solution to $Q_{\mathbf{S},\mathbf{S}}\vec{z} = \vec{p}_{\mathbf{S}}$ and solving for A^T with a pseudoinverse: $A^T = (Q_{\mathbf{S},\mathbf{S}}\text{Diag}(\vec{z}))^\dagger Q_{\mathbf{S}}^T$. This procedure can return an A matrix in which some columns contain all 0s. In that case we replace columns of 0s with a uniform distribution over the vocabulary words, $\frac{1}{V}\mathbf{1}$.

Negative values also often occur in the A matrix returned by the original Recover method. To project back onto the simplex, we clip all negative values to 0 and normalize the columns before evaluating the learned model.

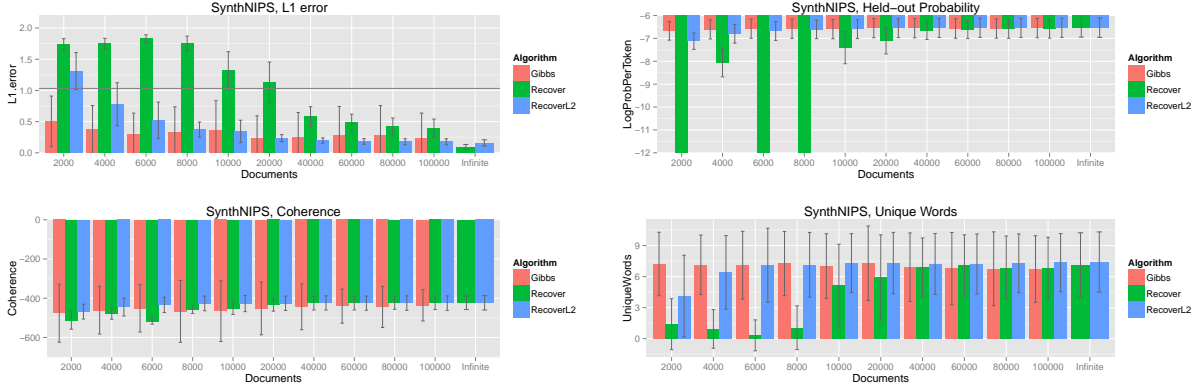


Figure G.7: Results for a semi-synthetic model generated from a model trained on NIPS papers with $K = 100$. For $D \in \{2000, 6000, 8000\}$, Recover produces log probabilities of $-\infty$ for some held-out documents.

G.4.3 Exponentiated gradient algorithm

The optimization problem that arises in RecoverKL and RecoverL2 has the following form:

$$\begin{aligned} & \min_{\vec{x}} d(\bar{Q}_i^T, \bar{Q}_S \vec{x}) \\ & \text{subject to: } \vec{x} \geq 0 \text{ and } \sum_{i=1}^K x_i = 1, \end{aligned}$$

where $d(\cdot, \cdot)$ is a Bregman divergence (in particular it is squared Euclidean distance for RecoverL2 and KL divergence for RecoverKL), \vec{x} is a column vector of size K , \mathbf{S} is the set of K anchor indices, \bar{Q}_i is a row vector of size V , and \bar{Q}_S is the $K \times V$ matrix formed by stacking the rows of \bar{Q} corresponding to the indices in \mathbf{S} .

This is a convex optimization problem with simplex constraints, which can be solved with the Exponentiated Gradient algorithm Kivinen and Warmuth (1995), described in Algorithm 12. The Exponentiated Gradient algorithm iteratively generates values of \vec{x} which are feasible and converge to the optimal value \vec{x}^* . In our experiments we show results using

Algorithm 12 Exponentiated Gradient

Input: Matrix \bar{Q}_S , vector \bar{Q}_i^T , divergence measure $d(\cdot, \cdot)$, tolerance parameter ϵ
Output: non-negative normalized vector \vec{x} close to \vec{x}^* , the minimizer of $d(\bar{Q}_i^T, \bar{Q}_S \vec{x})$

$\vec{x}_0 \leftarrow \frac{1}{K} \mathbf{1}$
 $t \leftarrow 0$
Converged \leftarrow False
while not Converged **do**
 $t \leftarrow t + 1$
 $\vec{g}_t = \nabla_{\vec{x}} d(\bar{Q}_i^T, \bar{Q}_S \vec{x})|_{\vec{x}_{t-1}}$
 Choose a step size η_t
 $\vec{x}_t \leftarrow \vec{x}_{t-1} e^{-\eta_t \vec{g}_t}$ (Gradient step)
 $\vec{x}_t \leftarrow \frac{\vec{x}}{|\vec{x}_t|_1}$ (Projection onto the simplex)
 $\mu_t \leftarrow -\min \left(\nabla_{\vec{x}} d(\bar{Q}_i^T, \bar{Q}_S \vec{x})|_{\vec{x}_t} \right)$
 $\vec{\lambda}_t \leftarrow \nabla_{\vec{x}} d(\bar{Q}_i^T, \bar{Q}_S \vec{x})|_{\vec{x}_t} + \mu_t \mathbf{1}$
 Converged $\leftarrow \vec{\lambda}_t^T \vec{x}_t < \epsilon$
end while

return x_t

both squared Euclidean distance and KL divergence for the divergence measure.

To determine whether the algorithm has converged, we test whether the KKT conditions (which are sufficient for optimality in this problem) hold to within some tolerance, ϵ . In our experiments ϵ varies between 10^{-6} and 10^{-9} depending on the data set.

The KKT conditions for our constrained minimization problem are:

1. Stationarity: $\nabla_{\vec{x}} d(\bar{Q}_i^T, \bar{Q}_S \vec{x}) - \vec{\lambda} + \mu \mathbf{1} = 0$.
2. Primal Feasibility: $\vec{x} \geq 0$, $\sum_{i=1}^K x_i = 1$.
3. Dual Feasibility: $\lambda_i \geq 0$ for $i \in \{1, 2, \dots, K\}$.
4. Complementary Slackness: $\lambda_i x_i = 0$ for $i \in \{1, 2, \dots, K\}$.

We define the following approximation to Condition 4:

- 4'. ϵ -Complementary Slackness: $0 \leq \vec{\lambda}^T \vec{x} < \epsilon$.

Let \vec{x}_t be the t^{th} value generated by Exponentiated Gradient. \vec{x}_t is ϵ -optimal if there exist $\vec{\lambda}$ and μ such that Conditions 1-3 and 4' are satisfied.

We initialize $\vec{x}_0 = \frac{1}{K}\mathbf{1}$ and Exponentiated Gradient preserves primal feasibility, so \vec{x}_t satisfies Condition 2. The following $\vec{\lambda}_t$ and μ_t minimize $\vec{\lambda}_t^T \vec{x}_t$ while satisfying conditions 1 and 3:

$$\begin{aligned}\mu_t &= -\min\left(\nabla_{\vec{x}}d(\bar{Q}_i^T, \bar{Q}_{\mathbf{S}}\vec{x})\Big|_{\vec{x}_t}\right) \\ \vec{\lambda}_t &= \nabla_{\vec{x}}d(\bar{Q}_i^T, \bar{Q}_{\mathbf{S}}\vec{x})\Big|_{\vec{x}_t} + \mu_t\mathbf{1}.\end{aligned}$$

The algorithm converges when Condition 4' is satisfied (i.e. $\vec{\lambda}_t^T \vec{x}_t < \epsilon$).

$\vec{\lambda}_t^T \vec{x}_t$ can also be understood as the gap between an upper and lower bound on the objective. To see this, note that the Lagrangian function is:

$$L(\vec{x}, \vec{\lambda}, \mu) = d(\bar{Q}_i^T, \bar{Q}_{\mathbf{S}}\vec{x}) - \vec{\lambda}^T \vec{x} + \mu(\vec{x}^T \mathbf{1} - 1),$$

The first term in the Lagrangian is exactly the primal objective, and $(\vec{x}_t^T \mathbf{1} - 1)$ is zero at every iteration. Since the Lagrangian lower bounds the objective, $\vec{\lambda}_t^T \vec{x}_t$ is the value of the gap. Strong duality holds for this problem, so at optimality, this gap is 0. Testing that the gap is less than ϵ is an approximate optimality test.

Stepsizes at each iteration are chosen with a line search to find an η_t that satisfies the Wolfe and Armijo conditions (For details, see Nocedal and Wright (2006)).

The running time of RecoverL2 is the time of solving V small ($K \times K$) quadratic programs. When using Exponentiated Gradient to solve the quadratic program, each word requires $O(KV)$ time for preprocessing and $O(K^2)$ per iteration. The total running time is $O(KV^2 + K^2VT)$ where T is the average number of iterations. The value of T is about 100 – 1000 depending on data sets.

Bibliography

- Agarwal, V., Lependu, P., Podchiyska, T., Barber, R., Boland, M., Hripcsak, G., et al. (2014). Using narratives as a source to automatically learn phenotype models. In *Workshop on Data Mining for Medical Informatics*.
- Agarwal, V., Podchiyska, T., Banda, J. M., Goel, V., Leung, T. I., Minty, E. P., Sweeney, T. E., Gyang, E., and Shah, N. H. (2016). Learning statistical models of phenotypes using noisy labeled training data. *Journal of the American Medical Informatics Association*, page ocw028.
- Allman, E. S., Matias, C., and Rhodes, J. A. (2009). Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, 37(6A):3099–3132.
- Anandkumar, A., Chaudhuri, K., Hsu, D., Kakade, S., Song, L., and Zhang, T. (2011). Spectral methods for learning multivariate latent tree structure. *Proceedings of NIPS 24*, pages 2025–2033.
- Anandkumar, A., Foster, D., Hsu, D., Kakade, S., and Liu, Y. (2012a). Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation. In *NIPS*.
- Anandkumar, A., Foster, D., Hsu, D., Kakade, S., and Liu, Y.-K. (2012b). A spectral algorithm for latent dirichlet allocation. *Proceedings of NIPS 25*, pages 926–934.

- Anandkumar, A., Ge, R., Hsu, D., and Kakade, S. M. (2014a). A tensor approach to learning mixed membership community models. *The Journal of Machine Learning Research*, 15(1):2239–2312.
- Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. (2014b). Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832.
- Anandkumar, A., Hsu, D., Javanmard, A., and Kakade, S. (2013a). Learning linear bayesian networks with latent variables. In *Proceedings of the 30th international conference on machine learning*, pages 249–257.
- Anandkumar, A., Hsu, D., and Kakade, S. (2012c). A method of moments for mixture models and hidden markov models. In *COLT 2012*.
- Anandkumar, A., Hsu, D. J., Janzamin, M., and Kakade, S. M. (2013b). When are overcomplete topic models identifiable? uniqueness of tensor tucker decompositions with structured sparsity. In *Advances in Neural Information Processing Systems*, pages 1986–1994.
- Anandkumar, A., Javanmard, A., Hsu, D. J., and Kakade, S. M. (2013c). Learning linear bayesian networks with latent variables. In *Proceedings of ICML*, pages 249–257.
- Arnal, J.-M., Wysocki, M., Novotni, D., Demory, D., Lopez, R., Donati, S., Granier, I., Corno, G., and Durand-Gasselini, J. (2012). Safety and efficacy of a fully closed-loop control ventilation (intellivent-asv®) in sedated icu patients with acute respiratory failure: a prospective randomized crossover study. *Intensive care medicine*, 38(5):781–787.
- Aronsky, D., Haug, P. J., Lagor, C., and Dean, N. C. (2005). Accuracy of administrative data for identifying patients with pneumonia. *American Journal of Medical Quality*, 20(6):319–328.

- Arora, S., Ge, R., Halpern, Y., Mimno, D., Moitra, A., Sontag, D., Wu, Y., and Zhu, M. (2013). A practical algorithm for topic modeling with provable guarantees. In *ICML*, volume 28 (2). JMLR: W&CP.
- Arora, S., Ge, R., Kannan, R., and Moitra, A. (2012a). Computing a nonnegative matrix factorization – provably. In *STOC*, pages 145–162.
- Arora, S., Ge, R., and Moitra, A. (2012b). Learning topic models – going beyond svd. In *FOCS*.
- Attenberg, J., Ipeirotis, P., and Provost, F. (2015). Beat the machine: Challenging humans to find a predictive model’s unknown unknowns. *Journal of Data and Information Quality (JDIQ)*, 6(1):1.
- Balcan, M.-f., Blum, A., and Yang, K. (2004). Co-training and expansion: Towards bridging theory and practice. In *Advances in Neural Information Processing Systems*, pages 89–96.
- Barnett, G. O., Cimino, J. J., Hupp, J. A., and Hoffer, E. P. (1987). Dxplain: an evolving diagnostic decision-support system. *Jama*, 258(1):67–74.
- Bartholomew, D. J., Knott, M., and Moustaki, I. (2011). *Latent variable models and factor analysis: A unified approach*, volume 904. John Wiley & Sons.
- Belanger, D., Sheldon, D., and McCallum, A. (2013). Marginal inference in mrfs using frank-wolfe. *NIPS Workshop on Greedy Optimization, Frank-Wolfe and Friends*.
- Berge, J. M. T. (1991). Kruskal’s polynomial for $2 \times 2 \times 2$ arrays and a generalization to $2 \times n \times n$ arrays. *Psychometrika*, 56:631–636.
- Berner, E. S. and Moss, J. (2005). Informatics challenges for the impending patient information explosion. *Journal of the American Medical Informatics Association*, 12(6):614–617.

- Birman-Deych, E., Waterman, A. D., Yan, Y., Nilasena, D. S., Radford, M. J., and Gage, B. F. (2005). Accuracy of icd-9-cm codes for identifying cardiovascular and stroke risk factors. *Medical care*, pages 480–485.
- Bisson, L. J., Komm, J. T., Bernas, G. A., Fineberg, M. S., Marzo, J. M., Rauh, M. A., Smolinski, R. J., and Wind, W. M. (2014). Accuracy of a computer-based diagnostic program for ambulatory patients with knee pain. *The American journal of sports medicine*, page 0363546514541654.
- Blei, D. (2012). Introduction to probabilistic topic models. *Communications of the ACM*, pages 77–84.
- Blei, D. and Lafferty, J. (2007). A correlated topic model of science. *Annals of Applied Statistics*, pages 17–35.
- Blei, D., Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, pages 993–1022. Preliminary version in *NIPS* 2001.
- Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.
- Buntine, W. L. (2009). Estimating likelihoods for topic models. In *Asian Conference on Machine Learning*.
- Carroll, R. J., Eyler, A. E., and Denny, J. C. (2011). Naive electronic health record phenotype identification for rheumatoid arthritis. In *AMIA Annu Symp Proc*, volume 2011, pages 189–196.
- Chaganty, A. and Liang, P. (2014). Estimating latent-variable graphical models using moments and likelihoods. In *International Conference on Machine Learning (ICML)*.

- Chang, J. T. (1996). Full reconstruction of markov models on evolutionary trees: identifiability and consistency. *Mathematical biosciences*, 137(1):51–73.
- Chapman, W. W., Bridewell, W., Hanbury, P., Cooper, G. F., and Buchanan, B. G. (2001). A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics*, 34(5):301 – 310.
- Charles, D., Gabriel, M., and Furukawa, M. F. (2013). Adoption of electronic health record systems among us non-federal acute care hospitals: 2008-2012. *ONC data brief*, 9:1–9.
- Chickering, D. (1996). Learning Bayesian networks is NP-Complete. In Fisher, D. and Lenz, H., editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag.
- Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467.
- Christoffersson, A. (1975). Factor analysis of dichotomized variables. *Psychometrika*, 40(1):5–32.
- Cipparone, C. W., Withiam-Leitch, M., Kimminau, K. S., Fox, C. H., Singh, R., and Kahn, L. (2015). Inaccuracy of icd-9 codes for chronic kidney disease: A study from two practice-based research networks (pbrns). *The Journal of the American Board of Family Medicine*, 28(5):678–682.
- Clavieras, N., Wysocki, M., Coisel, Y., Galia, F., Conseil, M., Chanques, G., Jung, B., Arnal, J.-M., Matecki, S., Molinari, N., et al. (2013). Prospective randomized crossover study of a new closed-loop control system versus pressure support during weaning from mechanical ventilation. *The Journal of the American Society of Anesthesiologists*, 119(3):631–641.

- Cohen, S. B., Stratos, K., Collins, M., Foster, D. P., and Ungar, L. H. (2013). Experiments with spectral learning of latent-variable pcfgs. In *HLT-NAACL*, pages 148–157.
- Conway, M., Berg, R., Carrell, D., Denny, J., Kho, A., Kullo, I., Linneman, J., Pacheco, J., Peissig, P., Rasmussen, L., et al. (2011). Analyzing the heterogeneity and complexity of electronic health record oriented phenotyping algorithms. In *AMIA... Annual Symposium proceedings/AMIA Symposium. AMIA Symposium*, volume 2011, pages 274–283.
- Cooper, G. F. (1987). Probabilistic inference using belief networks is NP-hard. Technical Report BMIR-1987-0195, Medical Computer Science Group, Stanford University.
- Courville, A. C., Eck, D., and Bengio, Y. (2009). An infinite factor model hierarchy via a noisy-or mechanism. In *Advances in Neural Information Processing Systems*, pages 405–413.
- Crosslin, D. R., McDavid, A., Weston, N., Nelson, S. C., Zheng, X., Hart, E., De Andrade, M., Kullo, I. J., McCarty, C. A., Doheny, K. F., et al. (2012). Genetic variants associated with the white blood cell count in 13,923 subjects in the emerge network. *Human genetics*, 131(4):639–652.
- Cussens, J. and Bartlett, M. (2013). Advances in bayesian network learning using integer programming. *UAI*.
- De Dombal, F., Leaper, D., Staniland, J. R., McCann, A., and Horrocks, J. C. (1972). Computer-aided diagnosis of acute abdominal pain. *Br Med J*, 2(5804):9–13.
- Dean, B. B., Lam, J., Natoli, J. L., Butler, Q., Aguilar, D., and Nordyke, R. J. (2009). Review: Use of electronic medical records for health outcomes research a literature review. *Medical Care Research and Review*, 66(6):611–638.

- Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Statist. Soc. Ser. B*, pages 1–38.
- Denny, J. C., Crawford, D. C., Ritchie, M. D., Bielinski, S. J., Basford, M. A., Bradford, Y., Chai, H. S., Bastarache, L., Zuvich, R., Peissig, P., et al. (2011). Variants near foxe1 are associated with hypothyroidism and other thyroid conditions: using electronic medical records for genome-and phenome-wide studies. *The American Journal of Human Genetics*, 89(4):529–542.
- Denny, J. C., Ritchie, M. D., Crawford, D. C., Schildcrout, J. S., Ramirez, A. H., Pulley, J. M., Basford, M. A., Masys, D. R., Haines, J. L., and Roden, D. M. (2010). Identification of genomic predictors of atrioventricular conduction using electronic medical records as a tool for genome science. *Circulation*, 122(20):2016–2021.
- Ding, W., Rohban, M. H., Ishwar, P., and Saligrama, V. (2014). Efficient distributed topic modeling with provable guarantees. *JMLR*, pages 167–175.
- Donoho, D. and Stodden, V. (2003). When does non-negative matrix factorization give the correct decomposition into parts? In *NIPS*.
- Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l_1 -ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM.
- Elidan, G., Lotner, N., Friedman, N., and Koller, D. (2001). Discovering hidden variables: A structure-based approach. *Advances in Neural Information Processing Systems*, pages 479–485.
- Elkan, C. and Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *KDD*.

- Elsner, L. (1985). An optimal bound for the spectral variation of two matrices. *Linear algebra and its applications*, 71:77–80.
- Fan, X., Yuan, C., and Malone, B. (2014). Tightening bounds for bayesian network structure learning. In *AAAI*, Quebec City, Quebec.
- Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. *13th International Joint Conference on Artificial Intelligence*, 2:1022–1027.
- Ferrucci, D. and Brown, E. (2011). Adaptwatson: A methodology for developing and adapting watson technology. *IBM, Armonk, NY, IBM Res. Rep., RC25244*.
- Finlayson, S. G., LePendou, P., and Shah, N. H. (2014). Building the graph of medicine from millions of clinical narratives. *Scientific data*, 1.
- Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:309–368.
- Fleurence, R. L., Curtis, L. H., Califf, R. M., Platt, R., Selby, J. V., and Brown, J. S. (2014). Launching pcoronet, a national patient-centered clinical research network. *Journal of the American Medical Informatics Association*, 21(4):578–582.
- Ford, E., Carroll, J. A., Smith, H. E., Scott, D., and Cassell, J. A. (2016). Extracting information from the text of electronic medical records to improve case detection: a systematic review. *Journal of the American Medical Informatics Association*, page ocv180.
- Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

- Friedman, C. and Elhadad, N. (2014). Natural language processing in health care and biomedicine. In *Biomedical Informatics*, pages 255–284. Springer.
- Gandhi, T. K., Zuccotti, G., and Lee, T. H. (2011). Incomplete care on the trail of flaws in the system. *New England Journal of Medicine*, 365(6):486–488.
- Gann, B. (2011). Giving patients choice and control: health informatics on the patient journey. *Yearbook of medical informatics*, 7:70–73.
- Glasziou, P. and Haynes, B. (2005). The paths from research to improved health outcomes. *Evidence Based Nursing*, 8(2):36–38.
- Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220.
- Gomez, C., Borgne, H. L., Allemand, P., Delacourt, C., and Ledru, P. (2007). N-findr method versus independent component analysis for lithological identification in hyperspectral imagery. *Int. J. Remote Sens.*, 28(23).
- Goodwin, T. and Harabagiu, S. M. (2013). Automatic generation of a qualified medical knowledge graph and its usage for retrieving patient cohorts from electronic medical records. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 363–370. IEEE.
- Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235.
- Groopman, J. E. and Prichard, M. (2007). *How doctors think*. Springer.

- Gurobi Optimization, I. (2014). Gurobi optimizer reference manual.
- Halpern, Y., Choi, Y., Horng, S., and Sontag, D. (2014). Using anchors to estimate clinical state without labeled data. In *Proceedings of the American Medical Informatics Association Annual Symposium*, pages 606–615.
- Halpern, Y., Horng, S., Choi, Y., and Sontag, D. (2016). Electronic medical record phenotyping using the anchor and learn framework. *Journal of the American Medical Informatics Association*.
- Halpern, Y., Horng, S., and Sontag, D. (2015). Anchored discrete factor analysis. *arXiv preprint arXiv:1511.03299*.
- Halpern, Y. and Sontag, D. (2013). Unsupervised learning of noisy-or bayesian networks. In *Conference on Uncertainty in Artificial Intelligence (UAI-13)*.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction 2 edition springer. *New York*.
- Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243. Technical Report MSR-TR-94-09.
- Heckerman, D. E. (1990). *A tractable inference algorithm for diagnosing multiple diseases*. Knowledge Systems Laboratory, Stanford University.
- Hemmerling, T. M. and Charabti, S. (2009). Mcsleepy—a completely automatic anesthesia delivery system. *Anesthesiology A*, 460.
- Henrion, M. (1989). Some Practical Issues in Constructing Belief Networks. In Kanal, L. N., Levitt, T. S., and Lemmer, J. F., editors, *Uncertainty in Artificial Intelligence 3*, pages 161–173. North-Holland.

- Henry, K. E., Hager, D. N., Pronovost, P. J., and Saria, S. (2015). A targeted real-time early warning score (trewscore) for septic shock. *Science Translational Medicine*, 7(299):299ra122–299ra122.
- Hider, P. N., Griffin, G., Walker, M., and Coughlan, E. (2009). The information-seeking behavior of clinical staff in a large health care organization. *Journal of the Medical Library Association*, 97(1):47.
- Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. W. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347.
- Hsu, D., Kakade, S. M., and Liang, P. (2012). Identifiability and unmixing of latent parse trees. In *Advances in Neural Information Processing Systems (NIPS)*.
- Ishteva, M., Park, H., and Song, L. (2013). Unfolding latent tree structures using 4th order tensors. In *ICML '13*.
- Jaakkola, T., Sontag, D., Globerson, A., and Meila, M. (2010). Learning bayesian network structure using lp relaxations. In *International Conference on Artificial Intelligence and Statistics*, pages 358–365.
- Jaakkola, T. S. and Jordan, M. I. (1999). Variational probabilistic inference and the qmr-dt network. *Journal of Artificial Intelligence Research*, 10:291–322.
- Jaggi, M. (2013). Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML*.
- Janda, M., Simanski, O., Bajorat, J., Pohl, B., Noeldge-Schomburg, G., and Hofmockel, R. (2011). Clinical evaluation of a simultaneous closed-loop anaesthesia control system for depth of anaesthesia and neuromuscular blockade. *Anaesthesia*, 66(12):1112–1120.

- Jensen, P. B., Jensen, L. J., and Brunak, S. (2012). Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405.
- Jernite, Y., Halpern, Y., Horng, S., and Sontag, D. (2013a). Predicting chief complaints at triage time in the emergency department. *NIPS Workshop on Machine Learning for Clinical Data Analysis and Healthcare*.
- Jernite, Y., Halpern, Y., and Sontag, D. (2013b). Discovering hidden variables in noisy-or networks using quartet tests. In *Advances in Neural Information Processing Systems 26*. MIT Press.
- Jones, S. (1975). Report on the need for and provision of an “ideal” information retrieval test collection.
- Kearns, M. and Mansour, Y. (1998). Exact inference of hidden structure from sample data in noisy-or networks. In *Proceedings of UAI 14*, pages 304–310.
- Kho, A. N., Hayes, M. G., Rasmussen-Torvik, L., Pacheco, J. A., Thompson, W. K., Armstrong, L. L., Denny, J. C., Peissig, P. L., Miller, A. W., Wei, W.-Q., et al. (2012). Use of diverse electronic medical record systems to identify genetic risk for type 2 diabetes within a genome-wide association study. *Journal of the American Medical Informatics Association*, 19(2):212–218.
- Kho, A. N., Pacheco, J. A., Peissig, P. L., Rasmussen, L., Newton, K. M., Weston, N., Crane, P. K., Pathak, J., Chute, C. G., Bielinski, S. J., et al. (2011). Electronic medical records for genetic research: results of the emerge consortium. *Science translational medicine*, 3(79):79re1–79re1.
- Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence,

- N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589.
- Kivinen, J. and Warmuth, M. K. (1995). Exponentiated gradient versus gradient descent for linear predictors. *Inform. and Comput.*, 132.
- Knol, D. L. and ten Berge, J. M. (1989). Least-squares approximation of an improper correlation matrix by a proper one. *Psychometrika*, 54(1):53–61.
- Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*.
- Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 8(42):30–37.
- Kruskal, J. B. (1989). Rank, decomposition, and uniqueness for 3-way and n-way arrays. *Multiway data analysis*, 33:7–18.
- Kullo, I. J., Ding, K., Jouni, H., Smith, C. Y., and Chute, C. G. (2010). A genome-wide association study of red blood cell traits using the electronic medical record. *PloS one*, 5(9):e13011.
- Kullo, I. J., Ding, K., Shameer, K., McCarty, C. A., Jarvik, G. P., Denny, J. C., Ritchie, M. D., Ye, Z., Crosslin, D. R., Chisholm, R. L., et al. (2011). Complement receptor 1 gene variants are associated with erythrocyte sedimentation rate. *The American Journal of Human Genetics*, 89(1):131–138.
- Kumar, A., Sindhvani, V., and Kambadur, P. (2012). Fast conical hull algorithms for near-separable non-negative matrix factorization. <http://arxiv.org/abs/1210.1190v1>.
- Lally, A., Bachi, S., Barborak, M. A., Buchanan, D. W., Chu-Carroll, J., Ferrucci, D. A., Glass, M. R., Kalyanpur, A., Mueller, E. T., Murdock, J. W., et al. (2014). Watsonpaths:

- scenario-based question answering and inference over unstructured information. *Yorktown Heights: IBM Research*.
- Lam, W. and Bacchus, F. (1994). Learning bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence*, 10:269–294.
- Larochelle, H. and Murray, I. (2011). The neural autoregressive distribution estimator. In *AISTATS*, volume 1, page 2.
- Lau, E. C., Mowat, F. S., Kelsh, M. A., Legg, J. C., Engel-Nitz, N. M., Watson, H. N., Collins, H., Nordyke, R., and Whyte, J. (2011). Use of electronic medical records (emr) for oncology outcomes research: assessing the comparability of emr information to patient registry and health claims data. *Clin Epidemiol*, 3(1):259–272.
- Lazarsfeld, P. (1950). Latent structure analysis. In Stouffer, S., Guttman, L., Suchman, E., Lazarsfeld, P., Star, S., and Clausen, J., editors, *Measurement and Prediction*. Princeton University Press, Princeton, New Jersey.
- Leeper, D., Horrocks, J. C., Staniland, J., and De Dombal, F. (1972). Computer-assisted diagnosis of abdominal pain using estimates provided by clinicians. *British Medical Journal*, 4(5836):350.
- Lee, M., Bindel, D., and Mimno, D. (2015). Robust spectral inference for joint stochastic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2710–2718.
- Lee, M. and Mimno, D. (2014). Low-dimensional embeddings for interpretable anchor-based topic inference. In *EMNLP*.
- Li, W. and McCallum, A. (2007). Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML*, pages 633–640.

- Liao, K. P., Cai, T., Savova, G. K., Murphy, S. N., Karlson, E. W., Ananthakrishnan, A. N., Gainer, V. S., Shaw, S. Y., Xia, Z., Szolovits, P., et al. (2015). Development of phenotype algorithms using electronic medical records and incorporating natural language processing. *bmj*, 350:h1885.
- Lin, J., Jiao, T., Biskupiak, J. E., and McAdam-Marx, C. (2013). Application of electronic medical record data for health outcomes research: a review of recent literature. *Expert review of pharmacoeconomics & outcomes research*, 13(2):191–200.
- Lingren, T. (2013). Autism.
- Lipton, Z. (2016). The mythos of model interpretability. *ICML Workshop on Human Interpretability in Machine Learning*.
- Liu, M., Hinz, E. R. M., Matheny, M. E., Denny, J. C., Schildcrout, J. S., Miller, R. A., and Xu, H. (2013). Comparative analysis of pharmacovigilance methods in the detection of adverse drug reactions using electronic medical records. *Journal of the American Medical Informatics Association*, 20(3):420–426.
- Liu, M., Shah, A., Jiang, M., Peterson, N. B., Dai, Q., Aldrich, M. C., Chen, Q., Bowton, E. A., Liu, H., Denny, J. C., et al. (2012). A study of transportability of an existing smoking status detection module across institutions. In *AMIA Annual Symposium Proceedings*, volume 2012, page 577. American Medical Informatics Association.
- Longhurst, C. A., Harrington, R. A., and Shah, N. H. (2014). A green button for using aggregate patient data at the point of care. *Health Affairs*, 33(7):1229–1235.
- Martin, J. and VanLehn, K. (1995). Discrete factor analysis: Learning hidden variables in bayesian networks. Technical report, Department of Computer Science, University of Pittsburgh.

- McCallum, A. (2002). Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.
- McCormick, P. J., Elhadad, N., and Stetson, P. D. (2008). Use of semantic features to classify patient smoking status. In *AMIA Annual Symposium Proceedings*, volume 2008, page 450. American Medical Informatics Association.
- Miller, R. and Masarie Jr, F. (1989). Use of the quick medical reference (qmr) program as a tool for medical education. *Methods of information in medicine*, 28(4):340–345.
- Miller, R. A. (1994). Medical diagnostic decision support systems past, present, and future. *Journal of the American Medical Informatics Association*, 1(1):8–27.
- Miller, R. A. and Masarie, F. E. (1990). The demise of the "greek oracle" model for medical diagnostic systems. *Methods of information in medicine*, 29(1):1–2.
- Miller, R. A., McNeil, M. A., Challinor, S. M., Fred E. Masarie, J., and Myers, J. D. (1986). The internist-1/quick medical reference project – status report. *West J Med*, 145:816–822.
- Miller, R. A., Pople, H. E., and Myers, J. D. (1982). Internist-i, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307(8):468–476.
- Mimno, D., Wallach, H., Talley, E., Leenders, M., and McCallum, A. (2011). Optimizing semantic coherence in topic models. In *EMNLP*.
- Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1791–1799.
- Mo, H., Thompson, W. K., Rasmussen, L. V., Pacheco, J. A., Jiang, G., Kiefer, R., Zhu, Q., Xu, J., Montague, E., Carrell, D. S., et al. (2015). Desiderata for computable representations

- of electronic health records-driven phenotype algorithms. *Journal of the American Medical Informatics Association*, 22(6):1220–1230.
- Morris, Q. (2001). Anonymised qmr kb to aqmr-dt.
- Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.
- Nascimento, J. P. and Dias, J. M. B. (2004). Vertex component analysis: A fast algorithm to unmix hyperspectral data. *IEEE TRANS. GEOSCI. REM. SENS*, 43:898–910.
- Natarajan, N., Dhillon, I., Ravikumar, P., and Tewari, A. (2013). Learning with noisy labels. In *NIPS*.
- Neal, R. M. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.
- Neamatullah, I., Douglass, M. M., Li-wei, H. L., Reisner, A., Villarroel, M., Long, W. J., Szolovits, P., Moody, G. B., Mark, R. G., and Clifford, G. D. (2008). Automated de-identification of free-text medical records. *BMC medical informatics and decision making*, 8(1):1.
- Newton, K. M., Peissig, P. L., Kho, A. N., Bielinski, S. J., Berg, R. L., Choudhary, V., Basford, M., Chute, C. G., Kullo, I. J., Li, R., et al. (2013). Validation of electronic medical record-based phenotyping algorithms: results and lessons learned from the emerge network. *Journal of the American Medical Informatics Association*, 20(e1):e147–e154.
- Nguyen, T., Hu, Y., and Boyd-Graber, J. (2014). Anchors regularized: Adding robustness and extensibility to scalable topic-modeling algorithms. In *ACL*.
- Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, 2nd edition.

- O'malley, K. J., Cook, K. F., Price, M. D., Wildes, K. R., Hurdle, J. F., and Ashton, C. M. (2005). Measuring diagnoses: Icd code accuracy. *Health services research*, 40(5p2):1620–1639.
- Onisko, A., Druzdzal, M. J., and Wasyluk, H. (1999). A bayesian network model for diagnosis of liver disorders. In *Proceedings of the Eleventh Conference on Biocybernetics and Biomedical Engineering*, volume 2, pages 842–846. Citeseer.
- Overby, C. L., Pathak, J., Gottesman, O., Haerian, K., Perotte, A., Murphy, S., Bruce, K., Johnson, S., Talwalkar, J., Shen, Y., et al. (2013). A collaborative approach to developing an electronic health record phenotyping algorithm for drug-induced liver injury. *Journal of the American Medical Informatics Association*, 20(e2):e243–e252.
- Panella, M., Marchisio, S., and Di Stanislao, F. (2003). Reducing clinical variations with clinical pathways: do pathways work? *International Journal for Quality in Health Care*, 15(6):509–521.
- Paparrizos, J., White, R. W., and Horvitz, E. (2016). Screening for pancreatic adenocarcinoma using signals from web search logs: Feasibility study and results. *Journal of Oncology Practice*, page JOPR010504.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- Pearl, J. (2009). *Causality*. Cambridge university press.
- Pearl, J. and Tarsi, M. (1986). Structuring causal trees. *Journal of Complexity*, 2(1):60–77.
- Pearl, J. and Verma, T. (1991). A theory of inferred causation. In *KR*, pages 441–452.
- Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110.

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.
- Peng, J., Hazan, T., Srebro, N., and Xu, J. (2012). Approximate inference by intersecting semidefinite bound and local polytope. In *Proceedings of AISTATS*, pages 868–876.
- Pinchin, V. (2016 (accessed Sept. 8, 2016)). *Im Feeling Yucky :(Searching for symptoms on Google*.
- Platanios, E. A., Blum, A., and Mitchell, T. M. (2014). Estimating Accuracy from Unlabeled Data. In *Conference on Uncertainty in Artificial Intelligence*, pages 1–10.
- Provan, J. S. and Ball, M. O. (1983). The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788.
- Ramaswami, P. (2015 (accessed Sept. 8, 2016)). *A remedy for your health-related questions: health info in the Knowledge Graph*.
- Ramnarayan, P., Kulkarni, G., Tomlinson, A., and Britto, J. (2004). Isabel: a novel internet-delivered clinical decision support system. *Current perspectives in healthcare computing*, pages 245–256.
- Razavian, N., Blecker, S., Schmidt, A. M., Smith-McLallen, A., Nigam, S., and Sontag, D. (2015). Population-level prediction of type 2 diabetes from claims data and analysis of risk factors. *Big Data*, 3(4):277–287.
- Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.
- Richesson, R., Horvath, M., and Rusincovitch, S. (2014). Clinical research informatics and electronic health record data. *Yearbook of medical informatics*, 9(1):215.

- Richesson, R. L., Hammond, W. E., Nahm, M., Wixted, D., Simon, G. E., Robinson, J. G., Bauck, A. E., Cifelli, D., Smerek, M. M., Dickerson, J., et al. (2013). Electronic health records based phenotyping in next-generation clinical trials: a perspective from the nih health care systems collaboratory. *Journal of the American Medical Informatics Association*, 20(e2):e226–e231.
- Roberts, M. E., Stewart, B. M., and Tingley, D. (2014). Navigating the local modes of big data: The case of topic models. In *Data Science for Politics, Policy and Government*.
- Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432.
- Salakhutdinov, R. (2009). *Learning deep generative models*. PhD thesis, University of Toronto.
- Sanchez, M., Bouveret, S., Givry, S. D., Heras, F., Jgou, P., Larrosa, J., Ndiaye, S., Rollon, E., Schiex, T., Terrioux, C., Verfaillie, G., and Zytnicki, M. (2008). Max-csp competition 2008: toulbar2 solver description.
- Sawade, C., Landwehr, N., Bickel, S., and Scheffer, T. (2010). Active risk estimation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 951–958.
- Scott, C., Blanchard, G., and Handy, G. (2013). Classification with asymmetric label noise: Consistency and maximal denoising. In Shalev-Shwartz, S. and Steinwart, I., editors, *COLT*, volume 30 of *JMLR Proceedings*, pages 489–511. JMLR.org.
- Semigran, H. L., Linder, J. A., Gidengil, C., and Mehrotra, A. (2015). Evaluation of symptom checkers for self diagnosis and triage: audit study. *BMJ: British Medical Journal*, 351.
- Shaban, A., Farajtabar, M., Xie, B., Song, L., and Boots, B. (2015). Learning latent variable

- models by improving spectral solutions with exterior point methods. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI-2015)*.
- Shivade, C., Raghavan, P., Fosler-Lussier, E., Embi, P. J., Elhadad, N., Johnson, S. B., and Lai, A. M. (2014). A review of approaches to identifying patient phenotype cohorts using electronic health records. *Journal of the American Medical Informatics Association*, 21(2):221–230.
- Shwe, M. A., Middleton, B., Heckerman, D., Henrion, M., Horvitz, E., Lehmann, H., and Cooper, G. (1991). Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base. *Meth. Inform. Med*, 30:241–255.
- Silva, R., Scheine, R., Glymour, C., and Spirtes, P. (2006). Learning the structure of linear latent variable models. *The Journal of Machine Learning Research*, 7:191–246.
- Šingliar, T. and Hauskrecht, M. (2006). Noisy-or component analysis and its application to link analysis. *The Journal of Machine Learning Research*, 7:2189–2213.
- Sondhi, P., Sun, J., Tong, H., and Zhai, C. (2012). Sympgraph: a framework for mining clinical notes through symptom relation graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1167–1175. ACM.
- Sontag, D. and Jaakkola, T. (2007). New outer bounds on the marginal polytope. In *NIPS 20*. MIT Press.
- Sontag, D. and Roy, D. (2011). Complexity of inference in latent dirichlet allocation. In *NIPS*, pages 1008–1016.
- Spirtes, P., Glymour, C., and Scheines, R. (2001). *Causation, Prediction, and Search, 2nd Edition*. The MIT Press.

- Stang, P. E., Ryan, P. B., Dusetzina, S. B., Hartzema, A. G., Reich, C., Overhage, J. M., and Racoosin, J. A. (2012). Health outcomes of interest in observational data: issues in identifying definitions in the literature. *Health Outcomes Research in Medicine*, 3(1):e37–e44.
- Steinhardt, J. and Liang, P. (2016). Unsupervised risk estimation with only structural assumptions.
- Stevens, K., Kegelmeyer, P., Andrzejewski, D., and Buttler, D. (2012). Exploring topic coherence over many models and many topics. In *EMNLP*.
- Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., and Fergus, R. (2014). Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.
- Tang, H. and Ng, J. H. K. (2006). Googling for a diagnosis use of google as a diagnostic aid: internet based study. *Bmj*, 333(7579):1143–1145.
- Teyssier, M. and Koller, D. (2005). Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 584–590, Edinburgh, Scotland, UK.
- Thurau, C., Kersting, K., and Bauckhage, C. (2010). Yes we can simplex volume maximization for descriptive web-scale matrix factorization. In *CIKM-10*.
- Tieder, J. S., Hall, M., Auger, K. A., Hain, P. D., Jerardi, K. E., Myers, A. L., Rahman, S. S., Williams, D. J., and Shah, S. S. (2011). Accuracy of administrative billing codes to detect urinary tract infection hospitalizations. *Pediatrics*, 128(2):323–330.
- Torr, P. H. (2003). Solving markov random fields using semi definite programming.
- Tucker, L. R. (1958). An inter-battery method of factor analysis. *Psychometrika*, 23(2):111–136.

- Van Melle, W. (1978). Mycin: a knowledge-based consultation program for infectious disease diagnosis. *International Journal of Man-Machine Studies*, 10(3):313–322.
- Wainwright, M. and Jordan, M. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, pages 1–305.
- Wallach, H., Murray, I., Salakhutdinov, R., and Mimno, D. (2009). Evaluation methods for topic models. In *ICML*.
- Wang, X., Sontag, D., and Wang, F. (2014). Unsupervised learning of disease progression models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 85–94, New York, NY, USA. ACM.
- Warner, H. R., Haug, P., Bouhaddou, O., Lincoln, M., Warner Jr, H., Sorenson, D., Williamson, J. W., and Fan, C. (1988). Iliad as an expert consultant to teach differential diagnosis. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 371. American Medical Informatics Association.
- Wedin, P. (1972). Perturbation bounds in connection with singular value decomposition. *BIT Numerical Mathematics*, 12(1):99–111.
- Weiskopf, N., Rusanov, A., and Weng, C. (2013). Sick patients have more data. In *AMIA... Annual Symposium proceedings/AMIA Symposium. AMIA Symposium*, volume 2013, pages 1472–1477.
- White, R. W. and Horvitz, E. (2009). Cyberchondria: studies of the escalation of medical concerns in web search. *ACM Transactions on Information Systems (TOIS)*, 27(4):23.
- Wilke, R., Xu, H., Denny, J., Roden, D., Krauss, R., McCarty, C., Davis, R., Skaar, T., Lamba, J., and Savova, G. (2011). The emerging role of electronic medical records in pharmacogenomics. *Clinical Pharmacology & Therapeutics*, 89(3):379–386.

- Wood, F., Griffiths, T., and Ghahramani, Z. (2006). A non-parametric bayesian method for inferring hidden causes. In *UAI '06, Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence*.
- Wright, A., Pang, J., Feblowitz, J. C., Maloney, F. L., Wilcox, A. R., McLoughlin, K. S., Ramelson, H., Schneider, L., and Bates, D. W. (2012). Improving completeness of electronic problem lists through clinical decision support: a randomized, controlled trial. *Journal of the American Medical Informatics Association*, 19(4):555–561.
- Xu, H., Fu, Z., Shah, A., Chen, Y., Peterson, N. B., Chen, Q., Mani, S., Levy, M. A., Dai, Q., and Denny, J. C. (2011). Extracting and integrating data from entire electronic health records for detecting colorectal cancer cases. In *AMIA Annu Symp Proc*, volume 2011, pages 1564–1572.
- Yao, L., Mimno, D., and McCallum, A. (2009). Efficient methods for topic model inference on streaming document collections. In *KDD*.
- Zhou, T., Bilmes, J. A., and Guestrin, C. (2014). Divide-and-conquer learning by anchoring a conical hull. In *NIPS*, pages 1242–1250.
- Zobel, J. (1998). How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314. ACM.