Graphical Model Structure Learning with ℓ_1-Regularization

by

Mark Schmidt

B.Sc., The University of Alberta, 2003
M.Sc., The University of Alberta, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August 2010

Abstract

This work looks at fitting probabilistic graphical models to data when the structure is not known. The main tool to do this is ℓ_1-regularization and the more general group ℓ_1-regularization. We describe limited-memory quasi-Newton methods to solve optimization problems with these types of regularizers, and we examine learning directed acyclic graphical models with ℓ_1-regularization, learning undirected graphical models with group ℓ_1-regularization, and learning hierarchical log-linear models with overlapping group ℓ_1-regularization.

Table of Contents

List of Figures

List of Acronyms

- **#P-hard**: Non-deterministic counting polynomial-time hard.
- **AS**: Active set.
- **BBSG**: Barzilai-Borwein sub-gradient.
- **BBST**: Barzilai-Borwein soft-threshold.
- **BFGS**: Broyden-Fletcher-Goldfarb-Shanno.
- **BIC**: Bayesian information criterion.
- **CPD**: Conditional probability distribution.
- **CRF**: Conditional random field.
- **DAG**: directed acyclic graph.
- **DSST** diagonally scaled soft-threshold.
- **GGM**: Gaussian graphical model.
- **HLLM**: Hierarchical log-linear model.
- **IGM**: Ising graphical model.
- **L-BFGS**: Limited-memory Broyden-Fletcher-Goldfarb-Shanno.
- **LASSO**: Least absolute shrinkage and selection operator.
- **L1MB**: ℓ_1-Markov blanket.
- **MMHC**: Max-min hill-climbing.
- **MMPC**: Max-min parents and children.
- **NP-hard**: Non-deterministic polynomial-time hard.
- **OPG**: Optimal projected gradient.
- **OWL**: Orthant-wise learning.
- **PSS**: Projected scaled sub-gradient.
- **PSSas**: PSS active set.
- **PSSgb**: PSS Gafni-Bertsekas.
- **PSSsp**: PSS sign projection.
- **PQN**: Projected quasi-Newton.
- **QNST**: Quasi-Newton soft-threshold.
- **SC**: Sparse candidate.
- **SCAD**: Smoothly clipped absolute deviation.
- **SPG**: Spectral projected gradient.
- **TMP**: Two-metric projection.

Acknowledgements

I would first like to thank my supervisor Kevin Murphy for sharing his knowledge, pushing me to work hard and constantly try to improve my work, and giving me the freedom to explore a diverse set of projects. I'd also like to thank my other supervisory committee members Michael Friedlander and Arnaud Doucet for their help and advice, as well as my other 'unnofficial' supervisors Russ Greiner, Albert Murtha, Glenn Fung, and Rómer Rosales. I would like to acknowledge my other co-authors for their help in letting me be more productive that I would have been able to on my own: Ewout van den Berg, Peter Carbonetto, Dana Cobzas, David Duvenaud, Daniel Eaton, Nando de Freitas, Emt Khan, Chi-Hoon Lee, Ilya Levner, Benjamin Marlin, Marianne Morris, Alexandru Niculescu-Mizil, Nic Schraudolph, Ian South-Dickinson, Jörg Sander, Kevin Swersky, Aline Tabet, and SVN Vishwanathan. The National Science and Engineering Research Council of Canada and the Li Tze Fong Memorial Fellowship provided funding for part of this work, while WestGrid provided computational resources. Finally and most importantly, I'd like to dedicate this thesis to my girlfriend Alisha, and my parents Joanne and Ken.

Chapter 1

Introduction

Graphical models [Whittaker, 1990, Lauritzen, 1996, Koller and Friedman, 2009] are used as efficient representations for probability distributions in a wide variety of applications. In many cases, the graphical structure describing the dependencies in the model is known. However, in some applications it is not clear what graphical structure should be used. Alternately, we may want to model a data set using a graphical model but not assume a particular graphical structure *a priori*. In this thesis we examine the problem of estimating the parameters of a graphical model given a data set, when the graphical structure is not given.

One approach to this task is to assume a graphical model where all possible interactions are present (a *dense* model), and estimate the parameters of this model given the data set. An alternative approach is to try and find a *sparse* set of edges that optimize a criterion assessing the quality of the structure. There are several reasons why we might prefer the sparse approach:

- **Statistical efficiency**: Since there are fewer parameters in the sparse model, we may be able to estimate them more effectively. For example, the number of parameters to be estimated in the dense model will grow quadratically or exponentially (depending on the particular model) in the number of variables present in the data. In contrast, the number of parameters needed by a sparse model might be much smaller.

- **Computational efficiency**: Due to the smaller number of parameters in a sparse structure, typically it will be much less costly to estimate the parameters. Further, performing inference tasks in the graphical model will require quadratic, cubic, or exponential time (depending on the particular model) in the dense model, while it may be possible to do these tasks more efficiently in a sparse model.

- **Structural discovery**: If we believe the dependencies in our data set can be accurately described within the class of graphical models we are searching over, then we might hope to find the 'true' structure that describes the dependencies in the data set. Even if the dependencies in the data set do not conform precisely to a particular graphical model, the edges discovered by a structure learning method may still be indicative of the dependencies (or independencies) present in the data. There has been substantial recent interest in this task due to applications in systems biology, such as [Sachs et al., 2005].

The disadvantage of taking the sparse approach is simply that there are an enormous number of possible structures. For example, in the case of the directed acyclic models we describe in Section 1.3, there are a super-exponential number of possible structures, and finding the optimal structure (under various definitions of optimality) is known to be NP-hard [Chickering, 1995]. Further, it can be computationally expensive to search through the space of graph structures. For example, in undirected graphical models we must re-fit all parameters if any edge is added or removed from the graph. Since fitting all parameters is typically computationally expensive, this means that even greedy apporaches that attempt to add/remove one edge at time are extremely

expensive. For these reason, in some scenarios we might want to consider fitting a *single* dense model, but use regularization to address the issue of statistical efficiency, use approximations to address the issue of computational efficiency, and try to interpret our estimates of the parameters for structural discovery.

In this work, we take an approach that is intermediate between fitting a single regularized dense model, and searching for an optimal sparse model. Specifically, we consider fitting a single dense model with a penalty on the ℓ_1-norm on the parameters. This ℓ_1-regularization has a sparsity-inducing property [Tibshirani, 1996, Chen et al., 1998]; if the penalty on the ℓ_1-norm is strong enough, then many of the parameters in the optimal solution will be zero. Further, we parameterize the dense graphical model such that if the parameters associated with an edge are set to zero, it is equivalent to removing the edge from the model. This allows us to learn a sparse graphical model by fitting a single dense graphical model. In addition to combining regularization and sparsity within a convex optimization framework, in Section 1.1 we discuss other appealing properties that are known about ℓ_1-regularization.

This idea of using ℓ_1-regularization to learn a sparse graphical model has recently been explored by various authors, and in this chapter we review related work on this topic. However, previous work on ℓ_1-regularization for structure learning has largely been used in very restricted scenarios. Specifically, nearly all of the previous work makes the assumptions that:

- The graphical model is undirected.
- There is a one-to-one correspondence between parameters and edges.
- The model only includes pairwise dependencies.

In Chapters 4, 5, and 6, we examine models that do not make these assumptions. Specifically, these chapters outline methods for structure learning using ℓ_1-regularization for the following scenarios:

- Chapter 4: Directed acyclic graphical models.
- Chapter 5: Undirected models with multi-parameter edges or edge groups.
- Chapter 6: Undirected models with higher-order dependencies.

Interspersed with our discussion of prior work, we discuss the motivations for examining these scenarios throughout the remainder of this chapter. In the latter two cases, we consider generalizations of ℓ_1-regularization that penalize groups of variables. In Chapter 2, we describe non-differentiable extensions of limited-memory quasi-Newton methods for solving the ℓ_1-regularization problems arising in Chapters 4 and 5, while in Chapter 3 we describe constrained and non-differentiable limited-memory quasi-Newton methods for solving the group ℓ_1-regularization problems arising in Chapters 5 and 6. Chapter 7 discusses some extensions of this work. Chapters 2-6 are based on (and extend) existing work. In particular, Chapter 2 is based on [Schmidt et al., 2007a], Chapter 3 is based on [Schmidt et al., 2009b], Chapter 4 is based on [Schmidt et al., 2007b], Chapter 5 is based on [Schmidt et al., 2008], and Chapter 6 is based on [Schmidt and Murphy, 2010].

The remainder of this chapter is structured as follows. First, in the next section we review using ℓ_1-regularization for variable selection in regression and classification. Next, we move on to using ℓ_1-regularization to learn dependency networks, a straightforward extension of the regression/classification methodology that allows us to visualize dependencies between variables, but that does not necessarily form a consistent probabilistic model. We then consider linearly-parameterized

directed acyclic graphical models, where we learn a dependency network under a variable ordering to yield a consistent probabilistic model. Subsequently we consider ℓ_1-regularization for structure learning in two special classes of undirected graphical models, namely Gaussian graphical models and pairwise Ising models of binary data. We then consider pairwise models of general discrete data, and higher-order log-linear models of discrete data. We then outline the data sets examined in this work, and finally we conclude the chapter with a summary of contributions.

1.1 Regression and Binary Classification

In regression, we have are given a set n real-valued targets y^i (for $i = 1, 2, \ldots, n$), and a corresponding set of n real-valued p-vectors that we denote by $\mathbf{x}^i$ (for $i = 1, 2, \ldots, n$). Our goal is to build a model that predicts y^i given the corresponding p-vector $\mathbf{x}^i$. Binary classification is similar, except that each y^i can only take values in the discrete set $\{-1, +1\}$. The most common regression method is the linear least-squares model [see Bishop, 2006, §3.1.1], where we assume that y^i is a linear function of $\mathbf{x}^i$ (and a bias term b), and we fit the parameters $\{\mathbf{w}, b\}$ of the model by minimizing the least-squares objective

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \frac{1}{2}(y^i - \mathbf{w}^T\mathbf{x}^i - b)^2.$$

The least-squares estimator can also be viewed as a maximum likelihood estimator, under the assumption that each y^i follows a Gaussian distribution with mean $\mathbf{w}^T\mathbf{x}^i + b$ and a positive variance σ (the exact value of σ does not affect the optimal values of $\{\mathbf{w}, b\}$). Formally,

$$p(y^i|\mathbf{x}^i, \mathbf{w}, b) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y^i - \mathbf{w}^T\mathbf{x}^i - b)^2}{2\sigma^2}\right).$$

We arrive at the least-squares objective if we consider minimizing the negative logarithm of the likelihood, $-\sum_{i=1}^{n} \log p(y^i|\mathbf{x}^i, \mathbf{w}, b)$, under this model with σ set to 1 and ignoring constant terms (the objective has the same minimizers for any other positive σ).

The most common binary classification method is logistic regression, where we assume that the logarithm of the odds of y^i taking on $+1$ (instead of -1) is a linear function of $\mathbf{w}^T\mathbf{x}^i + b$ [see Bishop, 2006, §4.3.2]. This implies that we assume y^i follows a logistic distribution with location $\mathbf{w}^T\mathbf{x}^i + b$ and scale 1:

$$p(y^i|\mathbf{x}^i, \mathbf{w}, b) = \frac{1}{1 + \exp(-y^i(\mathbf{w}^T\mathbf{x}^i + b))}.$$

Maximum likelihood estimation in this model is typically carried out by minimizing the negative log-likelihood,

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \log(1 + \exp(-y^i(\mathbf{w}^T\mathbf{x}^i + b))).$$

Unlike the least-squares objective, in general there will not be a closed-form solution for the parameters in the logistic regression model. However, we can obtain accurate numerical maximum likelihood estimates by minimizing the (differentiable, unconstrained, and convex) negative log-likelihood. Nevertheless, for both of these models there are several reasons why we might not want to use a maximum likelihood estimate of the parameters:

- The maximum likelihood estimate tends to have all coefficients w_i non-zero, even though it may be the case that some variables are irrelevant for prediction. If a variable is irrelevant for predicting y_i, then its value should be set to zero to nullify its effect on the prediction (and yield a more interpretable model).

- The maximum likelihood estimator may over-fit. That is, the average likelihood of the data used for estimation might be much higher than the average likelihood for data that was not used during estimation. This can arise if we do not have a sufficiently large sample size n (relative to the number of features p and their complexity), because in this case the maximum likelihood estimate of the parameters may have a high variance (the parameters can change substantially with small changes in the data). In the language of numerical computing, we say that estimating the parameters can be ill-posed.

Subset selection methods are a common strategy for addressing the first issue. That is, we do a search over the possible non-zero subsets of the coefficients, and choose the subset that optimizes some criteria judging the worthiness of the subset. Several heuristic strategies for doing the search exist such as forward and backward selection, but the general problem of choosing the best subset under most optimization criteria is known to be NP-hard [Huo and Ni, 2007]. Further, even if we were given the optimal subset this may not address the second issue with maximum likelihood estimation.

The most common method used to address the second issue is ℓ_2-regularization of the coefficients, known as Tikhonov regularization or ridge (logistic) regression [see Bishop, 2006, §3.1.4]. In ridge (logistic) regression, we optimize the negative log-likelihood subject to a penalty (with scale $\lambda > 0$) on the (squared) ℓ_2-norm of the regression coefficients $\mathbf{w}$:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} -\log p(y^i|\mathbf{x}^i, \mathbf{w}, b) + \lambda||\mathbf{w}||_2^2.$$

If we interpret the ℓ_2-regularization term as the negative logarithm of a prior, then we see that finding the ridge (logistic) regression parameters is equivalent to finding the parameters that maximize the posterior distribution, $p(y^i|\mathbf{x}^i, \mathbf{w}, b)p(\mathbf{w}, b)$, with a prior for the parameters $p(\mathbf{w}, b)$ that factorizes into an independent zero-mean Gaussian distribution for each element w_i, and an (improper) uniform distribution for b. The effect of this prior is to decrease the variance of the estimator, by adding a bias towards zero in the estimation of the coefficients. However, as with the maximum likelihood estimate the ℓ_2-regularized estimate tends to have all coefficients w_i non-zero.

In ℓ_1-regularized least-squares (or logistic regression), we minimize the negative log-likelihood subject to a penalty on the ℓ_1-norm of the coefficients:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} -\log p(y^i|\mathbf{x}^i, \mathbf{w}, b) + \lambda||\mathbf{w}||_1. \tag{1.1}$$

This type of regularization has been popularized under the name basis pursuit denoising [Chen et al., 1998] for the least-squares loss, and least absolute shrinkage and subset selection operator (LASSO) for the least-squares and logistic regression losses [Tibshirani, 1996]. Prior to these works ℓ_1-regularization had also been explored for the least-squares loss [Santosa and Symes, 1986] and least absolute error loss [Claerbout and Muir, 1973]. As opposed to problem (1.1), Tibshirani [1996]

proposed using an explicit bound τ on the ℓ_1 norm of the parameters leading to the problem

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} -\log p(y^i|\mathbf{x}^i, \mathbf{w}, b) \text{ s.t. } ||\mathbf{w}||_1 \leq \tau. \tag{1.2}$$

Problems (1.1) and (1.2) are very closely related; solving problem 1.1 is equivalent to minimizing the Lagrangian of (1.2) with a fixed Lagrange multiplier λ, and for any value of τ we can find a corresponding value of λ that gives the same solution. We focus on 1.1 since λ has the intuitive interpretation as the strength of a Laplace prior on the parameters.

In contrast to subset selection and ℓ_2-regularization, ℓ_1-regularization simultaneously achieves subset selection (by setting parameters w_i to 0 for sufficiently large λ) and regularization (by adding a bias towards zero in the estimation of the coefficients). Further, under suitable conditions and an appropriate choice of λ, ℓ_1-regularization will choose the correct subset of non-zero variables [for example, see Zhao and Yu, 2006]. Even if this structural discovery task is not the goal, ℓ_1-regularization is often still effective at building a regressor (or classifier) that predicts well on new examples, even if irrelevant features are present in the data. For example, Ng [2004] shows that ℓ_1-regularized logistic regression has an asymptotic sample complexity function that grows with the logarithm of the number of irrelevant features[1]. This means that ℓ_1-regularization can produce near-optimal models even if there are an exponential number of irrelevant features, in contrast to the linear sample complexity of ℓ_2-regularized logistic regression that would require an exponential number of samples to produce near-optimal models if there are an exponential number of irrelevant features. The generalization performance of logistic regression with ℓ_1-regularization is examined in [Krishnapuram et al., 2005], who prove non-trivial bounds on the generalization performance (i.e. bounds on the error obtained on data not seen during training).

When using ℓ_1-regularization, it is important to select an appropriate value of the hyper-parameter λ. There are a wide variety of criteria available to do this, but in this work we focus on two. The first criterion we consider is validation set likelihood, a score that tries to assess how effective the estimator is at modeling new instances. To compute the validation set likelihood for a fixed value of λ, we

1. randomly choose half of our data set;
2. compute the ℓ_1-regularized estimator on this half of the data set;
3. compute the likelihood of the other half of the data set with the estimated parameters.

The validation set likelihood gives us a criterion for assessing how well the estimator for a particular value of λ models $p(y^i|\mathbf{x}^i, \mathbf{w}, b)$ for new instances $\{\mathbf{x}^i, y^i\}$. We can choose a good value of λ by searching for a value that maximizes this validation score (where we use the same random half of the training data for each value of λ). In cases where the number n of training examples is small, a variation on the validation score is the cross-validation score [Bishop, 2006, §1.3], where we train on different subsets of the data. The advantage of this is that it makes greater use of the available data, but the disadvantages are that it is slower and it no longer represents an independent estimate of the generalization performance.

The validation score is used for many of our experiments, as it is typically accurate in assessing prediction error (assuming sufficient data is available to provide reliable estimates using half of the

[1]The sample complexity function is a two parameter function of (ϵ, δ), defined as the minimum number of training examples such that that we can be within ϵ of the optimal predictor with probability at least $1 - \delta$.

training data). However, as discussed in [Meinshausen and Buhlmann, 2006] the optimal parameters under the prediction-optimal value of λ will in general have too many non-zero variables. Because of this, we may want to consider a different criterion when the goal is structural discovery. When trying to do structural discovery, in some cases we will consider the Bayesian information criterion (BIC) [Schwarz, 1978],

$$BIC(y^i, \mathbf{x}^i, \hat{\mathbf{w}}, \hat{b}) \triangleq \sum_{i=1}^{n} -\log p(y^i|\mathbf{x}^i, \hat{\mathbf{w}}, \hat{b}) + (d/2)\log n.$$

Here, d is the number of free parameters in the model (i.e. the number of non-zero elements of $\mathbf{w}$, plus one for the bias), while $\hat{\mathbf{w}}$ and $\hat{b}$ are the maximum likelihood estimates for the set of non-zero coefficients. That is, it simultaneously tries to maximize the model fit of the training data while minimizing the number of free parameters used to do this. Schwarz [1978] derives this criterion as a large-sample approximation to the marginal likelihood of the data[2]. It can also be viewed as a large-sample approximation to a minimum description length criteria [Rissanen, 1978]. In particular, if we wish to compress the data set and model, optimizing the BIC approximates the optimal level of compression (as the size of the data set increases) [see Hastie et al., 2009, §7.8]. For exponential family models the BIC has appealing asymptotic consistency properties in terms of variable selection; if we compute the BIC on a set of models that includes the true model, the true model will achieve the lowest value as the size of the data set increases [Schwarz, 1978]. Further, [Haughton, 1988] shows that optimizing this criteria will choose the correct set of variables with probability tending to one as the size of the data set increases. The BIC can also be viewed from the perspective of regularization, in that it is equivalent to regularization by the ℓ_0 pseudo-norm, $||w||_0 \triangleq d$, where the regularization strength is chosen according to the size of the data set.

A complicating factor with using the BIC for selecting λ for ℓ_1-regularization is that the criterion is traditionally defined for the maximum likelihood estimate. Therefore, when we use BIC for model selection, we use ℓ_1-regularization as a filter; the ℓ_1-regularization is only used to select the set of non-zero variables, and we subsequently compute the maximum likelihood estimate of the coefficients when evaluating the BIC[3].

1.2 Dependency Networks

Now consider the unsupervised case where we are given a set of n real-valued p-vectors $\mathbf{x}^i$ (for $i = 1, 2, \ldots, n$) and no distinguished response variables y^i, and we want to build a graph that visualizes the direct dependencies between the variables. One way to do this is, for each variable j, we make variable j the target and compute the optimal parameters in a linear regression model $x_j^i = \mathbf{w}_j^T \mathbf{x}_{-j}^i + b_j$ (where we use $-j$ to denote all variables except j). This linear regression can be fit using the methods we describe in the previous section, and the sets of variables selected are used to draw a graph that visualizes dependencies in the data. Specifically, we draw these dependencies as a directed graph with p nodes (one for each variable), where the graph contains an edge going into each node from each of the variables that was selected when regressing on the node. The model

[2] In the case of regression with a linear least-squares loss and ℓ_2-regularization, it is possible to compute the marginal likelihood of the training data in closed form, but this is not possible in most scenarios.

[3] We discuss work on using ℓ_1-regularized estimates with the BIC at the end of Chapter 4.

resulting from doing this conditional regression (or classification) of each variable given all others is known as a dependency network [Heckerman et al., 2001].

Using ℓ_1-regularized least-squares to learn the structure of a dependency network on continuous variables was examined in [Meinshausen and Buhlmann, 2006][4]. Meinshausen and Buhlmann [2006] outline conditions under which this procedure is consistent in terms of variable selection in Gaussian graphical models (that we review in the next section), allowing the number of variables and density of the graph to increase as a function of the sample size. Analogously, Wainwright et al. [2006] proposed using ℓ_1-regularized logistic regression to learn the structure of a dependency network on binary variables, and examine consistency in terms of variable selection for Ising graphical models of binary data (that we also review in the next section).

While these approaches lead to a graph structure that may be useful in terms of visualization or structural discovery, they can be problematic as a probabilistic model of the p-vectors because given finite data the dependency network estimated in this way will typically be inconsistent. For example, Heckerman et al. [2001] give the simple case where the dependency network predicts that x_1 depends on x_2 in $p(x_1|x_2)$ but that x_2 does not depend on x_1 in $p(x_2|x_1)$. These inconsistencies can lead to cases where there may be no joint distribution over the variables that is consistent with the estimated conditional distributions. The set of *consistent* dependency networks is equivalent to the set of undirected graphical models [Heckerman et al., 2001], and these dependency network methods can be viewed as pseudo-likelihood approximations [Besag, 1975] of the corresponding undirected graphical models. To deal with potential structural asymmetry, Meinshausen and Buhlmann [2006], Wainwright et al. [2006] consider two heuristics to turn the directed graph into an undirected graph. Their first strategy includes the undirected edge if either corresponding directed edge was found, while the second strategy only includes the undirected edge if both directed edges were found. This still leaves the problem that we have two versions of the parameter associated with each edge, though [Hofling and Tibshirani, 2009] give two related heuristics for obtaining a single parameter.

Given that we can learn dependency networks with existing methods for regression and classification, it is useful at this point to discuss *why* we might want to use more complicated models that define (consistent) joint distributions over the p-vectors. Several of the tasks we can consider doing with a joint distribution (that can't be accomplished in general with inconsistent dependency networks) include:

- Compute joint probabilities: Given a new p-vector $\mathbf{x}$, we can try to assess its probability $p(x_1, x_2, \ldots, x_p)$ under the model. Similarly, we can check which of two p-vectors has a higher probability, search for the p-vector with highest probability (decoding), or test whether a p-vector has a very low probability (i.e. outlier detection).

- Compute marginals and conditionals: Given a distribution over $\mathbf{x}$, $p(x_1, x_2, \ldots, x_p)$, we can consider calculating marginal probabilities like $p(x_i)$, or conditional probabilities like $p(x_i|x_j)$, using the rules of marginalization and conditional probability.

- Generate samples: We can try to generate new p-vectors according to our estimate of the joint distribution. This can be useful for model assessment. We can also consider generating conditional samples from the distribution given the values of some of the variables (i.e. filling-in missing values).

[4] Gustafsson et al. [2003] present a closely related approach for estimating time-series dependencies.

While we can also perform these tasks by using heuristic methods to construct a consistent dependency network from the (typically inconsistent) result of learning a dependency network, pseudo-likelihood approximations are known to be inefficient estimators compared to using the likelihood of the probabilistic model explicitly [Besag, 1977, Liang and Jordan, 2008][5].

1.3 Directed Acyclic Graphical Models

As we see in the next section, the prior work on structure learning in probabilistic graphical models with ℓ_1-regularization largely focuses on pairwise undirected models. However, there are many reasons why we might prefer to use directed acyclic graph (DAG) models:

- Efficiency of computing joint probabilities, samples, and (approximate) marginals: As we have just mentioned, these types of operation are the main reasons for building a consistent model of the joint distribution. However, for discrete data these operations are intractable in general for pairwise undirected graphical models. In contrast, some of these operations can be done in polynomial time in analogous DAG models. This includes computing the probability of a vector and generating unbiased samples from the model. The latter can be used in Monte Carlo methods to efficiently approximate marginals in the model. Provided we condition on the first variables in an ordering, we can also efficiently compute the conditional probability of the remaining variables and generate unbiased conditional samples of the remaining variables (the latter can be used to efficiently approximate the corresponding conditionals).

- Parameter independence: The likelihood in DAG models factorizes into a product of single-variable conditional distributions. Thus, unlike undirected graphical models where estimating single-variable conditional distributions is used as an approximation, we can find the optimal parameters in the joint likelihood of DAG models by fitting the parameters of a set of single-variable conditional distributions. Further, DAGs allow us to mix different types of variables in a straightforward way. For example, we can model the joint likelihood of vectors containing both real-valued and binary-valued variables. Because parameter independence allows parameter estimation to separate into independent sub-problems, it also allows us to independently tune an individual regularization parameter λ_i in estimating the conditional of each variable i, and allows us to use caching of results to implement efficient local search methods for structure learning.

DAG models, also known as Bayesian networks, are one way to model the joint distribution $p(x_1, x_2, \ldots, x_p)$ of a set of p random variables. If we repeatedly use the definition of conditional probability, $p(x, y) = p(y|x)p(x)$, in the order n down to 1, then we obtain the factorization of the joint distribution

$$p(x_1, \ldots, x_p) = \prod_{i=1}^{p} p(x_i|\mathbf{x}_{1:i-1}).$$

This factorization of the joint distribution is valid for any probability distribution. In DAG models, we make the additional conditional independence assumption that

$$p(x_i|\mathbf{x}_{1:i-1}) = p(x_i|\mathbf{x}_{\pi(i)}), \tag{1.3}$$

[5]Here, the efficiency of a consistent estimator is defined as its asymptotic variance around the true parameter in terms of the number of training samples.

for some $\pi(i) \subseteq \{j|1 \leq j < i\}$. The elements of $\pi(i)$ are called the 'parents' of variable i (the 'child'), while the terms $p(x_i|\mathbf{x}_{\pi(i)})$ are referred to as the conditional probability distributions (CPDs) of the DAG model.

We can visualize the conditional independence properties implied by the variable ordering and the choices of $\pi(i)$ as a directed graph, where we draw a directed edge coming into each node from each of its parents. Since the order $1, \ldots, n$ will constitute a topological ordering of the graph (that is, an ordering where parents come before children), the graph is necessarily acyclic. In addition to the conditional independence properties directly encoded in the use of (1.3), the method of d-separation allows us to use the graph structure to test whether any other conditional independence statement is implied by the factorization [see Koller and Friedman, 2009, §3.3]. Note that it is possible for two different graphs to imply the same set of conditional independence statements about the distribution. In this case, we say that the graphs are Markov equivalent.

In this work, we focus on modeling binary data using logistic regression for the CPDs:

$$p(x_i|\mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i) = \frac{1}{1 + \exp(-x_i(\mathbf{w}_i^T \mathbf{x}_{\pi(i)} + b_i))}.$$

This is sometimes referred to as a *sigmoid belief network* [see, for example Saul et al., 1996], and we note that these are directed analogues to the Ising graphical models we discuss in the next section[6]. While sigmoid belief networks do not have the expressive power of the tabular CPDs traditionally used in DAG models [see Koller and Friedman, 2009, §5.1], their smaller number of parameters allows more efficient estimation of the parameters from data. Specifically, there are a linear number of parameters in the CPDs of a sigmoid belief net (in terms of the number of parents), rather than the exponential number associated with tabular CPDs. This allows us to fit DAG models where some nodes have a potentially large number of parents.

Using these CPDs, the dominant cost of evaluating the joint probability $p(\mathbf{x})$ of a vector $\mathbf{x}$ is the calculation of the p inner products $\mathbf{w}_i^T \mathbf{x}_{\pi(i)}$. In the worst case (a fully connected graph) this will require $\mathcal{O}(p^2)$, in contrast to the #P-hard problem of evaluating the probability of an observed p-vector in a general binary undirected model. Similarly, we can generate an independent sample from the distribution in $\mathcal{O}(p^2)$ (and a set of independent samples can be used to approximate any marginal). Further, if we use $|\mathcal{E}|$ to denote the number of edges in the graph, then computing all these operations is $\mathcal{O}(p + |\mathcal{E}|)$. This means that if there are more than p edges, then the cost of performing these operations is directly proportional to the sparsity of the graph structure.

In DAG models, the negative log-likelihood function for a set of n realizations of p-vectors $\mathbf{x}^i$ is given by

$$\sum_{i=1}^{n} \sum_{j=1}^{p} -\log p(x_j^i|\mathbf{x}_{\pi(j)}^i, \mathbf{w}_j, b_j).$$

This objective function is separable with respect to the parameters of the different CPDs. If the regularizer separates in the same way then we satisfy the parameter independence condition [Heckerman et al., 1995]. This means that we can optimize the parameters of each CPD independently. Thus, parameter estimation in this model is similar to the parameter estimation procedure used in dependency networks with logistic regression conditionals, except that each regression is done on the subset of nodes earlier in the ordering, and optimizing these independent logistic regressions

[6]While if we use Gaussian CPDs we obtain a directed model that is analogous to the Gaussian graphical models we discuss in the next section.

directly optimizes the joint likelihood of a consistent probabilistic model. That is, by placing a constraint on the variable ordering we guarantee that the parameters yield a consistent probabilistic model.

If we are given the variable ordering, then estimating the structure of a DAG model reduces to the problem of independently performing variable selection to select the parents of each node. Thus, we can learn sigmoid belief networks using ℓ_1-regularization by solving a series of independent ℓ_1-regularized logistic regression problems. Most previous work on structure learning in DAG models with ℓ_1-regularization has considered the case of a known ordering [Li and Yang, 2005, Huang et al., 2006, Levina et al., 2008][7]. However, in general we do not have a topological ordering available, and sub-optimal orderings may lead to models that are much more dense than the optimal ordering. Thus, in Chapter 4 we consider a method that uses ℓ_1-regularization for structure learning in sigmoid belief networks that does not assume a known topological ordering of the variables. The challenge associated with this problem is that if we relax the constraints imposed by the ordering (corresponding to fitting a dependency network) this typically leads to a structure violating the acyclicity constraint. Thus, our method uses a two-phase approach: in the first phase the method learns a dependency network using ℓ_1-regularization to obtain a set of candidate edges, then in the second phases it uses local search in the space of DAGs restricted to these candidates. Although various methods have been proposed for restricting the set of candidate edges in DAGs, an important aspect of the new algorithm is that the same criterion is used for variable selection in both phases. Although we state and evaluate the method for the case of sigmoid belief networks, it can trivially be applied to the case of Gaussian CPDs or other types of linearly-parameterized CPDs.

1.4 Gaussian and Ising Graphical Models

We now turn to the case of fitting two special types of pairwise undirected graphical models, and consider learning a sparse graph structure by using ℓ_1-regularization of the parameters corresponding to the edges in the graph. The advantage of working with undirected models is that we have no acyclicity constraint and thus we can use ℓ_1-regularization directly to estimate a sparse structure. However, the disadvantage of undirected models is that the log-likelihood does not separate into a set of independent problems, and this makes parameter estimation much more expensive.

In pairwise undirected graphical models, we model the joint distribution $p(x_1, x_2, \ldots, x_p)$ of a set of p random variables $\mathbf{x}$ as a globally normalized product of non-negative unary potentials $\phi_i(x_i)$ and non-negative pairwise potentials $\phi_{ij}(x_i, x_j)$:

$$p(\mathbf{x}) \triangleq \frac{1}{Z} \prod_{i=1}^{p} \phi_i(x_i) \prod_{(i,j)\in E}^{p} \phi_{ij}(x_i, x_j). \tag{1.4}$$

The normalizing constant Z is defined as the constant such that the distribution integrates to one over all possible assignments to $\mathbf{x}$. The set E contains the set of pairs of variables that we want to include a pairwise potential for. Typically, we will only include pairwise potentials for a relevant subset of the possible pairs of variables (but unlike DAG models that must be acyclic, we do not need to enforce any constraints on the set of edges in undirected models). If the potential $\phi_{ij}(x_i, x_j)$ is included in the model, we say that two nodes i and j are neighbors, and that in this case j is in

[7] These works use Gaussian CPDs instead of sigmoid CPDs.

the Markov blanket of node i (and vice versa). A local Markov property follows from the pairwise factorization (1.4) and the choice of pairwise potentials to include, namely that for node i with Markov blanket $MB(i)$ we have the conditional independence property

$$p(x_i|\mathbf{x}_{-i}) = p(x_i|\mathbf{x}_{MB(i)}).$$

Further, we can visualize *all* of the conditional independence properties implied by the factorization as an undirected graph, where each variable corresponds to a node in the graph and we place an undirected edge between each set of neighbors. Because of this we refer to unary potentials $\phi_i(x_i)$ as *node* potentials, and pairwise potentials $\phi_{ij}(x_i, x_j)$ as *edge* potentials. We can then test whether the factorization implies that two sets of variables are conditionally independent (given a third set of variables) by testing whether the conditioning set separates the two sets in this graph. For more details on the independence properties of undirected graphical models, we refer to [Koller and Friedman, 2009, §4.3].

Structure learning in pairwise undirected graphical models is the task of selecting the pairs of vairables to include as neighbors/edges in E. However, note that if a potential function $\phi_{ij}(x_i, x_j)$ takes the value 1 for all values of x_i and x_j, it is equivalent to removing the potential from the model[8]. Thus, if we parameterize our pairwise potentials such that zeros in the parameterization make $\phi_{ij}(x_i, x_j)$ take the value 1 for all x_i and x_j, then we can use ℓ_1-regularization of the fully-connected model to encourage that we learn a sparse graphical structure.

This idea was first explored for Gaussian graphical models (GGMs). In GGMs, we model the joint distribution $p(x_1, x_2, \ldots, x_p)$ of a set of p continuous random variables as a multivariate Gaussian with mean $\mathbf{b}$ and precision (inverse-covariance) matrix W:

$$p(x_1, x_2, \ldots, x_p) \triangleq \frac{1}{Z} \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{b})^T W(\mathbf{x} - \mathbf{b})). \tag{1.5}$$

In this case the normalizing constant Z is

$$Z \triangleq \frac{1}{(2\pi)^{p/2}|W^{-1}|^{1/2}}.$$

This normalization constant can be computed in$\mathcal{O}(p^3)$ time using a Cholesky factorization of W. If we expand out the quadratic form in (1.5) we see that GGMs are a special case of (1.4), and hence they are pairwise undirected graphical models. In particular, an edge is present between two variables i and j in a Gaussian graphical model if and only if the corresponding element W_{ij} of the precision matrix W is non-zero. This type of model was introduced in [Dempster, 1972], where it was referred to as covariance selection.

Because setting elements of the precision matrix to zero corresponds to removing edges from the graph, we can consider simultaneously estimating the parameters and a sparse structure in GGMs by minimizing the negative log-likelihood subject to ℓ_1-regularization of the elements of the precision matrix. This is referred to as the graphical LASSO by [Friedman et al., 2008], and it has been proposed by numerous authors [Dahl et al., 2005, Banerjee et al., 2006, Yuan and Lin, 2007]. In the graphical LASSO we set $\mathbf{b}$ to the sample mean of the training data, and then compute the

[8]Because of the global normalization, setting the potential to a constant value c for any choice of c is also equivalent to removing the node from the model

precision matrix by optimizing the negative log-likelihood with ℓ_1-regularization of the precision matrix elements. This latter problem can be written as the convex optimization problem

$$\min_{W \succ 0} - \log\det W + \mathrm{tr}(\hat{\Sigma} W) + \lambda ||W||_1. \tag{1.6}$$

In the above $||W||_1$ refers to the entry-wise ℓ_1 norm of W and $\hat{\Sigma}$ refers to the sample covariance matrix, $\hat{\Sigma} \triangleq (1/n)\sum_{i=1}^{n}(\mathbf{x}^i - \mathbf{b})(\mathbf{x}^i - \mathbf{b})^T$. The positive-definite constraint $W \succ 0$ is required to ensure that the solution is a valid distribution. Although this constraint may appear problematic since the positive-definite cone is an open set, the log-determinant term in the objective function acts as a log-barrier that ensures the solution is an interior point of this set[9]. Due to the appealing notion of combining regularization and sparsity within classical covariance selection methods, numerous authors have explored solution methods and applications of this model. A subset of the extensive work is [Dahl et al., 2005, Banerjee et al., 2006, Yuan and Lin, 2007, Friedman et al., 2008, d'Aspremont et al., 2008, Duchi et al., 2008a, Krishnamurthy and d'Aspremont, 2009, Lu, 2009, 2010, Yuan, 2009]. Further, as we discuss in the last section the dependency network methods of [Meinshausen and Buhlmann, 2006] represent pseudo-likelihood approximations to the graphical LASSO model.

We can also consider applying ℓ_1-regularization to the joint distribution in undirected graphical models over discrete variables; the first works to explore this were [Lee et al., 2006b, Wainwright et al., 2006, Dahinden et al., 2007]. Most of the work in this vein has considered the special case of pairwise undirected graphical models of binary data with Ising potentials. We refer to these models as Ising graphical models (IGMs). In IGMs, we write the joint distribution $p(x_1, x_2, \ldots, x_p)$ of a set of p binary random variables as

$$p(x_1, x_2, \ldots, x_p) \triangleq \frac{1}{Z}\exp(\sum_{i=1}^{p} x_i b_i + \sum_{(i,j)\in E} x_i x_j w_{ij}). \tag{1.7}$$

where in this case the normalizing constant Z is

$$Z \triangleq \sum_{\mathbf{x}'} \exp(\sum_{i=1}^{p} x'_i b_i + \sum_{(i,j)\in E} x'_i x'_j w_{ij}).$$

Some authors use a $\{0, 1\}$ representation of the binary variables in (1.7) [Wainwright et al., 2006, Hofling and Tibshirani, 2009], while other authors use a $\{-1, 1\}$ representation [Banerjee et al., 2008, Kolar and Xing, 2008][10]. Clearly, (1.7) is a pairwise undirected graphical model and setting the edge parameter w_{ij} to zero is equivalent to removing the pairwise potential from the model. Thus we can consider minimizing the negative log-likelihood with ℓ_1-regularization of the edge parameters to learn a regularized sparse structure. Specifically, we solve

$$\min_{W,\mathbf{b}} \sum_{m=1}^{n}[-\sum_{i=1}^{p}[-x_i^m b_i - \sum_{j=i+1}^{n} x_i^m x_j^m w_{ij}]] + n\log Z(W, \mathbf{b}) + \lambda \sum_{i=1}^{p}\sum_{j=i+1}^{p} |w_{ij}|. \tag{1.8}$$

[9]That the solution is an interior point of the constraint set is appealing from the perspective of optimization, since it means that we do not necessarily have to use constrained optimization methods to solve (1.6)

[10]Both representations can model arbitrary positive pairwise distributions over binary data, but these two parameterizations do not give necessarily lead to equivalent models if we regularize the parameters

We note that the first term in this expression is linear while the second is convex [Boyd and Vandenberghe, 2004, §3.1], so this is a convex optimization problem.

Unfortunately, solving this optimization problem is more complicated than the GGM case, because of the combinatorial nature of the normalizing constant Z. The complexity of computing Z and related quantities is discussed in (for example) [Koller and Friedman, 2009, §9-10]. In particular, it is #P-hard to evaluate Z. Hardness results also apply to other operations involving discrete undirected models, such as computing the most likely configuration (NP-hard in general) and computing marginal or conditional probabilities (#P-hard in general). In some practical cases of interest, it is possible to efficiently solve these problems. For example, variants of the belief-propagation message-passing algorithm can solve these problems in $\mathcal{O}(p)$ when the graph is tree-structured. However, the best known general methods for solving these problems require a runtime that is exponential in the treewidth of the graph. For more information on the complexity of inference, the relation to treewidth, and general message-passing algorithms, see (for example) [Koller and Friedman, 2009, §9-10].

Different authors have proposed different solutions to the computational intractability of evaluating the objective function. As we discuss in the previous section, Wainwright et al. [2006] fit a dependency network with logistic regression conditionals. This corresponds to a pseudo-likelihood approximation. For a generalization of IGMs, in [Schmidt et al., 2008] we considered the symmetric pseudo-likelihood approximation[11]

$$\min_{W,\mathbf{b}} -\sum_{m=1}^{n}[\sum_{i=1}^{p}\log p(x_i^m|\mathbf{x}_{-i}^m, W, \mathbf{b})] + \lambda\sum_{i=1}^{p}\sum_{j=i+1}^{p}|w_{ij}|. \tag{1.9}$$

We note that each conditional probability is the likelihood in a logistic regression model:

$$p(x_i|\mathbf{x}_{-i}, W, \mathbf{b}) = \frac{1}{Z_i}\exp(x_i b_i + \sum_{j\neq i} x_i x_j w_{ij}),$$

where the local normalizing constant Z_i only sums over possible assignments to x_i (and thus can be tractably evaluated). This is nearly identical to learning a dependency network with logistic regression conditionals, but in this formulation we use the same parameter w_{ij} in both $p(x_i|\mathbf{x}_{-i})$ and $p(x_j|\mathbf{x}_{-j})$ rather than using two versions of the parameter. [Hofling and Tibshirani, 2009] show that this approximation gives performance that is similar to or better than the performance of the dependency network approximation. A disadvantage of the symmetric version is that the optimization problem is slightly more difficult since the optimization problem is no longer separable in the conditional distributions. Instead of using a pseudo-likelihood approximation, Lee et al. [2006b] use the non-convex Bethe variational approximation to $\log Z$ (in the special case of tree-structured graphs, this approximation is convex and exact). Banerjee et al. [2008] proposed a convex variational approximation to $\log Z$, while Kolar and Xing [2008] outline a set of additional constraints that can be imposed to improve this approximation. Hofling and Tibshirani [2009] also consider using junction trees to evaluate the likelihood exactly, but the cost of this is exponential in the treewidth of the graph.

Because the complexity of using the model has such a strong dependency on the graph structure, one of the main interests in learning a sparse structure is to learn a model that is easier to use. However, we note that the degree of sparsity of a graph is not a perfect surrogate for the treewidth

[11] This approximation was subsequently used in [Hofling and Tibshirani, 2009]

of a graph[12]. In particular, it is possible for a graph with a large number of edges to have a lower treewidth than a graph that is more sparse. For example, a chain-structured graph on 100 nodes will have 99 edges and a treewidth of 1, while we if we construct a graph consisting of a three-clique and 97 other nodes with no edges then this graph has only 3 edges but has a treewidth of 2. Nevertheless, there is a simple local property that relates sparsity to treewidth: the treewidth of a graph is never decreased by adding an edge. Thus, although sparsity is not a perfect measure of treewidth, increased sparsity may lead to a decreased cost of using the model. Further, approximate inference methods typically scale linearly in the number of edges in the model. Thus, in cases where only high treewidth graphs provide good models of a data set, sparsity directly decreases the cost of using the model.

1.5 Pairwise Undirected Graphical Models

As we discuss in the previous section, there has been substantial interest in using ℓ_1-regularization for structure learning in GGMs and IGMs. However, GGMs and IGMs can only represent pairwise distributions over Gaussian and binary data, respectively. In this section we review the class of pairwise log-linear models, a generalization of IGMs that can be used to model arbitrary pairwise positive distributions over discrete data.

In log-linear models of discrete vectors $\mathbf{x} \in \{1, 2, \dots, k\}^p$, the logarithm of each of the potentials is a linear function of the parameters. For example, if variable x_j can take four states ($k = 4$), we can define the node potential $\phi_j(x_j)$ such that

$$\log \phi_j(x_j) = \mathbb{I}(x_j = 1)b_{j,1} + \mathbb{I}(x_j = 2)b_{j,2} + \mathbb{I}(x_3 = 3)b_{j,3},$$

where $b_{i,j}$ is the parameter associated with state j for node i, and $\mathbb{I}(\cdot)$ denotes an indicator function that returns a value of 1 if its argument is true and 0 otherwise. Note that even though x_j has four possible states in the example above, we only use three parameters. If we used four parameters then one of them would be redundant because the global normalization in (1.4) allows us to rescale each potential (or equivalently add or subtract a constant from the log-potential) without changing the model. Thus, we consider node potentials that have $k-1$ parameters for nodes that can take k possible states. We obtain node potentials that are equivalent to those used in IGMs in the special case where we have binary states. We find it convenient to use the notation $\mathbf{b}_j$ to denote the set of parameters $\{b_{j,1}, b_{j,2}, \dots, b_{j,k-1}\}$ associated with the node potential $\phi_j(x_j)$.

We consider several different parameterizations of the edge potentials. First, we note that in IGMs with binary variables that take the values $\{0, 1\}$ we can write each edge potential $\phi_{ij}(x_i, x_j)$ as

$$\begin{aligned}\log \phi_{ij}(x_i, x_j) &= x_i x_j w_{ij} \\ &= \mathbb{I}(x_i = 1, x_j = 1)w_{ij}.\end{aligned}$$

Note that this parameterization of the potentials treats the two states asymmetrically. That is, if $w_{ij} > 0$ then the edge encourages x_1 and x_2 to both take the state 1, but if $w_{ij} < 0$ it encourages them to both not take the state 1. In both cases, there is no distinction made between the three states $(1, 0)$, $(0, 1)$, and $(0, 0)$ (while as before if $w_{ij} = 0$ then the edge has no effect). This

[12] We discuss methods that use explicit constraints on the treewidth in Chapter 5

asymmetry is not present in the edge potentials used by IGMs with binary variables that take the values $\{-1, 1\}$:

$$\begin{aligned}
\log \phi_{ij}(x_i, x_j) &= x_i x_j w_{ij} \\
&= \mathbb{I}(x_i = x_j) w_{ij} - \mathbb{I}(x_i \neq x_j) w_{ij} \\
&\equiv \mathbb{I}(x_i = x_j) w_{ij} - \mathbb{I}(x_i \neq x_j) w_{ij} + w_{ij} \\
&= 2\mathbb{I}(x_i = x_j) w_{ij} \\
&= \mathbb{I}(x_i = x_j) \tilde{w}_{ij}, \qquad (1.10)
\end{aligned}$$

where in the third line we add the constant w_{ij} (the global normalization means this does not change the distribution) and in the last line we re-parameterize in terms of of $\tilde{w}_{ij} \triangleq 2w_{ij}$. In this form we see that the simple change in representation leads to the states being treated symmetrically, the edge encourages the nodes to take the same state if $w_{ij} > 0$ and encourages the nodes to have different states if $w_{ij} < 0$ (and the edge has no effect if $w_{ij} = 0$).

It is often convenient to represent our edge (log-)potentials as a matrix, where each entry (i, j) contains $\log \phi_{ij}(x_i, x_j)$. Below, we give the matrices corresponding to Ising edge potentials over two binary variables under the $\{0, 1\}$ (left) and $\{-1, 1\}$ (right) representations:

$$\log \phi_{ij}(\cdot, \cdot, w_{ij}) = \begin{bmatrix} w_{ij} & 0 \\ 0 & 0 \end{bmatrix}, \quad \log \phi_{ij}(\cdot, \cdot, w_{ij}) = \begin{bmatrix} w_{ij} & 0 \\ 0 & w_{ij} \end{bmatrix}.$$

The form of the right matrix as well as (1.10) suggests a generalization to data with more than two states, where we place a parameter on the diagonal elements of this matrix and no parameter on the off-diagonals. As an example, for variables with three states we use the following edge potential matrix:

$$\log \phi_{ij}(\cdot, \cdot, w_{ij}) = \begin{bmatrix} w_{ij} & 0 & 0 \\ 0 & w_{ij} & 0 \\ 0 & 0 & w_{ij} \end{bmatrix}.$$

In the remainder of this work we refer to potentials with this form as *Ising* potentials, and we call pairwise log-linear models with these types of potentials IGMs. If we have separate node and edge parameters for each node and edge (respectively), then Ising potentials are sufficient to model any pairwise positive distribution over binary data. However, we can only model a set of restricted distributions over general discrete data with Ising potentials. Thus, to model general distributions over discrete data we must consider other parameterizations of the edge potentials.

The next set of potentials we consider are a natural generalization of Ising potentials, where (for nodes taking values in $\{1, 2, \ldots, k\}$) we include a weight for each configuration where the nodes take the same state. For example, for an edge between two variables that can take three possible states we would use

$$\log \phi_{ij}(x_i, x_j) = \mathbb{I}(x_i = 1, x_j = 1) w_{ij1} + \mathbb{I}(x_i = 2, x_j = 2) w_{ij2} + \mathbb{I}(x_i = 3, x_j = 3) w_{ij3}.$$

Alternately, we can write the edge (log-)potentials as the matrix

$$\log \phi_{ij}(\cdot, \cdot, \mathbf{w}_{ij}) = \begin{bmatrix} w_{ij1} & 0 & 0 \\ 0 & w_{ij2} & 0 \\ 0 & 0 & w_{ij3} \end{bmatrix}.$$

Here, we use the notation $\mathbf{w}_{ij}$ to refer to the set of all parameters associated with an edge potential $\phi_{ij}(\mathbf{x}_{ij})$. These potentials distinguish between configurations where the variables take the same states, and can be used to model a wider class of distributions than Ising potentials. This form of potential was previously used in, for example, [Taskar et al., 2004] (who contrast it with the classic Potts model). Since we obtain Ising potentials if we set all the diagonal elements to the same value, we refer to potentials of this form as generalized Ising or *gIsing* potentials. However, note that with these potentials the edge is present in the model unless w_{ijk} is set to zero for all k.

We can also consider completely general potentials over discrete variables where we parameterize every element of the edge potential matrix, allowing us to model arbitrary pairwise positive distributions over discrete data. For example, for an edge between two variables that can each take three states we would use

$$\begin{aligned}\log \phi_{ij}(x_i, x_j) = &\mathbb{I}(x_i = 1, x_j = 1)w_{ij11} + \mathbb{I}(x_i = 1, x_j = 2)w_{ij12} + \mathbb{I}(x_i = 1, x_j = 3)w_{ij13} \\ &+ \mathbb{I}(x_i = 2, x_j = 1)w_{ij21} + \mathbb{I}(x_i = 2, x_j = 2)w_{ij22} + \mathbb{I}(x_i = 2, x_j = 3)w_{ij23} \\ &+ \mathbb{I}(x_i = 3, x_j = 1)w_{ij31} + \mathbb{I}(x_i = 3, x_j = 2)w_{ij32} + \mathbb{I}(x_i = 3, x_j = 3)w_{ij23},\end{aligned}$$

or in matrix form:

$$\log \phi_{ij}(\cdot, \cdot, \mathbf{w}_{ij}) = \begin{bmatrix} w_{ij11} & w_{ij12} & w_{ij13} \\ w_{ij21} & w_{ij22} & w_{ij23} \\ w_{ij31} & w_{ij32} & w_{ij33} \end{bmatrix}.$$

Since we assign a different potential to each configuration of the nodes, we refer to potentials of this form as *full* potentials. Here, we have k^2 parameters, and the edge is included in the model if *any* of these k^2 are non-zero. In [Schmidt et al., 2008] we used these types of potentials but fixed the value of one of the variables to zero (as with the node potentials) to decrease the number of parameters in the model. However, the choice of the particular variable to fix at zero can influence the particular structure learned, so in this work we use the full representation[13].

Below are the matrices for the Ising, gIsing, and full potentials:

$$\log \phi_{ij}(\cdot, \cdot, w_{ij}) = \begin{bmatrix} w_{ij} & 0 & 0 \\ 0 & w_{ij} & 0 \\ 0 & 0 & w_{ij} \end{bmatrix} \quad \text{(Ising edge potentials)};$$

$$\log \phi_{ij}(\cdot, \cdot, \mathbf{w}_{ij}) = \begin{bmatrix} w_{ij1} & 0 & 0 \\ 0 & w_{ij2} & 0 \\ 0 & 0 & w_{ij3} \end{bmatrix} \quad \text{(gIsing edge potentials)};$$

$$\log \phi_{ij}(\cdot, \cdot, \mathbf{w}_{ij}) = \begin{bmatrix} w_{ij11} & w_{ij12} & w_{ij13} \\ w_{ij21} & w_{ij22} & w_{ij23} \\ w_{ij31} & w_{ij32} & w_{ij33} \end{bmatrix} \quad \text{(full edge potentials)}.$$

An edge has no effect on the model if all entries of this matrix are set to zero[14]. In the Ising case this corresponds to setting w_{ij} to zero, while in the other cases we must set all elements of $\mathbf{w}_{ij}$ to zero. We close our discussion on potential parameterizations by noting that we can naturally extend the full potentials to scenarios where the two nodes have a different number of states.

[13] Under this parameterization, the optimal parameters are still identifiable if we use a strictly convex regularizer.

[14] In the case of full potentials, we could also set all the entries of the matrix to a constant.

In pairwise undirected models of discrete data, the negative log-likelihood function for a set of n realizations of p-vectors $\mathbf{x}^i$ is given by

$$-\sum_{m=1}^{n}[\sum_{i=1}^{p}[\log\phi_i(x_i^m,\mathbf{b}_i)+\sum_{j=i+1}^{p}\log\phi_{ij}(x_i^m,x_j^m,\mathbf{w}_{ij})]]+n\log Z(\mathbf{w},\mathbf{b}),$$

where we use the notation $\mathbf{b}$ to denote the set of all node parameters and $\mathbf{w}$ to denote the set of all edge parameters. As in the IGM case, this is a convex function. In log-linear models, the gradient of the average negative log-likelihood has a simple form. For example, the average gradient with respect to a node potential parameter $b_{i,j}$ is

$$\nabla_{b_{i,j}}-\frac{1}{n}\sum_{m=1}^{n}\log p(\mathbf{x}^m|\mathbf{b},\mathbf{w})=p(x_i=j)-\frac{1}{n}\sum_{m=1}^{n}\mathbb{I}(x_i^m=j).$$

Thus we see that at a maximum likelihood solution (where the gradient is zero), the model must have the same unary marginals as the data. Similarly, the average gradient with respect to an edge potential parameter w_{ijqr} (when using full potentials) is

$$\nabla_{w_{ijqr}}-\frac{1}{n}\sum_{m=1}^{n}\log p(\mathbf{x}^m|\mathbf{b},\mathbf{w})=p(x_i=q,x_j=r)-\frac{1}{n}\sum_{m=1}^{n}\mathbb{I}(x_i^m=q,x_j^m=r),$$

and thus at a maximum likelihood solution the model marginals will match the empirical frequencies for all edges that are included in the model.

The models in the previous sections have a one-to-one correspondence between parameters in the model and edges in the graph. However, the log-linear models we discuss in this section may have more than one parameter associated with each edge. Further, the edge is only removed from the model if *all* of the parameters associated with the edge are set to zero. Thus, if we would like to use regularization to directly encourage graphical sparsity we must consider *group* ℓ_1-regularization, a generalization of ℓ_1-regularization that penalizes *groups* of variables in order to directly encourage group-wise sparsity.

Utilizing group ℓ_1-regularization to encourage sparsity in terms of groups of variables was proposed by Bakin [1999][15]. In group ℓ_1-regularization, we penalize the ℓ_1 norm of the (non-squared) ℓ_2 norms of the groups. For our problem, we have one group for each edge and the group contains all parameters associated with the corresponding edge. Thus, we can write the problem of estimating a sparse regularized structure with group ℓ_1-regularization as

$$\min_{\mathbf{w},\mathbf{b}}-\sum_{m=1}^{n}[\sum_{i=1}^{p}[\log\phi_i(x_i^m,\mathbf{b}_i)+\sum_{j=i+1}^{p}\log\phi_{ij}(\mathbf{x}_{ij}^m,\mathbf{w}_{ij})]]+n\log Z(\mathbf{w},\mathbf{b})+\lambda\sum_{i=1}^{p}\sum_{j=i+1}^{p}||\mathbf{w}_{ij}||_2,\ (1.11)$$

(using an approximate objective function gives an analogous formulation). We obtain ℓ_1-regularization if each group contains only a single variable. We can interpret this "ℓ_1 of ℓ_2 norms" regularizer as an ℓ_1-regularizer of the lengths of the vectors $\mathbf{w}_{ij}$. Consequently, it encourages sparsity in the lengths of the vectors, leading to the entire group being set to zero when the length becomes zero.

Utilizing (1.11) was mentioned in [Lee et al., 2006b], but this work did not discuss how to solve the resulting optimization problem. Dahinden et al. [2007] use (1.11) to encourage graphical

[15]Sardy et al. [2000] discuss using ℓ_1-regularization of the complex modulus, a special case of group ℓ_1-regularization

sparsity, but their methodology is restricted to small data sets since they did not consider using approximate inference or efficient large-scale optimization strategies. In Chapter 3 we give large-scale optimization strategies that are especially suited to solving problems like (1.11), where we have a large number number of variables, a costly objective, and a regularizer (or constraints) with a simple structure. In Chapter 5, we consider several variations on (1.11). In particular, we show how different choices of the norm can lead to edge potentials with different properties (and in some cases better performance). We also extend the block-wise sparse strategy proposed in [Duchi et al., 2008a], where edges are placed in groups and we would like to encourage sparsity in terms of groups of edges. Finally, in Chapter 5 we consider extending (1.11) to include covariates for use in structured classification problems, leading to a discriminative structure learning method for structured classification.

1.6 General Log-Linear Models

Due to their relatively small number of parameters, pairwise log-linear have sometimes been advocated in scenarios where limited data is available [Whittaker, 1990, §9.3]. However, pairwise models only focus on the unary and pairwise statistical properties of the data, so the pairwise assumption can be fairly restrictive if higher-order moments of the data are important and we have sufficient training examples available to estimate such higher-order statistics. Despite this fact, with only one exception, all previous work on structure learning with ℓ_1-regularization has made the pairwise assumption. The one exception is Dahinden et al. [2007] who considered log-linear models of discrete data where all potentials up to a fixed order are considered, and used group ℓ_1-regularization to learn the structure.

For general log-linear models [Bishop et al., 1975], we can write the probability of a vector $\mathbf{x} \in \{1, 2, \dots, k\}^p$ as a globally normalized product of potential functions $\phi_A(\mathbf{x}_A)$ defined for each possible subset A of $S \triangleq \{1, 2, \dots, p\}$:

$$p(\mathbf{x}) \triangleq \frac{1}{Z} \prod_{A \subseteq S} \phi_A(\mathbf{x}_A).$$

As before the normalizing constant Z enforces that the distribution sums to one, and the logarithm of each potential $\phi_A(\mathbf{x}_A)$ is linear in the parameters of the potential. For models including higher-order terms, we use the short-hand $\mathbf{w}_A$ to refer to all the parameters associated with the potential $\phi_A(\mathbf{x}_A)$ (whether it be unary, pairwise, or higher-order), and we use $\mathbf{w}$ to refer to the concatenation of all $\mathbf{w}_A$. We define the unary potentials and pairwise potentials as before, and can define the threeway and higher-order potentials by generalizing the Ising, gIsing, and full potentials we discuss for pairwise models. In general, if A contains c elements that can each take k values, $\phi_A(\mathbf{x}_A)$ will have k^c parameters $\mathbf{w}_A$ when we use full potentials, k parameters when we use gIsing potentials, and one parameter when we use Ising potentials. In the case of full potentials, general log-linear models can be used to model arbitrary positive distributions over discrete data.

In practice, it is typically not feasible to include a potential $\phi_A(\mathbf{x}_A)$ for all 2^p subsets. As before, removing the potential $\phi_A(\mathbf{x}_A)$ from the model is equivalent to setting it to one (or any other constant) for all values of $\mathbf{x}_A$, or equivalently setting all elements of $\mathbf{w}_A$ to zero (or any other constant). We obtain the class of pairwise models if we enforce $\mathbf{w}_A = \mathbf{0}$ for all A with a cardinality greater than two. This effectively nullifies the effects of the higher-order statistics of the data on the model.

The group ℓ_1-regularization strategy from the previous section can naturally be extended to the case of general log-linear models. This results in the optimization problem

$$\min_{\mathbf{w}} -\sum_{i=1}^{n} \log p(\mathbf{x}^i|\mathbf{w}) + \sum_{A \subseteq S} \lambda_A ||\mathbf{w}_A||_2. \tag{1.12}$$

Here we include a separate regularization parameter $\lambda_A \geq 0$ for each group since we typically want to use a different degree of penalization for potentials of different orders.

This is (essentially) the approach taken in [Dahinden et al., 2007]. Dahinden et al. [2007] also consider a variant where we only consider potentials up to a certain order, and successively increase the order. This latter strategy can be viewed as an ℓ_1-regularization version of a classic strategy for structure learning in general log-linear models [see Bishop et al., 1975, §4.5.1]. However, a problem with (1.12) is that sparsity in the variable groups A does not directly correspond to conditional independencies in the model (except in the pairwise case). In particular, in a log-linear model variable sets B and C are conditionally independent given all other variables if and only if all elements of $\mathbf{w}_A$ are zero for all A that contain at least one element from B and at least one element from C [see Whittaker, 1990, Proposition 7.2.1].

In principle, we can use the optimization methods of Chapter 3 to solve (1.12) (with an approximate objective function, if necessary). Indeed, in Chapter 6 we consider using this formulation to learn the structure of threeway log-linear models. However, this formulation is only practical when the number of nodes p or the maximum size of the factors M is very small, since if we allow for M-way factors there are $\binom{p}{M}$ possible subsets of size M to examine. Further, if we allow factors of arbitrary size then there are 2^p factors to consider. For example, if we have 32 variables then we would have 2^{32} groups, and (with full potentials) each group would contain up to k^{32} parameters. This exponential number of variables makes the problem very difficult to solve if we don't enforce a strong cardinality restriction (such as restricting attention to pairwise or threeway models). Dahinden et al. [2007] did not address the problems associated with the exponential number of variables in this formulation, since their application only had five variables.

In Chapter 6, we consider using group ℓ_1-regularization for convex structure learning in the special case of hierarchical log-linear models, where a potential $\phi_A(\mathbf{x}_A)$ can only be included if the potentials on all subsets of A are also included. Although hierarchical models are a subset of the class of general log-linear models, they are a far larger class of models than the set of pairwise (or threeway) models. Further, one of the advantages of hierarchical models is that sparsity in the groups directly corresponds to conditional independencies in the model. Similar to [Bach, 2008b], we develop an active-set method that can incrementally add higher order factors, and places no restriction on the maximum cardinality of the potentials. This method uses the hierarchical property to potentially rule out an exponential number of higher-order potentials, and converges to a solution satisfying a set of necessary optimality conditions. Key to the convex parameterization of the space of hierarchical log-linear models is that we allow the groups to *overlap*. This results in a more difficult optimization problem, but in Chapter 6 we give a strategy to adapt the methods of Chapter 3 to the case of overlapping groups. Our experiments show that allowing for such higher order interactions can result in improved prediction accuracy.

1.7 Data Sets

In addition to experiments on synthetic data, in this work we test the performance of various methods on several real data sets. Several of the these latter data sets are used in multiple chapters and in multiple contexts. Thus, to avoid repeating information we introduce all of the real data sets in this section.

When we consider testing large-scale methods for ℓ_1-regularized logistic regression in Sections 2.6.1, we consider the following binary classification data sets:

- **sido**: This data set contains 4932 binary variables describing properties of 12678 molecules that have been tested against the AIDS HIV virus. The target indicates the molecular activity, and among the variables are several artificially generated 'probe' variables. This data set is made available as part of the Causality Workbench,
  `http://www.causality.inf.ethz.ch/home.php`.

- **thrombin**: This data set contains 139350 binary variables describing three-dimensional properties of 1909 molecules that have been tested for their ability to bind to thrombin, a key receptor in blood clotting. The target variables indicate whether the molecules are active (bind well). This data set has been made available by DuPont Pharmaceuticals Research Laboratories for the KDD Cup 2001 competition,
  `http://pages.cs.wisc.edu/~dpage/kddcup2001/`.

- **spam**: This data contains 823470 binary variables describing the presence of word tokens in 92189 e-mail messages involved in the legal investigations of the Enron corporation. The target variable indicates whether the e-mail was spam or not. This data set was made the TREC 2005 corpus [Cormack and Lynam, 2005],
  `http://plg.uwaterloo.ca/~gvcormac/treccorpus/`.
  This data set was prepared by Peter Carbonetto, who used the SpamBayes software for feature extraction,
  `http://spambayes.sourceforge.net/`.

Since many of the tasks related to learning probabilistic graphical models are NP-hard or #P-hard, in some cases we consider data sets that have a relatively small number of nodes and states. This makes it possible to solve the NP-hard and #P-hard problems exactly. This removes any confounding effects associated with using approximations when comparing optimization strategies in Sections 2.6.2 and 3.5, and comparing different models in Sections 5.8.1 and 6.5. These data sets also allows us to compare the quality of different approximations compared to the exact case in Section 5.8.2. We examine two small data sets:

- **cyto**: The data studied in [Sachs et al., 2005]. In this study, intracellular multivariate flow cytometry was used to simultaneously measure the expression levels of 11 phosphorylated proteins and phospholipid components in 5400 individual primary human immune system cells over 9 different stimulatory/inhibitory conditions. We used the targets of intervention and 3-state discretization strategy (into 'under-expressed', 'baseline', and 'over-expressed') of [Sachs et al., 2005]. This data set is available at the Causality Workbench Repository (we ignore the experimental conditions),
  `http://www.causality.inf.ethz.ch/repository.php`.

- **awma**: The coronary heart disease data studied in [Qazi et al., 2007]. In this study, expert cardiologists provided ratings from 1-5 of the motion of 16 segments of the left ventricle of the heart in 2602 patients [Qazi et al., 2007]. Here, a rating of 1 indicates normal, while classes 2-5 represent degrees of abnormality. Although the segments are rated from 1 to 5, as in [Qazi et al., 2007] we aggregate the four abnormal states (2-5) into a single state (classes 3 to 5 are severely under-represented in the data). This data set was provided by Siemens Medical Solutions, and is not available on-line.

When we examine learning probabilistic graphical models of larger binary data sets in Section 4.6.2, we focus on the following data sets:

- **rain**: We created a data set consisting of 28-vectors, representing a binarized version of the 'daily precipitation' amount for the first 28 of days of the month for the weather station in Steveston, British Columbia. We obtained values from 1896-2004, but removed months with missing (or accumulated) values (this left 1059 months). We only used the first 28 days for each month to make all of the samples have the same length. Measurements marked with a zero or trace precipitation values were assigned to one class, while fields with a non-zero value were assigned to the other class (approximately 41 percent of the values are non-zero). This data was extracted from the Canadian Daily Climate Archive from Environment Canada's National Climate and Data Information Archive,
`http://climate.weatheroffice.ec.gc.ca/`.

- **msweb**: The Anonymous Microsoft Web Data, a data set measuring whether each of 294 webpages were visited by 32711 anonymous randomly-selected users of `microsoft.com`. We focused on the 57 websites with greater than 250 visits. This data set is available from the UCI Machine Learning Repository,
`http://archive.ics.uci.edu/ml/index.html`.

- **news**: A data set measuring the occurrence of 100 words in 16242 newsgroup postings from the 20 Newsgroups data. This data set is available from Sam Roweis' data page,
`http://www.cs.toronto.edu/~roweis/data.html`.

- **usps**: A set of 11000 binary 16 by 16 images (256 variables), each representing a single digit. We binarized the pixels by assigning pixels with a value of zero to one state, and pixels with a non-zero value to the other state. This data set is available from Sam Roweis' data page,
`http://www.cs.toronto.edu/~roweis/data.html`.

When we examine learning probabilistic graphical models of larger non-binary discrete data sets in Sections 5.8.3 and 6.5, we focus on the following data sets:

- **awma-5**: Rather than aggregating the four abnormal states $(2-5)$ into one single state, we consider the full five-state version of the *awma* data from [Qazi et al., 2007]. Here, 1 indicates normal, 2 indicates hypokinetic, 3 indicates akinetic, 4 indicates dyskinetic, and 5 indicates aneurysm.

- **traffic**: The *traffic* data contains 32 four-state variables measuring the level of traffic flow at different San Francisco locations at 4413 time points [Krause and Guestrin, 2005]. This data sets was also previously analyzed in Shahaf et al. [2009], and was sent to us by Dafna Shahaf.

- **temperature**: The *temperature* data contains 54 four-state variables measuring temperature levels in the Intel Research, Berkeley lab [Deshpande et al., 2004]. This data set was also previously analyzed in Shahaf et al. [2009], and was also sent to us by Dafna Shahaf.

- **usps4**: Rather than binarizing the *usps* data, in this data set we discretize the pixels' intensity values into four equally-space bins (and we concentrated on the 16 pixels in the center of the images).

- **usps8**: This is similar to the *usps4* data, but using a discretization into eight bins.

When we consider learning blockwise-sparse GGMs in Section 5.8.4, we consider the following data set:

- **genes**: A subset of the data set examined in [Gasch et al., 2000], containing mRNA expression levels of 667 genes in the yeast genome measured under 174 different conditions. This data set was previously analyzed in [Duchi et al., 2008a], and we use the same pre-processing and assignment of the variables to the 86 'types' that they use.

Finally, when we consider structured binary classification in Section 5.8.5 we focus on the following data set:

- **awma-c**: In this data set we consider the classification problem of labeling 16 segments of the left ventricle as normal or abnormal based on multi-view ultrasound video [Schmidt et al., 2008]. This is similar to the *awma* data, but consists of 345 cases where we have cardiologist labels for all 16 segments as well as features measured from the associated videos. For this data, we have a total of 34 features for each segment measuring properties of the motion of the segment from the tracked contours of the ventricle. This data set was also provided by Siemens Medical Solutions.

1.8 Summary of Contributions

Below, we briefly summarize the contributions of each chapter:

- **Chapter 2**: We give extensions of limited-memory quasi-Newton methods for differentiable optimization to the case of optimizing a differentiable function with ℓ_1-regularization. We argue that these extensions have more appealing properties than previous extensions. Our experiments on ℓ_1-regularized logistic regression indicate that these extensions perform similar to or better than other methods for this problem.

- **Chapter 3**: We give new limited-memory quasi-Newton methods for optimizing differentiable functions subject to simple constraints or simple non-differentiable regularizers. We argue that these extensions are appealing when the differentiable function is high-dimensional and costly to evaluate, while the constraints (or non-differentiable regularizer) have a simple structure. Our experiments on group ℓ_1-regularized pairwise log-linear models indicate that these methods outperform existing methods for this problem.

- **Chapter 4**: We give a method that uses ℓ_1-regularization to learn a sigmoid dependency network to prune the set of edges that are considered in a search over the space of DAG models with sigmoid CPDs. Unlike previous pruning methods that prune based on a different

criteria than the subsequent search, our method uses the same score in both the pruning and the search phase. Our experiments indicate that this pruning strategy is advantageous over previous pruning strategies that do not take advantage of the structure of the CPDs or the form of the score. Although we concentrate on the case of sigmoid CPDs the method applies to a more general class of linearly-parameterized CPDs.

- **Chapter 5**: We consider methods for learning pairwise undirected graphical models with group ℓ_1-regularization to encourage graphical sparsity. Unlike previous work, we consider using group ℓ_1-regularization with different choices of the group norm, and argue that the structure offered by different choices can be advantageous. We show how to apply the methods of Chapter 3 to the case of different group norms, and we introduce a group version of the nuclear norm regularizer. Our experiments indicate that different choices of the group norm can lead to improved predictive performance, and that utilizing general pairwise log-linear models of discrete data can lead to better predictive performance than IGMs. We extend previous work on blockwise-sparse GGMs by considering different choices of the group. We also extend previous work on group ℓ_1-regularization of log-linear models to the case of conditional log-linear models, representing the first method that simultaneously and discriminatively learns both structure and parameters in a structured classification model.

- **Chapter 6**: We consider using overlapping group ℓ_1-regularization for structure learning in hierarchical log-linear models, with no restriction on the cardinality of the potentials. We give an active-set method for searching the exponential space of possible higher-order potentials, and show how to apply the methods of Chapter 3 to the case of overlapping groups. Our experiments indicate that removing the cardinality restriction leads to better predictive performance than pairwise (or threeway) models.

Chapter 2

Optimization with ℓ_1-Regularization

Many of the models examined in this work require solving an ℓ_1-regularized logistic regression problem. Although logistic regression is typically used for binary classification (§1.1), we must also solve a set of ℓ_1-regularized logistic regression problems for structure learning in dependency networks (§1.2), and for the DAG structure learning method we describe in Chapter 4. Further, pseudo-likelihood approximations in ℓ_1-regularzied IGM models (§1.4) and general binary log-linear models (§5.3) take the form of a set of dependent ℓ_1-regularized logistic regression problems. If we consider models with more than two states for each node, then these problems are replaced with the analogous multiclass logistic regression [Bishop, 2006, §4.3.4]. Further, the ℓ_1-regularized IGM model and general binary log-linear models (§1.5) also have a similar structure. Thus, in this work it is important to be able to efficiently optimize the parameters in ℓ_1-regularized logistic regression and related models. Fortunately, this has recently become a well-studied problem.

In this chapter, we describe algorithms for solving the general optimization problem

$$\min_{\mathbf{x}} f(\mathbf{x}) \triangleq L(\mathbf{x}) + \sum_i \lambda_i |x_i|, \tag{2.1}$$

where $L(\mathbf{x})$ is assumed to be convex and differentiable with respect to $\mathbf{x} \in \mathbb{R}^p$, and we may have a separate regularization parameter $\lambda_i \geq 0$ for each variable i. We particularly concentrate on the special case of the ℓ_1-regularized logistic regression problem we discuss in Section 1.1. In this case, $L(\mathbf{x})$ is the negative log-likelihood in a logistic regression model, the optimization parameters $\mathbf{x}$ are the concatenation of the weights $\mathbf{w}$ and bias b in the model, and λ_i is the same across all i, except for the bias term b where $\lambda_i = 0$. More precisely, logistic regression is the following special case of (2.1):

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} \log(1 + \exp(-y^i(\mathbf{w}^T\mathbf{x}^i + b))) + \lambda||\mathbf{w}||_1.$$

In this case, the gradients of $L(\mathbf{w}, b)$ with respect to b and $\mathbf{w}$ are given by

$$\nabla_b L(\mathbf{w}, b) = \sum_{i=1}^{n} -y^i/(1 + \exp(y^i(\mathbf{w}^T\mathbf{x}^i + b))),$$

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b) = \sum_{i=1}^{n} -y^i\mathbf{x}^i/(1 + \exp(y^i(\mathbf{w}^T\mathbf{x}^i + b))).$$

Although our focus is on logistic regression, we note that the algorithms we describe in this chapter are applicable to any optimization problem of the form (2.1), including (for example) ℓ_1-regularized IGMs.

Solving (2.1) is complicated by the non-differentiability of $|x_i|$ at $x_i = 0$. In the next section, we briefly discuss one of the most effective optimization methods for logistic regression when no

regularization is used, or when (differentiable) ℓ_2-regularization is used. We then proceed to outline several properties that we would like an optimization method for ℓ_1-regularized logistic regression to have, followed by a discussion of existing and then new methods for solving the non-differentiable ℓ_1-regularized logistic regression problem.

2.1 Logistic Regression with Differentiable Regularization

In the case of unregularized logistic regression or logistic regression with ℓ_2-regularization, several comparison studies indicate that quasi-Newton methods are among the most efficient methods available for solving large-scale (generalized) logistic regression problems [Malouf, 2002, Wallach, 2002, Minka, 2003, Sha and Pereira, 2003]. Quasi-Newton optimization algorithms are closely related to optimization methods based on Newton's method, but where the matrix of second partial derivatives of the objective function (the Hessian) is replaced by an approximation. Typically, for large-scale problems the approximation is constructed using limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) updates [Nocedal, 1980]. In this section we review a Newton-like algorithm for unconstrained optimization, and then discuss L-BFGS updates.

Newton-like algorithms for unconstrained differentiable optimization are iterative methods, where at each iteration we form a quadratic model

$$q_k(\mathbf{x}) \triangleq f(\mathbf{x}_k) + (\mathbf{x} - \mathbf{x}_k)^T \nabla f(\mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T H_k (\mathbf{x} - \mathbf{x}_k),$$

around the current iterate $\mathbf{x}_k$. Here, H_k is a positive-definite matrix (the Hessian or an approximation of it). To compute the new iterate $\mathbf{x}_{k+1}$ we move to the minimum of this quadratic. This minimum is given by

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - H_k^{-1} \nabla f(\mathbf{x}_k).$$

Unfortunately, this new iterate may not necessarily decrease the objective function. However, the direction of search, $\mathbf{d} \triangleq -H_k^{-1} \nabla f(\mathbf{x}_k)$, is a descent direction at $\mathbf{x}_k$. That is, for sufficiently small $\alpha > 0$ we have $f(\mathbf{x}_k + \alpha \mathbf{d}) < f(\mathbf{x}_k)$ provided that $\mathbf{x}_k$ is not an optimal solution (and that H_k is positive-definite). Therefore, we can decrease the objective function by moving in the direction $\mathbf{d}$ using iterates of the form

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha H_k^{-1} \nabla f(\mathbf{x}_k),$$

for some sufficiently small $\alpha > 0$. To choose the step length α, we can use a line search. Specifically, to guarantee a sufficient decrease on the objective value, we start with $\alpha = 1$ and decrease α until we satisfy the Armijo condition

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + \nu \nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k), \quad \text{with} \quad \nu \in (0, 1). \tag{2.2}$$

A typical value of the sufficient decrease parameter ν is 10^{-4}. In the case of logistic regression and if H_k is a reasonable approximation to the Hessian, the value $\alpha = 1$ will typically be accepted. If this value is not accepted, we can generate a new step length $\tilde{\alpha}$ in the interval $(0, \alpha)$. A common approach to do this is to set $\tilde{\alpha}$ to the minimum of the cubic polynomial that interpolates $f(\mathbf{x}_k)$, $f(\mathbf{x}_{k+1})$, and the directional derivatives $\nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k)$ and $\nabla f(\mathbf{x}_{k+1})^T (\mathbf{x}_{k+1} - \mathbf{x}_k)$. Typically, we use some safeguards to ensure that $\tilde{\alpha}$ is in the interval $(0, \alpha)$ and that it is not too close to either end point. For example, we can project the minimum of the cubic interpolant into the interval $[\xi_1 \alpha, \xi_2 \alpha]$, for $0 < \xi_1 \leq \xi_2 < 1$. For the logistic regression objective function, the minimum of the

cubic polynomial, $\tilde{\alpha}$, will typically be accepted, but if it is not we can repeat the cubic interpolation focusing on the interval $(0, \tilde{\alpha})$. Successively refining the interval generates a decreasing sequence of step lengths, and we must eventually find a value of α that satisfies the Armijo condition. Once a suitable value of α is found, we set $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha\mathbf{d}$ and start a new iteration at $\mathbf{x}_{k+1}$. We typically continue this process until the iterations no longer make substantial progress (the relative change in parameter values or function values is too small), or until we satisfy a criterion measuring the first-order optimality of the current iterate (i.e. that $||\nabla f(\mathbf{x}_k)||$ is too small). For more information on Newton's method for optimization and the other issues we discuss in this section, see [Gill et al., 1981, Nocedal and Wright, 1999] (among many other standard references).

2.1.1 L-BFGS Approximation

Rather than explicitly using the exact Hessian $H_k = \nabla^2 f(\mathbf{x}_k)$, quasi-Newton methods allow us to build an approximation to the Hessian (or its inverse) using successive differences in the parameter vector

$$\mathbf{s}_k \triangleq \mathbf{x}_{k+1} - \mathbf{x}_k,$$

and the gradient

$$\mathbf{y}_k \triangleq \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k).$$

Quasi-Newton methods typically begin with a scaled identity matrix approximation $B_0 \triangleq \sigma I$ to the Hessian (for some positive σ), and after each iteration B_{k+1} is updated so that the changes in parameters and gradients satisfy the secant equation

$$B_{k+1}\mathbf{s}_k = \mathbf{y}_k. \tag{2.3}$$

The solution B_{k+1} of (2.3) is not unique, and the most common way to choose a unique matrix B_{k+1} is with the BFGS formula [see Gill et al., 1981, §4.5.2]. In the limited-memory version of the BFGS update, L-BFGS, we don't explicitly store B_k but rather store a set of m differences $\mathbf{s}_k$ and $\mathbf{y}_k$. To pre-multiply a vector with the inverse of a matrix $B_0 = \sigma_k I$ updated m times with these stored vectors, we can use a simple algorithm that runs in $\mathcal{O}(mp)$ [Nocedal, 1980].

The choice of the scaling coefficient σ_k can have a significant impact on the performance of the method. A widely-used and typically very effective choice of this scaling is [Shanno and Phua, 1978]

$$\sigma_k \triangleq \frac{\mathbf{y}_k^T \mathbf{y}_k}{\mathbf{y}_k^T \mathbf{s}_k}. \tag{2.4}$$

In later sections we consider utilizing an inverse Hessian approximation that simply takes the form $\sigma_k I$, with σ_k given by (2.4) and without any quasi-Newton updates applied to it. This was proposed by Barzilai and Borwein [1988], and represents a σ_k that minimizes the squared error in (2.3) under this simple approximation[16].

[16] An important property of the Barzilai-Borwein approximation, as opposed to the L-BFGS approximation, is that it only changes the magnitude of the negative gradient, and not its direction

2.1.2 ℓ_1-Regularization over an Orthant

We define an orthant for some sign pattern $\{\zeta_1, \zeta_2, \ldots, \zeta_p\}$ to be the closed subset of $\mathbb{R}^p$ satisfying

$$\begin{aligned} \zeta_1 x_1 &\geq 0, \\ \zeta_2 x_2 &\geq 0, \\ &\ldots \\ \zeta_p x_p &\geq 0, \end{aligned}$$

where each ζ_i can take values in the set $\{-1, 1\}$. An important property that is used extensively in this chapter is that the ℓ_1-regularization problem (2.1) is differentiable over any given orthant. In particular, over an orthant with sign pattern element ζ_i the derivative of the regularizer with respect to a variable i is given by the linear function $\lambda_i \zeta_i$. Thus, if we are given an orthant containing the optimal solution, solving the ℓ_1-regularization problem (2.1) reduces to the problem of minimizing a convex differentiable objective function with bound constraints on the variables (the bound constraints ensure that we do not leave the orthant). Minimizing differentiable functions subject to simple bound constraints can be solved with straightforward modifications of problems for minimizing unconstrained differentiable functions (like quasi-Newton methods with an L-BFGS Hessian approximation); we describe one such method in Section 2.2.3.

2.2 Logistic Regression with ℓ_1-Regularization

Typically, we do not know the orthant of the optimal solution, so we must consider methods that deal with the non-differentiability of the regularizer. We would like to have an algorithm that allows us to efficiently solve the optimization problem even when the number of variables or the number of training examples is large. Toward this end, we can identify several properties that we would like of an optimization algorithm for solving the ℓ_1-regularized logistic regression problem:

1. Block updates: the algorithm is able to move more than one variable at a time to improve the objective function.

2. Linear time/space: the algorithm requires $\mathcal{O}(p)$ space, and $\mathcal{O}(p)$ time per iteration.

3. Warm start: if we initialize the algorithm close to the optimal solution, it will require fewer iterations to converge.

4. Sparse iterates: if the final solution is sparse, the algorithm does not necessarily need to evaluate the objective function with a dense parameter vector.

These four properties eliminate several of the available strategies. For example, the block-updates requirement eliminates coordinate descent methods such as [Fu, 1998] (such methods are extremely effective if $L(\mathbf{x})$ is close to separable, but their performance degrades sharply as the dependency between variables increases). The linear time/space requirement eliminates projected Newton methods like the constrained iteratively-reweighted least squares method [Lee et al., 2006c]. The warm-start requirement eliminates the possibility of applying interior point methods to a constrained re-formulation of the problem [Koh et al., 2007]. Finally, the sparse-iterates requirement also eliminates such interior point methods (since variables only become zero in the limit), and also eliminates

approaches based on the expectation maximization bound optimization algorithm [Figueiredo, 2003] (since variables cannot move away from zero once they are set to zero in this framework).

Despite eliminating some alternatives, these requirements still leave a variety of methods available. Ideally, we would also like a method for ℓ_1-regularized logistic regression that satisfies the following three properties:

5. Reduction to Newton's method: If at some iteration the algorithm identifies the orthant and set of non-zero variables in the optimal solution, and if it stays in this orthant with this set of non-zero variables on subsequent iterations, then the algorithm will take the same steps that an unconstrained (quasi-)Newton method applied to the non-zero variables would use to solve (2.1) over the orthant.

6. Fast modification of non-zero set: At each iteration, the algorithm is able to make many zero-valued variables non-zero. Similarly, it is able to make many non-zero variables take the value zero.

7. No increase in problem size: The algorithm is able to solve the problem in terms of the original variables, rather than solving an equivalent problem with a larger number of variables.

In [Schmidt et al., 2007a, 2009a], we review a large variety of the available methods for solving ℓ_1-regularized logistic regression problems, and experimentally compared 14 of the available methods. Unfortunately, none of the algorithms discussed in these reviews satisfy all 7 of the above properties. Rather than reviewing all of the methods discussed in this prior work, in this section we only review three of the most effective methods, namely the orthant-wise learning algorithm, active-set methods, and applying the two-metric projection method to a bound-constrained re-formulation of the problem. Though very effective, each one of these strategies is deficient in one of the last three properties. After reviewing these three methods, in the next section we present extensions of these three methods that (in two of the three cases) allow the method to satisfy all 7 properties and that (in all three cases) lead to better practical performance.

2.2.1 Orthant-Wise Learning

Andrew and Gao [2007] present one of the most effective methods currently available for solving large-scale ℓ_1-regularized logistic regression problems. It is based on choosing an appropriately defined *steepest descent* direction on (2.1) and taking a step resembling a Newton iteration in this direction (with an L-BFGS Hessian approximation). To define the steepest descent direction we note that even though $f(\mathbf{x})$ in (2.1) is not differentiable in general, that directional derivatives always exist (by convexity of $f(\mathbf{x})$). Thus, analogous to the differentiable case, we can define the steepest descent direction for $f(\mathbf{x})$ at a point $\mathbf{x}$ as the direction that minimizes the directional derivative (ie. the direction that locally decreases the objective most quickly). Closely related to this concept is what Andrew and Gao [2007] refer to as the *pseudo-gradient* of $f(\mathbf{x})$, defined as the element of the sub-differential of $f(\mathbf{x})$ at $\mathbf{x}$ with minimum norm. Following an argument in [Bertsekas et al., 2003, §8.4] (replacing maximization with minimization and concavity with convexity), it follows that the steepest descent direction (in the Euclidean norm) for a convex function is the negation of this pseudo-gradient.

The sub-differential of $f(\mathbf{x})$ in (2.1) (denoted $\partial f(\mathbf{x})$) with respect to a variable i is given by

$$\partial_i f(\mathbf{x}) = \nabla_i L(\mathbf{x}) + \lambda_i \operatorname{sgn}(x_i), \tag{2.5}$$

where the set-valued function $\text{sgn}(x_i)$ is defined by [see Bertsekas, 1999, Figure B.12]

$$\text{sgn}(x_i) \triangleq \begin{cases} \text{sign}(x_i), & x_i \neq 0 \\ [-1, 1], & x_i = 0 \end{cases}.$$

Since the sub-differential is separable in the variables, the problem of computing the minimum-norm element of the sub-differential is also separable in the variables. Hence, we can solve the minimum-norm problem coordinate-wise to yield that the pseudo-gradient with respect to a variable i is:

$$\tilde{\nabla}_i f(\mathbf{x}) \triangleq \begin{cases} \nabla_i L(\mathbf{x}), & \lambda = 0 \\ \nabla_i L(\mathbf{x}) + \lambda_i \text{ sign}(x_i), & \lambda > 0, |x_i| > 0 \\ \nabla_i L(\mathbf{x}) + \lambda_i, & \lambda > 0, x_i = 0, \nabla_i L(\mathbf{x}) < -\lambda_i \\ \nabla_i L(\mathbf{x}) - \lambda_i, & \lambda > 0, x_i = 0, \nabla_i L(\mathbf{x}) > \lambda_i \\ 0, & \lambda > 0, x_i = 0, |\nabla_i L(\mathbf{x})| \leq \lambda_i \end{cases} \tag{2.6}$$

In the first two cases the function is differentiable with respect to i, so the pseudo-gradient is simply the gradient with respect to i (the only element of the sub-differential). In the last case of (2.6) the pseudo-gradient is zero, since we can set $\text{sgn}(x_i)$ to $-\nabla_i L(\mathbf{x})/\lambda_i$ to achieve a norm of zero. In the remaining cases we obtain the minimum-norm solution by setting $\text{sgn}(x_i)$ to have the opposite sign of $\nabla_i L(\mathbf{x})$. The pseudo-gradient has several properties that are analogous to the gradient vector in unconstrained optimization. First, as we discuss above, the negative pseudo-gradient $-\tilde{\nabla} f(\mathbf{x}_k)$ is in the direction that minimizes the directional derivative at $\mathbf{x}_k$. Second, it follows from the first property that $\tilde{\nabla} f(\mathbf{x}_k) = \mathbf{0}$ is a necessary and sufficient condition for an iterate $\mathbf{x}_k$ to be a global minimum provided that $L(\mathbf{x})$ is a differentiable convex function.

Since the regularizer is piecewise-linear and is a simple linear function over a given orthant we might be tempted to use the pseudo-gradient in a quadratic approximation of $f(\mathbf{x})$ at $\mathbf{x}_k$, yielding the approximation

$$q_k(\mathbf{x}) \triangleq f(\mathbf{x}_k) + (\mathbf{x} - \mathbf{x}_k)^T \tilde{\nabla} f(\mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T H_k (\mathbf{x} - \mathbf{x}_k), \tag{2.7}$$

where H_k is a positive-definite approximation of $\nabla^2 L(\mathbf{x})$. We could then consider minimizing this quadratic approximation to generate a search direction $\mathbf{d} \triangleq -H_k^{-1} \tilde{\nabla} f(\mathbf{x}_k)$, leading to iterations of the form $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \mathbf{d}$ (where α is selected by a backtracking line search to satisfy the Armijo condition, with the gradient replaced by the pseudo-gradient). Unfortunately, there are two major problems with this approach: (i) the line search has no mechanism to set variables to exactly zero, and (ii) in general $\mathbf{d}$ will not be a descent direction.

To address the problem that the line search does not set variables to exactly zero, [Andrew and Gao, 2007] set variables $\mathbf{x}_{k+1}$ to zero if they differ in sign from $\mathbf{x}_k$. We use $\mathcal{P}_{\mathcal{O}}$ to denote this orthant projection applied to a parameter vector $\mathbf{x}$ for some arbitrary direction $\mathbf{d}$:

$$\mathcal{P}_{\mathcal{O}}(\mathbf{x} + \mathbf{d})_i \triangleq \begin{cases} 0 & \text{if } x_i(x_i + d_i) < 0, \\ x_i + d_i & \text{otherwise.} \end{cases}$$

Applying this projection to $\mathbf{x}_{k+1}$ is effective at sparsifying the parameter vector since it sets variables to exactly zero, and it also ensures that the line search does not cross points of non-differentiability (since the line search is truncated along some dimensions so that $\mathbf{x}_{k+1}$ is in an orthant containing $\mathbf{x}_k$).

There are many ways to address the problem that the method may not generate a descent direction, and the methods we discuss in this chapter will differ mainly in how they address this problem. However, the main insight used by all the methods is that $f(\mathbf{x})$ is differentiable over any single orthant, and that if we restrict attention to variables with non-zero pseudo-gradient then the quadratic approximation (2.7) is the truncated Taylor series expansion of the function restricted to a particular orthant. In particular, it is the truncated Taylor series expansion for the orthant containing $\mathbf{x}_k - \epsilon\tilde{\nabla} f(\mathbf{x}_k)$ for some extremely small $\epsilon > 0$. Thus, provided that $\mathbf{x}_k$ does not minimize $f(\mathbf{x})$ over this orthant (which would imply that $\mathbf{x}_k$ is a global optimum), using this quadratic approximation is guaranteed to yield a descent direction at $\mathbf{x}_k$ if $\mathbf{x}_{k+1}$ happens to lie in this 'right' orthant for sufficiently small positive α. Since in general this will not be the case, the methods considered in this chapter will give different strategies to ensure that $\mathbf{x}_{k+1}$ lies in the 'right' orthant for sufficiently small α. To ensure that $\mathbf{x}_{k+1}$ leads into the correct orthant, the sufficient condition that is used by the various methods is that the search direction $\mathbf{d}$ agrees with the steepest descent direction $-\tilde{\nabla} f(\mathbf{x}_k)$ for all variables that are zero[17].

The orthant-wise learning algorithm of [Andrew and Gao, 2007] uses a direct approach to enforce this sufficient condition. In particular, they compute $\mathbf{d}$ and set any value d_i in $\mathbf{d}$ to zero if its sign does not agree with $-\tilde{\nabla} f(\mathbf{x}_k)$. We use $\mathcal{P}_{\mathcal{S}}$ to denote this sign projection:

$$\mathcal{P}_{\mathcal{S}}(\mathbf{d})_i \triangleq \begin{cases} d_i, & \text{if } d_i(\tilde{\nabla}_i f(\mathbf{x}_k)) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

Andrew and Gao [2007] show that the positive-definiteness of H_k implies that $\mathcal{P}_S(\mathbf{d})$ will have at least one non-zero element and will represent a descent direction at $\mathbf{x}_k$ (for sub-optimal $\mathbf{x}_k$). Thus to generate the new iterate, the orthant-wise learning method uses steps of the form

$$\mathbf{x}_{k+1} = \mathcal{P}_{\mathcal{O}}[\mathbf{x}_k + \alpha\mathcal{P}_{\mathcal{S}}[-H_k^{-1}\tilde{\nabla} f(\mathbf{x}_k)]].$$

As with all the algorithms we discuss in this chapter, the line search parameter α is chosen to satisfy the Armijo sufficient decrease condition (2.2) (but using the pseudo-gradient in place of the gradient).

The orthant-wise learning method was the most effective method in experiments on ℓ_1-regularization problems by the authors [Andrew and Gao, 2007] and us [Schmidt et al., 2009b], while it was among the most effective methods in our comparison of quasi-Newton methods for ℓ_1-regularized logistic regression [Schmidt et al., 2009a]. Nevertheless, we might still hope to develop a better method since the orthant-wise learning method only satisfies six of our seven criteria; it does not reduce to Newton's method on the non-zero variables. This is because of the sign projection $\mathcal{P}_{\mathcal{S}}$, that may always restrict the method to only move along a subset of the non-zero variables.

2.2.2 Active-Set Methods

Active-set methods are widely used for solving ℓ_1-regularization problems. Osborne et al. [2000] outline an active-set method that is one of the first methods proposed for optimizing a linear least-squares objective subject to a constraint on the ℓ_1-norm of the parameter vector, as in (1.2). This algorithm was extended to the case of logistic regression in [Roth, 2004], while active-set methods have also been proposed for the λ formulation where we put a penalty of the ℓ_1-norm

[17] We note that this condition is trivially satisfied if the positive-definite matrix H_k is diagonal, and hence a diagonal-scaling or the Barzilai-Borwein approximation yield a descent direction.

of the parameter vectors [Perkins et al., 2003, Lee et al., 2006a]. Methods that seek to trace the regularization path of optimal coefficients as λ is varied for the least-squares loss [Osborne et al., 2000, Efron et al., 2004] or logistic regression [Rosset, 2004, Park and Hastie, 2007] are also closely related. In this section, we discuss an active-set method that is most closely related to the method proposed in [Perkins et al., 2003].

In the approach of [Perkins et al., 2003], we divide the variables into two sets: the working set $\mathcal{W}$ containing the non-zero variables, and the active set $\mathcal{A}$ containing the zero-valued variables (here, we use the terminology *active set* because of the analogy with active-set methods for constrained optimization). On each iteration we only update the working set, and on a typical iteration we generate the next iterate by projecting the Newton step:

$$\mathbf{x}_{\mathcal{W}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{x}_{\mathcal{W}} - \alpha H_{\mathcal{W}}^{-1} \nabla_{\mathcal{W}} f(\mathbf{x})]. \tag{2.9}$$

Here, we have used $\nabla_{\mathcal{W}} f(\mathbf{x})$ to denote the sub-vector of $\nabla f(\mathbf{x})$ corresponding to elements of $\mathcal{W}$, and $H_{\mathcal{W}}$ to denote the sub-matrix of H_k with all rows and columns of $\mathcal{W}$. Since this step is restricted to the non-zero variables, the gradient is defined and this step has the descent property (it decreases the objective function for sufficiently small α, provided that the non-zero variables are not at their optimal value given the fixed values of the remaining variables). Although it was not present in [Perkins et al., 2003], we have added the projection operator $\mathcal{P}_{\mathcal{O}}$ to allow the method to set variables to exactly zero and to prevent the possibility that a very small absolute value in $\mathbf{x}_{\mathcal{W}}$ will require that the line search chooses a very small value of α.

Once we have found the optimal values for the non-zero variables given the zero-valued variables, two things can happen. First, if $|\tilde{\nabla}_i f(\mathbf{x})| = 0$ for all $i \in \mathcal{A}$, then we terminate with the optimal solution. Otherwise, we move a single variable from the active set to the working set. In particular, we choose the variable i in $\mathcal{A}$ with the largest pseudo-gradient magnitude $|\tilde{\nabla}_i f(\mathbf{x})|$ (i.e. the zero-valued variable that gives the steepest decrease in the objective), and we move this variable from $\mathcal{A}$ to $\mathcal{W}$.

With the new working set $\mathcal{W}$, we apply the Newton step (2.9) where we replace the gradient with the pseudo-gradient. Interestingly, this yields a descent direction. This is because the scaling matrix $H_{\mathcal{W}}$ has positive diagonals (because H_k is positive-definite) and the pseudo-gradient with respect to the non-zero variables is zero (because they are at their conditionally optimal values), so the sign of $(H_{\mathcal{W}}^{-1} \tilde{\nabla}_{\mathcal{W}} f(\mathbf{x}))_i$ will match the sign of $\tilde{\nabla}_i f(\mathbf{x})$ for the single zero-valued variable i just added to $\mathcal{W}$. This means that the active-set method doesn't need to use the (potentially harmful) $\mathcal{P}_{\mathcal{S}}$ sign projection.

It is quite clear that this active-set method satisfies our definition of reducing to Newton's method. Once we have identified the correct working set (and its sign), and if the iterates maintain this working set (and its sign), then the active-set method will essentially be applying Newton iterations to optimize the working set. Unfortunately, we achieved this property at the cost of another; the active-set method no longer allows fast changes to the set of non-zero variables. In particular, if k variables have the value zero that must be non-zero in the optimal solution, then we must perform *at least* k iterations of the method. This can be impractical for large-scale problems with many non-zero variables.

We can imagine several solutions to this problem. First, we could consider initializing all variables non-zero, but this would lead to the loss of the sparse iterates property. We could also imagine trying to move more than one variable away from 0 on each iteration. Unfortunately, as soon as we consider moving two variables away from zero in a single iteration, we can no longer

guarantee that the Newton-like direction is a descent direction. We return to this latter idea in Section 2.3.

2.2.3 Two-Metric Projection

A classic strategy in linear programming for addressing ℓ_1-norm minimization problems is to transform the elements of the norm into positive and negative parts, then minimize a linear function of these parts [see Bertsimas and Tsitsiklis, 1997, §1.3]. Applied to (2.1), we write each x_i using new non-negative variables x_i^+ and x_i^- as $x_i = x_i^+ - x_i^-$, leading to the problem

$$\min_{\mathbf{x}^+,\mathbf{x}^-} L(\mathbf{x}^+ - \mathbf{x}^-) + \sum_i \lambda_i(x_i^+ + x_i^-), \text{ subject to } x_i^+ \geq 0, x_i^- \geq 0, \forall_i. \tag{2.10}$$

This is now a smooth optimization problem with simple non-negativity constraints on the variables. This problem shares the same minimizers as (2.1). To see this, note that the range of $L(\mathbf{x})$ is unchanged under the $\{\mathbf{x}^+, \mathbf{x}^-\}$ representation. Further, the objective function is an upper bound on (2.1), since by the non-negativity constraints and the triangle inequality we have $x_i^+ + x_i^- = |x_i^+| + |-x_i^-| \geq |x_i^+ - x_i^-| = |x_i|$. Finally, we note that the upper bound is tight at a minimizer; it must be the case that at least one of x_i^+ or x_i^- is 0, so $|x_i| = x_i^+ + x_i^-$ (otherwise, it violates that we are at a minimizer since we could decrease the regularization term without changing the value of $L(\mathbf{x})$ by decreasing x_i^+ and x_i^- by some small positive constant). In the discussion below, we use $\mathbf{y}$ to denote the concatenation of the positive and negative parts.

There are many efficient methods available for solving smooth optimization problems with bound constraints. We outline one, the two-metric projection algorithm discussed in Gafni and Bertsekas [1982]. In the two-metric projection algorithm, we divide the variables into two sets. By analogy with the previous section, we call them the active set $\mathcal{A}$ and the working set $\mathcal{W}$. In this algorithm, the active set is defined as the set of variables that are sufficiently close to zero and have a positive partial derivative. In other words, the active set is defined by

$$\mathcal{A} \triangleq \{i | y_i < \epsilon, \nabla_i f(\mathbf{y}) > 0\}.$$

The working set is the complement of this set, consisting of variables that are sufficiently non-zero or that have a negative partial derivative. In the two-metric projection method, we simultaneously take the projection of a Newton step for the working set and the projection of a diagonally-scaled gradient step for the active set. Specifically, we take the simultaneous iteration

$$\begin{aligned} \mathbf{y}_\mathcal{W} &\leftarrow [\mathbf{y}_\mathcal{W} - \alpha H_\mathcal{W}^{-1} \nabla_\mathcal{W} f(\mathbf{y})]^+ \\ \mathbf{y}_\mathcal{A} &\leftarrow [\mathbf{y}_\mathcal{A} - \alpha D \nabla_\mathcal{A} f(\mathbf{y})]^+. \end{aligned}$$

The diagonal matrix D must be positive-definite and the function $[x]^+ \triangleq \max\{x, 0\}$ projects the variables onto the non-negative orthant. A typical choice for D is the identity matrix, making the step for the active-set variables a projected-gradient step[18]. This simultaneous iteration is guaranteed to provide (feasible) descent on the objective function. Further, this iteration will reduce to Newton's method on the non-zero variables if the correct active set has been identified, and it allows us to move many variables between the zero and non-zero sets at each iteration.

[18]It is referred to as a *two-metric* projection method because we can write it as a scaled projected gradient step, but where we project using the Euclidean norm rather than a quadratic norm defined by the Hessian approximation.

Unfortunately, this method has the obvious drawback that we have increased the problem size. A more subtle potential disadvantage of this method lies in the re-formulation of the problem. In particular, the Hessian of the re-formulation is necessarily singular (it contains columns that differ only in sign), even if the Hessian of the original problem is positive-definite. This might indicate that the re-formulation may be more difficult to solve than the original problem.

2.3 Projected Scaled Sub-Gradient

We now consider extensions of the three methods above that alleviate each of their disadvantages, and in two of the cases give us methods that satisfy all 7 properties that we would like of an optimizer. We call these methods projected scaled sub-gradient (PSS) methods, because the iterations can be written as the projection of a scaling of a sub-gradient of the objective. In particular, the PSS methods use specific choices of the sub-gradient, scaling, and projection:

- For the projection, we use the (Euclidean) orthant projection $\mathcal{P}_{\mathcal{O}}$.
- For the sub-gradient, we use the pseudo-gradient $\tilde{\nabla} f(\mathbf{x})$.
- For the scaling, we require that the scaling leads to the correct orthant (ie. among variables currently set to zero, no element of the scaled direction has opposite sign from the negative pseudo-gradient).
- For the scaling, we also require that it is positive-definite with respect to some subset of the variables with non-zero pseudo-gradient (and zero with respect to the remaining rows).

Because we use a scaling matrix but project under the Euclidean norm, these methods could alternately be called two-metric sub-gradient projection methods, and we note that both the orthant-wise learning and active-set methods of the previous section satisfy the four properties above. These properties ensure that the method generates descent directions and, although we omit detailed convergence proofs, ensure convergence under fairly weak conditions. This can be shown by suitable modifications of the arguments made for gradient-related methods [Bertsekas, 1999, Proposition 1.2.1], as in [Andrew and Gao, 2007]

2.3.1 Gafni-Bertsekas Variant

The first variant we consider is analogous to the two-metric projection idea, but applied to ℓ_1-regularization instead of bound constraints. We refer to this as the PSS Gafni-Bertsekas variant (PSSgb). In this method, we define the working set as those variables that are sufficiently non-zero:

$$\mathcal{W} \triangleq \{i | |x_i| > \epsilon\}.$$

As usual, the active set is defined as the complement of this set. Similar to the two-metric projection method we perform a simultaneous iteration where we take a projection of the Newton step along the working set and a diagonally-scaled projected pseudo-gradient step for the active-set variables:

$$\mathbf{x}_{\mathcal{W}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{x}_{\mathcal{W}} - \alpha H_{\mathcal{W}}^{-1} \nabla_{\mathcal{W}} f(\mathbf{x})]$$
$$\mathbf{x}_{\mathcal{A}} \leftarrow \mathcal{P}_{\mathcal{O}}[\mathbf{x}_{\mathcal{A}} - \alpha D \tilde{\nabla}_{\mathcal{A}} f(\mathbf{x})].$$

Provided we use a positive diagonal scaling D, this combined direction is guaranteed to be a descent direction (ie. it has a negative directional derivative) unless $\mathbf{x}$ is optimal. To see this, note that a sub-optimal $\mathbf{x}$ must have at least one non-zero element in $\nabla_{\mathcal{W}} f(\mathbf{x})$ or $\tilde{\nabla}_{\mathcal{A}} f(\mathbf{x})$. If we have a non-zero element in $\nabla_{\mathcal{W}} f(\mathbf{x})$, then by positive-definiteness of $H_{\mathcal{W}}$ it follows that the contribution to the directional derivative of moving the variables in $\mathcal{W}$ in the direction $-H_{\mathcal{W}}^{-1} \nabla_{\mathcal{W}} f(\mathbf{x})$ is negative (and similarly, if $\nabla_{\mathcal{W}} f(\mathbf{x}) = \mathbf{0}$ then the contribution to the directional derivative is zero). If we have a non-zero element in $\tilde{\nabla}_{\mathcal{A}} f(\mathbf{x})$, it similarly follows that the component of the directional derivative with respect to the zero-valued variables is negative (while if $\tilde{\nabla}_{\mathcal{A}} f(\mathbf{x}) = \mathbf{0}$ then this contribution is zero). This is because $-\tilde{\nabla}_i f(\mathbf{x})$ is in the direction that coordinate-wise minimizes the directional derivative, so all non-zero elements in $\tilde{\nabla}_{\mathcal{A}} f(\mathbf{x})$ correspond to variables that have a negative contribution to the directional derivative (while variables with a pseudo-gradient of 0 contribute a value of zero to the directional derivative). Subsequently, in either case the overall directional derivative is negative and the combined direction is a descent direction.

Note that after identifying the correct set of non-zero variables, these iterations will perform Newton steps on the non-zero variables. Further, many variables can be made zero/non-zero at each iteration, and we have not increased the size of the problem. Thus, we have a simple algorithm that achieves all 7 properties.

In the two-metric projection algorithm, the choice of the diagonal scaling matrix D does not have a significant effect on the performance of the algorithm (it simply controls the rate that very small variables move towards zero). However, the choice of D in the PSSgb algorithm can have a significant effect on the performance of the method, since if D is too large we may need to perform several backtracking steps before the step length is accepted (while too small of a value will require many iterations to move variables from the active set to the working set). In our implementation of the method, we compute the Shanno-Phua/Barzilai-Borwein scaling σ_k of the variables given by (2.4), and set D to $\sigma_k^{-1} I$.

2.3.2 Sign Constraint Variant

The orthant-wise learning algorithm does not satisfy the property of reducing to Newton's method because of the $\mathcal{P}_{\mathcal{S}}$ sign projection. The problem with this projection is that it may set elements of the Newton direction to zero for a large portion of the zero and non-zero variables. However, note that to lie in the correct orthant (to guarantee descent) we only require that the zero-valued variables in the search direction agree with the negative pseudo-gradient sign. Thus, in the PSS sign projection (PSSsp) variant we apply the orthant-wise learning iteration but use a less constrained version of the $\mathcal{P}_{\mathcal{S}}$ sign projection. Specifically, we use

$$\mathcal{P}_{\mathcal{S}^*}(\mathbf{d})_i \triangleq \begin{cases} d_i, & \text{if } (\mathbf{x}_k)_i \neq 0 \text{ or } d_i(\tilde{\nabla}_i f(\mathbf{x}_k)) > 0 \\ 0, & \text{otherwise} \end{cases}$$

The only difference between the $\mathcal{P}_{\mathcal{S}^*}$ projection and the $\mathcal{P}_{\mathcal{S}}$ projection (2.8) is the presence of the condition "if $(\mathbf{x}_k)_i \neq 0$", which stops us from unnecessarily applying the sign projection to non-zero variables. The method still does not reduce to Newton's method because the zero-valued variables still affect the search direction for the non-zero variables. However, this simple modification allows greater use of the Hessian approximation for the non-zero variables and leads to improved practical performance.

2.3.3 Active-Set Variant

In the final PSS variant, the PSSas method, we extend the active-set method of Section 2.2.2 above so that it can add more than one variable to the working set at each iteration. In this method, we augment the simple working set $\mathcal{W}$ above with the k zero-valued variables with the largest (non-zero) pseudo-gradient magnitudes to obtain a new working set $\mathcal{W}^k$. We then compare the sign values of the elements of the pseudo-gradient to the signs in the product $H_{\mathcal{W}^k}^{-1}\tilde{\nabla}_{\mathcal{W}^k}f(\mathbf{x})$. If the signs for all k zero-valued variables agree, then the working set is valid and we simply take the active-set step given by (2.9), but using $\mathcal{W}^k$ and the pseudo-gradient. If any of the signs disagree, then we must find a different value for k that satisfies this property (k can range from 0 up to the number of zero-valued variables that have non-zero pseudo-gradient).

We would like k to be as large as possible, but can not test too many values because testing each value of k costs $\mathcal{O}(mp)$. For example, a naive approach is to start with $k = 0$ (which is always valid), and test each k in increasing order until we find an invalid value (then accept $k - 1$). This would lead to a worst-case iteration cost of $\mathcal{O}(mp^2)$, making it unsuitable for large-scale problems. In the PSSas method, we do a binary search for a k such that k is valid and $k + 1$ is invalid. This finds a value of k that is at least as large as the one found by the naive method, but has a worst-case iteration cost of $\mathcal{O}(mp \log p)$, only slightly higher than the iteration cost of the other methods we discuss in this section[19].

[19] We could re-gain the linear-time iteration cost by using a constant upper bound on k.

2.4 Implementation

To make the PSS algorithms concrete, we now give pseudo-code for the PSS methods. Algorithm 1 outlines a general framework that can be used as a basis for implementing a PSS method.

Input: Function $L(\mathbf{x})$, regularization parameters λ_i, initial parameter vector $\mathbf{x}_0$, optimality tolerance ϵ, number of corrections m, sufficient decrease parameter η, line search safeguard parameters ξ_1 and ξ_2, direction calculation function $\text{dir}(k, g, \sigma, S, Y)$

$k \leftarrow 0$;
$S \leftarrow []$; // initialize collection on of quasi-Newton vectors
$Y \leftarrow []$;
$f_k \leftarrow L(\mathbf{x}_0) + ||\lambda_i \bullet \mathbf{x}_0||_1$; // evaluate initial parameter vector
$\mathbf{g}_k \leftarrow \tilde{\nabla} f(\mathbf{x}_0)$; // compute pseudo-gradient, see (2.6)
while $||\mathbf{g}_k||_\infty > \epsilon$ **do**
  $\mathbf{d}_k \leftarrow \text{dir}(k, g, \sigma, S, Y)$; // compute algorithm-specific descent direction
  $\alpha \leftarrow 1$; // initial step length
  $\mathbf{x}_{k+1} \leftarrow \mathcal{P}_{\mathcal{O}}(\mathbf{x}_k + \alpha \mathbf{d}_k)$; // initial trial value
  $f_{k+1} \leftarrow L(\mathbf{x}_{k+1}) + ||\lambda_i \bullet \mathbf{x}_{k+1}||_1$; // evaluate new parameter vector
  $\mathbf{g}_{k+1} \leftarrow \tilde{\nabla} f(\mathbf{x}_{k+1})$; // compute pseudo-gradient, see (2.6)
  while $f_{k+1} > f_k + \eta \mathbf{g}_k^T (\mathbf{x}_{k+1} - \mathbf{x}_k)$ **do**
    Select $\alpha \in (\xi_1 \alpha, \xi_2 \alpha)$; // safeguarded cubic interpolation
    $\mathbf{x}_{k+1} \leftarrow \mathcal{P}_{\mathcal{O}}(\mathbf{x}_k + \alpha \mathbf{d}_k)$; // next trial value
    $f_{k+1} \leftarrow L(\mathbf{x}_{k+1}) + ||\lambda_i \bullet \mathbf{x}_{k+1}||_1$;
    $\mathbf{g}_{k+1} \leftarrow \tilde{\nabla} f(\mathbf{x}_{k+1})$;
  $\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$; // compute quasi-Newton differences
  $\mathbf{y}_k \leftarrow \mathbf{g}_{k+1} - \mathbf{g}_k$;
  if $k > m$ **then**
    Remove oldest vector from S and Y;
  $S \leftarrow [S \quad \mathbf{s}_k]$; // update quasi-Newton difference matrices
  $Y \leftarrow [Y \quad \mathbf{y}_k]$;
  $\sigma \leftarrow (\mathbf{y}_k^T \mathbf{y}_k)/(\mathbf{y}_k^T \mathbf{s}_k)$; // update diagonal Hessian scaling
  $k \leftarrow k + 1$;

Algorithm 1: PSS framework for ℓ_1-regularized optimization.

A practical implementation would be slightly more complicated than Algorithm 1, because typically we want to impose iteration or function evaluation limits, and we implement checks that assess whether sufficient progress continues to be made. Note that we compute the quasi-Newton vectors based on the pseudo-gradient rather than the differences in $\nabla L(\mathbf{x})$ as in [Andrew and Gao, 2007], since we found this gave better performance. Although it may seem to counter-intuitive to include the non-smooth component as part of the Hessian approximation, there has been some empirical work showing that the BFGS approximation may be effective for certain types of non-smooth problems [Lewis and Overton, 2008]. We have left the calculation of the descent direction ($\mathbf{d}_k$) unspecified in this pseudo-code, because this is the primary difference between the PSS methods. In particular, the orthant-wise learning method uses Algorithm 2 to calculate the

descent direction (the PSSsp direction calculation is identical, but using the $\mathcal{P}_{\mathcal{S}^*}$ sign projection).

Input: Iteration number k, pseudo-gradient g, scaling σ, quasi-Newton matrices S and Y
Output: Descent direction $\mathbf{d}$
if $k = 0$ **then**
  $\mathbf{d} \leftarrow -\min\{1, 1/||\mathbf{g}||_1\}\mathbf{g}$;
else
  $\mathbf{d} \leftarrow -H^{-1}\mathbf{g}$; // apply L-BFGS algorithm using g, S, Y, and σ
  $\mathbf{d} \leftarrow \mathcal{P}_{\mathcal{S}}(\mathbf{d})$; // sign projection, see (2.8)

Algorithm 2: Direction calculation in orthant-wise learning.

In this algorithm, we use $\min\{1, 1/||\mathbf{g}||_1\}$ as the scaling of the gradient step on the first iteration. For logistic regression, this heuristic tends to make the step small enough that it is typically accepted without the need to backtrack. Algorithm 3 outlines the descent direction calculation used in the active-set method.

Input: Iteration number k, pseudo-gradient g, scaling σ, quasi-Newton matrices S and Y
Output: Descent direction $\mathbf{d}$
$\mathbf{d} \leftarrow \mathbf{0}$;
$\mathcal{W} \leftarrow \{i|\lambda_i = 0\} \bigcup \{i|x_i \neq 0\}$; // default working set
if $||\mathbf{g}_{\mathcal{W}}||_\infty < \epsilon$ **then**
  $\mathcal{W} \leftarrow \mathcal{W} \bigcup \{i|i = \arg\max_j |\mathbf{g}_j|\}$; // add variable to working set
if $k = 0$ **then**
  $\mathbf{d}_{\mathcal{W}} \leftarrow -\min\{1, 1/||\mathbf{g}_{\mathcal{W}}||_1\}\mathbf{g}_{\mathcal{W}}$;
else
  $\mathbf{d}_{\mathcal{W}} \leftarrow -H_{\mathcal{W}}^{-1}\mathbf{g}_{\mathcal{W}}$; // apply L-BFGS algorithm using g, S, Y, and σ

Algorithm 3: Direction calculation in active-set method.

Algorithm 4 outlines the descent direction calculation in the PSSgb method. We note that the descent direction calculation for the two-metric projection method is similar.

Input: Iteration number k, pseudo-gradient g, scaling σ, quasi-Newton matrices S and Y
Output: Descent direction $\mathbf{d}$
if $k = 0$ **then**
  $\mathbf{d} \leftarrow -\min\{1, 1/||\mathbf{g}||_1\}\mathbf{g}$;
else
  $\mathcal{W} \leftarrow \{i|\lambda_i = 0\} \bigcup \{i|x_i \neq 0\}$;
  $\mathcal{A} \leftarrow \mathcal{W}^c$; // active set is complement of working set
  $\mathbf{d}_{\mathcal{A}} \leftarrow -\sigma^{-1}\mathbf{g}_{\mathcal{A}}$; // take steepest descent direction on active set
  $\mathbf{d}_{\mathcal{W}} \leftarrow -H_{\mathcal{W}}^{-1}\mathbf{g}_{\mathcal{W}}$; // apply L-BFGS algorithm using g, S, Y, and σ

Algorithm 4: Direction calculation in PSSgb.

Finally, Algorithm 5 outlines the descent direction calculation for the PSSas method.

Input: Iteration number k, pseudo-gradient g, scaling σ, quasi-Newton matrices S and Y
Output: Descent direction $\mathbf{d}$

$\mathbf{d} \leftarrow \mathbf{0}$;
$\mathcal{W} \leftarrow \{i|\lambda_i = 0\} \bigcup \{i|x_i \neq 0\}$; `// default working set`
if $k = 0$ **then**
  $\mathbf{d}_{\mathcal{W}} \leftarrow -\min\{1, 1/||\mathbf{g}_{\mathcal{W}}||_1\}\mathbf{g}_{\mathcal{W}}$;
else
  $\mathbf{d}_{\mathcal{W}} \leftarrow -H_{\mathcal{W}}^{-1}\mathbf{g}_{\mathcal{W}}$; `// default direction (` $k = 0$ `)`
  $LB \leftarrow 0$; `//` $k = 0$ `is always legal`
  $UB \leftarrow 1 + |\{i|x_i = 0 \text{ and } g_i \neq 0\}|$; `//` k `can not be greater than the number of zero-valued variables with non-zero pseudo-gradient`
  while $UB - LB \neq 1$ **do**
    $k \leftarrow \lceil (UB + LB)/2 \rceil$; `//new value for` k
    $\mathcal{W}^k \leftarrow \mathcal{W} \bigcup \{i|i \text{ among largest } k \text{ values of } |g_k| \text{ for } x_i = 0\}$;
    $\mathbf{d}^k \leftarrow \mathbf{0}$;
    $\mathbf{d}^k_{\mathcal{W}^k} \leftarrow -H^{-1}_{\mathcal{W}^k}\mathbf{g}_{\mathcal{W}^k}$;
    if $sgn(\mathbf{d}^k)_i \neq sgn(\mathbf{g})_i$ *for some* i *with* $x_i = 0$ **then**
      $UB \leftarrow k$; `// this is not a valid value of` k
    else
      $LB \leftarrow k$; `// largest valid value of` k `found so far`
      $\mathbf{d} \leftarrow \mathbf{d}^k$;

Algorithm 5: Direction calculation in PSSas.

We close this section by noting a final subtle but important implementation detail. On iterations where the working set changes, it isn't immediately obvious how the L-BFGS update should be defined. For example, one possible strategy would be to reset the L-BFGS approximation every time the working set changes. Unfortunately, this excludes the possibility that curvature information gathered with a related working set might still be useful. In our implementation, we store the vector and gradient differences from the previous m iterations for all variables, and define the L-BFGS update based on the differences in the working set variables that satisfy

$$(\mathbf{s}_k)^T_{\mathcal{W}}(\mathbf{y}_k)_{\mathcal{W}} > \epsilon. \tag{2.11}$$

This curvature condition is sufficient to guarantee that the quasi-Newton matrix is positive-definite [see Nocedal and Wright, 1999, §8.1]. In our implementation, we compute σ_k based on the differences in all variables.

2.5 Regularization Path and Active-Set Optimization

Hoerl and Kennard [1970] introduced the concept of a ridge trace, a plot of the optimal coefficients in an ℓ_2-regularized least-squares model as the regularization parameter λ is varied. More recently, there has been substantial interest in similar plots for ℓ_1-regularized coefficients as λ is varied, for both least-squares [Osborne et al., 2000, Efron et al., 2004] and logistic regression [Rosset, 2004, Park and Hastie, 2007]. In this section we consider the calculation of multiple points along this

regularization path. That is, we would like to solve (2.1) for a set of values of λ rather than a fixed value.

When solving (2.1) for multiple values of λ, as with many other algorithms we can improve the performance of the PSS algorithms by using a warm-start strategy; we reduce the number of PSS iterates needed by initializing the iterates using the solution with a closely related value of λ. This allows us to solve the optimization problem for a set of values of λ more efficiently than we would be able to if we ran the optimizer independently for each value of λ.

In addition to taking advantage of warm-starting (which is also possible with ℓ_2-regularization), for ℓ_1-regularization we can take advantage of the sparsity of the coefficients along the regularization path to solve the ℓ_1-regularized problem for a sequence of values of λ for a much smaller cost than if we were using ℓ_2-regularization. This idea is very important in Chapters 5 and 6 where it leads to an exponential speed-up for larger values of λ, but we introduce it here since it still yields modest computational gains in the case of logistic regression.

Consider the following set of necessary and sufficient conditions for a vector $\mathbf{x}$ to be a minimizer of $f(\mathbf{x})$ for given values of λ_i:

$$\begin{cases} \nabla_i L(\mathbf{x}) + \lambda_i \operatorname{sign}(x_i) = 0, & |x_i| > 0 \\ |\nabla_i L(\mathbf{x})| \leq \lambda_i, & x_i = 0 \end{cases} \tag{2.12}$$

These conditions are equivalent to the necessary and sufficient optimality condition of requiring the zero-vector to be an element of the sub-differential [Bertsekas, 1999, §B.5], or equivalently that the pseudo-gradient is the zero vector. Rather than applying a PSS method to all the variables, we could apply it to the set of non-zero variables combined with the variables satisfying $|\nabla_i L(\mathbf{x})| > \lambda_i$. Once we have computed the optimal solution restricted to this set, we update the set of variables and repeat the optimization. That is, we alternate between two steps:

- Find variables i such that $x_i \neq 0$, or $x_i = 0$ and $|\nabla_i L(\mathbf{x})| > \lambda_i$.
- Solve the problem with respect to these variables.

If the set of variables to optimize does not change between iterations of this procedure, then the parameter vector satisfies (2.12) and hence is globally optimal (if it does change, then we are at a sub-optimal solution and we must continue to loop between these two steps). This is essentially the same active-set procedure discussed in Hofling and Tibshirani [2009], and note that we can use an approximate solution in the second step provided we eventually solve the problem to optimality. If we consider beginning with a sufficiently large value of λ and running this procedure with a decreasing sequence of λ values (as is done in [Park and Hastie, 2007]), then most of the iterations for large values of λ will only be run on a small subset of the variables. This is in contrast to the case of ℓ_2-regularization, where all variables are non-zero for all values of λ and we would need to solve the problem explicitly with respect to all variables for all values of the regularization parameter.

The optimality conditions also allow us to determine the values of the λ_i variables where all of the variables that are subject to regularization are set to zero. For the ℓ_1-regularized logistic regression problem, it follows from (2.12) that the value of λ that sets all regression weights to zero is

$$\lambda_{max} \triangleq \max_i |\nabla_{w_i} L(\mathbf{0}, \tilde{b})|,$$

where $\tilde{b}$ is the optimal value of the bias parameter subject to the constraint that $\mathbf{w} = \mathbf{0}$. In general, we can compute λ_{max} by optimizing with respect to the unregularized variables, and then finding the maximum gradient magnitude.

2.6 Experiments

We compared the performance of several large-scale optimization methods for ℓ_1-regularized logistic regression. In particular, we compared the following methods:

- **OWL**: the orthant-wise learning method we discuss in 2.2.1.
- **AS**: the active-set method we discuss in 2.2.2.
- **TMP**: the two-metric projection method we discuss in 2.2.3.
- **PSSgb**: the projected scaled sub-gradient method (Gafni-Bertsekas variant) proposed in 2.3.1.
- **PSSsp**: the projected scaled sub-gradient method (sign projection variant) proposed in 2.3.2.
- **PSSas**: the projected scaled sub-gradient method (active-set variant) proposed in 2.3.3.
- **BBSG**: a *Barzilai-Borwein sub-gradient* method where we move along the negative pseudo-gradient with the step length given by (2.4), and project the iterates using the $\mathcal{P}_{\mathcal{O}}$ operator.
- **SPG**: applying a spectral projected gradient method to the bound constrained formulation (2.10), similar to [Figueiredo et al., 2007].
- **BBST**: applying the iterative soft-thresholding algorithm with the step length given by (2.4), similar to [Wright et al., 2009].
- **DSST**: applying a diagonally scaled soft-thresholding algorithm, similar to [Hofling and Tibshirani, 2009].
- **OPG**: applying Nesterov's optimal projected gradient method to the bound-constrained formulation, using the adaptive line-search suggested in [Liu et al., 2009].

For a comparison to other methods on some small-scale problems, see [Schmidt et al., 2007a, 2009a]. The list above contains three of the most effective methods in this previous work, the new PSS methods, as well as five very effective newer methods that were not included in the previous comparisons. The first six methods use L-BFGS updates and a backtracking line search using the Armijo condition. The BBSG/SPG/BBST methods use the Barzilai-Borwein step length [Barzilai and Borwein, 1988] with a backtracking line search using the non-monotone Armijo condition [Grippo et al., 1986, Raydan, 1997]. The DSST method uses a diagonal scaling, using the inverse of the diagonals of the Hessian (this is only the method that explicitly computes second-order information). Finally, the OPG method uses Nesterov's optimal worst-case gradient method for optimizing differentiable objectives over simple convex sets [Nesterov, 2004, §2.2.4], augmented with the adaptive line search and Lipschitz estimation procedure discussed in [Liu et al., 2009].

2.6.1 Logistic Regression

We tested the methods on the three binary classification data sets from Section 1.7. We measured the performance of the methods in terms of the objective value achieved against the number of function evaluations used by the methods. The termination criteria for all methods was that the infinity norm of the pseudo-gradient was less than 10^{-5}, or the change in objective value between successive

iterations, parameter values between successive iterations, or directional derivative of the descent direction, was below 10^{-9}. We set the initial step size of all the methods to $1/\min\{1, 1/||\tilde{\nabla} f(\mathbf{w})||_1\}$ (except for the OPG method where we used $1/n$ as used in the code of [Liu et al., 2009], which we found gave slightly better performance). We set the sufficient decrease parameter η in the line search to 10^{-4}, and the safeguard parameters for projecting the cubic interpolation $\{\xi_1, \xi_2\}$ to .001 and 0.6. For the methods based on an L-BFGS approximation of the Hessian, we stored 10 previous parameter and gradient vectors. For the methods that use the non-monotonic Armijo condition, we set the number of previous function values to store at 10. For the OWL method, we used a quadratic initialization of the line search [Nocedal and Wright, 1999, §3.4] since we found this gave better performance than initializing the line search with $\alpha = 1$.

In our experiments, we set λ_i to 1 for all x_i (except the bias, where λ_i was set to zero). Optimizing with this relatively small value of λ still leads to sparse solutions, and it makes the optimization difficult enough that we can see a difference in performance between the methods. For larger values of λ, the performance of the methods becomes more similar, while the relative performance of the methods for smaller values of λ_i is similar to the performance with each λ_i set to 1. We tested two choices for the initial parameter vector: (i) we initialized with the zero vector (cold-start), and (ii) we initialized with the solution for $\lambda = 2$ (warm-start). We estimated the optimal value of the objective function, f^*, in our experiments by taking the lowest objective value found across the methods.

In Figures 2.1 (cold-start) and 2.2 (warm-start), we plot the logarithm of the objective function minus f^* and the number of non-zero variables against the number of function evaluations for the PSSgb method and the five methods that are not based on an L-BFGS approximation. In these plots, we plot the minimum objective value found by each method rather than the function value at each evaluation (the methods may explore higher values if backtracking is required, while the non-monotonic SPG/BBST/BBSG/OPG methods may spend multiple iterations exploring higher values). In these figures, we see that the PSSgb method outperforms the methods that are not based on L-BFGS. In particular, it obtains a lower objective value (for the same number of function evaluations), it more quickly identifies the correct set of non-zero variables, and it terminates earlier. Comparing the two figures, we see that the performance of the methods is closer if we use warm-starting, but that the PSSgb method still outperforms the other methods based on this measure. Among the three methods based on Barzilai-Borwein steps, the BBSG method was the most effective, the SPG method was the least effective, while the BBST method tended to be intermediary. There was no clearly superior method between the OPG method, the DSST method, and the methods based on the Barzilai-Borwein steps.

In Figures 2.3 (cold-start) and 2.4 (warm-start), we focus on the six methods that are based on an L-BFGS approximation. In these figures, we see that the three new PSS methods were the three most effective strategies across all experiments. The TMP method also does reasonably well on two of the data sets, but does poorly on the *thrombin* data. The OWL method was effective at initially driving down the objective function, but its progress slowed down on later iterations (presumably because the $\mathcal{P}_{\mathcal{S}}$ operator slowed down the local convergence rate). The AS method tended to perform very poorly except on the *thrombin* data set, presumably because the final solution was more sparse on this data set. Finally, we note that the PSSgb method seemed to be the least effective among the three new PSS methods, while the PSSas method was the most effective method in most scenarios.

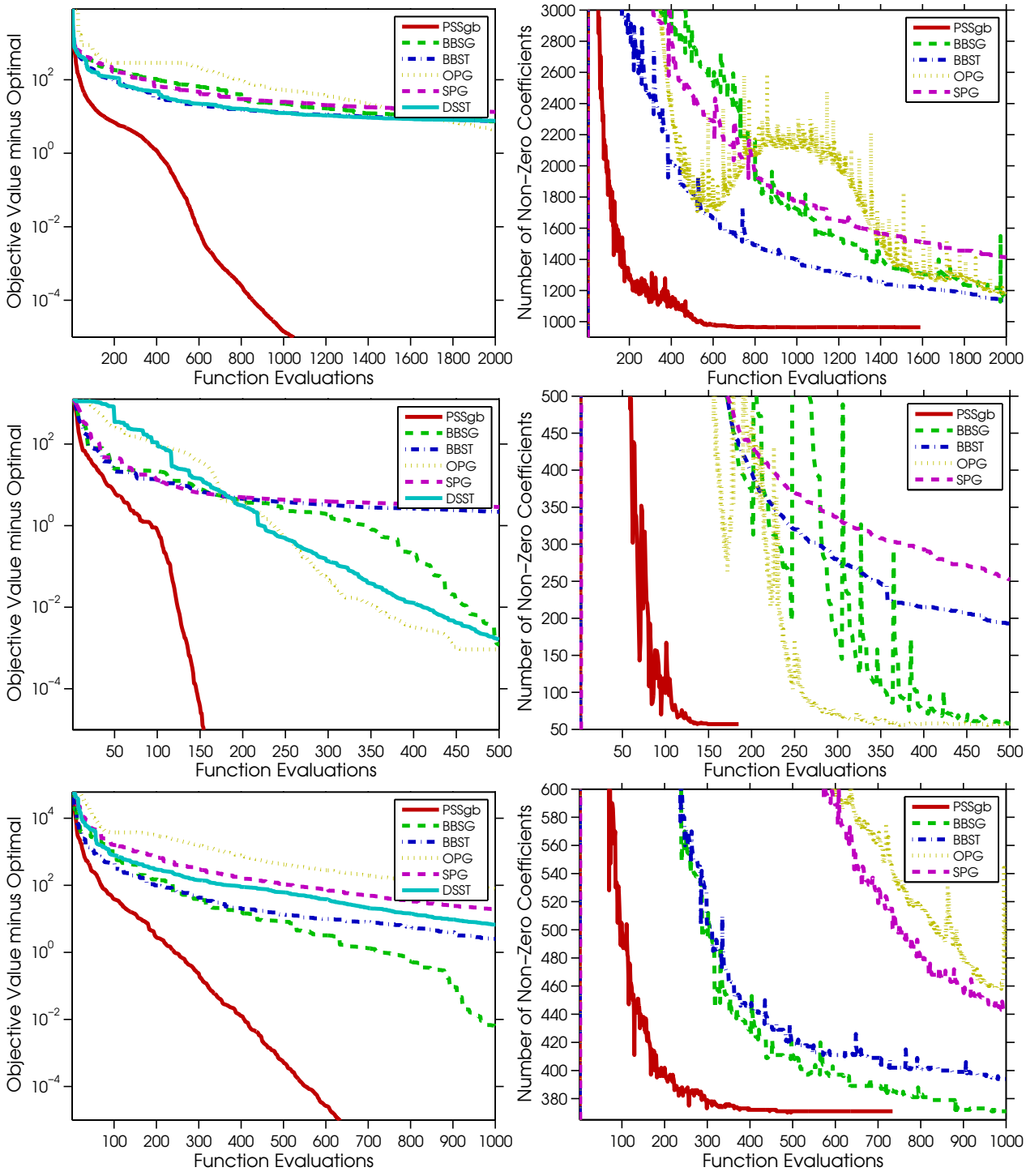

Figure 2.1: Function evaluations against objective value and number of non-zero coefficients for logistic regression ($\lambda = 1$) with ℓ_1-regularization for different optimization strategies initialized with the zero vector. Top to bottom: *sido* data, *thrombin* data, and *spam* data. This figure is best viewed in color.

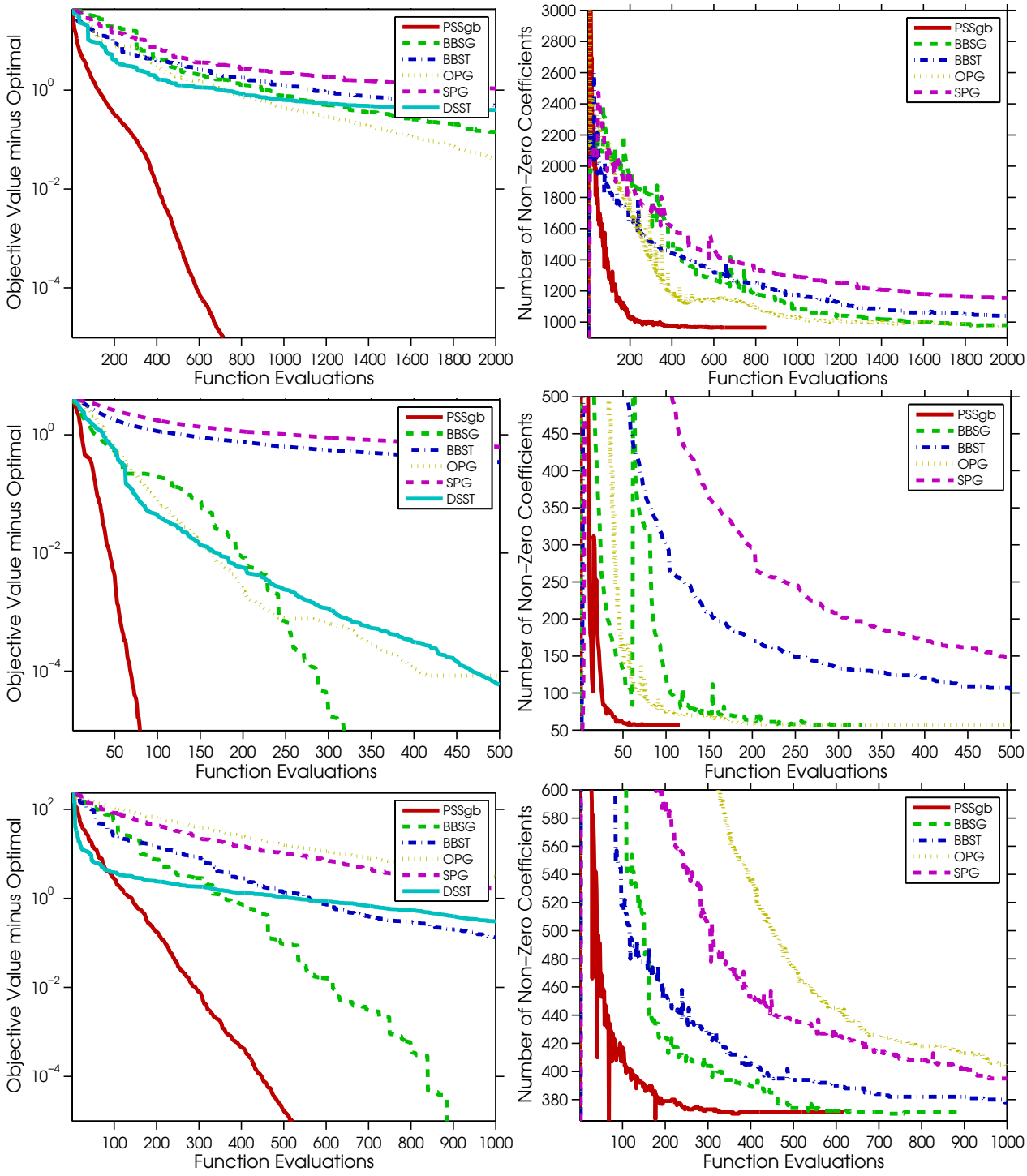

Figure 2.2: The same experiment as Figure 2.1, but using the optimal solution for $\lambda = 2$ as the starting vector.

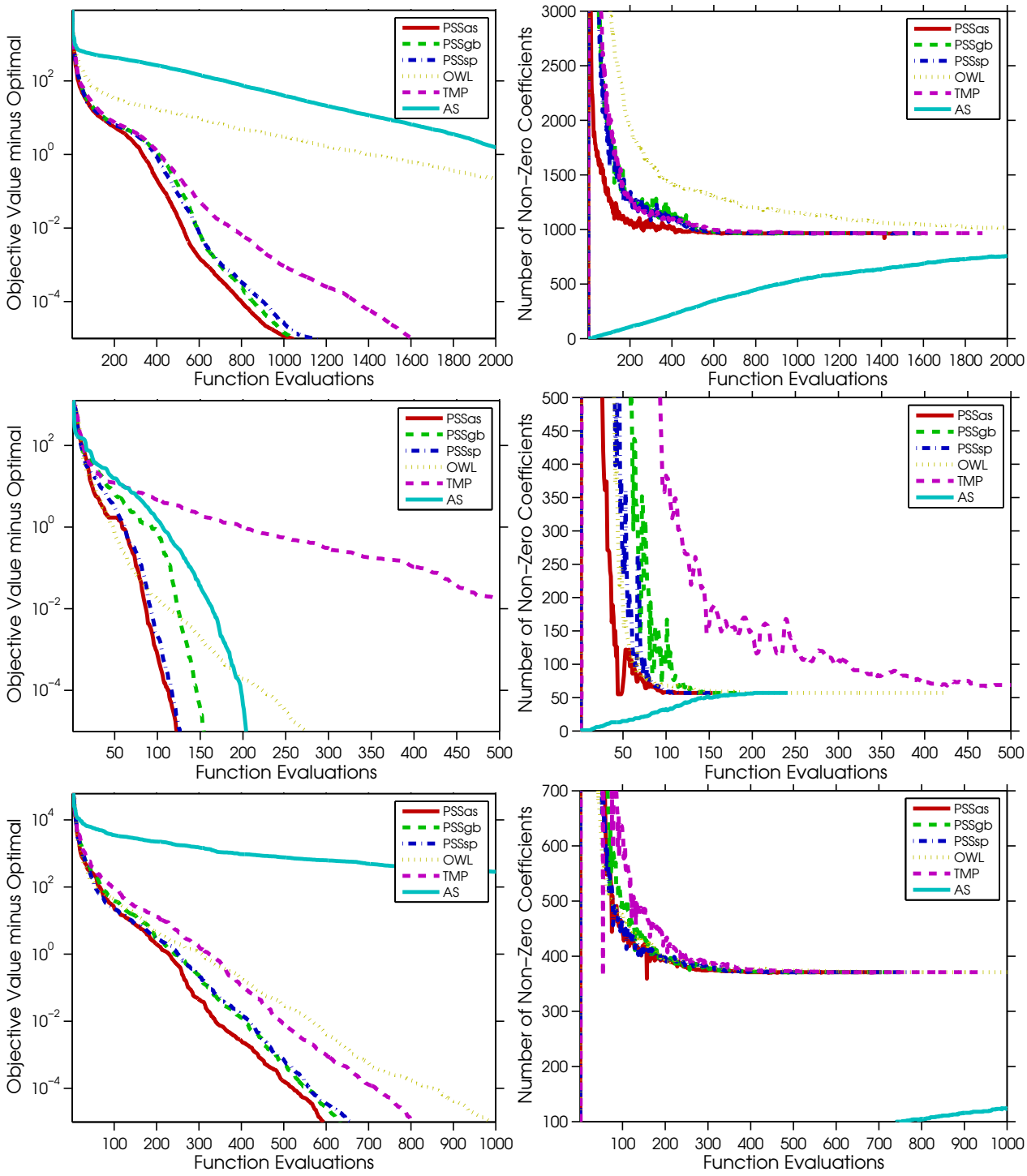

Figure 2.3: The same experiment as Figure 2.1, but focusing on methods that are based on L-BFGS.

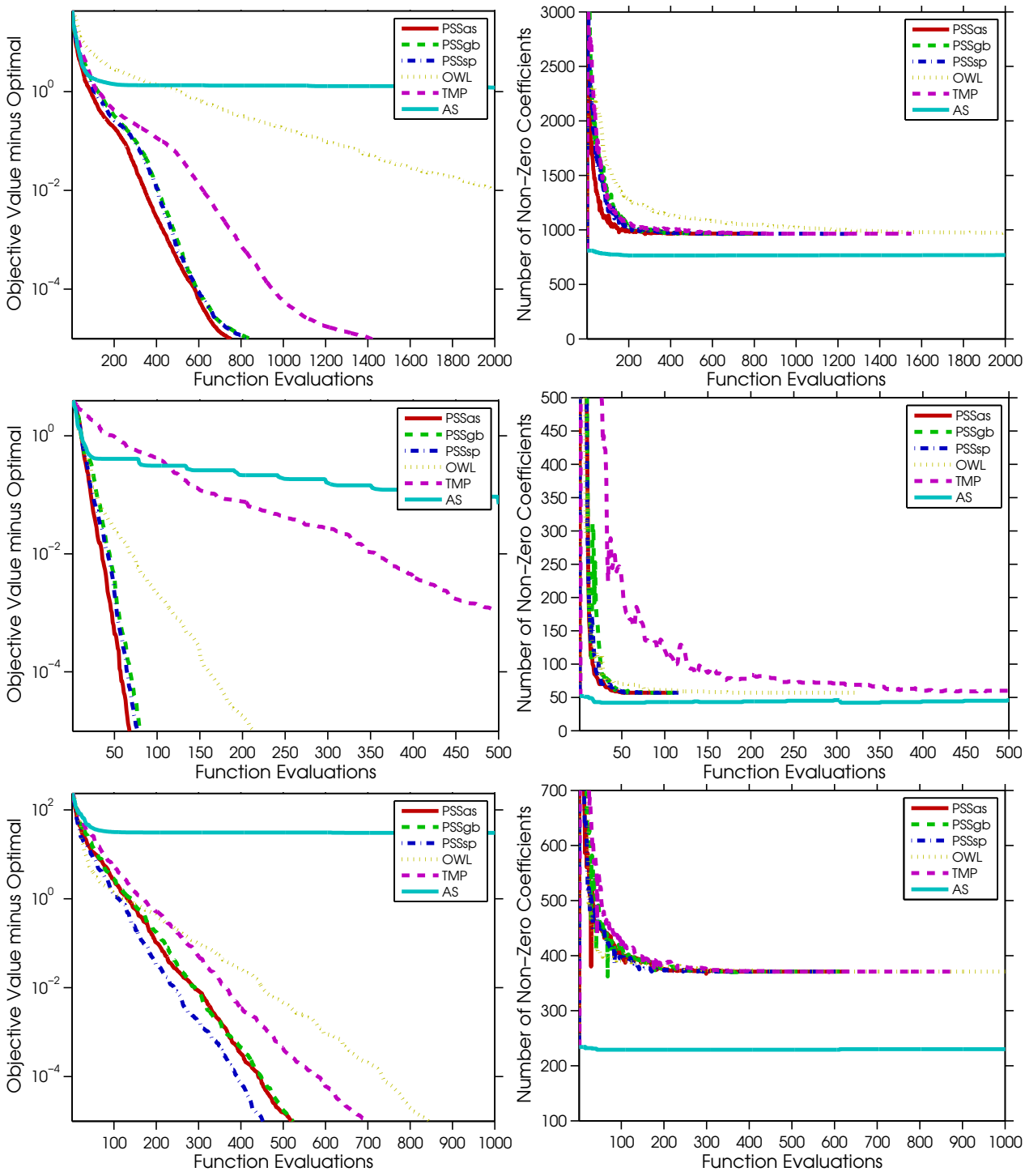

Figure 2.4: The same experiment as Figure 2.3, but using the optimal solution for $\lambda = 2$ as the starting vector.

2.6.2 Ising Graphical Models

We next tested the various optimization methods on the task of estimating ℓ_1-regularized IGMs. We did this to see if the trends observed in the logistic regression experiments carry over to related loss functions, and in particular if they carry over to using ℓ_1-regularization for structure learning in log-linear models for the special case where their is a direct correspondence between edges and parameters. Thus, we examined fitting ℓ_1-regularized IGMs to the *cyto* and *awma* data sets (where it is possible to compute the exact IGM objective) from Section 1.7. In these experiments we set the value of λ to 50, yielding a sufficiently difficult optimization problem that differences between the methods become apparent (for larger values of λ the methods are all very effective). We used essentially the same experimental set-up as in the case of logistic regression, with the following modifications: (i) we did not test the DSST method since even computing the diagonals of the Hessian is prohibitively expensive, and (ii) for the OPG method we used the same initial step length that the other methods used (we found that using $1/n$ gave poor performance for estimating IGMs).

In Figure 2.5, we plot the performance of the PSSgb method against the methods that are not based on L-BFGS, where we initialize with the zero vector and with the solution for $\lambda = 100$. As in the logistic regression experiments, the PSSgb method dominates the methods that are not based on L-BFGS. Further, we again see that BBSG is the best and SPG is the worst among the three methods based on the Barzilai-Borwein step size (SPG, BBST, and BBSG). However, unlike the logistic regression experiments we see in these experiments that the methods based on the Barzilai-Borwein step outperform the OPG method. Although it is possible that better performance could be obtained with the OPG method with a different choice of initial step size, we note that the performance of the other methods does not have this strong dependence on the initial step size.

In Figure 2.6, we plot the performance of the methods based on L-BFGS. In this plot, we again see that the new PSS methods typically outperform the other methods. The one exception to this was on the *awma* data with the warm-start, where the AS method proved very effective (since the set of non-zero variables didn't change much between $\lambda = 100$ and $\lambda = 50$). However, in the other scenarios the AS method is dominated by the new PSS methods.

2.7 Extensions

In this section, we consider several straightforward extensions of the work we describe in this chapter.

2.7.1 Other Objective Functions

We have presented an efficient large-scale optimization method for ℓ_1-regularized logistic regression. However, the only assumption needed in order to use this method is that the function we want to optimize with ℓ_1-regularization is differentiable and convex. We can further relax the assumption of convexity if we concede that the algorithm may find a local minimum that is not also a global minimum. Thus, we can apply this optimization algorithm in a wide variety of other scenarios. Besides the obvious problem of learning dependency networks with logistic regression conditionals (or other CPDs we discuss in Chapter 4), below we list several applications to structure learning:

- **Solving the graphical LASSO in the primal**: Most current methods for solving the graphical LASSO optimization problem (1.6) solve a Lagrangian dual of the optimization

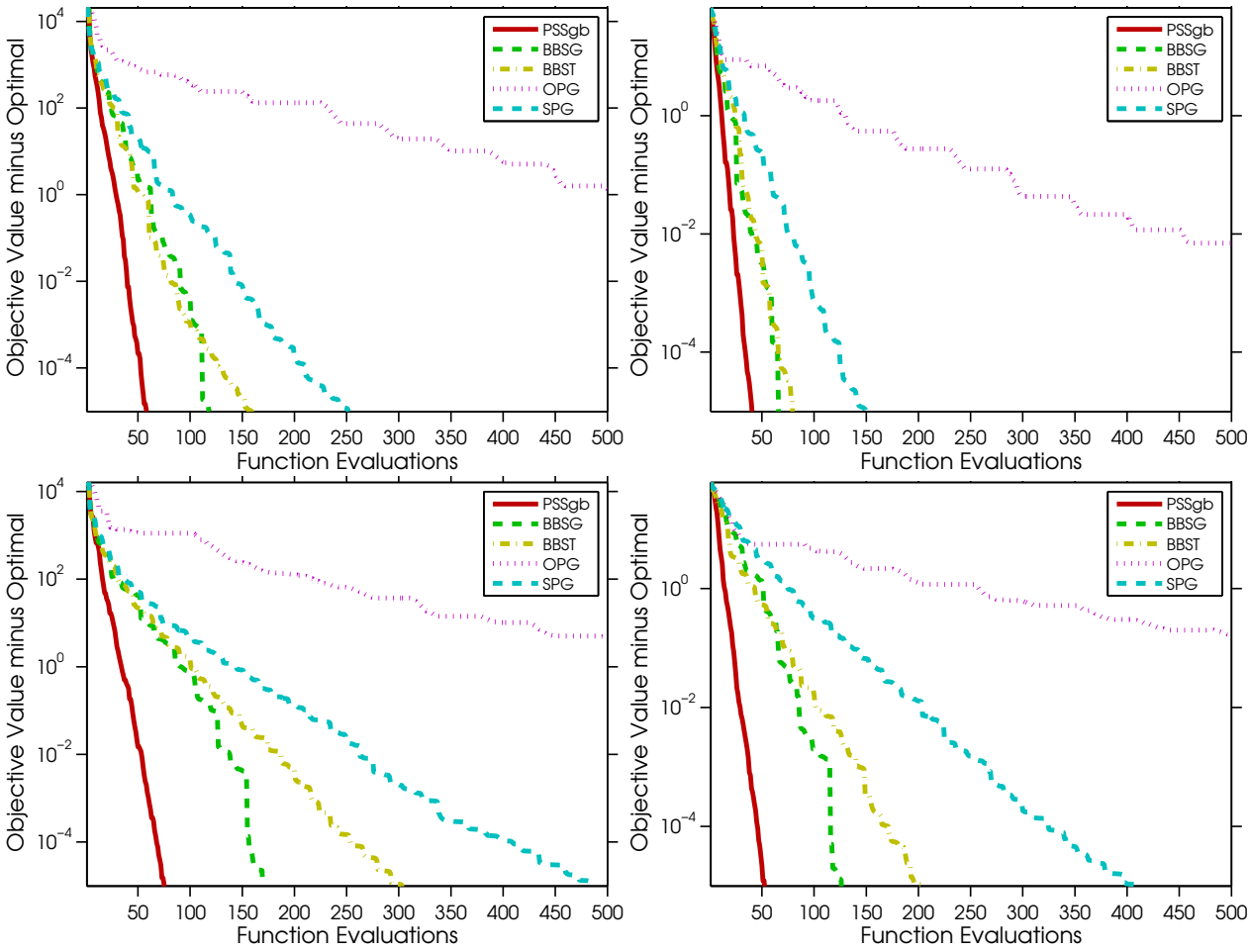

Figure 2.5: Function evaluations against objective value for training IGMs ($\lambda = 50$) with ℓ_1-regularization for different optimization strategies. Top row: *cyto* data. Bottom row: *awma* data. Left column: zero vector used for initialization. Right column: solution with $\lambda = 100$ used for initialization. This figure is best viewed in color.

problem [Banerjee et al., 2006, Friedman et al., 2008, Duchi et al., 2008a]. A potential disadvantage of working with the dual formulation is that the dual parameters are not sparse. In our experiments in [Marlin et al., 2009], we used the PSSgb algorithm to directly solve the graphical LASSO problem in the primal. Since the PSS iterations tend to be sparse, this lets us take advantage of techniques for sparse Cholesky factorizations [Rue and Held, 2005, §2.4] to efficiently evaluate the objective function in (1.6).

- **Sparse Conditional Random Fields**: Conditionals random fields are a class of log-linear models augmented with covariates, and they represent a natural generalization of logistic regression to the case where we have multiple target variables [Lafferty et al., 2001] (we discuss this type of model in more detail in Section 5.7). Goodman [2004] shows that training conditional random fields with ℓ_1-regularization offers improved performance in several natural language processing applications. The PSS algorithms can easily be applied in the case of

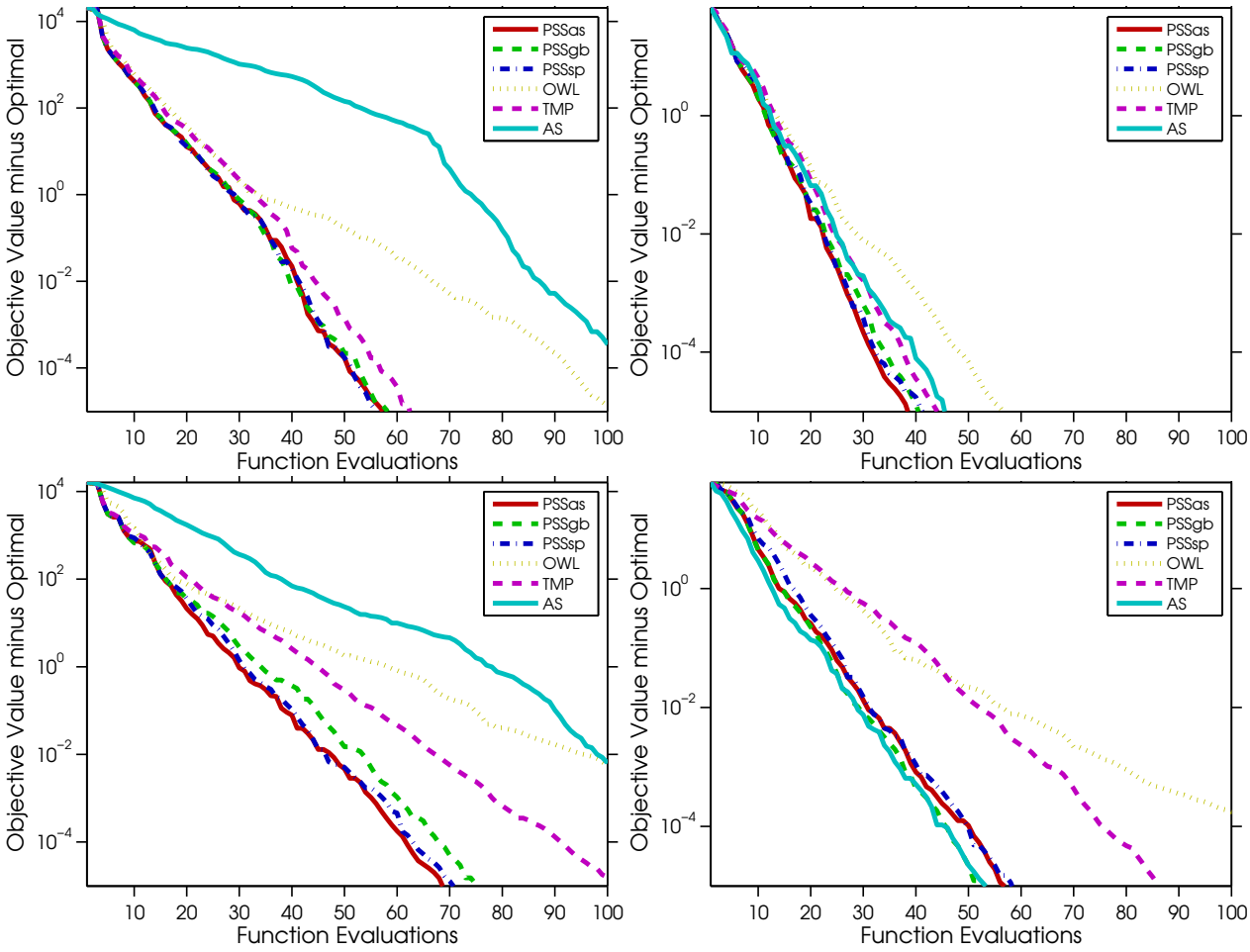

Figure 2.6: The same experiment as Figure 2.5, but focusing on methods based on an L-BFGS approximation.

conditional random fields (indeed, we do this in Chapter 5) to learn a sparse set of node and edge features.

- **Sparse neural networks**: Neural networks are a type of model that is widely used for non-linear regression and classification [see Bishop, 2006, §5]. In these models, the outputs are modeled through a sequence of non-linear transformations of the inputs. Typically, these non-linear transformations are the cumulative distribution function values for a set of linearly-parameterized logistic distributions with different parameters. Typically, each linearly-parameterized logistic distribution depends on the values of all variables in the previous layer, making the model very complex and difficult to interpret. To avoid over-fitting in these complex models, we typically use ℓ_2-regularization of the parameters. However, we can learn a sparse neural network model if we replace this ℓ_2-regularization with ℓ_1-regularization [Williams, 1995]. This can lead to much more parsimonious and interpretable models, since the elements of each layer will only depend on a subset of the variables in the previous layer. Although the objective function in this problem is non-convex, we can find a

local minimum of the (non-convex) objective function with the PSS methods.

2.7.2 Other Extensions

We conclude this chapter with several other possible extensions:

- **Hessian-Free Newton Methods**: Lin et al. [2007] recently showed that Hessian-free Newton methods can be competitive with L-BFGS for ℓ_2-regularized logistic regression. Rather than building a Hessian approximation, these methods seek to solve the Newton system $-\nabla^2 f(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$ up to a specific error tolerance by using Hessian-vector products within a linear conjugate gradient algorithm. The method is known as a *Hessian-free* Newton method because the Hessian-vector products can be computed without explicitly forming the Hessian. They are also known as *truncated* or *inexact* Newton methods because the Newton direction is only computed up to a specific error tolerance. It is straightforward to implement a Hessian-free version of the PSSgb or PSSsp methods where the linear conjugate gradient algorithm is used to solve the linear system involving the working set.

- **Improved Line Search**: It may be possible to improve the line search routine in various ways. Our line search uses a simple backtracking line search along the projection arc, with safeguarded cubic interpolation to generate trial values. This cubic interpolation ignores that the function is not smooth at locations where variables become exactly zero. Although a step size of 1 is typically accepted and the backtracking is typically not invoked (and only very rarely does the method backtrack more than once), it might be possible to get better performance by using a line search that takes advantage of the known locations of the non-differentiable points along the search direction. It might also be possible to consider non-smooth generalizations of the strong Wolfe conditions [Nocedal and Wright, 1999, §3.1] as a stronger measure of sufficient decrease than the Armijo condition.

- **Other Regularizers and Bound Constraints**: In principle, we could extend the PSS methods to find local optima in general problems of the form

$$\min_{\mathbf{x}} L(\mathbf{x}) + R(\mathbf{x}),$$

  where $L(\mathbf{x})$ is differentiable and $R(\mathbf{x})$ is continuous and separable into a set of functions that are each differentiable everywhere except at a countable number of (known) locations. This includes ℓ_1-regularization of differentiable objective functions as a special case, but also includes other regularizers such as the smoothly clipped absolute deviation (SCAD) penalty [Fan and Li, 2002]. To consider this case, we would need to re-define the $\mathcal{P}_{\mathcal{O}}$ projection operator so that it projects element-wise into the relevant interval where the function is differentiable, and re-define the pseudo-gradient so that its negative minimizes the directional derivative of the objective function. Further, by using ideas from the two-metric projection algorithm it would be straightforward to modify the PSS methods to incorporate lower and/or upper bounds on the variables.

Chapter 3

Optimization with Group ℓ_1-Regularization

In this chapter, we consider large-scale methods for solving optimization problems of the form

$$\min_{\mathbf{x}} f(\mathbf{x}) \triangleq L(\mathbf{x}) + \sum_{A} \lambda_A ||\mathbf{x}_A||_2. \tag{3.1}$$

where $L(\mathbf{x})$ is assumed to be convex and differentiable with respect to $\mathbf{x}$, and we may have a separate regularization parameter $\lambda_A \geq 0$ for each group A. In this chapter we assume that the groups A are disjoint. The algorithms we describe in this chapter are applicable to any optimization problem of this form, but our focus is on the case where $L(\mathbf{x})$ is the negative log-likelihood in a (possibly conditional) undirected graphical model. In this case, the optimization parameters $\mathbf{x}$ are the concatenation of the weights $\mathbf{w}$ and biases $\mathbf{b}$ (as well as feature weights $\mathbf{v}$ in the conditional case we discuss in Section 5.7), and each disjoint subset A contains all parameters associated with an individual edge in the model (though our discussion applies to other group structures, such as the blockwise-sparse models we discuss in Section 5.6). While this chapter focuses on the case of disjoint groups and penalizing the ℓ_2 norm of the groups, in Chapter 5 we extend the methods considered here for penalizing other norms of the groups while in Chapter 6 we extend the methods considered here to the case of overlapping groups.

Problem (3.1) is a generalization of the problem addressed in Chapter 2, in that we now penalize groups of variables instead of individual elements (we obtain problem (2.1) in the special case that each group A contains only one element). As before, this optimization is complicated by the non-differentiability of the regularizer term. In particular, the function is non-differentiable if an entire group of variables is exactly zero.

Since (conditional) undirected graphical models generalize logistic regression while group ℓ_1-regularization generalizes the previously examined ℓ_1-regularization, we might naturally consider extending the very efficient methods of Chapter 2 to solve this more general problem[20]. In Sections 3.1 we discuss applying methods based on the Barzilai-Borwein approximation to the group case, including the SPG and BBST methods of the previous chapter. However, these methods do not take into account that the objective function is very costly to evaluate, while the methods from Chapter 2 based on L-BFGS that require fewer evaluations (PSS, TMP, OWL) can not be extended in a straightforward way to the group case. Thus, in Section 3.2 we give new methods based on L-BFGS designed to reduce the number of objective evaluations (at the expense of a higher iteration cost).

[20] In the case of (unconditional) Gaussian graphical models or (unconditional) pairwise log-linear models with Ising potentials, each edge only has one parameter and the methods we discuss in Chapter 2 can be applied directly.

3.1 Barzilai-Borwein Methods

In this section discuss applying methods based on non-monotonic Barzilai-Borwein iterations, focusing on two variants. In the first variant, we formulate (3.1) as a differentiable optimization over a convex set and apply non-monotonic Barzilai-Borwein steps within a projected gradient iteration. This is referred to as a spectral projected gradient (SPG) algorithm. In the second variant, we apply non-monotonic Barzilai-Borwein steps within a soft-thresholding iteration that directly seeks to optimize (3.1). Due to the similarity to iterative soft-thresholding methods, we refer to this as a Barzilai-Borwein soft-threshold (BBST) algorithm.

3.1.1 Spectral Projected Gradient

In Chapter 2, we considered formulating the non-differentiable ℓ_1-regularized optimization problem as a smooth optimization problem with bound constraints. Then, we considered solving the bound constrained problem with the two-metric projection (TMP), optimal projected gradient (OPG), or SPG algorithm. Unfortunately, this problem transformation is no longer possible in the group case. However, it is still possible to transform (3.1) into a smooth optimization problem over a convex set. To do this, we introduce an additional variable g_A for each group A. We then replace each norm $||\mathbf{x}_A||_2$ with the variable g_A, and optimize subject to the constraint that $g_A \geq ||\mathbf{x}_A||_2$. That is, we solve problem

$$\min_{\mathbf{x},\mathbf{g}} \quad L(\mathbf{x}) + \sum_A \lambda_A \, g_A, \quad \text{subject to} \quad g_A \geq ||\mathbf{x}_A||_2, \forall_A. \tag{3.2}$$

This formulation replaces the non-linear, non-differentiable regularizer with a simple linear function. We note that these constraints are a special case of second-order cone constraints [Boyd and Vandenberghe, 2004, §4.4.2], and that each constraint defines a convex set called an ℓ_2 norm cone. For any feasible pair $\{\mathbf{x}, \mathbf{g}\}$, the objective function in (3.2) gives an upper bound on the objective (3.1), while at a minimizer it must be the case that $g_A = ||\mathbf{w}_A||_2$ for all groups (otherwise, we could decrease the objective by decreasing g_A to $||\mathbf{w}_A||_2$). It follows that because the range of $L(\mathbf{x})$ is unchanged a minimizer of (3.1) must correspond to a minimizer of (3.2). Although we can not apply the TMP algorithm for bound-constrained optimization to problem (3.2) because it is not a bound-constrained problem, we can still apply the SPG and OPG algorithms.

The projected-gradient method [Goldstein, 1964, Levitin and Poliak, 1966] is a constrained optimization algorithm for solving

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}),$$

where $f(\mathbf{x})$ is a differentiable function and $\mathcal{C}$ is a closed convex set. We consider a variant of the method that uses iterations of the form

$$\mathbf{x}_{k+1} \leftarrow \mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)).$$

Here, α is selected to satisfy the Armijo condition by a backtracking line search and $\mathcal{P}_{\mathcal{C}}$ is defined by

$$\mathcal{P}_{\mathcal{C}}(\mathbf{x}) \triangleq \arg\min_{\mathbf{y}\in\mathcal{C}} ||\mathbf{x} - \mathbf{y}||_2, \tag{3.3}$$

the Euclidean projection onto $\mathcal{C}$. This simple general-purpose method has two drawbacks: (i) in general solving (3.3) may itself be a computationally challenging problem, and (ii) the use of the steepest descent step results in slow convergence.

The SPG method [Birgin et al., 2000] uses two simple modifications of the projected gradient method to enhance its convergence rate. First, it initializes the line search with the step size

$$\alpha_{bb} = \frac{\mathbf{y}_k^T \mathbf{s}_k}{\mathbf{y}_k^T \mathbf{y}_k},$$

proposed by [Barzilai and Borwein, 1988]. Second, it uses a non-monotonic version of the Armijo condition [Grippo et al., 1986]:

$$f(\mathbf{x}_{k+1}) \leq \max_{i=k-m:k} f(\mathbf{x}_i) + \nu \nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k), \quad \text{with } \nu \in (0,1). \tag{3.4}$$

This non-monotonic Armijo condition typically accepts α_{bb} (even if it increases the objective function), but still ensures global convergence of the method. A typical value for the number m of previous function values to consider is 10. These two simple modifications have been shown to experimentally lead to a very large improvement in the convergence rate of the method, and due to its strong empirical performance SPG has recently been explored in several applications [Dai and Fletcher, 2005, Figueiredo et al., 2007, van den Berg and Friedlander, 2008].

Although the SPG strategy reduces the number of iterations of the method that we must perform, for the method to be efficient we must still be able to efficiently compute the projection onto the constraint set. Fortunately, in problem (3.2) each constraint only affects variables associated with the corresponding group. Thus, we can compute the projection across the groups by independently solving the projection problem for each group. For each group, the corresponding problem takes the form

$$\mathcal{P}_{\mathcal{C}_2}(\mathbf{x}_A, g_A) = \arg\min_{\mathbf{y},z} \left|\left|\begin{bmatrix} \mathbf{x}_A \\ g_A \end{bmatrix} - \begin{bmatrix} \mathbf{y} \\ z \end{bmatrix}\right|\right|_2, \text{ subject to } z \geq ||\mathbf{y}||_2.$$

The solution to this problem is [Boyd and Vandenberghe, 2004, Exercise 8.3(c)]

$$\mathcal{P}_{\mathcal{C}_2}(\mathbf{x}, g) = \begin{cases} (\mathbf{x}, g), & \text{if } ||\mathbf{x}||_2 \leq g, \\ (\frac{\mathbf{x}}{||\mathbf{x}||_2} \frac{||\mathbf{x}||_2 + g}{2}, \frac{||\mathbf{x}||_2 + g}{2}), & \text{if } ||\mathbf{x}||_2 > g, ||\mathbf{x}||_2 + g > 0, \\ (0,0), & \text{if } ||\mathbf{x}||_2 > g, ||\mathbf{x}||_2 + g \leq 0. \end{cases}$$

We give an explicit derivation of this result in Appendix B. Thus, we can solve this sub-projection in $\mathcal{O}(|A|)$ and we can solve the full projection in $\mathcal{O}(p)$ for a problem with p variables[21]. We close this section by noting that we could alternatively use this constrained formulation and the above projection operator within an OPG method [Nesterov, 2004, §2.2.4].

3.1.2 Barzilai-Borwein Soft Threshold

A wide variety of authors have recently considered using a class of algorithms known as iterative soft-thresholding (or forward-backward splitting) for optimization with sparse regularizers, including [Daubechies et al., 2004, Combettes and Wajs, 2005, Elad et al., 2006, Hale et al., 2007, Nesterov, 2007, Duchi and Singer, 2009] These methods addresses problems of the form

$$\min_{\mathbf{x}} f(\mathbf{x}) \triangleq L(\mathbf{x}) + R(\mathbf{x}). \tag{3.5}$$

[21] We show how to solve the related problem of projecting onto the norm ball defined by the ℓ_1 of ℓ_2 norms in [van den Berg et al., 2008].

Here, $R(\mathbf{x})$ is convex and possibly non-differentiable, while $L(\mathbf{x})$ is assumed to be differentiable and convex with a Lipschitz-continuous gradient. Rather than converting this problem to a constrained optimization problem, these algorithms solve the non-smooth optimization problem directly with a projection-like operator. In particular, these method take steps of the form

$$\mathbf{x}_{k+1} \leftarrow \mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \alpha \nabla L(\mathbf{x}_k), \alpha). \tag{3.6}$$

Here, we have used $\mathcal{S}_{\mathcal{R}}(\mathbf{x}, \alpha)$ to denote the solution of a 'soft-threshold' problem at $\mathbf{x}$ with step size α and regularizer $R(\mathbf{x})$. Specifically, the soft-threshold operator is given by the solution to the soft-threshold problem

$$\mathcal{S}_{\mathcal{R}}(\mathbf{x}, \alpha) \triangleq \arg\min_{\mathbf{y}} \frac{1}{2}||\mathbf{y} - \mathbf{x}||_2^2 + \alpha R(\mathbf{y}). \tag{3.7}$$

In our case, $R(\mathbf{x}) \triangleq \sum_A \lambda_A ||\mathbf{x}_A||_2$ so the soft-threshold step for problem (3.1) would be

$$\arg\min_{\mathbf{y}} \frac{1}{2}||\mathbf{y} - (\mathbf{x}^k - \alpha \nabla L(\mathbf{x}^k))||_2^2 + \alpha \sum_A \lambda_A ||\mathbf{y}_A||_2.$$

Thus, we first take a step along the negative gradient of the loss function, and then compute this projection-like soft-threshold operator to take into account the effect of the regularizer. The latter step effectively sparsifies the result of the (generally dense) gradient step. As discussed by [Combettes and Wajs, 2005], the soft-threshold operator is a generalization of the projection operator, and we recognize the iterative soft-thresholding algorithm as the classic gradient-projection algorithm but with projection replaced by soft-thresholding. Similar to the classic gradient projection algorithm, this algorithm may converge very slowly. However, analogous to the SPG algorithm, Wright et al. [2009] propose to use Barzilai-Borwein steps and a non-monotonic line search to speed the convergence of the method. We refer to this method as the Barzilai-Borwein soft-threshold (BBST) method[22].

Wright et al. [2009] discuss computing the soft-thresholding operator in the case of group ℓ_1-regularization. As before, the operator separates into solving a simple problem for each group. The solution for an individual group is

$$\mathcal{S}_{\mathcal{R}_2}(\mathbf{x}_A, \alpha) = \text{sgn}(\mathbf{x}_A) \max\{0, ||\mathbf{x}_A||_2 - \alpha\lambda_A\},$$

where we use $\text{sgn}(\mathbf{y})$ to denote a set-valued function that returns $\mathbf{y}/||\mathbf{y}||_2$ if $\mathbf{y} \neq \mathbf{0}$, and returns all values such that $||\mathbf{y}||_2 \leq 1$ if $\mathbf{y} = 0$.

In the next section we give an L-BFGS extension of the BBST algorithm, but first we establish some useful properties of the method (that will also apply in the new method). First, we note that computing $\mathbf{x}_{k+1}$ in (3.6) is equivalent to solving the optimization problem

$$\arg\min_{\mathbf{y}} \quad L(\mathbf{x}_k) + (\mathbf{y} - \mathbf{x}_k)^T \nabla L(\mathbf{x}_k) + \frac{1}{2\alpha}||\mathbf{y} - \mathbf{x}_k||_2^2 + R(\mathbf{y}). \tag{3.8}$$

Thus, we can view the soft-threshold step as the solution of a first-order approximation of $L(\mathbf{x})$ at $\mathbf{x}_k$, that is regularized by $R(\mathbf{x})$ as well as the distance to $\mathbf{x}_k$. Nesterov [2007] refers to (3.8) as the composite gradient mapping, while Wright et al. [2009] refers to it as a separable approximation. Using this equivalent formulation, we can establish that an iterate $\mathbf{x}_k$ is an optimal solution to the

[22] Soft-threshold variants of the OPG method are discussed in [Nesterov, 2007].

original problem if and only if $\mathbf{x}_k$ solves (3.8). To see this, first note that the sub-differential of our original optimization problem (3.5) is

$$\partial f(\mathbf{x}) = \nabla L(\mathbf{x}) + \partial R(\mathbf{x}).$$

A vector $\mathbf{x}^*$ is a minimizer of a convex function if and only if $\mathbf{0} \in \partial f(\mathbf{x}^*)$ [Bertsekas, 1999, §B.5]. The sub-differential of the objective function in (3.8) (that we denote by $q^k(\mathbf{y})$) is

$$\partial q^k(\mathbf{y}) = \nabla L(\mathbf{x}_k) + \frac{1}{\alpha}(\mathbf{y} - \mathbf{x}_k) + \partial R(\mathbf{y}).$$

Thus, if $\mathbf{y} = \mathbf{x}_k$ then the optimality conditions for (3.8) reduce to $\mathbf{0} \in \nabla L(\mathbf{x}_k) + \partial R(\mathbf{x}_k)$ and this is equivalent to $\mathbf{x}_k$ being an optimal solution [see also Combettes and Wajs, 2005, Proposition 3.1].

By re-writing the soft-threshold operator in the form (3.8), we can use an argument similar to [Bertsekas, 1999, Exercise 6.3.11] to establish the useful property that if the solution $\mathbf{x}_k^*$ to (3.8) is not a minimizer of $f(\mathbf{x})$, then $f(\mathbf{x}_k^*) < f(\mathbf{x}_k)$ for sufficiently small α. To do this, first note that $\mathbf{x}_k$ achieves an objective value of $L(\mathbf{x}_k) + R(\mathbf{x}_k)$ in (3.8), thus if $\mathbf{x}_k$ is not a minimizer of $f(\mathbf{x})$ then $\mathbf{x}_k^*$ achieves a lower objective value in (3.8) and we have

$$\begin{aligned} L(\mathbf{x}_k) + R(\mathbf{x}_k) &> L(\mathbf{x}_k) + (\mathbf{x}_k^* - \mathbf{x}_k)^T \nabla L(\mathbf{x}_k) + \frac{1}{2\alpha}||\mathbf{x}_k^* - \mathbf{x}_k||_2^2 + R(\mathbf{x}_k^*) \\ &\geq L(\mathbf{x}_k^*) + R(\mathbf{x}_k^*) \ \ (\text{for } 0 < \alpha \leq 1/\mathcal{L}). \end{aligned} \tag{3.9}$$

The last line follows from [Bertsekas, 1999, Proposition A.24], where $\mathcal{L}$ is the Lipschitz constant of the gradient of $L(\mathbf{x})$. This result is also given by [Nesterov, 2007, Theorem 1 and Remark 1], and a related result that backtracking along α satisfies a modified Armijo condition is given by [Wright et al., 2009, Lemma 3]. We note that the gradient of the negative log-likelihood in an undirected model is Lipschitz continuous because the gradient is continuously differentiable and the spectral norm of the Hessian is bounded. We also note that the descent property still holds if $L(\mathbf{x})$ is only locally Lipschitz continuous. Finally, an important property that is relevant to the next section is that (3.9) holds not only for the result of the soft-threshold operator, but for any $\mathbf{x}_k^*$ that achieves a lower objective value than $\mathbf{x}_k$ in (3.8).

3.2 Quasi-Newton Methods

The Barzilai-Borwein methods discussed in the previous section represent some of the most efficient methods currently available for solving problem (3.1). However, compared to simple objectives like logistic regression a complicating factor in optimizing the parameters of undirected graphical models is that it is very expensive to evaluate the objective function. Further, in our experiments in Chapter 2 we saw that the SPG, OPG, and BBST methods typically require many more function evaluations than methods that are based on an L-BFGS Hessian approximation (such as the PSS, TMP, and OWL methods). Unfortunately, the methods based on L-BFGS updates from Chapter 2 do not admit a straightforward extension to the group case. This is because we do not have an operator that is analogous to the $\mathcal{P}_{\mathcal{O}}$ orthant-projection from Chapter 2 (that sparsifies the solution and truncates the line search to a region where the Taylor expansion is valid). However, in the previous section we showed that we can convert problem (3.1) to a differentiable constrained optimization where it is straightforward to compute the projection onto the feasible set. Motivated by

problems with this structure, in [Schmidt et al., 2009b] we gave a limited-memory projected quasi-Newton (PQN) algorithm that uses an L-BFGS Hessian approximation to solve high-dimensional constrained optimization problems where it is substantially more expensive to evaluate the objective function than it is to project onto the feasible set. We review this method next. Subsequently, we consider a variant of this method that incorporates an L-BFGS Hessian approximation into a soft-thresholding algorithm.

3.2.1 Projected Quasi-Newton

As with the gradient-projection method, projected Newton methods address the problem of minimizing a function $f(\mathbf{x})$ over a convex set $\mathcal{C}$. Similar to unconstrained Newton-like methods, at each iteration projected Newton methods consider a quadratic approximation of the objective function around the current iterate $\mathbf{x}_k$:

$$q_k(\mathbf{x}) \triangleq f(\mathbf{x}_k) + (\mathbf{x} - \mathbf{x}_k)^T \nabla f(\mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^T B_k (\mathbf{x} - \mathbf{x}_k). \tag{3.10}$$

Here, B_k is a positive-definite approximation to the Hessian. In order to generate a direction of search that is both a descent direction and feasible, projected Newton methods find the minimizer x_k^* of this quadratic approximation over the set $\mathcal{C}$. That is, they solve

$$\mathbf{x}_k^* \triangleq \arg\min_{\mathbf{x} \in \mathcal{C}} q_k(\mathbf{x}). \tag{3.11}$$

This generates a descent direction $\mathbf{d} \triangleq \mathbf{x}_k^* - \mathbf{x}_k$, where $\mathbf{x}_k + \alpha \mathbf{d}_k$ is feasible for $\alpha \in [0, 1]$. As before, we can use this direction as part of a backtracking line search until we have a new iterate satisfying the Armijo condition. If B_k is the exact Hessian and we always test $\alpha = 1$ first, this method has a quadratic rate of convergence in the neighborhood of a minimizer satisfying second-order sufficiency conditions [Bertsekas, 1999, Proposition 2.3.5]. The drawbacks of this method in its unmodified form are that: (i) it requires computing/storing a dense p by p Hessian approximation, and (ii) finding the constrained minimizer of the quadratic model may be very expensive.

We use the L-BFGS Hessian approximation to address the first issue. As mentioned in Chapter 2, there is an efficient recursive formula that pre-multiplies a vector by the inverse of a matrix $B_0 = \sigma_k I$ updated m times with the BFGS formula. However, in order to evaluate the objective function in (3.11) we need to be able to multiply by B_k, not B_k^{-1}. This can be done using the compact representation of Byrd et al. [1994], that represents the updates B_k as a low rank matrix

$$B_k = \sigma_k I - N M^{-1} N^T, \tag{3.12}$$

where N is p-by-$2m$, and M is $2m$-by-$2m$. With this representation, we can compute $q_k(\mathbf{x})$ and $\nabla q_k(\mathbf{x})$ in $\mathcal{O}(mp)$ (both values can be obtained with one multiplication by B_k).

Given the L-BFGS representation of B_k, we minimize (3.11) by using the SPG algorithm discussed in the previous section. In addition to evaluating $q_k(\mathbf{x})$ (and its gradient), the cost of running SPG is dominated by computing the projection $\mathcal{P}_{\mathcal{C}}$. However, note that we do not need to evaluate the objective function in the SPG sub-routine. Hence, the proposed method is most effective on problems where computing the projection is much less expensive than evaluating the objective function[23]. In the case of group ℓ_1-regularized undirected graphical models, we can compute the projection in linear time while evaluating the objective function is #P-hard in general

[23]This is different than many classical optimization problems like quadratic programming, where evaluating the objective function is relatively inexpensive and computing the projection may be as difficult as solving the original problem.

(even evaluating the approximate objective functions from Section 1.4 will typically be much more costly than computing the projection). Thus, the conditions needed for the PQN method to be efficient are clearly satisfied.

In general, running the SPG sub-routine to obtain a high-accuracy solution may be computationally expensive. However, we must be careful about terminating the SPG sub-routine early because an approximate solution to (3.11) will not in general be a descent direction. Fortunately, we can guarantee that the SPG sub-routine yields a descent direction even under early termination if we initialize it with $\mathbf{x}_k$ and we run the method for at least one iteration (so that we obtain a vector $\mathbf{y}$ satisfying the Armijo condition on the quadratic approximation). To see this, first note that positive-definiteness of B_k implies that a sufficient condition for $\mathbf{y} - \mathbf{x}_k$ to be a descent direction for some vector $\mathbf{y}$ is that $q_k(\mathbf{y}) < f(\mathbf{x}_k)$ (since this implies that $(\mathbf{y} - \mathbf{x}_k)^T \nabla f(\mathbf{x}_k) < 0$). Subsequently, using $q_k(\mathbf{x}_k) = f(\mathbf{x}_k)$ we have that $q_k(\mathbf{y}) < f(\mathbf{x}_k)$ where $\mathbf{y}$ is the first point satisfying the Armijo condition on $q_k(\mathbf{x})$ if we initialize SPG with $\mathbf{x}_k$. Thus, if we initialize the SPG sub-routine with $\mathbf{x}_k$ then after the first iteration (and every subsequent iteration) the SPG solution gives a descent direction and it can safely be terminated early. Further, provided that the eigenvalues of the Hessian approximation B_k are bounded, the search directions generated by the SPG sub-routine are gradient related [see Bertsekas, 1999, §1.2] after the first iteration. Convergence of the PQN method thus follows from [Bertsekas, 1999, Proposition 2.2.1]. In our implementation we include an explicit maximum c on the number of iterations to run the SPG sub-routine for[24]. For problems where computing the projection onto the constraint set can be done in $\mathcal{O}(p)$, the iteration cost of the PQN method is therefore $\mathcal{O}(pmc)$.

3.2.2 Quasi-Newton Soft Threshold

The PQN method is a general technique for constrained optimization, and we can apply it in the special case of group ℓ_1-regularization problems after a suitable problem transformation. However, we saw in the last section that the soft-threshold operator provides a direct way to apply Barzilai-Borwein steps to solve group ℓ_1-regularization problems (in that we don't have to introduce auxiliary variables). In this section, we consider a method that is analogous to the PQN algorithm, but that is suitable for optimizing the sum of a costly objective function $L(\mathbf{x})$ (with Lipschitz-continuous gradient) and a convex regularizer $R(\mathbf{x})$ where we can efficiently compute the soft-threshold operator for the regularizer. We call this the quasi-Newton soft-threshold (QNST) algorithm.

At each iteration of the QNST algorithm, we form a regularized quadratic approximation to the function

$$q_k^\alpha(\mathbf{x}) \triangleq L(\mathbf{x}_k) + (\mathbf{x} - \mathbf{x}_k)^T \nabla L(\mathbf{x}_k) + \frac{1}{2\alpha}(\mathbf{x} - \mathbf{x}_k)^T B_k (\mathbf{x} - \mathbf{x}_k) + R(\mathbf{x}), \tag{3.13}$$

where B_k is an L-BFGS approximation of $\nabla^2 L(\mathbf{x}_k)$. That is, we use a quadratic approximation to the smooth function $L(\mathbf{x})$ but include the regularizer explicitly in the sub-problem. To find an (approximate) minimizer $\mathbf{x}_{k+1}$ of this sub-problem, we use c iterations of the BBST method. To set the step size length α, we use a backtracking line search.

Besides the use of a soft-thresholding method to solve (3.13), there is a close connection between the QNST method and soft-thresholding algorithms. We can see this by re-writing the optimization

[24] An alternative strategy would be to run the method until the sub-problem is solved up to a certain optimality tolerance. This tolerance could then be set using a forcing sequence [Nocedal and Wright, 1999, §6.1]

over (3.13) as follows:

$$\begin{aligned}
&\arg\min_{\mathbf{y}} \; L(\mathbf{x}_k) + (\mathbf{y}-\mathbf{x}_k)^T\nabla L(\mathbf{x}_k) + \frac{1}{2\alpha}(\mathbf{y}-\mathbf{x}_k)^T B_k(\mathbf{y}-\mathbf{x}_k) + R(\mathbf{y}) \\
=&\arg\min_{\mathbf{y}} \; (\mathbf{y}-\mathbf{x}_k)^T\nabla L(\mathbf{x}_k) + \frac{1}{2\alpha}(\mathbf{y}-\mathbf{x}_k)^T B_k(\mathbf{y}-\mathbf{x}_k) + R(\mathbf{y}) \\
=&\arg\min_{\mathbf{y}} \; \alpha(\mathbf{y}-\mathbf{x}_k)^T\nabla L(\mathbf{x}_k) + \frac{1}{2}(\mathbf{y}-\mathbf{x}_k)^T B_k(\mathbf{y}-\mathbf{x}_k) + \alpha R(\mathbf{y}) \\
=&\arg\min_{\mathbf{y}} \; \frac{1}{2}\alpha^2\nabla L(\mathbf{x}_k)^T B_k^{-1}\nabla L(\mathbf{x}_k) + \alpha(\mathbf{y}-\mathbf{x}_k)^T\nabla L(\mathbf{x}_k) + \frac{1}{2}(\mathbf{y}-\mathbf{x}_k)^T B_k(\mathbf{y}-\mathbf{x}_k) + \alpha R(\mathbf{y}) \\
=&\arg\min_{\mathbf{y}} \; \frac{1}{2}((\mathbf{y}-\mathbf{x}_k) + \alpha B_k^{-1}\nabla L(\mathbf{x}_k))^T B_k((\mathbf{y}-\mathbf{x}_k) + \alpha B_k^{-1}\nabla L(\mathbf{x}_k)) + \alpha R(\mathbf{y}) \\
=&\arg\min_{\mathbf{y}} \; \frac{1}{2}||(\mathbf{y}-\mathbf{x}_k) + \alpha B_k^{-1}\nabla L(\mathbf{x}_k))||_{B_k}^2 + \alpha R(\mathbf{y}) \\
=&\arg\min_{\mathbf{y}} \; \frac{1}{2}||\mathbf{y} - (\mathbf{x}_k - \alpha B_k^{-1}\nabla L(\mathbf{x}_k))||_{B_k}^2 + \alpha R(\mathbf{y}).
\end{aligned}$$

Here, we use $||\cdot||_H$ to denote the quadratic norm $||\mathbf{x}||_H = (\mathbf{x}^T H\mathbf{x})^{-1/2}$. In the last line, we see that the solution $\mathbf{x}_k^*$ of (3.13) is the result of a generalized soft-thresholding step

$$\mathbf{x}_k^* \leftarrow \mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \alpha B_k^{-1}\nabla L(\mathbf{x}_k), \alpha, B_k),$$

where we define the generalized soft-threshold operator $\mathcal{S}_{\mathcal{R}}(\mathbf{x}, \alpha, H)$ as

$$\mathcal{S}_{\mathcal{R}}(\mathbf{x}, \alpha, H) \triangleq \arg\min_y \frac{1}{2}||\mathbf{y}-\mathbf{x}||_H^2 + \alpha R(\mathbf{x}).$$

Thus, we see that the QNST method can be viewed as taking a standard unconstrained L-BFGS step on $L(\mathbf{x})$, followed by applying a soft-threshold operation with regularizer $R(\mathbf{x})$ where we measure distance based on the quasi-Newton approximation[25]. We obtain the standard soft-thresholding algorithm if we fix B_k to I. We note that this is analogous to the relationship between projected gradient and projected (quasi-)Newton methods [Bertsekas, 1999, §2.3]. Indeed, we obtain a standard unconstrained quasi-Newton method for differentiable optimization (as we describe in Section 2.1) if $R(\mathbf{x})$ is a constant function and we solve the sub-problem exactly. Further, the QNST method can be viewed as a generalization of the PQN method, since we obtain a version of the PQN method if $R(\mathbf{x})$ is an extended real-valued function that returns 0 if $\mathbf{x} \in \mathcal{C}$ and returns ∞ otherwise. This suggests that we could also use the QNST method to minimize differentiable function with simple non-differentiable regularizers over simple convex sets (provided that the soft-threshold operation can still be computed efficiently).

The steps of the QNST algorithm can be viewed as steps of a standard soft-threshold algorithm for minimizing $L(B_k^{-1/2}\tilde{\mathbf{x}}) + R(B_k^{-1/2}\tilde{\mathbf{x}})$ in terms of $\tilde{\mathbf{x}}$, which is equivalent to (3.5) with the transformation $\tilde{\mathbf{x}} = B_k^{1/2}\mathbf{x}$. It follows from our argument of the previous section that the QNST algorithm has the descent property that if $\mathbf{x}_k$ is not an optimal solution, then $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ for sufficiently small α. Finally, it follows from a similar argument to the one made in the PQN section, combined with (3.9), that we can terminate the BBST sub-rouinte early provided that we initialize it with $\mathbf{x}_k$ and find a solution with a lower objective value in the regularized quadratic approximation.

[25]Convergence rates under different choices of norm for soft-thresholding algorithms are discussed in [Chen and Rockafellar, 1997].

3.3 Implementation

In Algorithm 6 we give pseudo-code for the SPG method.

Input: Objective function $f(\mathbf{x})$, projection function $\mathcal{P}_{\mathcal{C}}(\mathbf{x})$, initial parameter vector $\mathbf{x}_0$, optimality tolerance ϵ, number of previous function value to store m, sufficient decrease parameter η, line search safeguard parameters ξ_1 and ξ_2, step length upper and lower limits $\alpha_{\max}$ and $\alpha_{\min}$.

$k \leftarrow 0$;
$\mathbf{x}_0 \leftarrow \mathcal{P}_{\mathcal{C}}(\mathbf{x}_0)$; // project initial parameter vector
$f_k \leftarrow f(\mathbf{x}_0)$; // evaluate objective function
$\mathbf{g}_k \leftarrow \nabla f(\mathbf{x}_0)$; // compute gradient
while $||\mathbf{x}_k - \mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \mathbf{g}_k)||_\infty > \epsilon$ **do**
  if $k = 0$ **then**
    $\alpha \leftarrow -\min(1, 1/||\mathbf{g}_k||_1)$; // initial step size
  else
    $\alpha \leftarrow \mathbf{y}_k^T \mathbf{s}_k / \mathbf{y}_k^T \mathbf{y}_k$; // Barzilai-Borwein step size
    $\alpha \leftarrow \max(\alpha_{\min}, \min(\alpha_{\max}, \alpha))$; // Safeguarded BB step
  $\mathbf{x}_{k+1} \leftarrow \mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \alpha \mathbf{g}_k)$; // initial trial value
  $f_{k+1} \leftarrow f(\mathbf{x}_{k+1})$; // evaluate new parameter vector
  $\mathbf{g}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$; // compute new gradient
  while $f_{k+1} > \max_{i=k-m:k} f_i + \eta \mathbf{g}_k^T (\mathbf{x}_{k+1} - \mathbf{x}_k)$ **do**
    Select $\alpha \in (\xi_1 \alpha, \xi_2 \alpha)$; // safeguarded cubic interpolation
    $\mathbf{x}_{k+1} \leftarrow \mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \alpha \mathbf{g}_k)$; // new trial value
    $f_{k+1} \leftarrow f(\mathbf{x}_{k+1})$; // evaluate new parameter vector
    $\mathbf{g}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$; // compute new gradient
  $\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$; // compute quasi-Newton differences
  $\mathbf{y}_k \leftarrow \mathbf{g}_{k+1} - \mathbf{g}_k$;
  $k \leftarrow k + 1$;

Algorithm 6: Spectral projected gradient algorithm for minimizing a function $f(\mathbf{x})$ over a convex set $\mathcal{C}$.

Note that the above algorithm uses one of the two step sizes proposed by Barzilai and Borwein [1988], we can use the alternate step size by simply replacing the appropriate line in the code above. Also, in the above code we are backtracking along the projection arc [see Bertsekas, 1999, §2.3]. Birgin et al. [2000] also considered a variant where we backtrack along a feasible direction. The latter strategy is more appealing in cases where the projection is expensive to compute.

The BBST algorithm is identical to SPG, with the following modifications: (i) we do not project the initial vector, (ii) we define f_k as $L(\mathbf{x}_k) + R(\mathbf{x}_k)$ but g_k as $\nabla L(\mathbf{x}_k)$, (iii) in the optimality condition we replace $\mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \mathbf{g}_k)$ with $\mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \mathbf{g}_k, 1)$, (iv) in the iterate update we replace $\mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \alpha\mathbf{g}_k)$ with $\mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \alpha\mathbf{g}_k, \alpha)$, and (v) in the non-monotonic Armijo condition we replace $\mathbf{g}_k^T(\mathbf{x}_{k+1} - \mathbf{x}_k)$ with α multiplied by the directional derivative of the objective at $\mathbf{x}_k$ in the direction $(\mathbf{x}_{k+1} - \mathbf{x}_k)$. We give pseudo-code for the BBST method below, where we use $R'(\mathbf{x}; \mathbf{y})$ to denote the directional derivative of $R(\mathbf{x})$ evaluated at $\mathbf{x}$ in the direction of $\mathbf{y}$[26].

Input: Differentiable convex function $f(\mathbf{x})$, regularization function $R(\mathbf{x})$, soft-threshold function $\mathcal{S}_{\mathcal{R}}(\mathbf{x})$, initial parameter vector $\mathbf{x}_0$, optimality tolerance ϵ, number of previous function value to store m, sufficient decrease parameter η, line search safeguard parameters ξ_1 and ξ_2, step length upper and lower limits $\alpha_{\max}$ and $\alpha_{\min}$.

$k \leftarrow 0$;
$f_k \leftarrow f(\mathbf{x}_0) + R(\mathbf{x}_0)$; // evaluate objective function
$\mathbf{g}_k \leftarrow \nabla f(\mathbf{x}_0)$; // compute gradient
while $||\mathbf{x}_k - \mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \mathbf{g}_k, 1)||_\infty > \epsilon$ **do**
  if $k = 0$ **then**
    $\alpha \leftarrow -\min(1, 1/||\mathbf{g}_k||_1)$; // initial step size
  else
    $\alpha \leftarrow \mathbf{y}_k^T\mathbf{s}_k / \mathbf{y}_k^T\mathbf{y}_k$; // Barzilai-Borwein step size
    $\alpha \leftarrow \max(\alpha_{\min}, \min(\alpha_{\max}, \alpha))$; // Safeguarded BB step
  $\mathbf{x}_{k+1} \leftarrow \mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \alpha\mathbf{g}_k, \alpha)$; // initial trial value
  $f_{k+1} \leftarrow f(\mathbf{x}_{k+1}) + R(\mathbf{x}_{k+1})$; // evaluate new parameter vector
  $\mathbf{g}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$; // compute new gradient
  while $f_{k+1} > \max_{i=k-m:k} f_i + \eta\alpha(\mathbf{g}_k + R'(\mathbf{x}_k; \mathbf{x}_{k+1} - \mathbf{x}_k))^T(\mathbf{x}_{k+1} - \mathbf{x}_k)$ **do**
    Select $\alpha \in (\xi_1\alpha, \xi_2\alpha)$; // safeguarded cubic interpolation
    $\mathbf{x}_{k+1} \leftarrow \mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \alpha\mathbf{g}_k, \alpha)$; // new trial value
    $f_{k+1} \leftarrow f(\mathbf{x}_{k+1}) + R(\mathbf{x}_{k+1})$; // evaluate new parameter vector
    $\mathbf{g}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$; // compute new gradient
  $\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$; // compute quasi-Newton differences
  $\mathbf{y}_k \leftarrow \mathbf{g}_{k+1} - \mathbf{g}_k$;
  $k \leftarrow k + 1$;

Algorithm 7: Barizilai-Borwein soft-threshold algorithm for minimizing the sum of a differentiable convex function $f(\mathbf{x})$ and a convex regularizer $R(\mathbf{x})$.

[26] If this directional derivative is difficult to compute, we could alternately use $\alpha||\mathbf{x}_{k+1} - \mathbf{x}_k||_2^2$ in the Armijo condition as in [Wright et al., 2009].

In Algorithm 8 we give pseudo-code for the PQN method. In this pseudo-code, we find it convenient to use $\text{SPG}(\mathbf{x}_k, c, \mathbf{g}_k, \sigma, S, Y)$ to denote applying c iterations of SPG starting from $\mathbf{x}_k$ to approximately solve problem (3.11) with the gradient set to $\mathbf{g}_k$ and with the L-BFGS approximation (3.12) constructed using σ, S, and Y.

Input: Objective function f, projection function $\mathcal{P}_{\mathcal{C}}$, inital parameter vector $\mathbf{x}_0$, optimality tolerance ϵ, number of corrections m, sufficient decrease parameter η, line search safeguard parameters ξ_1 and ξ_2, maximum number of SPG iterations c.

$k \leftarrow 0$;
$\mathbf{x}_0 \leftarrow \mathcal{P}_{\mathcal{C}}(\mathbf{x}_0)$; // project initial parameter vector
$f_k \leftarrow f(\mathbf{x}_0)$; // evaluate objective function
$\mathbf{g}_k \leftarrow \nabla f(\mathbf{x}_0)$; // compute gradient
while $||\mathbf{x}_k - \mathcal{P}_{\mathcal{C}}(\mathbf{x}_k - \mathbf{g}_k)||_\infty > \epsilon$ **do**
  $\alpha = 1$;
  if $k = 0$ **then**
    $\mathbf{d}_k = -\mathbf{g}_k \min(1, 1/||\mathbf{g}_k||_1)$; // use steepest descent
  else
    $\mathbf{x}_k^* \leftarrow \text{SPG}(\mathbf{x}_k, c, \mathbf{g}_k, \sigma, S, Y)$; // approximately minimize quadratic approximation
    $\mathbf{d}_k \leftarrow \mathbf{x}_k^* - \mathbf{x}_k$; // feasible descent direction
  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \mathbf{d}_k$; // initial trial value
  $f_{k+1} \leftarrow f(\mathbf{x}_{k+1})$; // evaluate new parameter vector
  $\mathbf{g}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$;
  while $f_{k+1} > f_k + \eta \mathbf{g}_k^T(\mathbf{x}_{k+1} - \mathbf{x}_k)$ **do**
    Select $\alpha \in (\xi_1 \alpha, \xi_2 \alpha)$; // safeguarded cubic interpolation
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \mathbf{d}_k$; // new trial value
    $f_{k+1} \leftarrow f(\mathbf{x}_{k+1})$; // evaluate new parameter vector
    $\mathbf{g}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$;
  $\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$; // compute quasi-Newton differences
  $\mathbf{y}_k \leftarrow \mathbf{g}_{k+1} - \mathbf{g}_k$;
  if $k > m$ **then**
    Remove oldest vector from S and Y;
  $S \leftarrow [S \quad \mathbf{s}_k]$; // update quasi-Newton difference matrices
  $Y \leftarrow [Y \quad \mathbf{y}_k]$;
  $\sigma \leftarrow (\mathbf{y}_k^T \mathbf{s}_k)/(\mathbf{y}_k^T \mathbf{y}_k)$; // update diagonal Hessian scaling
  $k \leftarrow k + 1$;

Algorithm 8: Limited-memory projected quasi-Newton algorithm for minimizing a function $f(\mathbf{x})$ over a convex set $\mathcal{C}$.

In this code, we have used backtracking along the feasible direction, but we could also consider a variant of the method where we backtrack along the projection arc [see Bertsekas, 1999, §2.3]. Here, during the iterations of the line search we would incorporate the step size α into the quadratic approximation (3.11) (similar to the QNST method discussed next), and use SPG to directly solve for $\mathbf{x}_{k+1}$ (increasing the cost of backtracking, but possibly generating better trial values).

We obtain the QNST algorithm by using the same replacements we used to obtain the BBST algorithm from the SPG algorithm, in addition to: (i) replacing SPG by BBST, (ii) replacing (3.11) by (3.13), and (iii) directly solving for $\mathbf{x}_{k+1}$ for the trial value of α instead of computing $\mathbf{d}_k$ and then setting $\mathbf{x}_{k+1}$ to $\mathbf{x}_k + \alpha\mathbf{d}_k$ (both before and during the line search). We give pseudo-code for the QNST method below, where $\text{BBST}(\mathbf{x}_k, c, \mathbf{g}_k, \sigma, S, Y, \alpha)$ is defined analogously to the SPG function in the PQN pseudo-code (but augmented to include the step size α).

Input: Differentiable convex function $f(\mathbf{x})$, regularization function $R(\mathbf{x})$, soft-threshold function $\mathcal{S}_{\mathcal{R}}(\mathbf{x})$, initial parameter vector $\mathbf{x}_0$, optimality tolerance ϵ, number of corrections m, sufficient decrease parameter η, line search safeguard parameters ξ_1 and ξ_2, maximum number of BBST iterations c.

$k \leftarrow 0$;
$f_k \leftarrow f(\mathbf{x}_0) + R(\mathbf{x}_0)$; // evaluate objective function
$\mathbf{g}_k \leftarrow \nabla f(\mathbf{x}_0)$; // compute gradient
while $||\mathbf{x}_k - \mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \mathbf{g}_k, 1)||_\infty > \epsilon$ **do**
  if $k = 0$ **then**
    $\alpha \leftarrow \min(1, 1/||\mathbf{g}_k||_1)$; // initial step size
    $\mathbf{x}_{k+1} = \mathcal{S}_{\mathcal{R}}(\mathbf{x}_k - \alpha\mathbf{g}_k, \alpha)$; // use basic soft-threshold step
  else
    $\alpha \leftarrow 1$;
    $\mathbf{x}_{k+1} \leftarrow \text{BBST}(\mathbf{x}_k, c, \mathbf{g}_k, \sigma, S, Y, \alpha)$; // approximately minimize approximation
  $f_{k+1} \leftarrow f(\mathbf{x}_{k+1}) + R(\mathbf{x}_{k+1})$; // evaluate new parameter vector
  $\mathbf{g}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$;
  while $f_{k+1} > f_k + \eta\alpha(\mathbf{g}_k + R'(\mathbf{x}_k; \mathbf{x}_{k+1} - \mathbf{x}_k))^T(\mathbf{x}_{k+1} - \mathbf{x}_k)$ **do**
    Select $\alpha \in (\xi_1\alpha, \xi_2\alpha)$; // safeguarded cubic interpolation
    $\mathbf{x}_{k+1} \leftarrow \text{BBST}(\mathbf{x}_k, c, \mathbf{g}_k, \sigma, S, Y, \alpha)$; // new trial value
    $f_{k+1} \leftarrow f(\mathbf{x}_{k+1}) + R(\mathbf{x}_{k+1})$; // evaluate new parameter vector
    $\mathbf{g}_{k+1} \leftarrow \nabla f(\mathbf{x}_{k+1})$;
  $\mathbf{s}_k \leftarrow \mathbf{x}_{k+1} - \mathbf{x}_k$; // compute quasi-Newton differences
  $\mathbf{y}_k \leftarrow \mathbf{g}_{k+1} - \mathbf{g}_k$;
  if $k > m$ **then**
    Remove oldest vector from S and Y;
  $S \leftarrow [S \quad \mathbf{s}_k]$; // update quasi-Newton difference matrices
  $Y \leftarrow [Y \quad \mathbf{y}_k]$;
  $\sigma \leftarrow (\mathbf{y}_k^T\mathbf{s}_k)/(\mathbf{y}_k^T\mathbf{y}_k)$; // update diagonal Hessian scaling
  $k \leftarrow k + 1$;

Algorithm 9: Limited-memory quasi-Newton soft-threshold algorithm for minimizing the sum of a differentiable convex function $f(\mathbf{x})$ and a convex regularizer $R(\mathbf{x})$.

3.4 Regularization Path and Active-Set Optimization

As with the methods from Chapter 2, the methods we discuss in this chapter can make use of good starting parameter values when we want to solve for multiple values of λ. In this section, we consider a method for solving for a sequence of values of λ that is analogous to the one we discuss in Section 2.5.

Consider the following set of necessary and sufficient conditions for a vector $\mathbf{x}$ to be a minimizer of $f(\mathbf{x})$ for given values of λ_A:

$$\begin{cases} \nabla_A L(\mathbf{x}) + \lambda_A \operatorname{sgn}(\mathbf{x}_A) = 0, & \mathbf{x}_A \neq \mathbf{0}, \\ ||\nabla_A L(\mathbf{x})||_2 \leq \lambda_A, & \mathbf{x}_A = \mathbf{0}. \end{cases}$$

These conditions are equivalent to the necessary and sufficient optimality condition that the zero-vector is an element of the sub-differential of (3.1). Similar to (2.5), these conditions allow us to determine the value of λ_{max} that sets all (regularized) groups to zero (after we have optimized with respect to the unregularized variables). In particular, if we denote the unregularized variables by $\mathbf{b}$ and the regularized variables by $\mathbf{w}$, then we have

$$\lambda_{max} \triangleq \max_A ||\nabla_{\mathbf{w}_A} L(\mathbf{0}, \tilde{\mathbf{b}})||_2,$$

where $\tilde{\mathbf{b}}$ optimizes $L(\mathbf{w}, \mathbf{b})$ with respect to $\mathbf{b}$ (with $\mathbf{w}$ fixed at $\mathbf{0}$).

Analogous to the method in Section 2.5, we could consider the following active-set method:

- Find groups A such that $\mathbf{x}_A \neq \mathbf{0}$, or $\mathbf{x}_A = \mathbf{0}$ and $||\nabla_A L(\mathbf{x})||_2 > \lambda_A$.
- Solve the problem with respect to these groups.

We can again consider applying this procedure for a decreasing sequence of values of the regularization parameter. The only difference between this procedure and the procedure of Section 2.5 is that the selection of variables to include in the optimization is done at the group level rather than the individual variable level. However, the computational gains achieved by applying this strategy to undirected graphical models can be much more dramatic than the gains achieved for logistic regression. In particular, for large values of λ the graph defined on the subset of groups that we optimize over will have low treewidth and thus we can evaluate the objective function efficiently. Thus, for sufficiently large values of λ we can evaluate the objective function exactly in polynomial time, while the objective function associated with the corresponding ℓ_2-regularization problem (where the graph is dense) will require exponential time even for large values of λ.

3.5 Experiments

We compared the performance of several large-scale optimization methods for group ℓ_1-regularized (unconditional) log-linear models. In particular, we compared the following methods:

- **SPG**: The spectral projected gradient method we discuss in Section 3.1.1.
- **OPG**: The optimal projected gradient using the line search suggested in [Liu et al., 2009], applied to the constrained formulation we discuss in Section 3.1.1.
- **BBST**: The Barzilai-Borwein soft-threshold method we discuss in Section 3.1.2.

- **PQN10**: The projected quasi-Newton method we discuss in Section 3.2.1, where we run the SPG sub-routine for 10 iterations.
- **PQN100**: The projected quasi-Newton method we discuss in Section 3.2.1, where we run the SPG sub-routine for 100 iterations.
- **QNST10**: The quasi-Newton soft-threshold method we discuss in Section 3.2.2, where we run the BBST sub-routine for 10 iterations.
- **QNST100**: The quasi-Newton soft-threshold method we discuss in Section 3.2.2, where we run the BBST sub-routine for 100 iterations.

Although other methods exist, our experiments in [van den Berg et al., 2008] indicated that the SPG algorithm outperformed several competing methods for estimating conditional log-linear models, while in [Schmidt et al., 2009a] our experiments indicated that both SPG and PQN outperformed competing methods for estimating log-linear models and (blockwise-sparse) Gaussian graphical models. We tested the methods on the two data sets from Section 1.7 where we can evaluate the objective function exactly, namely the *cyto* and *awma* data sets. We used the same experimental setup and optimization parameters as in Chapter 2. We set the optimality tolerance for the SPG and BBST sub-routines to be 10^{-6}, and the tolerance for lack of progress in these sub-routines at 10^{-10}. We set the value of λ to 50, yielding a sufficiently difficult problem that differences between the methods become apparent (for larger values of λ, the methods perform similarly).

3.5.1 Pairwise Log-Linear Models

In our first experiment we used full potentials and we initialized the methods with all elements of $\mathbf{b}$ and $\mathbf{w}$ set to zero. Figure 3.1 plots the logarithm of objective function value minus f^* and number of non-zero edges against the number of function evaluations (in this case, the extreme cost of function evaluations makes this a very good surrogate for the runtimes of the various methods). As in the case of ℓ_1-regularized logistic regression, in this experiment the methods based on L-BFGS (PQN and QNST) outperformed the other methods (SPG, OPG, and BBST). This was true even for the PQN10 and QNST10 methods, that only make limited use of the second-order approximation. We also see that the PQN100 and QNST100 methods that solve the direction finding sub-problem more accurately tend to give better performance than the PQN10 and QNST10 methods. In Figure 3.2, we repeat the experiment but initialize the methods with the solution for $\lambda = 100$. We see that the methods have better performance with this initialization, but we see the same trends across the methods.

3.5.2 Ising Graphical Models

Our second experiment sought to test whether the PQN and QNST are competitive with the most effective method from Chapter 2 (the PSSas method), in the special case of IGMs where each group has only one variable and the methods from either this chapter or Chapter 2 can be applied. We thus applied the group ℓ_1-regularization methods in the experimental set-up from Section 2.6.2. We compare the group ℓ_1-regularization methods to the PSSas method in Figure 3.3. Here, we see that the QNST10, PQN100, and QNST100 methods have similar performance to the PSSas method even though they use an approximate solution of the sub-problem (though the lower iteration cost makes the PSSas method more appealing for regular ℓ_1-regularization problems), while the PQN10

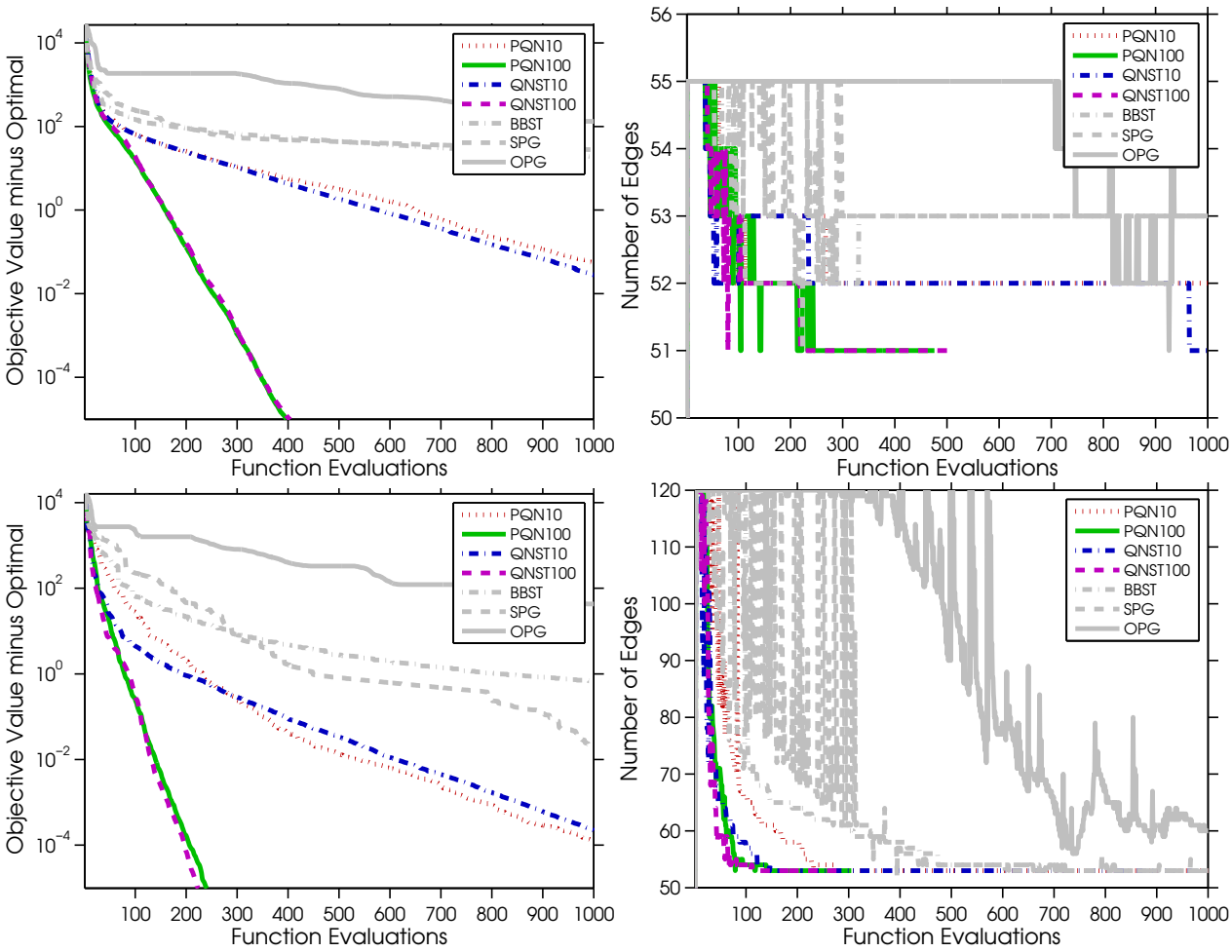

Figure 3.1: Function evaluations and number of edges against objective value and number of non-zero coefficients for training a log-linear model with full potentials and group ℓ_1-regularization for different optimization strategies initialized with the zero vector ($\lambda = 50$). The top row is for the *cyto* data and the bottom row is for the *awma* data. This figure is best viewed in color.

method performed similarly to the PSSas method on the *cyto* data but slightly worse on the *awma* data.

3.6 Extensions

The PQN and QNST represent general optimization strategies for optimizing high-dimensional costly objective functions subject to simple constraints or regularizers, respectively. Hence, they may also be useful other optimization problems. We encounter several examples in Chapters 5 and 6. Below, we give several examples:

- **Blockwise-sparse graphical models**: In [Schmidt et al., 2009b] we use PQN to solve the Lagrangian dual of the blockwise-sparse GGM model examined in Duchi et al. [2008a], and that we discuss further in Chapter 5. We could alternately consider applying PQN with the

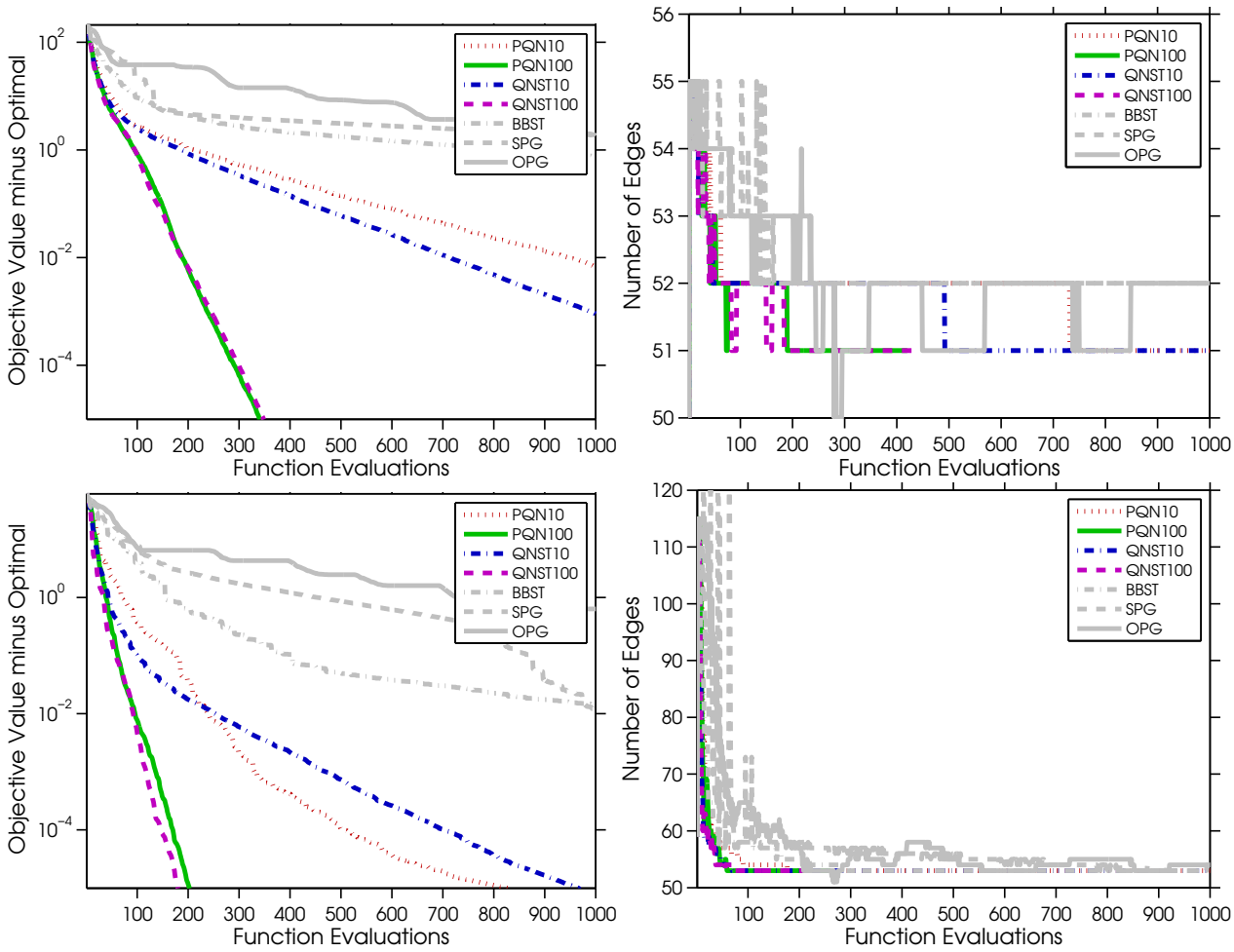

Figure 3.2: The same experiment as Figure 3.1, but using the optimal solution for $\lambda = 100$ as the starting vector.

constrained formulation to solve the primal problem, or applying QNST directly to the primal problem (the advantage of solving in the primal is that the primal variables are sparse).

- **Feature selection in conditional random fields**: In many applications of conditional random fields we have either non-binary discrete target variables, or categorical features that are represented as a set of binary indicator variables. In both cases, there is more than one variable associated with each feature and we must consider group ℓ_1-regularization to encourage sparsity in terms of the features. Since the objective function in these scenarios is costly to evaluate, the PQN and QNST methods are well-suited to solving the resulting optimization problems. Further, in Chapter 5 we discuss performing structure learning in conditional random fields with group ℓ_1-regularization. In this scenario the objective function is even more costly to evaluate than in log-linear models, so the advantages of the PQN and QNST methods are more pronounced.

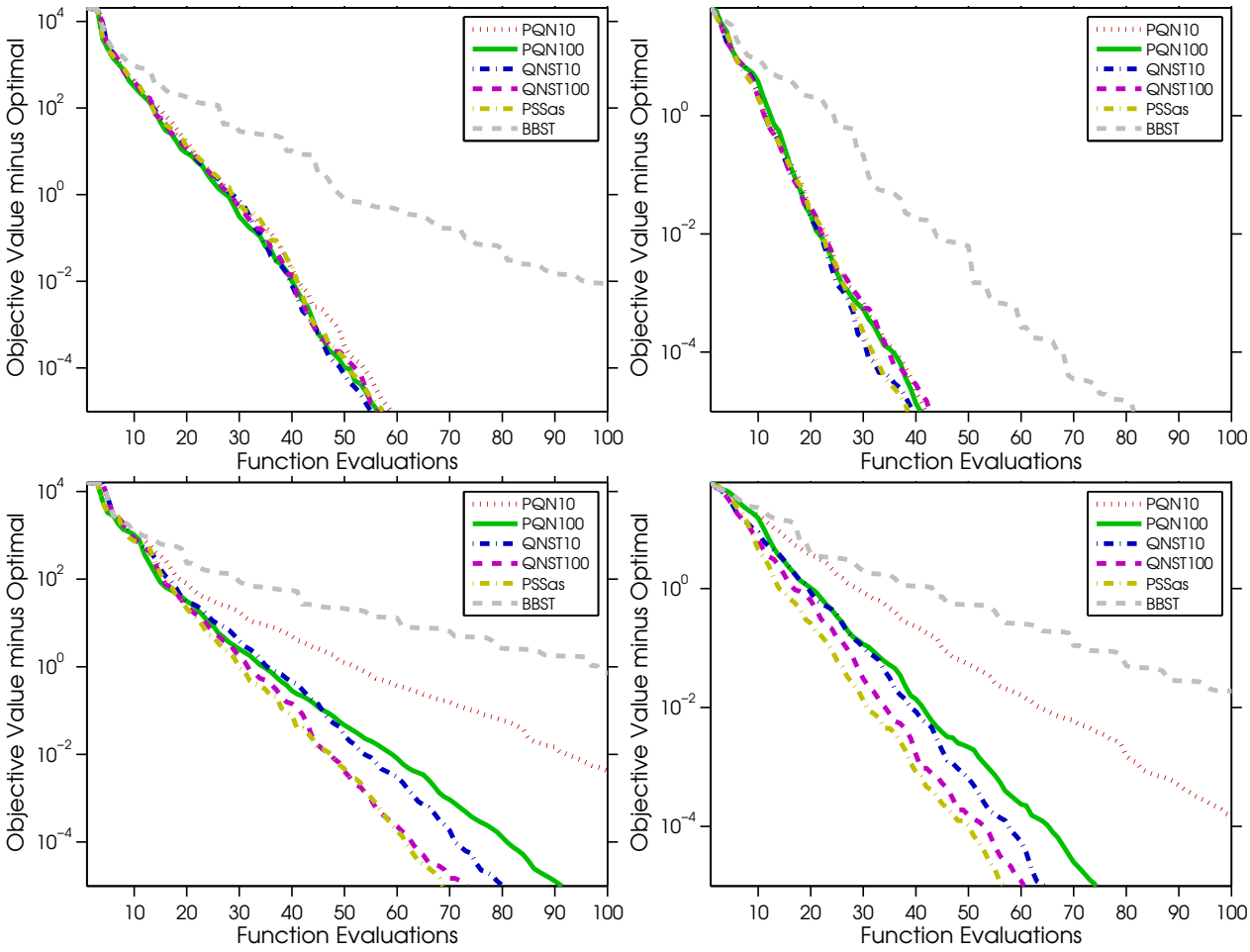

Figure 3.3: Function evaluations against objective value for training IGMs ($\lambda = 50$) with ℓ_1-regularization for different optimization strategies. Top row: *cyto* data. Bottom row: *awma* data. Left column: zero vector used for initialization. Right column: solution with $\lambda = 100$ used for initialization. This figure is best viewed in color.

- **Different choices of group norm**: In Chapter 5, we discuss computing the projection and soft-threshold operations for different choices of the group norm. This allows us to apply the methods in this chapter to different choices of the group norm. This includes the ℓ_∞ norm of the groups, and the nuclear norm of the groups in cases where the groups form matrices.

- **Overlapping groups**: In Chapter 6 we discuss computing the projection and soft-threshold operations when the groups overlap. That is, cases where each variable belongs in multiple groups. This allows us to apply the methods in this chapter to the case of general groups. Further, Jacob et al. [2009] describe an alternative generalization to the case of overlapping groups, and the methods in this chapter can be directly applied in this formulation.

Chapter 4

Directed Graphical Model Structure Learning

As we discuss in Chapter 1, the prior work on structure learning in probabilistic graphical models with ℓ_1-regularization largely focuses on pairwise undirected models. However, given the estimated model performing standard operations (ie. computing the probability of a vector, computing marginals, generating unbiased samples) with undirected models is computationally intractable in general. In contrast, as we discuss in Section 1.3 it is possible to perform many operations exactly or approximately in DAG models in polynomial time. In scenarios where the estimated model will ultimately be used to perform these types of operations, DAG models are an appealing alternative to undirected models. Further, parameter estimation in DAG models is separable in the CPDs and there is no need to compute an intractable normalizing constant. Since parameter estimation is separable, this allows us to independently tune an individual regularization parameter for each CPD (unlike undirected models where the same regularization parameter is typically used for all edges), to mix different types of data (ie. we can have both Gaussian and binary variables in the same data set), and allows us to more efficiently search the space of graphs since single edge modifications only change the parameters of a subset of the CPDs.

Given n realizations of p-vectors $\mathbf{x}^i$, the goal of structure learning in DAG models is to find a graph structure $\mathcal{G}$ (and corresponding parameters $\{\mathbf{w}_j, b_j\}$ for each node j) that optimize some criteria measuring the quality of the DAG model. In the special case where we are given a topological ordering of the nodes, this problem reduces to performing variable selection (among variables earlier in the ordering) independently for each of the CPDs [Buntine, 1991, Cooper and Herskovits, 1992][27]. For sigmoid belief networks, this corresponds to performing variable selection in a set of independent logistic regression models. Thus, we can learn DAG models with a known topological ordering using a straightforward extension of the methods we discuss in Chapter 1; we perform structure learning by using ℓ_1-regularization to solve each of these variable selection tasks. Other works using ℓ_1-regularization for structure learning in DAG models have focused on this relatively simple case [Li and Yang, 2005, Huang et al., 2006, Levina et al., 2008].

Even if we are not given a topological ordering, if we enforce that each node can have at most one parent then finding the optimal graph can be formulated and solved as a minimum spanning tree problem [Chow and Liu, 1968][28]. If we are not given a topological ordering and allow each node to have more than one parent, then finding the optimal DAG is NP-hard in most reasonable scenarios [Chickering, 1995, Dasgupta, 1999, Chickering et al., 2004]. Indeed, even if the graph structure is restricted to be a tree but each node is allowed to have at most $k \geq 2$ parents (also known as a poly-tree), it is NP-hard to even approximate the best graph structure to within a

[27] Assuming that the parameters of each CPD are independent.

[28] This relies on the scoring criteria satisfying the property of pairwise score equivalence, namely that the score of having x_i as the only parent of x_j is the same as the reverse. If the scoring criteria does not have this property, the optimal tree can be found in polynomial time by solving an optimal branching problem [Heckerman et al., 1995]

constant factor [Dasgupta, 1999]. Nevertheless, we can typically obtain a better model by not assuming a fixed ordering, and this general case is the focus of this chapter. The main challenge arising in the general case is the acyclicity constraint. Because the graph must be acyclic, we can not simply regress each node on all other nodes. Subsequently, we need to consider searching through the space of topological orderings, or directly searching through the space of directed acyclic graphs.

4.1 Search and Score Methods

Traditionally, there have been two different approaches to structure learning in general DAG models. In *search and score* methods, we use some criterion to assess the quality of a particular structure (such as the BIC or validation set likelihood), and we optimize this criterion by using a local search method to search through the space of DAGs [see Lam and Bacchus, 1993, Heckerman et al., 1995]. The BIC is widely used for evaluating the quality of a candidate structure. Early work that used the BIC includes [Lam and Bacchus, 1993, Bouckaert, 1993, Suzuki, 1999]. Under certain assumptions, the BIC gives the same score to Markov equivalent graphs [Bouckaert, 1993][29]. In [Friedman and Yakhini, 1996], asymptotic properties of the BIC for evaluating DAG structures are examined. They derive an asymptotic bound on the sample complexity of structure learning by optimizing the BIC score (in terms of Kullback-Leibler divergence), and show that in addition to asymptotic consistency that the BIC score also leads to asymptotic minimality (that is, it will choose the most sparse structure that describes the distribution). The most widely used alternative to the BIC for measuring structural quality are methods that compute the marginal likelihood of the CPDs (i.e. the likelihood after integrating over all possible parameters) given the graph structure under a suitable prior [Cooper and Herskovits, 1992, Heckerman et al., 1995]. This work will focus on the BIC, since except in special cases (such as Gaussian or tabular CPDs with conjugate priors), it is not possible to compute the marginal likelihood in closed form.

The prototypical search and score procedure is a greedy local search through the space of DAGs where at each iteration we perform the edge addition/deletion/reversal that improves the score by the largest amount, subject to satisfying the acyclicity constraint [Heckerman et al., 1995]. If no legal addition/deletion/reversal improves the score, the method can be reset to a different randomly generated DAG. We call this procedure *DAG-search*. The efficiency of this procedure is substantially improved if we (as in almost all related work on this subject) make the assumption of parameter modularity [Heckerman et al., 1995], meaning that if the same CPD appears in two graph structures, then the parameters of the CPD are the same in both structures (this assumptions follows as a consequence of assuming that the parameters of different CPDs are independent). Parameter modularity allows us to efficiently evaluate the effect of single edge additions/deletions/reversals. Further, we can use a hash data structure to prevent re-evaluating the same CPDs, while we note that the scores for most of the candidate additions/deletions/reversals will not change after a single addition/deletion/reversal.

There have been a variety of approaches proposed to enhance the basic DAG-search, and we briefly review a variety of these modifications here. Some authors have considered using different local search procedures, such as genetic algorithms [Larrafiag et al., 1996], ant colony optimization [de Campos et al., 2002a], and the greedy randomized adaptive search procedure [de Cam-

[29]These assumptions are not satisfied for sigmoid belief networks where nodes have more than one parent, since using sigmoid CPDs imposes additional structure on the model in these scenarios.

pos et al., 2002b]. Instead of searching through the space of DAGs, some authors have proposed searching through the space of topological orderings [Larrafiag et al., 1996, de Campos et al., 2002a, Teyssier and Koller, 2005] or Markov equivalent graphs [Spirtes and Meek, 1995, Madigan et al., 1996, Munteanu and Bendou, 2001, Chickering, 2003, Nielsen et al., 2003]. Steck [2000] proposes a local search move where all directions are removed and then re-oriented. Elidan et al. [2002] considers data re-weighting schemes that may allow DAG-search to escape local minima. Hulten and Domingos [2002] use Hoeffding's inequality for structure learning with tabular CPDs when the number of training examples is enormous. Moore and Wong [2003] propose a local search move where all edges connected to a node are severed, and the node is then optimally reinserted into the graph. Nachman et al. [2004] proposes a method for efficiently finding the best node to add for each variable with regression-based CPDs. Some authors have also considered exact methods that find the highest scoring structure, but that may require an exponential amount of time [Suzuki, 1999, Koivisto and Sood, 2004].

Despite the large number of more complicated methods that have been proposed in the literature, it has proven surprisingly difficult to devise a method that consistently outperforms an efficient DAG-search implementation, an issue discussed in [Teyssier and Koller, 2005]. Although search and score methods are often surprisingly effective, the main drawback of the search and score methodology is simply that the search space is very large; the space of DAGs is super-exponential in the number of nodes [Robinson, 1976].

4.2 Constraint-Based Methods

In contrast to search and score methods that try to directly optimize a criteria measuring the quality of the model, *constraint-based* methods seek to prune the set of possible edges. The original methods of this type are described in [Verma and Pearl, 1990, Spirtes and Glymour, 1991]. For each pair of variables, these methods search for a conditioning set that makes the pair satisfy a conditional independence hypothesis test (or makes their conditional dependence fall below a threshold [Cheng et al., 2002]). If we assume that the data is generated according to a DAG model and if a conditioning set is found that makes the pair of variables independent, then there can not exist an edge between the pair and the edge can be removed from consideration. After this edge pruning phase, further constraints may be used to determine the directionality of a subset of the remaining edges. Of particular interest to the present work is the observation of Verma and Pearl [1990] that the search space can be reduced to the *Markov blanket* of each node; the set of nodes that are conditionally dependent on the node given all other nodes, consisting graphically of the node's parents, children, and co-parents (other nodes that are parents of one of the node's children).

Unfortunately, the constraint-based approaches have several disadvantages. First, there are an exponential number of possible conditioning sets. Although implementations of constraint-based methods typically use heuristics that only consider a limited number of conditioning sets [Spirtes and Glymour, 1991], these methods must still perform a very large number of hypothesis tests. If corrections for multiple tests were incorporated into these methods, their statistical power would be very low. Indeed, it is not clear how to set the threshold value(s) in these tests such that the correct structure is identified asymptotically, as briefly discussed in [Heckerman et al., 1999]. Further, with finite data it is possible that an error in an independence test early in the procedure may lead to a propagation of errors. While constraint-based approaches typically output a valid

equivalence class, it is possible that they will output a cyclic graph, as illustrated in the pin-wheel example of [Dash and Druzdzel, 1999]. Some constraint-based methods have sought to address some of the common criticisms of constraint-based methods [Margaritis and Thrun, 1999], but a final noteworthy criticism is that it isn't clear how the results of these hypothesis tests relate to the quality of the model.

4.3 Hybrid Methods

The disadvantages of the constraint-based methods and the search and score methods have led to the development of hybrid methods. In hybrid methods, constraint-based reasoning is used to prune the set of edges to consider within a search and score method. This can lead to an enormous reduction in the number of possible graphs to search over. Much of the early work on methods of this type focused on forming constraints by eliciting domain knowledge from human experts. One example we have already seen is the case where the expert is asked to provide a topological ordering [Cooper and Herskovits, 1992]. Other examples include methods that attempt to construct an ordering given statements of domain knowledge [Srinivas et al., 1990], and methods that use partial orderings or require that some known edges are included in the model [Lam and Bacchus, 1993]. Unfortunately, these strategies crucially rely on the existence of a domain expert to provide the constraints.

One of the most popular methods to incorporate automatic pruning is the *sparse candidate* (SC) algorithm [Friedman et al., 1999]. In the SC algorithm, we compute a measure of dependence between each pair of variables (such as mutual information), and for a fixed k we only consider those k variables with the highest dependence as potential parents. Although this approach makes it feasible to learn DAG models with thousands of variables, this approach is somewhat problematic for the following reason: we can construct DAG distributions where no value of k less than $(p-1)$ will include all true parents among the k most dependent variables. For example, consider a chain-structured graph where x_1 is a parent of x_2, x_2 is a parent of x_3, x_3 is a parent of x_4, and so on up to x_n. If this structure is a parameterized so that the variables have a very high mutual information, and we add an extra node x_{n+1} that is a parent of x_n but with a low mutual information, then it might be the case that x_1 through x_{n-2} all have a higher mutual information with x_n than its true parent x_{n+1}. To address this problem, after using a search and score method in the reduced space, [Friedman et al., 1999] suggest re-computing the set of candidate parents (this time using a conditional measure of dependence) when a local minimum is reached.

We might hope to avoid the unsound pruning caused by the sparse candidate algorithm by accepting all parents whose pairwise mutual information is above a threshold. Unfortunately, this is not a particularly effective pruning strategy, since even if the underlying graph is sparse the variables may not be marginally independent. As an example, consider a simple chain-structured model with Gaussian CPDs. The precision matrix in this model is tri-diagonal, corresponding to a very sparse graph. However, the inverse of a tri-diagonal matrix will (in general) be completely dense; all variables are marginally dependent so no interactions are pruned.

Because tests of marginal (in)dependence are not particularly effective at pruning the set of possible edges, more recent hybrid approaches have considered directly applying constraint-based structure learning methods to prune the set of edges [Dash and Druzdzel, 1999, Li and Yang, 2004, Tsamardinos et al., 2006]; these algorithms rely on *conditional* independence tests rather than marginal independence tests. These methods typically lead to a substantial reduction in the search

space, and one of the empirically most effective current methods for learning DAG structures, the max-min hill-climbing (MMHC) algorithm, is of this type [Tsamardinos et al., 2006]. Further, if we assume that a perfect conditional independence oracle is available, and that all conditional independencies in the distribution follow from the graph structure, then reducing the search space by pruning edges between conditionally independent nodes is a sound pruning strategy (it will never remove a true dependency from the model).

An alternative strategy to pruning the space of DAGs is to use conditional independence tests to obtain a variable ordering, and then apply a variable selection method assuming that the ordering represents a topological ordering [Singh and Valtorta, 1993, Acid et al., 2001, Dobra et al., 2004]. The disadvantage of this type of approach is simply that it may be very difficult to find a correct topological ordering. Of particular note is the method of [Dobra et al., 2004], where the authors initially learn a dependency network on the variables (to approximate each node's Markov blanket), and use this to construct an ordering.

4.4 A Hybrid Method with ℓ_1-regularization

The recent hybrid methods are appealing compared to strict constraint-based methods, because the second phase (search and score) of the methods attempts to optimize a score measuring the quality of the structure. Further, they can be advantageous over strict score-based methods, due to the much smaller search space. However, the hypothesis tests (or pairwise dependency measures) used by existing hybrid methods ignore the score during the first phase (edge pruning). As an example where this might be problematic, consider the case where two variables are weakly dependent and we want to find a structure optimizing the BIC score. Here, the edge may pass the independence test and not be pruned during the first phase, even though including this edge is unlikely to improve the BIC score. Similarly, an independence test might prune an edge between variables that appear to be almost independent (recall that these hypothesis tests are not corrected for multiple testing), even though including the edge would later lead to an improved validation score.

Towards developing a hybrid method that takes into account our scoring criteria during both phases, we propose the following two-phase hybrid method for a given scoring criteria:

1. **Edge pruning**: We use ℓ_1-regularization to learn a dependency network with logistic regression conditionals, *that optimizes the proposed scoring criteria*. We refer to this as the ℓ_1-Markov blanket (L1MB) algorithm, and we note that this problem can be solved even with a very large number of nodes using the methods of Chapter 2.

2. **Search**: We run a DAG-search algorithm to search through the space of possible DAG structures, restricted to the edges found by the L1MB algorithm.

Below we give pseudo-code for the L1MB algorithm. In our implementation, for a network with p nodes we compute the ℓ_1-regularized solution (and corresponding score) for $(p-1)$ equally spaced values along the regularization path between λ set to zero and λ_{max}, where λ_{max} is the value where

all (non-bias) variables become zero (see Section 2.5).

Input: Data x_j^i for $i = 1, \ldots, n$ and $j = 1, \ldots, p$.
Output: Markov blanket MB_j for each node j.
for $j = 1$ *to* p **do**
  $MB_j \leftarrow \emptyset$; // initially try using empty Markov blanket
  $s \leftarrow \text{score}(\mathbf{x}_j, \mathbf{x}_\emptyset)$; // compute score with empty Markov blanket
  $b \leftarrow \min_b \sum_{i=1}^n \log(1 + \exp(x_j^i b))$; // optimize for bias variable
  $\mathbf{g} \leftarrow \sum_{i=1}^n x_j^i \mathbf{x}_{-j}^i / (1 + \exp(x_j^i b))$; // gradient of regression weights at zero
  $\lambda_{max} \leftarrow \max_i\{\mathbf{g}_i\}$; // maximum value of regularization parameter
  for $\lambda = ((p-1)/p)\lambda_{max}$ *down to* 0 *in increments of* λ_{max}/p **do**
    $\{\mathbf{w}, b\} \leftarrow \arg\min_{\mathbf{w},b} \sum_{i=1}^n \log(1 + \exp(-x_j^i(\mathbf{w}^T \mathbf{x}_{-j}^i + b))) + \lambda ||\mathbf{w}||_1$; // Chapter 2
    $nz = \{v | w_v \neq 0\}$; // find non-zero variables
    $s_{sub} = \text{score}(\mathbf{x}_j, \mathbf{x}_{nz})$; // compute score with selected Markov blanket
    if $s_{sub} > s$ **then**
      $s \leftarrow s_{sub}$; // new maximum value found
      $MB_j \leftarrow nz$; // record higher-scoring Markov blanket

Algorithm 10: L1MB Algorithm.

If we assume that the true structure is a DAG and that ℓ_1-regularization is able to perfectly select the relevant variables, then the L1MB algorithm will identify each variable's Markov blanket. For the second phase, we use an implementation of the DAG-search method, where at each iteration we choose the variable addition/deletion/reversal (among edges found by the L1MB algorithm) that improves the score by the largest amount. To address the criticism that DAG-search requires costly acyclicity checks [Teyssier and Koller, 2005], we used the *ancestor matrix* data structure described in [Giudici and Castelo, 2003] for improving the speed of Markov chain Monte Carlo methods that explore the space of DAGs. With this data structure, it is possible to check whether an addition will cause a cycle in $\mathcal{O}(1)$, while testing whether a reversal of an existing edge leads to a cycle can be done in $\mathcal{O}(p)$. In Appendix A, we review this data structure and present several enhancements to it, including a *reversal witness matrix* data structure that allows us to test whether reversing an edge will cause a cycle in $\mathcal{O}(1)$. In [Schmidt et al., 2007b], we also examined a variant of the method where we used the known-ordering ℓ_1-regularization method to find the optimal structure given an ordering, and we used the local swap moves described in [Teyssier and Koller, 2005] for searching the space of orderings. Although this gives a somewhat more elegant procedure, we found that this was not as effective as searching through the space of DAGs when the ancestor matrix data structure is used for testing acyclicity[30].

The hybrid ℓ_1-regularization method we discuss here is closely related to the work described in [Li and Yang, 2004][31]. Li and Yang [2004] also first learn a dependency network using ℓ_1-regularization. This is followed by running a constraint-based method to further prune the edges and fix the directionality of some edges, and the final step runs a DAG-search to optimize a scoring criteria. Besides removing the (potentially error-prone) second phase (and the focus on sigmoid

[30] Note that we are using sigmoid CPDs with no bound on the in-degree of nodes in the graph, while Teyssier and Koller [2005] used tabular CPDs with a bounded in-degree. This allowed them to pre-compute all possible scores, while in our work computing all possible scores is intractable so we compute the scores as needed. This requires solving a logistic regression problem to test any changed edges.

[31] At the time that [Schmidt et al., 2007b] was published, we were not aware of this work (nor were our reviewers)

CPDs instead of Gaussian CPDs), the crucial difference between our method and this previous work is that we use the scoring criteria when constructing the dependency network, while [Li and Yang, 2004] use hypothesis testing. As we discuss above, it is not necessarily clear how the results of the hypothesis tests relate to the score that is optimized in the final stage.

4.5 Causal DAGs

Unfortunately, without being given a topological ordering it will only be possible to identify the optimal DAG structure up to Markov equivalence. That is, it may not be possible to identify the directionality of some of the edges. This may not be a bad thing if our goal is to build a density model, since it might imply that multiple DAGs will achieve the globally optimal score (and we only need to find one of them). However, this property is less appealing from the perspective of structural discovery, since if we believe our data is generated from a DAG model, it means that we may not be able to distinguish the 'true' structure from other candidates. A notable special case where we can hope to identify the true structure without a topological ordering is the case of causal DAGs, where the data includes interventions.

A causal DAG model [Pearl, 2000] is a DAG model where we assume that the directions of the edges represent causal influences (the *causal Markov* assumption). Under this assumption, we distinguish between conditioning *by observation* and conditioning *by intervention*. When conditioning on a variable by observation, we use the standard rules of conditional probability to answer conditional queries. When conditioning on a variable by intervention, we create a modified DAG model, and then use the standard rules of conditional probability to answer conditional queries using the modified model. Specifically, when variable j is set by intervention (denoted $do(j)$), we use a modified DAG where the CPD for variable j has been removed. In other words, the interventional distribution uses

$$p(x_1, \ldots, x_p | do(j)) = \prod_{i \neq j} p(x_i | \mathbf{x}_{\pi(i)}).$$

Graphically, the effect of removing this CPD is to remove all incoming edges into j, while preserving outgoing edges. Thus setting j by intervention makes it independent of its causes, but preserves the dependency on its effects. Because of this asymmetry between cause and effect, it is possible to distinguish between Markov equivalent graphs in causal DAGs, given data that includes interventions.

Utilizing interventional data within structure learning was first explored in [Cooper and Yoo, 1999], and extending the hybrid method based on ℓ_1-regularization above to model interventional data with causal DAGs is straightforward. When estimating the conditional of node j during parameter estimation (during either the L1MB or DAG-search phase), we use the modified distribution in cases where j was set by intervention. Similarly, we remove the CPD for node j when evaluating the BIC or validation likelihood. All other aspects of the method remain the same.

4.6 Experiments

We now experimentally examine the performance of various methods for learning sparse DAG models. We first did a series of experiments on synthetic data where the structure was known, detailed in the next section. After these experiments, we apply the methods to learn sparse DAG models of real data in §4.6.2

4.6.1 Synthetic Data

We first considered a set of synthetic data sets, where we generated samples from a known structure and then tried to recover the structure from the samples. In particular, we obtained seven graph structures from the Bayesian Network Repository,
`http://compbio.cs.huji.ac.il/Repository/`.

In the table below, we give the names, number of nodes, number of edges, and maximum number of parents for each of the seven networks we considered.

Name	Nodes	Edges	Max Parents
insurance	27	52	3
water	32	66	5
mildew	32	46	3
alarm	37	46	3
barley	48	84	4
hailfinder	56	66	4
carpo	61	74	5

To parameterize the networks as a sigmoid belief network with strong edge weights, we set the bias for each node to zero and each edge weight was set according to the formula

$$w_{ij} \leftarrow \text{sign}(\mathcal{N}(0,1)) + \mathcal{N}(0,1)/4,$$

where $\mathcal{N}(0,1)$ is a sample from a standard normal distribution. In our experiments, we used the BIC as the scoring criterion.

In our first experiment, we compared the performance of several different possible strategies for pruning the set of edges:

- **SC**: The set of parents selected on the first iteration of the sparse candidate method [Friedman et al., 1999], where we used pairwise mutual information to rank the candidates. We tested the method with two parameters, the true maximum in-degree across the networks (5) and double this amount (10).

- **MMPC**: The set of parents remaining after the max-min parents and children pruning procedure [Tsamardinos et al., 2006], where conditional hypothesis tests are used to prune the set of parents. The experiments in Tsamardinos et al. [2006] indicate that this constraint-based procedure leads to state-of-the-art results against a wide variety of alternative methods. We used the implementation in the author's Causal Explorer software [Aliferis et al., 2003]. We tested the method with two values of the hypothesis test threshold, the software default of 0.05 and a more conservative value of 0.10.

- **L1MB**: The proposed procedure for finding the Markov blanket of each node using ℓ_1-regularization, where the hyper-parameter is selected to optimize the scoring criterion. Other than the selection of points to evaluate along the regularization path, this algorithm has no parameters.

An ideal pruning procedure would remove as many edges as possible, while minimizing the number of true edges that are removed. In Figure 4.1, we plot the the percent of edges remaining (top) and the number of true edges removed (bottom) for all of the methods on the seven data sets

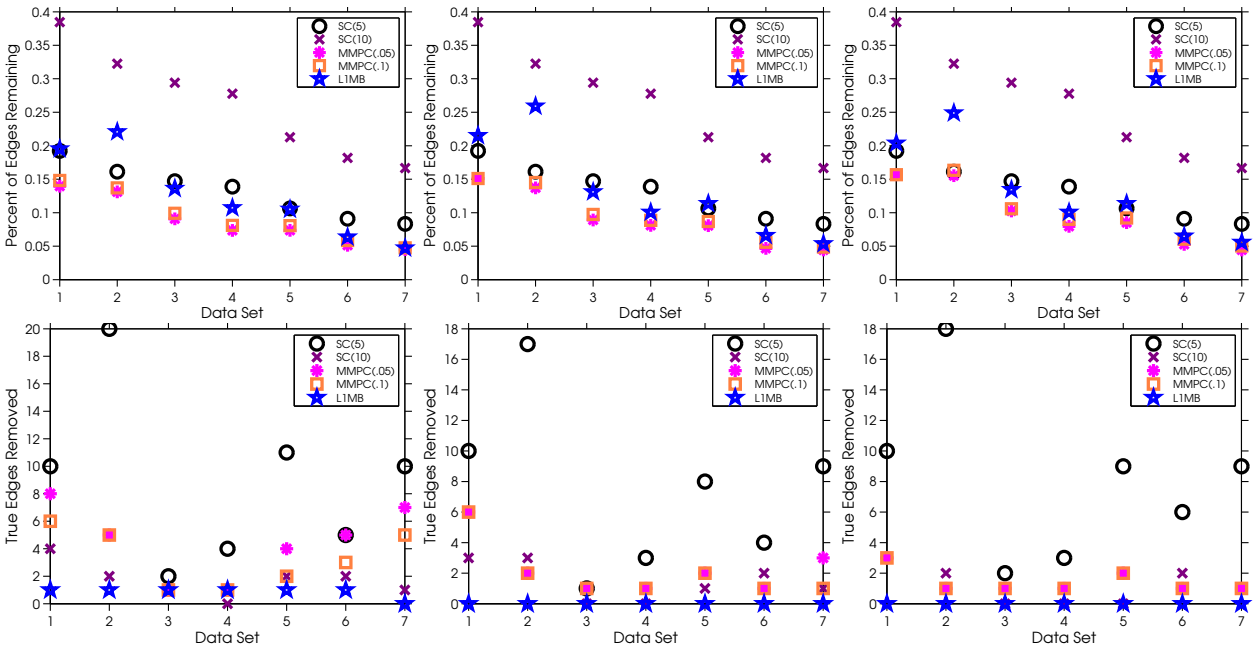

Figure 4.1: The percent of edges remaining (top) and number of true edes removed (bottom) for different edge pruning strategies for seven structures from the Bayesian network repository. From left to right, the plots show the results with samples sizes of 1000, 5000, and 20000. We see that the L1MB pruning method leads to a reasonable amount of pruning while tending not to remove true edges.

for three different sample sizes (1000, 5000, and 20000). In this figure, we see that the SC method has a fairly sharp trade-off between the two objectives: SC(5) removes a large number of edges but removes many true edges, while SC(10) removes fewer true edges but does not prune much of the search space. The MMPC method is more effective, it reduces the search space substantially and does not remove many true edges, decreasing the number of true edges that are removed as the sample size increases. The L1MB method has similar behaviour; L1MB does not prune quite as much as the MMPC method but removes fewer true edges. Indeed, the L1MB method removed no true edges in any data set for any of the experiments with 5000 or 20000 samples (and it never removed more than one true edge), while the other methods removed multiple true edges in almost every case.

In our next experiment, we sought to assess the effectiveness of a DAG-search routine under these different pruning strategies. We compared the five methods examined in the previous experiment, as well applying the DAG-search with no pruning. To test the different pruning strategies, we started the DAG-search from the empty graph and ran it until it had made 10000 score evaluations. If a local minimum was found before this limit, the methods were restarted to a randomly generated DAG (we generated the DAGs by generating a random topological ordering, and adding each edge consistent with the pruning and the ordering with probability 0.5). The same random DAGs were used across the methods. We restarted the hash of score values after each local minimum was found, but better performance would be achieved by keeping the same hash table between runs. We plot the BIC after 10000 evaluations against the data sets for the different methods in Figure 4.2. Since

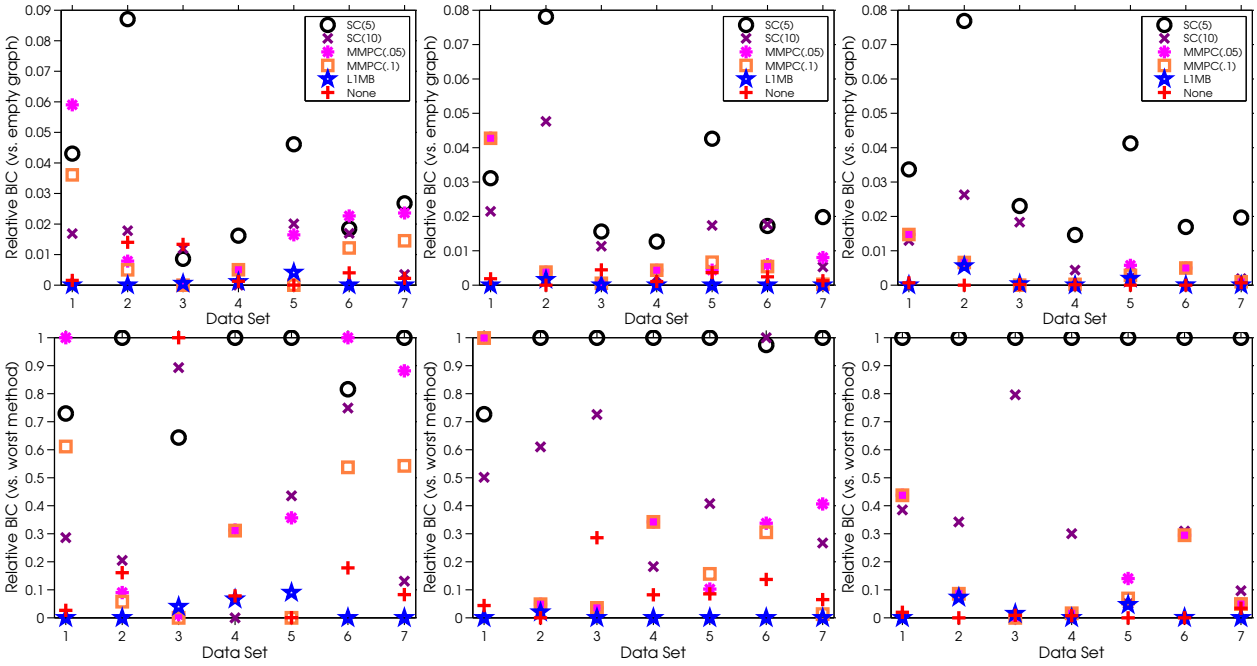

Figure 4.2: The relative BIC after 10000 score evaluations in a DAG-search for different pruning strategies on the seven synthetic data sets from the Bayesian Network Structure Learning Repository. We the BIC relative to the empty graph (top) and relative to the highest score for each data set (bottom). From left to right, the plots show the results with samples sizes of 1000, 5000, and 20000. We see that the L1MB pruning consistently achieves among the lowest scores.

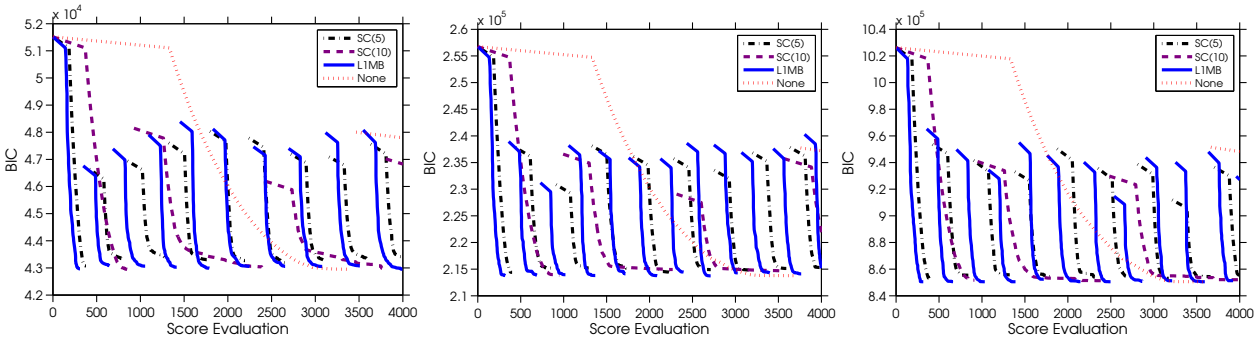

Figure 4.3: The BIC against the number of score evaluations in a DAG-search for different pruning strategies with 1000 (left), 5000 (middle), and 20000 (right) samples from the *alarm* data set. We see that no pruning eventually leads to a good score, that the pruning strategies allow the method to explore multiple local optima, and that the L1MB algorithm achieves both of these properties.

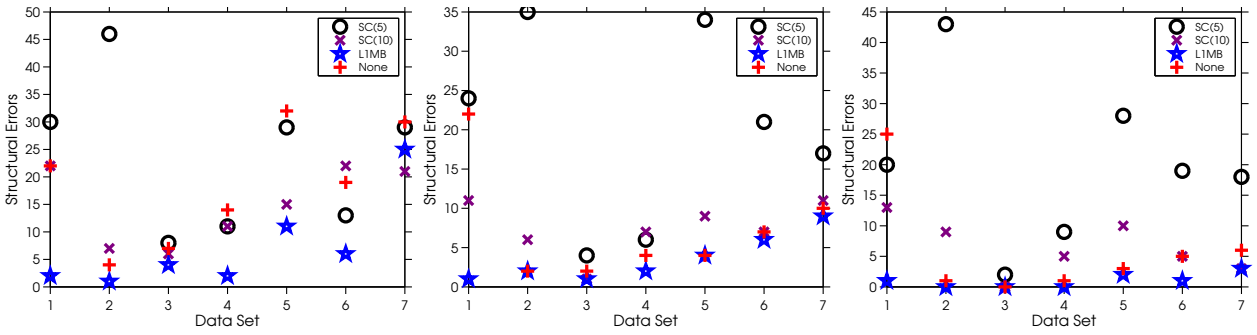

Figure 4.4: Structural errors for the highest scoring structure after 10000 score evaluations in an interventional DAG-search for different pruning strategies on the seven synthetic data sets from the Bayesian Network Structure Learning Repository. From left to right, the plots show the results with samples sizes of 1000, 5000, and 20000. We see that the L1MB pruning leads to the fewest structural errors in almost every case.

the absolute BIC varies across data sets and sample sizes, these figures plot a relative BIC. In the top of Figure 4.2, we computed the relative BIC by scaling the scores to values between the lowest BIC found across the methods, and the BIC of the empty graph. Under this criteria, the empty graph would have a relative BIC of 1, and the best graph found across the methods gets a value of 0. In the bottom of Figure 4.2, we plot the score relative to the pruning strategy that had the highest BIC over each data set. In this figure, we see that the L1MB pruning consistently lead to low BIC across the sample sizes, while the SC methods were less effective and the MMPC methods were in between. Interestingly, not using any pruning seemed to be more effective as the sample size increased. This might be because the BIC favours the true model more heavily as the sample size increases.

To gain more insight into the performance disparities between different pruning methods, in Figure 4.3 we plot the BIC of the current structure against the score evaluation for the different pruning methods for the three sample sizes on the *alarm* data set (we omit the MMPC methods for clarity, but note that these methods resemble the SC and L1MB methods). Here, we see that the None method takes substantially longer to reach a local minimum than the other methods, but eventually reaches a good local minimum. In contrast, the pruning methods reach local minima very quickly, and this allows them to explore multiple modes. However, because the SC and MMPC methods tend to remove true edges from the model, the minima they found tend to be poorer than those found by the None and L1MB method.

In general, without a topological ordering we can only expect to recover the true structure up to its Markov equivalence class. This is the reason we used the BIC as a measure of performance in the previous experiment. In our final experiment on synthetic data, we generated interventional data to test the ability of the different pruning strategies to recover the true structure. To generate interventional data, for each sample we generated a random integer between 0 and p, and intervened on the corresponding node by setting it to 1 with probability 0.5 (when 0 was drawn, we did not intervene on any nodes and generated a purely observational sample). We plot the number of structural errors in the structure with the lowest BIC after 10000 evaluations for the different pruning strategies on the different data sets and sample sizes in Figure 4.4. We did not run the MMPC method on this data set, since that software does not support interventions. In this figure,

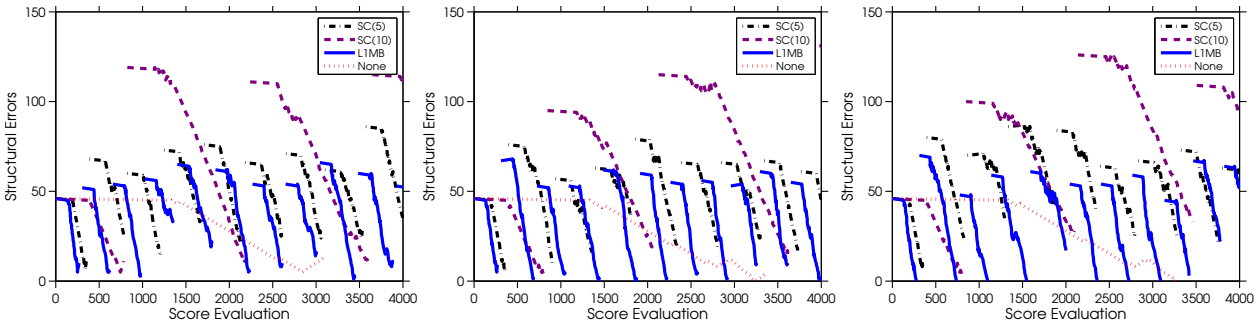

Figure 4.5: The structural errors against the number of score evaluations in an interventional DAG-search for different pruning strategies with 1000 (left), 5000 (middle), and 20000 (right) samples from the *alarm* data set.

we see that the L1MB method consistently achieves among the lowest structural errors, making fewer errors as the sample size increases.

In Figure 4.5, we plot the number of structural errors achieved by the current structure against the number of score evaluations on the *alarm* data set for the different sample sizes. An interesting aspect of these plots is that for small sample sizes the number of structural errors does not decrease monotonically with the BIC. As a consequence, we see that with 5000 samples that the None method finds the true structure, but it does not choose this structure since it finds a different structure with a lower score. With 5000 samples the L1MB method also finds the true structure (three times) during its search. The L1MB method also finds nine local optima with a single structural error. That is, these methods are one edge away from the true structure, but the modification can not be made without violating acyclicity. With 20000 samples both the None and L1MB pruning methods find the true structure, but the L1MB method finds it seven times before the None method finds it.

4.6.2 Real Data

Because the true structure is generally unknown in real data, it is generally not possible to evaluate a structure learning method in terms of structural errors. However, we might still be interested in testing whether a method recovers a plausible structure. Thus, we first sought to test the method on a real data set where we had a reasonable guess of both a topological ordering of the variables and the structure of the model. Towards this end, we focused on the *rain* data we describe in Section 1.7. For this data, we assumed that using the days of the month in order would represent a reasonable topological ordering. Further, we might expect to learn a structure that connects adjacent days, under the intuition that if it rains on one day it is likely to also rain the next day. This would lead to a 28-node Markov chain structure. We might also expect to see connections between non-adjacent but close days, although connections between distant days seem less likely.

In Figure 4.6, we plot the structure given by four different structure learning methods: (i) finding the optimal tree structure subject to the topological ordering, (ii) exhaustive enumeration to find the structure with highest BIC that is consistent with the ordering and the SC(5) pruning, (iii) greedily selecting parents starting from the empty graph (this is similar to the K2 algorithm [Cooper and Herskovits, 1992]), and (iv) using the L1MB method constrained to be consistent with the ordering (in this case no search is necessary). In this plot we see that the optimal tree for this data set is a

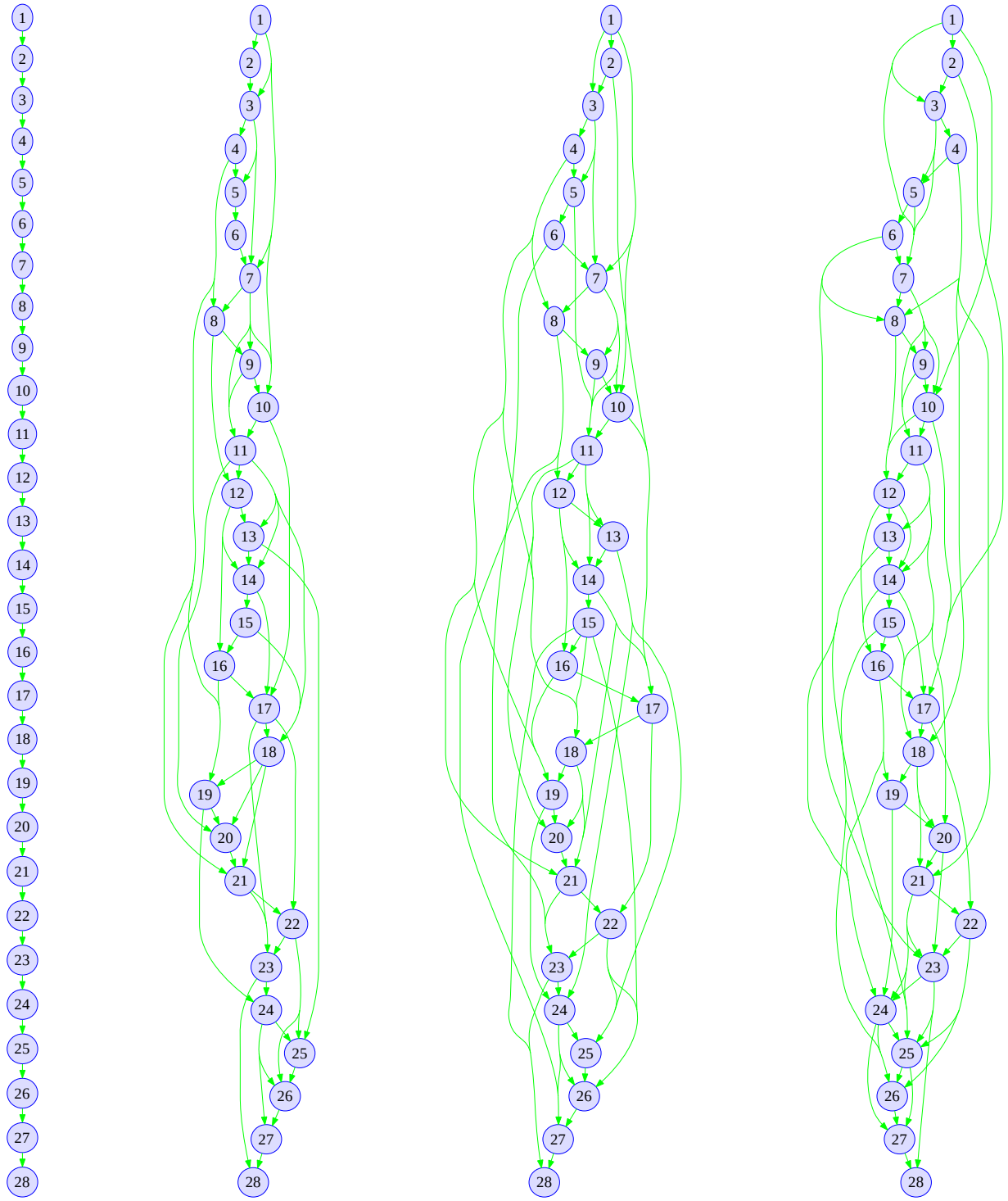

Figure 4.6: Structures estimated on the *rain* data set under a topological ordering. From left to right: optimal tree-structure consistent with ordering, optimal parents consistent with the ordering and SC(5) pruning, greedy parent selection given the ordering, and the L1MB algorithm constrained to be consistent with the ordering.

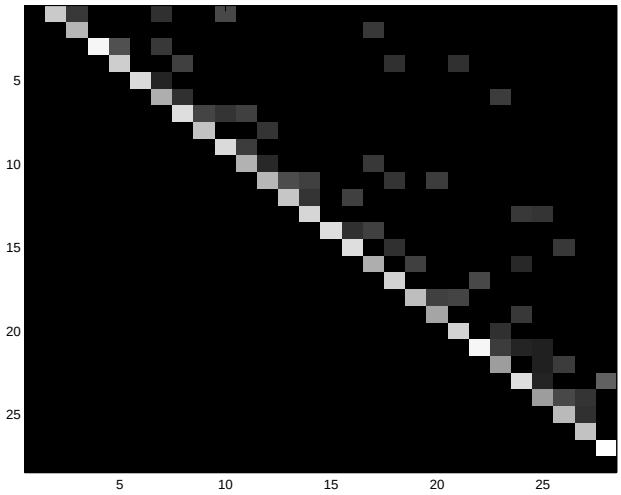

Figure 4.7: The regression weights for the *rain* data set using the L1MB algorithm for a topological ordering. We see that weights between adjacent days (first diagonal above the main diagonal) are much larger than the other weights.

28-node Markov chain, as expected. In contrast, the structures learned by the other methods are much less interpretable, including not only edges between adjacent days but also edges between more distant days. The structures learned by exhaustive enumeration after using using the other pruning strategies from Figure 4.1 (namely, the SC(10), MMPC(0.05), and MMPC(0.1) methods) were qualitatively similar to these latter structures, in that they included all edges between adjacent days but also included edges between temporally close nodes as well as nodes that are not temporally close. This would seem to indicate that these methods are not ideal for structural discovery (or that the BIC is not appropriate for judging the quality of the structure). However, we gain additional insight if we look at the regression weights.

In Figure 4.7 we plot as a matrix the absolute value of the (non-bias) regression weights of the L1MB method. In this plot, we see that the first diagonal above the main diagonal contains substantially larger weights than the rest of the matrix. The elements on this diagonal represent the effect of the previous day on each day. We also see some much weaker weights on the next two upper diagonals and spread out throughout the rest of the upper triangle of the matrix (the main diagonal is zero since it does not correspond to a parameter, while the lower triangle part of the matrix is zero because we assumed that the order represents a topological ordering). With logistic regression CPDs over binary parents encoded as $\{-1, 1\}$ binary variables, we can interpret the regression weights in terms of the odds of the child taking the same value as its parent. For example, a regression weight of 0.5 means that the logarithm of the odds of a child taking the same value as its parent is increased by 0.5 (over its bias value). By looking at the regression weights, we see that the 28-node Markov chain has the strongest influence on the model and that it is recovered if we only concentrate on the largest regression weights (this isn't unique to the L1MB method, the same is true of the other pruning methods, too). Thus, the unexpected extra edges present in the L1MB graph structure represent weaker statistical dependencies. These might be spurious correlations detected by the method that happen to improve the BIC, or they might reflect that the data is not perfectly modeled by a sigmoid belief network.

In general, we may not have a topological ordering, so our next experiment compared the

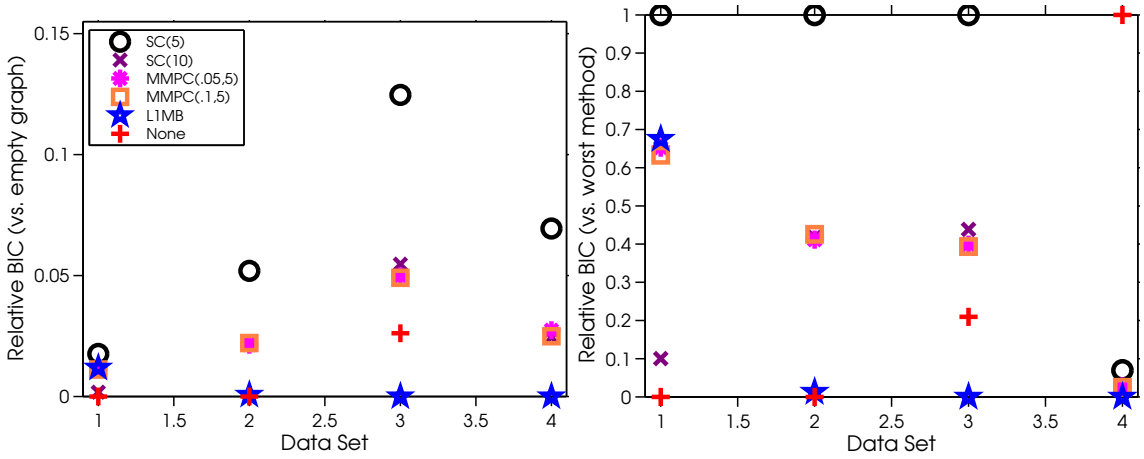

Figure 4.8: The relative BIC compared to the empty graph (left) and method with highest BIC (right) after 50000 score evaluations in a DAG-search for different pruning strategies on the real data sets. The data are ordered by node size: (1) *rain* (28 nodes), (2) *msweb* (57 nodes), (3) *news* (100 nodes), and (4) *usps* (256 nodes). Note that the None method has a relative BIC of 0 on the *usps* data set in the left figure.

various DAG-search pruning strategies on the four larger binary data sets we discuss in Section 1.7. For most of these larger data sets the MMPC pruning did not finish after a week of computation, so we decreased the maximum cardinality of the conditioning sets to 5 for the MMPC pruning methods. We plot the relative BIC for the different data sets after 50000 score evaluations in Figure 4.8. In these plots, we see that the basic DAG-search (None) is effective on the two smaller data sets but its performance decreases substantially on the two data sets with the larger number of nodes (on the *usps* data set, the evaluation limit is exceeded before all neighboring graphs can be considered). In contrast, the L1MB method was more effective than the other methods on the two higher-dimensional data sets.

We now look at one of the learned structures in more detail, focusing on the *news* data. The 100 words measured in this data set are

> aids, baseball, bible, bmw, cancer, car, card, case, children, christian, computer, course, data, dealer, disease, disk, display, doctor, dos, drive, driver, earth, email, engine, evidence, fact, fans, files, food, format, ftp, games, god, government, graphics, gun, health, help, hit, hockey, honda, human, image, insurance, israel, jesus, jews, launch, law, league, lunar, mac, mars, medicine, memory, mission, moon, msg, nasa, nhl, number, oil, orbit, patients, pc, phone, players, power, president, problem, program, puck, question, religion, research, rights, satellite, science, scsi, season, server, shuttle, software, solar, space, state, studies, system, team, technology, university, version, video, vitamin, war, water, win, windows, won, and world.

The Markov blankets estimated by L1MB for the first ten words are

- **aids**: children, disease, evidence, fact, food, health, president, program, research
- **baseball**: case, christian, computer, drive, email, fact, fans, games, god, government, help, hit, league, memory, nhl, players, power, puck, question, season, software, state, system,

team, win, windows

- **bible**: car, card, christian, course, earth, fact, god, jesus, orbit, program, question, religion, version, windows, world
- **bmw**: car, christian, engine, god, government, help, university, windows
- **cancer**: disease, health, medicine, patients, research, studies

Many of the words present in these estimated Markov blankets represent fairly natural associations (aids:disease, baseball:fans, bible:god, bmw:car, cancer:patients, etc.). However, some of the estimated statistical dependencies seem less intuitive, such as baseball:windows and bmw:christian. As before, we gain more insight if we look at not only the sparsity pattern but also the regression weights. Below we repeat the list along with the values of the corresponding regression weights:

- **aids**: children (0.53), disease (0.84), fact (0.47), health (0.77), president (0.50), research (0.53)
- **baseball**: *christian* (-0.98), *drive* (-0.49), games (0.81), *god* (-0.46), *government* (-0.69), hit (0.62), *memory* (-1.29), players (1.16), season (0.31), *software* (-0.68), *windows* (-1.45)
- **bible**: *car* (-0.72), *card* (-0.88), christian (0.49), fact (0.21), god (1.01), jesus (0.68), orbit (0.83), *program* (-0.56), religion (0.24), version (0.49)
- **bmw**: car (0.60), *christian* (-11.54), engine (0.69), *god* (-0.74), *government* (-1.01), *help* (-0.50), *windows* (-1.43)
- **cancer**: disease (0.62), medicine (0.58), patients (0.90), research (0.49), studies (0.70)

Here, we see that some of the less intuitive statistical dependencies have negative regression weights (italicized), indicating that they represent a dissociative relationship (i.e. the model reflects that baseball:windows is an unlikely combination). Closer investigation reveals that these dissociative relationships heavily influence the model. For example, if we examine all the regression weights, the strongest dissociative relationship is government:nhl (with a weight of -13.31), while the strongest associative relationship is food:msg (with a weight of 2.52). Further, there are 1173 negative regression weights, and only 286 positive regression weights (while 8541 are zero).

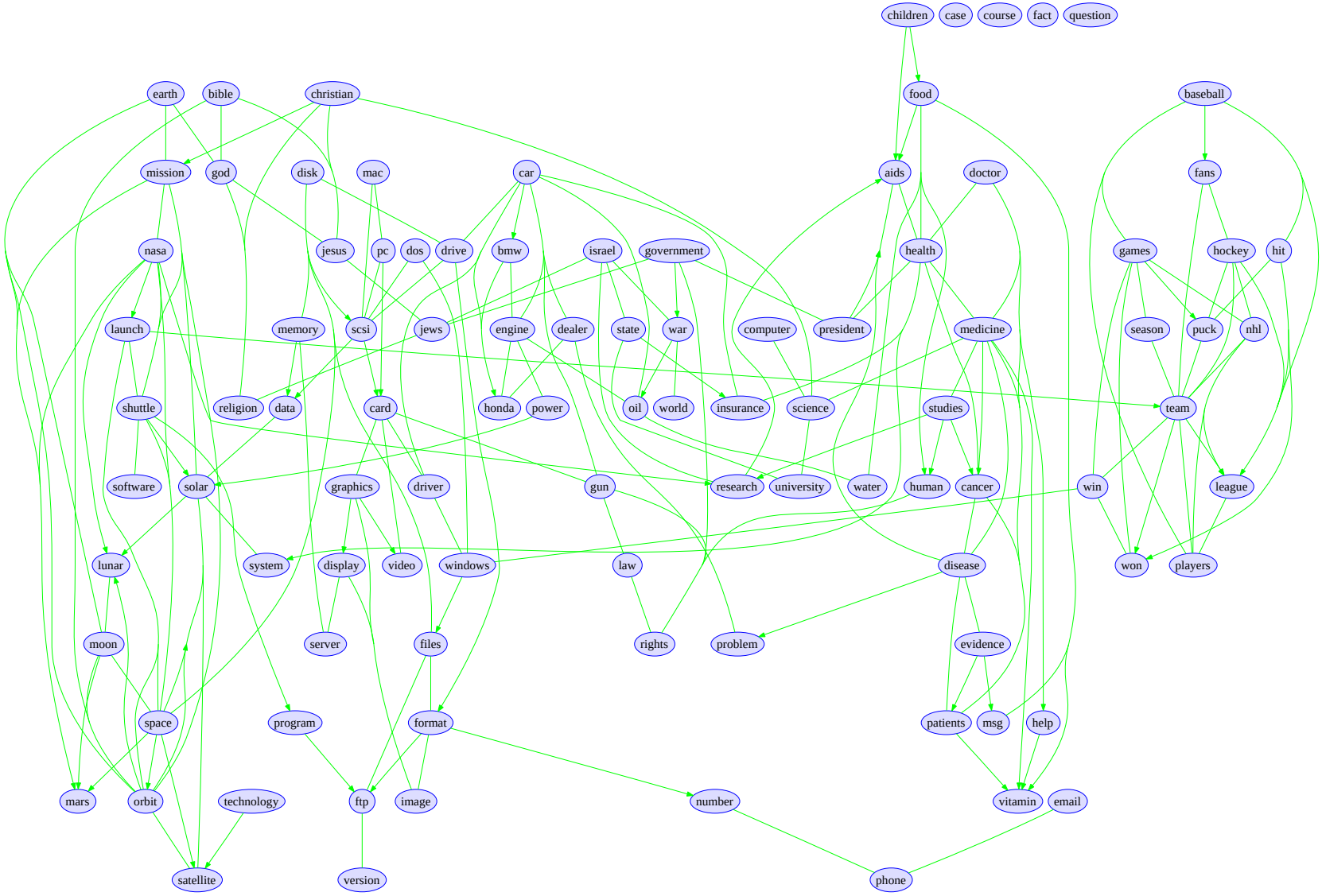

Figure 4.9: All edges with regression weight above 0.5 in the Markov blankets estimated by L1MB on the *news* data. Undirected edges represent cases where the directed edge was found in both directions.

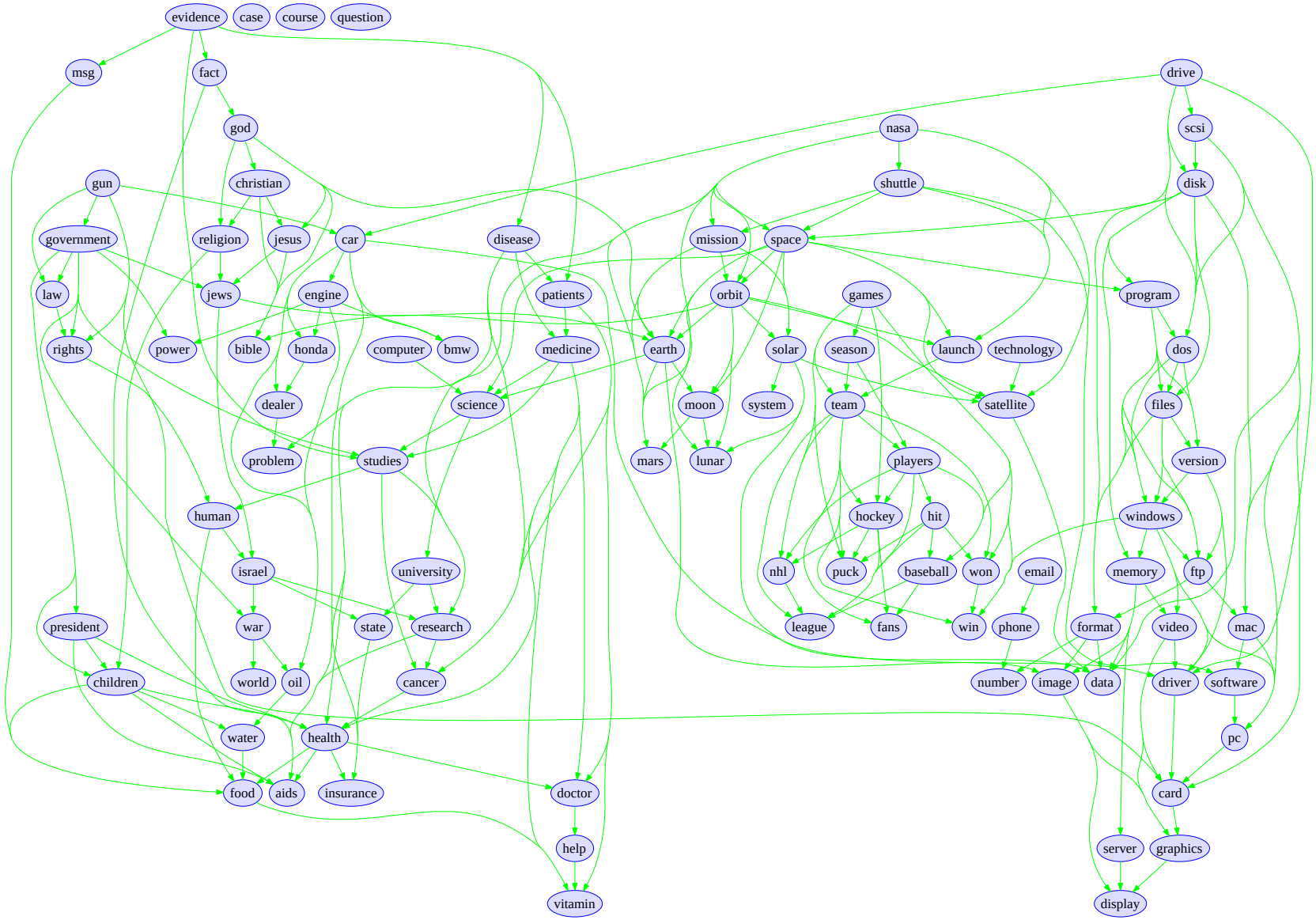

Figure 4.10: All edges with regression weight above 0.5 in the model found by DAG-search with L1MB pruning on the *news* data.

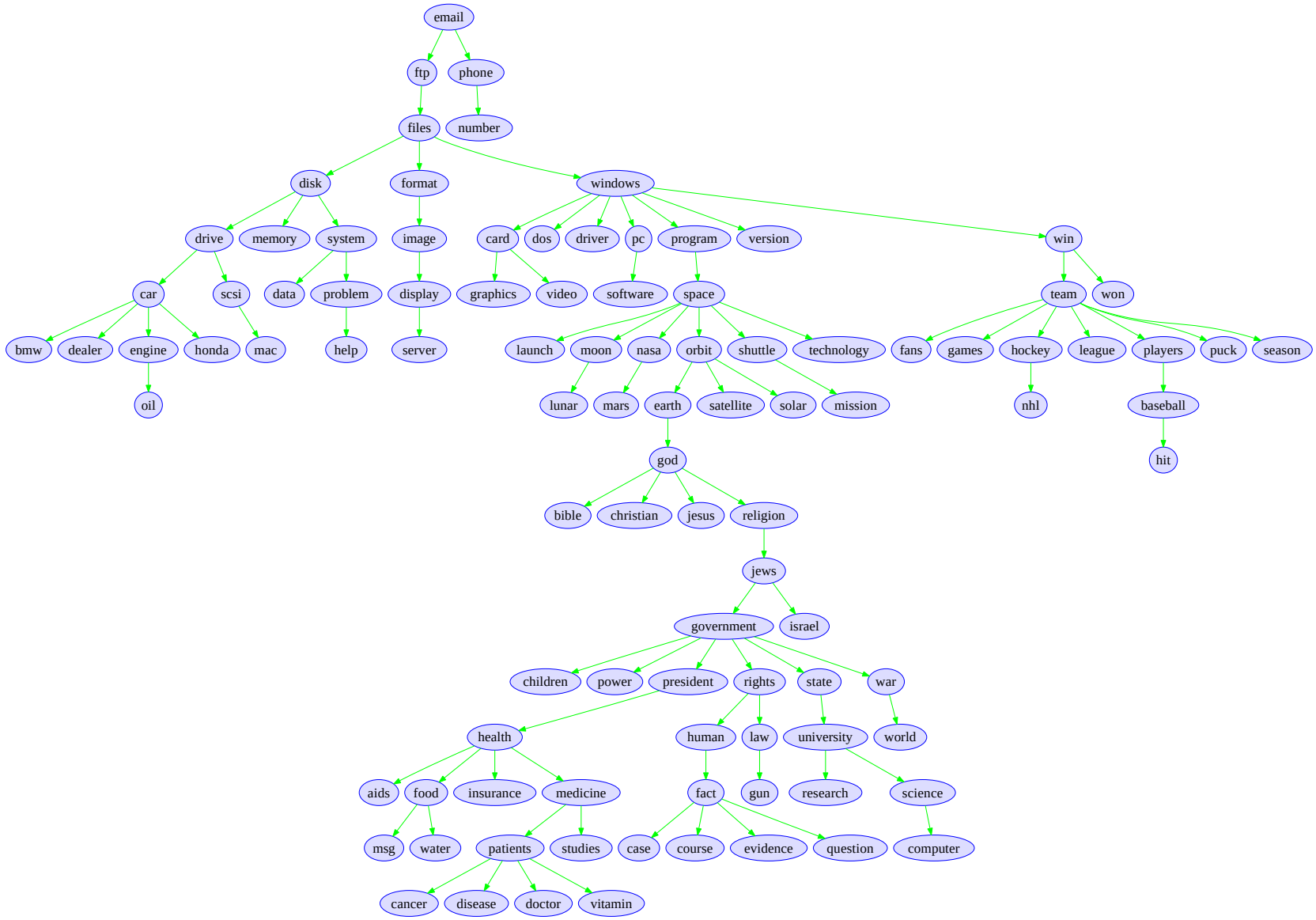

Figure 4.11: The tree structure that maximizes the BIC on the *news* data.

Given this large number of non-zero weights, it is difficult to appropriately visualize the many relationships present in the model. Visualization is further complicated by the number of weak relationships (that might represent false positives). Thus, to visualize the strongest associative effects in the estimated Markov blankets, we plot in Figure 4.9 all edges where the regression weight is above 0.5. In this figure, undirected edges represent edges where the edge was selected in both directions, while directed edges represent edges that were selected asymmetrically. In Figure 4.10, we plot the first local minimum found by the DAG-search with L1MB pruning (again restricted to edges where the weight is above 0.5). In both of these graphs, we can clearly see trends in different regions of the graph, including areas of words related to sports, cars, politics, religion, computers, and outer space. Unlike the dependency network estimated by L1MB, the DAG structure is a consistently parameterized density model. Thus, we can use it to measure likelihoods (this could be used to test whether a newsgroup post is spam, for example) or to generate independent samples from the distribution.

In Figure 4.11 we plot the tree structure that optimizes the BIC, calculated using the generalization of the Chow-Liu algorithm discussed in [Heckerman et al., 1995, §7.1] (note that the edge directions in this plot are meaningless as long as the they do not create a v-structure, hence there is no special significance to e-mail being the root of the graph). The tree structure is much less dense (containing only 99 edges) and hence much more interpretable than the L1MB or DAG structures. Further, we see a similar grouping of topics. However, because each node can have at most one parents, this model does not place direct edges between several highly related concepts. For example, the tree model assumes that the words 'hockey' and 'puck' are independent given the value of the 'team' variable. As a more extreme example, we must traverse six nodes to reach the word 'mac' from 'pc' (both the hockey:puck and mac:pc interactions are direct edges in the L1MB and DAG structures).

A further potential advantage of using general DAG models instead of restricting to trees is the ability of DAG models to 'explain away' different competing hypotheses. For example, in the DAG structure the word 'program' has both 'space' and 'disk' as parents. This reflects that we are more likely to see the word 'program' if we see the word 'space' or if we see the word 'disk'. Further, this also means that if we see the words 'program' and 'disk' then we are *less* likely to see the word 'space' (observing 'disk' explains why 'program' was observed, making it less likely that the word 'space' is present). This explaining away between parent variables does not happen in trees, since each edge can have at most one parent. The phenomenon of explaining away also does not happen in pairwise undirected graphical models, although it is possible that explaninig away can be modeled in undirected graphical models with higher-order potentials.

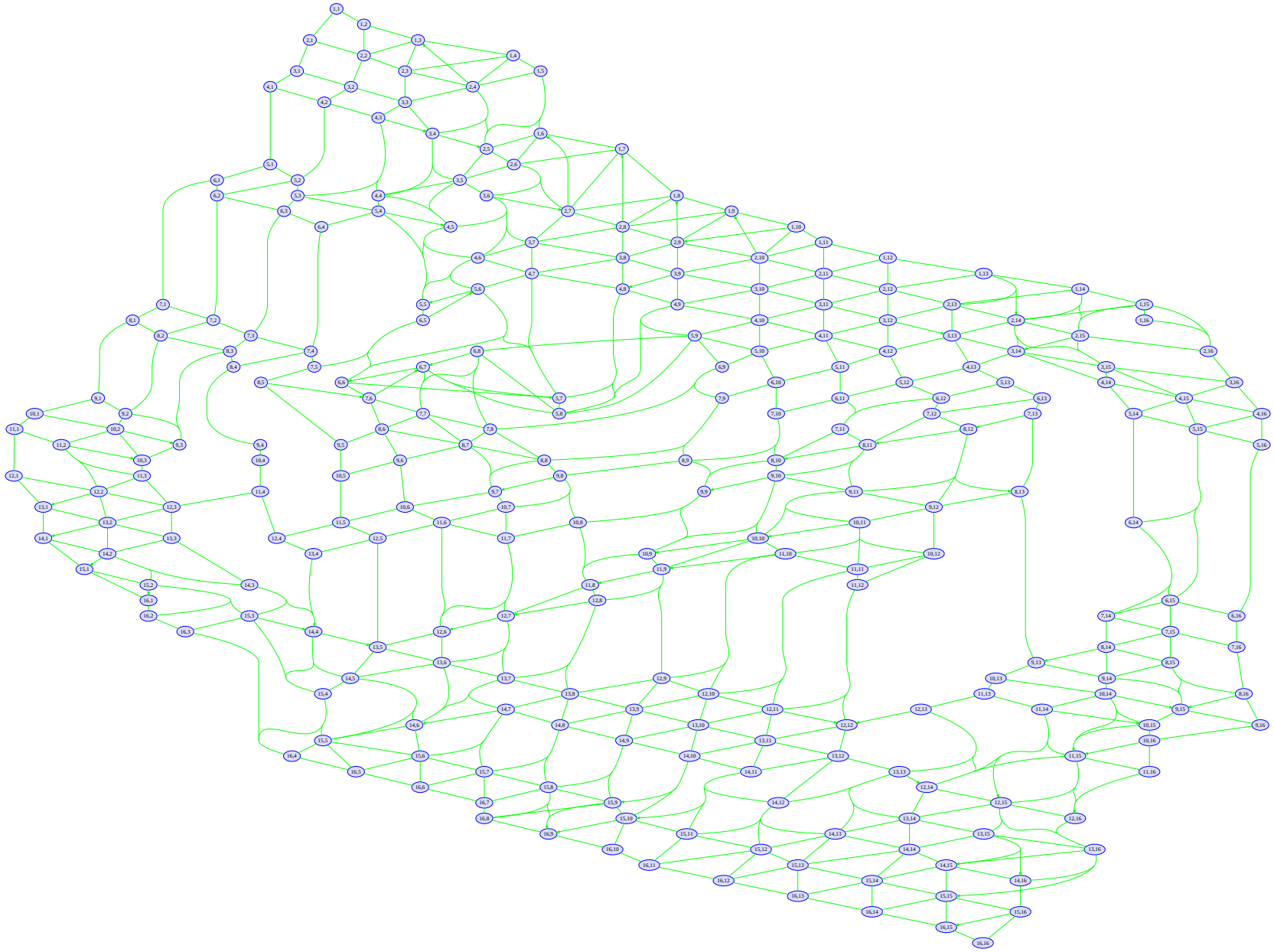

Figure 4.12: All edges with regression weight above 1 in the Markov blankets estimated by L1MB on the *usps* data. Undirected edges represent cases where the directed edge was found in both directions.

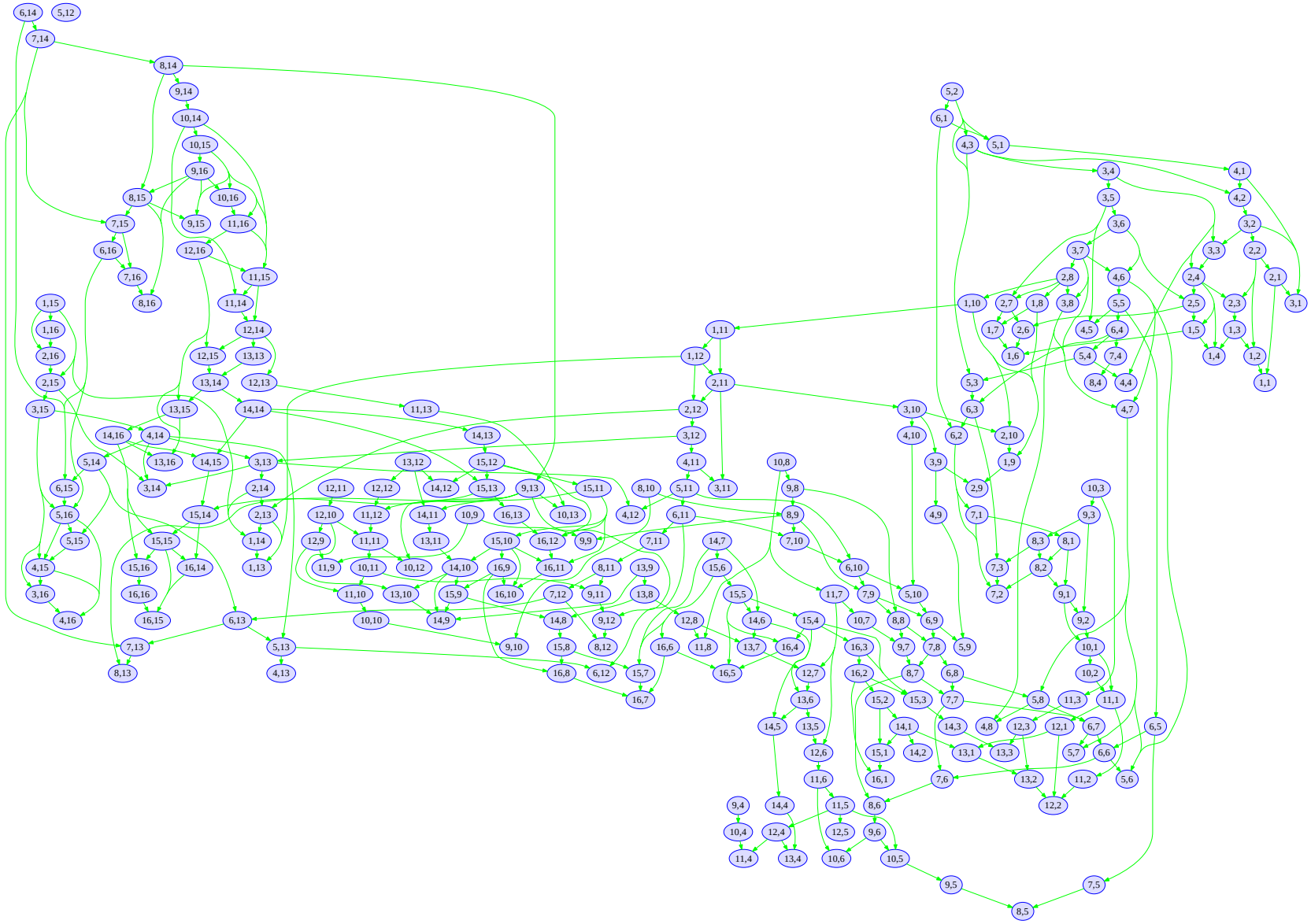

Figure 4.13: All edges with regression weight above 1.5 in the model found by DAG-search with L1MB pruning on the *usps* data.

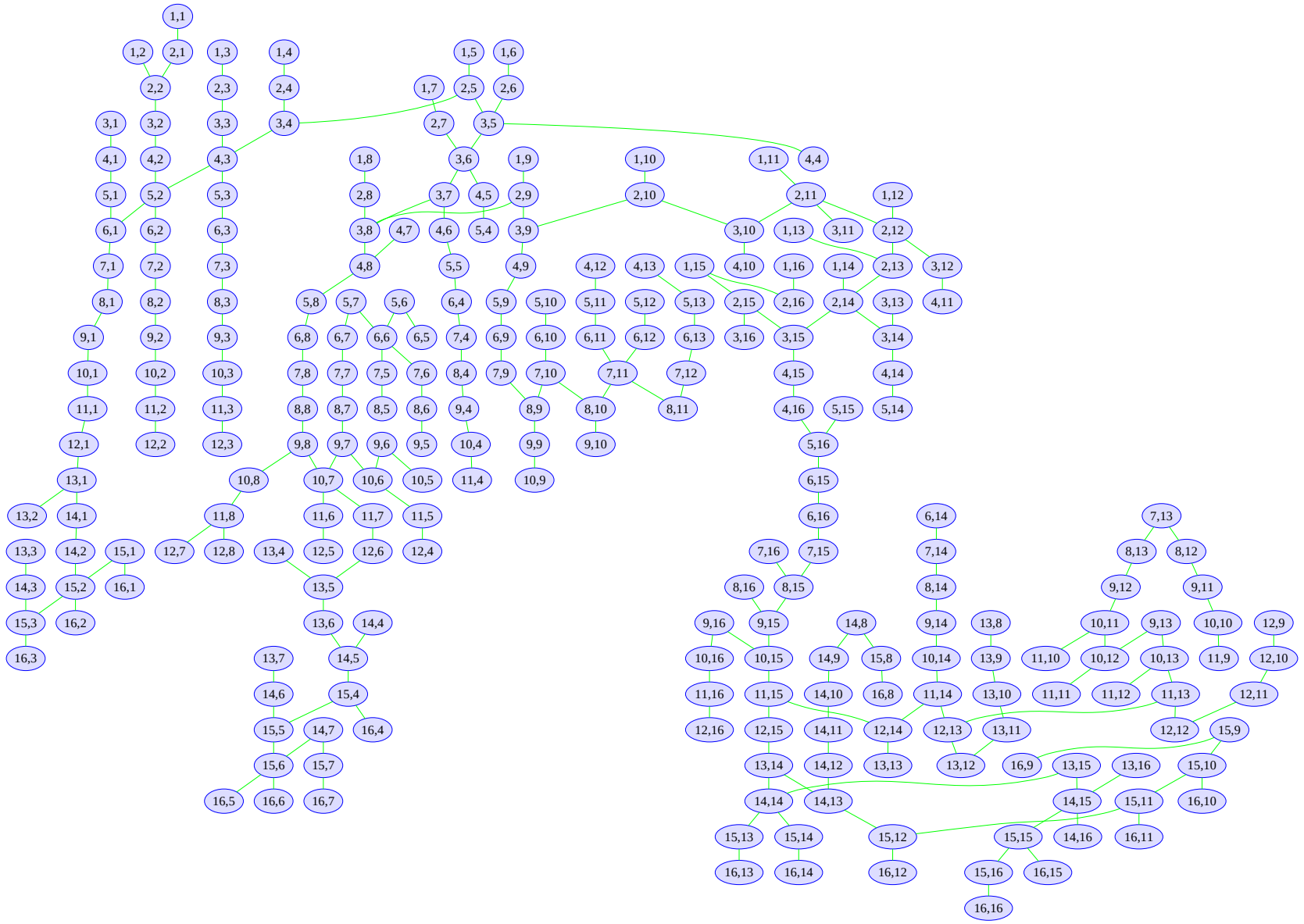

Figure 4.14: The optimal tree structure on the *usps* data.

We also examined the graph structures learned for the *usps* data. Since the 256 variables in this data set are the intensity values at individual pixels in a 16 by 16 image, we might expect the method to learn a structure resembling a two-dimensional grid structure where each node is connected to its horizontal and vertical neighbors. As in the *news* data set, the graphs learned on the *usps* data are relatively dense. For example, the dependency network estimated by the L1MB algorithm contained 4678 edges. Though this is small subset of the 65280 possible edges, it makes visualization of the graph difficult. To visualize the strongest interactions in the model, in Figure 4.12 we show all edges with a regression weight above 1 in the Markov blankets estimated by the L1MB method. Here, we see a structure roughly resembling what we might expect. Some nodes in the graph, such as $(12, 5)$, are connected to their horizontal and vertical neighbors in the image. There are also a large number of nodes that are connected to not only their horizontal and vertical neighbors, but also their diagonal neighbors. Some areas of the graph bear less of a resemblance to a grid model, including some areas that are more dense and some areas that are more sparse. In Figure 4.13, we plot the edges with strongest regression weights found by the DAG-search procedure with L1MB pruning (the full structure has 1813 edges). In this figure, we see that most of the strongest edges represent interactions between pixels that are adjacent in the image. However, there are a large number of nodes in this graph that are not connected to all of their horizontal and vertical neighbors. The likely cause for the model not including these obvious interactions is the acyclicity constraint, and the difficulty of searching through the space of graphs without violating this constraint. In Figure 4.14, we plot the tree structure that optimizes the BIC. In this structure we see that all edges are between adjacent pixels. However, we also see that (because the structure is constrained to be a tree) the structure is missing most of the dependencies between adjacent pixels, and the tree reveals very little about the true nature of the data.

In Chapter 5, we see that using ℓ_1-regularization for structure learning in undirected graphical models (that do not have an acyclicity constraint) can yield a much more intuitive structure on the *usps* data set. However, we note that while this undirected structure is more intuitive, we must make approximations when using the structure since it is computationally intractable to perform many operations with the structure. In contrast, the advantage of the DAG model in Figure 4.13 is that we can perform many operations with the model (such as computing probabilities of fully observed vectors and generating unbiased samples) in polynomial-time.

4.7 Similar Methods

In this chapter, we discuss a method that uses ℓ_1-regularization for structure learning in DAG models. Using ℓ_1-regularization in DAG models was also explored in [Li and Yang, 2004, 2005, Huang et al., 2006, Levina et al., 2008]. In this setion, we highlight the differences between the method outlined in this chapter and this prior work[32].

The first notable distinction between the present work and this prior work is that we focus on binary data, while these prior works all focused on Gaussian data. This may seem like a small difference, but the computational difference between using the analogous undirected model can be substantial. For example, in the worst case computing the probability of a continuous vector in GGMs costs $\mathcal{O}(p^3)$, while in Gaussian DAGs it costs $\mathcal{O}(p^2)$. In contrast, in the worst case computing the probability of a binary vector in IGMs is $\#P$-hard, while in sigmoid DAGs it only

[32]There has also been subsequent work done after the publication of [Schmidt et al., 2007b] that extends our work, we discuss this in the next section.

costs $\mathcal{O}(p^2)$. Thus, the computational savings achieved by considering DAGs in the binary case are much more substantial than in the Gaussian case. Further, the work we describe in this chapter can also be applied to the Gaussian case by simply using Gaussian CPDs (we give further details in [Schmidt et al., 2007b]). The method we discuss in this chapter also extends easily to other data types, including general discrete data or continuous data modeled using a robust distribution like student's t-distribution (we give details in the next section). Further, we can combine these different data types to model vectors with mixed types.

Another notable difference with the methods of [Li and Yang, 2005, Huang et al., 2006, Levina et al., 2008] is that we do not assume a known ordering of the variables. These prior works can be interpreted as a variant on the graphical LASSO where we parameterize the precision matrix in terms of an LDL^T factorization (where D is diagonal with positive entries and L is lower triangular with unit diagonal). Li and Yang [2005] seek to optimize L and D under a prior similar to the graphical LASSO, Huang et al. [2006] use direct ℓ_1-regularization of the coefficients in L, while Levina et al. [2008] use a variant of ℓ_1-regularization on the elements of L that encourages L to have low-bandwidth. If we consider the Gaussian case, the method we discuss in this chapter is most closely related to [Huang et al., 2006], since there is a direct correspondence between sparsity in linearly-parameterized Gaussian CPDs and sparsity in the Cholesky factor L. However, since we do not assume a fixed ordering, in the Gaussian case our method can be interpreted as using a $PLDL^TP^T$ factorization of the precision matrix instead of an LDL^T factorization, where P is a permutation matrix. That is, we additionally consider searching for the best permutation of the variables (since some permutations will yield higher degrees of sparsity than others).

Besides the extension beyond the Gaussian case, the distinction between this work and [Li and Yang, 2004] is more subtle. This method can also interpreted as being parameterized in terms of a $PLDL^TP^T$ factorization of the precision matrix. However, the optimization objective function is not clear in the three stage procedure of [Li and Yang, 2004]. The first stage of this procedure computes an ℓ_1-regularized Gaussian dependency network, where hypothesis testing is used to find the value of λ. The second phase uses the sparsity pattern obtained by the first phase, along with a series of independence tests, to increase the sparsity of the model and direct some of the edges. Finally, the third phase uses a search algorithm that explicitly optimizes a scoring function to determine the directionality of the remaining edges. It is only in this third phase that the method seeks to find a high scoring structure, and it is not clear how the first two phases relate to the score of the structure. In contrast, our method only has two phases; in the first phase we estimate a dependency network using ℓ_1-regularization, and in the second phase we search for a high scoring network restricted to the edges present in the dependency network. However, unlike this previous work we use the same score in both phases. This leads to a simpler and more elegant framework, and avoids the need to rely on the indirect performance measures provided by the results of hypothesis tests.

4.8 Extensions

This chapter has considered using ℓ_1-regularization for learning sparse DAG models of binary data with logistic regression CPDs. However, the method we discuss in this chapter can be extended in a straightforward way to a variety of other scenarios. This section gives an overview of several of these extensions.

4.8.1 Other CPDs

Instead of using logistic regression to represent CPDs over binary data, we could consider a wide variety of alternatives. In the following list, we discuss several examples:

- **Generalized linear models of binary data**: Logistic regression is a particular instance of a generalized linear model for binary classification. We can apply the ideas present in this chapter to other models in this family. For example, one of the simplest modifications we could consider is to use probit regression instead of logistic regression. In probit regression, we model the conditional probability of a child given a linear function of its parents using the cumulative distribution function of the standard normal,

$$p(x_i|\mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i) = \Phi(x_i(\mathbf{w}_i^T\mathbf{x}_{\pi(i)} + b_i)).$$

  The hybrid ℓ_1-regularization method can be easily modified to use probit regression for one or more of the node's CPDs, while the methods of Chapter 2 can also be applied to optimize the probit regression negative log-likelihood with ℓ_1-regularization. As another example of a generalized linear model of binary data, we could use complementary log-log regression,

$$p(x_i = 1|\mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i) = 1 - \exp(-\exp(\mathbf{w}_i^T\mathbf{x}_{\pi(i)} + b_i).$$

  As opposed to the logistic and probit regression models that are derived from cumulative distribution functions that are symmetric around zero, the complementary log-log regression is derived from the cumulative distribution function of the extreme-value distribution, and is asymmetric around zero. This property may be useful in cases where one of the binary states is much more likely than the other, such as in the *news* data where most words appear infrequently. For additional details about the probit, logistic, and complementary log-log models, see [Johnson and Albert, 1999, §3.1].

- **Gaussian models of real-valued data**: We could consider modeling real-valued data using Gaussian CPDs,

$$p(x_i|\mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mathbf{w}_i^T\mathbf{x}_{\pi(i)} - b_i)^2}{2\sigma^2}\right).$$

  In this case, optimizing the CPDs in terms of $\{\mathbf{w}_i, b_i\}$ involves solving an ℓ_1-regularized least-squares problem (for a fixed set of regression weights we can solve for σ analytically). The algorithms of Chapter 2 can also be applied to this case, although there exist many other solvers for this particular problem. If all CPDs are Gaussian then the joint distribution is also Gaussian. This makes it a possible alternative to the graphical LASSO as an efficient representation for multivariate Gaussian distributions. Indeed, while the graphical LASSO estimates a multivariate Gaussian with zeros in the precision matrix, a sparse Gaussian DAG will have zeros in the corresponding Cholesky factor L of a $PLDL^TP^T$ factorization of the precision matrix (the topological ordering determines the permutation matrix P). We discuss the case of Gaussian CPDs further in [Schmidt et al., 2007b], and in the appendix of that paper we give an extension of the LARS algorithm that allows efficient calculation of the BIC of all subsets of variables along the regularization path.

- **Robust distributions of real-valued data**: The Gaussian distribution has very thin tails, making it sensitive to outliers. A more robust alternative for real-valued data is to use a linearly parameterized Laplace distribution,

$$p(x_i|\mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i, \sigma) = \frac{1}{2\sigma} \exp\left(-\frac{|x_i - \mathbf{w}_i^T \mathbf{x}_{\pi(i)} - b_i|}{\sigma}\right).$$

  Here, computing the optimal ℓ_1-regularized $\{\mathbf{w}_i, b_i\}$ can be formulated as a linear program while the optimal σ for fixed regression weights can be computed analytically. An alternative to a linearly-parameterized Laplace distribution is a linearly-parameterized version of Student's t-distribution,

$$p(x_i|\mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i, \sigma, \nu) = \frac{\Gamma(\nu/2 + 1/2)}{\Gamma(\nu/2)\sqrt{\sigma\pi\nu}} \left[1 + \frac{(x_i - \mathbf{w}_i^T \mathbf{x}_{\pi(i)} - b_i)^2}{\sigma\nu}\right]^{-\nu/2-1/2}.$$

  In this case, one way to optimize the parameters $\{\mathbf{w}, \sigma, \nu\}$ is to alternate between the three types of parameters, optimizing each in turn. There is no simple correspondence between the parameters of a multivariate Laplace (or t) distribution and conditional independencies in the model, so fitting a DAG with Laplace (or t) distributions for CPDs might be a reasonable alternative for obtaining a robust multivariate distribution with conditional independence properties. More details about the univariate and multivariate Laplace and t distribution (as well as other multivariate distributions) can be found in [Lindsey and Lindsey, 2006].

- **Generalized linear models for discrete data**: Rather than being binary, it might be the case that a variable comes from a discrete set $\{1, 2, \ldots, k\}$. In this case, it is possible to use multinomial logistic regression for the CPDs [see Bishop, 2006, §4.3.4],

$$p(x_i = c|\mathbf{x}_{\pi(i)}, \mathbf{w}_{i\cdot}, \mathbf{b}_i) = \frac{\exp(\mathbf{w}_{ic})^T \mathbf{x}_{\pi(i)}}{\sum_{c'=1}^{k} \exp(\mathbf{w}_{ic'})^T \mathbf{x}_{\pi(i)}}.$$

  In this case, we have a vector $\mathbf{w}_{ic}$ associated with each state c for each variable i (typically, the vector for one of the states c is set to the zero vector if we are doing maximum likelihood estimation). In this case, we might also want to consider a different representation of the parent variables $\mathbf{x}_{\pi(i)}$. For example, we might encode a parent as a set of binary indicator variables (one for each state of the parent). Because of this, there is no longer a one-to-one correspondence between parameters of the model and edges of the graph. For cases like this, it might be more appropriate to use the group ℓ_1-regularization methods that are the focus of subsequent chapters.

  The multinomial logistic regression model assumes that the set of states $\{1, 2, \ldots, k\}$ is unordered. However, in many cases there may be a natural ordering among the states (i.e. 2 is closer to 3 than 4). In this case, ordinal logistic regression CPDs might be more appropriate [see Johnson and Albert, 1999, §4.1],

$$p(x_i = c|\mathbf{x}_{\pi(i)}, \mathbf{w}_i, b_i, \gamma_{i,\cdot}) = \frac{1}{1 + \exp(\gamma_{i,c} - \mathbf{w}_i^T \mathbf{x}_{\pi(i)} + b_i)} - \frac{1}{1 + \exp(\gamma_{i,c-1} - \mathbf{w}_i^T \mathbf{x}_{\pi(i)} + b_i)}.$$

  Included in this model is a set of adaptive thresholds $\{\gamma_{i,0}, \gamma_{i,1}, \ldots, \gamma_{i,k}\}$ on the cumulative distribution function, where $\gamma_{i,j} \leq \gamma_{i,j+1}$. Here, $\gamma_{i,0}$ is taken to be $-\infty$, while $\gamma_{i,k}$ is taken to be

∞, and one of the remaining values $\gamma_{i,c}$ is typically fixed at zero for identifiability. Optimizing the parameters of an ordinal regression CPD with ℓ_1-regularization can be formulated as a problem with bound constraints. This type of problem can be handled with a straightforward extension of the algorithms we describe in Chapter 2.

Beyond the above extensions to different types of data, one of the main advantages of DAG models is that they allow us to specify a multivariate distribution over vectors that contain multiple types of data. However, in these cases it is important to consider the representation of the parent variables $\mathbf{x}_{\pi(i)}$, and it will often be the case that the group ℓ_1-regularization methods of subsequent chapters are needed.

4.8.2 Other Extensions

We conclude this chapter by noting several other possible extensions:

- **Conditional DAGs**: In some cases, we may not want to model the distribution of some variables in the model. That is, we might want to model the conditional distribution of some subset of variables given another subset. We can learn a conditional DAG model using the techniques we describe in this chapter, by simply adding the constraint that nodes that are being conditioned on can have no parents (these constraints are simply added to the set of excluded edges generated by the L1MB algorithm). With these additional constraints, learning a sparse conditional DAG model proceeds as in the unconditional case we describe in this chapter.

- **Dynamic Bayesian Networks**: Dynamic Bayesian networks are a type of structured DAG model that generalizes hidden Markov models and Kalman filters for modeling multivariate time series data [see Murphy, 2002]. If we are given p-vectors at consecutive time points, we can consider trying to learn the structure of a dynamic Bayesian network that describes the dependency between time points. In these models, we learn an initial graph of the variables, as well as an inter-slice graph that models the variables conditioned on the variables at the previous time point. As discussed in [Friedman et al., 1998], we can extend structure learning methods for DAGs to the case of dynamic Bayesian networks. In particular, we can apply the methods present in this section to learn the structure of the initial graph, while learning the structure of the transition graph takes the form of learning a conditional DAG model (where we condition on the previous time point and tie the transition CPDs across time)[33].

- **Linear Non-Gaussian data**: Under some choices of the CPDs it is possible to distinguish between Markov-equivalent DAGs from observational data alone. In [Shimizu et al., 2005], the authors consider the case of a DAG model where each child is a linear function of its parents, with additive but non-Gaussian noise. They show that the optimal DAG can be recovered by post-processing the results of running an independent component analysis of the data (zeros in the independent components correspond to missing edges in the DAG). However, the independent component analysis does not yield entries that are exactly zero, and hence the method uses a set hypothesis tests to determine whether an effect is significant

[33] [Gustafsson et al., 2003] consider using ℓ_1-regularization to estimate the inter-slice graph under the constraint that parents must come from the previous time point. This constraint substantially simplifies structure learning since it excludes the possibility of creating cycles.

or not. If we used a differentiable measure of independence, we could apply ℓ_1-regularization to the factor loading matrix using the methods of Chapter 2 to learn a set of independent components with elements that are exactly zero.

- **More general models of intervention**: In our discussion of interventional data, we considered the case of *perfect* interventions. That is, we assumed that the effect of an intervention is to perfectly control the value of a single variable. However, in general we might want to consider more general interventions that affect multiple variables, or cases where we do not know the effect of the intervention. Eaton and Murphy [2007] consider the more general case of *uncertain* interventions. Here, the DAG model is augmented with a binary node for each intervention. By convention these intervention (or action) nodes are not allowed to have any parents in the DAG model, and the effect of an intervention is simply to set the value of the corresponding intervention node. This model is a special case of a conditional DAG model, so the methods we discuss in this section can be applied.

- **Removing false positives from L1MB**: In our experiments the L1MB algorithm typically did not exclude true edges, but included many false positive edges. Three potential sources of these false positives are (i) errors associated with estimating the Markov blanket, (ii) errors associated with using ℓ_1-regularization for variable selection, and (iii) errors associated with using the BIC.

  Estimating the edge set based on estimating the Markov blanket leads to false positives because the Markov blanket includes co-parents. We could consider several heuristics to try and remove these co-parents, such as testing whether variables in the Markov blanket are marginally independent (this is a straightforward calculation under the BIC or validation score). Such a procedure could remove false positives associated with co-parents that do not share common ancestor. Alternately, we could consider more elaborate schemes where we condition on different subsets of the variables in order to remove co-parents from consideration. We discuss a related approach in the appendix of [Schmidt et al., 2007b], in the context of applying the L1MB algorithm to graphs with a very large number of nodes.

  Using ℓ_1-regularization for variable selection is another potential source of false positives. As discussed in [Bach, 2008a], ℓ_1-regularization chooses all relevant variables with a probability tending to one exponentially fast (as the number of samples increases), but also chooses irrelevant variables with non-zero probability. This leads to false positives. To alleviate this problem, Bach [2008a] suggests applying ℓ_1-regularization to a set of bootstrap samples of the data set, and taking the intersection of the variables selected in the samples. We could consider applying this strategy in order to reduce the number of false positives.

  Alternately, we could consider several alternatives to ℓ_1-regularization. For example, the adapative LASSO [Zou, 2006] and SCAD penalties [Fan and Li, 2002] are two regularizers that have been proposed to give better properties than ℓ_1-regularization. The adaptive LASSO has been used for learning the structure in Gaussian dependency networks [Shimamura et al., 2007], while both the adaptive LASSO and SCAD penalties were examined for learning Gaussian graphical models in [Fan et al., 2009]. The methods of Chapter 2 can be used directly for regularization with the adaptive LASSO (it simply consists of a suitable setting of the individual regularization weights λ_i). Recent approaches for the (non-convex) SCAD regularizer use weighted ℓ_1-regularization as a sub-routine within a bound optimization scheme [Zou and Li, 2008], so the methods of Chapter 2 could also be used for this regularizer. Alternately,

as we discuss in the extensions section of Chapter 2, it is possible to extend the methods of Chapter 2 to be used directly for optimization with SCAD regularization.

Another approach that might remove false positives (from both phases) is to define a suitable prior and compute a Bayesian score [Cooper and Herskovits, 1992, Heckerman et al., 1995], instead of the simple BIC approximation. In general the Bayesian score can not be computed in closed form, so approximations to these integrals would be needed. Moghaddam et al. [2009] show that even simple approximations can lead to performance improvements over using the BIC. Further, as long as the score is separable across nodes, it is trivial to replace the BIC in our method with another score assessing the quality of graph structures.

Closely related to the Bayesian score is the sparse Bayesian learning regularizer discussed in [Wipf and Nagarajan, 2009], a generalization of automatic relevance determination methods. This prior is motivated by the form of the marginal likelihood in the Gaussian case, and the authors of this work show that this non-separable, non-convex regularizer has several appealing advantages over ℓ_1-regularization. As with the SCAD regularizer, current methods for using this regularizer use a bound optimization strategy where weighted ℓ_1-regularization is used as a sub-routine, so the methods of Chapter 2 could be used to solve this sub-routine.

- **Other structure search methods**: We have considered a simple DAG-search method to perform the search over possible DAG structures. However, we could augment/replace this search procedure with any of the methods we discuss in Section 4.1. Indeed, [Vidaurre et al., 2010] have applied our hybrid ℓ_1-regularization method (with Gaussian CPDs) where the DAG-search has been replaced with a search through the space of Markov-equivalent structures. Another possible search strategy is the constrained optimal search of [Perrier et al., 2008]. Given a structure constraining the set of possible edges, Perrier et al. [2008] describe a method for finding the optimal DAG that is exponential in the degree of this structure. Hence, if the L1MB algorithm returns a structure with a sufficiently low degree, it is possible to find the optimal DAG structure even if the number of nodes in the original graph is very large.

- **Regularization of the CPD parameters**: The experiments in this chapter have used maximum likelihood estimates of the parameters. In many cases, we are able to obtain a better model in terms of validation score by using a regularized estimate, such as an ℓ_2-regularized or ℓ_1-regularized estimate. As long as the regularizer does not violate parameter independence or parameter modularity, searching for an optimal regularizer within each CPD only adds a small computational overhead to the L1MB and DAG-search procedure. There has also been work on estimating the number of degrees of freedom of ℓ_1-regularized estimates. For example, [Zou et al., 2007] shows that the number of non-zero coefficients is an unbiased estimate of the number of degrees of freedom when using ℓ_1-regularized parameter estimates within the BIC.

- **Non-linear CPDs**: We have concentrated on CPDs that are linear in the values of the parent variables. This is similar to the pairwise assumption in undirected graphical models, and it similarly may be restrictive in some scenarios. We can relax this assumption if we consider using non-linear transformations of the parent variables. For example, to gain the representational power of tabular CPDs we could use CPDs that are linear in the set of indicator functions over possible parent configurations. Alternately, we could add products

of the parent variables (or other such transformations) as additional terms in the CPDs. Under many choices of non-linear CPDs, it will typically be more appropriate to use (disjoint or overlapping) group ℓ_1-regularization of the CPD parameters, similar to the methods we discuss in the next two Chapters.

- **Convex approaches to DAG learning**: In this chapter, we resorted to a search-based method because of the acyclity constraint on the graph structure. However, there has been a limited amount of work on convex formulations of DAG learning. Guo and Schuurmans [2006] consider a convex relaxation involving semi-definite programming to approximate a node ordering, while Jaakkola et al. [2010] formulate DAG learning as a binary linear program with linear constraints that enforce acyclitiy. It may be possible to apply ℓ_1-regularization with one of these characterizations of the acyclicity constraint.

Chapter 5

Undirected Graphical Model Structure Learning

We now turn to task of the structure learning in pairwise undirected graphical models. In some sense, structure learning is easier in the undirected case because we do not have a global acyclicity constraint; given some candidate undirected structure, we still obtain a legal undirected structure after any edge addition. However, in another sense structure learning in undirected models is much harder because of the global normalization. Unlike in the DAG case where we used separability of the log-likelihood to efficiently evaluate single-edge modifications, the lack of separability of the log-likelihood in the undirected case means that we must re-fit all parameters after any edge addition or deletion (while even evaluating the score given a fixed graph structure can be computationally expensive or intractable). This makes methods based on local search extremely expensive. In the next two sections, we briefly review several of the search-based and constraint-based strategies that have been proposed for structure learning in undirected graphical models. After this, the remainder of the section focuses on the (potentially much faster) methods based on ℓ_1-regularization.

5.1 Search-based and Constraint-based Methods

Whittaker [1990, §8.2] contains a list of references from the statistics literature from the 1970s and 1980s on structure learning in (Gaussian and log-linear) undirected graphical models. Typically, these methods start with the empty structure, and search for the best possible edge addition. A deviance score (based on a maximum likelihood estimate) is used in measuring the quality of a structure, and a hypothesis test is used to determine whether each new edge improves the score by a sufficient margin. The algorithm terminates once one of these hypothesis tests fails. Because a likelihood-based criteria is used, this termination criteria is used to avoid adding all possible edges. Alternative methods also exist that start with the dense model and successively remove edges. Classic examples of these types of methods include [Dempster, 1972] for the Gaussian case, and [Goodman, 1971] for the log-linear case. In general, this procedure is extremely expensive if the number of variables is non-trivial; these procedures must fit $\mathcal{O}(p^2)$ undirected models at each of the $\mathcal{O}(p^2)$ steps. These types of greedy methods seem to have fallen out of favor with the introduction of methods that use a score that encourages sparsity (such as the BIC or marginal likelihood criteria), and methods that directly seek to optimize these types of scores. These and other classical methods, as well as methods based on the BIC, are discussed further in Edwards [2000, §6].

In order to avoid the expensive computations associated with general undirected graphical models, many authors have considered restricting the search to the set of decomposable graphical models. A subset of the extensive work on this topic includes [Wermuth, 1976, Malvestuto, 1991, Dawid and Lauritzen, 1993, Madigan and Raftery, 1994, Xiang et al., 1997, Giudici and Green,

1999, Deshpande et al., 2001]. Decomposable models correspond to the subset of undirected graphical models where the graph structure is chordal [Whittaker, 1990, §12]. The set of conditional independencies in decomposable models can be represented as both an undirected and a directed acyclic graphical model (in particular, a chordal undirected graph encodes the same set of conditional independences as a DAG with no v-structures). Subsequently, in chordal undirected models it is possible to take advantage of many of the convenient properties of directed models. Particularly relevant for structure learning is that the likelihood can be evaluated efficiently, and that parameter estimation can be done locally. Another noteworthy property of decomposable models is that the marginal likelihood given a fixed graph structure can be evaluated in closed form (with a suitably chosen conjugate prior). This is in contrast to general undirected graphical models, where even evaluating the BIC may be intractable since it requires computing the rank of an exponential-sized matrix [Koller and Friedman, 2009, §20.7.3]. However, like acyclicity, the constraint that a graph must be chordal is a global (non-convex) constraint. This negates the (potential) advantage of structure learning in undirected models, since we would have local optima (that are not global optima) even if we use ℓ_1-regularization of the parameters for structure learning. Thus, we would need to consider techniques similar to those used in Chapter 4 (i.e. greedy search) to learn chordal graphs. Further, if we did this we would still be restricted to a strict subset of the set of distributions whose independence properties can be represented by DAGs.

An alternative to using chordal graphs is to place an explicit restriction on the treewidth of the graph. By placing a restriction on the treewidth we guarantee that inference in the model can be performed in polynomial time. Indeed, bounded treewidth networks allow polynomial-time computation of quantities that are difficult to compute even in directed and chordal models (such as computing conditional probabilities). If the treewidth is restricted to be 1 (corresponding to tree-structured graphs) the optimal maximum likelihood structure can be found in polynomial time [Chow and Liu, 1968]. Heckerman et al. [1995] discuss extending this methodology to other scores. For any bound greater than 1, finding the optimal bounded treewidth structure (under various scoring criteria) is NP-hard [Srebro, 2003] (even determining the treewidth of a graph is NP-hard in general). Nevertheless, several recent works have examined this case. For example, [Karger and Srebro, 2001] give a polynomial-time approximation scheme for this problem, while Bach and Jordan [2001] consider searching in the space of graphs with bounded treewidth. Evaluating the score achieved by edge modifications is still relatively expensive in these graphs, since low treewidth graphs will not in general be chordal. Thus, Bach and Jordan [2001] consider heuristics for evaluating the scores of neighboring graphs. Narasimhan and Bilmes [2004] have considered constraint-based polynomial-time strategies for learning bounded tree-width networks in the probably approximately correct (PAC) learning framework for the consistent case (when the data is generated according to an undirected graphical model), by solving a series of submodular optimization problems to discover conditional independencies. Chechetka and Guestrin [2007] give a related constraint-based method that is polynomial-time for a more general class of data-generating distributions (though the algorithm remains exponential in the bound on the treewidth). Shahaf et al. [2009] consider a graph cut procedure for recursively partitioning the nodes to learn bounded-treewidth networks, that has certain theoretical guarantees and has shown good empirical performance. The disadvantage of considering only networks with bounded treewidth is simply that many distributions can not be represented as a network with bounded treewidth, so a non-trivial treewidth might be necessary to build a good model of a particular data set.

There has also been work on constraint-based methods for learning the structure with a con-

straint on the number of neighbors. For example, Koller and Friedman [2009, §20.7.1] give a polynomial-time constraint-based method for learning bounded-degree networks (in the consistent case). Such constraint-based methods are appealing because they do not require parameter estimation. However, we note that these methods must rely on the same assumptions and be subject to the same criticisms as the constraint-based methods for learning DAGs we discuss in Section 4.2. Further, as discussed in [Koller and Friedman, 2009, §20], constraint-based methods do not distinguish between Markov-equivalent graph structures that use different factorizations (this is applicable when we remove the pairwise assumption). Recently, Abbeel et al. [2006] give an exponential-time algorithm for learning bounded-degree networks in the PAC learning framework (for the consistent case) that also learns the factorization.

5.2 ℓ_1-Regularization

Let us temporarily assume that each edge has only a single parameter associated with it. Then, as we discuss in Chapter 1, we can formulate the problem of learning a sparse pairwise structure with ℓ_1-regularization as

$$\min_{\mathbf{w},\mathbf{b}} -\sum_{m=1}^{n}[\sum_{i=1}^{p}[\log \phi_i(x_i^m, \mathbf{b}_i) + \sum_{j=i+1}^{p} \log \phi_{ij}(x_i^k, x_j^k, \mathbf{w}_{ij})]] + n \log Z(\mathbf{w}, \mathbf{b}) + \lambda||\mathbf{w}||_1. \tag{5.1}$$

During the past five years, there has been intense interest from various communities in using this formulation for structure learning in undirected graphical models. The reasons for this are simple. First, it is an appealing notion to formulate the problem of fitting a sparse regularized model to data with unknown structure as a convex optimization problem. Second, this formulation does not impose any constraints (such as decomposability, bounded treewidth, or bounded degree) on the structures that can be learned. Third, we might hope to inherit the appealing properties of ℓ_1-regularization that are known for regression and classification that we discuss in Chapter 1. Finally, unlike search-based methods where we must solve a non-separable convex optimization problem for every possible edge addition/deletion that we consider, when we use ℓ_1-regularization we only need to solve one convex optimization problem that (arguably) has a comparable difficulty level. This means that using ℓ_1-regularization is much faster than using a search-based method.

5.3 Approximate Objectives

As we discuss in Section 1.4, most of the work that examines ℓ_1-regularization for structure learning in undirected graphical models focuses on GGMs and IGMs. In the case of GGMs, the normalizing constant can be computed in polynomial time and hence the problem can tractably be solved even with a non-trivial number of nodes (for example, using the methods of Chapter 2). For IGMs and the more general pairwise log-linear models we describe in Section 1.5, it may be intractable to evaluate the objective function in (5.1) since the graph structure resulting from the sparsity pattern may not have a low treewidth. Thus, for discrete data we must consider approximate objective functions.

A classic technique for addressing the intractability of evaluating the likelihood in undirected models is to replace the likelihood with the product of univariate conditionals. This is known as a pseudo-likelihood [Besag, 1975]. This approximation is consistent, in the sense that if the data is

generated from an undirected graphical model then as the number of training examples grows the maximum pseudo-likelihood estimate converges to the maximum likelihood estimate [see Koller and Friedman, 2009, Theorem 20.3]. In [Schmidt et al., 2008], we considered using a pseudo-likelihood in (5.1), giving the problem

$$\min_{\mathbf{w},\mathbf{b}} -\sum_{k=1}^{m}[\sum_{i=1}^{p} \log p(x_i^m|\mathbf{x}_{-i}^m, \mathbf{w}, \mathbf{b})] + \lambda||\mathbf{w}||_1, \tag{5.2}$$

We note that the conditional probability of a node given all other nodes takes the form

$$p(x_i|\mathbf{x}_{-i}, \mathbf{w}, \mathbf{b}) = \frac{1}{Z_i}\phi_i(x_i, \mathbf{b}_i) \prod_{\{j|j\neq i\}} \phi_{ij}(\mathbf{x}_{ij}, \mathbf{w}_{ij}),$$

where the local normalizing constant Z_i only sums over possible assignments to x_i (and thus can be tractably evaluated). As we discuss in 1.4, in the IGM case these conditionals take the form of logistic regression likelihoods. In the general pairwise discrete case, these conditionals take the form of a multiclass logistic regression likelihood [Bishop, 2006, §4.3.4], where the features are defined in terms of the values assigned to neighboring nodes. Thus, the ℓ_1-regularized pseudo-likelihood takes the form of a set of dependent (multiclass) logistic regression problems, where the dependency arises because each set of edge parameters $\mathbf{w}_{ij}$ is present in the conditional $p(x_i|\mathbf{x}_{-i})$ and $p(x_j|\mathbf{x}_{-j})$. However, the joint optimization of these dependent (multiclass) logistic regression problems is easily handled by the methods of Chapter 2.

As we discuss in Section 1.2, an alternative pseudo-likelihood approximation is to learn a dependency network. This is identical to (5.2), but where we make a separate copy of each edge parameter set $\mathbf{w}_{ij}$ for each conditional. Although this problem can be solved slightly more efficiently, we must heuristically construct each edge parameter out of its two copies. Hofling and Tibshirani [2009] compared the symmetric pseudo-likelihood (5.2) to two ways of obtaining a single estimate out of this asymmetric dependency network pseudo-likelihood. They found that while all three estimates were good approximations, that the symmetric approximation had an advantage over the two asymmetric versions.

An alternative to using a pseudo-likelihood approximation of the likelihood is to use a variational approximation of the logarithm of the normalizing constant (known as the log-partition function) [see Wainwright and Jordan, 2008]. In general, such approximations are not consistent. However, theoretical arguments by Wainwright [2006] suggest that it can be beneficial to use such an approximation in certain scenarios, if the same approximation will subsequently be adopted when using the model. Lee et al. [2006b] considered the Bethe free energy approximation to the log-partition function, implemented using the loopy belief-propagation message-passing algorithm. This approximation is appealing because it is exact for tree-structured graphs. Thus, as we move along the regularization path this approximation is exact until the graph has loops (in contrast, pseudo-likelihood approximations are only exact if we have no edges). However, for graphs with loops the Bethe approximation will not generally be convex, nor does it give an upper bound on the log-partition function. Further, the use of loopy belief propagation might lead to discontinuities in the objective function because of non-convergence of the algorithm or because it converges to different local optima.

As an alternative to the (non-convex) Bethe approximation, in this work we also consider using the (non-convex) mean-field variational approximation (with a fully factorized approximating

distribution), and consider using a "convexified" (tree-reweighted) Bethe approximation [see Wainwright and Jordan, 2008, §5 and §7]. The latter approximation uses a convex combination of tree-structured approximations to give a convex upper bound on the log-partition function, but uses a clever re-parameterization that allows the number of trees to potentially be very large without an increase in computation. In particular, the method uses a minor variant on the loopy belief-propagation message-passing algorithm that utilizes a set of edge appearance probabilities (the mean field approximation is also computed by a message-passing algorithm). For each edge, the edge appearance probability is the (weighted) distribution of times the edge appears in one of the tree-structured approximations. We obtain the regular Bethe approximation if these are all set to 1, but this leads to non-convexity as it is not a valid distribution over tree-structured graphs (unless the graph is actually a tree). In our experiments, we considered using all possible spanning trees of the dense graph (with equal weight) in the approximation. The probability of an edge appearing in a random spanning tree of a fully connected graph on p nodes is $2/p$ for $p \geq 2$ (each spanning tree consists of $(p-1)$ edges selected in an exchangeable way from the $p(p-1)/2$ edges). Note that we use these edge appearance probabilities even if some of the edges have all parameters set to zero.

5.4 Group ℓ_1-Regularization

In the case of IGMs, sparsity in the parameters directly corresponds to sparsity in the graph structure. However, this is no longer the case if we consider more general potentials like the gIsing or full edge potentials from Section 1.5 where each edge has multiple parameters. In these models we must set all parameters associated with an edge to zero in order to remove the edge from the model, and thus ℓ_1-regularization does not directly encourage graphical sparsity. Indeed, ℓ_1-regularization completely ignores that graphical sparsity might lead to a more parsimonious graph structure or greater computational savings than sparsity of individual edge weights. In order to encourage sparsity in terms of edges instead of individual edge parameters, we can use group ℓ_1-regularization.

Utilizing group ℓ_1-regularization to encourage sparsity in terms of groups of variables was proposed by Bakin [1999] in the context of regression. In this work Bakin considered penalizing the ℓ_1 norm of the ℓ_2 norms of the groups in order to encourage sparsity at the group level. For our problem, we have one group for each edge and the group contains all parameters associated with the corresponding edge. Thus, we can write the problem of estimating a sparse regularized structure with group ℓ_1-regularization as

$$\min_{\mathbf{w},\mathbf{b}} -\sum_{m=1}^{n}[\sum_{i=1}^{p}[\log \phi_i(x_i^m, \mathbf{b}_i) + \sum_{j=i+1}^{p} \log \phi_{ij}(x_i^m, x_j^m, \mathbf{w}_{ij})]] + n \log Z(\mathbf{w}, \mathbf{b}) + \lambda \sum_{i=1}^{p}\sum_{j=i+1}^{p} ||\mathbf{w}_{ij}||_p, \tag{5.3}$$

for some norm $||\cdot||_p$ (using an approximate objective gives an analogous formulation). While Bakin [1999] considered penalizing the ℓ_2-norms of the groups (corresponding to ℓ_1-regularization of the lengths of the vectors $\mathbf{w}_{ij}$), other authors have subsequently considered using other norms that also achieve group sparsity[34]. For example, Turlach et al. [2005] use the ℓ_∞ norm of the groups in the context of multiple linear regressions, corresponding to ℓ_1-regularization of the maximum

[34]We obtain ℓ_1-regularization as in (5.1) if each group contains only a single variable (under any choice of norm), or if we use the ℓ_1-norm of the individual groups.

absolute values within the groups (but not penalizing elements of the groups that do not achieve the maximum value). We considered using (5.3) with the ℓ_∞ norm of the groups in [Schmidt et al., 2008].

Since the ℓ_2 norm places no bias on the direction, in some sense it is the only norm that does not encourage additional structure in the edge potentials. This is as opposed to the degenerate case of using the ℓ_1 norm that prefers sparsity within the groups, and it also differs from the ℓ_∞ norm that encourages elements within the same group to have exactly the same magnitude. However, this latter property produces some interesting biases when using the ℓ_∞ norm. For example, with the gIsing potentials it encourages all edge weights (associated with the same edge) to have the same magnitude. If these weights also have the same sign, it encourages the gIsing weights to take the exact same value and to subsequently become Ising potentials. With full potentials, using the ℓ_∞ norm also encourages patterns of tied weights within the potentials, but places no restriction on what elements of the individual edge potential matrices are tied. Thus, it might lead to some edges using Ising potentials, some edges using gIsing potentials, some edges taking other patterns, and some edges having no pattern (in general there will be no pattern when the ℓ_2 norms of the groups is used).

While previous work on group ℓ_1-regularzation has only considered the ℓ_2 or ℓ_∞ norms of the groups, these are not the only possible choices of the group norm. For the case of full potentials, in this work we also consider using the nuclear norm of the edge weight matrix. This can be viewed as an extension of the nuclear norm regularizer described in [Fazel et al., 2001] to the case of groups. The nuclear norm penalizes the sum of the singular values of the matrix, and using it within a group ℓ_1-regularization framework encourages not only group sparsity[35] but encourages the edge weight matrices to be low rank. The advantage of this is that, for $k > 2$, this may lead to a more parsimonious representation of the full edge weight matrix. For models with many states this might lead to a substantial reduction in the number of parameters (and degrees of freedom) in the final model, and because of this it represents an alternative to the weight-tieing used in the Ising or gIsing potentials.

In cases where groups have a single element, the methods of Chapter 2 can be used to solve the optimization problem, while the methods of Chapter 3 can be used to solve the general case when we use the ℓ_2 norms of the groups. In the next section, we discuss simple extensions to the methods of Chapter of 3 that allow us to handle the ℓ_∞ and nuclear norms of the groups.

5.5 Optimization with General Group Norms

In this section we consider the generalization of (3.1) where we penalize some norm $||\cdot||_p$ of the groups:

$$\min_{\mathbf{x}} f(\mathbf{x}) \triangleq L(\mathbf{x}) + \sum_A \lambda_A ||\mathbf{x}_A||_p. \tag{5.4}$$

Note that in this expression, we do not necessarily have to use the same norm for each group. By the positive homogeneity property of norms, for any choice of norm this function is non-differentiable if an entire group of variables is exactly zero. Depending on the particular norm, there may be other non-differentiabilities. For example, with the ℓ_∞ norm the objective is also non-differentiable whenever more than one variable in a group achieves the largest magnitude within the group.

[35] All elements of the matrix are necessarily zero if all singular values are zero.

To apply the SPG or PQN method to solve (5.4), we must convert it into a differentiable optimization problem over a convex set. As before, we do this by introducing an additional variable g_A for each group A and optimize subject to the constraint that $g_A \geq ||\mathbf{x}_A||_p$:

$$\min_{\mathbf{x},\mathbf{g}} \quad L(\mathbf{x}) + \sum_A \lambda_A \, g_A, \quad \text{subject to} \quad g_A \geq ||\mathbf{x}_A||_p, \forall_A. \tag{5.5}$$

By convexity of norms, the constraints in this problem define a convex set for any choice of norm.

In an addendum to [Schmidt et al., 2008], we show how to compute the projection for this problem when we penalized the ℓ_∞ norm of the groups[36]. The cost of solving the sub-problem in this case is $\mathcal{O}(|A| \log |A|)$, since in the worst case we may need to sort the elements of $\mathbf{x}_A$. In the degenerate case where we use the ℓ_1 norm of the groups, this problem can be solved in $\mathcal{O}(|A|)$ (expected time) using a simple extension of the randomized algorithm outlined in [Duchi et al., 2008b]). Although penalizing the ℓ_1 norm of the groups is not interesting on its own since this choice of group norm reduces to regular ℓ_1-regularization, we can use this as a sub-routine for computing the projection when we penalize the nuclear norm of the groups. In particular, similar to [Cai et al., 2010], the projection for an individual group can be computed in $\mathcal{O}(|A|^{3/2})$ by computing the singular value decomposition of the group [Golub and Van Loan, 1996, §2], applying the ℓ_1 norm method to the singular values, then reforming the matrix with the modified singular values to form the projected matrix. We give more details regarding these projections in Appendix B, but for now we simply note that for all the norms we consider it is possible to compute the projection efficiently for reasonably-sized groups.

Wright et al. [2009] discuss computing the soft-threshold operator for different choices of the norm in group ℓ_1-regularization. In the case of the ℓ_∞ norm, the solution for an individual group is given by an explicit element-wise threshold operator

$$\mathcal{S}_{\mathcal{R}}(\mathbf{x}_A, \alpha)_i = \text{sgn}(x_i) \min\{|x_i|, \theta_A\},$$

where the threshold θ_A used by the group is given by the maximum over i of the absolute difference between $\mathbf{x}_A$ and the result of projecting $\mathbf{x}_A$ onto the ℓ_1-ball of radius $\alpha\lambda_A$ [Duchi and Singer, 2009]. Duchi et al. [2008b] give a randomized algorithm with an expected $\mathcal{O}(|A|)$ runtime for computing this projection. In the case of the nuclear norm, the soft-threshold is given by applying the soft-threshold rule $\sigma_i \leftarrow \max\{0, \sigma_i - \alpha\lambda_A\}$ to the singular values σ_i of the matrix groups [Cai et al., 2010].

We can also generalize the active-set method from Section 3.4 to the case of an arbitrary group norm $||\cdot||_p$. To test whether groups with all elements zero can locally improve the objective function by moving them away from zero, we need to characterize the sub-differential of the regularizer in (5.4). To do this, we use a non-standard (but equivalent) definition of the sub-differential of a convex function $R(\mathbf{x})$ [Combettes and Wajs, 2005, §2.1]:

$$\partial R(\mathbf{x}) \triangleq \{\mathbf{g} | R(\mathbf{x}) + R^*(\mathbf{x}) = \mathbf{x}^T\mathbf{g}\},$$

where $R^*(\mathbf{x})$ is the convex conjugate of $R(\mathbf{x})$ [see Boyd and Vandenberghe, 2004, §3.3]. If $R(\mathbf{x})$ is a norm, $R(\mathbf{x}) \triangleq ||\cdot||_p$, the convex conjugate is given by a $(\infty, 0)$ indicator function on the dual

[36] Quattoni et al. [2009] show how to solve the related problem of projecting onto the norm ball defined by the ℓ_1 of ℓ_∞ norms.

norm unit ball [Boyd and Vandenberghe, 2004, Exercise 3.26]

$$R^*(\mathbf{x}) \triangleq \begin{cases} 0 & \text{if } ||\mathbf{x}||_q \leq 1, \\ \infty & \text{otherwise.} \end{cases}$$

Here, we use $||\cdot||_q$ to denote the dual norm of $||\cdot||_p$ [Boyd and Vandenberghe, 2004, §A.1.6]. It follows that the sub-differential of the regularizer for a group with $\mathbf{x}_A = 0$ is all vectors with dual norm less than or equal to λ_A. Thus the optimality condition that $\mathbf{0} \in \partial f(\mathbf{x})$ in (5.4) for a group with $\mathbf{x}_A = \mathbf{0}$ is

$$||\nabla_A L(\mathbf{x})||_q \leq \lambda_A.$$

Using this, an active-set method generalizing the one in Section 3.4 to the case of an arbitrary group norm $||\cdot||_p$ is

- Find groups A such that $\mathbf{x}_A \neq \mathbf{0}$, or $\mathbf{x}_A = \mathbf{0}$ and $||\nabla_A L(\mathbf{x})||_q > \lambda_A$.
- Solve the problem with respect to these groups.

The dual norms for all norms considered in this work are given in [Boyd and Vandenberghe, 2004, §A.1.6]. The ℓ_2 norm is its own dual, giving the algorithm of Section 3.4. The dual norm of the ℓ_1 norm is the ℓ_∞ norm, giving the algorithm of Section 2.5. Since the dual norm of the ℓ_∞ norm is the ℓ_1 norm, if we penalize the ℓ_∞ norm of the groups we add a group if the absolute value of the gradient of any element of the group is above λ_A. Finally, the dual of the nuclear norm is the ℓ_2 operator norm, so we add matrix groups if the largest singular value of the matrix containing the values of the gradient elements exceeds λ_A. To end this section we note that when we repeated the experiments of Chapter 3 with other choices of the group norm, the relative performance of the different optimization methods was very similar.

5.6 Blockwise Sparsity

Duchi et al. [2008a] consider an alternate use of group ℓ_1-regularization within the context of GGMs. Their model assigns each node in the graph a *type*. They consequently use ℓ_1-regularization of the edges between variables of the same type, but group ℓ_1-regularization of the set of edges between different types. That is, they encourage sparsity in terms of the blocks of the precision matrix that represent interactions between variables of different types. We refer to this as blockwise-sparsity, since it encourages sparsity in terms of pre-defined blocks of the precision matrix. Duchi et al. [2008a] penalize the ℓ_∞ norm of the blocks, and give a projected gradient method for solving a Lagrangian dual problem in the case of GGMs. In [Schmidt et al., 2009a], we showed that the PQN method of Chapter 3 outperforms this projected gradient method at solving the Lagrangian dual, and we considered using the PQN method to solve the Lagrangian dual that arises when we penalize the ℓ_2 norm of the groups.

The methods in Chapter 3 can also be used to encourage blockwise-sparsity in IGM models, for the ℓ_2 or ℓ_∞ norm of the blocks. Further, we can also use them to encourage blockwise-sparsity in general pairwise log-linear models. In this case, we simply define each group to be all edge parameters associated with all edges in the block.

5.7 Conditional Random Fields

Thus far, we have considered building a probabilistic model of all variables present in a data set. However, in many cases we might be interested in predicting the values of some variables (the targets) given the others (the features). This is similar to the regression and classification tasks we discuss in (1.1), but here we consider the generalization where we have *more than one target variable*. Further, the target variables may be *dependent*, even after after conditioning on the features. Analogous to the regression case, we use $\mathbf{x}$ to denote the features and we use $\mathbf{y}$ to denote the target variables. One way to address this problem is to model $p(\mathbf{y}, \mathbf{x})$ with an undirected model, and then use the conditional distribution $p(\mathbf{y}^m|\mathbf{x}^m)$ to answer conditional queries about instance m. However, in cases where the features are very complicated, it may be very difficult to build a good model of $p(\mathbf{y}, \mathbf{x})$.

For this multiple-target scenario, Lafferty et al. [2001] introduced conditional random fields (CRFs). In CRFs, we fit an undirected graphical model by optimizing the conditional likelihood $p(\mathbf{y}^m|\mathbf{x}^m)$ over all m training examples (this is typically referred to as a *discriminative* model). In the case of log-linear models, this is a natural generalization of logistic regression to the multi-target scenario (while for GGMs it is a natural generalization of least-squares). The advantage of optimizing the conditional likelihood instead of the likelihood is that we treat the variables $\mathbf{x}$ as fixed, instead of addressing the potentially difficult task of building a model of them. Liang and Jordan [2008] show that, if the model is misspecified (as is typically the case when dealing with real data), that optimizing $p(\mathbf{y}|\mathbf{x})$ is asymptotically more efficient both in terms of parameter estimation and generalization error than optimizing $p(\mathbf{y}, \mathbf{x})$.

In this work, we consider CRFs with a log-linear parameterization. For example, for a three-state node i we use node potentials of the form

$$\log \phi_i(\cdot, \mathbf{x}_i^m, \mathbf{b}_i, \mathbf{v}_i) = \begin{bmatrix} b_{i,1} + \mathbf{v}_{i1}^T \mathbf{x}_i^m \\ b_{i,2} + \mathbf{v}_{i2}^T \mathbf{x}_i^m \\ b_{i,3} + \mathbf{v}_{i3}^T \mathbf{x}_i^m \end{bmatrix},$$

where each node has its own set of bias parameters $\mathbf{b}_i$, its own set of (vector-valued) feature weights $\mathbf{v}_i$, and its own set of features $\mathbf{x}_i^m$ for instance m (some of these features may be shared between nodes). Typically, we fix the value of $b_{i,j}$ to zero for one of the states j, and we may also fix the vector $\mathbf{v}_{i,j}$ to the zero vector for one of the states. For full edge potentials on the edge between two three-state nodes i and j, we use

$$\log \phi_{ij}(\cdot, \cdot, \mathbf{x}_{ij}^m, \mathbf{w}_{ij}, \mathbf{v}_{ij}) = \begin{bmatrix} w_{ij11} + \mathbf{v}_{ij11}^T \mathbf{x}_{ij}^m & w_{ij12} + \mathbf{v}_{ij12}^T \mathbf{x}_{ij}^m & w_{ij13} + \mathbf{v}_{ij13}^T \mathbf{x}_{ij}^m \\ w_{ij21} + \mathbf{v}_{ij21}^T \mathbf{x}_{ij}^m & w_{ij22} + \mathbf{v}_{ij22}^T \mathbf{x}_{ij}^m & w_{ij23} + \mathbf{v}_{ij23}^T \mathbf{x}_{ij}^m \\ w_{ij31} + \mathbf{v}_{ij31}^T \mathbf{x}_{ij}^m & w_{ij32} + \mathbf{v}_{ij32}^T \mathbf{x}_{ij}^m & w_{ij33} + \mathbf{v}_{ij33}^T \mathbf{x}_{ij}^m \end{bmatrix},$$

where we note that each edge has its own set of weights $\mathbf{w}_{ij}$, its own set of (vector-valued) feature weights $\mathbf{v}_{ij}$, and its own set of edge features $\mathbf{x}_{ij}^m$ for instance m. We can fix the values of some of these weights to zero if we want a restricted class of potentials like the Ising or gIsing potentials. However, note that even in the Ising case, each edge will have multiple parameters. We can write the negative log-likelihood function with these potentials as

$$-\sum_{m=1}^{n}[\sum_{i=1}^{p}[\log \phi_i(y_i^m, \mathbf{x}_i^m, \mathbf{b}_i, \mathbf{v}_i) + \sum_{j=i+1}^{p} \log \phi_{ij}(y_i^m, y_j^m, \mathbf{x}_{ij}^m, \mathbf{w}_{ij}, \mathbf{v}_{ij})] + \log Z(\mathbf{w}, \mathbf{b}, \mathbf{v}, \mathbf{x}^m)],$$

where we have used $\mathbf{v}$ to refer to all node and edge feature weights. As in the unconditional case, this function is jointly convex in all of its parameters. However, since the normalizing constant for each training example is a function of $\mathbf{x}^m$, we now have a normalizing constant for each training instance m. This makes parameter estimation in the conditional case much more expensive.

While there has been some work towards discriminative structure learning in the context of Bayesian network classifiers [see Schmidt et al., 2008, Table 1], all previous work on CRFs has assumed that the graphical structure is known. Further, the high cost of evaluating the likelihood even for a fixed structure makes search-based methods unappealing. Thus, to apply a CRF model to a data set with unknown structure, in [Schmidt et al., 2008] we considered using group ℓ_1-regularization. More precisely, we used ℓ_2-regularization of the node feature weights and group ℓ_1-regularization of all edge weights corresponding to the same edge to learn a sparse regularized CRF by solving

$$\min_{\mathbf{w},\mathbf{b},\mathbf{v}} -\sum_{m=1}^{n} p(\mathbf{y}^m|\mathbf{x}^m,\mathbf{b},\mathbf{w},\mathbf{v}) + \lambda_1 \sum_{i=1}^{p} ||\mathbf{v}_i||_2^2 + \lambda_2 \sum_{i=1}^{p}\sum_{j=i+1}^{p} ||[\mathbf{w}_{ij} \;\; \mathbf{v}_{ij}]^T||_p, \tag{5.6}$$

Note that solving this problem is not the same as the computationally more efficient approach of first learning the structure of $\mathbf{y}$ as an unconditional log-linear model, and subsequently using this as the structure of the CRF. Our experiments indicate that this latter strategy under-performs using (5.6) to simultaneously and conditionally learn both structure and parameters.

5.7.1 Associative Conditional Random Fields

In all models up to this point we have considered sparsity as a rough approximation of the treewidth of the graph. This is because adding an edge will never decrease the treewidth of a graph, so models with fewer edges may have lower treewidths and thus allow efficient calculations with the model. In many applications of CRFs, the calculation that we are often most interested in is finding the *conditional optimal decoding*. That is, given the covariates $\mathbf{x}$ we would like to find the assignment of labels $\mathbf{y}^*$ with highest probability under the model:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}). \tag{5.7}$$

Although this problem is NP-hard in general, for the special of case of binary variables with *sub-modular* edge potentials it is possible to solve this problem in polynomial time [Kolmogorov and Zabih, 2002]. The sub-modularity condition is equivalent to the requirement that, for each edge, the log-potentials for assignments where the two variables take the same state are greater than the log-potentials for assignments where the variables have different states:

$$\log\phi_{ij}(1,1) + \log\phi_{ij}(2,2) \geq \log\phi_{ij}(1,2) + \log\phi_{ij}(2,1), \forall_{ij}. \tag{5.8}$$

We call a CRF satisfying this condition an *associative* CRF, analogous to the associative max-margin Markov networks examined in [Taskar et al., 2004]. Note that satisfying this condition allows us to perform optimal decoding in polynomial time as a minimum graph-cut problem, independent of the treewidth of the graph. Thus, enforcing that (5.8) is true for all edges and all possible values of the features $\mathbf{x}$ ensures that we can efficiently solve (5.7). In [Cobzas and Schmidt, 2009] we consider two simple conditions that are sufficient to ensure that the estimated parameters in a binary CRF with Ising edge potentials (and a fixed structure) satisfy (5.8). First, we require that all

features $\mathbf{x}$ are non-negative. Second, during parameter estimation we constrain all edge parameters to also be non-negative. These conditions ensure that $\log \phi_{ij}(1,1) \geq 0$ and $\log \phi_{ij}(2,2) \geq 0$, while since we use Ising edge potentials we have that $\log \phi_{ij}(1,2) = 0$ and $\log \phi_{ij}(2,1) = 0$. Adding these constraints yields a bound-constrained optimization problem that was solved with the two-metric projection algorithm discussed in Section 2.2.3.

We can use ℓ_1-regularization to extend this prior work to learn the graph structure while still constraining the model to be associative. If we use ℓ_1-regularization of the edge weights, then the problem reduces to optimizing a differentiable function with ℓ_1-regularization over the non-negative orthant. As discussed in 2.1.2, applying ℓ_1-regularization with an orthant constraint can be written as a bound-constrained smooth optimization problem. Thus, we can estimate the parameters of an associative CRF with ℓ_1-regularization using the two-metric projection algorithm discussed in Section 2.2.3. If we use group ℓ_1-regularization of the edge parameters to encourage a sparse graph structure, then computing the projection (or soft-threshold) subject to non-negativity constraints is straightforward; we set to zero all negative elements before computing the projection (or soft-threshold) for the remaining elements [van den Berg, 2010]. This allows us to apply the methods of Chapter 3 to solve the bound-constrained problem.

5.8 Experiments

We first examined two small real data sets where we could compare the effects of different regularization and edge potential types with the exact objective (§5.8.1), and then with approximate objectives (§5.8.2). We then compared the methods on some larger data sets using the pseudo-likelihood approximation (§5.8.3). We then looked at blockwise sparse models of a real data set (§5.8.4), and finally compared different ways to train CRF models on synthetic and real data (§5.8.5).

5.8.1 Edge Potentials and Regularization Types

We fist sought to assess the effects on prediction performance of different choices of regularization and edge potential type. To do this we used the two small data (*cyto* and *awma*) from Section 1.7, where the number of nodes (and states) is sufficiently small that we can evaluate the objective function exactly even with a densely connected graph (thus removing the use of an approximate objective function as a potential confounding factor). On these data sets, we tested the three edge potentials from Section 1.5:

- *Ising*: Here we have one parameter on each edge, giving the potential of the two nodes taking the same state. In the binary case this yields the IGM model of Section 1.4.
- *gIsing*: Here we have k parameters on each edge, giving the potential of the two nodes taking the same state for each state.
- *full*: Here we have a matrix of k^2 parameters on each edge, giving the potential for all k^2 combinations of the states.

We compared the following regularization strategies:

- *Tree*: We compute the maximum likelihood tree structure, then fit its parameters using ℓ_2-regularization.

- L_2: We fit the fully connected structure with ℓ_2-regularization. This does not yield a sparse structure, but may still perform well at prediction.
- L_1: We fit the fully connected structure with ℓ_1-regularization. This encourages sparsity in the edge parameters but does not directly encourage graphical sparsity.
- L_{12}: We fit the fully connected structure with group ℓ_1-regularization of the ℓ_2 norms of the edge parameters. This encourages graphical sparsity.
- $L_{1\infty}$: We fit the fully connected structure with group ℓ_1-regularization of the ℓ_∞ norms of the edge parameters. This encourages graphical sparsity and also encourages elements of the same edge potential to have the same magnitude.
- $L_{1\sigma}$: We fit the fully connected structure with group ℓ_1-regularization of the nuclear norms of the edge parameters. This encourages graphical sparsity and also encourages the edge potential matrices to have low rank.

In our experiments, we trained on one third of the data, evaluated the likelihood of a separate third of the data to estimate λ, and used the final third of the data to evaluate the model with the selected value of λ. We repeated this set-up with 10 different partitions of the data to estimate the variability of the results. We note that the particular split of the data into training/validation/testing is a confounding factor that affects of the performances of the method. Thus, in addition to computing the test set negative log-likelihood of the methods on each trial, we also computed a relative test set negative log-likelihood where we scaled the values to lie in the range $[0, 1]$ for each split (the best method on each data split is assigned a value of 0, and the worst method is assigned a value of 1). This latter score removes the particular data split as a potential confounding factor, giving a measure of relative performance across the different splits. For each model we tested $\lambda = 2^r$, where r was decreased from 10 down to -7 in increments of 0.25 (and we used warm-starting to solve these related optimization problems in order from the largest to the smallest value of λ). For these experiments, we added a weak ℓ_2-regularizer (with $\lambda = 10^{-4}$) to all models. This only has a small effect on the estimated parameters, but makes the objective strictly convex and removes the possibility that an observed difference between methods is due to the particular global optima found by the methods.

In Figure 5.1 we the compare the various methods on the *cyto* data in terms of the absolute score (left), and we compare the most effective methods in terms of relative score (right)[37]. Several trends are obvious. First, the Ising potentials do substantially worse than the gIsing and full potentials. With *Ising* potentials even the L_2 and L_1 methods that consider all possible edges do substantially worse than using the simple *tree* model with the slightly more general *gIsing* potentials. The *full* potentials also do better than the *gIsing* potentials for a given regularization type, but the difference in this case is not as dramatic. Another trend we see is that the regularization methods dominate the *tree* method, for a given edge potential type. However, we see no differences between different regularization types (for a given edge potential type) in terms of the absolute score. In terms of the relative score, we see that the sparse regularization methods tended to outperform using ℓ_2-regularization, but the different sparse regularization methods had similar performance.

[37]Not all regularization types are included for all edge types in this plot. This is because the group ℓ_1-regularization methods are equivalent to ℓ_1-regularization for *Ising* potentials, while group ℓ_1-regularization with the nuclear norm is equivalent to ℓ_1-regularization for *gIsing* potentials

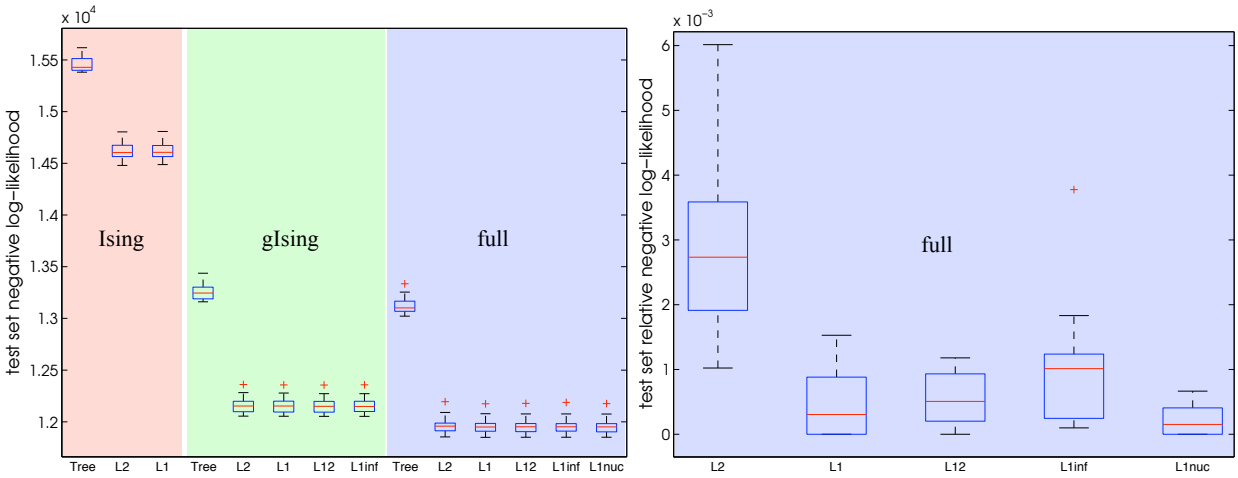

Figure 5.1: Test set negative log-likelihood (left) and relative negative log-likelihood (right) on the *cyto* data using different regularization and edge potential types.

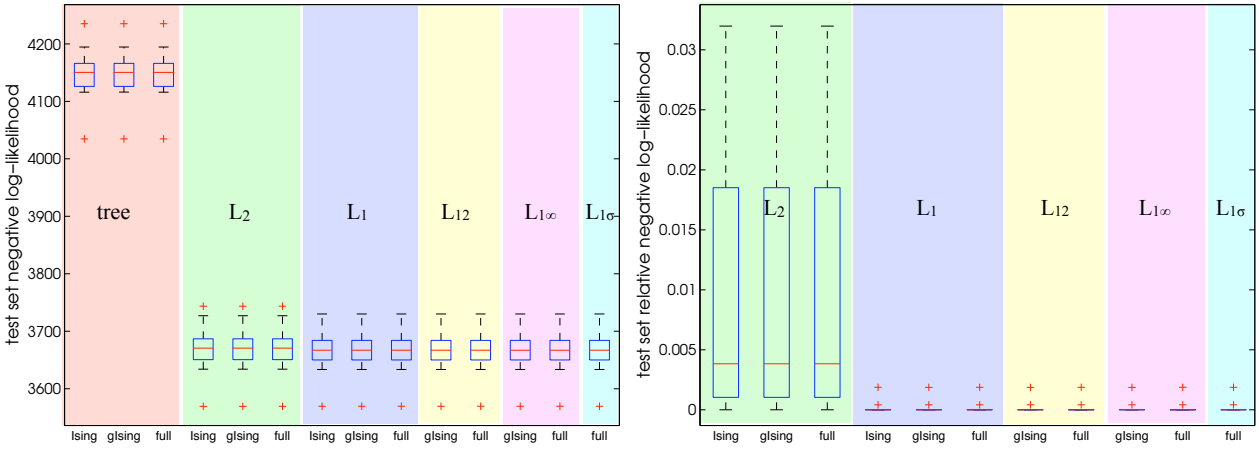

Figure 5.2: Test set negative log-likelihood (left) and relative negative log-likelihood (right) on the *awma* data using different regularization and edge potential types.

In Figure 5.2 we compare the various methods on the *awma* data in terms of the absolute score (left), and we compare the non-tree methods in terms of relative score (right). On this data set, we again see that dense regularization methods dominate the *tree* methods. However, on this binary data set we see that the choice of edge potentials makes no difference. This makes sense because using multi-parameter edge potentials does not increase the expressive power of the model for binary data. Further, we see no difference in absolute score between the various regularization methods, though we again see that the different sparse regularization methods tended to outperform using ℓ_2-regularization in terms of the relative score.

5.8.2 Approximate Objectives

We next sought to compare the performance of different approximations to the objective function on these two small data sets. We compared the following objective functions from Section 5.3:

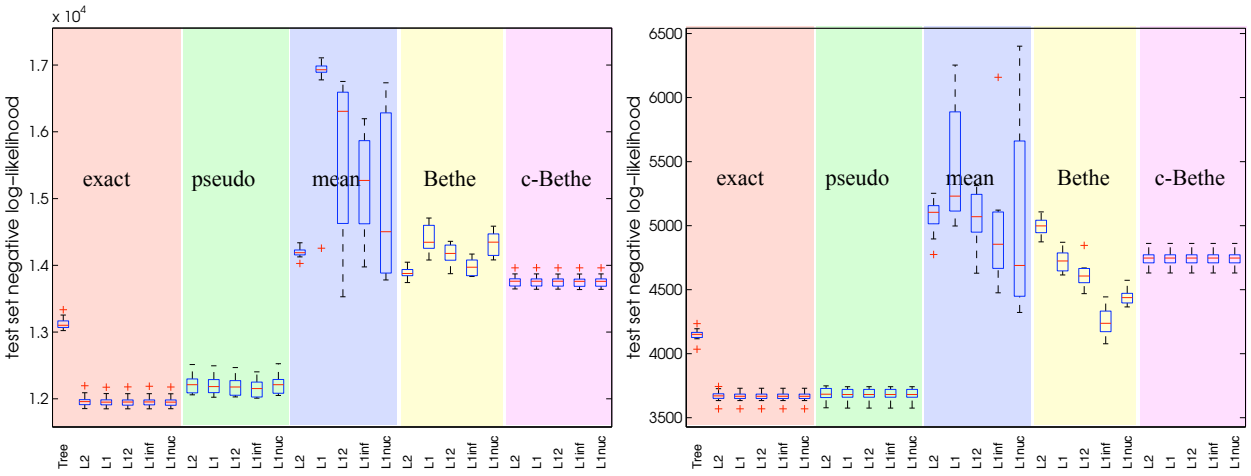

Figure 5.3: Test set negative log-likelihood on the *cyto* (left) and *awma* (right) data sets using different approximate objective functions.

- *exact*: The exact (convex) objective.
- *pseudo*: The (convex) pseudo-likelihood approximation.
- *mean*: The (non-convex) mean-field variational approximation.
- *Bethe*: The (non-convex) Bethe variational approximation.
- *c-Bethe*: The convexified Bethe variational approximation.

Our experimental set-up was identical to the previous sub-section, except that we trained the different regularization methods under these different approximations. Note that we still compute the exact validation and test score.

In Figure 5.3 we plot the performance of the different regularization methods under different approximate objective functions (using full potentials). In this plot, we see that the objective function that leads to the best performance is (unsurprisingly) the exact objective function. Among the approximate objectives, the pseudo-likelihood approximation proved to give results that are much closer to the exact objective function than the variational approximations. Indeed, it was surprising that in almost every case the variational approximations proved to give worse parameters than the optimal tree (the only exception to this was using group ℓ_1-regularization with the ℓ_∞ norm of the groups under the Bethe approximation). In this Figure, we also see that the performance of the three convex objective functions changed little across the regularization types, but that the two non-convex approximations were more erratic. It is somewhat surprising that the performance of the Bethe approximation changes substantially under different choices of the regularization norm, but that the performance for a given norm was consistent across trials (this is especially surprising on the binary *awma* data set). In contrast to the Bethe approximation, for most choices of regularization the mean field method was erratic, except when using ℓ_2-regularization, and when using ℓ_1-regularization in the *cyto* data (where it had consistent but poor performance).

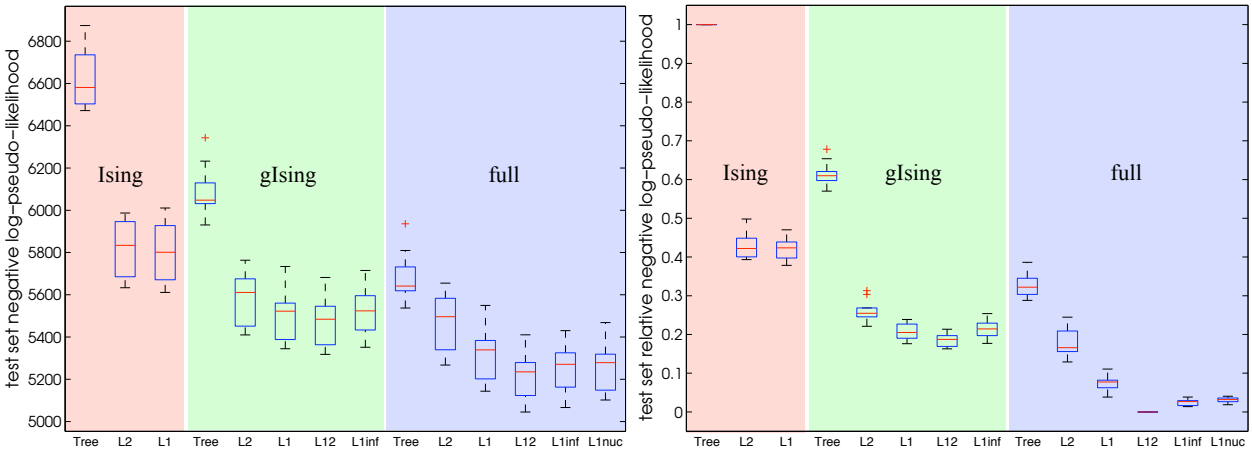

Figure 5.4: Test set negative log-pseudo-likelihood (left) and relative negative log-pseudo-likelihood (right) on the *awma5* data using different regularization and edge potential types.

5.8.3 Larger Real Data

We next sought to compare the different regularization strategies and edge potential types on larger data sets (where evaluating the exact objective function is intractable in general). Specifically, we considered the four larger (non-binary) discrete data sets from Section 1.7. On these data sets we concentrated on the pseudo-likelihood objective function for training and testing, but otherwise we used the same experimental set-up.

In Figure 5.4 we plot the absolute and relative results of different regularization strategies and edge potential types on the *awma5* data set. Unlike the binary version of the data set, for the full five state version of the data set we see differences between the different regularization and edge potential types. In particular, we see that all models achieve the best performance with full potentials, while all models achieve the worst performance with Ising potentials. Further, we see that for a given edge potential type that the sparse regularization methods outperform ℓ_2-regularization (for *gIsing* and *full* potentials), while for a fixed edge potential type ℓ_2-regularization outperforms the *tree* model. In this experiment, we see significant differences in the relative scores between the different sparse regularization methods when using full potentials. In particular, group ℓ_1-regularization with the ℓ_2 norm achieved the best value across all training set splits, following by the other group ℓ_1-regularization strategies, and regular ℓ_1-regularization gave the worst performance among the sparse regularizers.

We plot the results on the four-state *traffic* and *temperature* data sets in Figure 5.5. Similar to the three-state *cyto* and five-state *awma5* data sets, we again observe that utilizing more expressive potentials leads to better performance. Further, similar to the *awma5* data set, using group ℓ_1-regularization with the ℓ_2 norm (and full potentials) achieved the best performance across all 10 trials for both of these data sets.

Finally, we plot the results on the four-state *usps4* and eight-state *usps8* data sets in Figure 5.6. On these data sets, the best performance across all trials was achieved by group ℓ_1-regularization with the nuclear norm. Further, on the *usps8* data set we even see a significant advantage in the absolute score over all other methods when using group ℓ_1-regularization with the nuclear norm. This makes intuitive sense, since we would expect the edge weight matrices resulting from

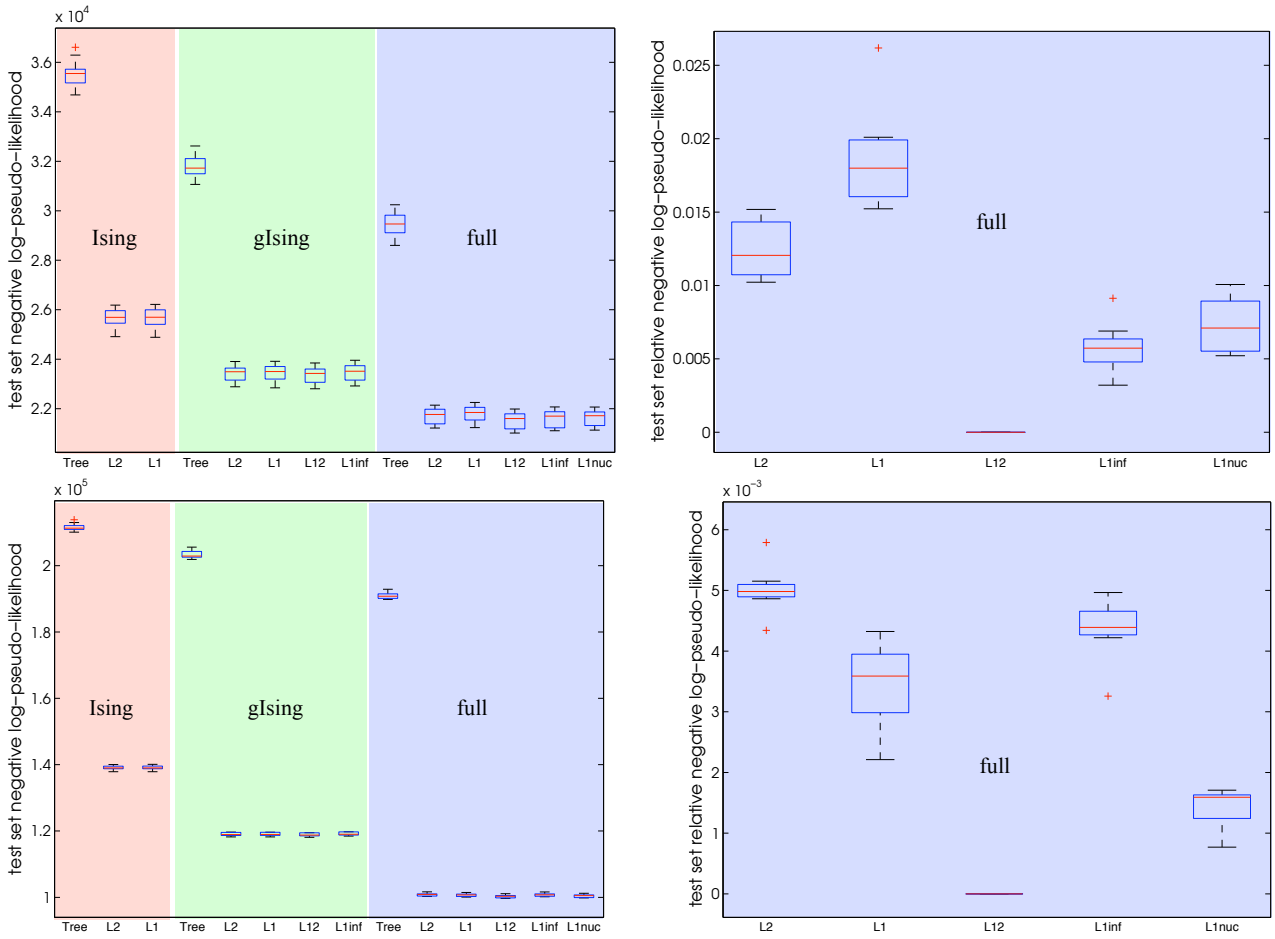

Figure 5.5: Test set negative log-pseudo-likelihood (left) and relative negative log-pseudo-likelihood (right) on the *traffic* (top) and *temperature* (bottom) data using different regularization and edge potential types.

the discretized states to be highly structured and well-approximated by a low rank matrix. In contrast, group ℓ_1-regularization with the ℓ_2 norm outperforms all methods except the nuclear norm method on the *usps4* data, but does worse than using regular ℓ_1-regularization on the *usps8* data. We believe this is because no structure is assumed by using ℓ_2-regularization of the edge weight matrices, making it more difficult to estimate the 64 parameters associated with each edge.

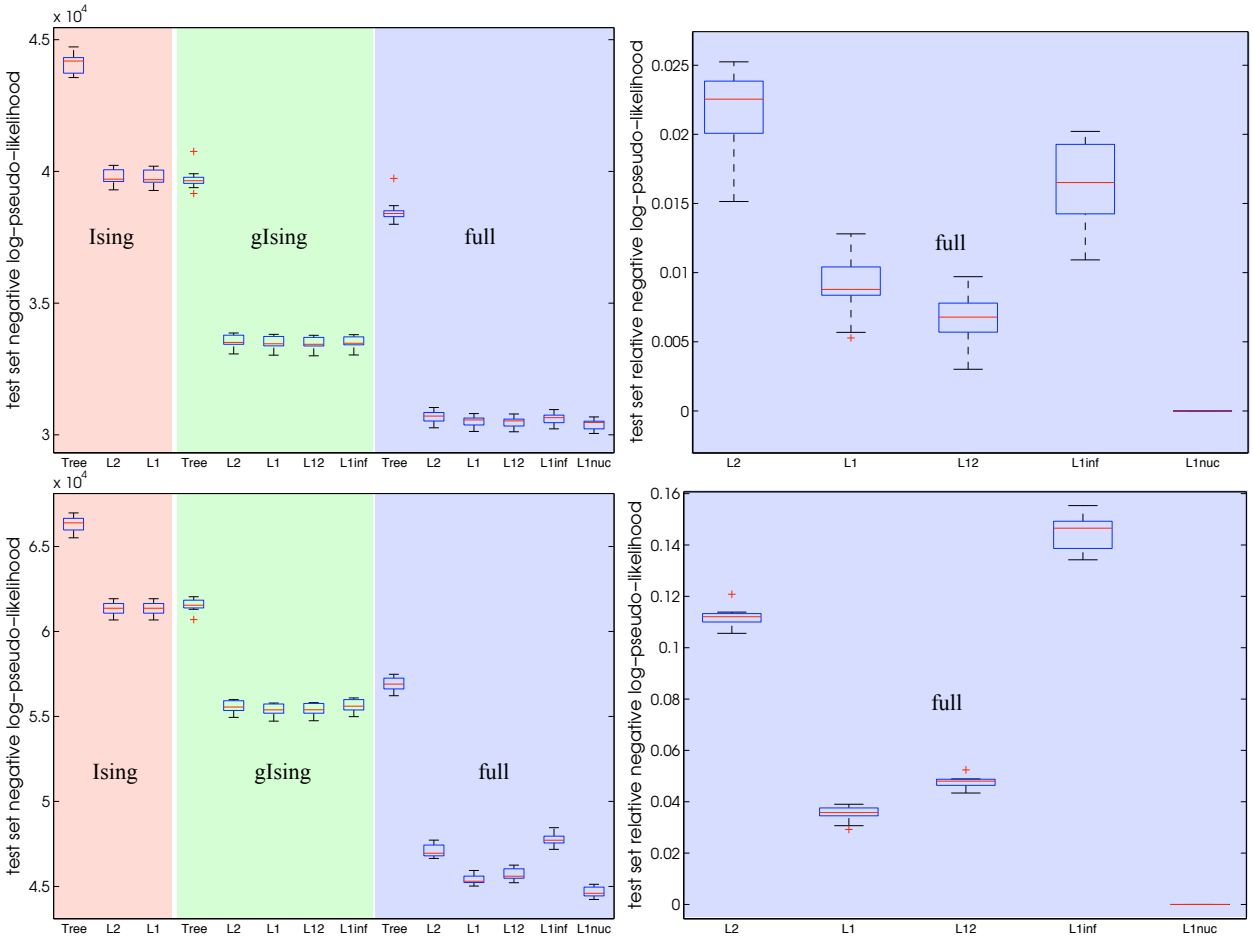

Figure 5.6: Test set negative log-pseudo-likelihood (left) and relative negative log-pseudo-likelihood (right) on the *usps4* (top) and *usps8* (bottom) data using different regularization and edge potential types.

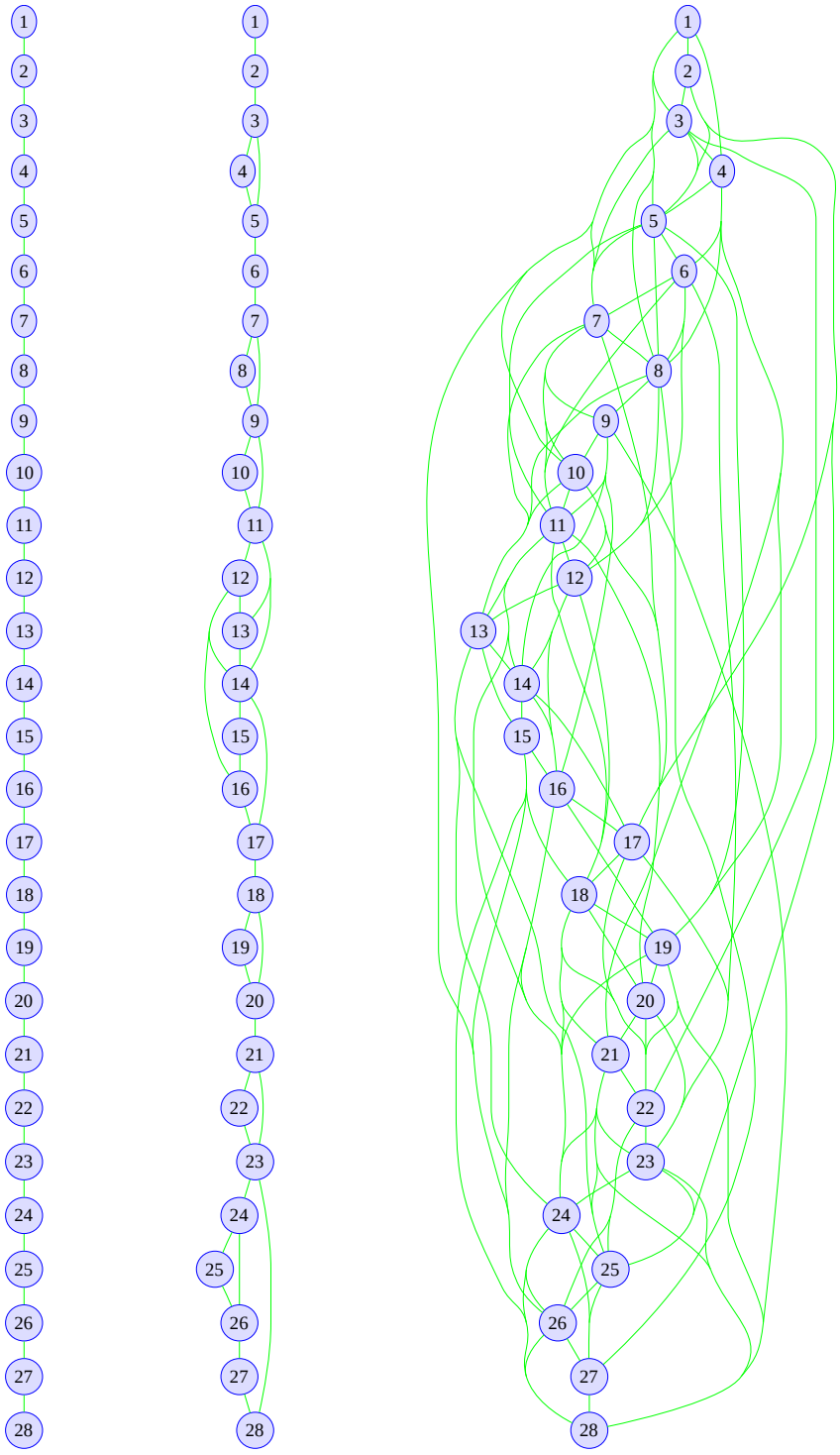

Figure 5.7: Structures estimated on the *rain* data set with group ℓ_1-regularization for different regularization parameter values. From left to right, $\lambda = 256, 128, 64$ (for $\lambda = 512$ the graph is disconnected).

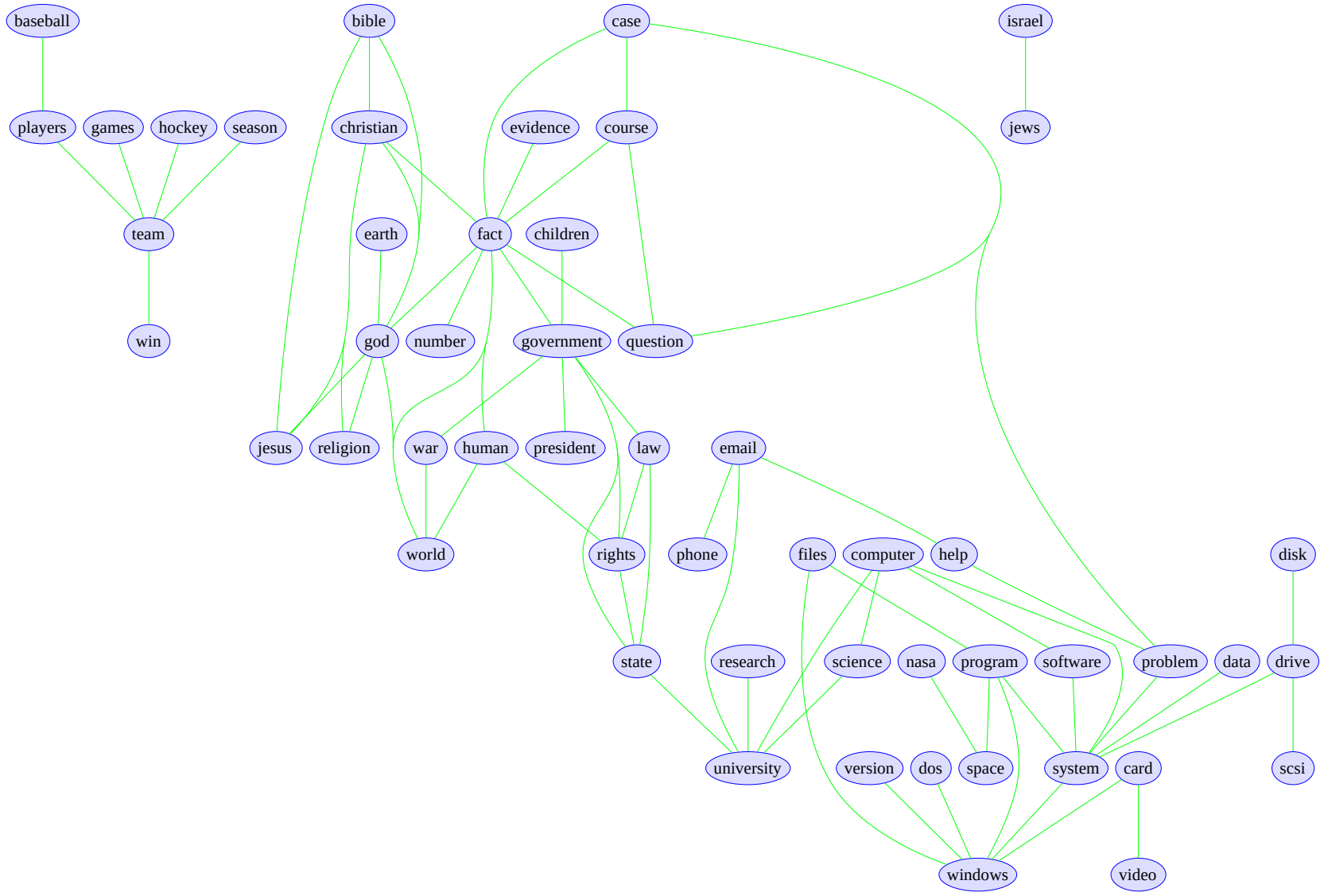

Figure 5.8: Structure estimated on the *news* data set with group ℓ_1-regularization ($\lambda = 512$, isolated nodes are not plotted).

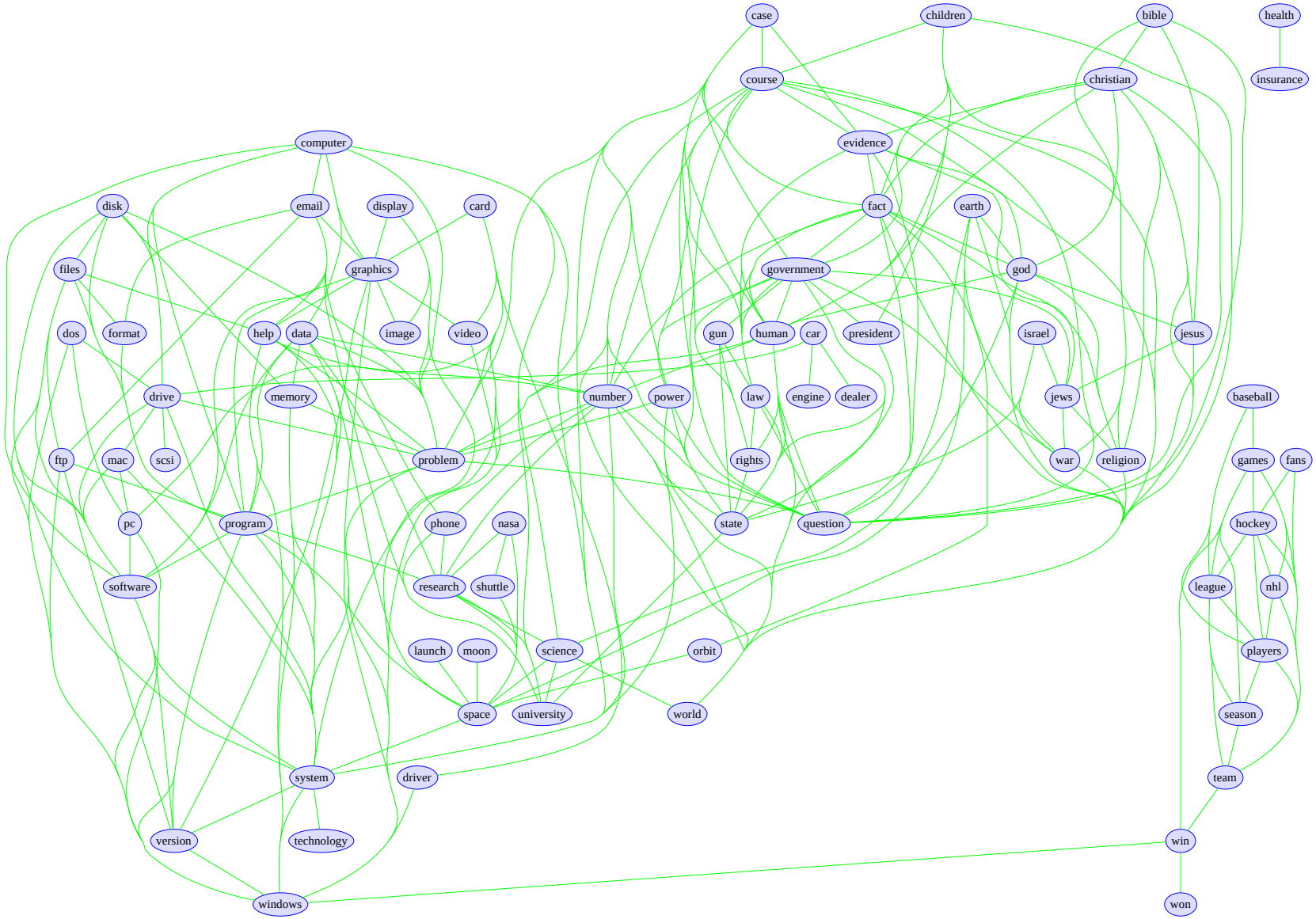

Figure 5.9: Structure estimated on the *news* data set with group ℓ_1-regularization ($\lambda = 256$, isolated nodes are not plotted).

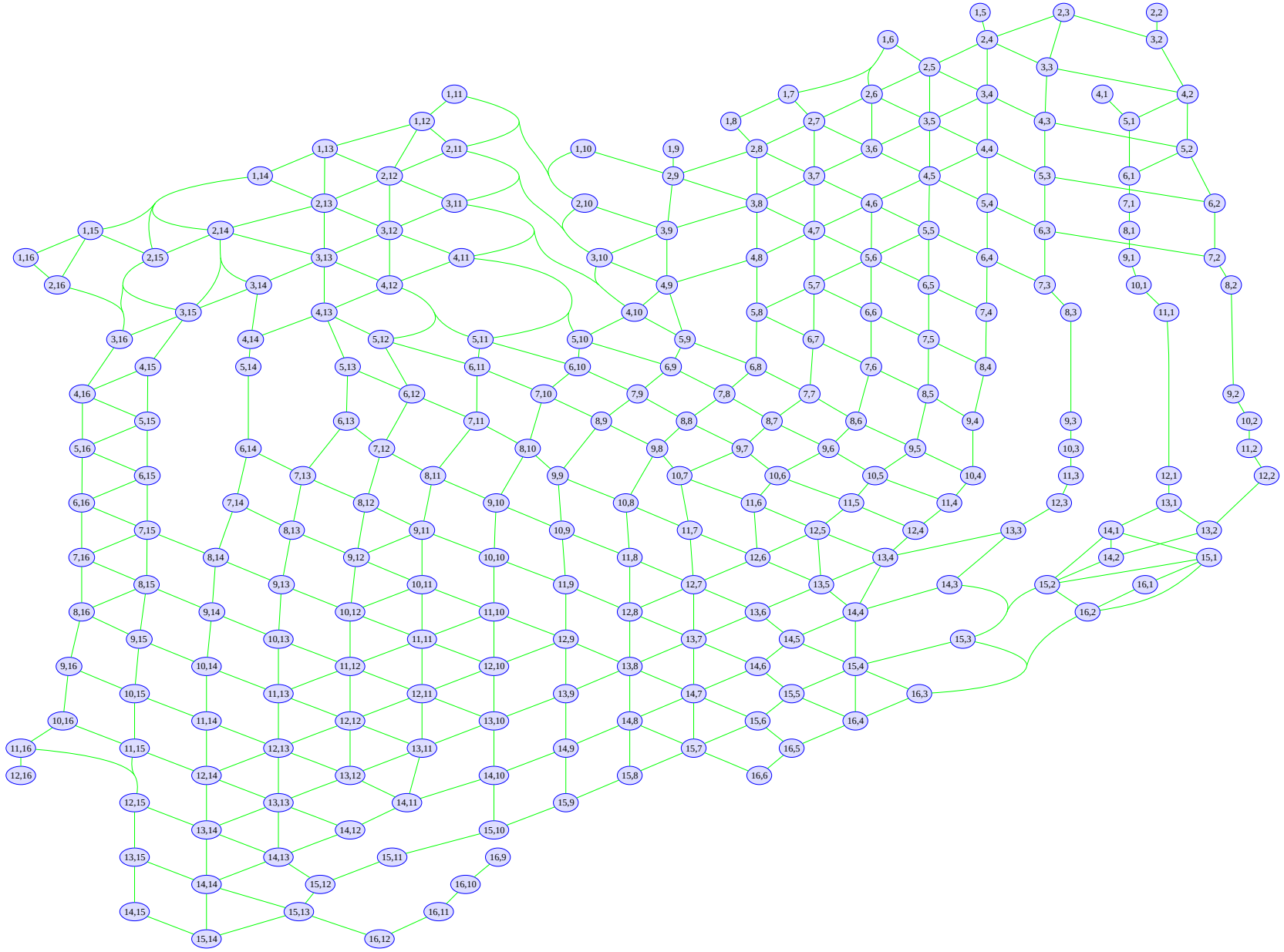

Figure 5.10: Structure estimated on the *usps* data set with group ℓ_1-regularization ($\lambda = 4096$).

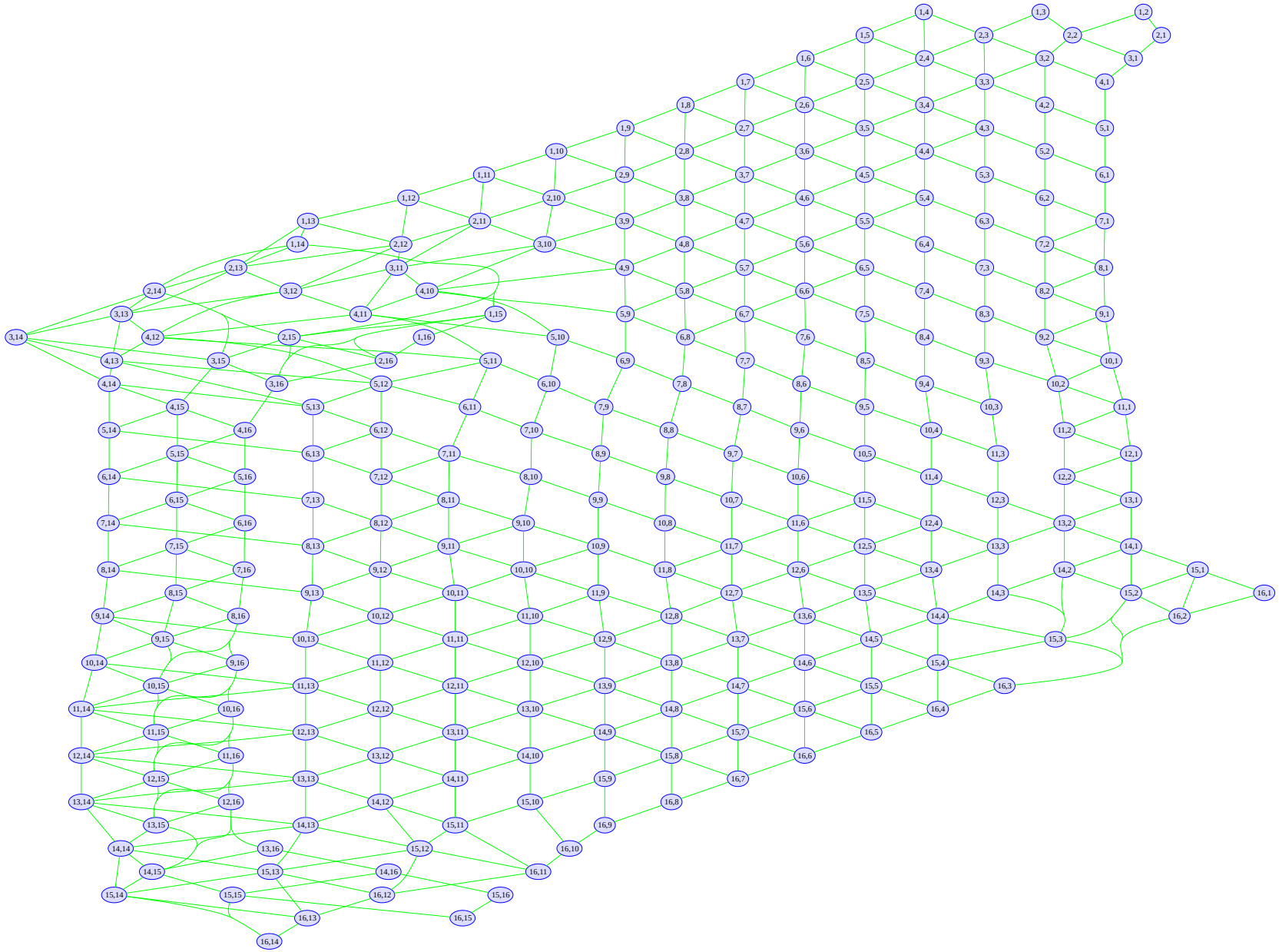

Figure 5.11: Structure estimated on the *usps* data set with group ℓ_1-regularization ($\lambda = 2048$).

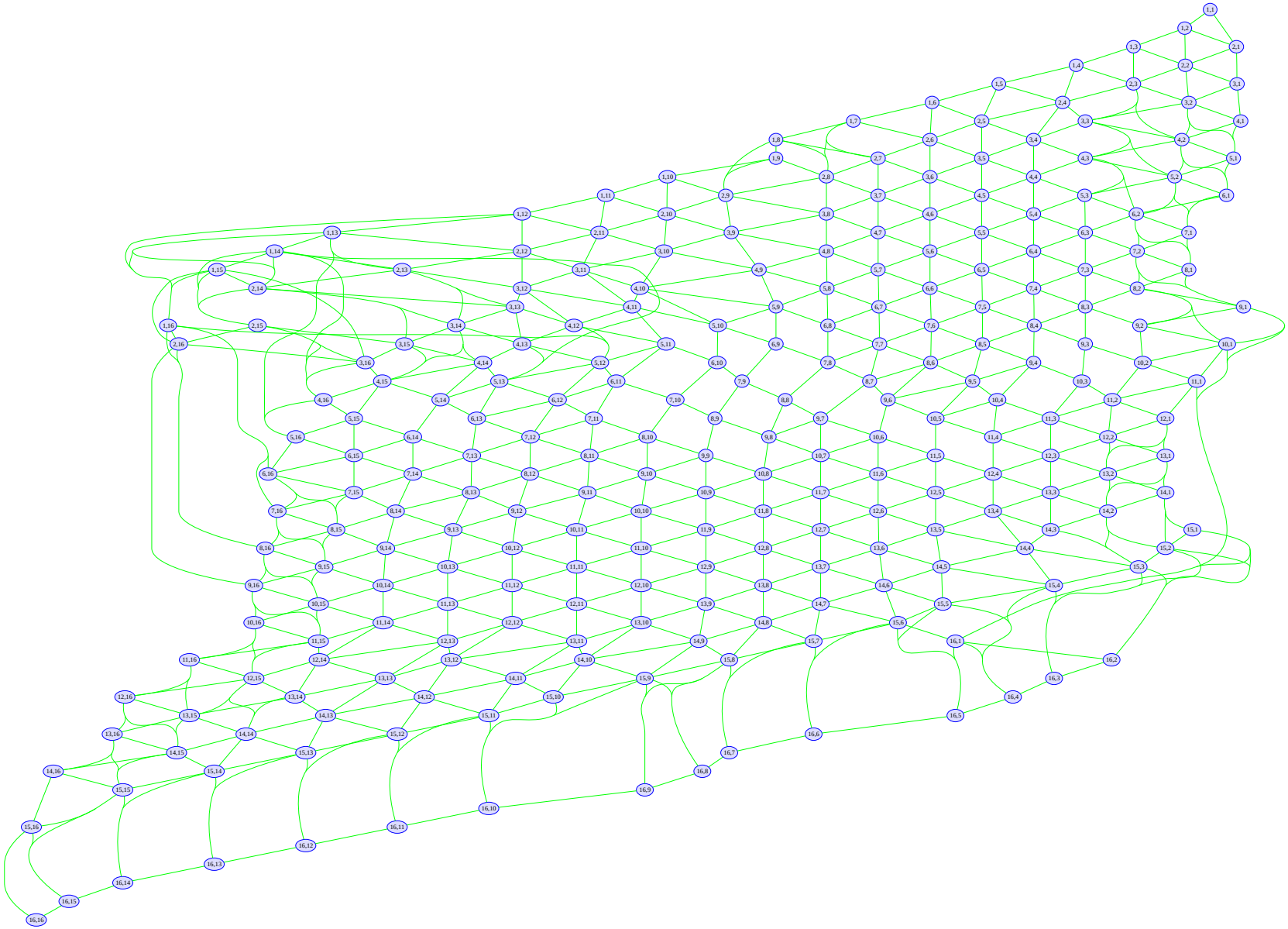

Figure 5.12: Structure estimated on the *usps* data set with group ℓ_1-regularization ($\lambda = 1024$).

We finally sought to assess whether the group ℓ_1-regularization method learns a reasonable graph structure. We fit a group ℓ_1-regularized undirected graphical model (with full potentials, the ℓ_2 group norm, and the pseudo-likelihood approximation) on the full *rain*, *news*, and *usps* data sets examined in the previous chapter. We tested integer powers of 2 for the regularization parameter, and examined the largest such values that produced non-empty graphs.

In Figure 5.7, we plot the structure estimated on the *rain* data with $\lambda = 256$, 128, and 64. With $\lambda = 256$ the model estimates a 28-node Markov chain, which (as discussed in the previous chapter) is a reasonable structure for this data set and is the optimal tree structure. As λ is decreased more edges are added, between temporally close nodes for $\lambda = 128$ and between more distant nodes for $\lambda = 64$. With $\lambda = 512$, the graph is disconnected.

In Figures 5.8 and 5.9, we plot the structure estimated on the *news* data set with λ set to 512 and 256, respectively. The graph with $\lambda = 512$ is very interpretable and intuitive, even though it is not a tree structure. The graph with $\lambda = 256$ is more dense and less interpretable, but the edges still tend to represent intuitive associations. With $\lambda = 128$, the graph was very dense and not particularly interpretable, while with $\lambda = 1024$ the graph only contained four edges: bible:god, christian:god, dos:windows, and god:jesus.

The most common application of pairwise undirected models (ignoring time-series data where there is no distinction in the graphical properties of directed and undirected models) is image processing, where a two-dimensional grid graph structure is typically assumed. Thus, for the *usps* data set we might expect the method to estimate a two-dimensional grid graph structure, where each node/pixel is connected to its four horizontal and vertical neighbors. We plot the structure estimated with $\lambda = 4096$, 2048, and 1024 for the *usps* data in Figures 5.10-5.12. Here, we see that (for large values of λ) the model learns structures that are close to two-dimensional grid models. Indeed, these structure are much closer to a grid structure than the three graph structures we examined for the *usps* data set in Chapter 4 (Figures 4.12-4.14). However, there are still some discrepancies between the structures in Figures 5.10-5.12 and a two-dimensional grid structure. The first discrepancy is that extra edges are present near the boundaries (and two of the corners in particular), with the number of extra edges increasing as λ decreases. This might be because the fewer neighboring pixels present at the boundaries means that it is important to not only look at neighboring pixel's values. The second discrepancy is that while in some parts of the image the graph forms a perfect grid structure where each pixel is connected to its four horizontal and vertical neighbors (around (7,9), for example), throughout most of the image the model also connects each pixel to its diagonal neighbors (i.e. the median number of neighbors for each node is 6). These extra edges are intuitive, since diagonal neighbors may contain additional information that is not present in the horizontal and vertical neighbors. With lower values of λ, edges between more distant nodes are added and the graphs become less interpretable.

5.8.4 Blockwise Sparsity

In [Schmidt et al., 2009b], we sought to test the performance of fitting blockwise-sparse GGMs to the *genes* data. In this experiment we sought to reproduce Figure 4 of [Duchi et al., 2008a], and to test the effect of using the ℓ_2 norm of the blocks instead of the ℓ_∞ norm used in this previous work. We first followed the same experimental set-up as [Duchi et al., 2008a], and performed 50 random train/test splits. In Figure 5.13 we give our version of Figure 4 from [Duchi et al., 2008a], augmented with the blockwise sparse model that penalizes the ℓ_2 norm of the blocks. This figure suggests that using the ℓ_2 norm of the blocks gives a further improvement over the existing

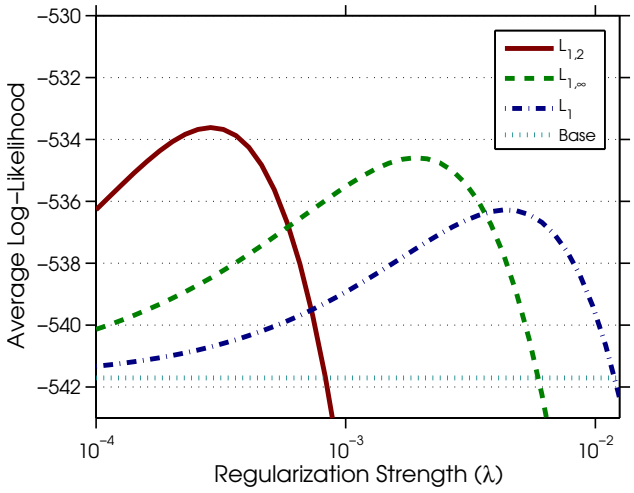

Figure 5.13: Average cross-validated log-likelihood against regularization strength under different blockwise-sparse regularization schemes applied to the regularized empirical covariance for the *genes* data [Schmidt et al., 2009b].

.

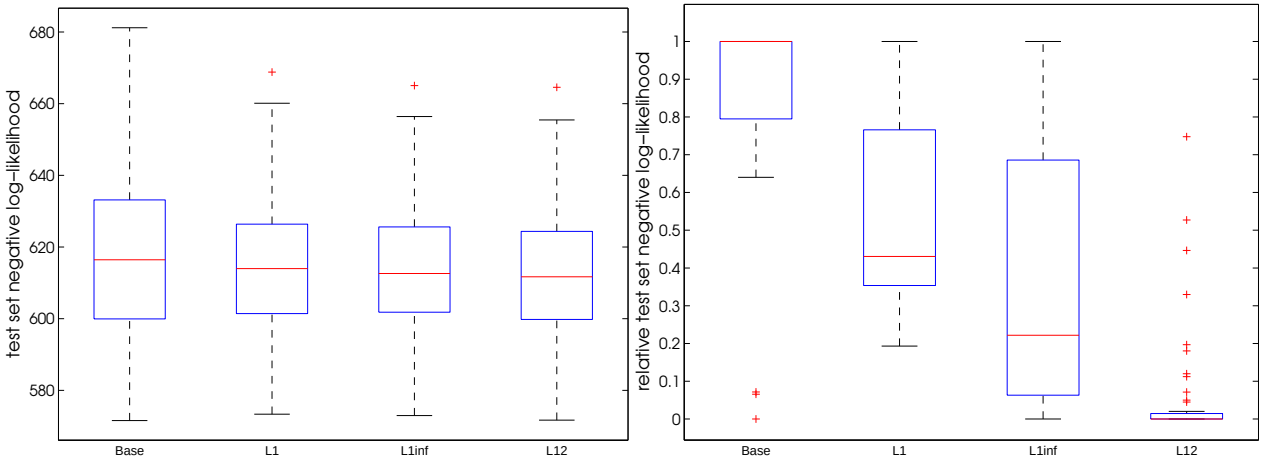

Figure 5.14: Test set negative log-likelihood (left) and relative negative log-likelihood (right) on the *genes* data using different regularization methods.

blockwise sparse model.

To assess the usefulness of the various models for prediction, we divided the data into equal-sized training/validation/testing sets, and measured the test set absolute and relative negative log-likelihoods. The level of Tikhonov regularization discussed in [Duchi et al., 2008a] was selected using the validation set likelihood on each training split. We plot the distribution of these values over 50 trials in Figure 5.14. Here we see that there is no difference in the performance of the methods in absolute score, but that (though very noisy due to the small number of training examples) the blockwise-sparse model that penalizes the ℓ_2 norms of the blocks may have an advantage over the other methods.

5.8.5 Conditional Random Fields

In [Schmidt et al., 2008], we experimentally compared an extensive variety of approaches to learning CRF models. Below we divide up these approaches into several groups:

- **Fixed Structure**: We learn the parameters of a CRF with a fixed structure. We considered an *Empty* structure (corresponding to an independent logistic regression model for each target), a *Chain* structure (the structure most commonly used in CRFs), a *Full* structure (assuming all edges are present), and the *True* structure. For the synthetic experiments, the *True* structure was set to the actual generating structure, while for the real data we used a structure constructed from expert knowledge.

- **Generative Acyclic**: We first learn the graph structure based on the labels alone, and then learn the parameters of a CRF with this fixed structure. We considered the generative models from [Qazi et al., 2007], namely finding the maximum likelihood *Tree* and using *DAG-Search*.

- **Generative** ℓ_1: Here we use (group) ℓ_1-regularization to learn a fixed structure based on the labels alone, and then learn the parameters of a CRF with this fixed structure. We considered using ℓ_1-regularization and group ℓ_1-regularization with the ℓ_2 and ℓ_∞ norms.

- **Discriminative** ℓ_1: Here we used simultaneous conditional estimation of the structure and parameters using (group) ℓ_1-regularization, where again we considered ℓ_1-regularization and group ℓ_1-regularization with the ℓ_2 and ℓ_∞ norms.

To compare methods and test the effects of both discriminative structure learning and approximate inference for training, we created a synthetic dataset from a small (10-node) binary CRF. We used 10 local features for each node (sampled from a standard Normal) plus a bias term. We chose the graph structure by including each possible edge with probability 0.5. Similarly, we sampled random node weights $\mathbf{v}_i \sim \mathcal{N}(0, \sqrt{2})$, and edge weights $\mathbf{w}_{ij} \sim U(-b, b)$, where $b \sim \mathcal{N}(0, \sqrt{2})$ for each edge (the results were similar under different sampling schemes). We drew 500 training samples and 1000 test samples from the exact distribution $p(\mathbf{y}|\mathbf{x})$.

In all models, we impose an ℓ_2 penalty on the node weights, and we also impose an ℓ_2 penalty on the edge weights for all models that do not use ℓ_1 regularization of the edge weights. For each of the models compared, the scale of these two regularization parameters is selected by cross-validation on the training set. In our experiments, we explored 10 different permutations of training and testing instances in order to quantify variation in the performance of the methods. For testing the quality of the models, we computed the classification error associated with the exact conditionals $p(y_i|\mathbf{x})$.

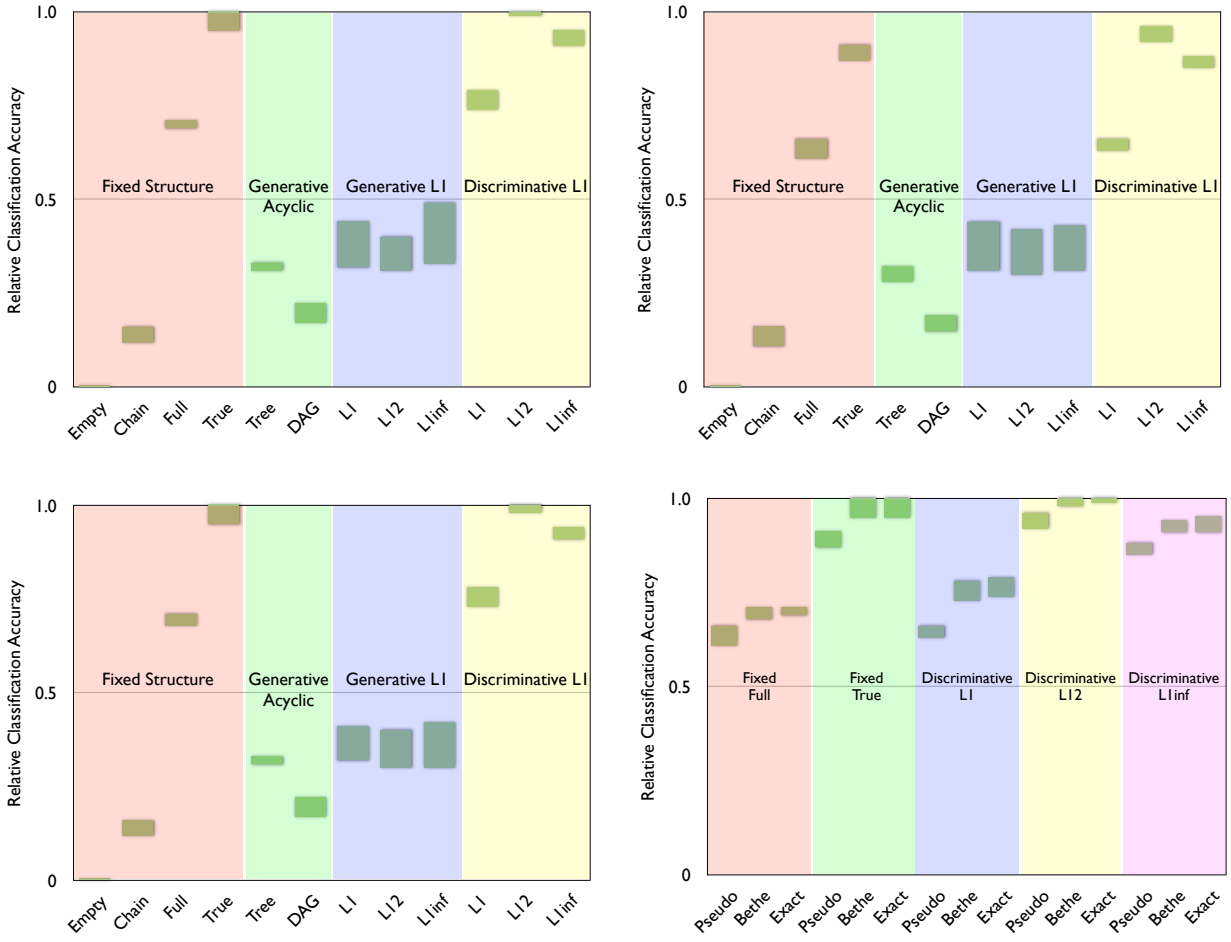

Figure 5.15: Interquartile range of relative test-set classification accuracy for different methods of training CRFs on synthetic data using the exact objective (top-left), pseudo-likelihood approximation (top-right), Bethe approximation (bottom-left), and selected methods under different approximations (bottom-right). Note that the empty graph, corresponding to logistic regression, always had a relative accuracy of zero.

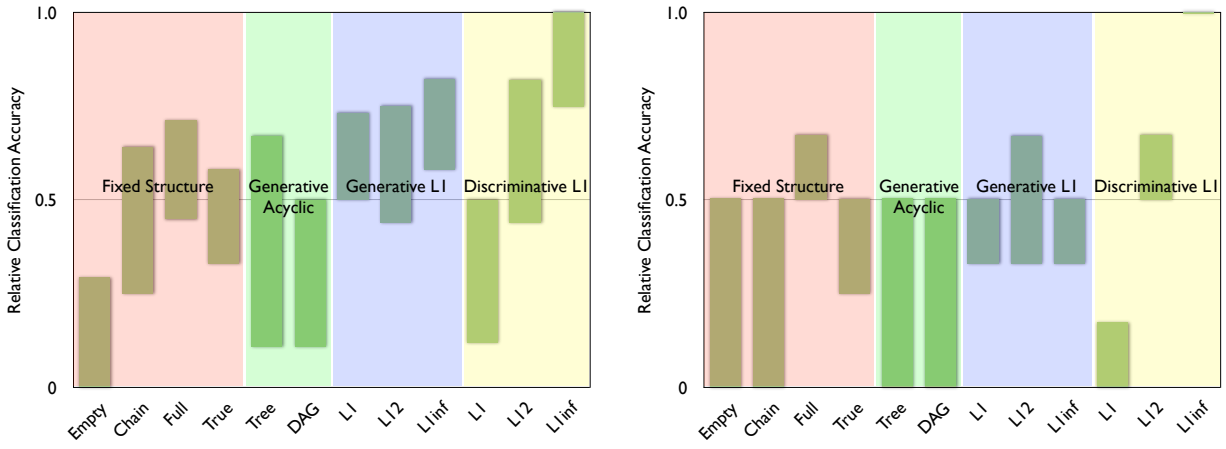

Figure 5.16: Interquartile range of relative test-set classification accuracy for different methods of training CRFs on the coronary heart disease data at the segment level (left) and heart level (right). Note that the discriminative structure learning method with group ℓ_1-regularization with the ℓ_∞ norm always has a relative accuracy of one on the heart-level classification task (rightmost column).

We compared learning with the exact objective, the (conditional) pseudo-likelihood objective, and the Bethe variational approximation.

In Figures 5.15, we show the relative classification accuracy of different methods on the test set for different objective functions (the best possible score is 1, and the worst is 0). Although not necessary for the synthetic data, we use this measure since the real data examined next is relatively small with a class imbalance, and even though the ranking of the methods is consistent across trials, the particular data split on a given trial represents a confounding factor that obscures the relative performance of the methods. We summarize this distribution in terms of its interquartile range (a measure of the width of the central 50% interval of the distribution); this is a more robust summary than the standard mean and standard deviation.

The results show several broad trends: (a) pseudo-likelihood and the Bethe approximation are almost as good as the exact likelihood (and the Bethe approximation is slightly better than pseudo-likelihood), (b) discriminatively learned structures outperform generatively learned structures, (c) any kind of structure is better than no structure at all, (d) in the generative case, group ℓ_1-regularization (under both norms) and regular ℓ_1-regularization are very similar (consistent with our earlier experiments in binary data), and (e) both group ℓ_1-regularization methods outperform ℓ_1-regularization in the discriminative case. Results on other synthetic data sets yield qualitatively similar conclusions, with two exceptions: (i) as we decrease the number of features the performance of group ℓ_1-regularization becomes more similar to regular ℓ_1-regularization, and (ii) on some data sets the Bethe approximation produced results that were much worse than the pseudo-likelihood approximation.

We next examined the *awma-c* classification problem. In this data set, we have 19 local image features for each node calculated from the tracked contours of the ventricle. Among these features we include local ejection fraction ratio, radial displacement, circumferential strain, velocity, thickness, thickening, timing, eigenmotion, curvature, and bending energy. We also have 15 global image features (that are ths same across nodes). For the node features we used the concatenation

of the 15 global features and the 19 local features for the node. For the edge features we used the 15 global features and the 38 features consisting of the concatenation of the local features for each node. We used 2/3 of the data for training and selecting the two regularization parameters, and 1/3 of the data for testing (across 10 different splits). We generated the *True* structure by adding edges between all nodes sharing a face in the heart diagram, constructed by expert cardiologists, from Qazi et al. [2007]. We trained various models using pseudo-likelihood and tested them using exact inference[38].

In Figure 5.16, we show the relative classification accuracy on the test set at the segment level and the heart level (the heart level decision is made by cardiologists by testing whether two or more segments are abnormal). We see that the discriminative model with group ℓ_1-regularization with the ℓ_∞ norms of the groups performs among the best at the segment level (achieving a median *absolute* classification accuracy of 0.92), and is typically the best method at the important heart-level prediction task (achieving a median absolute accuracy of 0.86 and the lowest error rate at this task in 9 out of the 10 trials). These encouraging results can also help less-experienced cardiologists improve their diagnostic accuracy; the agreement between less-experienced cardiologists and experts is often below 50% [Schmidt et al., 2008].

5.9 Similar Methods

In this chapter, we discuss using group ℓ_1-regularization for structure learning in pairwise undirected graphical models of discrete data that do not make the Ising assumption. However, it should be noted that similar extensions have been proposed prior and concurrently with this work. In this section, we highlight the differences between the prior (and concurrent) work and the work outlined in this chapter.

Models that do not make the Ising assumption were also explored in [Lee et al., 2006b, Dahinden et al., 2007]. In [Lee et al., 2006b], they use the non-convex Bethe approximation, and their experiments indicate that the method reaches different local optima with different optimization strategies. Further, they ignore that each edge can have multiple parameters and simply use the standard ℓ_1-regularization. However, they note that this does not directly encourage graphical sparsity and that graphical sparsity could be achieved with group ℓ_1-regularization. However, they do not provide a method to solve the resulting problem. In contrast, [Dahinden et al., 2007] ignore the computational infeasibility of evaluating the likelihood function but use group ℓ_1-regularization to directly encourage graphical sparsity. However, they use an optimization algorithm that may require many evaluations of the (generally intractable) likelihood, and do not present results on data with more than 5 variables. In both cases, they only consider using the ℓ_2 norm of the groups.

Our work is distinct from this prior work in several ways. First, we consider the convex pseudo-likelihood and convexified Bethe approximations to the likelihood. Using convex approximations means that the estimated parameters are not sensitive to initialization of the optimization procedure, or to the particular optimization strategy used. Second, we consider choices of the group norm other than the ℓ_2 norm. This includes the proposed group extension of the nuclear norm, which is novel (as far as we are aware). Our experiments indicate that in some cases other choices of the group norm give better results than the ℓ_2 norm. Third, we give a method for adding covariates to the model to yield the more powerful CRF models, while this is the first work to consider

[38] We also tested using the Bethe approximation for this task, but learning with this approximation typically lead to parameters where the message-passing algorithm would not converge and lead to poor results.

structure learning in CRFs. Finally, in Chapter 3 we outline a new optimization algorithm that is especially suited to solving the resulting convex optimization problems, taking into account the very large number of optimization variables, the high cost of evaluating the objective function, and the relatively simple form of the regularizer.

Using group ℓ_1-regularization to learn blockwise-sparse models was originally proposed in [Duchi et al., 2008a], where they use group ℓ_1-regularization in GGMs with the ℓ_∞ norm for blockwise-sparsity. The work presented here considers other choices of group norms, as well as blockwise-sparse discrete models. Further, the optimization algorithms outlined in Chapter 3 are also well-suited to solving this type of optimization problem, although the improvement in the group-sparse GGM case is not as dramatic as in the group-sparse discrete case.

5.10 Extensions

To conclude this chapter, below we list some extensions of the work presented here:

- **Composite likelihoods**: The maximum pseudo-likelihood approximation is asymptotically less efficient than the maximum likelihood estimator [Besag, 1977, Liang and Jordan, 2008]. A generalization of pseudo-likelihood approximations is the class of composite likelihoods [Lindsay, 1988], where we can consider using the conditional (or marginal) distributions of groups of variables rather than individiual variables. We would likely obtain better results by using a more powerful composite likelihood. For example, we could consider using a composite likelihood where we optimize the conditionals of all pairs of variables. This would only lead to a constant increase in computational complexity but might result in a much better approximation.

- **Other variational inference methods**: In this work we have considered some of the most common methods for variational inference, but it is straightforward to use other variational inference methods. For example, Banerjee et al. [2008] use ℓ_1-regularization to learn the structure of an (unconditional) undirected graphical model with binary states and Ising potentials that uses a log-determinant approximation. Subsequently, [Kolar and Xing, 2008] proposed a cutting plane strategy that iteratively refines this approximation. We might also consider using convergent message-passing algorithms to improve the stability of the optimization [Kolmogorov, 2006], or trying to optimize the edge appearance probabilities in the covexified Bethe free energy [Wainwright et al., 2002]. An extensive survey of variational inference methods is [Wainwright and Jordan, 2008].

- **Other group structures**: The optimization methods we discuss in Chapter 3 make no assumptions about the group structure except that the groups are disjoint (we consider removing this assumption in Chapter 6), so it is possible to use them for a wide variety of group structures. Besides the cases we discuss here where we use groups to encourage graphical or blockwise sparsity, another interesting possible grouping of the variables occurs for CRFs where the features are binary $\{0, 1\}$ variables. Instead of assigning all edge weights associated with a single edge to the same group, we could consider using groups that only contain the edge weights associated with a single feature for a single edge. In the case of bias variable edge groups, these would correspond to unconditional interactions (interactions between the target variables that exist regardless of the features). In contrast, the $\{0, 1\}$ feature-edge groups would correspond to context-specific dependencies. That is, these would

indicate dependencies that only exist between the target variables when the corresponding feature value is set to 1.

- **Learning the variable types in blockwise-sparse models**: Instead of assuming that the variable types are given, in [Marlin et al., 2009] we consider the problem of learning blockwise-sparse models while estimating the variable types. This work also considers a variation on the blockwise-sparse model where we use ℓ_1-regularization of the blocks but estimate the appropriate scale of the regularization parameter for each block (leading to a form of *soft* blockwise-sparsity). The models in this work rely on a variational Bayesian procedure, where the methods of Chapters 2 and 3 are used as sub-routines in the variational parameter update.

- **Interventional Potentials**: In Section 4.5, we discuss modeling interventions in DAG models using Pearl's *do*-calculus. However, for many data sets the assumption of acycility is often inappropriate; many models of biological networks contain feedback cycles (for example, see Sachs et al. [2005]). In contrast, undirected graphical models allow cycles. However, under most interpretations of the data generating processes associated with undirected graphs there is no difference between conditioning by observation and conditioning by intervention [Lauritzen and Richardson, 2002]; undirected models do not distinguish between observing a variable ('seeing') and setting it by intervention ('doing'). Motivated by the problem of using cyclic models for interventional data, in [Schmidt and Murphy, 2009] we defined the notion of an *interventional* potential. These are undirected potential functions that are augmented with interventional semantics. In [Schmidt and Murphy, 2009], we consider structure learning using group ℓ_1-regularization with interventional potentials on the *cyto* data, and show that this leads to a better model of this data set than causal DAGs or undirected models that ignore the effects of interventions (as in this chapter).

- **Uncertain Interventions**: In [Duvenaud et al., 2010], we consider using general conditional density estimators for making causal predictions. As in the DAG-based uncertain intervention framework of Eaton and Murphy [2007], this model includes explicit binary intervention variables in the model and considers modeling the variables conditional on these intervention variables (alternately, we can consider feature variables that measure properties of the interventions). If we use a CRF as the conditional density estimator, then we have a CRF with binary $\{0, 1\}$ variables. In this case, we could consider using the feature-edge groups above and learning context-specific interactions (i.e. interactions present under different interventions). In the case where we use feature variables that characterize properties of different interventions, this framework would allow the model to make predictions about previously unseen interventions.

- **Optimization-based search**: In this chapter we assumed that the scale of the regularization parameter λ is the same across the groups, but the optimization algorithms in Chapter 3 allow a separate λ_A for each group A. Given that the cost of evaluating a single edge addition/deletion in a search-based structure learning strategy may be similar to solving the convex optimization problem in an ℓ_1-regularization approach to structure learning, we might consider using a search-based method where we simply solve the convex optimization problem for different assignments to the different λ_A variables. We might be able to use this to propose more global moves than single edge additions/deletions, which would not cost more to evaluate since the non-separability the log-likelihood means that evaluating single edge addi-

tions/deletions may be similar to the cost of evaluating completely new graphs. An approach closely related to this was examined in [Moghaddam et al., 2009].

Chapter 6

Hierarchical Log-Linear Model Structure Learning

In Chapter 5, we considered using group ℓ_1-regularization for structure learning in *pairwise* log-linear models. However, on many real data sets it may be important to model higher-order interactions. Thus we would like to relax the pairwise assumption, but as we discuss in Section 1.6 it is challenging to consider general log-linear models without including an explicit cardinality restriction due to the exponential number of possible higher-order potentials.

As an alternative to using an explicit cardinality restriction, we consider fitting general log-linear models (as we describe in Section 1.6) subject to the following constraint:

- **Hierarchical Inclusion Restriction**: If $\mathbf{w}_A = \mathbf{0}$ and $A \subset B$, then $\mathbf{w}_B = \mathbf{0}$.

This is the class of *hierarchical* log-linear models [Bishop et al., 1975, Whittaker, 1990, §7]. While a subset of the space of general log-linear models, the set of hierarchical log-linear models is much larger than the set of pairwise models, and can include interactions of any order. Further, group-sparsity in hierarchical models directly corresponds to conditional independence.

The hierarchical inclusion restriction imposes constraints on the possible sparsity pattern of $\mathbf{w}$, beyond that obtained using (disjoint) group ℓ_1-regularization. In the context of linear regression and multiple kernel learning, several authors have recently shown that group ℓ_1-regularization with *overlapping* groups can be used to enforce hierarchical inclusion restrictions [Zhao et al., 2009, Bach, 2008b]. As an example, if we would like to enforce the restriction that B must be zero when A is zero, we can do this using two groups: The first group simply includes the variables in B, while the second group includes the variables in both A and B. Regularization using these groups encourages A to be non-zero whenever B is non-zero, since when B is non-zero A is not penalized for moving away from zero [see Zhao et al., 2009, Theorem 1].

As an example, consider the simple case where we have a differentiable loss function $L(\mathbf{x})$ where $\mathbf{x}$ has two variables x_1 and x_2, and we want to enforce that variable x_2 is allowed to be non-zero only when x_1 is non-zero. To do this, we use the regularizer $\lambda_{12}||\mathbf{x}_{12}||_2 + \lambda_2|x_2|$. Now, consider a point $\tilde{\mathbf{x}}$ where where x_1 is zero but x_2 is non-zero. At this point the regularizer is differentiable with respect to x_1 with derivative zero, so unless it happens by chance that $\nabla_{x_1} L(\tilde{\mathbf{x}}) = 0$ we can improve the objective function by moving x_1 away from zero.

Generalizing this basic idea, to enforce that the solution of our regularized optimization problem satisfies the hierarchical inclusion restriction we can solve the convex optimization problem[39]

$$\min_{\mathbf{w}} -\sum_{i=1}^{n} \log p(\mathbf{x}^i|\mathbf{w}) + \sum_{A \subseteq S} \lambda_A \Big(\sum_{\{B|A \subseteq B\}} ||\mathbf{w}_B||_2^2 \Big)^{1/2}.$$

[39] Although we will focus on using the ℓ_2 norm of the groups in this chapter, it is possible to use analogous methods where we penalize other norms of the groups.

If we define the set of parameters $\mathbf{w}_A^*$ as the concatenation of the parameters $\mathbf{w}_A$ with all parameters $\mathbf{w}_B$ such that $A \subset B$, we can write this as

$$\min_{\mathbf{w}} -\sum_{i=1}^{n} \log p(\mathbf{x}^i|\mathbf{w}) + \sum_{A \subseteq S} \lambda_A ||\mathbf{w}_A^*||_2. \tag{6.1}$$

This is very similar to applying group ℓ_1-regularization to learn the structure of general log-linear models as in (1.12), except that the parameters of higher-order terms are added to the corresponding lower-order groups. Similar to Theorem 1 of Zhao et al. [2009], we can show that under reasonable assumptions a minimizer of (6.1) will satisfy hierarchical inclusion. We give details about this in the next section after discussing optimality conditions for this problem.

6.1 Optimality Conditions

Using $f(\mathbf{w})$ to denote the objective in (6.1), the sub-differential of $f(\mathbf{w})$ is

$$\partial f(\mathbf{w}) = -\nabla \sum_{i=1}^{n} \log p(\mathbf{x}^i|\mathbf{w}) + \sum_{A \subseteq S} \lambda_A \text{sgn}(\mathbf{w}_A^*),$$

where $\text{sgn}(\mathbf{y})$ is defined as in Section 3.1.2 (we pad the output of this signum function with zeros so that it has the right dimension). Recall that a vector $\tilde{\mathbf{w}}$ is a minimizer of a convex function if and only if $\mathbf{0} \in \partial f(\tilde{\mathbf{w}})$ [Bertsekas, 1999, §B.5].

We call A an *active group* if $\mathbf{w}_B \neq \mathbf{0}$ for some B such that $A \subseteq B$. If A is not an active group and $\mathbf{w}_B = \mathbf{0}$ for some $B \subset A$, we call A an *inactive group*. We refer to the remaining groups as *boundary groups*; a boundary group A satisfies $\mathbf{w}_B \neq \mathbf{0}$ for all $B \subset A$ and $\mathbf{w}_C = \mathbf{0}$ for all $A \subseteq C$. In other words, the boundary groups are the groups that can be made non-zero without violating hierarchical inclusion.

The optimality conditions with respect to an active group A reduce to

$$\nabla_{\mathbf{w}_A} \sum_{i=1}^{n} \log p(\mathbf{x}^i|\mathbf{w}) = \sum_{B \subseteq A} \lambda_A \mathbf{w}_A / ||\mathbf{w}_B^*||_2. \tag{6.2}$$

If we treat all inactive groups as fixed, the optimality conditions with respect to a boundary group A become

$$||\nabla_{\mathbf{w}_A} \sum_{i=1}^{n} \log p(\mathbf{x}^i|\mathbf{w})||_2 \leq \lambda_A. \tag{6.3}$$

The combination of (6.2) and (6.3) constitute necessary and sufficient conditions for a minimizer of (6.1) under the constraint that inactive groups are fixed at zero. These also comprise necessary (but not necessarily sufficient) conditions for global optimality of (6.1).

We can now show that under reasonable assumptions that minimizers of (6.1) satisfy hierarchical inclusion. Assume we have a minimizer $\tilde{\mathbf{w}}$ of (6.1) that does not. Then there exists some A such that $\tilde{\mathbf{w}}_A = \mathbf{0}$ and some B such that $A \subset B$ and $\tilde{\mathbf{w}}_B \neq \mathbf{0}$. This implies group A is active and must satisfy (6.2). Using $\tilde{\mathbf{w}}_A = \mathbf{0}$, we have that $\nabla_{\mathbf{w}_A} \log p(\mathbf{x}|\tilde{\mathbf{w}})$ is exactly $\mathbf{0}$, and assuming the set where this happens has zero probability it contradicts that $\tilde{\mathbf{w}}_A$ is a minimizer.

Unfortunately, there are several complicating factors in solving (6.1). In particular, (i) there remains an exponential number of groups to consider and (ii) we can no longer compute the projection (or soft-threshold) operator used by the optimization algorithms in Chapter 3. We address the former issue first.

6.2 Regularization Path and Active-Set Optimization

We would like to avoid having to consider the exponential number of groups present in (6.1). Since we know that the solution is a hierarchical model, we propose to use an active-set method that incrementally adds variables to the problem until (6.2) and (6.3) are satisfied, that uses hierarchical inclusion to exclude the possibility of adding most variables. The method alternates between two phases:

- Find the set of active groups, and the boundary groups violating (6.3).
- Solve the problem with respect to these variables.

We repeat this until no new groups are found in the first step, and at this point we have (by construction) found a point satisfying (6.2) and (6.3). This is analogous to the active-set methods of Chapters 2 and 3, but note that here we only consider adding groups that satisfy hierarchical inclusion. In this algorithm, the addition of boundary groups has an intuitive interpretation; we only add the zero-valued group A if it satisfies hierarchical inclusion and the difference between the model marginals and the empirical frequencies exceeds λ_A. Such an addition rule is very reminiscent of the method of [Gevarter, 1987]. This method greedily adds constraints on higher-order marginals if the observed higher-order marginals differ significantly from the model's higher-order marginals after fitting lower-order marginals, for the closely related problem of computing a maximum entropy distribution subject to given marginal constraints [Cheeseman, 1983]. However, the proposed method differs from the prior work in that the active-set method can add or remove variables, and it makes progresses towards solving a convex optimization problem.

Consider a simple 6-node hierarchical log-linear model containing non-zero potentials on (1)(2)(3)(4)(5)(6)(1,2)(1,3)(1,4)(4,5)(4,6)(5,6)(4,5,6). Though there are 20 possible threeway interactions in a 6-node model, only one satisfies hierarchical inclusion, so our method would not consider the other 19. Further, we do not need to consider any fourway, fiveway, or sixway interactions since none of these satisfy hierarchical inclusion. In general, we might need to consider more higher-order interactions, but we will never need to consider more than a polynomial number of groups more than the number present in the final model. That is, hierarchical inclusion and the active-set method can save us from looking at an exponential number of irrelevant higher-order factors.

To stop us from considering overly complicated models that do not generalize well, to set the regularization parameter(s) we can start with the unary model and incrementally decrease the regularization until a measure of generalization error starts to increase. This is analogous to the regularization path methods mentioned in Chapters 2 and 3, but augmented with a termination criteria. Before moving on to how we can solve the problem with respect to a subset of the groups, we summarize the computational gains that can be achieved for computing the ℓ_1-regularization path compared to the ℓ_2-regularization path for the optimization problems we discuss in Chapters 2, 3, and 6:

- Chapter 2: For logistic regression with ℓ_1-regularization, for large values of λ we may reduce the cost of evaluating the objective function by a polynomial factor, and reduce the number of variables by a polynomial factor.
- Chapter 3: For pairwise log-linear models with group ℓ_1-regularization, for large values of λ we may reduce the cost of evaluating the objective function by an exponential factor, and reduce the number of variables by a polynomial factor.
- Chapter 6: For hierarchical log-linear models with overlapping group ℓ_1-regularization, for large values of λ we may reduce the cost of evaluating the objective function by an exponential factor, and reduce the number of variables by an exponential factor.

6.3 Constrained Formulation

In step 1 of the active-set method we must solve (6.1) with respect to a subset of the groups. This comprises a group ℓ_1-regularization problem with overlapping groups. Besides a special case discussed in [Zhao et al., 2009] where the solution can be computed directly, previous approaches to solving group ℓ_1-regularization problems with overlapping groups include a boosted LASSO method [Zhao et al., 2009] and a re-formulation of the problem as a smooth objective with a simplex constraint [Bach, 2008b]. Unfortunately, applying these methods to graphical models would be relatively inefficient since they might require a very large number of function evaluations.

As before, we can solve the problem by writing it as an equivalent differentiable but constrained problem. In particular, we again introduce a scalar auxiliary variable g_A to bound the norm of each group $\mathbf{w}_A^*$, leading to a smooth objective with second-order cone constraints:

$$\min_{\mathbf{w},\mathbf{g}} -\log p(\mathbf{x}|\mathbf{w}) + \sum_{A \subseteq S} \lambda_A g_A, \quad s.t.\ \ g_A \geq ||\mathbf{w}_A^*||_2, \forall_A. \tag{6.4}$$

As we saw in the Chapter 3, the projection for each group has a simple closed-form solution. Thus, we might consider solving this problem with the SPG or PQN method. However, because the groups now overlap, we can no longer compute the projection onto each group independently.

6.4 Dykstra's Algorithm

We would like to solve the problem of computing the projection onto a convex set defined by the intersection of sets, where we can efficiently project onto each individual set. One of the earliest results on this problem is due to von Neumann [1950, §13], who proved that the limit of cyclically projecting a point onto two closed linear sets is the projection onto the intersection of the sets. Bregman [1965] proposed to cyclically project onto a series of general convex sets in order to find a point in their intersection, but this method will not generally converge to the projection. The contribution of Dykstra [1983] was to show that by taking the current iterate and removing the difference calculated from the previous cycle, then subsequently projecting this value, that the cyclic projection method converges to the optimal solution for general (closed) convex sets. Deutsch and Hundal [1994] have shown that Dykstra's algorithm converges at a geometric rate for polyhedral sets (the set defined with the ℓ_2 group norm is not polyhedral, but the set defined with the ℓ_∞

group norm is polyhedral). Algorithm 11 gives pseudo-code for an implementation of Dykstra's algorithm (we obtain Bregman's method if we fix I_i at $\mathbf{0}$).

Input: Point $\mathbf{w}_0$, convex sets $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_q$, tolerance ϵ
Output: $\mathcal{P}_{\mathcal{C}}(\mathbf{w}_0)$, the projection of $\mathbf{w}_0$ onto $\mathcal{C} \triangleq \bigcap_{i=1}^{q} \mathcal{C}_i$.
$\forall_i, I_i \leftarrow \mathbf{0}$;
$j \leftarrow 0$;
while $\mathbf{w}_j$ *is changing by more than* ϵ **do**
  for $i = 1$ *to* q **do**
    $\mathbf{w}_j \leftarrow \mathcal{P}_{\mathcal{C}_i}(\mathbf{w}_{j-1} - I_i)$;
    $I_i \leftarrow \mathbf{w}_j - (\mathbf{w}_{j-1} - I_i)$;
    $j \leftarrow j + 1$;

Algorithm 11: Dykstra's cyclic projection algorithm for finding the projection of a point onto an intersection of convex sets.

Despite its simplicity, Dykstra's algorithm is not widely used because of its high storage requirements. In its unmodified form, applying Dykstra's algorithm to compute the projection in (6.4) would be impractical, since for each group we would need to store a copy of the entire parameter vector. Fortunately, in (6.4) each constraint only affects a small subset of the variables. By taking advantage of this it is straightforward to derive a sparse variant of Dykstra's algorithm that only needs to store a copy of each variable for each group that it is associated with (rather than one copy of the entire parameter vector for each group). This leads to an enormous reduction in the memory requirements. Further, although using Dykstra's algorithm rather than an analytic update leads to a higher iteration cost, the cost of running the cyclic projection algorithm will typically be much smaller than the cost of evaluating the objective function.

6.4.1 Soft-Dykstra's Algorithm

Allowing the groups to overlap also means that the soft-threshold operator can not be applied independently to the different groups. Given the similarity between the projection and the soft-threshold operator, we might expect to be able to derive a variant on Dykstra's algorithm that is able to solve the soft-threshold problem with overlapping groups. Bauschke and Combettes [2008] present a generalization of Dykstra's algorithm that can be used to solve this problem, outlined in Algorithm 12.

Input: Point $\mathbf{w}_0$, convex regularizers $\mathcal{R}_1(\mathbf{w}), \mathcal{R}_2(\mathbf{w}), \ldots, \mathcal{R}_q(\mathbf{w})$, tolerance ϵ, step size α
Output: $\mathcal{S}_{\mathcal{R}}(\mathbf{w}_0, \alpha)$, the soft-threshold operator with input $\mathbf{w}_0$, step size α, and regularizer $\mathcal{R}(\mathbf{w}) \triangleq \sum_{i=1}^{q} \mathcal{R}_i(\mathbf{w})$.
$\forall_i, I_i \leftarrow \mathbf{0}$;
$j \leftarrow 0$;
while $\mathbf{w}_j$ *is changing by more than* ϵ **do**
  for $i = 1$ *to* q **do**
    $\mathbf{w}_j \leftarrow \mathcal{S}_{\mathcal{R}_i}(\mathbf{w}_{j-1} - I_i, \alpha)$;
    $I_i \leftarrow \mathbf{w}_j - (\mathbf{w}_{j-1} - I_i)$;
    $j \leftarrow j + 1$;

Algorithm 12: Variant of Dykstra's algorithm for computing soft-threshold operators for a regularizer consisting of the sum of convex regularizers.

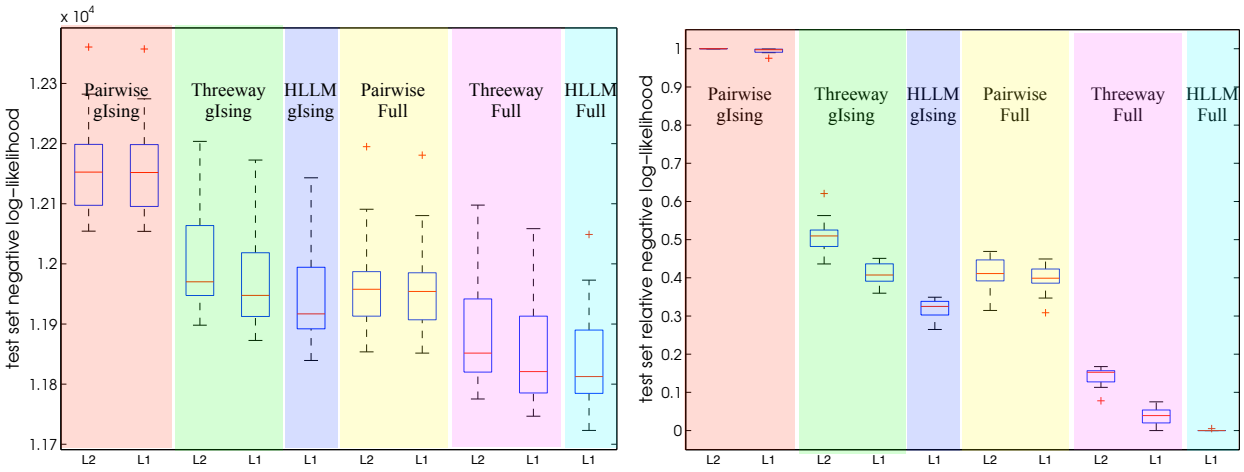

Figure 6.1: Test set negative log-likelihood (left) and relative negative log-likelihood (right) on the *cyto* data using different regularization types and potential restrictions.

The appeal of using Algorithm 12 extends beyond the fact that the soft-threshold algorithm is a simpler, more direct, and potentially more efficient strategy than projection methods. This is because Dykstra's projection method typically approaches the optimal projection through a sequence of infeasible iterates. Thus, in the projection framework we must solve it sufficiently accurately to guarantee that we are (numerically close enough to) feasible. In contrast, since there is no notion of feasibility (in the BBST and QNST algorithms) we might be able to terminate the soft-threshold variant of the algorithm early.

6.5 Experiments

In this section we re-visit building generative models of the data sets examined in the last chapter, but consider fitting models that relax the pairwise assumption. In the next section we re-visit the two data sets where exact likelihood calculation was possible, and then we turn to several of the larger data sets.

6.5.1 Smaller Data

We first re-visit building generative models of the *cyto* and *awma* small data sets, where we use exact likelihood calculation and consider both the full and gIsing parameterizations. On each data set we compared our hierarchical log-linear model with overlapping group ℓ_1-regularization (labeled HLLM in the figures) to fitting log-linear models restricted to both pairwise and threeway potentials with both ℓ_2-regularization and group ℓ_1-regularization with the group ℓ_2 norm. Note that unlike the pairwise and threeway models, an ℓ_2-regularized version of the hierarchical log-linear model is infeasible. We trained on a random half of the data set, and tested on the remaining half as the regularization parameter λ was varied. For the pairwise and threeway models, we set λ_A to the constant λ. For the hierarchical model, we set λ_A to $\lambda 2^{|A|-2}$, where $|A|$ is the cardinality of A and we placed no explicit restriction on the cardinality of A. For all the models, we did not regularize the unary weights.

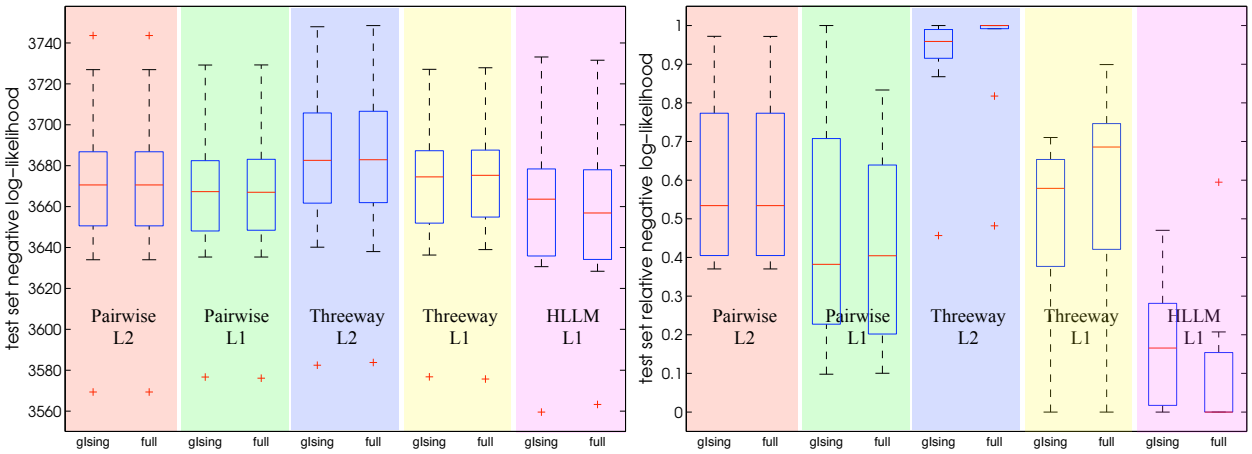

Figure 6.2: Test set negative log-likelihood (left) and relative negative log-likelihood (right) on the *awma* data using different regularization types and potential restrictions.

We plot the results obtained on the *cyto* data in Figure 6.1. On this data set, we see that allowing for threeway interactions leads to better performance than using pairwise interactions (for both types of potentials), and further that the hierarchical model that allows higher-order interactions leads to a further improvement. The HLLM with full potentials included up to fourway potentials, while with Ising potentials fiveway potentials were also included.

We plot the results obtained on the *awma* data in Figure 6.2. On this data set that the threeway models do no better than the pairwise models, and the ℓ_2-regularized threeway model seems to do worse than the pairwise models. In contrast, the hierarchical model seems to have an advantage over the pairwise models. On this data set the HLLMs included fourway interactions on nine of the ten trials when using full potentials, and additionally included fiveway potentials on two of the ten trials when using Ising potentials.

6.5.2 Larger Data

We next tested the various methods on several larger data sets, concentrating on the case of gIsing potentials and the pseudo-likelihood approximation. We plot the test-set pseudo-log-likelihood for the *awma5*, *traffic*, and *usps4* data sets in Figures 6.3-6.5. In these figures we see that modeling threeway interactions gives improved results for two of these three data sets. However, the hierarchical model dominated the threeway models, and did substantially better than the pairwise models except on the *awma5* data where the pairwise model with ℓ_1-regularization does similar. On the *awma5* data set the HLLM only included pairwise and threeway factors, while it included fourway factors on the *traffic* data set and fiveway factors on the *usps4* data set.

Although we concentrated on these relatively small data sets in our experiments, these data sets are still larger than previous data where higher-order models have been applied. For example, Dahinden et al. [2007] use (disjoint) group ℓ_1-regularization and only considered up to 5 binary variables, while [Dobra and Massam, 2010] considered log linear models over 16 binary variables and used stochastic local search to identify the structure (this search-based method requires fitting each model during the search, which is very expensive). In contrast, the *traffic* data examined in this work contains 32 four-state variables. Our method can in principle be used to learn models with

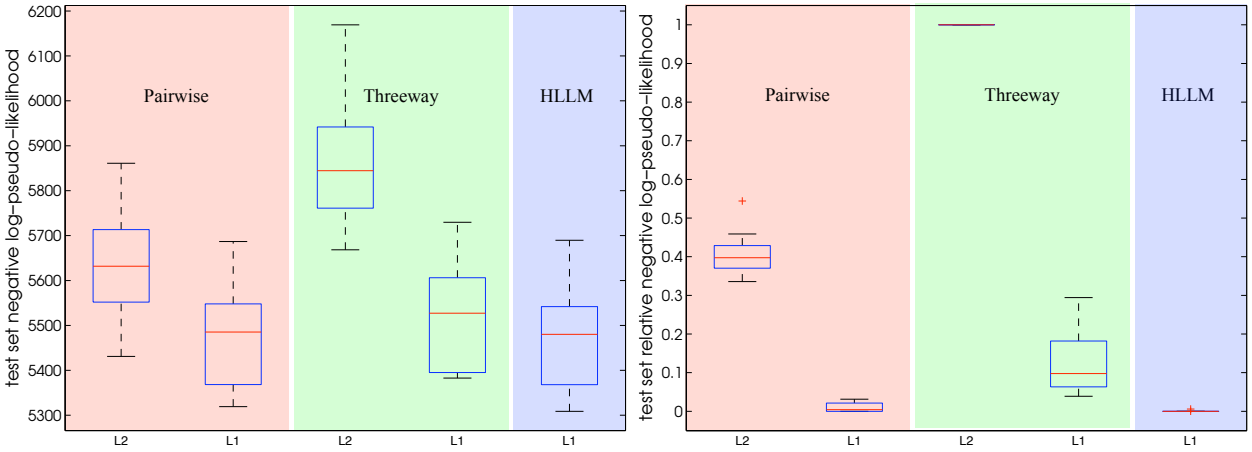

Figure 6.3: Test set negative pseudo-log-likelihood (left) and relative negative log-likelihood (right) on the *awma5* data using different regularization types and potential restrictions.

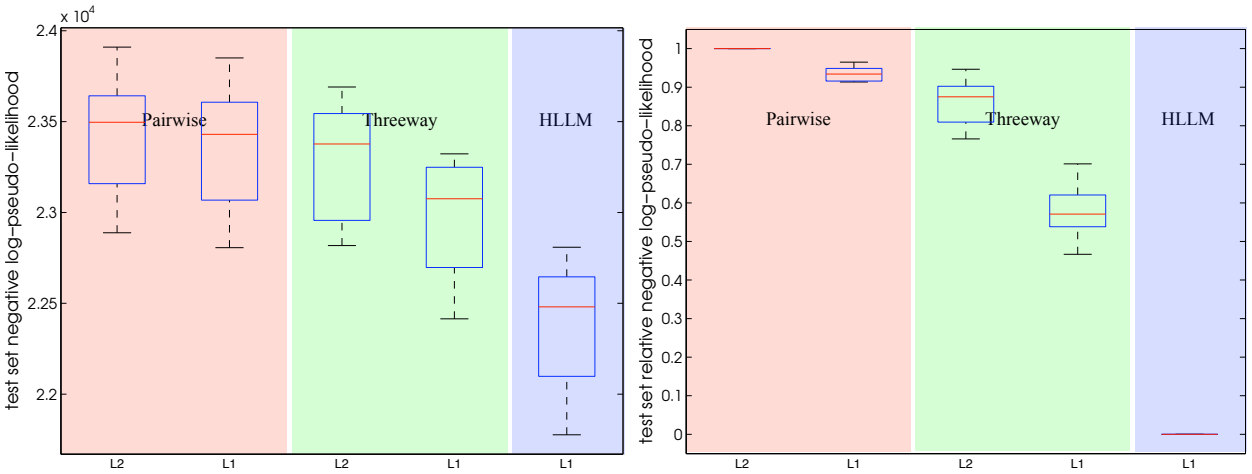

Figure 6.4: Test set negative pseudo-log-likelihood (left) and relative negative log-likelihood (right) on the *traffic* data using different regularization types and potential restrictions.

higher-order interactions on even larger data sets, provided that the solution of the optimization problem is sufficiently sparse.

6.5.3 Structure Estimation

We next sought to assess the performance of the HLLM for structure estimation. We created a 10-node synthetic data set that includes all unary factors as well as the factors $(2,3)(4,5,6)$ $(7,8,9,10)$ (a non-hierarchical model), where the model weights were generated from a $\mathcal{N}(0,1)$ distribution. In Figure 6.5.3, we plot the number of false positives of different orders present in the first model along the regularization path that includes all three factors in the true structure against the number of training examples (we define a false positive as a factor where none of its supersets are present in the true model). For example, with 20000 samples the order of edge additions was (with false positives in square brackets) (8,10)(7,9)(9,10)(7,10)(4,5)(8,9)(2,3)(4,6)(8,9,10)(7,8)(7,8,9)(7,8,10)

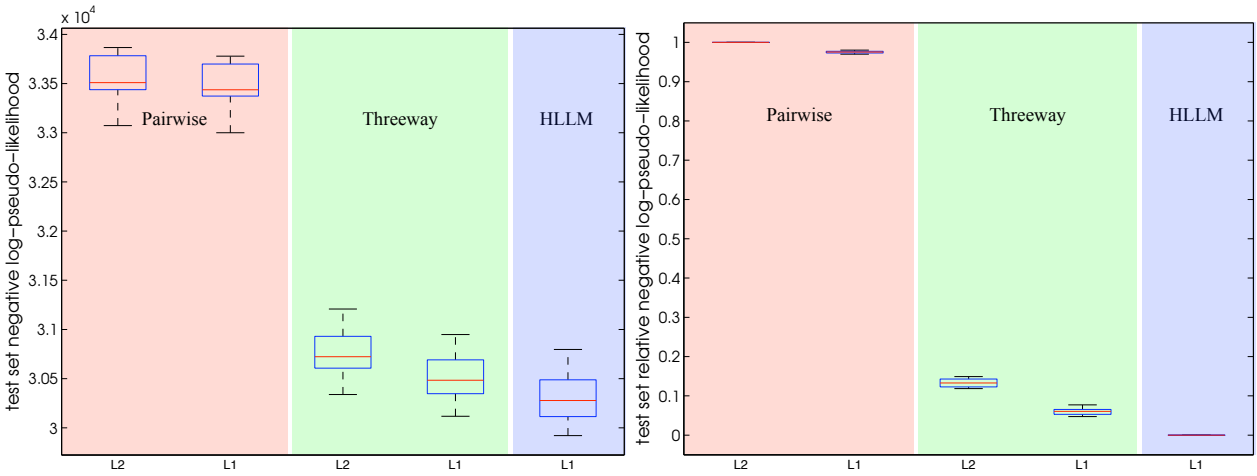

Figure 6.5: Test set negative pseudo-log-likelihood (left) and relative negative log-likelihood (right) on the *usps4* data using different regularization types and potential restrictions.

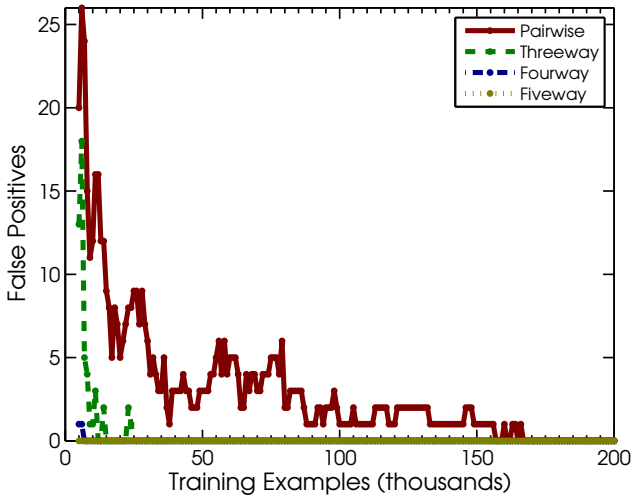

Figure 6.6: False positives of different orders against training set size for the first model along the regularization path where the HLLM selects a superset of the true data-generating model [Schmidt and Murphy, 2010].

(5,6)[1,8][5,9][3,8][3,7](4,5,6)[1,7](7,9,10)(7,8,9,10) (at this point it includes all three factors in the true structure, with 5 pairwise false positives and no higher-order false positives). In the figure, we see that the model tends to include false positives before it adds all true factors, but the number decreases as the sample size increases. Further, there tend to be few higher-order false positives; although it includes spurious pairwise factors even with 150000 samples, the model includes no spurious threeway factors beyond 30000 samples, no spurious fourway factors beyond 10000 samples, and no fiveway factors for any sample size (the plot begins at 5000).

We next examined the coronary heart disease data set analyzed in [Edwards and Havránek, 1985]. The first fifteen factors added along the HLLM-L1 regularization path on this data set are

(B,C)(A,C)(B,E)(A,E)(C,E)(D,E)(A,D)(B,F)(E,F)[C,D][A,F](A,D,E)(D,F)[D,E,F][A,B]. We have used square brackets to denote factors that are not recognized in the prior work, and may represent false positives due to the use of a point estimate with this small sample size. The first seven factors are the union of the minimally sufficient hierarchical models from the analysis by Edwards and Havránek. These are also the factors with posterior mode greater than 0.5 for a prior strength of 2 and 3 in the hierarchical models of [Dobra and Massam, 2010], while the first eight are the factors selected with a prior strength of 32 and 64. With a prior strength of 128 Dobra and Massam [2010] find the ninth factor introduced by our model, as well as the factor (D,F) introduced later. The remaining factor with this prior strength is the factor (B,C,F), that is not found until much later in the regularization path in our model. In contrast, the first three-way factor introduced by our model is (A,D,E). This factor is present in both of the accepted graphical models in [Edwards and Havránek, 1985], and is the only threeway factor with a posterior greater than 0.5 (under a Laplace approximation) in the graphical models of [Dobra and Massam, 2010] for a prior strength of 1, 2, 3, 32, and 64.

6.6 Similar Methods

In this chapter we considered using group ℓ_1-regularization for structure learning in discrete undirected graphical models where the pairwise assumption is relaxed. The only prior work we are aware of that has considered this case is [Dahinden et al., 2007]. However, in [Dahinden et al., 2007] they use disjoint group ℓ_1-regularization and thus in general group sparsity in their model does not correspond to conditional independence. Further, Dahinden et al. [2007] ignore the challenges associated with considering higher-order factors when the number of variables is non-trivial. In contrast, this chapter has provided methods for addressing the problems associated with the intractable objective function and the exponential number of higher-order terms. These considerations allow the method we discuss in this chapter to be applied to much larger data sets, without any explicit restriction on the cardinality of the model.

6.7 Extensions

Below we discuss several extensions of the work we discuss in this chapter:

- **DAGs**: We have considered using hierarchical inclusion in order to provide a tractable way to relax the pairwise assumption in undirected graphical models. The use of sigmoid (or Gaussian) CPDs in Chapter 4 is very similar to the pairwise assumption, and hierarchical inclusion would also be useful in DAG models. For example, if we are given a variable ordering and use Gaussian CPDs then hierarchical inclusion can be used to encourage the Cholesky matrix to have a low-bandwidth. Alternatively, we could consider sigmoid (or Gaussian) CPDs where we use a non-linear basis expansion of the parent variables. Hierarchical inclusion could then be used to tractably search through the exponential space of possible terms to include, as in [Bach, 2008b].

- **Conditional and interventional models**: We have considered unconditional models in this chapter, but as in Chapter 5 we could also consider conditional models and models with interventional potentials.

- **Other group structures**: Rather than using the ℓ_2 group norm, we could apply Dykstra's algorithm with the ℓ_∞ group norm (or any norm where we can efficiently compute the projection). Further, we can still apply Dykstra's algorithm under different assignments of variables to overlapping groups. It is also possible that better performance would be achieved by a different selection of the regularization weights λ_A.

- **Other overlapping group schemes**: Jacob et al. [2009] consider a different notion of overlapping groups to encourage a sparsity pattern that is a union of groups. They represent each variable as a combination of auxiliary variables and penalize these (disjoint) variables. We could enforce hierarchical inclusion in this framework by adding to each group all *subsets* of the group, as opposed to all supersets in (6.1). An advantage of this is that the projection (or soft-threshold) could easily be computed using the methods of Chapter 3, but a disadvantage is that it would be grossly over-parameterized (we would have an auxiliary variable for every subset of each non-zero factor). Further, although the result would still be hierarchical it might be the case that the auxiliary variables associated with lower-order groups would be zero (since the parameters of the lower-order group would be represented by the version associated with a higher-order group), so it seems less likely that an efficient active-set method that finds the globally optimal solution could be developed.

- **Sufficient conditions**: The active-set method converges to a method satisfying necessary optimality conditions for (6.1) and conditions that are sufficient under the constraint that inactive groups are fixed at zero. However, it may terminate at a point that is not a global optimum of the full problem (6.1) since it may terminate at a point where making an inactive group non-zero could improve the objective. Thus, an outstanding issue is deriving an efficient way to test (or bound) sufficient optimality conditions over all variables in order to ensure global optimality, as in [Bach, 2008b]. Since the gradient of the objective function is bounded in magnitude, it is likely that such tests are possible. Given such a test, a related issue is developing an efficient search procedure for finding sub-optimal inactive groups. Even if such a test is intractable in general, several heuristic strategies are possible that would improve the likelihood that we find the global optimum. For example, we could test not only all boundary groups, but all groups that would become boundary groups if the current boundary groups were non-zero. This test would still only need to consider a polynomial number of groups more than is non-zero at the current iterate.

Chapter 7

Discussion

In this chapter, we discuss several issues that have been ignored in this work, as well as several interesting extensions of this work and possible directions of future work.

- **Missing data and hidden variables**: In this work, we have assumed that all variables are observed in all training samples. We can also consider scenarios where the values of some variables are missing or where the values of some variables are hidden by marginalizing over the missing values. In the case of undirected models, if we use O to denote the observed variables and H to denote the hidden variables, we could write the probability of the observed variables in this scenario as

$$p(\mathbf{x}_O) \triangleq \sum_{\mathbf{x}_H} p(\mathbf{x}_O, \mathbf{x}_H) = \frac{1}{Z} \sum_{\mathbf{x}_H} \tilde{p}(\mathbf{x}_O, \mathbf{x}_H),$$

  where we have used $\tilde{p}(\mathbf{x}_O, \mathbf{x}_H)$ to denote the unnormalized product of the potential functions. Although in principle this is a straightforward extension, the sum over values of the hidden variables complicates the optimization. In particular, computing this sum might require approximate inference. Further, this sum leads to a (non-linear) concave term in the negative log-likelihood, so the objective function is no longer convex. We could consider directly finding a local minimum of the resulting non-convex optimization problem with one of the methods we describe in Chapter 3. Alternately, we could find a local minimum by using the expectation maximization (EM) algorithm [Dempster et al., 1977] to yield a sequence of convex optimization problems of the form addressed in Chapter 3 that would upper bound the objective function.

- **Mixture models**: In Chapter 6 we considered using higher-order potentials to model complicated distributions that are not fully-characterized by pairwise statistical dependencies. An alternative and *complementary* strategy to increase the representational power of models is with mixtures. Here, we would represent the probability of an observed vector as a convex combination of independent graphical models

$$p(\mathbf{x}) \triangleq \sum_{c=1}^{C} \pi_c p(\mathbf{x}|\mathbf{w}_c),$$

  where $\sum_{c=1}^{C} \pi_c = 1$. Even though the individual graphical models are independent, the use of a convex combination introduces dependencies between all variables (assuming we have at least $C = 2$ mixture components). For example, even if we use a completely disconnected graph, all variables are dependent in the joint distribution [Bishop, 2006, §9.3.3]. Previous work has examined mixtures of tree-structured graphical models [Meila and Jordan, 2000], but we could consider mixtures of general graphical models. As with the case of missing

variables, this formulation is not convex but we could use the EM algorithm to find a local minimum by solving a sequence of convex problems of the form addressed in Chapter 3.

- **Stochastic inference and online estimation**: In this work we have focused on the case of deterministic approximations to the marginals in undirected graphical models. An alternative class of methods exist that generate stochastic samples from the distribution in order to approximate the marginals [Koller and Friedman, 2009, §12]. The advantage of these methods is that, as the sample size increases, they converge to the true marginals. However, with finite sample sizes the approximation will not be exact and there may be discontinuities in the associated objective function. One way to optimize under this sort of approximation is with stochastic approximation methods where we alternate between generating samples and updating the parameters [Younes, 1989]. It is well known that projections can be used within stochastic approximation methods [Kushner and Yin, 2003, §5], while more recent work has examined stochastic approximation methods that use the soft-threshold operator [Duchi and Singer, 2009]. We could also use the stochastic approximation framework to apply the techniques we describe in an *online* setting, where rather than a fixed training set we receive training samples one at a time.

- **Other types of structure learning**: This work has concentrated on the cases of linearly-parameterized DAG models, pairwise log-linear models, and hierarchical log-linear models. However, it is possible to extend the ideas we discuss here to other types of models. For example, we could consider learning the structure of chain-graph models [Lauritzen, 1996, §3.2.3] (models that combine directed and undirected edges), by using a search-based method to search through the space of (block-)DAG models and using group ℓ_1-regularization to learn the undirected structure within the blocks. Similarly, the methods of Chapter 4 may be useful for structure learning in ancestral graph Markov models, a generalization of DAG models that is closed under marginalization and conditioning [Richardson and Spirtes, 2002]. It might also be possible to use the ideas of Chapter 6 to learn probabilistic context free grammars or first-order probabilistic models [Rusell and Norvig, 2003, §14.6 and §23.1], or Markov logic networks [Richardson and Domingos, 2006].

- **Bayesian methods**: The regularized parameter estimates we use in this work can be interpreted as the maximizers of a posterior distribution under an appropriate prior. If we have a small sample size and are interested in the task of structural estimation, it may prove more useful to find the posterior probability of an edge parameter taking a value of zero in this posterior distribution. In this case, it does not make sense to use an ℓ_1-regularizer because the edge posterior is zero in the posterior with probability zero. Thus, in this case we would need to consider a prior/regularizer that places an atom at zero in the prior distribution. Although there has been some work on exact computation of edge posteriors in models with a small number of variables [Koivisto and Sood, 2004], a variational or stochastic approximation would likely be needed to approximate the edge posteriors. In the variational case, the methods of Chapter 2 or 3 may be useful in implementing the variational update, similar to our work in [Marlin et al., 2009].

- **Max-margin training**: For CRFs, an alternative to optimizing the conditional likelihood is to use a maximum-margin training objective [Taskar et al., 2003]. The maximum-margin objective is non-differentiable (when formulated as an unconstrained optimization), but does

not depend on the normalizing constant in the model. Instead, it depends on the most likely configuration under the model (or the second most likely, in some variations). While computing the normalizing constant is still NP-hard (as opposed to the #P-hard task of evaluating the normalizing constant), there exist several special cases where we can compute the most likely configuration even though we can not compute the normalizing constant. For example, [Taskar et al., 2004] consider using maximum-margin training in binary models with constraints that enforce sub-modular potentials (assuming a fixed structure). We could consider a variant on this method (or other cases where we can efficiently compute the most likely configuration) where we use (group) ℓ_1-regularization of the edge parameters to learn a sparse structure. Unlike training with the conditional likelihood, if we enforce sub-modularity of the edge potentials it would be possible to evaluate the objective function in this problem (without using approximations) even for a non-trivial number of nodes.

We conclude by noting that implementations of the methods discussed in this thesis will be made available on the author's homepage.

Bibliography

P. Abbeel, D. Koller, and A. Y. Ng. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, 7:1743–1788, 2006.

S. Acid, L. de Campos, and J. Huete. The search of causal orderings: A short cut for learning belief networks. *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 2001.

C. Aliferis, I. Tsamardinos, A. Statnikov, and L. Brown. Causal explorer: A causal probabilistic network learning toolkit for biomedical discovery. *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, 2003.

G. Andrew and J. Gao. Scalable training of L_1-regularized log-linear models. *International Conference on Machine Learning*, 2007.

F. Bach. Bolasso: model consistent Lasso estimation through the bootstrap. *International Conference on Machine Learning*, 2008a.

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. *Conference on Neural Information Processing Systems*, 2008b.

F. Bach and M. Jordan. Thin junction trees. *Conference on Neural Information Processing Systems*, 2001.

S. Bakin. *Adaptive regression and model selection in data mining problems.* PhD thesis, Australian National University, Canberra, 1999.

O. Banerjee, L. E. Ghaoui, A. d'Aspremont, and G. Natsoulis. Convex optimization techniques for fitting sparse gaussian graphical models. *International Conference on Machine Learning*, 2006.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

J. Barzilai and J. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8 (1):141–148, 1988.

H. Bauschke and P. Combettes. A Dykstra-like algorithm for two monotone operators. *Pacific Journal of Optimization*, 4(3):383–391, 2008.

D. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999.

D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex Analysis and Optimization.* Athena Scientific, 2003.

D. Bertsimas and J. Tsitsiklis. *Introduction to linear optimization.* Athena Scientific, 1997.

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, 1975.

J. Besag. Efficiency of pseudolikelihood estimators for simple gaussian fields. *Biometrika*, 64(3):616–618, 1977.

E. Birgin, J. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

C. Bishop. *Pattern recognition and machine learning.* Springer, 2006.

Y. Bishop, F. E., and H. P. *Discrete multivariate analysis: Theory and practice.* MIT Press, 1975.

R. Bouckaert. Probabilistic network construction using the minimum description length principle. *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, 1993.

S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge University Press, 2004.

L. Bregman. The method of successive projection for finding a common point of convex sets. *Doklady Akademii Nauk*, 162(3):487–490, 1965. An English translation appears in Soviet Mathematics Doklady, 6:688–692, 1965.

W. Buntine. Theory refinement on Bayesian networks. *Conference Uncertainty in Artificial Intelligence*, 1991.

R. Byrd, J. Nocedal, and R. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(2):129–156, 1994.
J. Cai, E. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.
A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. *Conference on Neural Information Processing Systems*, 2007.
P. Cheeseman. A method of computing generalized bayesian probability values for expert systems. *International Joint Conference on Artificial Intelligence*, 1983.
G. Chen and R. Rockafellar. Convergence rates in forward-backward splitting. *SIAM Journal on Optimization*, 7(2):421–444, 1997.
S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.
J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning Bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, 137(1-2):43–90, 2002.
D. Chickering. Learning Bayesian networks is NP-complete. *Conference on Artificial Intelligence and Statistics*, 1995.
D. Chickering. Optimal structure identification with greedy search. *The Journal of Machine Learning Research*, 3:507–554, 2003.
D. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *The Journal of Machine Learning Research*, 5:1287–1330, 2004.
C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.
J. Claerbout and F. Muir. Robust modeling with erratic data. *Geophysics*, 38:826–844, 1973.
D. Cobzas and M. Schmidt. Increase Discrimination in Level Set Methods with Embedded Conditional Random Fields. *Conference on Computer Vision and Pattern Recognition*, 2009.
P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005.
G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.
G. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. *Conference Uncertainty in Artificial Intelligence*, 1999.
G. Cormack and T. Lynam. Spam corpus creation for TREC. *Conference on Email and Anti-Spam*, 2005.
T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2nd edition, 2001.
C. Dahinden, G. Parmiggiani, M. Emerick, and P. Bühlmann. Penalized likelihood for sparse contingency tables with an application to full-length cDNA libraries. *BMC Bioinformatics*, 8:476, 2007.
J. Dahl, V. Roychowdhury, and L. Vandenberghe. Maximum likelihood estimation of gaussian graphical models: numerical implementation and topology selection. Technical report, UCLA, 2005.
Y. Dai and R. Fletcher. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. *Numerische Mathematik*, 100(1):21–47, 2005.
S. Dasgupta. Learning polytrees. *Proc. UAI*, pages 134–141, 1999.
D. Dash and M. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse data. *Conference Uncertainty in Artificial Intelligence*, 1999.
A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-Order Methods for Sparse Covariance Selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2008.
I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.
A. Dawid and S. Lauritzen. Hyper Markov laws in the statistical analysis of decomposable graphical models. *Annals of Statistics*, 21(3):1272–1317, 1993.
L. de Campos, J. Fernández-Luna, J. Gámez, and J. Puerta. Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, 31(3):291–311, 2002a.
L. de Campos, J. Fernandez-Luna, and J. Puerta. Local search methods for learning Bayesian networks

using a modified neighborhood in the space of dags. *Ibero-American Conference on Artificial Intelligence*, 2002b.

A. Dempster. Covariance selection. *Biometrics*, 28(1):157–175, 1972.

A. Dempster, N. Laird, D. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–38, 1977.

A. Deshpande, M. Garofalakis, and M. Jordan. Efficient stepwise selection in decomposable models. *Conference Uncertainty in Artificial Intelligence*, 2001.

A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. *Conference on Very Large Data Bases*, 2004.

F. Deutsch and H. Hundal. The rate of convergence of Dykstra's cyclic projections algorithm: The polyhedral case. *Numerical Functional Analysis and Optimization*, 15(5-6):537–565, 1994.

A. Dobra and H. Massam. The mode oriented stochastic search (MOSS) algorithm for log-linear models with conjugate priors. *Statistcal Methodology*, 2010. In Press.

A. Dobra, C. Hans, B. Jones, J. Nevins, G. Yao, and M. West. Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90(1):196–212, 2004.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse gaussians. *Conference Uncertainty in Artificial Intelligence*, 2008a.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. *International Conference on Machine Learning*, 2008b.

D. Duvenaud, D. Eaton, K. Murphy, and M. Schmidt. Causal learning without DAGs. *Journal of Machine Learning Research Workshop and Conference Proceedings*, 6:177–190, 2010.

R. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

D. Eaton and K. Murphy. Exact Bayesian structure learning from uncertain interventions. *Conference on Artificial Intelligence and Statistics*, 2007.

D. Edwards. *Introduction to graphical modelling*. Springer, 2000.

D. Edwards and T. Havránek. A fast procedure for model search in multidimensional contingency tables. *Biometrika*, 72(2):339–351, 1985.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2): 407–499, 2004.

M. Elad, B. Matalon, and M. Zibulevsky. Image denoising with shrinkage and redundant representations. *Conference on Computer Vision and Pattern Recognition*, 2006.

G. Elidan, M. Ninio, N. Friedman, and D. Shuurmans. Data perturbation for escaping local maxima in learning. *National Conference on Artificial Intelligence*, 2002.

J. Fan and R. Li. Variable selection for Cox's proportional hazards model and frailty model. *Annals of Statistics*, 30(1):74–99, 2002.

J. Fan, Y. Feng, and Y. Wu. Network exploration via the adaptive LASSO and SCAD penalties. *Annals of Applied Statistics*, 3(2):521–541, 2009.

M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. *American Control Conference*, 2001.

M. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1050–1159, 2003.

M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1 (4):586–597, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

N. Friedman and Z. Yakhini. On the sample complexity of learning Bayesian networks. *Conference Uncertainty in Artificial Intelligence*, 1996.

N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. *Conference Uncertainty in Artificial Intelligence*, 1998.

N. Friedman, D. Pe'er, and I. Nachman. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. *Conference Uncertainty in Artificial Intelligence*, 1999.

W. Fu. Penalized regressions: the bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3):397–416, 1998.

E. Gafni and D. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22(6):936–964, 1982.

A. Gasch, P. Spellman, C. Kao, O. Carmel-Harel, M. Eisen, G. Storz, D. Botstein, and P. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular biology of the cell*, 11(12):4241–4257, 2000.

W. B. Gevarter. Automatic probabilistic knowledge acquisition from data. *International Conference on Data Engineering*, 1987.

P. Gill, W. Murray, and M. Wright. *Practical optimization*. Academic Press, 1981.

P. Giudici and R. Castelo. Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50(1-2):127–158, 2003.

P. Giudici and P. Green. Decomposable graphical Gaussian model determination. *Biometrika*, 86(4):785–801, 1999.

A. Goldstein. Convex programming in Hilbert space. *Bulletin of the American Mathematical Society*, 70(5): 709–710, 1964.

G. Golub and C. Van Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996.

J. Goodman. Exponential Priors for Maximum Entropy Models. *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, 2004.

L. Goodman. The analysis of multidimensional contingency tables: Stepwise procedures and direct estimation methods for building models for multiple classifications. *Technometrics*, 13(1):33–61, 1971.

L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for Newton's method. *SIAM Journal on Numerical Analysis*, 23(4):707–716, 1986.

Y. Guo and D. Schuurmans. Convex structure learning for Bayesian networks: Polynomial feature selection and approximate ordering. *Conference Uncertainty in Artificial Intelligence*, 2006.

M. Gustafsson, M. Hornquist, and A. Lombardi. Large-scale reverse engineering by the lasso. *International Conference on Systems Biology*, 2003.

E. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for ℓ_1-regularized minimization with applications to compressed sensing. Technical report, Rice University, 2007.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

D. Haughton. On the choice of a model to fit data from an exponential family. *Annals of Statistics*, 16(1): 342–355, 1988.

D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

D. Heckerman, C. Meek, and G. Cooper. A Bayesian approach to causal discovery. *Computation, causation, and discovery*, pages 141–165, 1999.

D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2001.

A. Hoerl and R. Kennard. Ridge regression: applications to nonorthogonal problems. *Technometrics*, 12(1): 69–82, 1970.

H. Hofling and R. Tibshirani. Estimation of Sparse Binary Pairwise Markov Networks using Pseudo-likelihoods. *Journal of Machine Learning Research*, 10:883–906, 2009.

J. Huang, N. Liu, M. Pourahmadi, and L. Liu. Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93:85–98, 2006.

G. Hulten and P. Domingos. Mining complex models from arbitrarily large databases in constant time. *International Conference on Knowledge Discovery and Data Mining*, 2002.

X. Huo and X. Ni. When do stepwise algorithms meet subset selection criteria? *Annals of Statistics*, 35(2): 870–887, 2007.
T. Jaakkola, D. Sontag, A. Globerson, and M. Meila. Learning Bayesian network structure using LP relaxations. *Conference on Artificial Intelligence and Statistics*, 2010.
L. Jacob, G. Obozinski, and J. Vert. Group Lasso with overlap and graph Lasso. *ICML*, 2009.
V. Johnson and J. Albert. *Ordinal data modeling*. Springer, 1999.
A. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.
D. Karger and N. Srebro. Learning Markov networks: Maximum bounded tree-width graphs. *ACM-SIAM Symposium on Discrete Algorithms*, 2001.
K. Koh, S. Kim, and S. Boyd. An interior-point method for large-scale l 1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.
M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.
M. Kolar and E. Xing. Improved Estimation of High-dimensional Ising Models. Technical report, Carnegie Mellon University, 2008.
D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.
V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *European Conference on Computer Vision*, 2002.
A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. *Conference Uncertainty in Artificial Intelligence*, 2005.
V. Krishnamurthy and A. d'Aspremont. A Pathwise Algorithm for Covariance Selection. *NIPS Workshop on Optimization for Machine Learning*, 2009.
B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, 2005.
H. Kushner and G. Yin. *Stochastic approximation and recursive algorithms and applications*. Springer, 2003.
J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learning*, 2001.
W. Lam and F. Bacchus. Using causal information and local measures to learn Bayesian networks. *Conference Uncertainty in Artificial Intelligence*, 1993.
P. Larrafiag, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996.
S. Lauritzen. *Graphical models*. Oxford University Press, USA, 1996.
S. Lauritzen and T. Richardson. Chain graph models and their causal interpretations. *Journal of the Royal Statistical Society: Series B*, 64(3):321–361, 2002.
H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. *Conference on Neural Information Processing Systems*, 2006a.
S. Lee, V. Ganapathi, and D. Koller. Efficient Structure Learning of Markov Networks using L_1-Regularization. *Conference on Neural Information Processing Systems*, 2006b.
S. Lee, H. Lee, P. Abbeel, and A. Ng. Efficient L_1 1 Regularized Logistic Regression. *National Conference on Artificial Intelligence*, 2006c.
E. Levina, A. Rothman, and J. Zhu. Sparse estimation of large covariance matrices via a nested Lasso penalty. *Annals of Applied Statistics*, 2(1):245–263, 2008.
E. Levitin and B. Poliak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6:1–50, 1966. English translation of a paper in Zh. Vychisl. Mat. i Mat. Fiz., vol. 6, pp. 787-823, 1965.
A. Lewis and M. Overton. Nonsmooth optimization via BFGS. *Optimization Online*, 2008.
F. Li and Y. Yang. Recovering genetic regulatory networks from micro-array data and location analysis

data. *International Conference on Genome Informatics*, 2004.
F. Li and Y. Yang. Using modified lasso regression to learn large undirected graphs in a probabilistic framework. *National Conference on Artificial Intelligence*, 2005.
P. Liang and M. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. *International Conference on Machine Learning*, 2008.
C. Lin, R. Weng, and S. Keerthi. Trust region newton method for logistic regression. *International Conference on Machine Learning*, 2007.
B. Lindsay. Composite likelihood methods. *Contemporary Mathematics*, 80(1):221–39, 1988.
J. Lindsey and P. Lindsey. Multivariate distributions with correlation matrices for nonlinear repeated measurements. *Computational Statistics and Data Analysis*, 50(3):720–732, 2006.
J. Liu, J. Chen, and J. Ye. Large-scale sparse logistic regression. *International Conference on Knowledge Discovery and Data Mining*, 2009.
Z. Lu. Smooth optimization approach for sparse covariance selection. *SIAM Journal on Optimization*, 19 (4):1807–1827, 2009.
Z. Lu. Adaptive first-order methods for general sparse inverse covariance selection. *SIAM Journal on Matrix Analysis and Applications (accepted)*, 2010.
D. Madigan and A. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546, 1994.
D. Madigan, S. Andersson, M. Perlman, and C. Volinsky. Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Communications in Statistics - Theory and Methods*, 25 (11):2493–2519, 1996.
R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. *International Conference On Computational Linguistics*, 2002.
F. Malvestuto. Approximating discrete probability distributions with decomposablemodels. *IEEE Transactions on systems, Man and Cybernetics*, 21(5):1287–1294, 1991.
D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. *Conference on Neural Information Processing Systems*, 1999.
B. Marlin, M. Schmidt, and K. Murphy. Group Sparse Priors for Covariance Estimation. *Conference Uncertainty in Artificial Intelligence*, 2009.
M. Meila and M. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.
N. Meinshausen and P. Buhlmann. High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.
T. Minka. Algorithms for maximum-likelihood logistic regression. Technical report, CMU, 2003.
B. Moghaddam, B. Marlin, M. Khan, and K. Murphy. Accelerating Bayesian Structural Inference for Non-Decomposable Gaussian Graphical Models. *Conference on Neural Information Processing Systems*, 2009.
A. Moore and W. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. *International Conference on Machine Learning*, 2003.
P. Munteanu and M. Bendou. The EQ framework for learning equivalence classes of Bayesian networks. *IEEE International Conference on Data Mining*, 2001.
K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.
I. Nachman, G. Elidan, and N. Friedman. "Ideal Parent" structure learning for continuous variable networks. *Conference Uncertainty in Artificial Intelligence*, 2004.
M. Narasimhan and J. Bilmes. PAC-learning bounded tree-width graphical models. *Conference Uncertainty in Artificial Intelligence*, 2004.
Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, 2004.
Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Université Catholique do Louvain, 2007.
A. Ng. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. *International Conference on Machine Learning*, 2004.

J. Nielsen, T. Kocka, and J. Pena. On local optima in learning Bayesian networks. *Conference Uncertainty in Artificial Intelligence*, 2003.
J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35(151): 773–782, 1980.
J. Nocedal and S. Wright. *Numerical optimization*. Springer, 1999.
M. Osborne, B. Presnell, and B. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000.
M. Park and T. Hastie. L_1 Regularization Path Algorithm for Generalized Linear Models. *Journal of the Royal Statistical Society: Series B*, 69(4):659–677, 2007.
J. Pearl. *Causality: Models, reasoning, and inference*. Cambridge Univ Press, 2000.
S. Perkins, K. Lacker, and J. Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.
E. Perrier, S. Imoto, and S. Miyano. Finding optimal Bayesian network given a super-structure. *Journal of Machine Learning Research*, 9:2251–2286, 2008.
M. Qazi, G. Fung, S. Krishnan, R. Rosales, H. Steck, R. Rao, D. Poldermans, and D. Chandrasekaran. Automated heart wall motion abnormality detection from ultrasound images using Bayesian networks. *IJCAI*, 2007.
A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An efficient projection for $l_{1,\infty}$ regularization. *International Conference on Machine Learning*, 2009.
M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization*, 7(1):26–33, 1997.
M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
T. Richardson and P. Spirtes. Ancestral graph Markov models. *Annals of Statistics*, 30(4):962–1030, 2002.
J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
R. Robinson. Counting unlabeled acyclic digraphs. *Australian Conference on Combinatorial Mathematics*, 1976.
S. Rosset. Tracking curved regularized optimization solution paths. *Conference on Neural Information Processing Systems*, 2004.
V. Roth. The generalized LASSO. *IEEE Transactions on Neural Networks*, 15(1):16–28, 2004.
H. Rue and L. Held. *Gaussian Markov random fields: theory and applications*. Chapman & Hall, 2005.
S. Rusell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.
K. Sachs, O. Perez, D. Pe'er, D. Lauffenburger, and G. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.
F. Santosa and W. Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.
S. Sardy, A. Bruce, and P. Tseng. Block coordinate relaxation methods for nonparametric wavelet denoising. *Journal of Computational and Graphical Statistics*, 9(2):361–379, 2000.
L. Saul, T. Jaakkola, and M. Jordan. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4:61–76, 1996.
M. Schmidt and K. Murphy. Modeling Discrete Interventional Data using Directed Cyclic Graphical Models. *Conference Uncertainty in Artificial Intelligence*, 2009.
M. Schmidt and K. Murphy. Convex Structure Learning in Log-Linear Models: Beyond Pairwise Potentials. *Conference on Artificial Intelligence and Statistics*, 2010.
M. Schmidt, G. Fung, and R. Rosales. Fast optimization methods for L1 regularization: A comparative study and two new approaches. *European Conference on Machine Learning*, 2007a.
M. Schmidt, A. Niculescu-Mizil, and K. Murphy. Learning graphical model structure using l1-regularization paths. *National Conference on Artificial Intelligence*, 2007b.
M. Schmidt, K. Murphy, G. Fung, and R. Rosales. Structure learning in random fields for heart motion abnormality detection. *Conference on Computer Vision and Pattern Recognition*, 2008.
M. Schmidt, G. Fung, and R. Rosales. Optimization methods for ℓ_1-regularization. Technical report, University of British Columbia, 2009a.

M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. *Conference on Artificial Intelligence and Statistics*, 2009b.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, 2003.

D. Shahaf, A. Chechetka, and C. Guestrin. Learning thin junction trees via graph cuts. *Conference on Artificial Intelligence and Statistics*, 2009.

D. Shanno and K. Phua. Matrix conditioning and nonlinear optimization. *Mathematical Programming*, 14 (1):149–160, 1978.

T. Shimamura, S. Imoto, R. Yamaguchi, and S. Miyano. Weighted lasso in graphical Gaussian modeling for large gene network estimation based on microarray data. *International Conference on Genome informatics*, 2007.

S. Shimizu, A. Hyvarinen, Y. Kano, and P. Hoyer. Discovery of non-gaussian linear causal models using ICA. *Conference Uncertainty in Artificial Intelligence*, 2005.

M. Singh and M. Valtorta. An algorithm for the construction of Bayesian network structures from data. *Conference Uncertainty in Artificial Intelligence*, 1993.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72, 1991.

P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. *International Conference on Knowledge Discovery and Data Mining*, 1995.

N. Srebro. Maximum likelihood bounded tree-width Markov networks. *Artificial intelligence*, 143(1):123–138, 2003.

S. Srinivas, S. Russell, and A. Agogino. Automated construction of sparse Bayesian networks from unstructured probabilistic models and domain information. *Conference Uncertainty in Artificial Intelligence*, 1990.

H. Steck. On the use of skeletons when learning in bayesian networks. *Conference Uncertainty in Artificial Intelligence*, 2000.

J. Suzuki. Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique. *IEICE Transactions on Information and Systems*, 82:356–367, 1999.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. *Conference on Neural Information Processing Systems*, 2003.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative Markov networks. *International Conference on Machine Learning*, 2004.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. *Conference Uncertainty in Artificial Intelligence*, 2005.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1):267–288, 1996.

I. Tsamardinos, L. Brown, and C. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

B. Turlach, W. Venables, and S. Wright. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

E. van den Berg. *Convex optimization for generalized sparse recovery.* PhD thesis, UBC, 2010.

E. van den Berg and M. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

E. van den Berg, M. Schmidt, and K. M. M. Friedlander. Group sparsity via linear-time projection. Technical report, University of British Columbia, 2008.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. *Conference Uncertainty in Artificial Intelligence*, 1990.

D. Vidaurre, C. Bielza, and P. Larranaga. Learning a L1-regularized Gaussian Bayesian network in the space of equivalence classes. *IEEE Transactions on Systems, Man and Cybernetics: Part B*, 2010.

J. von Neumann. *Functional Operators, vol. II, The Geometry of Orthogonal Spaces*, volume 22 of *Annals of Mathematical Studies*. Princeton University Press, 1950. This is a reprint of notes first distributed in 1933-34.

M. Wainwright. Estimating the "Wrong" Graphical Model: Benefits in the Computation-Limited Setting. *Journal of Machine Learning Research*, 7:1829–1859, 2006.

M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

M. Wainwright, T. Jaakkola, and A. Willsky. A new class of upper bounds on the log partition function. *Conference Uncertainty in Artificial Intelligence*, 2002.

M. Wainwright, P. Ravikumar, and J. Lafferty. High-Dimensional Graphical Model Selection Using ℓ_1-Regularized Logistic Regression. *Conference on Neural Information Processing Systems*, 2006.

H. Wallach. Efficient training of conditional random fields. Master's thesis, University of Edinbrugh, 2002.

N. Wermuth. Model search among multiplicative models. *Biometrics*, 32(2):253–263, 1976.

J. Whittaker. *Graphical models in applied multivariate analysis*. John Wiley and Sons, 1990.

P. Williams. Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7(1):117–143, 1995.

D. Wipf and S. Nagarajan. Sparse Estimation Using General Likelihoods and Non-Factorial Priors. *Conference on Neural Information Processing Systems*, 2009.

S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

Y. Xiang, S. Wong, and N. Cercone. A "microscopic study of minimum entropy search in learning decomposable markov networks. *Machine Learning*, 26(1):65–92, 1997.

L. Younes. Parameter estimation for imperfectly observed Gibbsian fields. *Probability Theory and Related Fields*, 82:625–645, 1989.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1): 19–35, 2007.

X. Yuan. Alternating direction methods for sparse covariance selection. Technical report, Hong Kong Baptist University, 2009.

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7: 2541–2563, 2006.

P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*, 37(6A):3468–3497, 2009.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101 (476):1418–1429, 2006.

H. Zou and R. Li. One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics*, 36(4):1509–1533, 2008.

H. Zou, T. Hastie, and R. Tibshirani. On the "degrees of freedom" of the lasso. *Annals of Statistics*, 35(5): 2173–2192, 2007.

Appendix A

Data Structures for Checking Acyclicity

The following material is needed for the fast of implementation of acyclicity checks in the DAG-search method used in Chapter 4. Giudici and Castelo [2003] propose using an ancestor matrix data structure to efficiently test whether local moves preserve acyclicity, and they give procedures for updating the ancestor matrix. However, the authors do not give a procedure for constructing the ancestor matrix given a graph, while the analysis of the runtimes of the updates is not correct. In this section, we give an efficient procedure for building an ancestor matrix given a DAG, review the ancestor matrix update rules and their runtimes, present several special cases that lead to faster updates of the ancestor matrix, and present the reversal witness matrix data structure that allows us to quickly check whether reversals will introduce a cycle.

A.1 Ancestor Matrix

The ancestor matrix for a DAG with n nodes is an n by n binary matrix, that we will denote by A. We set element A_{ij} of the matrix to 1 if i is an ancestor of j in the graph, meaning that there is a directed path from i to j. Otherwise, we set A_{ij} to 0 (by convention, we set $A_{ii} = 0$). In this section we will express the runtimes of all operations in terms of n, since it is possible that we will need to use the data structure on a maximally connected directed acyclic graph.

Given the ancestor matrix for a directed acyclic graph, it is trivial to test whether adding a new edge will introduce a cycle. To see this, note that the graph is acyclic before introducing the new edge, so if a cycle is introduced the new edge must be part of the cycle. We can thus test whether a new edge from i to j introduces a cycle by simply testing whether j is an ancestor of i;

Input: Ancestor matrix A, edge (i, j) to test.
Output: Returns 1 if adding (i,j) will cause a cycle
return A_{ji};

Algorithm 13: Using an ancestor matrix to test whether an addition preserves acyclicity.

If we want to test whether each of the $\mathcal{O}(n^2)$ possible edges will introduce a cycle, with the ancestor matrix we can do this in $\mathcal{O}(n^2)$. This is substantially more efficient than the $\mathcal{O}(n^4)$ cost of naively checking each single-edge-augmented graph independently with an $\mathcal{O}(n^2)$ search.

Above, we assume that the ancestor matrix is given. However, it will only lead to a net computational gain if we can efficiently construct it from a given graph, and efficiently update it after single edge changes. Below, we make use of topological sorting to give an efficient procedure

for constructing an ancestor matrix.

```
Input: Graph G
Output: A valid ancestor matrix A for G
initialize all elements of A to zero;
find a topological ordering of G;
foreach node c in order do
    foreach parent p of c do
        A_pc ← 1 ;
        foreach ancestor a of p do
            A_ac ← 1;
```

Algorithm 14: Constructing an ancestor matrix.

The topological sort can be done in $\mathcal{O}(n^2)$ [Kahn, 1962, Cormen et al., 2001, §22.4], while constructing the ancestor matrix by computing the ancestors of each node in topological order requires at most $\mathcal{O}(n^3)$ (it is possible that the total cost could be reduced to $\mathcal{O}(n^2)$, the size of the structure). Below, we give the procedure for updating the ancestor matrix after a single edge addition.

```
Input: Ancestor matrix A and edge (p, c) to add
Output: A valid ancestor matrix A with (p, c) added
if A_pc = 1 then
    return ; // fast update, p was already an ancestor of c
A_pc ← 1 ; // p is now an ancestor of c
foreach descendant d of c do
    A_pd ← 1 ; // p is now an ancestor of all descendants of c
foreach ancestor a of p do
    A_ac ← 1 ; // ancestors of p are now ancestors of c
    foreach descendant d of c do
        A_ad ← 1 ; // ancestors of p are now ancestors of descendants of c
```

Algorithm 15: Updating an ancestor matrix after an addition.

Similar to finding all ancestors of a node by looking at the corresponding column of the ancestor matrix, we can find all descendants of a node by looking at the corresponding row. The above procedure require $\mathcal{O}(n^2)$ in the worst-case, due to the need to update the up to $\mathcal{O}(n^2)$ members of the product of ancestors and descendants of the two nodes (the runtime was incorrectly state as $\mathcal{O}(n)$ by Giudici and Castelo [2003]). Note the line marked as *fast update*; if p is already an ancestor of c when we add the edge (p, c), then no new ancestor relations can arise and the update only costs $\mathcal{O}(1)$. Also note that if we constructed the ancestor matrix for a given graph by repeatedly applying the addition algorithm (starting from an empty graph), that this would require $\mathcal{O}(n^4)$. Thus, our topological ordering construction algorithm is more efficient than this naive method.

Next, we give a procedure for updating the ancestor matrix after deleting an edge. The method

below is called after an edge (p, c) has just been removed from the graph G:

Input: Graph G, ancestor matrix A, and edge (p, c) to delete
Output: A valid ancestor matrix A with (p, c) deleted
foreach *parent* p^* *of* c **do**
  if $A_{pp^*} = 1$ **then**
    return ; `// fast update,` c `is still a descendant of` p
find a topological ordering of G;
foreach *node* j *in order starting from* c **do**
  $A_{kj} \to 0, \forall_k$; `// clear ancestors of` j
foreach *node* j *in order starting from* c **do**
  foreach *parent* i *of* j **do**
    $A_{ij} \leftarrow 1$;
    foreach *ancestor* a *of* i **do**
      $A_{aj} \leftarrow 1$;

Algorithm 16: Updating an ancestor matrix after a deletion.

In the worst case, the above procedure will cost $\mathcal{O}(n^3)$ since we may need to rebuild the ancestor matrix for most of the graph. However, the update will only require up to $\mathcal{O}(n)$ in the case marked *fast update*. In this case, p remains an ancestor of c after deleting (p, c) so the ancestor relationships do not change. To update the ancestor matrix after a reversal, we call the above deletion procedure followed by the addition procedure.

A.2 Reversal Witness Matrix

By similar reasoning to the addition case, reversing an existing edge (i, j) in a directed acyclic graph will introduce a cycle if and only if i is an ancestor of some ancestor of j. In other words, if some descendant of i is an ancestor of j, then reversing the edge from i to j will introduce a path from j to itself via the newly reversed edge. We make this more formal below.

Input: Ancestor matrix A, edge (i, j) to test.
Output: Returns 1 if reversing (i,j) will cause a cycle
foreach *ancestor* a *of* j **do**
  if $A_{ia} = 1$ **then**
    return 1;
return 0;

Algorithm 17: Using an ancestor matrix to test whether a reversal preserves acyclicity.

Since we have the current ancestor matrix available, we can easily find the up to $\mathcal{O}(n)$ ancestors of j. Since we do a constant amount of work for each ancestor the procedure checks whether reversing (i, j) will introduce a cycle in at most $\mathcal{O}(n)$. Subsequently, we can check all edges in $\mathcal{O}(n^3)$.

We now outline a data structure that we refer to as a *reversal witness* matrix. In this context, we say that a *witness* exists for an edge (i, j) if some descendant of i is an ancestor of j. Thus, reversing an edge will cause a cycle if and only if a witness exists. The reversal witness matrix is simply an n by n sparse binary matrix that is set to 1 if the edge from i to j exists and has a

witness. We consider the following simple procedure for constructing the reversal witness matrix given a graph. It assumes that the above procedure for testing a reversal given the ancestor matrix is available, and can be called to test an edge (i, j) with the interface testReversal(A,p,c).

Input: Graph G and ancestor matrix A
Output: A valid reversal witness matrix R for G
initialize all elements of R to zero;
foreach *edge* (i, j) *in* G **do**
  $R(i, j) \leftarrow$ testReversal(A,p,c);

Algorithm 18: Constructing a reversal witness matrix.

Above, we can construct the reversal witness matrix in $\mathcal{O}(n^3)$ by simply using the ancestor matrix to check whether each edge of the $\mathcal{O}(n^2)$ edges can be reversed. We now give a procedure for updating the reversal witness matrix for a single edge (p, c) after the addition of the edge (i, j).

Input: Reversal witness matrix R, ancestor matrix A, an edge (p, c) to update after adding edge (i, j)
Output: A valid reversal witness matrix R with (p, c) added
if $R_{pc} = 1$ **then**
  return ; `// fast update, this edge already has a witness`
if *(p is not i or an ancestor of i) and (c is not j or a descendant of j)* **then**
  return ; `// fast update,` p `has no new descendants,` c `has no new ancestors`
$R(i, j) \leftarrow$ testReversal(A,p,c);

Algorithm 19: Updating a reversal witness matrix after an addition.

In the above, the fast updates require $\mathcal{O}(1)$ while the slow updates require $\mathcal{O}(n)$. Thus, this data structure is more efficient than using the ancestor matrix alone whenever a fast update is performed. We update the reversal witness matrix after a deletion by rebuilding it.

Appendix B

Projection onto Norm Cones

When applied to group ℓ_1-regularization problems, the SPG and PQN methods discussed in Chapter 3 employ the operation of projecting onto a norm cone. That is, for a given $\mathbf{x}_0$ and g_0 they compute the projection

$$\mathcal{P}_{\mathcal{C}_p}(\mathbf{x}_0, g_0) \triangleq \arg\min_{||\mathbf{x}||_p \leq g} \left|\left|\begin{bmatrix} \mathbf{x} \\ g \end{bmatrix} - \begin{bmatrix} \mathbf{x}_0 \\ g_0 \end{bmatrix}\right|\right|_2.$$

for the given norm $||\cdot||_p$. By non-negativity of norms, we can equivalently solve

$$\arg\min_{\mathbf{x},g} \frac{1}{2}||\mathbf{x} - \mathbf{x}_0||_2^2 + \frac{1}{2}(g - g_0)^2, \;\; \text{s.t.} \;\; ||\mathbf{x}||_p \leq g. \tag{B.1}$$

In this appendix, we give simple algorithms for solving (B.1) for different norms examined in this work.

B.1 Scalar Norm

We first consider the one-dimensional case where we simply have a scalar x. In this case, problem (B.1) can be written as

$$\arg\min_{x,g} \frac{1}{2}(x - x_0)^2 + \frac{1}{2}(g - g_0)^2, \;\; \text{s.t.} \;\; |x| \leq g. \tag{B.2}$$

In this case, the projection onto the scalar norm cone $\mathcal{C}_a$ is given by

$$\mathcal{P}_{\mathcal{C}_a}(x_0, g_0) = \begin{cases} (x_0, g_0), & \text{if } |x_0| \leq g_0, \\ (\text{sign}(x_0)\frac{|x_0|+g_0}{2}, \frac{|x_0|+g_0}{2}), & \text{if } |x_0| > g_0, |x_0| + g_0 > 0, \\ (0, 0), & \text{if } |x_0| > g_0, |x_0| + g_0 \leq 0. \end{cases} \tag{B.3}$$

Proof. If $|x_0| \leq g_0$, then the first case follows because $x = x_0$ and $g = g_0$ satisfy the constraints and achieve the minimum possible objective value of zero in (B.2). Thus, it remains to show the other two cases and below we assume that $|x_0| > g_0$.

First, note that $|x| \geq 0$ implies that in a solution (x^*, g^*) we must have that $g^* \geq 0$. Further, x^* can not have the opposite sign to x_0: $x^* x_0 \geq 0$. To show this, assume that $x^* x_0 < 0$. Then

$$\begin{aligned} \frac{1}{2}(x^* - x_0)^2 + \frac{1}{2}(g^* - g_0)^2 &= \frac{1}{2}(x^*)^2 - x^* x_0 + x_0^2 + \frac{1}{2}(g^* - g_0)^2 \\ &> \frac{1}{2}(x^*)^2 + x_0^2 + \frac{1}{2}(g^* - g_0)^2 \\ &> x_0^2 + \frac{1}{2}(g^* - g_0)^2 \\ &= (0 - x_0)^2 + \frac{1}{2}(g^* - g_0)^2. \end{aligned}$$

This would imply that $(0, g^*)$ achieves a lower objective value than (x^*, g^*), and since $g^* \geq 0$ we obtain a contradiction.

We can similarly show that $|x^*| \leq |x_0|$, since if $|x^*| > |x_0|$ and $g^* \geq |x^*|$ then $(|x_0|, g^*)$ would achieve a lower objective value while remaining feasible. Further, using that $|x^*| \leq |x_0|$ and $g_0 < |x_0|$ we can similarly show that $g^* \leq |x_0|$, since if $g^* > |x_0|$ then $(x^*, |x_0|)$ would achieve a lower objective (while remaining feasible).

We now establish the second two cases of (B.3) under the assumption that $x_0 \geq 0$. Since we know $x^* x_0 \geq 0$ this implies $x^* \geq 0$ and we can re-write (B.2) as

$$\arg\min_{x,g} \frac{1}{2}(x - x_0)^2 + \frac{1}{2}(g - g_0)^2, \;\; \text{s.t.} \;\; 0 \leq x \leq g.$$

Ignoring the trivial case where $x_0 < g_0$, in a solution of this problem it must be the case that $x^* = g^*$. To see this, assume that $g^* > x^*$. Then we can increase x^* to improve the objective function since we know that $g^* \leq x_0$. We use that $x^* = g^*$ to eliminate x and obtain the simple problem

$$\arg\min_{g} \frac{1}{2}(g - x_0)^2 + \frac{1}{2}(x - g_0)^2, \;\; \text{s.t.} \;\; g \geq 0.$$

Introducing a Lagrange multiplier $\mu \geq 0$ for the inequality constraint the Lagrangian of this problem is

$$\frac{1}{2}(g - x_0)^2 + \frac{1}{2}(g - g_0)^2 - \mu g.$$

Setting the derivative of the Lagrangian to zero we have that

$$0 = g - x_0 + g - g_0 - \mu.$$

From this we obtain that

$$g^* = \frac{x_0 + g_0 + \mu}{2},$$

for some $\mu \geq 0$. By complementary slackness we must have $g^* \mu = 0$. If $g^* = 0$, this implies that $\mu = -x_0 - g_0$, which can only be positive if $x_0 + g_0 \leq 0$. This establishes the third case of (B.3) (when $x_0 \geq 0$). Otherwise we have $\mu = 0$ and

$$g^* = \frac{x_0 + g_0}{2}.$$

We can use a simlar argument to show that we obtain the same result but with x_0 replaced by $|x_0|$ and $x^* = -g^*$ when we have $x_0 < 0$. □

B.2 ℓ_2 Norm

We next consider projecting onto the Euclidean norm cone. In this case, we can write problem (B.1) as

$$\arg\min_{\mathbf{x},g} \frac{1}{2}||\mathbf{x} - \mathbf{x}_0||_2^2 + \frac{1}{2}(g - g_0)^2, \;\; \text{s.t.} \;\; ||\mathbf{x}||_2 \leq g. \tag{B.4}$$

The projection onto the Euclidean norm cone $\mathcal{C}_2$ is given by [Boyd and Vandenberghe, 2004, Exercise 8.3(c)]

$$\mathcal{P}_{\mathcal{C}_2}(\mathbf{x}_0, g_0) = \begin{cases} (\mathbf{x}_0, g_0), & \text{if } ||\mathbf{x}_0||_2 \leq g_0, \\ (\frac{\mathbf{x}_0}{||\mathbf{x}_0||_2} \frac{||\mathbf{x}_0||_2 + g_0}{2}, \frac{||\mathbf{x}_0||_2 + g_0}{2}), & \text{if } ||\mathbf{x}_0||_2 > g, ||\mathbf{x}_0||_2 + g_0 > 0, \\ (0, 0), & \text{if } ||\mathbf{x}_0||_2 > g, ||\mathbf{x}_0||_2 + g_0 \leq 0. \end{cases} \tag{B.5}$$

Proof. We first establish that in an optimal solution $(\mathbf{x}^*, g^*)$ of (B.4), that $\mathbf{x}^*$ is in the same direction as $\mathbf{x}_0$. To do this, assume we can write $\mathbf{x}^* = \alpha\mathbf{x}_0 + \mathbf{y}$, where α is a scalar and $\mathbf{y}$ is a non-zero vector containing the part of $\mathbf{x}^*$ that is orthogonal to $\mathbf{x}_0$. Then we have that

$$\begin{aligned}
\frac{1}{2}||\mathbf{x}^* - \mathbf{x}_0||_2^2 + \frac{1}{2}(g^* - g_0)^2 &= \frac{1}{2}||(\alpha\mathbf{x}_0 + \mathbf{y}) - \mathbf{x}_0||_2^2 + \frac{1}{2}(g^* - g_0)^2 \\
&= \frac{1}{2}||(\alpha - 1)\mathbf{x}_0 + \mathbf{y}||_2^2 + \frac{1}{2}(g^* - g_0)^2 \\
&= \frac{1}{2}|\alpha - 1|^2||\mathbf{x}_0||_2^2 + (\alpha - 1)\mathbf{x}_0^T\mathbf{y} + \frac{1}{2}||\mathbf{y}||_2^2 + \frac{1}{2}(g^* - g_0)^2 \\
&= \frac{1}{2}|\alpha - 1|^2||\mathbf{x}_0||_2^2 + \frac{1}{2}||\mathbf{y}||_2^2 + \frac{1}{2}(g^* - g_0)^2 \\
&> \frac{1}{2}|\alpha - 1|^2||\mathbf{x}_0||_2^2 + \frac{1}{2}(g^* - g_0)^2 \\
&= \frac{1}{2}||(\alpha - 1)\mathbf{x}_0||_2^2 + \frac{1}{2}(g^* - g_0)^2 \\
&= \frac{1}{2}||\alpha\mathbf{x}_0 - \mathbf{x}_0||_2^2 + \frac{1}{2}(g^* - g_0)^2.
\end{aligned}$$

Since $\mathbf{y} \neq \mathbf{0}$, this implies that $(\alpha\mathbf{x}_0, g^*)$ achieves a lower objective value than $\mathbf{x}^*$ while it is feasible due to the feasilibility of $(\alpha\mathbf{x}_0 + \mathbf{y}, g^*)$ and orthogonality of $\mathbf{x}_0$ and $\mathbf{y}$. This establishes that $\mathbf{y}$ must be zero and that $\mathbf{x}^* = \alpha\mathbf{x}_0$ for some α. By a similar argument to the scalar case we can show that $x_i^*(\mathbf{x}_0)_i \geq 0$ for all i, so we have that $\alpha \geq 0$.

We next review a basic identity, that if $\mathbf{x} = \alpha\mathbf{x}_0$ for some scalar $\alpha \geq 0$ then

$$\begin{aligned}
||\mathbf{x} - \mathbf{x}_0||_2^2 &= ||\alpha\mathbf{x}_0 - \mathbf{x}_0||_2^2 \\
&= \alpha^2\mathbf{x}_0^T\mathbf{x}_0 - 2\alpha\mathbf{x}_0^T\mathbf{x}_0 + \mathbf{x}_0^T\mathbf{x}_0 \\
&= (\alpha||\mathbf{x}_0||_2 - ||\mathbf{x}_0||_2)^2 \\
&= (||\mathbf{x}||_2 - ||\mathbf{x}_0||_2)^2.
\end{aligned}$$

We can use this identity to re-write (B.4) as

$$\arg\min_{\mathbf{x},g} \frac{1}{2}(||\mathbf{x}||_2 - ||\mathbf{x}_0||_2)^2 + \frac{1}{2}(g - g_0)^2, \;\; \text{s.t.} \;\; ||\mathbf{x}||_2 \leq g.$$

Except in the trivial first case of (B.5), by similar reasoning to the scalar case we will have that $g^* = ||\mathbf{x}^*||_2$ in the solution of this problem. Thus, we can eliminate $||\mathbf{x}||_2$ to give the much simpler problem

$$\arg\min_{\mathbf{x},g} \frac{1}{2}(g - ||\mathbf{x}_0||_2)^2 + \frac{1}{2}(g - g_0)^2, \;\; \text{s.t.} \;\; g \geq 0.$$

This is identical to the scalar case for $x_0 \geq 0$ but with x_0 replaced by the non-negative scalar $||\mathbf{x}_0||_2$. We can thus derive the optimal g^* in (B.5) from the same argument used in the previous proof. In the case where $g^* > 0$, the constraint that $||\mathbf{x}^*||_2 = g^*$ along with knowing that $\mathbf{x}^*$ is in the direction of $\mathbf{x}_0$ imply that $\mathbf{x}^* = (\mathbf{x}_0/||\mathbf{x}_0||_2)g^*$. □

B.3 ℓ_∞ Norm

We next consider projecting onto the ℓ_∞ norm cone. We first concentrate on the case where $\mathbf{x}_0$ is a 2-vector with $(\mathbf{x}_0)_1 \geq (\mathbf{x}_0)_2 \geq 0$. In this case, we can write problem (B.1) as

$$\arg\min_{x_1,x_2,g} \frac{1}{2}(x_1 - (\mathbf{x}_0)_1)^2 + \frac{1}{2}(x_2 - (\mathbf{x}_0)_2)^2 + \frac{1}{2}(g - g_0)^2, \text{ s.t. } g \geq x_1 \geq 0, g \geq x_2 \geq 0. \tag{B.6}$$

The solution of this special case of projecting onto the ℓ_∞ norm cone is given by

$$\mathcal{P}_{\mathcal{C}_\infty}((\mathbf{x}_0)_1, (\mathbf{x}_0)_2, g_0) = \tag{B.7}$$

$$\begin{cases} ((\mathbf{x}_0)_1, (\mathbf{x}_0)_2, g_0), & \text{if } ||\mathbf{x}_0||_\infty \leq g_0, \\ (\frac{(\mathbf{x}_0)_1+g_0}{2}, (\mathbf{x}_0)_2, \frac{(\mathbf{x}_0)_1+g_0}{2}), & \text{if } ||\mathbf{x}_0||_\infty > g_0, \frac{(\mathbf{x}_0)_1+g_0}{2} > (\mathbf{x}_0)_2, \\ (\frac{(\mathbf{x}_0)_1+(\mathbf{x}_0)_2+g_0}{3}, \frac{(\mathbf{x}_0)_1+(\mathbf{x}_0)_2+g_0}{3}, \frac{(\mathbf{x}_0)_1+(\mathbf{x}_0)_2+g_0}{3}), & \text{if } ||\mathbf{x}_0||_\infty > g_0, \frac{(\mathbf{x}_0)_1+g_0}{2} \leq (\mathbf{x}_0)_2, \frac{(\mathbf{x}_0)_1+(\mathbf{x}_0)_2+g_0}{3} > 0 \\ (0,0), & \text{if } ||\mathbf{x}_0||_\infty > g_0, \frac{(\mathbf{x}_0)_1+g_0}{2} \leq (\mathbf{x}_0)_2, \frac{(\mathbf{x}_0)_1+(\mathbf{x}_0)_2+g_0}{3} \leq 0 \end{cases} \tag{B.8}$$

Proof. We start by noting that if the inputs satisfy the constraints then we once again simply return the inputs in the first case. If this is not the case, then a similar argument to the scalar case shows that in an optimal solution (x_1^*, x_2^*, g^*) it must be the case that $x_1^* = g^*$. Subsequently, we can (as before) eliminate x_1 from (B.6) to give the problem

$$\arg\min_{x_2,g} \frac{1}{2}(g - (\mathbf{x}_0)_1)^2 + \frac{1}{2}(x_2 - (\mathbf{x}_0)_2)^2 + \frac{1}{2}(g - g_0)^2, \text{ s.t. } g \geq 0, g \geq x_2 \geq 0.$$

Since $(\mathbf{x}_0)_2 \geq 0$ and we require $g \geq x_2$, the constraint $x_2 \geq 0$ will be satisfied at a solution even if it is not included explicitly, so we remove it and write the Lagrangian for this problem is

$$\frac{1}{2}(g - (\mathbf{x}_0)_1)^2 + \frac{1}{2}(x_2 - (\mathbf{x}_0)_2)^2 + \frac{1}{2}(g - g_0)^2 - \mu_1 g + \mu_2(x_2 - g)$$

At a solution we require that the gradient of the Lagrangian with respect to both g and x_2 is equal to zero:

$$\begin{aligned} 0 &= (g - (\mathbf{x}_0)_1 + g - g_0 - \mu_1) - \mu_2 \\ 0 &= x_2 - (\mathbf{x}_0)_2 + \mu_2 \end{aligned}$$

Note that the first term in the first equation is the gradient of the Lagrangian for the problem of projecting onto the scalar norm cone. If we use $(\tilde{x}_1, \tilde{g})$ to denote the result of projecting $((\mathbf{x}_0)_1, g_0)$ onto the scalar norm cone, then the first term in the first equation is zero at $(\tilde{x}_1, x_2, \tilde{g})$ for any x_2. Thus, If it happens to be the case that $\tilde{g} > (\mathbf{x}_0)_2$, then all constraints are satisfied and complementary slackness implies that $\mu_2 = 0$ so the solution to the problem is $(\tilde{x}_1, (\mathbf{x}_0)_2, \tilde{g})$. This establishes the second case of (B.8).

If $\tilde{g} \leq (\mathbf{x}_0)_2$, then we can show that $x_1^* = x_2^* = g^*$. Thus, we can eliminate both x_1 and x_2 and write the optimization as a bound constrained optimization in g. Solving this problem as we did in the scalar case yields the third and fourth cases of (B.8). □

Although the result above only applies to a very restricted scenario, we can generalize it to compute the general ℓ_∞ norm cone projection. In particular, we can clearly remove the restriction $(\mathbf{x}_0)_1 \geq (\mathbf{x}_0)_2$ by sorting the elements of $\mathbf{x}_0$ before projecting. We can further remove the constraint that $(\mathbf{x}_0)_1$ and $(\mathbf{x}_0)_2$ are non-negative by projecting their absolute values and then assigning the appropriate signs to the results. Finally, we can use an inductive argument to generalize the result to arbitrary p-vectors. Below, we give pseudo-code for a general method that requires $\mathcal{O}(p \log p)$ time (due to the need to sort $\mathbf{x}_0$).

Input: Scalar g and p-vector $\mathbf{x}$
if $g \geq ||\mathbf{x}||_\infty$ **then**
  return; `// input value satisfies constraints`
sorted $\leftarrow \{\text{sort}(|\mathbf{x}|), 0\}$; `// sort absolute values in descending order, append zero`
$s \leftarrow 0$;
for $k \leftarrow 1$ *to* p **do**
  $s \leftarrow s + \text{sorted}(k)$;
  $\alpha \leftarrow (s + g)/(k + 1)$; `// trial value for` g
  if $\alpha > 0$ *and* $\alpha <$ *sorted*$(k + 1)$ **then**
    for $i \leftarrow 1$ *to* p **do**
      $x_i \leftarrow \text{sign}(x_i) \min\{|x_i|, \alpha\}$; `// threshold values.`
    $g \leftarrow \alpha$;
    return;
$\mathbf{x} \leftarrow \mathbf{0}$;
$g \leftarrow 0$;

Algorithm 20: Projection onto ℓ_∞ norm cone.

B.4 ℓ_1 Norm

We now turn to the task of projecting onto the ℓ_1 norm cone. We first concentrate on the case where $\mathbf{x}_0$ only contains non-negative elements. In this case, we can write problem (B.1) as

$$\arg\min_{\mathbf{x},g} \frac{1}{2}||\mathbf{x} - \mathbf{x}_0||_2^2 + \frac{1}{2}(g - g_0)^2, \text{ s.t. } \sum_{i=1}^{p} x_i \leq g, \mathbf{x} \geq \mathbf{0}. \tag{B.9}$$

The solution of this special case of projecting onto the ℓ_1 norm cone is given by

$$\mathcal{P}_{\mathcal{C}_1}(\mathbf{x}_0, g_0) = (\max\{0, \mathbf{x}_0 - \theta\}, g_0 + \theta), \tag{B.10}$$

where the max operation is done element-wise and where $\theta \geq 0$ is the minimum (scalar) value such that the constraints are satisfied.

Proof. The Lagrangian for problem (B.9) is

$$\frac{1}{2}||\mathbf{x} - \mathbf{x}_0||_2^2 + \frac{1}{2}(g - g_0)^2 + \theta(\sum_{i=1}^{p} x_i - g) - \mathbf{y}^T\mathbf{x},$$

with a scalar Lagrange multiplier θ for the sum constraint and a vector of Lagrange multipliers $\mathbf{y}$ for the non-negativity constraints. Setting the gradient of the Lagrangian with respect to g to zero we obtain

$$0 = g - g_0 - \theta.$$

From this we obtain that the optimal g^* has the form

$$g^* = g_0 + \theta. \tag{B.11}$$

Setting the gradient of the Lagrangian with respect to $\mathbf{x}$ to zero we obtain

$$\mathbf{0} = \mathbf{x} - \mathbf{x}_0 + \theta - \mathbf{y}.$$

From this we obtain that the optimal $\mathbf{x}^*$ has the form

$$\mathbf{x}^* = \mathbf{x}_0 - \theta + \mathbf{y}.$$

By complementary slackness, we have that $x_i^* y_i = 0$ for all i. If $y_i = 0$ then we have

$$x_i^* = (\mathbf{x}_0)_i - \theta.$$

Similarly, if $x_i^* = 0$ we have that

$$y_i = -(\mathbf{x}_0)_i + \theta.$$

Since we require $y_i \geq 0$, we see that x_i^* can be zero only if $(\mathbf{x}_0)_i - \theta < 0$. Combining both cases, we have that x_i^* has the form

$$x_i^* = \max\{0, (\mathbf{x}_0)_i - \theta\}. \tag{B.12}$$

We have now established the form of (B.10), and it remains to show that we must find the minimum $\theta \geq 0$. Using (B.11) and (B.12) to eliminate $\mathbf{x}$ and g in (B.9), we obtain

$$\arg\min_{\theta} \frac{1}{2}||\max\{0, \mathbf{x}_0 - \theta\} - \mathbf{x}_0||_2^2 + \frac{1}{2}(g_0 + \theta - g_0)^2, \;\; \text{s.t.} \;\; \sum_{i=1}^{p} \max\{0, (\mathbf{x}_0)_i - \theta\} \leq g_0 + \theta, \theta \geq 0.$$

We can simplify the objective function in this expression to give

$$\arg\min_{\theta} \frac{1}{2}||\min\{\mathbf{x}_0, \theta\}||_2^2 + \frac{1}{2}\theta^2, \;\; \text{s.t.} \;\; \sum_{i=1}^{p} \max\{0, (\mathbf{x}_0)_i - \theta\} \leq g_0 + \theta, \theta \geq 0.$$

where the min operation is done element-wise. We see that for $\theta \geq 0$ that the first term in this objective function is monotonically increasing in θ while the second term is *strictly* monotonically increasing in θ. Thus, the constrained minimizer of this objective function is the minimum $\theta \geq 0$ satisfying the constraints. □

As before, we can extend (B.10) to allow negative elements of $\mathbf{x}$ by multiplying the result of projecting the absolute values by the signs of the corresponding input elements. Below, we give

pseudo-code for a general method that computes the projection onto the ℓ_1 norm cone.

Input: Scalar g and p-vector $\mathbf{x}$
if $g \geq ||\mathbf{x}||_1$ **then**
  return; `// input value satisfies constraints`
sorted $\leftarrow$ {0,sort($|\mathbf{x}|$)}; `// sort absolute values in ascending order, append zero`
for $k \leftarrow 1$ *to* p **do**
  $\theta \leftarrow$sorted$(k+1)$;
  if $\alpha + \theta > \sum_{i=1}^{p} \max\{0, \mathbf{x} - \theta\}$ **then**
    break;
$\theta =$sorted$(k) + \sum_{i=1}^{p} \max\{0, \mathbf{x} - (g+\text{sorted}(k))\}/(k+1)$;
$g = g + \theta$;
$\mathbf{x} = \max\{0, \mathbf{x} - \theta\}$;

Algorithm 21: Projection onto ℓ_1 norm cone.

By replacing the *for* loop with a binary search for k, the implementation above can be modified to run in $\mathcal{O}(p \log p)$ time. Similar to [Duchi et al., 2008b], this can be futher reduced to $\mathcal{O}(p)$ by using a linear-time median-finding algorithm rather than sorting.

B.5 Nuclear Norm

We finally consider the case of projecting onto the nuclear norm cone. In this case, we can write problem (B.1) for an input matrix X_0 as

$$\arg\min_{X,g} \frac{1}{2}||X - X_0||_F^2 + \frac{1}{2}(g - g_0)^2, \text{ s.t. } ||X||_\sigma \leq g. \tag{B.13}$$

Here, we use $||X||_\sigma$ to denote the nuclear norm of X, the sum of the singular values of the matrix X. We denote the singular value decomposition of X_0 by $X_0 = U_0 \Sigma_0 V_0^T$, where Σ_0 a diagonal matrix containing the singular values σ_0. Using this notation, the solution of (B.13) is given by

$$\mathcal{P}_{\mathcal{C}_\sigma}(X_0, g_0) = (U_0 \tilde{\Sigma} V_0^T, \tilde{g}), \tag{B.14}$$

where $\tilde{\Sigma}$ is a diagonal matrix with elements $\tilde{\sigma}$, and $(\tilde{\sigma}, \tilde{g})$ is the result of projecting (σ_0, g_0) onto the ℓ_1 norm cone.

Proof. We first establish that in an optimal solution (X^*, g^*) that X^* must have the same singular vectors as X_0 (for all non-zero singular values). To do this we note that solving (B.13) is equivalent to minimizing the Lagrangian for some $\mu \geq 0$:

$$\min_{X,g} \frac{1}{2}||X - X_0||_F^2 + \frac{1}{2}(g - g_0)^2 + \mu||X||_\sigma - \mu g,$$

We can re-write this problem as

$$\min_{g} \frac{1}{2}(g - g_0)^2 - \mu g + \min_{X} \frac{1}{2}||X - X_0||_F^2 + \mu||X||_\sigma.$$

Focusing on the inner minimization over X, Cai et al. [2010, Theorem 2.1] implies that the optimal solution X^* has the same singular vectors as X_0.

Using this property we can re-write (B.13) as

$$\arg\min_{\sigma,g} \frac{1}{2}||U_0\text{diag}(\sigma)V_0^T - U_0\text{diag}(\sigma_0)V_0^T||_F^2 + \frac{1}{2}(g-g_0)^2, \text{ s.t. } ||U_0\text{diag}(\sigma)V_0^T||_\sigma \leq g.$$

Since U_0 and V_0 are orthogonal, we can re-write this as

$$\arg\min_{\sigma,g} \frac{1}{2}||\sigma-\sigma_0||_2^2 + \frac{1}{2}(g-g_0)^2, \text{ s.t. } \sum_{i=1}^{p}\sigma_i \leq g, \sigma \geq \mathbf{0}.$$

the projection of (σ_0, g_0) onto the ℓ_1 norm cone. □