Journal of Machine Learning Research 10 (2009) 883-906 Submitted 4/08; Revised 1/09; Published 4/09

Estimation of Sparse Binary Pairwise Markov Networks using Pseudo-likelihoods

Holger Höfling HHOEFLIN@GMAIL.COM
Department of Statistics
Stanford University
Stanford, CA 94305, USA

Robert Tibshirani TIBS@STANFORD.EDU
Depts. of Health, Research & Policy and Statistics
Stanford University
Stanford, CA 94305, USA

Editor: Michael Jordan

Abstract

We consider the problems of estimating the parameters as well as the structure of binary-valued Markov networks. For maximizing the penalized log-likelihood, we implement an approximate procedure based on the pseudo-likelihood of Besag (1975) and generalize it to a fast exact algorithm. The exact algorithm starts with the pseudo-likelihood solution and then adjusts the pseudo-likelihood criterion so that each additional iterations moves it closer to the exact solution. Our results show that this procedure is faster than the competing exact method proposed by Lee, Ganapathi, and Koller (2006a). However, we also find that the approximate pseudo-likelihood as well as the approaches of Wainwright et al. (2006), when implemented using the coordinate descent procedure of Friedman, Hastie, and Tibshirani (2008b), are much faster than the exact methods, and only slightly less accurate.

Keywords: Markov networks, logistic regression, L_1 penalty, model selection, Binary variables

1. Introduction

In recent years a number of authors have proposed the estimation of sparse undirected graphical models for continuous as well as discrete data through the use of L_1 (lasso) regularization. For continuous data, one assumes that the observations have a multivariate Gaussian distribution with mean μ and covariance matrix Σ. Then an L_1 penalty is applied to $\Theta = \Sigma^{-1}$. That is, the penalized log-likelihood $\ell(\Theta) - \rho||\Theta||_1$ is maximized, where ℓ is the Gaussian log-likelihood, $||\Theta||_1$ is the sum of the absolute values of the elements of Θ and $\rho \geq 0$ is a user-defined tuning parameter. Several papers proposing estimation procedures for this Gaussian model have been published. Meinshausen and Bühlmann (2006) develop a lasso based method for estimating the graph structure and give theoretical consistency results. Yuan and Lin (2007), Banerjee et al. (2008) and Dahl et al. (2008) as well as Friedman, Hastie, and Tibshirani (2008a) propose algorithms for solving this penalized log-likelihood with the procedure in Friedman, Hastie, and Tibshirani (2008a), the *graphical lasso*, being especially fast and efficient.

Here we focus on estimation of networks of discrete, more specifically binary-valued, units with pairwise interactions. These are a special class of Markov networks. The use of L_1 penalties for this

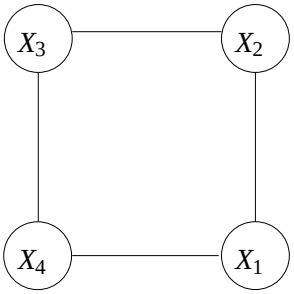

Figure 1: A simple graph for illustration.

x_1	x_2	x_3	x_4
1	1	1	0
1	1	0	1
0	0	1	0
1	1	0	1
1	1	1	0
1	1	0	1
0	0	1	0
1	1	0	1
0	1	1	0
1	1	0	1

Table 1: Sample data for graph of Figure 1

special class as well as more general Markov networks was proposed by Lee, Ganapathi, and Koller (2006a). This problem is more difficult than the continuous Gaussian version because of the need to compute the first and second moments under the model, which are derivatives of the log-partition function. Figure 1 shows an example, and Table 1 shows some sample data from this graph. Given this data and a model for binary graphical data (detailed later), we would like to a) infer a structure something like that of Figure 1, and b) estimate the link parameters itself. For the data in Table 1, Figure 2 shows the path of solutions for varying penalty parameter. Most edges for L_1-norm ≤ 2 are correctly identified as in the graph in Figure 1. However, edge $(1,3)$ is absent in the true model but included in the L_1 penalized model relatively early.

Our main focus in this paper is to develop and implement fast approximate and exact procedures for solving this class of L_1-penalized binary pairwise Markov networks and compare the accuracy and speed of these to other methods proposed by Lee, Ganapathi, and Koller (2006a) and Wainwright et al. (2006). Here, by "exact" procedures we refer to algorithms that find the exact maximizer of the penalized log-likelihood of the model whereas "approximate" procedures only find an approximate solution.

In Section 2 we describe the Ising model as well as the details of the competing methods of Lee, Ganapathi, and Koller (2006a) and Wainwright et al. (2006). Section 3 then introduces the basic pseudo-likelihood model and outlines the computational approach to increase the speed of the algorithm. The pseudo-likelihood is a very interesting and fast approximate method which has

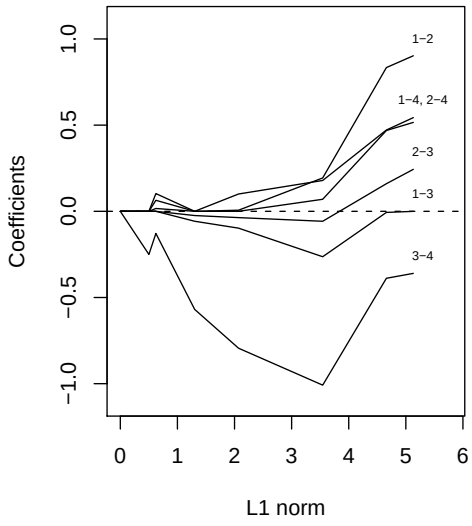

Figure 2: Toy example: profiles of estimated edge parameters as the penalty parameter is varied.

the added advantage that it can be used as a building block to a new algorithm for maximizing the penalized log-likelihood exactly. The adjustments necessary to achieve this are described in Section 4. Finally, Section 5 discusses the results of the simulations with respect to speed and accuracy of the competing algorithms.

2. The Model and Competing Methods

In this section we briefly outline the competing methods for maximizing the penalized log-likelihood. Apart from the method proposed in Lee, Ganapathi, and Koller (2006a), which we already mentioned above, we also discuss a very simple solution that was presented in Wainwright et al. (2006). We first describe the underlying model in more detail. Consider data generated under the Ising model

$$p(\mathbf{x},\Theta) = \exp\left[\sum_{s\in V}\theta_s x_s + \sum_{(s,t)\in E}\theta_{st}x_s x_t - \Psi(\Theta)\right].$$

for a single observation $\mathbf{x} = (x_1,\ldots,x_p)^T \in \{0,1\}^p$ and model parameters θ_s and θ_{st} for $s,t \in V = \{1,\ldots,p\}$. Here, V denotes the vertices and E the edges of a graph. $\Psi(\Theta)$ is the normalization constant, which is also known as the log-partition function. By setting $\theta_{st} = 0$ for $(s,t) \notin E$, and using the fact that $\mathbf{x}$ is a binary vector we can write the log-likelihood as

$$l(\mathbf{x},\Theta) = \log p(\mathbf{x},\Theta) = \sum_{s\geq t\geq 1}^{p}\theta_{st}x_s x_t - \Psi(\Theta)$$

where Θ is a symmetric $p \times p$ matrix with $\theta_{ss} = \theta_s$ for $s = 1, \ldots, p$. Note that for notational convenience, we do not distinguish between θ_{st} and θ_{ts} and therefore we enforce symmetry of Θ although the log-likelihood only uses the lower-triangular matrix of Θ. For the L_1-penalty let $\mathbf{R}$ be a $p \times p$ lower triangular matrix of penalty parameters. The penalized log-likelihood for all N observations is

$$\sum_{s \geq t \geq 1}^{p} (\mathbf{X}^T \mathbf{X})_{st} \theta_{st} - N\Psi(\Theta) - N||\mathbf{R} * \Theta||_1$$

where $*$ denotes component-wise multiplication.

The algorithms that we will discuss in the following sections could be generalized to more general categorical variables or higher order interaction terms than those included in the Ising model. However, as we will see, solving these problems exactly is already computationally demanding in the pairwise binary case so that we chose not to adapt the algorithms to these more general settings.

2.1 The Lee, Ganapathi, and Koller (2006a) Method

The penalized log-likelihood is a concave function, so standard convex programming techniques can be used to maximize it. The main difficulty is that the log-partition function $\Psi(\Theta)$ is the sum over 2^p elements, and therefore it is computationally expensive to calculate Ψ or its derivatives in general. However, for sparse matrices Θ, algorithms such as the junction tree algorithm exist that can calculate Ψ and its derivatives efficiently. Therefore, it is especially important to maintain sparsity of Θ for any optimization method. Lee, Ganapathi, and Koller (2006a) achieve this by optimizing the penalized log-likelihood only over a set F of *active variables* which they gradually enlarge until an optimality criterion is satisfied.

To be more precise, they start out with a set of active variables $F = F_0$ (e.g., the diagonal of Θ if it is unpenalized). Using either conjugate gradients or BFGS, the penalized log-likelihood is maximized over the set of variables F. Then one of the currently inactive variables is selected by the *grafting* procedure (see Perkins et al., 2003) and added to the set F. These steps are repeated until grafting does not add any more features. The algorithm can be used for more general Markov networks, but for ease of implementation they choose to work with binary random variables and we do the same as well.

Their procedure provides an exact solution to the problem when the junction tree algorithm is used to calculate $\Psi(\Theta)$ and its derivatives. In their implementation, however, they used loopy belief propagation, which is faster on denser matrices, but only provides approximate results. In their method as well as ours, any procedure to evaluate $\Psi(\Theta)$ can be "plugged in" without any further changes to the rest of the algorithm; we decided to evaluate the speed and performance of only the exact algorithms. The relative performance of an approximate method using loopy belief propagation would likely be similar. They also provide a proof that under certain assumptions and using the L_1 regularized log-likelihood, it is possible to recover the true expected log-likelihood up to an error ε.

2.2 The Wainwright et al. (2006) Method

Wainwright et al. (2006) propose estimation of the Markov network by applying a separate L_1-penalized logistic regression to each of the p variables on the remaining variables. For every $s \in V$ regress x_s onto $\mathbf{x}_{\setminus s} = (x_1, \ldots, x_{s-1}, x_{s+1}, \ldots, x_p)^T$. Let the $p \times p$ matrix $\tilde{\Theta}$ denote the estimate of Θ.

Then set $\tilde{\theta}_{ss} = \beta_0$, the intercept of the logistic regression, and $\tilde{\theta}_{st} = \beta_t$, where β_t is the coefficient associated with x_t in the regression.

In the outline above, we assumed that Θ is a symmetric matrix. However, due to the way it was constructed, $\tilde{\Theta}$ is not necessarily symmetric. We investigate two methods for symmetrizing $\tilde{\Theta}$. The first way is to define Θ as

$$\theta_{st} = \theta_{ts} = \begin{cases} \tilde{\theta}_{st} & \text{if } |\tilde{\theta}_{st}| > |\tilde{\theta}_{ts}| \\ \tilde{\theta}_{ts} & \text{if } |\tilde{\theta}_{st}| \leq |\tilde{\theta}_{ts}| \end{cases}$$

which we call "Wainwright-max". Similarly, "Wainwright-min" is defined by

$$\theta_{st} = \theta_{ts} = \begin{cases} \tilde{\theta}_{st} & \text{if } |\tilde{\theta}_{st}| < |\tilde{\theta}_{ts}| \\ \tilde{\theta}_{ts} & \text{if } |\tilde{\theta}_{st}| \geq |\tilde{\theta}_{ts}| \end{cases}.$$

Wainwright et al. (2006) mainly intended their method to be used in order to estimate the presence or absence of an edge in the underlying graph of the model. They show that under certain assumptions, their method correctly identifies the non-zero edges in a Markov graph, as $N \to \infty$ even for increasing number of parameters p or neighborhood sizes of the graph d, as long as N grows more quickly than $d^3 \log p$ (see Wainwright et al., 2008). Due to the simplicity of the method it is obvious that it could also be used for parameter estimation itself and here we will compare its performance in these cases to the pseudo-likelihood approach proposed below and the exact solution of the penalized log-likelihood. Furthermore, as an important part of this article is the comparison of the speeds of the underlying algorithms, we implement their method, using the fast coordinate descent algorithm for logistic regression with a lasso penalty (see Friedman, Hastie, and Tibshirani, 2008b).

3. Pseudo-likelihood Model

In this section we first introduce an approximate method to infer the structure of the graph that is based on pseudo-likelihoods (Besag, 1975). As we will see in the simulations section, the results are very close to the exact solution of the penalized log-likelihood. In the next section, we use the pseudo-likelihood model to design a very fast algorithm for finding an exact solution for the penalized Ising model.

The main computational problem in the Ising model is the complexity of the partition function. One possibility in this case is to solve an approximate version of the likelihood instead. Approaches of this kind have been proposed in various papers in the statistical literature before, for example the pseudo-likelihood approach of Besag (Besag, 1975) and the treatments of composite likelihoods in Lindsay (1998) and Cox and Reid (2004) among others. Here, we want to apply the approximation proposed in Besag (1975) to our problem. This approach is also related to the method of Wainwright et al. (2006), however instead of performing separate logistic regressions for every column of the parameter matrix Θ, the pseudo-likelihood approach here allows us to estimate all parameters at the same time. This way, the resulting matrix Θ is symmetric and no additional step like the max or min-rule described above is necessary. The log-pseudo-likelihood is then given by

$$\tilde{l}(\Theta|\mathbf{x}) = \sum_{s=1}^{p} \log p(x_s, \Theta|\mathbf{x}_{\setminus s})$$

with

$$\log p(x_s, \Theta|\mathbf{x}_{\setminus s}) = x_i(\theta_{ss} + \sum_{t \neq s} x_t \theta_{st}) - \Psi_s(\mathbf{x}, \Theta).$$

Here, $\Psi_s(\mathbf{x},\Theta)$ is the log-normalization constant when conditioning x_s on the other variables, which is exactly the same as in logistic regression with a logit link-function, that is,

$$\Psi_s(\mathbf{x},\Theta) = \log(1+\exp(\theta_{ss}+\sum_{t\neq s} x_t\theta_{st}))$$

where as above for notational convenience we set $\theta_{st}=\theta_{ts}$ for $t\neq s$. Putting all this together, the pseudo-likelihood for a single observation $\mathbf{x}$ is given by

$$\tilde{l}(\Theta|\mathbf{x}) = \sum_{s=1}^{p}\sum_{t=1}^{p} x_s x_t \theta_{st} - \sum_{s=1}^{p}\Psi_s(\mathbf{x},\Theta).$$

In the usual way, the pseudo-likelihood for all N observations is given by the sum of the pseudo-likelihood of the individual observations

$$\tilde{l}(\Theta|\mathbf{X}) = \sum_{k=1}^{N}\tilde{l}(\Theta|x_k)$$

where $\mathbf{x}_k$ is the kth row of matrix with observations $X\in\mathbb{R}^{N\times p}$.

As this is just a sum of logistic likelihoods, the pseudo-likelihood is a concave function and therefore the L_1-penalized pseudo-likelihood

$$\sum_{k=1}^{N}\tilde{l}(\Theta|\mathbf{x}_k) - N||\mathbf{S}*\Theta||_1$$

is concave as well. Here $\mathbf{S}=2\mathbf{R}-diag(\mathbf{R})$ and is chosen to be roughly equivalent to the penalty terms in the penalized log-likelihood. The penalty term is doubled on the off-diagonal, as the derivative of the pseudo-likelihood on the off-diagonal is roughly twice as large as the derivative of the log-likelihood (see Equation 1).

3.1 Basic Optimization Algorithm

Due to its simple structure, a wide range of standard convex programming techniques can be used to solve this problem, although the non-differentiability of the L_1 penalty poses a problem. Here, we want to use a local quadratic approximation to the pseudo-likelihood. As the number of variables is $p(p+1)/2$, the Hessian could get very large, we restrict our quadratic approximation to have a diagonal Hessian.

In order to construct the approximation, we need the first and second derivative of $\tilde{l}$ w.r.t θ_{st}. These are

$$\begin{aligned}\frac{\partial\tilde{l}}{\partial\theta_{st}} &= 2(\mathbf{X}^T\mathbf{X})_{st} - (\hat{\mathbf{p}}_s^T\mathbf{X})_t - (\hat{\mathbf{p}}_t^T\mathbf{X})_s \quad s\neq t, \\ \frac{\partial\tilde{l}}{\partial\theta_{ss}} &= (\mathbf{X}^T\mathbf{X})_{ss} - \sum_{k=1}^{N}\hat{p}_{sk}\end{aligned} \tag{1}$$

where $1-\hat{p}_{sk} = 1/(1+\exp(\theta_{ss}+\sum_{s\neq t}X_{kt}\theta_{st}))$. The second derivatives are

$$\begin{aligned}\frac{\partial^2\tilde{l}}{(\partial\theta_{st})^2} &= -(\mathbf{X}^T\mathrm{diag}(\hat{\mathbf{p}}_s)\mathrm{diag}(1-\hat{\mathbf{p}}_s)\mathbf{X})_{tt} - (\mathbf{X}^T\mathrm{diag}(\hat{\mathbf{p}}_t)\mathrm{diag}(1-\hat{\mathbf{p}}_t)\mathbf{X})_{ss} \quad s\neq t, \\ \frac{\partial^2\tilde{l}}{(\partial\theta_{ss})^2} &= -\sum_{k=1}^{N}\hat{p}_{sk}(1-\hat{p}_{sk}).\end{aligned}$$

Algorithm 1: Estimation for L_1 penalized pseudo-likelihood

if $\Theta^{(0)}$ *not given* **then**
  Set $\Theta^{(0)} = \text{diag}(\text{logit}(\hat{\mathbf{p}}^{(0)}))$ where $\hat{p}_s^{(0)} = \frac{1}{N}\sum_{k=1}^{N} x_{ks}$
end
Set k:=0;
while *not converged* **do**
  With current estimate $\Theta^{(k)}$, define local approximation $f_{\Theta^{(k)}}(\Theta)$ to $\tilde{l} - N||\mathbf{S} * \Theta||_1$;
  Find solution Θ^* of $f_{\Theta^{(k)}}(\Theta)$;
  Perform backtracking line search on the line from $\Theta^{(k)}$ to Θ^* to find $\Theta^{(k+1)}$;
  Set k:=k+1
end

Assume that at the k-th step the parameter estimate is $\Theta^{(k)}$. Then define the local approximation to $\tilde{l}(\Theta|\mathbf{X}) - N||\mathbf{S} * \Theta||_1$ as

$$f_{\Theta^{(k)}}(\Theta) = C + \sum_{s \geq t} \frac{\partial \tilde{l}}{\partial \theta_{st}}(\theta_{st} - \theta_{st}^{(k)}) + \frac{1}{2}\frac{\partial^2 \tilde{l}}{(\partial \theta_{st})^2}(\theta_{st} - \theta_{st}^{(k)})^2 - N||\mathbf{S} * \Theta||_1$$

where C is some constant. As stated above, this is just a quadratic approximation with linear term equal to the gradient of $\tilde{l}$ and a diagonal Hessian with diagonal elements equal to the diagonal of the Hessian of $\tilde{l}$. The main reasons for using this simple structure are that it keeps the computation complexity per iteration low and it is very easy to solve this L_1 penalized local approximation. Let $\tilde{\Theta}$ be the minimizer of the unpenalized $f_{\Theta^{(k)}}(\Theta)$, then

$$\tilde{\theta}_{st} = \theta_{st}^{(k)} - \frac{\frac{\partial \tilde{l}}{\partial \theta_{st}}}{\frac{\partial^2 \tilde{l}}{(\partial \theta_{st})^2}}.$$

As the Hessian is diagonal, the L_1-penalized solution Θ^* of $f_{\Theta^{(k)}}(\Theta)$ can be obtained by soft thresholding as

$$\theta_{st}^* = \text{sign}(\tilde{\theta}_{st})\left(\tilde{\theta}_{st} - s_{st}/\frac{\partial^2 \tilde{l}}{(\partial \theta_{st})^2}\right)_+.$$

Using Θ^*, the next step $\Theta^{(k+1)}$ can now be obtained by, for example, a backtracking line search. The whole algorithm can be seen in Algorithm 1 and a proof of convergence that closely follows Lee, Lee, Abbeel, and Ng (2006b) is given in the appendix.

3.2 Speed Improvements

In practice, there are several things that can be done to speed up the algorithm given above. First of all, as Θ will in general be sparse, all computations to calculate $\hat{\mathbf{p}}$ should exploit this sparse structure. However, the sparseness of Θ can also be used in another way:

3.2.1 USING ACTIVE VARIABLES

As the solutions are usually sparse, calculating the gradient for all variables in every step is wasteful. Most variables are zero and will not change from one iteration to the next. In order to be more

Algorithm 2: Pseudo-likelihood algorithm using active variables

if $\Theta^{(0)}$ *not given* **then**
  Set $\Theta^{(0)} = \text{diag}(\text{logit}(\hat{\mathbf{p}}^{(0)}))$ where $\hat{p}_s^{(0)} = \frac{1}{N}\sum_{k=1}^{N} x_{ks}$
end
Set k:=0;
Set $\mathcal{A} = \{(s,t) : s \geq t, \theta_{st} \neq 0\}$ as active variables;
repeat
  while *not converged over variables in* $\mathcal{A}$ **do**
    Find $\Theta^{(k+1)}$ using local approximation over variables in $\mathcal{A}$;
    Set k:=k+1;
  end
  Set $\mathcal{A} = \left\{(s,t) : \theta_{st} \neq 0 \text{ or } \left|\frac{\partial \tilde{l}}{\partial \theta_{st}}\right| > s_{st}\right\}$;
until $\mathcal{A}$ *did not change* ;

efficient, it is possible to move variables that are zero only once in a while. Several different kinds of methods have been proposed to exploit this situation, for example grafting (Perkins et al., 2003) and the implementation of the penalized logistic regression in Friedman, Hastie, and Tibshirani (2008b) among others. In our case here, we use an outer and an inner loop. The outer loop decides which variables are active. The inner loop then optimizes over only the active variables until convergence occurs. Active variables are those that are either non-zero, or that have a gradient large enough so that they would become non-zero in the next step. More details are given in Algorithm 2.

When using this method, convergence is still guaranteed. In the outer loop, the criterion chooses variables to be active that are either non-zero already or will be non-zero after one step of the local approximation over all variables. Therefore, if the active set stays the same, no variables would be moved in the next step as all active variables are already optimal and therefore, we have a solution over all variables. However, if the active set changes, the inner loop is guaranteed to improve the penalized pseudo-likelihood and find the optimum for the given set of active variables. As there are only a finite number of different active variables sets, the algorithm has to converge after a finite number of iterations of the outer loop.

3.2.2 SUBSTITUTING THE LINE SEARCH

Calculating the pseudo-likelihood is computationally expensive compared to the cost of the local approximation. Therefore, we save time by not performing a line search after every step. Instead, we fix a step size γ and skip the line search. However, with this method, the algorithm may diverge. In order to detect this, we calculate the penalized pseudo-likelihood every 100 iterations and check that we improved during the last 100 steps. If yes, the step size remains the same. If no, reset the variables to where they were 100 steps ago and divide the step size by 2. If the step size drops below a pre-specified threshold, we revert to the original line-search algorithm. This way, in most cases, we do not have to perform the line-search. However, the algorithm is still guaranteed to converge as it automatically detects convergence problems when the fixed step size is used and reverts to the line-search algorithm, which as stated above is guaranteed to converge.

4. Exact Solution Using Pseudo-likelihood

We now turn to our new method for optimizing the penalized log-likelihood. As the log-likelihood is a concave function, there are several standard methods that can be used for maximizing it, for example, gradient descent, BFGS and Newton's method among others. For each step of these methods, the gradient of the log-likelihood has to be calculated. This requires the evaluation of the partition function and its derivatives, which is computationally much more expensive than any other part of the algorithms. Taking this into considerations, the standard methods mentioned above have the following drawbacks:

Gradient descent: It can take many steps to converge and therefore require many evaluations of the partition function and its derivatives (using the junction tree algorithm). Also, it does not control the number of non-zero variables in intermediate steps well to which the runtime of the junction tree algorithm is very sensitive. Therefore, intermediate steps can take very long.

BFGS: Takes less steps than gradient descent, but similar to gradient descent, intermediate steps can have more non-zero variables than the solution. Thus, same as above, computations of intermediate steps can be slow.

Newton's method: In order to locally fit the quadratic function, the second derivatives of the log-likelihood are needed. Computing these is computationally prohibitive.

Lee, Ganapathi, and Koller (2006a) use the BFGS method only on a set of active variables onto which additional variables are "grafted" until convergence. This mitigates the problem of slow intermediate steps and makes using BFGS feasible. However, this comes at the expense of an increased total number of steps, as only one variable at a time is being grafted. Here, we want to use the pseudo-likelihood in a new algorithm for maximizing the penalized log-likelihood.

The functional form of the pseudo-likelihood is by its definition closely related to the real likelihood and in Section 5 we will see that its solutions are also very similar, indicating that it approximates the real likelihood reasonably well. Furthermore, we can also maximize the pseudo-likelihood very quickly. We want to leverage this by using a quasi-Newton method in which we fit a "tilted" pseudo-likelihood instead of a quadratic function. Specifically, among all functions of the form

$$f_{\Theta^{(k)}} = \frac{1}{2}\tilde{l} + \sum_{s \geq t} a_{st}(\theta_{st} - \theta_{st}^{(k)}) - \gamma \sum_{s \geq t} (\theta_{st} - \theta_{st}^{(k)})^2 - N||\mathbf{R} * \Theta||_1,$$

we fit the one that at $\Theta^{(k)}$ is a first order approximation to the penalized log-likelihood l. Essentially, $f_{\Theta^{(k)}}$ is an L_1-penalized pseudo-likelihood with an added linear term as well as a very simple quadratic term for some $\gamma > 0$. Here, a_{st} will be chosen so that the sub-gradient is equal to the penalized log-likelihood l at $\Theta^{(k)}$. The additional quadratic term with coefficient γ is only being included to ensure the existence of a global maximum. In practice, we can set $\gamma = 0$, unless a convergence problem occurs. In our simulations, $\gamma = 0$ always worked very well.

In particular, for the approximation at $\Theta^{(k)}$, choose a_{st} such that

$$f_{\Theta^{(k)}} = \frac{1}{2}\Bigg(\tilde{l}(\Theta|\mathbf{X}) + \sum_{s>t}(\theta_{st}-\theta_{st}^{(k)})\left((\hat{\mathbf{p}}(\Theta^{(k)})_s^T\mathbf{X})_t + (\hat{\mathbf{p}}(\Theta^{(k)})_t^T\mathbf{X})_s - 2\cdot N\cdot w_{st}(\Theta^{(k)})\right) + \\ + \sum_s(\theta_{ss}-\theta_{ss}^{(k)})\left(\sum_k \hat{p}_{sk}(\Theta^{(k)}) + (\mathbf{X}^T\mathbf{X})_{ss} - 2\cdot N\cdot w_{ss}(\Theta^{(k)})\right)\Bigg) - \\ - \sum_{s\geq t}\gamma(\theta_{st}-\theta_{st}^{(k)})^2 - N||\mathbf{R}*\Theta||_1$$

where $\mathbf{W}$ is a matrix with elements $w_{st}(\Theta) = \frac{\partial\Psi}{\partial\theta_{st}}(\Theta) = \mathbb{E}_\Theta(x_s x_t)$ is the derivative of the partition function and $\hat{\mathbf{p}}_s$ is as defined in the pseudo-likelihood section.

For the algorithm, we need an initial parameter estimate $\Theta^{(0)}$, which we pick as follows: Let $Z(\theta) = \frac{e^\theta}{1+e^\theta}$ be the logistic function and let Z^{-1} denote its inverse. Then choose

$$\theta_{st}^{(0)} = \begin{cases} 0 & \text{if } s\neq t \\ Z^{-1}\left(\frac{1}{N}\sum_{k=1}^N x_{ks}\right) & \text{if } s=t \end{cases}.$$

In this case we than have $\hat{p}_{sk} = \frac{1}{N}\sum_{k=1}^N x_{ks} \quad \forall k$ and also

$$W_{st} = \begin{cases} \left(\frac{1}{N}\sum_{k=1}^N x_{ks}\right)\left(\frac{1}{N}\sum_{k=1}^N x_{kt}\right) & \text{if } s\neq t \\ \left(\frac{1}{N}\sum_{k=1}^N x_{ks}\right) & \text{if } s=t \end{cases}$$

and together with $\gamma = 0$ we then get that

$$f_{\Theta^{(0)}} = \frac{1}{2}\tilde{l}(\Theta|\mathbf{X}) - N||\mathbf{R}*\Theta||_1$$

and thus just a regular pseudo-likelihood step. The only slight difference is that in this case the penalty term on the diagonal is twice as large as in the pseudo-likelihood case presented above. However, in practice we recommend not to penalize the diagonal at all, so that this difference vanishes. Therefore, this algorithm is a natural extension of the pseudo-likelihood approach that starts out by performing a regular pseudo-likelihood calculation and then proceeds to converge to the solution by a series of adjusted pseudo-likelihood steps.

The algorithm for maximizing the penalized log-likelihood is now very similar to the one presented for maximizing the penalized pseudo-likelihood. Assume our current estimate is $\Theta^{(k)}$. Then approximate the log-likelihood locally by $f_{\Theta^{(k)}}$ and find the maximizer Θ^* of $f_{\Theta^{(k)}}$. This is essentially the pseudo-likelihood and the algorithm presented above can easily be adjusted to accommodate the additional terms. Now, using a line search on the line between $\Theta^{(k)}$ and Θ^*, find the next estimate $\Theta^{(k+1)}$. This algorithm is guaranteed to converge by the same argument as the pseudo-likelihood algorithm. The proof that $f_{\Theta^{(k)}}$ approximates l at $\Theta^{(k)}$ to first order can be found in Appendix B, which is a prerequisite for the convergence proof of the algorithm in Appendix A.

4.1 Speed Improvement

As for the pseudo-likelihood algorithm, we can again save computations by using the active variables technique presented above. Here, the savings in time are especially large due to a special feature of the junction tree algorithm that we use to calculate the derivatives of the partition function.

Algorithm 3: Likelihood algorithm using active variables

if $\Theta^{(0)}$ *not given* **then**
  Set $\Theta^{(0)} = \text{diag}(\text{logit}^{-1}(\hat{\mathbf{p}}^{(0)}))$ where $\hat{p}_s^{(0)} = \frac{1}{N}\sum_{k=1}^{N} x_{ks}$
end
Set k:=0;
Set $\mathcal{A} = \{(s,t) : s \geq t, \theta_{st} \neq 0\}$ as active variables;
repeat
  while *not converged over variables in* $\mathcal{A}$ **do**
    Calculate $\mathbf{W}$ only over variables in $\mathcal{A}$;
    Find $\Theta^{(k+1)}$ using local approximation over variables in $\mathcal{A}$;
    Set k:=k+1;
  end
  Calculate the whole matrix $\mathbf{W}$;
  Set $\mathcal{A} = \left\{(s,t) : \theta_{st} \neq 0 \text{ or } \left|\frac{\partial l}{\partial \theta_{st}}\right| > r_{st}\right\}$;
until $\mathcal{A}$ *did not change* ;

In order to calculate derivatives with respect to non-zero variables, only one pass of the junction tree algorithm is necessary. However, p passes of the junction tree are needed in order to get the full matrix of derivatives $\mathbf{W}$. Therefore, depending on the size of p, using only active variables can be considerable faster. For details, see Algorithm 3.

4.2 Graphical Lasso for the Discrete Case

In the case of Gaussian Markov networks, the graphical lasso algorithm (see Friedman, Hastie, and Tibshirani, 2008a) is an very efficient method and implementation for solving the L_1-penalized log-likelihood. In order to leverage this speed, we extended the methodology to the binary case treated in this article. However, the resultant algorithm was not nearly as fast as expected. In the Gaussian case, the algorithm is very fast as the approach to update the parameter matrix Θ one row at a time allows for a closed form solution and efficient calculations. In the binary case on the other hand, the computational bottleneck is not all of the calculation involved in the update of Θ, but specifically by a large margin the evaluations of the partition function itself. Therefore, any fast algorithm for solving the penalized log-likelihood exactly has to use as few evaluations of the partition function as possible. The graphical lasso approach is thus not suitable for the binary case as it takes a lot of small steps towards the solution.

This observation also explains the improvement in speed of the pseudo-likelihood based exact algorithm over the specific methods proposed in Lee, Lee, Abbeel, and Ng (2006b). Standard convex optimization procedures often rely on either the first or second derivatives of the functions they seek to optimize. Newton-like procedures that use the second derivative often converge in very few steps, however these cannot be used here as it is prohibitively expensive to evaluate the second derivative of the partition function, even in small examples. Approaches like conjugate gradients of BFGS as proposed in Lee, Lee, Abbeel, and Ng (2006b) are somewhat less efficient and take more steps. This is where the advantage of using the pseudo-likelihood as a local approximation comes into play. It usually only takes very few steps to converge and is therefore faster than standard methods.

5. Simulation Results

In order to compare the performance of the estimation methods for sparse graphs described in this article as well as Lee, Ganapathi, and Koller (2006a) and Wainwright et al. (2006), we use simulated data and compare the speed as well as the accuracy of these methods.

5.1 Setup

Before simulating the data it is necessary to generate a sparse symmetric matrix Θ. First, the diagonal is drawn uniformly from the set $\{-0.5, 0, 0.5\}$. Then, using the average number of edges per node, upper-triangular elements of Θ are drawn uniformly at random to be non-zero. These non-zero elements are then set to either -0.5 or 0.5, again each uniformly. In order for Θ to by symmetric, the lower triangular matrix is set equal to the upper triangular matrix. The actual data is generated by Gibbs sampling using Θ as described above.

With respect to the penalty parameters that we use for the different methods, we always leave the diagonal unpenalized and all off-diagonal elements have the same parameter ρ, that is we set

$$r_{st} = \begin{cases} 0 & \text{if } s = t \\ \rho & \text{otherwise} \end{cases}$$

and the penalty term matrix for the pseudo-likelihood $\mathbf{S} = 2\mathbf{R} - \text{diag}(\mathbf{R})$ as defined above. For the Wainwright-methods, the penalty parameter is ρ with no penalty on the intercept. Although the log-likelihood functions that are being penalized are somewhat different, this choice of parameters makes them perform roughly equivalent, as can be seen in Figure 3. The number of edges is plotted against the penalty parameter used and all methods behave very similar. However, in order not to confound some results by these slight differences of the effects of the penalty, all the following plots are with respect to the number of edges in the graph, not the penalty parameter itself.

5.2 Speed Comparison

First, we compare the speed of the four methods for the L_1-penalized model. We used an annealing schedule for Lee, Ganapathi, and Koller (2006a) to improve convergence as suggested in their article. Plots of the speeds of the exact methods can be seen in Figure 4 and the approximate methods are shown in Figure 5. Each plot shows the time the algorithm needed to converge versus the number of edges in the estimated graph. As can be seen, the pseudo-likelihood based exact algorithm described above is considerable faster than the one proposed in Lee, Ganapathi, and Koller (2006a). For the approximate algorithms, we can see that the p logistic regressions in the Wainwright et al. (2006) algorithm take roughly the same amount of time as the pseudo-likelihood algorithm presented above. This is not surprising due to the similarity of the optimization methods and any difference that can be observed in Figure 5 is mostly due to the specific implementations used. Furthermore, we would like to note that we decided to plot the speed against the number of edges in the graph instead of the penalty parameter, as the runtime of the algorithm is very closely related to the actual sparseness of the graph.

Overall, when comparing the computation times for the exact algorithms to the approximate algorithms, we can see that the approximate methods are orders of magnitude faster and do not suffer from an exponentially increasing computation time for decreasing sparsity as the exact methods. Therefore, if an exact solution is required, our exact algorithm is preferable. On the other hand, the

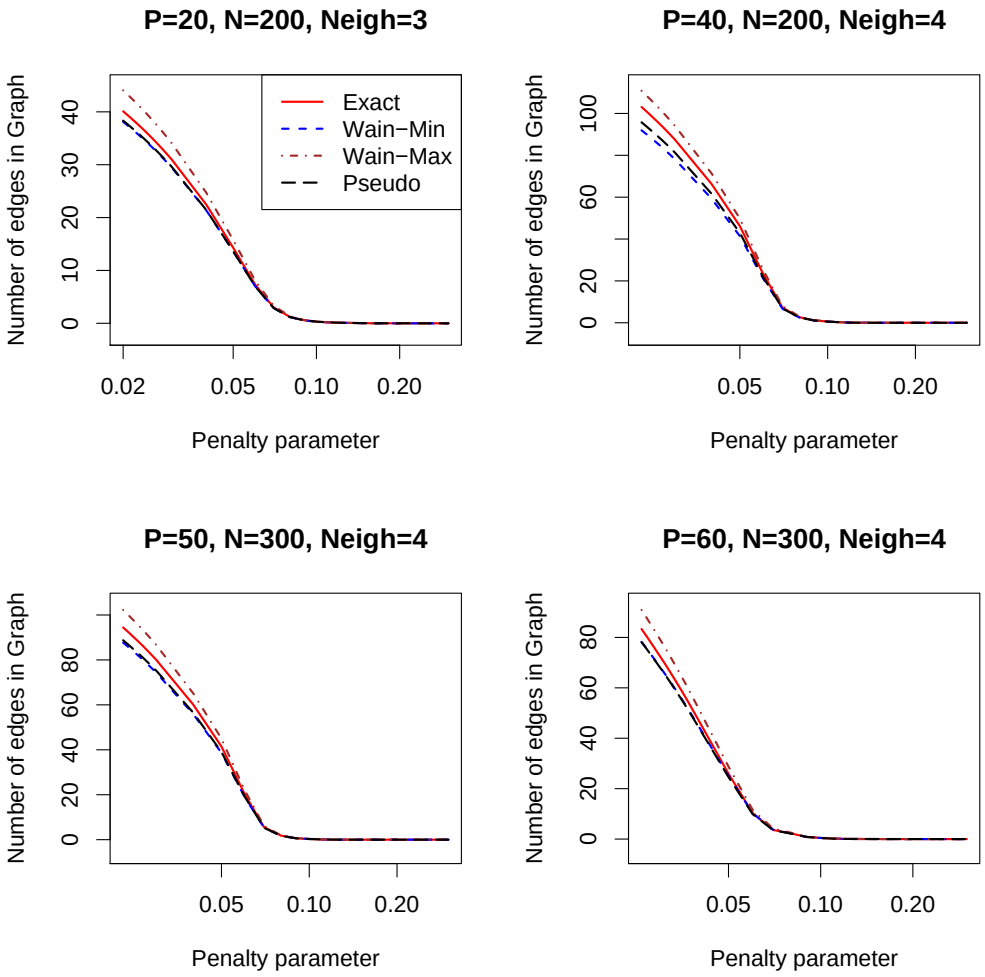

Figure 3: Number of edges in the graph vs. penalty parameter for different problem sizes, averaged over 20 simulations.

superior speed of the pseudo-likelihood and Wainwright et al. (2006) algorithm warrants a closer look at the trade-off with respect to accuracy.

5.3 Accuracy Comparisons

In this subsection we compare the algorithms mentioned above with respect to the accuracy with which they recover the original model. As both our L_1 penalized exact algorithm and the one by Lee, Ganapathi, and Koller (2006a) find the exact maximizer of the L_1-penalized log-likelihood, we only use our algorithm in the comparison. The other 3 methods we compare to are "Wainwright-min", "Wainwright-max" and the pseudo-likelihood.

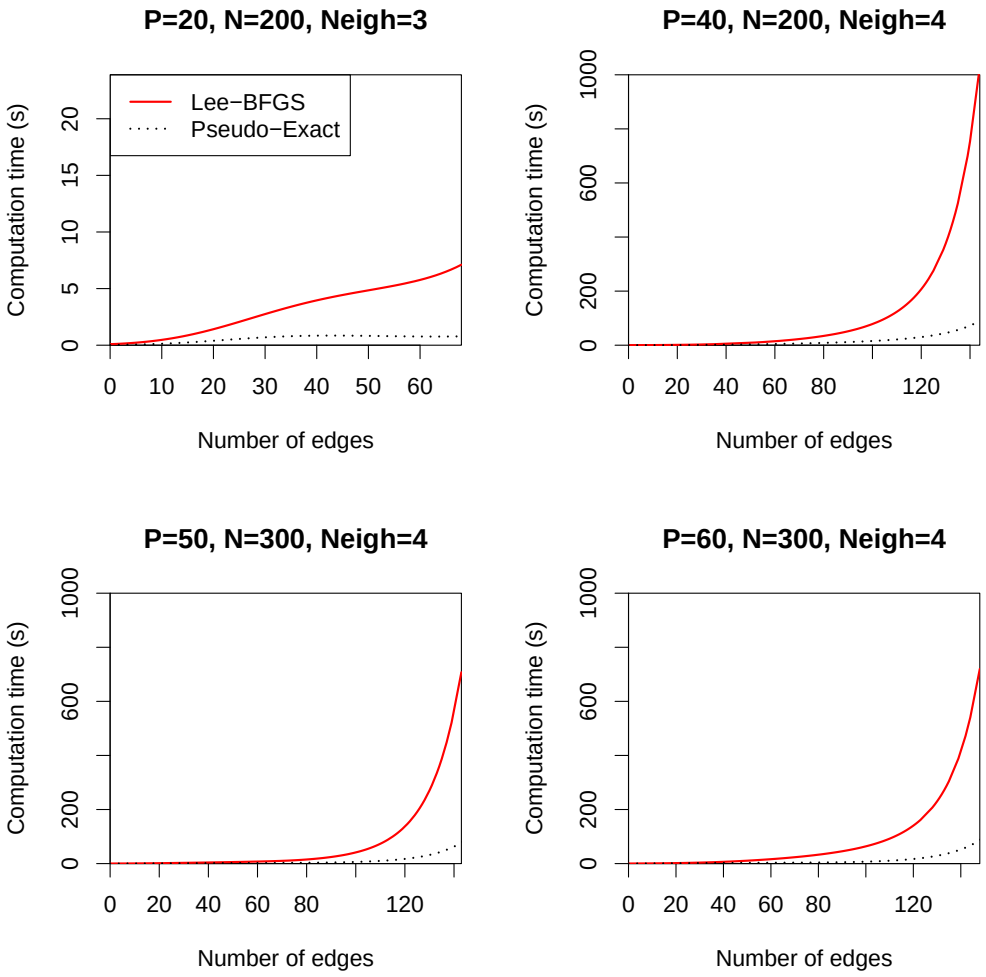

Figure 4: Computation time of the exact algorithms versus the number of non-zero elements in Θ. Values are averages over 20 simulation runs, along with ± 2 standard error curves. Also, p is the number of variables in the model, N the number of observations and *Neigh* is the average number of neighbors per node in the simulated data. Here, Pseudo-Exact refers to the the exact solution algorithm that uses adjusted pseudo-likelihoods as presented in Section 4.

First we investigate how closely the edges in the estimated graph correspond to edges in the true graph. In Figure 6, ROC curves are shown, plotting the false positive (FP) rates against true positive (TP) rates for edge identification, for various problem sizes. Note that only partial ROC curves are shown since our method cannot estimate non-sparse graphs due to very long computation times. Overall, we see that all approximate algorithms match the results of the exact solution very closely and in some of the plots, the curves even lie almost on top of each other.

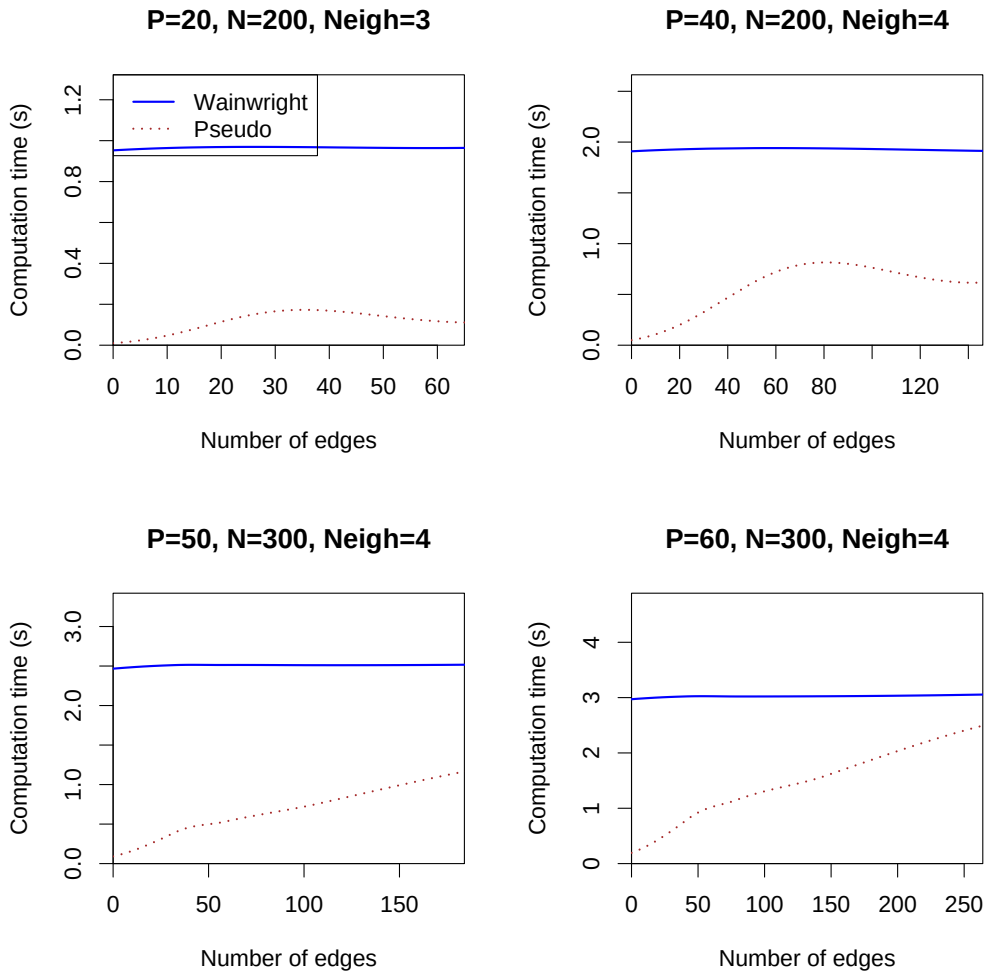

Figure 5: Computation time for the approximate algorithms versus the number of non-zero elements in Θ. Values are averages over 20 simulation runs, along with ± 2 standard error curves. Also, p is the number of variables in the model, N the number of observations and *Neigh* is the average number of neighbors per node in the simulated data.

Apart from the accuracy of edge identification, we also consider other statistics. The unpenalized log-likelihood is a measure of how well the estimated model fits the observed data (higher values are better). Again, the approximate solutions are all very close to the exact solution (see Figure 7) and the differences are always smaller than 2 standard deviations. In Figure 8, we plot the difference of the log-likelihood with respect to the exact solution. Also in this plot, no clear "winner" can be identified.

We also use the Kullback-Leibler divergence $D_{KL}(\mathbb{P}||\mathbb{Q})$, which is a measure of difference between a true probability distribution $\mathbb{P}$ and an arbitrary other distribution $\mathbb{Q}$. Here, for the true

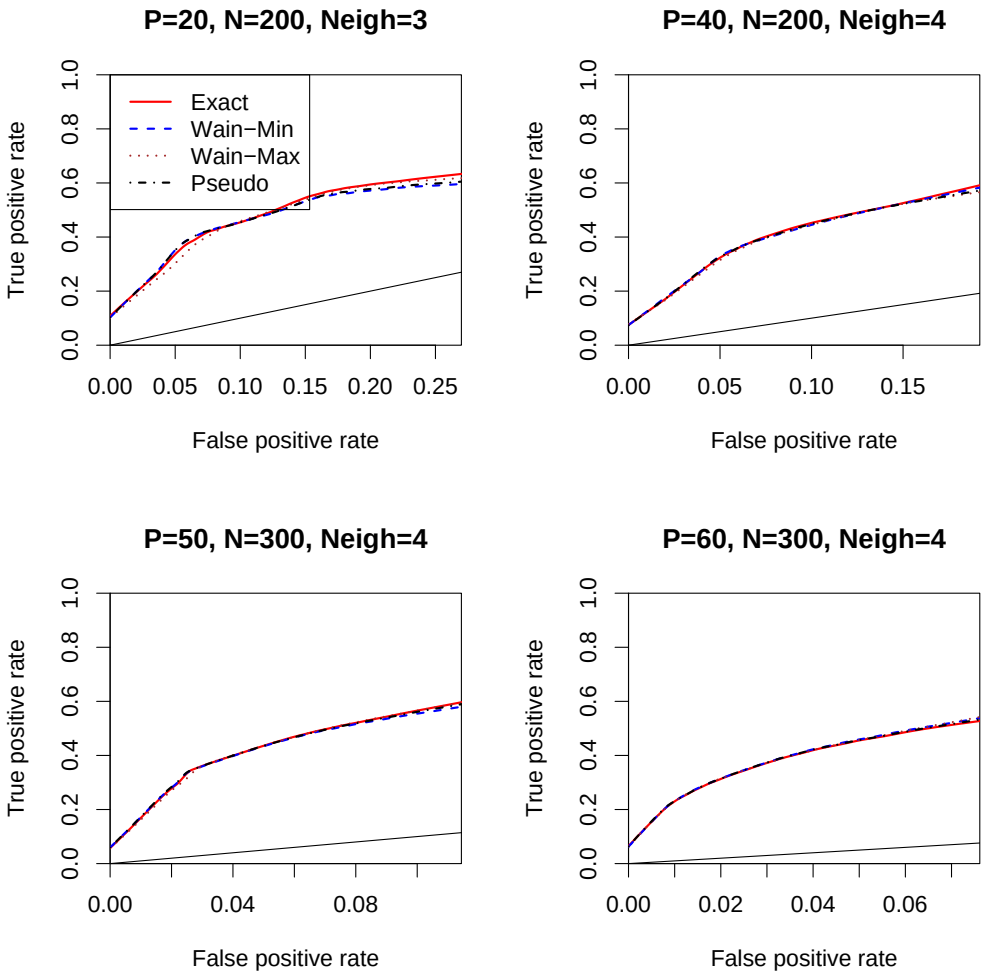

Figure 6: ROC curves: false positive versus true positive rate for edge identification. Values are averages over 20 simulation runs.

probability distribution we use the distribution of the binary Markov network using the true Θ_0-matrix that was used to generate the simulated data. The distribution $\mathbb{Q}$ in our case is the binary Markov network using the estimated $\hat{\Theta}$-matrix. We can compute $D_{KL}(\mathbb{P}||\mathbb{Q})$ as

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sum_x \mathbb{P}(x) \log \frac{\mathbb{P}(x)}{\mathbb{Q}(x)} = \Psi(\Theta_0) - \Psi(\hat{\Theta}) + \sum_x \mathbb{P}(x) \mathrm{tr}\left(xx^T(\hat{\Theta} - \Theta_0)\right) =$$
$$= \Psi(\Theta_0) - \Psi(\hat{\Theta}) + \mathrm{tr}\left(\mathbb{E}_{\mathbb{P}}(xx^T)(\hat{\Theta} - \Theta_0)\right).$$

If the distributions $\mathbb{P}$ and $\mathbb{Q}$ are identical, then $D_{KL}(\mathbb{P}||\mathbb{Q}) = 0$. In our simulations, the exact solution has lower KL-divergence than the other methods, however the differences are very small. For a plot of the KL-divergence against the number of edges in the model see Figure 9. Again, all approximate

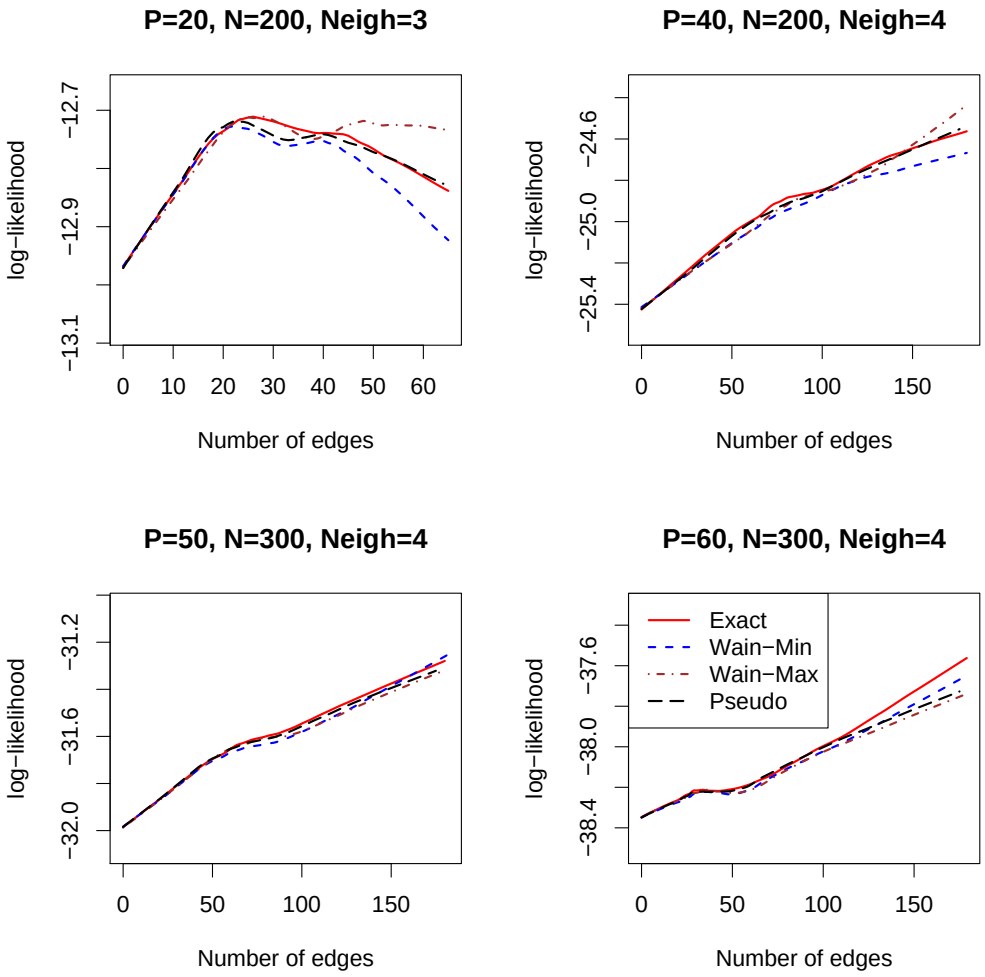

Figure 7: Log-likelihood of the estimated model vs number of edges in the graph for different problem sizes, averaged over 20 simulations.

methods match the exact solution very closely and any differences are well within the 2 standard deviation error band. In Figure 10 the differences of the KL-divergence of the approximate to the exact method can be seen. Again, all methods are very close with the pseudo-likelihood approach performing the best in this case.

6. Discussion

When we embarked on this work, our goal was to find a fast method for maximizing the L_1 penalized log-likelihood of binary-valued Markov networks. We succeeded in doing this, and found that the resulting procedure is faster than competing exact methods. However, in the course of this work,

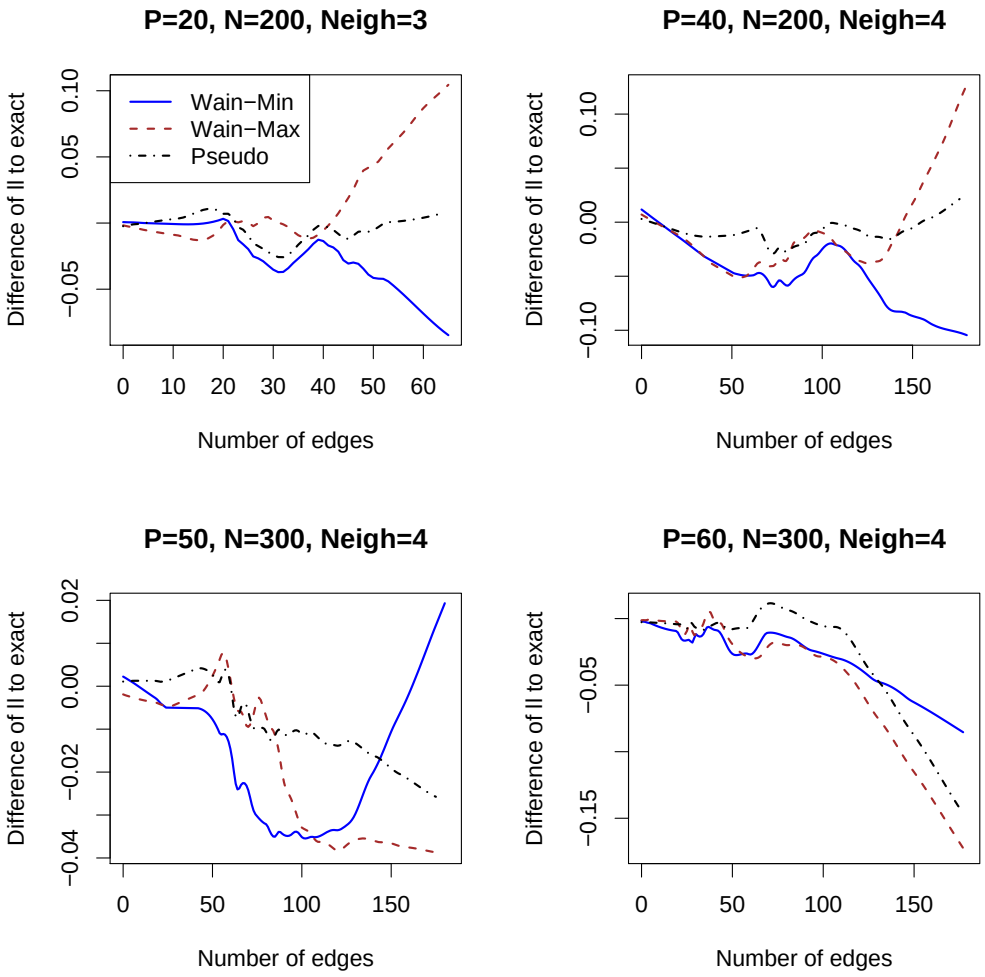

Figure 8: Difference of log-likelihood of the estimated model to the exact model vs number of edges in the graph for different problem sizes, averaged over 20 simulations.

we also learned something surprising: several approximate methods exist that are *much* faster and only slightly less accurate than the exact methods. In addition, when a dense solution is required, the exact methods become infeasible while the approximate methods can still be used. Our implementation of the methods of Wainwright et al. (2006) uses the fast coordinate descent procedure of Friedman, Hastie, and Tibshirani (2008b), a key to its speed. The pseudo-likelihood algorithm also uses similar techniques, which make it very fast as well. We conclude that the Wainwright and pseudo-likelihood methods should be seriously considered for computation in Markov networks.

In this article, we treated the case of pairwise Markov networks with a binary response variable. We think these methods can also be extended to more general cases. With respect to the response variables, a multinomial instead of a binary response could be used. In addition to this, it would

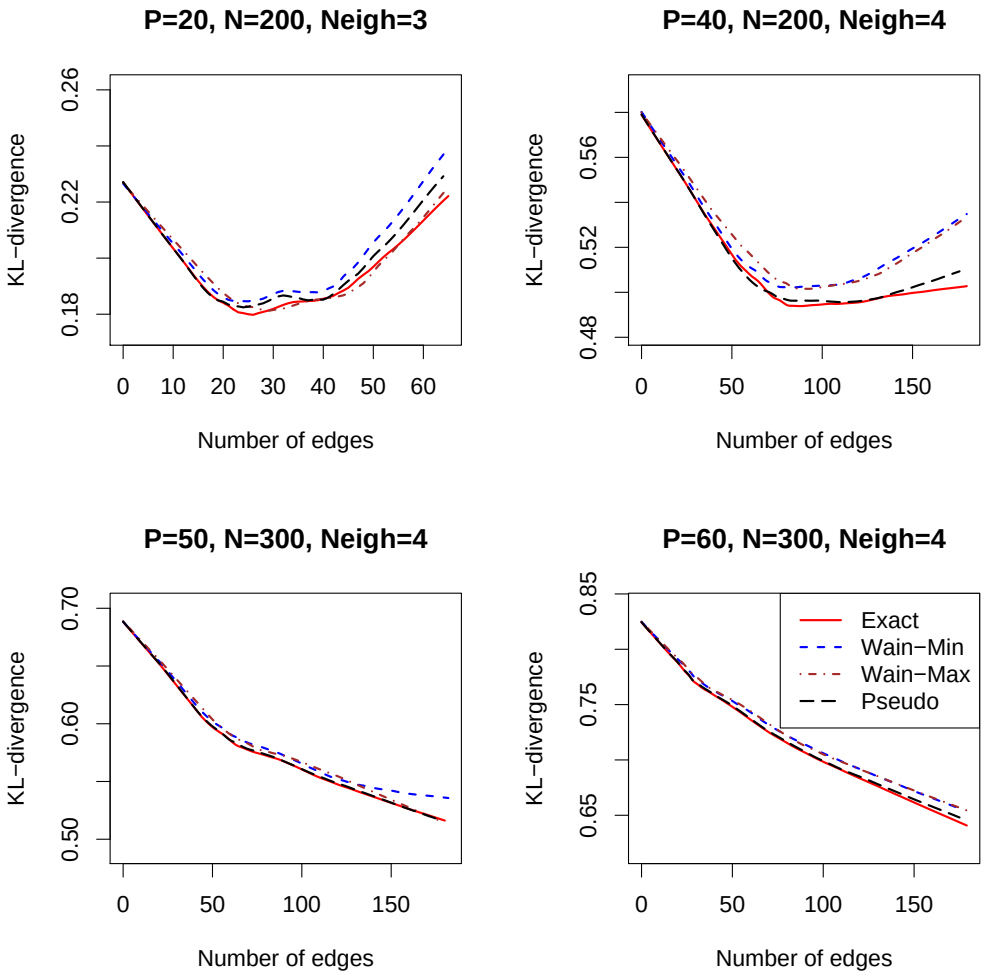

Figure 9: Kullback-Leibler divergence of the estimated model vs. number of edges in the graph for different problem sizes, averaged over 20 simulations.

also be possible to generalize the graph structure by introducing higher order interaction terms. Apart from these extensions, an interesting possibility for future work would also be to prove the theoretical results of Wainwright et al. (2006) for the pseudo-likelihood model. Furthermore, we believe that both the exact and fast approximate methods can also be applied to the learning of multilayer generative models, such as restricted Boltzmann machines (see Hinton, 2007).

An R language package for fitting sparse graphical models, both by exact and approximate methods, will be made available on the authors' websites.

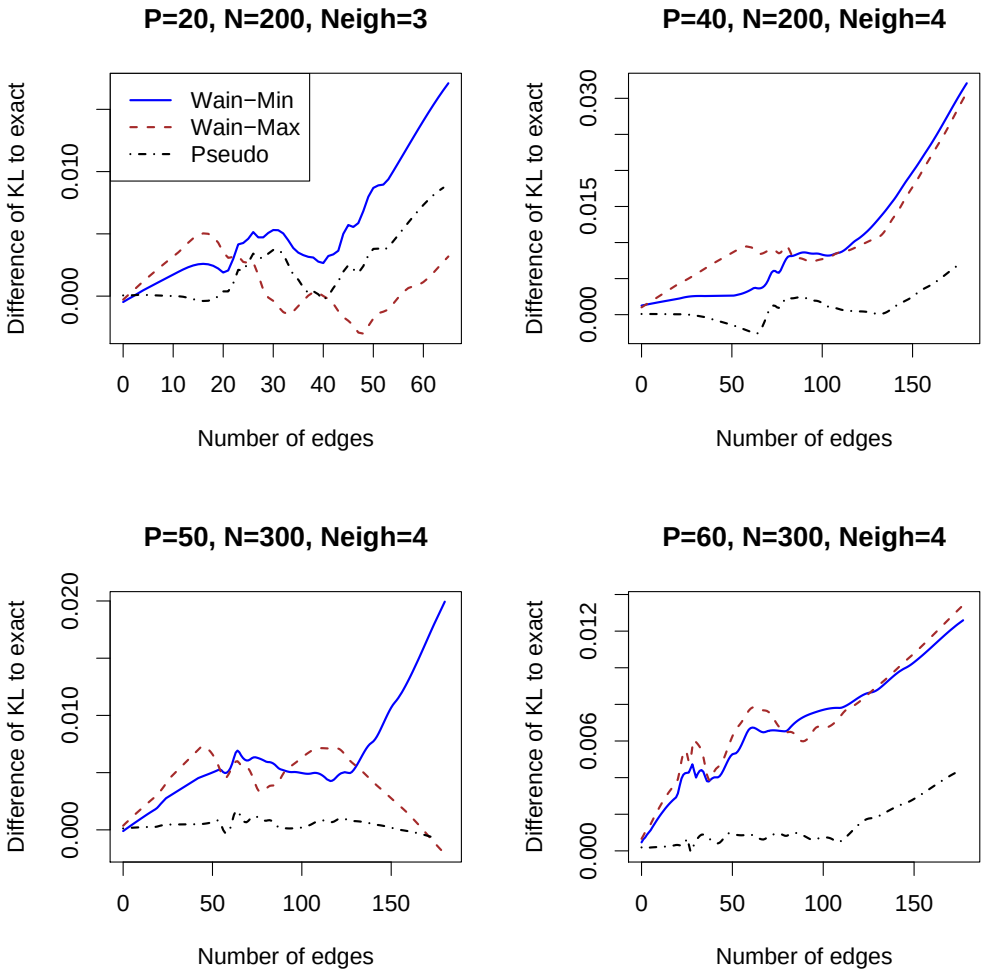

Figure 10: Difference of Kullback-Leibler divergence of the approximate methods to the exact solution vs. number of edges in the graph for different problem sizes, averaged over 20 simulations.

Acknowledgments

Hoefling was supported by a Stanford Graduate Fellowship. Tibshirani was partially supported by National Science Foundation Grant DMS-9971405 and National Institutes of Health Contract N01-HV-28183. We would like to thank Trevor Hastie and Jerome Friedman for code implementing the fast L_1 logistic regression and Daniela Witten for helpful comments on a draft of this article. We also want to thank the editor and two anonymous reviewers for their work and helpful reviews.

Appendix A. Proof of Convergence for Penalized Pseudo-likelihood and Penalized Log-likelihood Algorithms

Lee, Lee, Abbeel, and Ng (2006b) gives a proof of convergence for an algorithm that solves an L_1 constrained problem by quadratic approximation of the objective function. Here, we will follow this proof very closely and make few changes to accommodate that we are only using a first order approximation and are working with the Lagrangian form of the L_1 constrained problem instead of the standard form.

Assume that $g(\Theta)$ is a strictly convex function with a global minimum that we want to minimize. Furthermore, let $f_{\Theta_0}(\Theta)$ be a first order approximation of g at Θ_0 and assume that $f_{\Theta_0}(\Theta)$ is strictly convex, has a global optimum and is jointly continuous in (Θ_0, Θ). Here, by first order approximation at Θ_0, we mean that $f_{\Theta_0} - g$ is twice continuously differentiable with derivative 0 at Θ_0. Assume that our algorithm works as follows:

---

initialize $\Theta^{(0)}$;
Set k:=0;
while *not converged* **do**
  With current estimate $\Theta^{(k)}$, define local approximation $f_{\Theta^{(k)}}(\Theta)$ to g;
  Find solution Θ^* of $f_{\Theta^{(k)}}(\Theta)$;
  Perform backtracking line search on the line from $\Theta^{(k)}$ to Θ^* to find $\Theta^{(k+1)}$;
  Set k:=k+1;
end

Then $\Theta^{(k)}$ converges to the global minimizer of $g(\Theta)$. In order to show this, we first need the following lemma:

Lemma 1 *Let Θ_0 be any point that is not the global optimum. Then there is an open subset S_{Θ_0} and a constant K_{Θ_0} such that for every Φ_0 in S_{Θ_0} every iteration of the algorithm starting at Φ_0 will return a point Φ_1 that improves the objective by at least K_{Θ_0}, that is, $g(\Phi_1) \leq g(\Phi_0) - K_{\Theta_0}$.*

Proof First, let f_{Θ_0} be an approximation to g at Θ_0 with global optimum Θ_1. Then set

$$\delta = f_{\Theta_0}(\Theta_0) - f_{\Theta_0}(\Theta_1)$$

and we know that $\delta > 0$ as Θ_0 is not the global optimum. Now, there exists an $\varepsilon > 0$ such that for $\Phi_0 \in S_{\Theta_0} := \{\Theta : ||\Theta - \Theta_0||_2 < \varepsilon\}$ the following holds:

$$|g(\Theta_0) - g(\Phi_0)| \leq \frac{\delta}{8}$$

and

$$|f_{\Phi_0}(\Phi_1) - f_{\Theta_0}(\Theta_1)| \leq \frac{\delta}{4}$$

where Φ_1 is the global minimum of f_{Φ_0}. The existence of this ε follows from the continuity and convexity of f and g. Then

$$\begin{aligned} f_{\Phi_0}(\Phi_1) &\leq f_{\Theta_0}(\Theta_1) + \frac{\delta}{4} \leq f_{\Theta_0}(\Theta_0) - \frac{3\delta}{4} = \\ &= g(\Theta_0) - \frac{3\delta}{4} \leq g(\Phi_0) - \frac{\delta}{2}. \end{aligned}$$

With step size $0 < t < 1$ in the line search, and using the previous result, it holds that

$$f_{\Phi_0}(\Phi_0 + t(\Phi_1 - \Phi_0)) \leq (1-t) f_{\Phi_0}(\Phi_0) + t f_{\Phi_0}(\Phi_1) \leq g(\Phi_0) - t\frac{\delta}{2}$$

For the next step observe that the minimizer Φ_1 of f_{Φ_0} is a continuous function of Φ_0 due to the convexity of f in the second argument and the continuity in both arguments. Then, as S_{Θ_0} is a compact set, there exists a compact set T_{Θ_0} with $\Phi_1(\Phi_0) \in T_{\Theta_0}$ for all $\Phi_0 \in S_{\Theta_0}$. Thus, as f_{Θ_0} is a first order approximation of g at Θ_0, there exists a C such that for all $\Theta \in T_{\Theta_0}$

$$g(\Theta) \leq f_{\Theta_0}(\Theta) + C||\Theta - \Theta_0||_2^2.$$

Therefore

$$\begin{aligned} g(\Phi_0 + t(\Phi_1 - \Phi_0)) &\leq g(\Phi_0) - t\frac{\delta}{2} + Ct^2||\Phi_1 - \Phi_0||_2^2 \leq \\ &\leq g(\Phi_0) - t\frac{\delta}{2} + t^2 CD^2 \end{aligned}$$

where D is the diameter of T_{Θ_0}. Now set $t^* = \min\left(1, \frac{\delta}{4CD^2}\right)$ and thus we know that it exists a Φ^* such that

$$g(\Phi^*) \leq g(\Phi_0) - t^*\frac{\delta}{2} + t^{*2} CD^2.$$

Setting $K_{\Theta_0} = t^*\frac{\delta}{2} - t^{*2}CD^2 > 0$ now finishes the proof. ■

Now, using the lemma the rest of the proof is again very similar as in Lee, Lee, Abbeel, and Ng (2006b) and we only repeat it here for completeness.

Theorem 1 *The algorithm converges in a finite number of steps.*

Proof Pick $\delta > 0$ arbitrary. Let Θ^* the global optimum and Θ_0 the starting point of the algorithm. Then there exists a compact set K such that $g(\Theta) > g(\Theta_0)$ for every $\Theta \notin K$. Define $P_\delta = \{\Theta : ||\Theta - \Theta^*|| \geq \delta\} \cap K$. We will show convergence by showing that the algorithm can only spend a finite number of steps in P_δ. For every Θ in P_δ there exists an open set S_Θ. So

$$P_\delta \subseteq \cup_{\Theta \in P_\delta} S_\Theta$$

As P_δ is compact, Heine-Borel guarantees that there is a finite set Q_δ such that

$$P_\delta \subseteq \cup_{\Theta \in Q_\delta} S_\Theta.$$

Furthermore, as Q_δ is finite, define

$$C_\delta = \min_{\Theta \in Q_\delta} K_\Theta.$$

As the lemma guarantees that every step of the algorithm inside P_δ improves the objective by at least C_δ and a global optimum exists by assumption, the algorithm can at most spend a finite number of steps in P_δ. Therefore, the algorithm has to converge in a finite number of steps. ■

For the penalized pseudo-likelihood algorithm, by definition of the approximation it is evident that it is a first order approximation. The situation for the penalized log-likelihood algorithm is a little more complicated and it will be shown in the next section of the appendix that the proposed approximation is to first order and therefore satisfies the assumptions of the proof.

Appendix B. First Order Approximation of Log-likelihood

In Section 4, we defined a function $f_{\Theta^{(k)}}$ to calculate the next estimate $\Theta^{(k+1)}$. The convergence proof in Appendix A requires that $f_{\Theta^{(k)}}$ is a first order approximation of the objective $l(\Theta|\mathbf{X}) - N||\mathbf{R} * \Theta||_1$. Here, we want to show that this is in fact the case. For this, we need to show that $f_{\Theta^{(k)}} - l(\Theta|\mathbf{X}) + N||\mathbf{R} * \Theta||_1$ is twice continuously differentiable with derivative 0 at $\Theta^{(k)}$.

First, inserting $f_{\Theta^{(k)}}$ from Section 4 yields

$$\begin{aligned}
d_{\Theta^{(k)}} &= f_{\Theta^{(k)}} - l(\Theta|\mathbf{X}) + N||\mathbf{R} * \Theta||_1 = \\
&= \frac{1}{2}\left(\tilde{l}(\Theta|\mathbf{X}) + \sum_{s>t} (\theta_{st} - \theta_{st}^{(k)}) \left((\hat{\mathbf{p}}(\Theta^{(k)})_s^T \mathbf{X})_t + (\hat{\mathbf{p}}(\Theta^{(k)})_t^T \mathbf{X})_s - 2 \cdot N \cdot w_{st}(\Theta^{(k)}) \right) + \right. \\
&\left. + \sum_s (\theta_{ss} - \theta_{ss}^{(k)}) \left(\sum_k \hat{p}_{sk}(\Theta^{(k)}) + (\mathbf{X}^T\mathbf{X})_{ss} - 2 \cdot N \cdot w_{ss}(\Theta^{(k)}) \right) \right) - \\
&- \sum_{s \geq t} \gamma(\theta_{st} - \theta_{st}^{(k)})^2 - l(\Theta|\mathbf{X})
\end{aligned}$$

which has derivative

$$\begin{aligned}
2\frac{\partial d_{\Theta^{(k)}}}{\partial \theta_{st}} &= 2(\mathbf{X}^T\mathbf{X})_{st} - (\hat{\mathbf{p}}(\Theta)_s^T \mathbf{X})_t - (\hat{\mathbf{p}}(\Theta)_t^T \mathbf{X})_s + (\hat{\mathbf{p}}(\Theta^{(k)})_s^T \mathbf{X})_t + (\hat{\mathbf{p}}(\Theta^{(k)})_t^T \mathbf{X})_s - \\
&\quad - 2 \cdot N \cdot w_{st}(\Theta^{(k)}) - 4\gamma(\theta_{st} - \theta_{st}^{(k)}) - 2(\mathbf{X}^T\mathbf{X})_{st} + 2 \cdot N \cdot w_{st}(\Theta)
\end{aligned}$$

for $s \neq t$ and

$$\begin{aligned}
2\frac{\partial d_{\Theta^{(k)}}}{\partial \theta_{st}} &= (\mathbf{X}^T\mathbf{X})_{ss} - \sum_k \hat{p}_{sk}(\Theta) + \sum_k \hat{p}_{sk}(\Theta^{(k)}) + (\mathbf{X}^T\mathbf{X})_{ss} - 2 \cdot N \cdot w_{ss}(\Theta^{(k)}) - \\
&\quad - 4\gamma(\theta_{ss} - \theta_{ss}^{(k)}) - 2(\mathbf{X}^T\mathbf{X})_{ss} + 2 \cdot N \cdot w_{ss}(\Theta)
\end{aligned}$$

for $s = t$. These are clearly continuous and differentiable. Furthermore, inserting $\Theta = \Theta^{(k)}$ yields that the derivative is 0. Therefore, $f_{\Theta^{(k)}}$ is a first order approximation and our proof holds.

References

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research*, 9:485–516, 2008.

J. Besag. Statistical analysis of non-lattice data. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, volume 24, pages 179–195, 1975.

D. R. Cox and N. Reid. A note on pseudolikelihood constructed from marginal densities. *Biometrika*, 91:729–737, 2004.

J. Dahl, L. Vandenberghe, and V. Roychowdhury. Covariance selection for non-chordal graphs via chordal embedding. Optimization Methods and Software, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441, 2008a.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Technical Report, Stanford University*, 2008b. Submitted.

G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11:428–434, 2007.

S.-I. Lee, V. Ganapathi, and D. Koller. Efficient structure learning of Markov networks using L_1-regularization. In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2006a.

S.-I. Lee, H. Lee, P. Abbeel, and A.Y. Ng. Efficient L_1 regularized logistic regression. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 2006b.

B. Lindsay. Composite likelihood methods. In *Contemporary Mathemtics*, volume 80, pages 221–239, 1998.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2006.

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.

M. J. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using L_1-regularized logistic regression. In *Advances in Neural Information Processing Systems, Vancouver*, 2006.

M. J. Wainwright, P. Ravikumar, and J. Lafferty. High-dimensional graphical model selection using L_1-regularized logistic regression. Technical report, University of California, Berkeley, April 2008.

M. Yuan and Y. Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.