Sparse Gaussian Graphical Models with Unknown Block Structure

Benjamin M. Marlin BMARLIN@CS.UBC.CA
Kevin P. Murphy MURPHYK@CS.UBC.CA
Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver, Canada

Abstract

Recent work has shown that one can learn the structure of Gaussian Graphical Models by imposing an L1 penalty on the precision matrix, and then using efficient convex optimization methods to find the penalized maximum likelihood estimate. This is similar to performing MAP estimation with a prior that prefers sparse graphs. In this paper, we use the stochastic block model as a prior. This prefer graphs that are blockwise sparse, but unlike previous work, it does not require that the blocks or groups be specified a priori. The resulting problem is no longer convex, but we devise an efficient variational Bayes algorithm to solve it. We show that our method has better test set likelihood on two different datasets (motion capture and gene expression) compared to independent L1, and can match the performance of group L1 using manually created groups.

1. Introduction

Estimating a covariance matrix Σ from high dimensional data using a small number of samples is known to be statistically challenging, and yet it is a problem that arises frequently in practice. In the case where the number of samples N is less than the number of dimensions D, the sample covariance matrix S, which is equal to the MLE, is not positive definite. But even when $N > D$, the eigenstructure of the MLE tends to be distorted unless D/N is very small (see e.g., (Dempster, 1972)).

There have been many different attempts to devise regularized estimates of Σ. A very simple approach, which we shall call Tikhonov regularization, is to use $\hat{\Sigma} = S + \nu I$, where $\nu \geq 0$ can be chosen by cross validation or the Ledoit-Wolf formula (Ledoit & Wolf, 2004). An alternative is to form a regularized estimate of the precision matrix, $\Omega = \Sigma^{-1}$ (also called the concentration matrix). A particularly useful approach is based on penalizing the L1 norm of the elements of Ω, to encourage sparsity in the precision matrix; the resulting objective function is convex and can be optimized by a variety of methods (Banerjee et al., 2006; Banerjee et al., 2008; Friedman et al., 2007; Yuan & Lin, 2007; Duchi et al., 2008; Schmidt et al., 2009). Zeros in the precision matrix correspond to absent edges in the corresponding Gaussian graphical model (GGM), so this penalty can be interpreted as preferring graphs that are sparse, that is, which have few edges. However, this approach is different from standard model selection methods for GGMs, such as (Drton & Perlman, 2004), which estimate the graph structure but not the parameters.

For some kinds of data, it is reasonable to assume that the variables can be clustered or grouped into types, which share similar connectivity or correlation patterns. For example, genes can be grouped into pathways, and connections within a pathway might be more likely than connections between pathways. If the group structure is known, one can extend the above L1 penalized likelihood framework in a straightforward way, by penalizing the infinity norm (Duchi et al., 2008) or the two-norm (Schmidt et al., 2009) of each block separately; the resulting objective function is still convex, and encourages blockwise sparse graphs.

In this paper, we present a method that estimates sparse block-structured precision matrices, when the block structure is unknown. We first describe related work in Section 2, and then describe our method in Section 3. In Section 4, we apply our method to two different datasets: the gene expression data used in (Duchi et al., 2008), and a motion capture dataset. Our method outperforms Tikhonov regularization and independent L1 regularization. More interestingly, it

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

can achieve performance that approaches that of the known grouping in both data sets.

2. Related work

One approach to learning sparse GGMs it to form a modified Cholesky decomposition of the precision matrix, $\Sigma^{-1} = B^T D B$, where $B = I - W$, W is a lower triangular matrix of regression weights, $x_j = \mu_j + \sum_{i<j} w_{ij}(x_i - \mu_i)$, and D is a diagonal matrix of variances, and then to impose an L1 penalty (Huang et al., 2006), or a spike-and-slab prior (Smith & Kohn, 2002), on the elements of W. Note that this is equivalent to learning a sparse directed acyclic graph (Shachter & Kenley, 1989), and requires knowing a total ordering of the nodes. This limits the usefulness of the technique for unordered, or cross-sectional, data.

A natural generalization of the sparse DAG approach is to regress each node on all the others, $x_j = w_j^{(0)} + \sum_{i \neq j} w_{ij} x_i$, and then to impose an L1 prior on each row of W and optimize the penalized pseudolikelihood, given by

$$\prod_{n=1}^{N} p(x_{nj}|x_{n,-j}, w_j, \sigma_j^2) + \lambda \sum_{j=1}^{D} ||w_{j,:}||_1$$

where $||w_{j,:}||_1 = \sum_{i \in \{1,\ldots,d\} \setminus \{j\}} |w_{ji}|$ is the one-norm of the weight vector into node j. One can then infer the undirected graph structure from the non-zero elements of W. This technique is a consistent estimator of the graph topology under certain conditions, analogous to those that ensure lasso is model selection consistent for variable selection (Meinshausen & Buhlmann, 2006). We shall refer to this as the "MB" technique. The MB technique is very similar to the dependency net approach proposed in (Heckerman et al., 2000), except we use L1 penalized linear regression rather than decision trees to infer the graph. Given the graph, regression weights can be estimated for each node and assembled into a precision-like matrix. However, this matrix is not necessarily positive definite for small sample sizes, and thus not necessarily a valid precision matrix Ω. One way around this is to compute an MLE of Ω subject to the estimated set of structural zeros using the IPF algorithm (Speed & Kiiveri, 1986) or other convex optimization methods (Dahl et al., 2008).

A technique that can estimate Ω and achieve sparsity at the same time was independently proposed by (Yuan & Lin, 2007) and (Banerjee et al., 2006). In this approach, we impose an L1 penalty on the elements of the precision matrix, and maximize the penalized log likelihood, subject to the constraint that Σ be positive definite. The objective function is given by

$$\frac{1}{2}\log\det(\Omega) - \frac{1}{2N}\sum_{n=1}^{N} x_n^T \Omega x_n - \tfrac{1}{2}\sum_{i=1}^{D}\sum_{j=1}^{D}\lambda_{ij}|\Omega_{ij}|$$
$$\propto \log\det(\Omega) - \mathrm{tr}(S\Omega) - \sum_{i=1}^{D}\sum_{j=1}^{D}\lambda_{ij}|\Omega_{ij}| \quad (1)$$

where $S = \frac{1}{N}\sum_{n=1}^{N} x_n x_n^T$ is the empirical covariance matrix, assuming the data has been mean-centered for notational simplicity. This is a convex objective function, and various efficient algorithms (typically $O(D^3)$ time complexity) have been proposed to solve it (Friedman et al., 2007; Rothman et al., 2008; Duchi et al., 2008; Schmidt et al., 2009). Typically we use a different penalization strength for the diagonal and off-diagonal terms:

$$\log\det(\Omega) - \mathrm{tr}(S\Omega) - \lambda\sum_{i=1}^{D}\sum_{j\neq i}^{D}|\Omega_{ij}| - \nu\sum_{i=1}^{D}|\Omega_{ii}| \quad (2)$$

Note that setting $\lambda = 0$ is equivalent to Tikhonov regularization, since $\Omega_{ii} > 0$ implies

$$\mathrm{tr}((S + \nu I)\Omega) = \mathrm{tr}(S\Omega) + \nu\sum_{i=1}^{D}|\Omega_{ii}|$$

In the case that we have known groups, denoted by S_g, we can extend this objective as follows:

$$\log\det(\Omega) - \mathrm{tr}(S\Omega) - \sum_{g}\lambda_g||\{\Omega_{ij} : (i,j) \in S_g\}||_p \quad (3)$$

where p specifies which norm to apply to the elements in each group. Note that this objective is still convex. If we use $p = \infty$, as in (Duchi et al., 2008), the penalty has the form $\max_{i,j\in S_g}|\Omega_{ij}|$. If we use $p = 2$, as in (Schmidt et al., 2009), the penalty has the form $\sqrt{\sum_{i,j\in S_g}\Omega_{ij}^2}$. This tends to work better, since it forces a block to be sparse when all the elements within "want" to be small, rather than having the behavior of the group be dominated by the largest element. In the limit where each edge is its own group, the group L1 objective reduces to the original L1 objective.

Learning sparse GGMs with unknown block structure is much harder, for reasons we explain in Section 3, and we are not aware of any prior work on this problem. However, there has been work on learning sparse DAG models with unknown block structure. Specifically, (Mansinghka et al., 2006) showed how one can use the stochastic blocks model (Nowicki & Snijders, 2001) as a prior over graphs, combined with a uniform prior over node orderings, to learn block structured DAGs from discrete data. Our work is related to

(Mansinghka et al., 2006), but has the following key differences: we learn undirected graphs rather than DAGs, we learn Gaussian models rather than multinomial models, we use a finite mixture model rather than an infinite one, we use a pseudo likelihood rather than a likelihood when inferring the clustering (but not when estimating Σ), we use variational Bayes instead of MCMC, and we apply our method to real data rather than synthetic data.

3. Method

3.1. Overview

We propose a two stage method for learning sparse GGMs with unknown blocks. In the first stage, we optimize a pseudolikelihood criterion, as in the MB method, combined with a sparsity promoting prior on the weights. The sparsity level of each edge weight, W_{ij}, is controlled by the clusters to which nodes i and j belong, as well as the probability of an edge between these cluster types. Having identified the clusters, we then estimate Σ using the block L1 method.

We give the details of our method below, but first we motivate why we adopted this two stage approach. It is well known that Lasso (L1 regularized linear regression) is equivalent to MAP estimation with a Laplace prior. By analogy, we can see that L1 penalization of each element of the precision matrix corresponds to an independent Laplace prior on each element of Ω, normalized over the space of positive definite matrices $\mathcal{P}$ in D dimensions. More precisely, optimizing Equation 1 is equivalent to MAP estimation with the following prior, where the indicator function $I[\Omega \in \mathcal{P}]$ ensures that the prior is zero for precision matrices that are not positive definite.

$$p(\Omega|\lambda) = \frac{I[\Omega \in \mathcal{P}] \prod_{d=1}^{D} \prod_{d' \geq d}^{D} \lambda_{dd'} \exp(-\lambda_{dd'}|\Omega_{dd'}|)}{\int_{\mathcal{P}} \prod_{d=1}^{D} \prod_{d' \geq d}^{D} \lambda_{dd'} \exp(-\lambda_{dd'}|\Omega_{dd'}|) d\Omega}$$

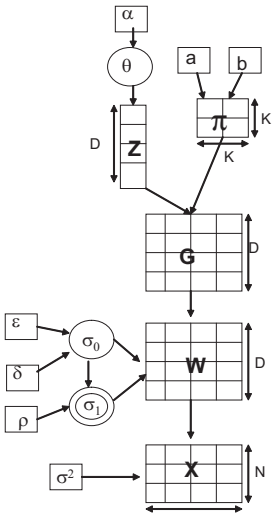
If the prior parameters $\lambda_{dd'}$ are fixed, the normalization term is constant with respect to the precision matrix and can be ignored, leading to an efficiently solvable convex optimization problem. If the prior parameters $\lambda_{dd'}$ are themselves endowed with a structured hierarchical prior, and we wish to adapt these parameters at the same time as we fit the covariance matrix Σ, the normalization term in the Equation above varies and is intractable to compute exactly, unless we restrict attention to the class of decomposable graphical models (see e.g., (Rajarratnam et al., 2008)). One could use a Monte Carlo estimate, as in (Lenkoski & Dobra, 2008), but this would be very expensive, since it would have to be recomputed every time λ changes.

By working with the pseudolikelihood, we can solve a series of related sparse regression problems without worrying about the positive definite constraint. Once we have identified suitable clusters in the data, we then optimize Equation 3. In the following sections, we describe our method in more detail.

3.2. Model

We define our model by explaining the generative process in a top-down fashion. Steps 1-4 define the stochastic block model on undirected graphs (Nowicki & Snijders, 2001). We assume the number of groups K is fixed, and discuss how to estimate this below. The remaining steps explain how we use the graph as a prior for the weights, which are then used to generate the data. The overall model is illustrated in Figure 1(left).

1. We sample the fraction of nodes in each group from a symmetric Dirichlet distribution, $\theta \sim \mathrm{Dir}(\frac{\alpha}{K})$.
2. For each variable d, we sample a group membership using a multinomial distribution, $z_d \sim \mathrm{Multi}(\theta, 1)$, i.e., $p(z_d = k|\theta) = \theta_k$.
3. For each pair of groups k, k', we sample the probability of an edge between them, $\pi_{k,k'} \sim \mathrm{Beta}(a_{k,k'}, b_{k,k'})$.
4. For each pair of distinct variables d, d', we sample an edge between them according to a Bernoulli, $G_{d,d'} \sim \mathrm{Ber}(\pi_{z_d,z_{d'}})$. This results in a symmetric undirected graph.
5. We sample σ_0^2 from a Gamma prior, $\sigma_0^2 \sim \mathrm{Ga}(\epsilon, \delta)$ and set $\sigma_1^2 = \rho\sigma_0^2$ as described below.
6. For each pair of nodes $1 \leq d \leq D, d' \neq d$, we sample an edge weight according to
$$w_{d,d'} \sim \mathcal{N}(0, \sigma_0^2)^{G_{d,d'}} \mathcal{N}(0, \sigma_1^2)^{1-G_{d,d'}}$$
This is similar to the standard "spike and slab" prior used in the Bayesian variable selection literature (George & McCulloch, 1997).
7. Finally we generate the data via a series of independent linear regressions:
$$x_{d,n} \sim \mathcal{N}(w_d^T x_{-d,n}, \sigma^2)$$
where $x_{-d,n}$ is the n'th training vector with the d'th component removed and σ^2 is the overall



$$a^*_{k,k'} \leftarrow a_{k,k'} + \sum_{d=1}^{D}\sum_{d'=1}^{D} \phi_{d,k}\phi_{d',k'}\gamma_{d,d'} \qquad \epsilon^* \leftarrow \epsilon + \frac{D(D-1)}{2}$$

$$b^*_{k,k'} \leftarrow b_{k,k'} + \sum_{d=1}^{D}\sum_{d'=1}^{D} \phi_{d,k}\phi_{d',k'}(1-\gamma_{d,d'}) \qquad \delta^* \leftarrow \delta + \sum_{d=1}^{D}\sum_{d'\neq d} \frac{w^2_{d,d'}}{2}\left(\frac{\gamma_{d,d'}}{\rho} + (1-\gamma_{d,d'})\right)$$

$$\overline{\pi}_{k,k',1} \leftarrow \Psi(a^*_{k,k'}) - \Psi(a^*_{k,k'} + b^*_{k,k'}) \qquad \overline{\pi}_{k,k',0} \leftarrow \Psi(b^*_{k,k'}) - \Psi(a^*_{k,k'} + b^*_{k,k'})$$

$$\phi_{dk} \leftarrow \text{softmax}\left(\sum_{d'\neq d}\sum_{k'=1}^{K} \phi_{d',k'}(\gamma_{d,d'}\overline{\pi}_{k,k',1} + (1-\gamma_{d,d'})\overline{\pi}_{k,k',0}) + \Psi(\alpha^*_k) - \Psi(\sum_{k=1}^{K}\alpha^*_k)\right)$$

$$\gamma_{d,d'} \leftarrow \text{logistic}\left(\sum_{k\leftarrow 1}^{K}\sum_{k'\leftarrow 1}^{K} \phi_{d,k}\phi_{d',k'}(\overline{\pi}_{k,k',1} - \overline{\pi}_{k,k',0}) + \tfrac{1}{2}\left(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2}\right)(w^2_{d,d'} + w^2_{d',d}) - \tfrac{1}{2}\log\rho\right)$$

$$\alpha^*_k \leftarrow \alpha_k + \sum_{d=1}^{D}\sum_{k=1}^{K}\phi_{dk}$$

$$\Lambda_d \leftarrow \text{diag}\left(\frac{\epsilon^*}{\delta^*}\left(\frac{\gamma_{d,d'}}{\rho} + (1-\gamma_{d,d'})\right)\right)$$

$$\hat{w}_d \leftarrow (\sigma^2\Lambda_d + \sum_{n=1}^{N} X^T_{-dn}X_{-dn})^{-1}(\sum_{n=1}^{N} X_{dn}X_{-dn})$$

Figure 1. Summary of the model (left) and the variational updates (right). On the left, small square nodes are fixed hyperparameters. The double-ringed σ_1 node is a deterministic function of σ_0 and ρ. Only the X matrix is observed, all other quantities are inferred.

noise level. (Note that this is just an interpretation of the pseudolikelihood, rather than a proper generative mechanism for the data.)

3.3. Hyperparameters

The model has several hyperparameters which we now discuss. The key hyperparameters that we have identified are the number of clusters K, the weight variance scale parameter δ, and the data noise variance σ^2. The number of clusters K is determined in the learning algorithm using the model free energy and explicit cluster splitting steps as described in Section 3.5. Preliminary cross validation experiments showed that setting $\delta = 0.01$ and $\sigma^2 = 0.1$ led to good performance in terms of the final precision matrix estimate on both of the data sets we consider, and these settings are used throughout the experiments.

We set the Beta hyperparameters on π to $a_{k,k'} = 2$, $b_{k,k'} = 1$ if $k' \neq k$, and $a_{k,k'} = 1$, $b_{k,k'} = 2$ if $k = k'$. This encodes a weak prior favoring a graph structure with more edges between nodes in the same cluster than between nodes in different clusters. We set the weight variance shape parameter to $\epsilon = 1$ and note that it is overwhelmed in the posterior update by the $D(D-1)/2$ factor as described in Section 3.5. We set the Dirichlet parameter $\alpha = 1/K$.

The σ_1^2 parameter is defined to be a constant multiple ρ of σ_0. This construction for σ_0^2 and σ_1^2 is important since its value (in conjunction with σ^2) determines the sparsity of the inferred graph. Intuitively, we determine if an edge $G_{d,d'}$ is present or absent by classifying its corresponding weight $w_{d,d'}$ under the two Gaussian distributions $\mathcal{N}(0, \sigma_0^2)$ and $\mathcal{N}(0, \sigma_1^2)$. We want to choose the variances so that this decision boundary occurs at some "reasonable" distance from the origin to avoid turning on edges if their weights are too small. We choose ρ such that the two pdfs, $\mathcal{N}(0, \sigma_0^2)$ and $\mathcal{N}(0, \sigma_1^2)$, intersect at $c\sigma_0$, i.e., we solve the following for ρ

$$\mathcal{N}(c\sigma_0|0, \sigma_0^2) = \mathcal{N}(c\sigma_0|0, \rho\sigma_0^2)$$

which yields

$$\rho = \exp\left(LW\left[-\frac{c^2}{\exp(c^2)}\right] + c^2\right)$$

where LW is the Lambert W function (the inverse of $f(x) = xe^x$). We require $c > 1$, otherwise the broad $\mathcal{N}(0, \sigma_1^2)$ pdf will not dominate $\mathcal{N}(0, \sigma_0^2)$ in the tails. We chose $c = 2$, which results in a fairly sharp decision boundary, and hence a low entropy posterior on G. (Other approaches for setting σ_0^2 and σ_1^2 are discussed in (George & McCulloch, 1997).)

3.4. Variational Bayes Approximation

The full posterior is proportional to

$$p(Z, \theta, \pi, G, W, \sigma_0, X|K, \sigma^2, \alpha, \epsilon, \delta, \rho, a, b)$$

We use variational Bayes (Ghahramani & Beal, 2000) to approximate the posterior. In particular, we use the

following fully factorized approximation:

$$Q(Z, \theta, \pi, G, W, \sigma_0) = Q(Z)Q(\theta)Q(\pi)Q(G)Q(W)Q(\sigma_0)$$

The individual variational distributions and corresponding variational parameters are as follows:

$$\begin{aligned} Q(Z_d) &= \text{Multi}(\phi_d, 1) \\ Q(\theta) &= \text{Dir}(\alpha^*) \\ Q(\pi_{k,k'}) &= \text{Beta}(a^*_{k,k'}, b^*_{k,k'}) \\ Q(G_{d,d'}) &= \text{Ber}(\gamma_{d,d'}) \\ Q(1/\sigma_0^2) &= \text{Ga}(\epsilon^*, \delta^*) \\ Q(W) &= \delta(W - \hat{w}) \end{aligned}$$

It is important that the quantities that change dimensionality with the number of clusters, namely θ and π, have non-degenerate distributions, otherwise we cannot use the free energy for model selection (see Section 3.5). In particular, if we perform MAP estimation of θ, there is nothing in the model to encourage a small number of clusters, but by putting a distribution on θ, we get an automatic form of complexity control (see e.g. (Bishop, 2006, p480) for discussion). Finally, note that we choose to use point estimation for W since representing the uncertainty in W using a full covariance Gaussian would be intractable (requiring $O(D^4)$ space), and using a diagonal approximation would not provide much benefit over just using a point estimate.

3.5. Learning

We learn the model parameters by optimizing the free energy. The inner loop of the learning algorithm consists of updating the variational parameters of each Q distribution (and several auxiliary parameters) in turn until the free energy converges. The required parameter updates are given in Figure 1(right) and are derived following the usual freeform optimization method (see e.g. (Bishop, 2006, p466)).

The strategy we use to select the number of clusters K consists of starting with all nodes in a single cluster ($K = 1$) and running the variational updates until the free energy converges to within a specified tolerance ($1e-6$). Each time the variational updates converge for a given value of K we consider splitting each of the current clusters. We accept the first split that results in an increase in the variational free energy using a 20 iteration look-ahead. We then continue iterating the variational updates with $K + 1$ clusters. We terminate the algorithm if no split is found that increases the free energy, or the learning algorithm exceeds 1000 iterations of the variational updates.

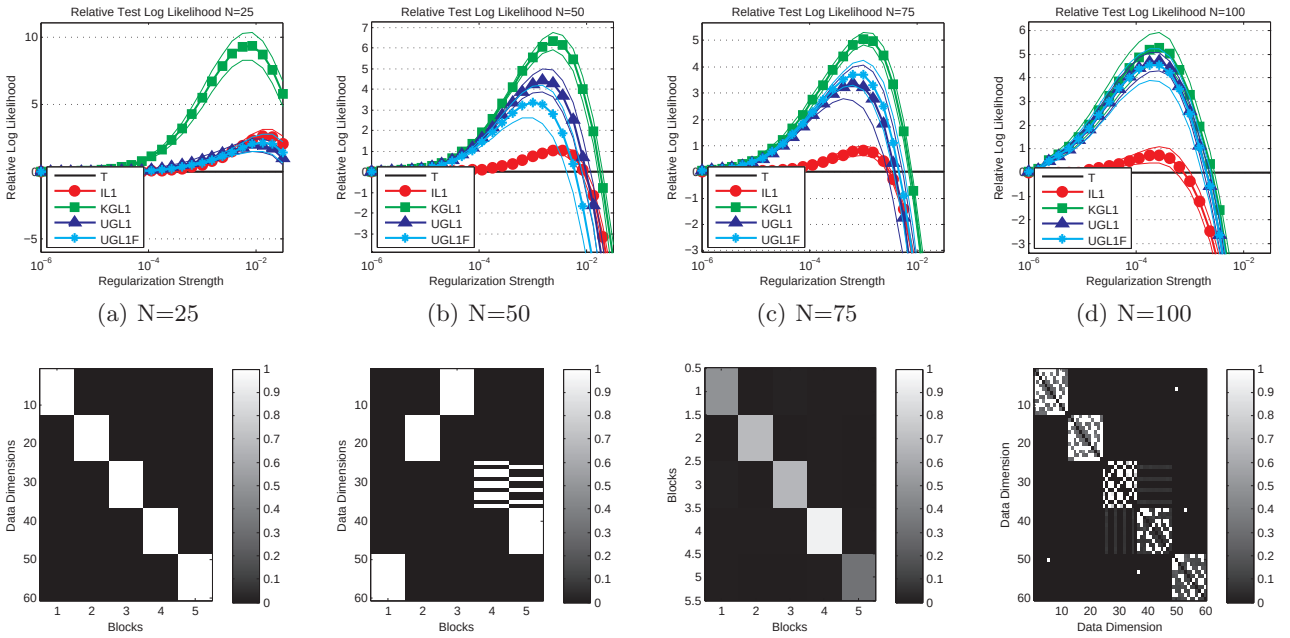
The split proposal mechanism is an important component of the learning method. We propose a split for a given cluster k by deriving a similarity matrix H from the current estimate of the regression weights $\hat{w}$ and applying spectral clustering to H to partition it into two clusters. More precisely, if $S = \{i : z_i = k\}$ is the set of nodes belonging to cluster k, and $\overline{S}$ are the other nodes, we compute the similarity matrix $H = |\hat{w}(S, S)| + 0.5|\hat{w}(S, \overline{S})||\hat{w}(S, \overline{S})^T|$. Intuitively, two data dimensions d and d' are similar under this measure if they are useful for predicting each other (the first term), and there is significant overlap in the subsets of dimensions outside of cluster k that are useful for predicting both d and d' (the second term).

3.6. Estimating the precision matrix

Once our main learning algorithm has run to completion, we compute the marginal MAP assignment of nodes to groups, and then use this known grouping structure as input to the group L1 method to infer the precision matrix. Specifically, we optimize Equation 3 (using the algorithm and code of (Schmidt et al., 2009)), where we use the groups $S_{k,k'} = \{i, j : z_i = k, z_j = k'\}$. Following (Duchi et al., 2008), we set the penalty parameter for each group to $\lambda_g = \lambda|S_g|$, where λ is an overall scale parameter. This ensures that all the blocks have a comparable level of sparsity, regardless of their size.

3.7. Complexity and Fast Update Schedule

The computational bottleneck in our method is solving the D independent ridge regression problems required to update the weight matrix $\hat{w}$, each at a cost of $O((D-1)^3)$. This takes $O(D^4)$ time per iteration of the inner loop of the learning algorithm. However, by using intelligent scheduling of the updates, one can reduce this cost substantially. In particular, we only perform a full update of the $\hat{w}$ matrix every 10 iterations. On the remaining iterations we sort the nodes by the L1 norm of the change in Λ_d and only update the top 10% of nodes. The intuition is that since the only quantity that is changing in the variational update for $\hat{w}_d$ is Λ_d, we don't need to update nodes if the change in Λ_d is small relative to the last time the node was updated. This fast update schedule gives roughly a 10-fold speed improvement, without significantly affecting the quality of the estimates (see Figure 2(a-d)). The fast update schedule leads to a total training time (including precision matrix estimation) of approximately 5 hours on current single CPU hardware for the genes data discussed in Section 4.2 where $D = 667$.



(a) N=25 (b) N=50 (c) N=75 (d) N=100

(e) Known Blocks (f) Clustering $E(z)$ (g) Block connectivity $E(\pi)$ (h) Graph $E(G)$

Figure 2. Results on CMU mocap data ($D = 60$). Figures 2(a) to 2(d) show the test set loglikelihood (relative to Tikhonov) vs λ of Tikhonov (T), Independent L1 (IL1), Known Group L1 (KGL1), Unknown Group L1 (UGL1) and its fast version (UGL1F) for training set sizes $N = 25, 50, 75, 100$. Figures 2(e) to 2(h) show the known block structure, the inferred block structure, the stochastic block model parameters, and the pairwise interaction probabilities for the first fold of the CMU data set with 50 training cases. The color axis in all four figures is scaled from zero (black) to one (white).

4. Experimental results

In this section, we apply our method to two different datasets, and compare its performance with three other methods: Tikhonov regularization, independent L1 regularization, and group L1 regularization with known groups. Since the datasets are small, we use 5-fold cross validation (CV), and compute the mean and standard error of the unpenalized log likelihood on each test fold. We also consider varying the size of the training set. All the data is preprocessed by standardizing it.

To select the strength of the Tikhonov regularizer ν, we use 5-fold CV within each training fold. We use the same value of ν for all the methods, and report performance relative to the Tikhonov baseline. Rather than picking λ for the L1 methods, we plot performance vs λ, as in (Duchi et al., 2008), to better illustrate the differences between the methods.

4.1. Motion capture data

The data set used in this section is based on the "Dance" data set in the CMU motion capture library (available at `http://mocap.cs.cmu.edu/`). This is more expressive than simple walking sequences. The joint angles for all sequences were extracted from the raw motion capture files. The root position and orientation were kept fixed and combined with the remaining joint angles to obtain skeleton positions using the Matlab Motion Capture Toolbox. There are a total of 93 variables defined by the skeleton consisting of 31 markers (x, y, z) triples, however not all markers vary when the root node is fixed, and some, like thumb and front foot markers, are overly correlated with their parent parts in the skeleton. We use a subset of the markers including two markers for the head and neck, and four markers for the trunk, each arm, and each leg. This gives a total of 20 markers and 60 variables.

Skeleton positions within the same motion capture sequence were selected by considering a threshold of 0.1 on the average distance between the current skeleton position and the previously selected skeleton position. This processing was performed to eliminate portions in the sequence where the motion capture subject is not moving.

To enhance the block structure in the data set, the skeleton was manually partitioned into five parts: head and neck, left arm, right arm, left leg, and right leg. A

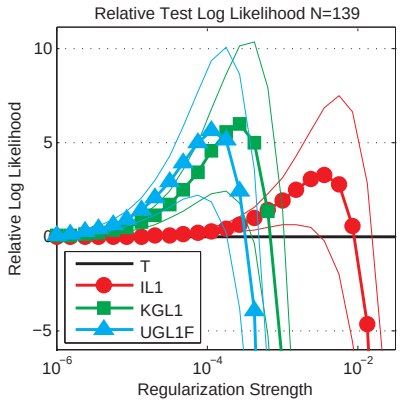


Figure 3. Log likelihood vs λ results on Gene data ($N = 174, D = 667$).

new data set was constructed where each data case is created by sampling parts independently from the empirical distribution over positions for that part. This is roughly equivalent to recording someone as they move their arms and legs independently.

Figure 2 shows the results on this dataset. In the top row, we plot average test set log likelihood vs λ for the different methods, for training set sizes of $N \in \{25, 50, 75, 100\}$. (Recall that $D = 60$.) We draw the following conclusions from these graphs: all the L1 methods are better than Tikhonov for a reasonably broad range of λ; the known groups method is the best; and our method for handling unknown groups achieves performance that is almost as good as known groups, especially for larger sample sizes. (Regular L1 penalized GGMs without grouping was previously applied to mocap data in (Gu et al., 2007), but no quantitative performance measures were given.)

In the bottom row, we examine the model which was learned when $N = 50$. (Results are for one particular test fold; results are similar for the other folds.) In Figure 2(a), we plot the "true" clustering, corresponding to the 5 body parts. In Figure 2(b), we show the estimated clustering. We see that the method chose the correct number of clusters, and assigned most of the nodes to the correct groups, except for a few ambiguous points. (Note that due to label switching, Figure 2(b) need not look identical to Figure 2(a) even if the assignment is perfect.) In Figure 2(c) we plot the π matrix; we see that each cluster is fairly densely connected within itself, but there are essentially no connections between clusters. This is due to the way that the data was created. Finally, in Figure 2(d) we plot the weight matrix W. We see that there are some non-zero off-diagonal elements, even though the prior says that this is unlikely to occur.

4.2. Gene expression data

In this section, we apply our method to the gene expression dataset used in (Duchi et al., 2008), which consists of 174 samples of 667 genes. In (Duchi et al., 2008), the GGM was estimated using a known grouping, where the genes were partitioned into 86 different groups based on prior biological knowledge. In Figure 3, we see that our method results in similar predictive performance to the method which uses the known grouping; and both grouping methods do better than independent L1 on average. However, we note that the latent structure inferred by our method (not shown) contains approximately 30 groups with no obvious relationship to the known structure based on 86 groups. Nevertheless, both groupings improve the regularized estimate of Ω.

5. Conclusions

We have shown how to learn sparse GGMs where the sparsity pattern has a block structure, which we estimate simultaneously with the graph itself. There are several possibilities for future work. One is to try to eliminate the two-step process, by replacing the depedendency network with a Cholesky decomposition of Σ, which is always positive definite, perhaps combined with a search over node orderings, as in (Dobra et al., 2004). Another possibility is to consider discrete data. This introduces the usual computational difficulty of evaluating the likelihood and its gradient, but standard approximations exist for this task (Wainwright et al., 2007; Lee et al., 2006).

Acknowledgments

We would like to thank John Duchi for sharing data and answering questions about his method, Mark Schmidt for useful discussions and sharing code, and Francois Caron for help on an earlier version of this model.

References

Banerjee, O., Ghaoui, L. E., & d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. of Machine Learning Research*, *9*, 485–516.

Banerjee, O., Ghaoui, L. E., d'Aspremont, A., & Natsoulis, G. (2006). Convex optimization techniques for fitting sparse gaussian graphical models. *Intl. Conf. on Machine Learning* (pp. 89–96).

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer.

Dahl, J., Vandenberghe, L., & Roychowdhury, V. (2008). Covariance selection for non-chordal graphs via chordal embedding. *Optimization Methods and Software*, *23*, 501–502.

Dempster, A. (1972). Covariance selection. *Biometrics*, *28*, 157–175.

Dobra, D., Hans, C., Jones, B., Nevins, J., Yao, G., & West, M. (2004). Sparse graphical models for exploring gene expression data. *J. Multivariate analysis*, *90*, 196–212.

Drton, M., & Perlman, M. D. (2004). Model selection for Gaussian concentration graphs. *Biometrika*, *91*, 591–602.

Duchi, J., Gould, S., & Koller, D. (2008). Projected subgradient methods for learning sparse gaussians. *Proc. of the Conf. on Uncertainty in AI*.

Friedman, J., Hastie, T., & Tibshirani, R. (2007). Sparse inverse covariance estimation the graphical lasso. *Biostatistics*, 432–441.

George, E., & McCulloch, R. (1997). Approaches for Bayesian variable selection. *Statistica Sinica*, *7*, 339–373.

Ghahramani, Z., & Beal, M. (2000). Propagation algorithms for variational Bayesian learning. *Advances in Neural Info. Proc. Systems* (pp. 507–513).

Gu, L., Xing, E., & Kanade, T. (2007). Learning gmrf structures for spatial priors. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*.

Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R., & Kadie, C. (2000). Dependency networks for density estimation, collaborative filtering, and data visualization. *J. of Machine Learning Research*, *1*, 49–75.

Huang, J., Liu, N., Pourahmadi, M., & Liu, L. (2006). Covariance selection and estimation via penalized normal likelihood. *Biometrika*, *93*, 85–98.

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *J. of Multivariate Analysis*, *88*, 365–411.

Lee, S.-I., Ganapathi, V., & Koller, D. (2006). Efficient structure learning of Markov networks using L1-regularization. *Advances in Neural Info. Proc. Systems* (pp. 817–824).

Lenkoski, A., & Dobra, A. (2008). *Bayesian structural learning and estimation in Gaussian graphical models* (Technical Report 545). Department of Statistics, University of Washington.

Mansinghka, V., Kemp, C., Tenenbaum, J., & Griffiths, T. (2006). Structured priors for structure learning. *Proc. of the Conf. on Uncertainty in AI*.

Meinshausen, N., & Buhlmann, P. (2006). High dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, *34*, 1436–1462.

Nowicki, K., & Snijders, T. A. B. (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, *96*, 1077–1087.

Rajarratnam, B., Massam, H., & Carvahlo, C. (2008). Flexible covariance estimation in graphical Gaussian models. *Annals of Statistics*, *36*, 2818–2849.

Rothman, A., Bickel, P., Levina, E., & Zhu, J. (2008). Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, *2*, 494–515.

Schmidt, M., van den Berg, E., Friedlander, M., & Murphy, K. (2009). Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm. *AI & Statistics*.

Shachter, R., & Kenley, C. R. (1989). Gaussian influence diagrams. *Managment Science*, *35*, 527–550.

Smith, M., & Kohn, R. (2002). Parsimonious covariance matrix estimation for longitudinal data. *J. of the Am. Stat. Assoc.*, 1141–1153.

Speed, T., & Kiiveri, H. (1986). Gaussian Markov distributions over finite graphs. *Annals of Statistics*, *14*, 138–150.

Wainwright, M., Ravikumar, P., & Lafferty, J. (2007). High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In *Advances in neural info. proc. systems*, 1465–1472.

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