18 A Pathwise Algorithm for Covariance Selection

Vijay Krishnamurthy kvijay@princeton.edu
ORFE, Princeton University
Princeton, NJ 08544, USA

Selin Damla Ahipaşaoğlu sahipasa@princeton.edu
ORFE, Princeton University
Princeton, NJ 08544, USA

Alexandre d'Aspremont aspremon@princeton.edu
ORFE, Princeton University
Princeton, NJ 08544, USA

Covariance selection seeks to estimate a covariance matrix by maximum likelihood while restricting the number of nonzero inverse covariance matrix coefficients. A single penalty parameter usually controls the tradeoff between log-likelihood and sparsity in the inverse matrix. We describe an efficient algorithm for computing a full regularization path of solutions to this problem.

18.1 Introduction

We consider the problem of estimating a covariance matrix from sample multivariate data by maximizing its likelihood while penalizing the inverse covariance so that its graph is *sparse*. This problem is known as covariance selection and can be traced back at least to Dempster (1972). The coefficients of the inverse covariance matrix define the representation of a particular Gaussian distribution as a member of the exponential family; hence sparse maximum likelihood estimates of the inverse covariance yield sparse representations of the model in this class. Furthermore, in a Gaussian model, zeros in the inverse covariance matrix correspond to *conditionally* independent variables, so this penalized maximum likelihood procedure simultaneously stabilizes estimation and isolates *structure* in the underlying graphical model Lauritzen (1996, see).

Given a sample covariance matrix $\Sigma \in \mathbf{S}_n$, the covariance selection problem is written as

$$\text{maximize} \quad \log\det X - \mathbf{Tr}(\Sigma X) - \rho\,\mathbf{Card}(X)$$

in the matrix variable $X \in \mathbf{S}_n$, where $\rho > 0$ is a penalty parameter controlling sparsity and $\mathbf{Card}(X)$ is the number of nonzero elements in X. This is a combinatorially hard (nonconvex) problem and, as in Dahl et al. (2008), Banerjee et al. (2006), and Dahl et al. (2005), we form the convex relaxation

$$\text{maximize} \quad \log\det X - \mathbf{Tr}(\Sigma X) - \rho\|X\|_1, \tag{18.1}$$

which is a convex problem in the matrix variable $X \in \mathbf{S}_n$, where $\|X\|_1$ is the sum of absolute values of the coefficients of X here. After scaling, the $\|X\|_1$ penalty can be understood as a convex lower bound on $\mathbf{Card}(X)$. Another completely different approach, derived in Meinshausen and Bühlmann (2006), reconciles the local dependence structure inferred from n distinct ℓ_1-penalized regressions of a single variable against all the others. Both this approach and the convex relaxation (18.1) have been shown to be consistent in Meinshausen and Bühlmann (2006) and Banerjee et al. (2008), respectively.

In practice, however, both methods are computationally challenging when n gets large. Various algorithms have been employed to solve (18.1) with Dahl et al. (2005), using a custom interior-point method, and Banerjee et al. (2008), using a block coordinate descent method where each iteration required solving a Lasso-like problem, among others. This last method is efficiently implemented in the Glasso package by Friedman et al. (2008), using coordinate descent algorithms from Friedman et al. (2007) to solve the inner regression problems.

One key issue in all these methods is that there is no a priori obvious choice for the penalty parameter. In practice, at least a partial regularization path of solutions has to be computed, and this procedure is then repeated many times to get confidence bounds on the graph structure by cross-validation. Pathwise Lasso algorithms such as LARS (Efron et al., 2004) can be used to get a full regularization path of solution using the method in Meinshausen and Bühlmann (2006), but this still requires solving and reconciling n regularization paths on regression problems of dimension n.

Our contribution here is to formulate a pathwise algorithm for solving problem (18.1) using numerical continuation methods (see Bach et al. (2005) for an application in kernel learning). Each iteration requires solving a large structured linear system (predictor step), then improving precision using a block coordinate descent method (corrector step). Overall, the cost of moving from one solution of problem (18.1) to another is typically much lower than that of solving two separate instances of (18.1). We also derive a coordinate descent algorithm for solving the corrector step, where each iteration is closed form and requires only solving a cubic equation. We illustrate the performance of our methods on several artificial and realistic data sets.

The paper is organized as follows. Section 18.2 reviews some basic convex optimization results on the covariance selection problem in (18.1). Our main pathwise algorithm is described in Section 18.3. Finally, we present some numerical results in Section 18.4.

In what follows, we write $\mathbf{S}_n$ for the set of symmetric matrices of dimension n. For a matrix $X \in \mathbb{R}^{m\times n}$, we write $\|X\|_{\mathrm{F}}$ its Frobenius norm; $\|X\|_1 = \sum_{ij}|X_{ij}|$, the ℓ_1 norm of its vector of coefficients; and $\mathbf{Card}(X)$, the number of nonzero coefficients in X.

18.2 Covariance Selection

Starting from the convex relaxation defined above

$$\text{maximize} \quad \log\det X - \mathbf{Tr}(\Sigma X) - \rho\|X\|_1 \tag{18.2}$$

in the variable $X \in \mathbf{S}_n$, where $\|X\|_1$ can be understood as a convex lower bound on the $\mathbf{Card}(X)$ function whenever $|X_{ij}| \leq 1$ (we can always scale ρ otherwise). Let us write $X^*(\rho)$ for the optimal solution of problem (18.2). In what follows, we will seek to compute (or approximate) the entire regularization path of solutions $X^*(\rho)$ for $\rho \in \mathbb{R}_+$. To remove the nonsmooth penalty, we can set $X = L - M$ and rewrite Problem (18.2) as

$$\begin{array}{ll} \text{maximize} & \log\det(L-M) - \mathbf{Tr}(\Sigma(L-M)) - \rho\mathbf{1}^T(L+M)\mathbf{1} \\ \text{subject to} & L_{ij}, M_{ij} \geq 0, \quad i,j = 1,\ldots,n, \end{array} \tag{18.3}$$

in the matrix variables $L, M \in \mathbf{S}_n$. We can form the following dual to problem (18.2) as

$$\begin{array}{ll} \text{minimize} & -\log\det(U) - n \\ \text{subject to} & U_{ij} \leq \Sigma_{ij} + \rho, \quad i,j = 1,\ldots,n, \\ & U_{ij} \geq \Sigma_{ij} - \rho,, \quad i,j = 1,\ldots,n, \end{array} \tag{18.4}$$

in the variable $U \in \mathbf{S}_n$. As in Bach et al. (2005), for example, in the spirit of barrier methods for interior-point algorithms, we then form the following (unconstrained) regularized problem

$$\min_{U \in \mathbf{S}_n} -\log\det(U) - t\left(\sum_{i,j=1}^{n} \log(\rho + \Sigma_{ij} - U_{ij}) + \sum_{i,j=1}^{n} \log(\rho - \Sigma_{ij} + U_{ij})\right) \tag{18.5}$$

in the variable $U \in \mathbf{S}_n$, and $t > 0$ specifies a desired tradeoff level between centrality (smoothness) and optimality. From every solution $U^*(t)$ corresponding to each $t > 0$, the barrier formulation also produces an explicit *dual* solution $(L^*(t), M^*(t))$ to Problem (18.4). Indeed, we can define matrices $L, M \in \mathbf{S}_n$ as follows

$$L_{ij}(U,\rho) = \frac{t}{\rho + \Sigma_{ij} - U_{ij}} \quad \text{and} \quad M_{ij}(U,\rho) = \frac{t}{\rho - \Sigma_{ij} + U_{ij}}.$$

First-order optimality conditions for Problem (18.5) then imply

$$(L - M) = U^{-1}.$$

As t tends to 0, (18.5) traces a central path toward the optimal solution to Problem (18.4). If we write $f(U)$ for the objective function of (18.4) and call p^* its optimal value, we get (as in Boyd and Vandenberghe (2004, §11.2.2)),

$$f(U^*(t)) - p^* \leq 2n^2 t.$$

Hence t can be understood as a surrogate duality gap when solving the dual Problem (18.4).

18.3 Algorithm

In this section we derive a predictor-corrector algorithm to approximate the entire path of solutions $X^*(\rho)$ when ρ varies between 0 and $\max_i \Sigma_{ii}$ (beyond which the solution matrix is diagonal). Defining

$$H(U,\rho) = L(U,\rho) - M(U,\rho) - U^{-1},$$

we trace the curve $H(U,\rho) = 0$, the first-order optimality condition for Problem (18.5). Our pathwise covariance selection algorithm is defined in Algorithm 18.1.

Typically, in Algorithm 18.1, h is a small constant, $\rho_0 = \max_i \Sigma_{ii}$, and U_0 is computed by solving a single (very sparse) instance of problem (18.5) for

Algorithm 18.1 Pathwise Covariance Selection

```
Input:  $\Sigma \in \mathbf{S}_m$
1: Start with $(U_0, \rho_0)$ s.t $H(U_0, \rho_0) = 0$.
2: for $i = 1$ to $k$ do
3:     Predictor Step. Let $\rho_{i+1} = \rho_i + h$. Compute a tangent direction by solving the
       linear system
           $\frac{\partial H}{\partial \rho}(U_i, \rho_i) + J(U_i, \rho_i)\frac{\partial U}{\partial \rho} = 0$
       in $\partial U/\partial\rho \in \mathbf{S}_n$, where $J(U_i, \rho_i) = \partial H(U, \rho)/\partial U \in \mathbf{S}_{n^2}$ is the Jacobian matrix of the
       function $H(U, \rho)$.
4:     Update $U_{i+1} = U_i + h\partial U/\partial \rho$.
5:     Corrector Step. Solve problem (18.5) starting at $U = U_{i+1}$.
6: end for
Output:  Sequence of matrices $U_i$, $i = 1, \ldots, k$.
```

example.

18.3.1 Predictor: Conjugate Gradient Method

In Algorithm 18.1, the tangent direction in the predictor step is computed by solving a linear system $Ax = b$ where $A = (U^{-1} \otimes U^{-1} + D)$ and D is a diagonal matrix. This system of equations has dimension n^2, and we solve it using the conjugate gradient (CG) method.

18.3.1.1 CG iterations

The most expensive operation in the CG iterations is the computation of a matrix vector product Ap_k, with $p_k \in \mathbb{R}^{n^2}$. Here, however, we can exploit problem structure to compute this step efficiently. Observe that $(U^{-1} \otimes U^{-1})p_k = \text{vec}(U^{-1}P_kU^{-1})$ when $p_k = \text{vec}(P_k)$, so the computation of the matrix vector product Ap_k needs only $O(n^3)$ flops instead of $O(n^4)$. The CG method then needs at most $O(n^2)$ iterations to converge, leading to a total complexity of $O(n^5)$ for the predictor step. In practice, we will observe that CG needs considerably fewer iterations.

18.3.1.2 Stopping criterion

To speed up the computation of the predictor step, we can stop the conjugate gradient solver when the norm of the residual falls below the numerical tolerance t. In our experiments here, we stopped the solver after the residual decreases by two orders of magnitude.

18.3.1.3 Scaling and warm start

Another option, much simpler than the predictor step detailed above, is warm starting. This means simply scaling the current solution to make it feasible for the problem after ρ is updated. In practice, this method turns out to be as efficient as the predictor step, as it allows us to follow the path starting from the sparse end (where more interesting solutions are located). Here, we start the algorithm from the sparsest possible solution, a diagonal matrix U such that

$$U_{ii} = \Sigma_{ii} + (1-\varepsilon)\rho_{\max} I, \quad i = 1, \ldots, n,$$

where $\rho_{\max} = \max_i \Sigma_{ii}$. Suppose, now, that iteration k of the algorithm produced a matrix solution U_k corresponding to a penalty ρ_k. Then, the algorithm with (lower) penalty ρ_{k+1} is started at the matrix

$$U = (1 - \rho_{k+1}/\rho_k)\Sigma + (\rho_{k+1}/\rho_k)U_k,$$

which is a feasible starting point for the corrector problem that follows. This is the method that was implemented in the final version of our code and that is used in the numerical experiments detailed in Section 18.4.

18.3.2 Corrector: Block Coordinate Descent

For small problems, we can use Newton's method to solve (18.5). However, from a computational perspective, this approach is not practical for large values of n. We can simplify iterations by using a block coordinate descent algorithm that updates one row/column of the matrix in each iteration (Banerjee et al., 2008). Let us partition the matrices U and Σ as

$$U = \begin{pmatrix} V & u \\ u^T & w \end{pmatrix} \quad \text{and} \quad S = \begin{pmatrix} A & b \\ b^T & c \end{pmatrix}.$$

We keep V fixed in each iteration and solve for u and w. Without loss of generality, we can always assume that we are updating the last row/column.

18.3.2.1 Algorithm

Problem (18.5) can be written in block format as

$$\begin{aligned} \text{minimize} \quad & -\log(w - u^T V^{-1} u) - t(\log(\rho + c - w) + \log(\rho - c + w)) \\ & -2t\left(\textstyle\sum_i \log(\rho + b_i - u_i) + \sum_i \log(\rho - b_i + u_i)\right), \end{aligned} \tag{18.6}$$

in the variables $u \in \mathbb{R}^{(n-1)}$ and $w \in \mathbb{R}$. Here $V \in \mathbf{S}^{(n-1)}$ is kept fixed in each iteration. We use the Sherman-Woodbury-Morrison (SWM) formula

Algorithm 18.2 Block coordinate descent corrector steps

```
Input:  $U_0,\ \Sigma \in \mathbf{S}_n$
1: for $i = 1$ to $k$ do
2:     Pick the row and column to update.
3:     Solve the inner problem (18.6) using coordinate descent (each coordinate descent
       step requires solving a cubic equation).
4:     Update $U^{-1}$.
5: end for
Output:  A matrix $U_k$ solving (18.5).
```

(see e.g., Boyd and Vandenberghe, 2004, Section C.4.3) to efficiently update U^{-1} at each iteration, so it suffices to compute the full inverse only once, at the beginning of the path. The choice and order of row/column updates significantly affect performance. Although predicting the effect of a whole ith row/column update is numerically expensive, we use the fact that the impact of updating diagonal coefficients usually dominates all others and can be computed explicitly at a very low computational cost. It corresponds to the maximum improvement in the dual objective function that can be achieved by updating the current solution U to $U + we_ie_i^T$, where e_i is the ith unit vector. The objective function value is a decreasing function of w and w must be lower than $\rho + \Sigma_{ii} - U_{ii}$ to preserve dual feasibility, so updating the ith diagonal coefficient will decrease the objective by $\delta_i = (\rho+\Sigma_{ii}-U_{ii})U_{ii}^{-1}$ after minimizing over w. In practice, updating the top 10 percent row/columns with the largest δ is often enough to reach our precision target, and very significantly speeds up computations. We also solve the inner problem (18.6) by a coordinate descent method (as in (Friedman et al., 2007)), taking advantage of the fact that a point minimizing (18.6) over a single coordinate can be computed in closed form by solving a cubic equation. Suppose (u, w) is the current point and that we wish to optimize coordinate u_j of the vector u. We define

$$\begin{aligned} &\alpha = -V_{jj}^{-1} \\ &\beta = -2u_j(\textstyle\sum_{k\neq j} V_{kj}^{-1}u_k) \\ &\gamma = w - u^TV^{-1}u - \alpha u_j - \beta u_j^2. \end{aligned} \tag{18.7}$$

The optimality conditions imply that the the optimal u_j^* must satisfy the cubic equation

$$p_1x^3 + p_2x^2 + p_3x + p_4 = 0, \tag{18.8}$$

where

$$\begin{array}{l} p_1 = 2(1+2t)\alpha,\ p_2 = (1+4t)\beta - 4(1+2t)\alpha b_j \\ p_3 = 4t\gamma - 2(1+2t)\beta b_j + 2\alpha(b_j^2 - 2\rho^2),\ p_4 = \beta(b_j^2 - \rho^2) - 4t\gamma b_j. \end{array}$$

Similarly, the diagonal update w satisfies the following quadratic equation:

$$(1+2t)w^2 - 2(t(u^TV^{-1}u) + c(1+t))w + c^2 - \rho^2 + 2tc(u^TV^{-1}u) = 0$$

Here too, the order in which we optimize the coordinates has a significant impact.

18.3.2.2 Dual Block Problem

We can derive a dual to Problem (18.6) by rewriting it as a constrained optimization problem to get

$$\begin{array}{ll} \text{minimize} & -\log x_1 - t(\log x_2 + \log x_3) - 2t\left(\sum_i(\log y_i + \log z_i)\right) \\ \text{subject to} & x_1 \le w - u^TV^{-1}u \\ & x_2 = \rho + c - w,\ x_3 = \rho - c + w \\ & y_i = \rho + b_i - u_i,\ z_i = \rho - b_i + u_i, \end{array} \tag{18.9}$$

in the variables $u \in \mathbb{R}^{(n-1)}, w \in \mathbb{R}, x \in \mathbb{R}^3, y \in \mathbb{R}^{(n-1)}, z \in \mathbb{R}^{(n-1)}$. The dual to Problem (18.9) is written

$$\begin{array}{ll} \text{maximize} & 1 + 2t(2n-1) + \log\alpha_1 - \alpha_2(\rho + c) - \alpha_3(\rho - c) \\ & -\sum_i\left(\beta_i(\rho + b_i) + \eta_i(\rho - b_i)\right) \\ & +t\log(\alpha_2/t) + t\log(\alpha_3/t) + 2t\left(\sum_i\left(\log(\beta_i/2t) + \log(\eta_i/2t)\right)\right) \\ \text{subject to} & \alpha_1 = \alpha_2 - \alpha_3 \\ & \alpha_1 \ge 0, \end{array} \tag{18.10}$$

in the variables $\alpha \in \mathbb{R}^3, \beta \in \mathbb{R}^{(n-1)}$ and $\eta \in \mathbb{R}^{(n-1)}$. Surrogate dual points then produce an explicit stopping criterion.

18.3.3 Complexity

Solving for the predictor step using conjugate gradient as in Section 18.3.1 requires $O(n^2)$ matrix products (at a cost of $O(n^3)$ each) in the worst case, but the number of iterations necessary to get a good estimate of the predictor is typically much lower (see experiments in Section 18.4). Scaling and warm start, on the other hand, have complexity $O(n^2)$. The inner and outer loops of the corrector step are solved using coordinate descent, with each coordinate iteration requiring the (explicit) solution of a cubic equation.

Results on the convergence of the coordinate descent in the smooth case can be traced back at least to Luo and Tseng (1992) or Tseng (2001), who focus on local linear convergence in the strictly convex case. More precise convergence bounds have been derived in Nesterov (2010), who shows linear convergence (with complexity growing as $\log(1/\varepsilon)$) of a randomized variant of coordinate descent for strongly convex functions, and a complexity bound growing proportionally to $1/\varepsilon$ when the gradient is Lipschitz continuous coordinatewise. Unfortunately, because it uses a randomized step selection strategy, the algorithm in its standard form is inefficient in our case here, as it requires too many SWM matrix updates to switch between columns. Optimizing the algorithm in Nesterov (2010) to adapt it to our problem (e.g., by adjusting the variable selection probabilities to account for the relative cost of switching columns) is a potentially promising research direction.

The complexity of our algorithm can be summarized as follows.

- Because our main objective function is strictly convex, our algorithm converges locally linearly, but we have no explicit bound on the total number of iterations required.
- Starting the algorithm requires forming the inverse matrix V^{-1} at a cost of $O(n^3)$.
- Each iteration requires solving a cubic equation for each coordinatewise minimization problem to form the coefficients in (18.7), at a cost of $O(n^2)$. Updating the problem to switch from one iteration to the next, using SWM updates, then costs $O(n^2)$. This means that scanning the full matrix with coordinate descent requires $O(n^4)$ flops.

While the lack of a precise complexity bound is a clear shortcoming of our choice of algorithm for solving the corrector step, as discussed by Nesterov (2011), algorithm choices are usually guided by the type of operations (projections, barrier computations, inner optimization problems) that can be solved very efficiently or in closed form. In our case here, it turns out that coordinate descent iterations can be performed very fast, in closed form (by solving cubic equations), which seems to provide a clear (empirical) complexity advantage to this technique.

18.4 Numerical Results

We compare the numerical performance of several methods for computing a full regularization path of solutions to Problem (18.2) on several realistic data sets: the senator votes covariance matrix from Banerjee et al. (2006), the *Science* topic model in Blei and Lafferty (2007) with 50 topics, the covariance matrix of 20 foreign exchange rates, the UCI SPECTF heart dataset (diagnosing of cardiac images), the UCI LIBRAS hand movement dataset, and the UCI HillValley dataset. We compute a path of solutions using the methods detailed here (Covpath) and repeat this experiment using the Glasso path code (Friedman et al., 2008), which restarts the covariance selection problem at $\rho+\varepsilon$ at the current solution of (18.2) obtained at ρ. We also tested the smooth first-order code with warm start ASPG (described in Lu (2010)) as well as the greedy algorithm SINCO by Scheinberg and Rish (2009). Note that the latter only identifies good sparsity patterns but does not (directly) produce feasible solutions to problem (18.4). Our prototype code here is written in MATLAB (except for a few steps in C), ASPG and SINCO are also written in MATLAB, and Glasso is compiled from FORTRAN and interfaced with R. We use the scaling/warm start approach detailed in Section 18.3 and scan the full set of variables at each iteration of the block-coordinate descent algorithm (optimizing over the 10 percent most promising variables sometimes significantly speeds up computations but is more unstable), so the results reported here describe the behavior of the most robust implementation of our algorithm. We report CPU time (in seconds) versus problem dimension in Table 18.1. Unfortunately, Glasso does not use the duality gap as a stopping criterion, but rather lack of progress (average absolute parameter change less than 10^{-4}). Glasso fails to converge on the HillValley example.

Dataset	Dimension	Covpath	Glasso	ASPG	SINCO
Interest Rates	20	0.036	0.200	0.30	**0.007**
FX Data	20	**0.016**	1.467	4.88	0.109
Heart	44	**0.244**	2.400	11.25	5.895
ScienceTopics	50	**0.026**	2.626	11.58	5.233
Libras	91	**0.060**	3.329	35.80	40.690
HillValley	100	**0.068**	-	47.22	68.815
Senator	102	**4.003**	5.208	10.44	5.092

Table 18.1: CPU time (in seconds) versus problem type for computing a regularization path for 50 values of the penalty ρ, using the path-following method detailed here (Covpath), the Glasso code with warm-start (Glasso), the pathwise code (ASPG) in Lu (2010) and the SINCO greedy code by Scheinberg and Rish (2009).

As in Banerjee et al. (2008), to test the behavior of the algorithm on examples with known graphs, we also sample sparse random matrices with Gaussian coefficients, add multiples of the identity to make them positive semidefinite, then use the inverse matrix as our sample matrix Σ. We use these examples to study the performance of the various algorithms listed above on increasingly large problems. Computing times are listed in Table 18.2 for a path of length 10, and in Table 18.3 for a path of length 50. The penalty coefficients ρ are chosen to produce a target sparsity around 10 percent.

Dimension	Covpath	Glasso	ASPG	SINCO
20	**0.0042**	2.32	0.53	0.22
50	**0.0037**	0.59	4.11	3.80
100	**0.0154**	1.11	13.36	13.58
200	**0.0882**	4.73	73.24	61.02
300	**0.2035**	13.52	271.05	133.99

Table 18.2: CPU time (in seconds) versus problem dimension for computing a regularization path for 10 values of the penalty ρ, using the path-following method detailed here (Covpath), the Glasso code with warm-start (Glasso), the pathwise code (ASPG) in Lu (2010) and the SINCO greedy code by Scheinberg and Rish (2009) on randomly generated problems.

Dimension	Covpath	Glasso	ASPG	SINCO
20	**0.0101**	0.64	2.66	1.1827
50	**0.0491**	1.91	23.2	22.0436
100	**0.0888**	10.60	140.75	122.4048
200	**0.3195**	61.46	681.72	451.6725
300	**0.8322**	519.05	5203.46	1121.0408

Table 18.3: CPU time (in seconds) versus problem dimension for computing a regularization path for 50 values of the penalty ρ, using the path-following method detailed here (Covpath), the Glasso code with warm-start (Glasso), the pathwise code (ASPG) in Lu (2010) and the SINCO greedy code by Scheinberg and Rish (2009) on randomly generated problems.

In Figure 18.1, we plot the number of nonzero coefficients (cardinality) in the inverse covariance versus the penalty parameter ρ, along a path of solutions to problem (18.2). We observe that the solution cardinality appears to be linear in the log of the regularization parameter. We then plot the number of conjugate gradient iterations required to compute the predictor in Section 18.3.1 versus number of nonzero coefficients in the inverse covariance matrix. We notice that the number of CG iterations decreases significantly for sparse matrices, which makes computing predictor directions faster at the sparse (i.e., interesting) end of the regularization path. Nevertheless, the complexity of corrector steps dominates the total complexity of the algorithm, and there was little difference in computing time between using the scaling method detailed in Section 18.3 and using the predictor step. Hence the final version of our code and the CPU time results listed here make use of scaling/warm start exclusively, which is more robust.

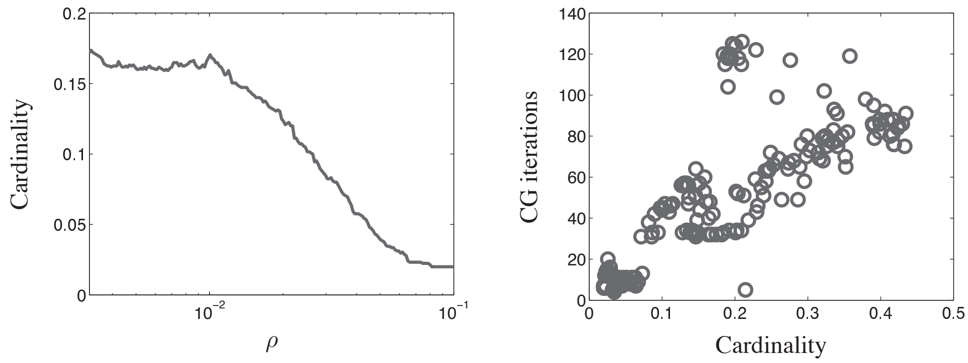

Figure 18.1: *Left:* We plot the fraction of nonzero coefficients in the inverse covariance versus penalty parameter ρ, along a path of solutions to Problem (18.2). *Right:* Number of conjugate gradient iterations required to compute the predictor step versus number of nonzero coefficients in the inverse covariance matrix.

Finally, to illustrate the method on intuitive data sets, we solve for a full regularization path of solutions to Problem (18.2) on financial data consisting of the covariance matrix of U.S. forward rates for maturities ranging from 6 months to 10 years from 1998 until 2005. Forward rates move as a curve, so we expect their inverse covariance matrix to be close to band diagonal. Figure 18.2 shows the dependence network obtained from the solution of Problem (18.2) on this matrix along a path, for $\rho = .02$, $\rho = .008$, and $\rho = .006$. The graph layout was formed using the yFiles–Organic option in Cytoscape.

The string like dynamics of the rates clearly appear in the last plot. We also applied our algorithm to the covariance matrix extracted from the correlated topic model calibrated in Blei and Lafferty (2007) on 10 years of articles from the journal *Science*, targeting a graph density low enough to reveal some structure. The corresponding network is detailed in Figure 18.3. Graph edge color is related to the sign of the conditional correlation (green for positive, red for negative), while edge thickness is proportional to the correlation magnitude. The five most important words are listed for each topic.

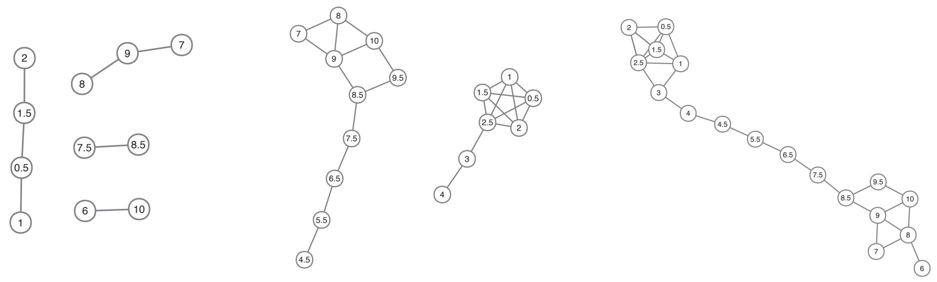

Figure 18.2: Three sample dependence graphs corresponding to the solution of problem (18.2) on a U.S. forward rates covariance matrix for $\rho = .02$ (left), $\rho = .008$ (center), and $\rho = .006$ (right).

18.5 Online Covariance Selection

In this section we will briefly discuss the *online* version of the covariance selection problem. This version arises if we obtain a better estimate of the covariance matrix after the problem is already solved for a set of parameter values. We will assume that the new (positive definite) covariance matrix $\hat{\Sigma}$ is the sum of the old covariance matrix Σ and an arbitrary symmetric matrix C. With such a change, the "new" dual problem can be written as

$$\begin{array}{ll} \text{minimize} & -\log\det(U) - n \\ \text{subject to} & U_{ij} \le \rho + \Sigma_{ij} + \mu C_{ij}, \quad i,j = 1,\ldots,n, \\ & U_{ij} \ge \Sigma_{ij} + \mu C_{ij} - \rho,, \quad i,j = 1,\ldots,n, \end{array} \tag{18.11}$$

in the variable $U \in \mathbf{S}_n$, where ρ is a parameter value for which the corresponding optimal solution is already calculated with the old covariance matrix Σ. The problem is parameterized with μ, so that $\mu = 0$ gives the original problem whereas $\mu = 1$ corresponds to the new problem.

For many applications, one would expect C to be small and the optimal solution U^* of the original problem to be close to the optimal solution of the new problem, say $\hat{U}^*$. Hence, regardless of the algorithm, U^* should be used as an initial solution instead of solving the problem from scratch.

In the spirit of the barrier methods and the predictor-corrector method that we have devised in this chapter, we can develop a predictor-corrector algorithm to solve the online version of the problem fast, as follows. We form a parameterized version of the regularized problem

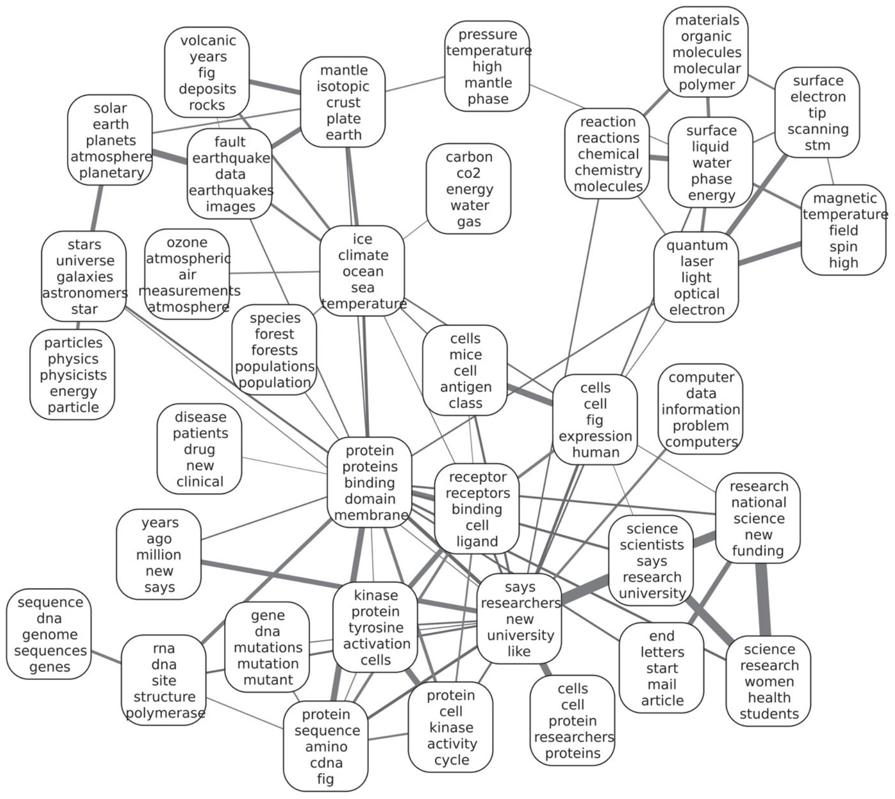
volcanic years fig deposits rocks
mantle isotopic crust plate earth
pressure temperature high mantle phase
materials organic molecules molecular polymer
surface electron tip scanning stm
solar earth planets atmosphere planetary
fault earthquake data earthquakes images
reaction reactions chemical chemistry molecules
surface liquid water phase energy
carbon co2 energy water gas
magnetic temperature field spin high
stars universe galaxies astronomers star
ozone atmospheric air measurements atmosphere
ice climate ocean sea temperature
quantum laser light optical electron
species forest forests populations population
particles physics physicists energy particle
cells mice cell antigen class
cells cell fig expression human
computer data information problem computers
disease patients drug new clinical
protein proteins binding domain membrane
receptor receptors binding cell ligand
research national science new funding
years ago million new says
science scientists says research university
sequence dna genome sequences genes
kinase protein tyrosine activation cells
says researchers new university like
gene dna mutations mutation mutant
rna dna site structure polymerase
end letters start mail article
science research women health students
protein sequence amino cdna fig
protein cell kinase activity cycle
cells cell protein researchers proteins

Figure 18.3: Topic network for the Science Correlated Topic Model in Blei and Lafferty (2007). Network layout using cytoscape. Graph edge grayscale is related to the magnitude of the conditional correlation while edge thickness is proportional to the correlation magnitude.

$$\begin{array}{ll} \min\limits_{U \in \mathbf{S}_n} & -\log\det(U) - t\sum_{i,j=1}^{n} \log(\rho + \Sigma_{ij} + \mu C_{ij} - U_{ij}) \\ & -t\sum_{i,j=1}^{n} \log(\rho - \Sigma_{ij} - \mu C_{ij} + U_{ij}) \end{array} \tag{18.12}$$

in the variable $U \in \mathbf{S}_n$, and $t > 0$ is the tradeoff level as before. Let us define matrices $\hat{L}, \hat{M} \in \mathbf{S}_n$ as follows:

$$\hat{L}_{ij}(U,\mu) = \frac{t}{\rho + \Sigma_{ij} + \mu C_{ij} - U_{ij}} \quad \text{and} \quad \hat{M}_{ij}(U,\mu) = \frac{t}{\rho - \Sigma_{ij} - \mu C_{ij} + U_{ij}}.$$

As before, optimal $\hat{L}$ and $\hat{M}$ should satisfy $(\hat{L} - \hat{M}) = U^{-1}$, and Problem (18.12) traces a central path toward the optimal solution to Problem (18.11) as t goes to 0.

Defining

$$\hat{H}(U,\mu) = \hat{L}(U,\mu) - \hat{M}(U,\mu) - U^{-1},$$

we trace the curve $\hat{H}(U,\mu) = 0$, the first-order optimality condition for problem (18.12), from the solution for the original problem to one for the new problem as μ goes from 0 to 1. The resulting predictor-corrector algorithm is Algorithm 18.3, which solves the online version efficiently.

Algorithm 18.3 Online Pathwise Covariance Selection

```
Input: $\Sigma, U^* \in \mathbf{S}_m$, $\rho \in \mathbb{R}$, and $c \in \mathbb{R}^{n\times r}$.
1: Start with $(U_0,\mu_0)$ s.t $\hat{H}(U_0,\mu_0) = 0$, specifically, set $\mu_0 = 0$ and $U_0 = U^*$.
2: for $i = 1$ to $k$ do
3:     Predictor Step. Let $\mu_{i+1} = \mu_i + 1/k$. Compute a tangent direction by solving the
       linear system
           $\frac{\partial \hat{H}}{\partial \mu}(U_i,\mu_i) + J(U_i,\mu_i)\frac{\partial U}{\partial \mu} = 0$
       in $\partial U/\partial\mu \in \mathbf{S}_n$, where $J(U_i,\mu_i) = \partial\hat{H}(U,\mu)/\partial U \in \mathbf{S}_{n^2}$ is the Jacobian matrix of the
       function $\hat{H}(U,\mu)$.
4:     Update $U_{i+1} = U_i + (\partial U/\partial\mu)/k$.
5:     Corrector Step. Solve Problem (18.12) for $\mu_{i+1}$ starting at $U = U_{i+1}$.
6: end for
Output: Matrix $U_k$ that solves Problem (18.11).
```

As for the offline version, the most demanding computation in this algorithm is the calculation of the tangent direction, which can be carried out by the CG method discussed above. When carefully implemented and tuned, it produces a solution for the new problem very fast. Although one can try different values of k, setting $k = 1$ and applying one step of the algorithm is usually enough in practice. This algorithm, and the online approach discussed in this section in general, would be especially useful and sometimes necessary for very large datasets, as solving the problem from scratch is an expensive task for such problems and should be avoided whenever possible.

Acknowledgements

The authors are grateful to two anonymous referees whose comments significantly improved the chapter. The authors would also like to acknowledge support from NSF grants SES-0835550 (CDI), CMMI-0844795 (CAREER), CMMI-0968842, a Peek junior faculty fellowship, a Howard B. Wentz Jr. award, and a gift from Google.

18.6 References

F. Bach, R. Thibaux, and M. Jordan. Computing regularization paths for learning multiple kernels. In *Advances in Neural Information Processing Systems*, volume 17, pages 73–80. MIT Press, 2005.

O. Banerjee, L. El Ghaoui, A. d'Aspremont, and G. Natsoulis. Convex optimization techniques for fitting sparse Gaussian graphical models. In *Proceedings of the 23rd 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.

D. Blei and J. Lafferty. A correlated topic model of science. *Annals of Applied Statistics*, 1(1):17–35, 2007.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

J. Dahl, V. Roychowdhury, and L. Vandenberghe. Maximum likelihood estimation of gaussian graphical models: numerical implementation and topology selection. *UCLA preprint*, 2005.

J. Dahl, L. Vandenberghe, and V. Roychowdhury. Covariance selection for non-chordal graphs via chordal embedding. *Optimization Methods and Software*, 23 (4):501–520, 2008.

A. Dempster. Covariance selection. *Biometrics*, 28(1):157–175, 1972.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

S. Lauritzen. *Graphical Models*. Clarendon Press, 1996.

Z. Lu. Adaptive first-order methods for general sparse inverse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2000–2016, 2010.

Z. Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *CORE Discussion Papers*, 2010/2, 2010.

Y. Nesterov. Barrier subgradient method. *Mathematical Programming, Series B*, 127:31–56, 2011.

K. Scheinberg and I. Rish. SINCO—a greedy coordinate ascent method for sparse inverse covariance selection problem. 2009.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.