Shrinking Characteristics of Precision Matrix Estimators: An Illustration via Regression

Matt Galloway

2019-04-24

Contents

1 Introduction **3**

2 Precision Matrix Estimation **6**

2.1 Background 6

2.2 ADMM Algorithm 9

2.3 Simulations 12

3 SCPME **16**

3.1 Augmented ADMM Algorithm 17

3.2 Regression Illustration 20

3.3 Simulations 21

3.4 Discussion 27

A Appendix **30**

B `ADMMsigma` R Package **39**

B.1 Installation 40

B.2 Tutorial 40

C `SCPME` R Package **44**

C.1 Installation 45

C.2 Tutorial 45

Preface

I would like to thank my advisor Professor Adam Rothman for his guidance and support this past year. His enthusiasm towards this research area is ultimately what got me interested and I truly enjoyed having the opportunity to learn from him.

My parents also deserve a note of appreciation and thanks – one probably much longer than this. Graduate school would not have been possible without their love and encouragement.

Chapter 1

Introduction

Inferential statistics consists of two primary components: estimation and prediction. Estimation refers to the process by which we infer values, properties and behavior about individual parameters in our model. Prediction, on the other hand, refers to the process by which we draw inference about an event that has not-yet occurred. The latter component is abundant in industry where the demand to predict stock prices, movies individuals prefer, and even to predict new friend connections in a network has sparked huge investments in research and given rise to research groups whose primary goal is pushing the bounds and developing new predictive models.

However, though most attention is directed towards prediction, these two components are not distinct. In fact, the success and reliability of many predictive models is contingent upon good, efficient parameter estimation. Many of these models require the estimation of a precision matrix - the inverse of the covariance matrix (frequently denoted as Ω) - that establishes the interaction and covariance between random variables. For this reason, the last decade has seen an ever-expanding community devoted to precision matrix estimation.

Among this community of researchers is Professor Adam Rothman and Aaron Molstad, Ph.D. whose research will be a focal point of this manuscript. The two have published work on indirect multivariate linear regression (Molstad and Rothman (2016)) and classification with matrix-valued predictors (Molstad and Rothman (2018)) but the focus of this manuscript is their 2017 paper titled *Shrinking Characteristics of Precision Matrix Estimators* (Molstad and Rothman (2017)). In it, they outline a framework to estimate *characteristics* of precision matrices - a concept that exploits the fact that in many predictive models estimation of the precision matrix is only neces-

sary through its product with another feature. They write in their manuscript that "to fit many predictive models, only a characteristic of the population precision matrix needs to be estimated... In binary linear discriminant analysis, the population precision matrix is needed for prediction only through the product of the precision matrix and the difference between the two conditional distribution mean vectors." The purpose of the research detailed here began with the desire to expand on this concept and to explore avenues that were mentioned but were not further investigated.

One of the research directions mentioned in the original paper was the application of their framework to regression. Utilizing the fact that the population regression coefficient matrix $\beta \equiv \Omega_x \Sigma_{xy}$ (where Σ_{xy} is the cross-covariance matrix between the predictors, X, and the responses, Y, and Ω_x is the precision matrix for X), their framework allows for the simultaneous estimation of β and Ω_x with an embedded assumption potentially useful for superior prediction performance. In close communication and collaboration with Professor Rothman, we wanted to explore this research direction further. However, in order to build upon their work and contribute new material, there were a number of concepts that needed to be learned along the way and this document will follow that journey.

We will begin chapter two with a brief introduction to precision matrix estimation and the gaussian log-likelihood function. This section will mention popular estimation methods and algorithms but most discussion will be directed towards the ADMM algorithm. Discussion of the ADMM algorithm will be useful as we begin detailing the shrinking characteristics of precision matrix estimators framework (which may be referred to as SCPME), the so-called augmented ADMM algorithm, and later the framework's application to regression. Lastly, the document will end with two brief tutorials for the R packages `ADMMsigma` and `SCPME`. These packages were developed by myself to aid in simulation experiments and make it easier to branch into related research directions. Both packages have since been published on CRAN.

1.0.1 Notation and Definitions

For strictly positive integers n and p, we will denote $\mathbb{R}^{n \times p}$ as the class of real matrices with dimenson $n \times p$. The class of real, symmetric matrices with dimension $p \times p$ will be denoted as $\mathbb{S}^p$ and $\mathbb{S}^p_+$ if we further require the object to be positive definite. The sample size and dimension of the predictor vector in a given data set will most often be denoted as n and p, respectively. If the dimension of the response vector exceeds one, we will denote it as r.

Most matrices will take the form of either Σ, the population covariance matrix, or Ω, the population precision matrix. Note that the precision matrix is simply the inverse of the covariance matrix ($\Omega \equiv \Sigma^{-1}$) and a subscript may be added to each if more than two random variables are considered in a problem (Ω_x). A subscript star may also be added if the object is oracle - or known - a priori (Ω_*). The oracle's estimator that optimizes a pre-specified objective function will be denoted with a hat ($\widehat{\Omega}$).

There will be significant matrix algebra notation throughout the manuscript. The trace operator sums the diagonal elements of a matrix and will take the form $tr(\cdot)$ and the *exponential* trace operator will be denoted similarly as $etr(\cdot)$. The vector operator, $vec(\cdot)$, stacks the columns of a matrix into a column vector. The determinant of a matrix $\mathbf{A}$ will be denoted as $|\mathbf{A}|$ but may also take the form $det(\mathbf{A})$. The kronecker product of two matrices $\mathbf{A}$ and $\mathbf{B}$ will be denoted as $\mathbf{A} \otimes \mathbf{B}$ and the element-wise product will be denoted as $\mathbf{A} \circ \mathbf{B}$. Lastly, the Frobenius norm which sums the square of all entries in a matrix will be denoted as $\|\mathbf{A}\|_F$ and we will define $\|\mathbf{A}\|_1 := \sum_{i,j} |\mathbf{A}_{ij}|$ where the i-jth element in matrix $\mathbf{A}$ is typically denoted as $(\mathbf{A})_{ij}$ or simply $\mathbf{A}_{ij}$.

Chapter 2

Precision Matrix Estimation

2.1 Background

The foundation of much of the precision matrix estimation literature is the gaussian negative log-likelihood. Consider the case where we observe n independent, identically distributed (iid) copies of the random variable X, where the ith observation $X_i \in \mathbb{R}^p$ is normally distributed with mean, μ, and variance, Ω^{-1}. That is, X_i follows a p-dimensional normal distribution which is typically denoted as $X_i \sim N_p\left(\mu, \Omega^{-1}\right)$. By definition, this multivariate formulation implies the probability distribution function, f, is of the form

$$f\left(X_i; \mu, \Omega\right) = (2\pi)^{-p/2} \left|\Omega\right|^{1/2} \exp\left[-\frac{1}{2}\left(X_i - \mu\right)' \Omega \left(X_i - \mu\right)\right]$$

Furthermore, because we assume that each observation is independent, the probability distribution function for all n observations $X_1, ..., X_n$ is equal to

$$\begin{aligned} f\left(X_1, ..., X_n; \mu, \Omega\right) &= \prod_{i=1}^{n} (2\pi)^{-p/2} \left|\Omega\right|^{1/2} \exp\left[-\frac{1}{2}\left(X_i - \mu\right)' \Omega \left(X_i - \mu\right)\right] \\ &= (2\pi)^{-np/2} \left|\Omega\right|^{n/2} \operatorname{etr}\left[-\frac{1}{2}\sum_{i=1}^{n}\left(X_i - \mu\right)\left(X_i - \mu\right)' \Omega\right] \end{aligned}$$

Therefore, the gaussian log-likelihood, l, for μ and Ω given $X = (X_1, .., X_n)$ can be written as

$$l(\mu, \Omega|X) = constant + \frac{n}{2}\log|\Omega| - tr\left[\frac{1}{2}\sum_{i=1}^{n}(X_i - \mu)(X_i - \mu)'\,\Omega\right]$$

The estimator for μ that maximizes the log-likelihood is $\hat{\mu}^{mle} = \bar{X} \equiv \sum_{i=1}^{n} X_i/n$, so that the partially maximized gaussian log-likelihood function for Ω is

$$\begin{aligned} l(\Omega|X) &= \frac{n}{2}\log|\Omega| - tr\left[\frac{1}{2}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(X_i - \bar{X}\right)'\Omega\right] \\ &= \frac{n}{2}\log|\Omega| - \frac{n}{2}tr\left(S\Omega\right) \end{aligned}$$

where $S = \sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(X_i - \bar{X}\right)'/n$ is the usual sample estimator for the population covariance matrix, Σ. In addition to μ, one could also derive the maximum likelihood estimator for Ω. By setting the gradient of the partially maximized log-likelihood equal to zero, one could show that

$$\begin{aligned} \hat{\Omega}^{mle} &= \arg\max_{\Omega \in S_+^p}\left\{\frac{n}{2}\log|\Omega| - \frac{n}{2}tr\left(S\Omega\right)\right\} \\ &= \arg\min_{\Omega \in S_+^p}\left\{tr\left(S\Omega\right) - \log|\Omega|\right\} \\ &= S^{-1} \end{aligned}$$

so that the MLE for Ω, when it exists, is $\hat{\Omega}^{mle} = S^{-1} = \left[\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(X_i - \bar{X}\right)'/n\right]^{-1}$. The reality, however, is that this object does *not* always exist. In settings where the number of observations is exceeded by the number of features in a sample, the sample covariance matrix is rank deficient and no longer invertible. For this reason, many papers in the last decade have proposed *shrinkage estimators* of the population precision matrix similar to the shrinkage estimators in regression settings. That is, instead of minimizing solely the negative log-likelihood function, researchers have proposed minimizing the gaussian log-likelihood *plus* a penalty term, P, where P is often a function of the precision matrix.

$$\hat{\Omega} = \arg\min_{\Omega \in S_+^p}\left\{tr\left(S\Omega\right) - \log|\Omega| + P\left(\Omega\right)\right\} \tag{2.1}$$

The penalties that have been proposed for precision matrix estimation are typically a variation of the ridge penalty $P(\Omega) = \lambda\|\Omega\|_F^2/2$ or the lasso penalty $P(\Omega) = \lambda\|\Omega\|_1$ (here λ is a tuning parameter). The authors, Yuan and Lin (2007), initially proposed the lasso-penalized gaussian log-likelihood defined as

$$\hat{\Omega} = \arg\min_{\Omega \in S_+^p} \left\{ tr(S\Omega) - \log|\Omega| + \lambda \sum_{i \neq j} |\Omega_{ij}| \right\} \tag{2.2}$$

so as not to penalize the diagonal elements of the precision matrix estimate. Other papers published on the lasso-penalized gaussian likelihood precision matrix estimator include Rothman et al. (2008) and Friedman et al. (2008). In addition, many efficient algorithms have been proposed to solve for $\hat{\Omega}$, however, the most popular method is the graphical lasso algorithm (glasso) introduced by Friedman et al. (2008). Their method utilizes an iterative block-wise coordinate descent algorithm that builds upon the coordinate descent algorithm used in lasso-penalized regression.

Non-lasso, non-convex penalties were considered in Lam and Fan (2009) and Fan et al. (2009) and other papers considered penalizations like the Frobenius norm (Rothman et al. (2014); Witten and Tibshirani (2009); Price et al. (2015)). In fact, the latter two papers show that the resulting minimizer can be solved in closed-form - which will be discussed later in this manuscript. However, the penalty explored through the remainder of this chapter is not a lasso penality nor a ridge penalty but, in fact, a convex combination of the two known as the *elastic-net* penalty:

$$P(\Omega) = \lambda \left[\frac{1-\alpha}{2} \|\Omega\|_F^2 + \alpha \|\Omega\|_1 \right]$$

with additional tuning parameter $0 \leq \alpha \leq 1$. Clearly, when $\alpha = 0$ this penalty reduces to a ridge penalty and when $\alpha = 1$ it reduces to a lasso penalty. Originally proposed in Zou and Hastie (2005) in the context of regression, this penalty has since been popularized and is used in the penalized regression R package `glmnet`. This penalty allows for additional flexibility but, despite this, no published work to our knowledge has explored the elastic-net penalty in the context of precision matrix estimation. We will show how to solve the following optimization problem in the next section using the ADMM algorithm.

$$\hat{\Omega} = \arg\min_{\Omega \in S_+^p} \left\{ tr(S\Omega) - \log|\Omega| + \lambda \left[\frac{1-\alpha}{2} |\Omega|_F^2 + \alpha \|\Omega\|_1 \right] \right\} \tag{2.3}$$

2.2 ADMM Algorithm

ADMM stands for alternating direction method of multipliers. The algorithm was largely popularized by Stephen Boyd and his fellow authors in the book *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers* (Boyd et al., 2011). As the authors state in the text, the "ADMM is an algorithm that is intended to blend the decomposability of dual ascent with the superior convergence properties of the method of multipliers." By closely following Boyd's descriptions and guidance in the published text, we will show in this section that the ADMM algorithm is particularly well-suited to solve the penalized log-likelihood optimization problem we are interested in here.

In general, the ADMM algorithm supposes that we want to solve an optimization problem of the form

$$\begin{aligned} &\text{minimize } f(x) + g(z) \\ &\text{subject to } Ax + Bz = c \end{aligned}$$

where we can assume here that $x \in \mathbb{R}^n, z \in \mathbb{R}^m, A \in \mathbb{R}^{p \times n}, B \in \mathbb{R}^{p \times m}$, $c \in \mathbb{R}^p$, and f and g are convex functions. In order to find the pair (x^*, z^*) that achieves the infimum, the ADMM algorithm uses an *augmented lagrangian*, L, which Boyd et al. (2011) define as

$$L_\rho(x, z, y) = f(x) + g(z) + y'(Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2$$

In this formulation, $y \in \mathbb{R}^p$ is called the lagrange multiplier and $\rho > 0$ is some scalar that acts as the step size for the algorithm. Note that any infimum under the augmented lagrangian is equivalent to the infimum of the traditional lagrangian since any feasible point (x, z) must satisfy the constraint $\rho \|Ax + Bz - c\|_2^2 / 2 = 0$. Boyd et al. (2011) show that using the ADMM algorithm the infimum will be approached under the following repeated iterations:

$$
\begin{aligned}
x^{k+1} &= \arg\min_{x\in\mathbb{R}^n} L_\rho(x, z^k, y^k) \\
z^{k+1} &= \arg\min_{z\in\mathbb{R}^m} L_\rho(x^{k+1}, z, y^k) \\
y^{k+1} &= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)
\end{aligned}
$$

where the superscript, k, denotes the number of iterations. Conveniently, this general algorithm can be coerced into a format useful in precision matrix estimation. Suppose we let f be equal to the non-penalized gaussian log-likelihood, g equal to the elastic-net penalty, $P(\Omega)$, and we use the constraint that $\Omega \in \mathbb{S}_+^p$ must be equal to some matrix $Z \in \mathbb{R}^{p\times p}$, then the augmented lagrangian in the context of precision matrix estimation is of the form

$$
L_\rho(\Omega, Z, \Lambda) = f(\Omega) + g(Z) + tr\left[\Lambda(\Omega - Z)\right] + \frac{\rho}{2}\left\|\Omega - Z\right\|_F^2
$$

where Λ takes the role of y as the lagrange multiplier. The ADMM algorithm now consists of the following repeated iterations:

$$
\begin{aligned}
\Omega^{k+1} &= \arg\min_{\Omega\in\mathbb{S}_+^p} \left\{ tr(S\Omega) - \log|\Omega| + tr\left[\Lambda^k(\Omega - Z^k)\right] + \frac{\rho}{2}\left\|\Omega - Z^k\right\|_F^2 \right\} \\
Z^{k+1} &= \arg\min_{Z\in\mathbb{S}^p} \left\{ \lambda\left[\frac{1-\alpha}{2}\left\|Z\right\|_F^2 + \alpha\left\|Z\right\|_1\right] + tr\left[\Lambda^k(\Omega^{k+1} - Z)\right] + \frac{\rho}{2}\left\|\Omega^{k+1} - Z\right\|_F^2 \right\} \\
\Lambda^{k+1} &= \Lambda^k + \rho(\Omega^{k+1} - Z^{k+1})
\end{aligned}
\tag{2.4}
$$

Furthermore, it turns out that each step in this algorithm can be solved efficiently in closed-form. The full details of each can be found in the appendix A.0.1 but the following theorem provides the simplified steps in the algorithm.

Theorem 2.1 (ADMM Algorithm for Elastic-Net Penalized Precision Matrix Estimation)**.** *Define the soft-thresholding function as soft(a, b) = sign$(a)(|a|-b)_+$ and S as the sample covariance matrix. Set $k = 0$ and initialize Z^0, Λ^0, and ρ. Repeat steps 1-3 until convergence:*

1. *Decompose $S + \Lambda^k - \rho Z^k = VQV'$ via spectral decomposition*[1]*. Then*

[1] Proof of (2.5) in section A.0.1.

$$\Omega^{k+1} = \frac{1}{2\rho} V \left[-Q + \left(Q^2 + 4\rho I_p \right)^{1/2} \right] V' \tag{2.5}$$

2. Elementwise soft-thresholding for all $i = 1, ..., p$ and $j = 1, ..., p$[2].

$$\begin{aligned} Z_{ij}^{k+1} &= \frac{1}{\lambda(1-\alpha)+\rho} sign\left(\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k}\right)\left(\left|\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k}\right| - \lambda\alpha\right)_+ \\ &= \frac{1}{\lambda(1-\alpha)+\rho} soft\left(\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k}, \lambda\alpha\right) \end{aligned} \tag{2.6}$$

3. Update Λ^{k+1}.

$$\Lambda^{k+1} = \Lambda^k + \rho\left(\Omega^{k+1} - Z^{k+1}\right)$$

2.2.1 Scaled-Form ADMM

Another popular, alternative form of the ADMM algorithm can be used when scaling the dual variable (Λ^k) which we will briefly mention in the context of precision matrix estimation here.

Define $R^k = \Omega - Z^k$ and $U^k = \Lambda^k/\rho$ then

$$\begin{aligned} tr\left[\Lambda^k\left(\Omega - Z^k\right)\right] + \frac{\rho}{2}\left\|\Omega - Z^k\right\|_F^2 &= tr\left[\Lambda^k R^k\right] + \frac{\rho}{2}\left\|R^k\right\|_F^2 \\ &= \frac{\rho}{2}\left\|R^k + \Lambda^k/\rho\right\|_F^2 - \frac{\rho}{2}\left\|\Lambda^k/\rho\right\|_F^2 \\ &= \frac{\rho}{2}\left\|R^k + U^k\right\|_F^2 - \frac{\rho}{2}\left\|U^k\right\|_F^2 \end{aligned}$$

Therefore, a scaled-form ADMM algorithm can now be written as

$$\begin{aligned} \Omega^{k+1} &= \arg\min_{\Omega\in\mathbb{S}_+^p}\left\{tr\left(S\Omega\right) - \log|\Omega| + \frac{\rho}{2}\left\|\Omega - Z^k + U^k\right\|_F^2\right\} \\ Z^{k+1} &= \arg\min_{Z\in\mathbb{S}^p}\left\{\lambda\left[\frac{1-\alpha}{2}\left\|Z\right\|_F^2 + \alpha\left\|Z\right\|_1\right] + \frac{\rho}{2}\left\|\Omega^{k+1} - Z + U^k\right\|_F^2\right\} \\ U^{k+1} &= U^k + \Omega^{k+1} - Z^{k+1} \end{aligned} \tag{2.7}$$

[2]Proof of (2.6) in section A.0.2.

Note that there are limitations to using this method. Because the dual variable is scaled by ρ (the step size), this form limits one to using a constant step size for all k steps if no further adjustments are made to U^k.

2.2.2 Stopping Criterion

There are three optimality conditions for the ADMM algorithm that we can use to inform the convergence and stopping criterion. These general conditions were outlined in Boyd et al. (2011) but here we cater them to precision matrix estimation. The first condition is the primal optimality condition $\Omega^{k+1} - Z^{k+1} = 0$ and the other two conditions are the dual conditions $0 \in \partial f\left(\Omega^{k+1}\right) + \Lambda^{k+1}$ and $0 \in \partial g\left(Z^{k+1}\right) - \Lambda^{k+1}$. The first dual optimality condition is a result of taking the sub-differential of the lagrangian (non-augmented) with respect to Ω^{k+1} and the second is a result of taking the sub-differential of the lagrangian with respect to Z^{k+1}. Note that in the first condition we must honor the symmetric constraint in Ω but the second condition does not require it.

If we define the left-hand side of the primal optimality condition as the *primal residual* $r^{k+1} = \Omega^{k+1} - Z^{k+1}$, then at convergence we must require that $r^{k+1} \approx 0$. Likewise, if we define the *dual residual* $s^{k+1} = \rho\left(Z^{k+1} - Z^k\right)$, we must also require that $s^{k+1} \approx 0$ for proper convergence. This dual residual is the direct result of the fact that Ω^{k+1} is the minimizer of the augmented lagragian[3] so that $0 \in \partial L_p\left(\Omega, Z^k, \Lambda^k\right)$ and consequently $0 \in \rho\left(Z^{k+1} - Z^k\right)$[4].

Combining these three optimality conditions, Boyd suggests a stopping criterion similar to $\epsilon^{pri} \leq \left\|r^{k+1}\right\|_F$ and $\epsilon^{dual} \leq \left\|s^{k+1}\right\|_F$ where $\epsilon^{rel} = \epsilon^{abs} = 10^{-3}$ and

$$\epsilon^{pri} = p\epsilon^{abs} + \epsilon^{rel} \max\left\{\left\|\Omega^{k+1}\right\|_F, \left\|Z^{k+1}\right\|_F\right\}$$
$$\epsilon^{dual} = p\epsilon^{abs} + \epsilon^{rel}\left\|\Lambda^{k+1}\right\|_F$$

2.3 Simulations

As a proof-of-concept that the elastic-net penalty in the context of precision matrix estimation can provide useful results and that the ADMM algorithm used in this process works, this section offers a short simulation. For the simulation, we generated data from multiple, unique precision matrices

[3] Proof in section A.0.3.

[4] Note that the second dual optimality condition $0 \in \partial g\left(Z^{k+1}\right) - \Lambda^{k+1}$ is always satisfied. More details can be found in section A.0.4.

with various oracle structures. For each data-generating procedure, the algorithm was run with a 5-fold cross validation to tune parameters λ and α. After 20 replications, the cross validation errors were totalled and the optimal tuning parameters were selected (results are in the top half of the figures). These results were then compared with the Kullback Leibler (KL) losses between the estimated matrices and the oracle matrices (results are in the bottom half of the figures).

The first figure shows the results when the data was generated from a multivariate normal distribution with mean equal to zero and a tri-diagonal oracle precision matrix. This oracle matrix was first generated as $\left(S_{ij}\right) = 0.7^{|i-j|}$ for $i, j = 1, ..., p$ and then inverted. The results show that because the oracle precision matrix is sparse, the algorithm correctly chooses a sparse solution with $\alpha = 1$ - indicating a lasso penalty.

The second figure shows the results when the data was generated from a multivariate normal distribution with mean equal to zero and a *dense* oracle precision matrix (non-sparse). Here, we randomly generated an orthogonal basis, set all eigen values equal to 1000, and then combined the matrices using QR decomposition. Interestingly, we find that the optimal α in this case is 0.6 which closely matches the optimal result based on the KL loss. This shows that there are cases where an elastic-net penalty can provide useful results and that using only a lasso penalty may unnecessarily restrict our penalized estimation.

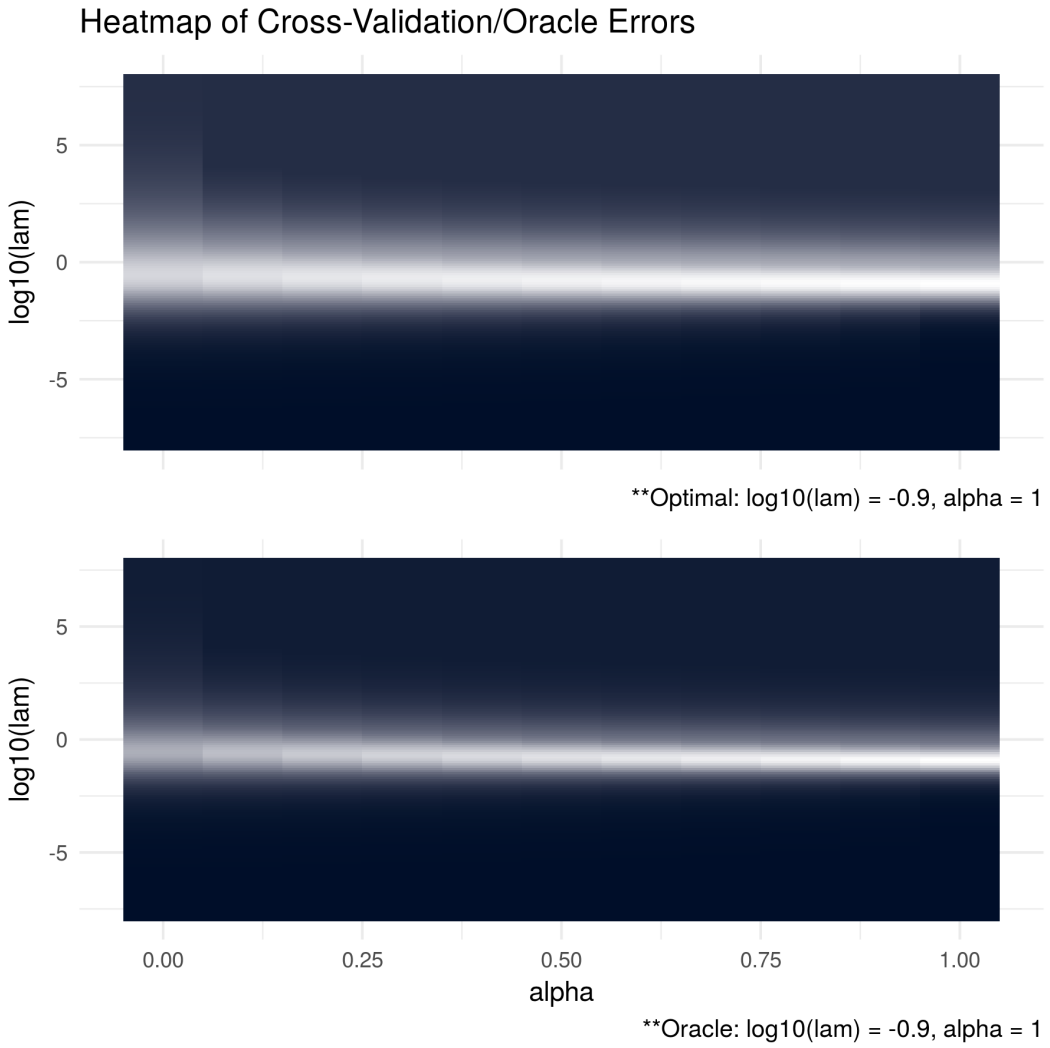

Figure 2.1: The oracle precision matrices were tri-diagonal with dimension p = 100 and the data was generated with a sample size of n = 50. The cross validation errors are in the top figure and the KL losses between the estimated matrices and the oracle matrices are shown in the bottom figure. The optimal tuning parameter pair for each heatmap was found to be log10(lam) = -0.9 and alpha = 1. Note that brighter areas signify smaller losses.

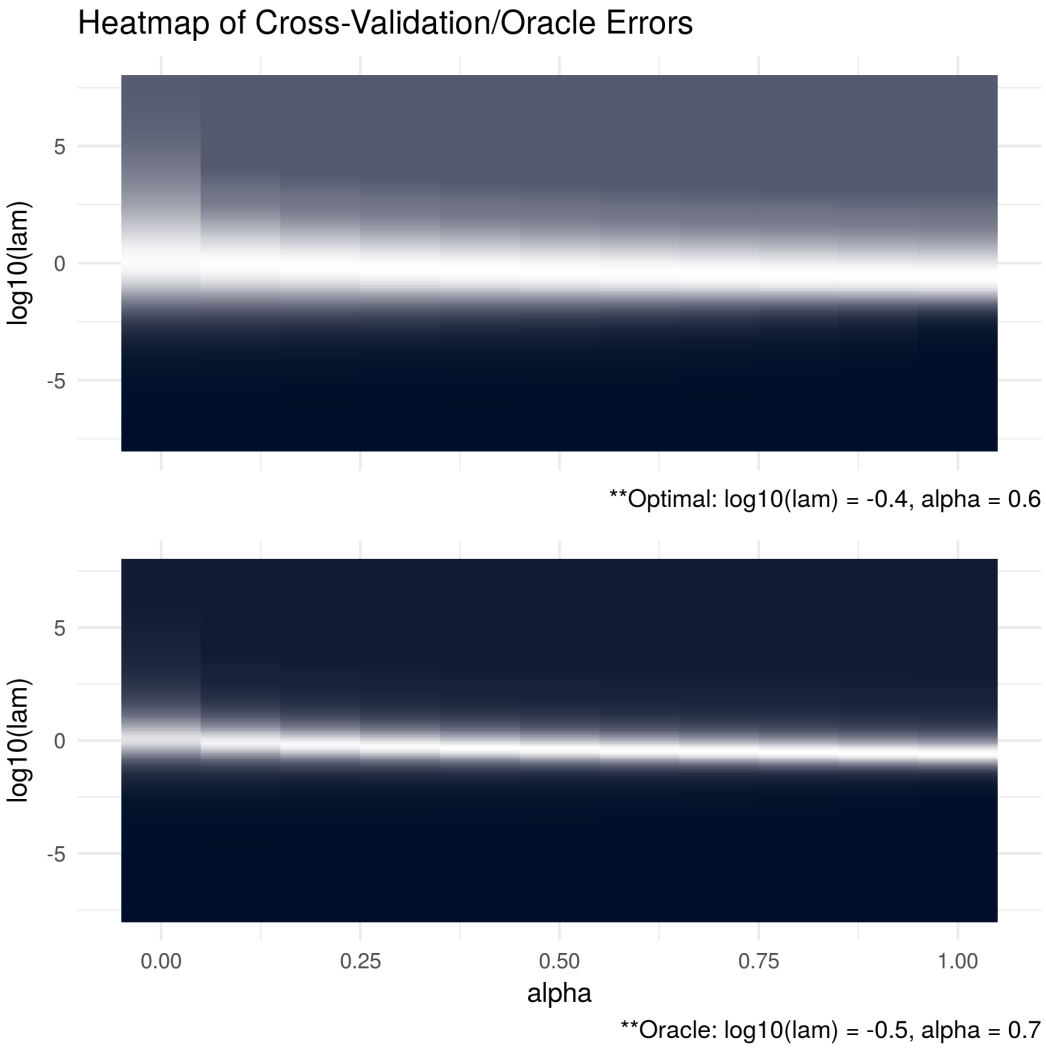

Figure 2.2: The oracle precision matrices were dense with dimension p = 100 and the data was generated with a sample size of n = 50. The cross validation errors are in the top figure and the KL losses between the estimated matrices and the oracle matrices are shown in the bottom figure. The optimal tuning parameter pair for the cross validation errors was found to be log10(lam) = -0.4 and alpha = 0.6 and log10(lam) = -0.5 and alpha = 0.7 for the KL losses. Note that brighter areas signify smaller losses.

Chapter 3

SCPME

In their 2017 paper, titled *Shrinking Characteristics of Precision Matrix Estimators*, Aaron Molstad, Ph.D. and Professor Adam Rothman outline a framework to estimate *characteristics* of precision matrices. This concept, inspired by others like Cai and Liu (2011), Fan et al. (2012), and Mai et al. (2012), exploits the fact that in many predictive models estimation of the precision matrix is only necessary through its product with another feature, such as a mean vector. The example they offer in Molstad and Rothman (2017) is in the context of Fisher's linear discriminant analysis model. If a response vector Y is categorical such that Y can take values in $\{1, ..., J\}$, then the linear discrimnant analysis model assumes that the design matrix X conditional on the response vector Y is normally distributed:

$$X|Y = j \sim N_p\left(\mu_j, \Omega^{-1}\right) \tag{3.1}$$

for each $j = 1, ..., J$. One can see from this formulation that clearly an estimation of both μ and Ω are required for this model. However, they note that if prediction is the primary concern, then for a given observation X_i only the *characteristic* $\Omega\left(\mu_l - \mu_m\right)$ is needed to discern between response categories l and m. In other words, prediction only requires the characteristic $\Omega\left(\mu_l - \mu_m\right)$ for each $l, m \in \{1, ..., J\}$ and does *not* require estimation of the full precision matrix Ω. Cai and Liu (2011) were among the first authors to propose estimating this characteristic directly but the interesting facet that distinguishes Molstad and Rothman's approach is that their framework simultaneously fits the model in (3.1) and performs variable selection.

The general framework has applications outside of linear discriminant analysis and we will be exploring a regression application in later sections, but first we will outline their approach. The penalty proposed by Molstad and Rothman (2017) is of the form

$$P(\Omega) = \lambda \|A\Omega B - C\|_1 \tag{3.2}$$

where $A \in \mathbb{R}^{m\times p}, B \in \mathbb{R}^{p\times q},$ and $C \in \mathbb{R}^{m\times q}$ are matrices that are assumed to be known and specified so that solving the full penalized gaussian negative log-likelihood for Ω results in solving

$$\hat{\Omega} = \arg\min_{\Omega\in S_+^p} \left\{ tr(S\Omega) - \log|\Omega| + \lambda \|A\Omega B - C\|_1 \right\} \tag{3.3}$$

A similar estimator was proposed by Dalal and Rajaratnam (2017) when $C = 0$ but here we do not require it. This form of penalty is particularly useful because it is extremely general. Note that by letting matrices $A = I_p, B = I_p,$ and $C = 0$, this penalty reduces to a lasso penalty - but clearly this form allows for much more creative penalties and $A, B,$ and C can be constructed so that we penalize the sum, absolute value of *many* characteristics of the precision matrix Ω. We will explore how to solve for $\hat{\Omega}$ in (3.3) in the next section.

3.1 Augmented ADMM Algorithm

Solving for $\hat{\Omega}$ in (3.3) uses what we are going to call an *augmented ADMM algorithm*. Molstad and Rothman do not offer a name for this specific algorithm but it leverages the majorize-minimize principle in one of the steps in the algorithm - augmenting the original ADMM algorithm discussed in the previous chapter. Within the context of the proposed penalty, the original ADMM algorithm for precision matrix estimation would consist of iterating over the following three steps:

$$\Omega^{k+1} = \arg\min_{\Omega\in\mathbb{S}_+^p} L_\rho(\Omega, Z^k, \Lambda^k) \tag{3.4}$$

$$Z^{k+1} = \arg\min_{Z\in\mathbb{R}^{n\times r}} L_\rho(\Omega^{k+1}, Z, \Lambda^k) \tag{3.5}$$

$$\Lambda^{k+1} = \Lambda^k + \rho\left(A\Omega^{k+1}B - Z^{k+1} - C\right) \tag{3.6}$$

where L, the augmented lagrangian, is defined as

$$L_\rho(\Omega, Z, \Lambda) = f(\Omega) + g(Z) + tr\left[\Lambda'\left(A\Omega B - Z - C\right)\right] + \frac{\rho}{2}\left\|A\Omega B - Z - C\right\|_F^2 \tag{3.7}$$

Similar to the previous chapter, $f(\Omega) = tr(S\Omega) - \log|\Omega|$ and $g(Z) = \lambda\|Z\|_1$. In fact, the details of the algorithm thus far are identical to the previous approach except that we are replacing $\Omega - Z$ with $A\Omega B - Z - C$ in the augmented lagrangian and the dual update Λ^{k+1}.

Instead of solving the first step directly, the authors propose an alternative, approximating objective function, which we will denote as $\tilde{L}$, that is based on the majorize-minimize principle[1]. The purpose of this approximating function is the desire to solve the first step of the algorithm in closed-form. The ADMM algorithm with this modification based on majorize-minimize principle is also found in Lange (2016) but here we define the approximating function as

$$\begin{aligned}\tilde{L}_\rho(\Omega, Z^k, \Lambda^k) = f(\Omega) + g(Z^k) + tr\left[(\Lambda^k)'(A\Omega B - Z^k - C)\right] + \frac{\rho}{2}\left\|A\Omega B - Z^k - C\right\|_F^2 \\ + \frac{\rho}{2} vec\left(\Omega - \Omega^k\right)' Q\left(\Omega - \Omega^k\right)\end{aligned} \tag{3.8}$$

where $Q = \tau I_p - (A'A \otimes BB')$ and τ is chosen such that Q is positive definite. Note that if Q is positive definite, then $L_\rho(\cdot) \leq \tilde{L}(\cdot)$ for all Ω and $\tilde{L}$ is a majorizing function[2]. The *augmented ADMM* algorithm developed by Molstad and Rothman, which now includes the majorize-minimize principle, consists of the following repeated iterations:

$$\begin{aligned}\Omega^{k+1} &= \arg\min_{\Omega \in \mathbb{S}_+^p}\left\{tr\left[\left(S + G^k\right)\Omega\right] - \log|\Omega| + \frac{\rho\tau}{2}\left\|\Omega - \Omega^k\right\|_F^2\right\} \\ Z^{k+1} &= \arg\min_{Z \in \mathbb{R}^{n\times r}}\left\{\lambda\|Z\|_1 + tr\left[(\Lambda^k)'(A\Omega B - Z^k - C)\right] + \frac{\rho}{2}\left\|A\Omega B - Z^k - C\right\|_F^2\right\} \\ \Lambda^{k+1} &= \Lambda^k + \rho\left(A\Omega^{k+1}B - Z^{k+1} - C\right)\end{aligned}$$

where $G^k = \rho A'\left(A\Omega^k B - Z^k - C + \rho^{-1}\Lambda^k\right)B'$. Each step in this algorithm can now conveniently be solved in closed-form and the full details of each can be found in the appendix A.0.6. The

[1] Further explanation of the majorizing function (3.8) in section A.0.5

[2] If Q is positive definite, then $vec\left(\Omega - \Omega^k\right)' \rho Q\left(\Omega - \Omega^k\right)/2 > 0$ since $\rho > 0$ and $vec\left(\Omega - \Omega^k\right)$ is always nonzero whenever $\Omega \neq \Omega^k$.

following theorem provides the simplified steps in the algorithm.

Theorem 3.1 (Augmented ADMM Algorithm for Shrinking Characteristics of Precision Matrix Estimators.)**.** *Define the soft-thresholding function as* $soft(a,b) = sign(a)(|a|-b)_+$ *and* S *as the sample covariance matrix. Set* $k=0$ *and initialize* Z^0, Λ^0, Ω^0*, and* ρ *and repeat steps 1-5 until convergence.*

1. *Compute* G^k*.*

$$G^k = \rho A' \left(A\Omega^k B - Z^k - C + \rho^{-1}\Lambda^k\right) B'$$

2. *Via spectral decomposition, decompose*

$$S + \left(G^k + (G^k)'\right)/2 - \rho\tau\Omega^k = VQV'$$

3. *Update* Ω^{k+1}*.*[3]

$$\Omega^{k+1} = V\left(-Q + \left(Q^2 + 4\rho\tau I_p\right)^{1/2}\right) V'/(2\rho\tau) \tag{3.9}$$

4. *Update* Z^{k+1} *with element-wise soft-thresholding for the resulting matrix.*[4]

$$Z^{k+1} = soft\left(A\Omega^{k+1}B - C + \rho^{-1}\Lambda^k, \rho^{-1}\lambda\right) \tag{3.10}$$

5. *Update* Λ^{k+1}*.*

$$\Lambda^{k+1} = \rho\left(A\Omega^{k+1}B - Z^{k+1} - C\right)$$

[3]Proof of (3.9) in section A.0.6
[4]Proof of (3.10) in section A.0.7

3.1.1 Stopping Criterion

A possibe stopping criterion for this framework is one derived from similar optimality conditions used in the previous chapter in section 2.2.2. The primal optimality condition here is that $A\Omega^{k+1}B - Z^{k+1} - C = 0$ and the two dual optimality conditions are $0 \in \partial f\left(\Omega^{k+1}\right) + \left(B(\Lambda^{k+1})'A + A'\Lambda^{k+1}B'\right)/2$ and $0 \in \partial g\left(Z^{k+1}\right) - \Lambda^{k+1}$. Similarly, we will define the left-hand side of the primal optimality condition as the primal residual $r^{k+1} = A\Omega^{k+1}B - Z^{k+1} - C$ and the dual residual[5] as

$$s^{k+1} = \frac{\rho}{2}\left(B(Z^{k+1} - Z^k)'A + A'(Z^{k+1} - Z^k)B'\right)$$

For proper convergence, we will require that both residuals are approximately equal to zero. Similar to the stopping criterion discussed previously, one possibility is to set $\epsilon^{rel} = \epsilon^{abs} = 10^{-3}$ and stop the algorithm when $\epsilon^{pri} \leq \left\|r^{k+1}\right\|_F$ and $\epsilon^{dual} \leq \left\|s^{k+1}\right\|_F$ where

$$\begin{aligned}
\epsilon^{pri} &= \sqrt{nr}\epsilon^{abs} + \epsilon^{rel}\max\left\{\left\|A\Omega^{k+1}B\right\|_F, \left\|Z^{k+1}\right\|_F, \|C\|_F\right\} \\
\epsilon^{dual} &= p\epsilon^{abs} + \epsilon^{rel}\left\|\left(B(\Lambda^{k+1})'A + A'\Lambda^{k+1}B'\right)/2\right\|_F
\end{aligned}$$

3.2 Regression Illustration

One of the research directions mentioned in Molstad and Rothman (2017) that was not further explored was the application of the SCPME framework to regression. Utilizing the fact that the population regression coefficient matrix $\beta \equiv \Omega_x\Sigma_{xy}$ for predictors, X, and the responses, Y, they point out that their framework could allow for the simultaneous estimation of β and Ω_x. Like Witten and Tibshirani (2009), this approach would estimate the forward regression coefficient matrix while using shrinkage estimators for the marginal population precision matrix for the predictors. For example, recall that the general optimization problem outlined in the SCPME framework is to estimate $\hat{\Omega}$ such that

$$\hat{\Omega} = \arg\min_{\Omega \in \mathbb{S}_+^p}\left\{tr(S\Omega) - \log|\Omega| + \lambda\left\|A\Omega B - C\right\|_1\right\}$$

[5] Proof of (3.11) in section A.0.8.

If the user specifies that $A = I_p, B = \Sigma_{xy}, C = 0,$ and Ω_x is the precision matrix for the predictors, then the optimization problem of interest is now

$$\hat{\Omega}_x = \arg\min_{\Omega_x \in \mathbb{S}_+^p} \left\{ tr(S_x \Omega_x) - \log|\Omega_x| + \lambda \left\| \Omega_x \Sigma_{xy} \right\|_1 \right\}$$

Specifically, this optimization problem has the effect of deriving an estimate of Ω_x while assuming sparsity in the forward regression coefficient β. Of course, in practice we do not know the true covariance matrix Σ_{xy} but we might consider using the sample estimate $\hat{\Sigma}_{xy} = \sum_{i=1}^{n} \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)' / n$ in place of Σ_{xy}. We could then use our estimator, $\hat{\Omega}_x$, to construct the estimated forward regression coefficient matrix $\hat{\beta} = \hat{\Omega}_x \hat{\Sigma}_{xy}$. Estimators such as these are truly novel an can conveniently be estimated by the SCPME framework. Another such estimator that is a product of this new framework is one where we construct A and C similarly but take $B = \left[\Sigma_{xy}, I_p\right]$ so that the identity matrix is appended to the cross-covariance matrix of X and Y. In this case, not only are we assuming that β is sparse, but we are also assuming sparsity in Ω.

$$P_\lambda(\Omega) = \lambda \left\| A\Omega B - C \right\|_1 = \lambda \left\| \Omega \left[\Sigma_{xy}, I_p\right] \right\|_1 = \lambda \left\| \beta \right\|_1 + \lambda \left\| \Omega \right\|_1$$

Like before, we could use $\hat{\Sigma}_{xy}$ as a replacement and take our resulting estimator, $\hat{\Omega}_x$, to construct the estimated forward regression coefficient matrix $\hat{\beta} = \hat{\Omega}_x \hat{\Sigma}_{xy}$. The embedded assumptions here are that not all predictors in X are useful in predicting the response, Y, *and* that a number of the predictors are conditionally independent of one another. These are assumptions that are quite reasonable in practice and in the next section we offer a short simulation comparing these new estimators to related and competing prediction methods.

3.3 Simulations

In this simulation, we compare, under various data realizations, the performance of several competing regression prediction methods including the new so-called SCPME regression estimators discussed in the previous section. In total, we consider the following:

- `OLS` = ordinary least squares estimatior. In high dimensional settings ($p >> n$), the Moore-

Penrose estimator is used as a replacement.

- `ridge` = ridge regression estimator.
- `lasso` = lasso regression estimator.
- `oracleB` = oracle estimator for β.
- `oracleO` = regression estimator with oracle Ω_x^* (let $\beta = \Omega_x^* \hat{\Sigma}_{xy}$).
- `oracleS` = regression estimator with oracle Σ_{xy}^* (let $\beta = \hat{\Omega}_x \Sigma_{xy}^*$) and Ω_x is estimated using an elastic-net penalty.
- `shrinkB` = SCPME regression estimator with penalty $\lambda \left\| \Omega_x \hat{\Sigma}_{xy} \right\|_1$ so that $\beta = \hat{\Omega}_x \hat{\Sigma}_{xy}$.
- `shrinkBS` = SCPME regression estimator with oracle Σ_{xy}^* so that the penalty is $\lambda \left\| \Omega_x \Sigma_{xy}^* \right\|_1$ and we let $\beta = \hat{\Omega}_x \Sigma_{xy}^*$.
- `shrinkBO` = SCPME regression estimator with penalty $\lambda \left\| \Omega_x [\hat{\Sigma}_{xy}, I_p] \right\|_1$ so that $\beta = \hat{\Omega}_x \hat{\Sigma}_{xy}$.
- `glasso` = graphical lasso estimator with penalty $\lambda \left\| \Omega_x \right\|_1$ so that $\beta = \hat{\Omega}_x \hat{\Sigma}_{xy}$.

For each estimator, if selection of a tuning parameter is required, the tuning parameter was chosen so that the mean squared prediction error (MSPE) was minimized over 3-fold cross validation.

The data generating procedure for the simulations is the following. The oracle regression coefficient matrix β^* was constructed so that $\beta^* = \mathbb{B} \circ \mathbb{V}$ where $vec(\mathbb{B}) \sim N_{pr}\left(0, I_p \otimes I_r/\sqrt{p}\right)$ and $\mathbb{V} \in \mathbb{R}^{p \times r}$ is a matrix containing p times r random bernoulli draws with 50% probability being equal to one. The covariance matrices $\Sigma_{y|x}$ and Σ_x were constructed so that $\left(\Sigma_{y|x}\right)_{ij} = 0.7^{|i-j|}$ and $\left(\Sigma_x\right)_{ij} = 0.7^{|i-j|}$, respectively. This ensures that their corresponding precision matrices will be tridiagonal and sparse. Then for 100 independent, identically distributed samples, we had $X_i \sim N_p\left(0, \Sigma_x\right)$ and $E_i \sim N_r\left(0, \Sigma_{y|x}\right)$ so that $\mathbb{Y} = \mathbb{X}\beta + \mathbb{E}$ where $\mathbb{X} \in \mathbb{R}^{n \times p}$ and $\mathbb{Y} \in \mathbb{R}^{n \times r}$ are the matrices with stacked rows X_i and E_i, respectively, for $i = 1, ..., n$ and the script notation denotes the fact that the columns have been centered so as to remove the intercept in our prediction models. A sample of an additional 1000 observations was generated similarly for the testing set. Each prediction method was evaluated on both model error and mean squared prediction error.

Figure 3.1 displays the model error for each method by dimension of the design matrix. Here we took the sample size equal to $n = 100$ and the response matrix dimension $r = 10$ with each data

generating procedure replicated a total of 20 times. Note `OLS` and `oracleS` are not shown due to extremely poor performance. We find that in high dimensional settings, `shrinkBO` and `shrinkB` perform increasingly well relative to the others as the predictor dimension increases. `shrinkBO` performed the best. Interestingly, when $n > p$ `shrinkBO` is still one of the best-performing estimators - though worse than both `lasso` and `ridge` - but the performance of `shrinkB` decreases drastically (plot not shown).

In the high dimension setting, the oracle estimators `oracleO` and `oracleS` performed worse or comparable to the `OLS` estimator. The poor performance of `oracleS` is likely due to the fact that the sample estimate of Ω_x is not identifiable when $p > n$.

The following table shows the average model error for each estimator when $n = 100$, $p = 150$, and $r = 10$.

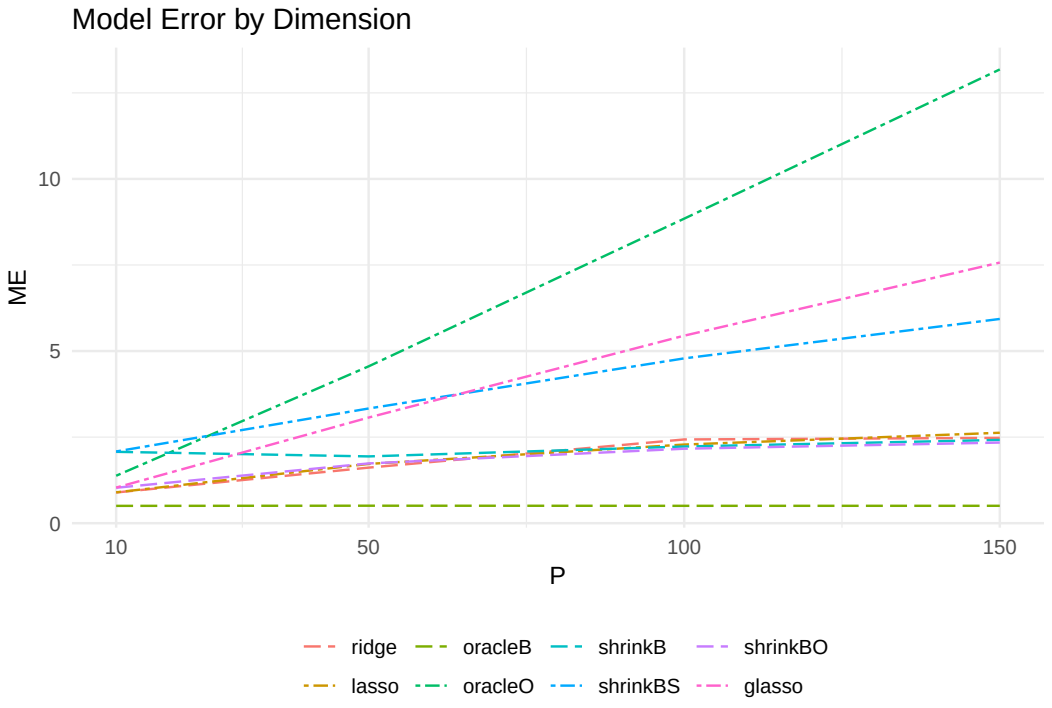

Figure 3.1: The oracle precision matrices were tri-diagonal with variable dimension (p) and the data was generated with sample size n = 100 and response vector dimension r = 10. The model errors (ME) for each estimator with variable dimension of the design matrix are plotted. shrinkBO and shrinkB were the two best-performing estimators closely follow by the ridge and lasso regression estimators.

Table 3.1: Average model error for dimension p = 150.

Model	Error	SD
oracleB	0.5056	0.5122
shrinkBO	2.341	1.014
shrinkB	2.419	1.083
ridge	2.479	1.128
lasso	2.627	1.27
shrinkBS	5.932	4.001
glasso	7.57	5.412
oracleO	13.18	10.11
OLS	14.41	11.24
oracleS	52.07	44.62

3.3.1 Regression Simulations with Covariance Shrinkage

This final simulation was explored due to the fact that each of the SCPME regression estimators require an estimation of the cross-covariance matrix. In the previous simulations, we naively used the maximum likelihood estimate. However, because the data-generating procedure constructs settings that are inherently sparse, we were interested to determine if additional shrinkage of the maximum likelihood estimate for the *cross-covariance* matrix would also prove beneficial. For instance, in a number of settings, we were finding that the estimator `oracleO` was performing worse than `oracleS`. This perhaps suggests that estimating the covariance matrix Σ_{xy} well is *more* important than estimating Ω well. This simulation explores that theory a bit further.

The data-generating procedure for this simulation is similar to previous one but here we take $n = 100$, $p = 200$, $r = 10$, and, in addition, we multiply the population cross-covariance estimator by a factor of k where $k \in (0.1, 0.2, ..., 0.9, 1)$. Figure 3.2 plots the MSPE for `shrinkB` for each tuning parameter, λ, and constant, k, pair. Figure 3.3 plots the MSPE similarly for `shrinkBO`.

Interestingly, we find that in this high dimension setting, it does appear that shrinking the sample covariance matrix by a constant factor helps the overall prediction performance of the SCPME estimators. This is indicated by the fact that the optimal constant, k, hovers between 0.3 and 0.6

for each of the estimators. We also performed a simulation in low dimensions (not shown) but we did not see the same benefit.

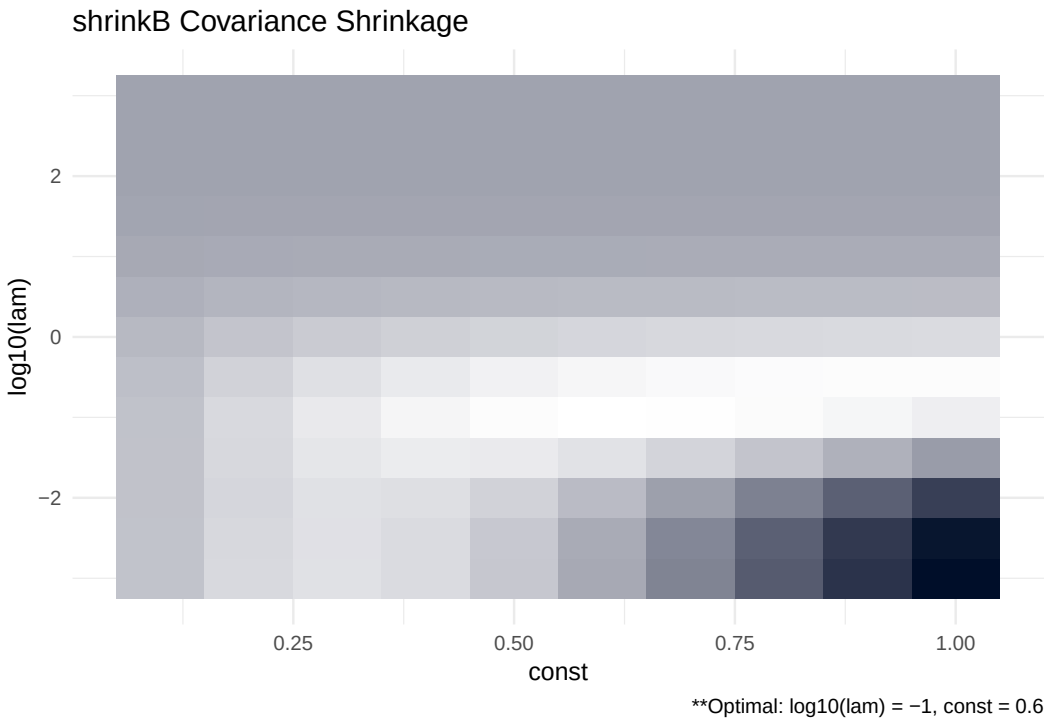

Figure 3.2: The oracle precision matrices were tri-diagonal with dimension p = 200 and the data was generated with a sample size n = 100 and response vector dimension r = 10. The cross validation MSPE are plotted for each lambda and constant tuning parameter pair. The optimal tuning parameter pair was found to be log10(lam) = -1 and const = 0.6. Note that brighter areas signify smaller losses.

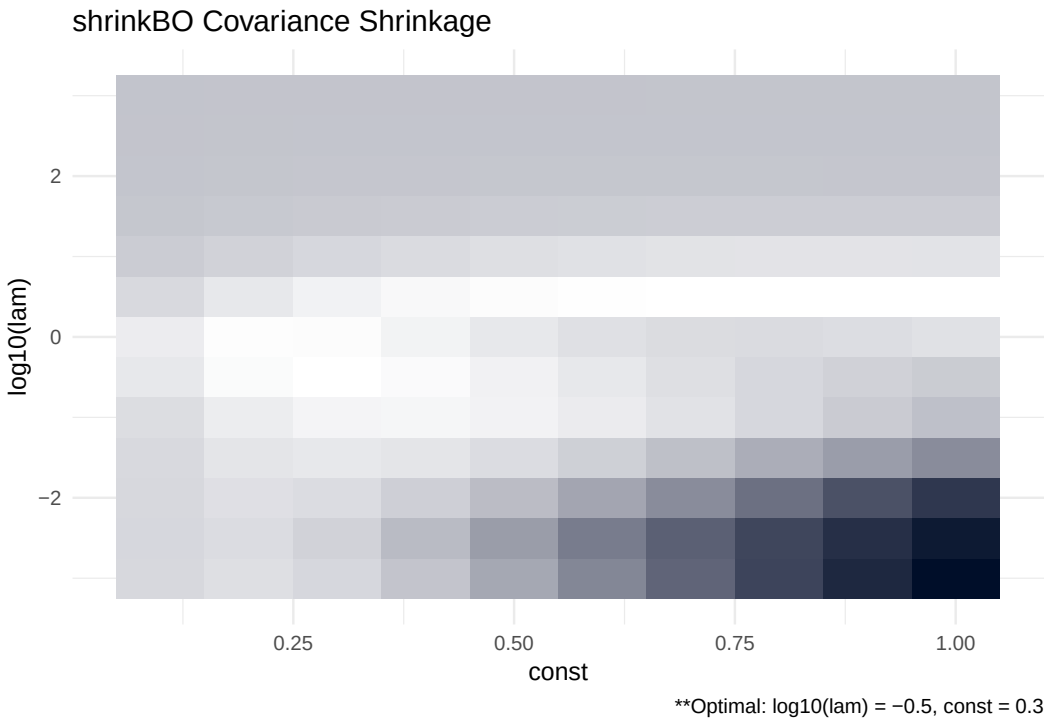

Figure 3.3: The oracle precision matrices were tri-diagonal with dimension p = 200 and the data was generated with a sample size n = 100 and response vector dimension r = 10. The cross validation MSPE are plotted for each lambda and constant tuning parameter pair. The optimal tuning parameter pair was found to be log10(lam) = -0.5 and const = 0.3. Note that brighter areas signify smaller losses.

3.4 Discussion

Apart from the two SCPME estimators discussed in the previous section, the generality of the SCPME framework allows for many previously unconceptualized precision matrix estimators to be explored. One estimator that was concieved of in our work but deserves further attention in future research is of the form

$$\hat{\Omega}_x = \arg\min_{\Omega_x \in \mathbb{S}_+^p} \left\{ tr(S_x \Omega_x) - \log|\Omega_x| + \frac{\lambda}{2} \left\| \mathbb{X} \Omega_x \Sigma_{xy} - \mathbb{Y} \right\|_F^2 \right\}$$

For clarity in regards to the SCPME framework, $A = \mathbb{X}, B = \Sigma_{xy},$ and $C = \mathbb{Y}$. As before, $\mathbb{X}$ is the $n \times p$ matrix with rows $X_i \in \mathbb{R}^p$ for $i = 1, ..., n$ and the script notation denotes that the matrix has been column-centered. The matrix $\mathbb{Y}$ is a similar representation for the observed responses $Y_i \in \mathbb{R}^r$. Also note that here we are using the Frobenius norm instead of the matrix l_1-norm but this replacement requires only a slight modification of the augmented ADMM algorithm for optimization - details of which will be presented later.

This optimization problem estimates a precision matrix that balances minimizing the gaussian negative log-likelihood for Ω_x with minimizing the squared prediction error for the forward regression model. In other words, the objective function aims to penalize estimates, determined by λ, that too heavily favor maximizing the marginal likelihood for X over the predictive performance of the conditional model Y given X. This estimator also reveals an interesting connection to the joint log-likelihood of the two random variables. To see this, let us suppose that we have n independent copies of the random pair (Y_i, X_i) and we assume a linear relationship such that

$$Y_i = \mu_y + \beta' (X_i - \mu_x) + E_i \tag{3.11}$$

where $E_i \sim N_r\left(0, \Omega_{y|x}^{-1}\right)$ and $X_i \sim N_p\left(\mu_x, \Omega_x^{-1}\right)$. This implies that the conditional distribution of $Y_i|X_i$ is of the form

$$Y_i|X_i \sim N_r\left(\mu_y + \beta' (X_i - \mu_x), \Omega_{y|x}^{-1}\right) \tag{3.12}$$

We can use this conditional distribution along with the marginal distribution of X to derive the joint log-likelihood of X and Y. Recall that $\beta \equiv \Omega_x \Sigma_{xy}$ and S_x is the marginal sample covariance

matrix of X. Without loss of generality, we will assume here that $\mu_x = \mu_y = 0$.

$$
\begin{aligned}
l\left(\Omega_{y|x}, \Omega_x, \Sigma_{xy} | Y, X\right) &= constant + \frac{n}{2}\log\left|\Omega_{y|x}\right| - \frac{1}{2}\sum_{i=1}^{n} tr\left[\left(Y_i - \Sigma_{xy}'\Omega_x X_i\right)\left(Y_i - \Sigma_{xy}'\Omega_x X_i\right)'\Omega_{y|x}\right] \\
&\quad + \frac{n}{2}\log|\Omega| - \frac{1}{2}\sum_{i=1}^{n} tr\left(X_i X_i'\Omega_x\right) \\
&= constant + \frac{n}{2}\log\left|\Omega_{y|x}\right| - \frac{1}{2} tr\left[\left(\mathbb{Y} - \mathbb{X}\Omega_x\Sigma_{xy}\right)'\left(\mathbb{Y} - \mathbb{X}\Omega_x\Sigma_{y|x}\right)\Omega_{y|x}\right] \\
&\quad + \frac{n}{2}\log|\Omega_x| - \frac{n}{2} tr\left(S_x\Omega_x\right)
\end{aligned}
$$

Optimizing this joint log-likelihood with respect to Ω_x reveals strong similiarities to the estimator that was derived from the SCPME framework:

$$
\begin{aligned}
\hat{\Omega}_x &= \arg\min_{\Omega_x \in \mathbb{S}_+^p} \left\{ \frac{1}{2} tr\left[\left(\mathbb{Y} - \mathbb{X}\Omega_x\Sigma_{xy}\right)'\left(\mathbb{Y} - \mathbb{X}\Omega_x\Sigma_{xy}\right)\Omega_{y|x}\right] + \frac{n}{2} tr\left(S_x\Omega_x\right) - \frac{n}{2}\log|\Omega_x| \right\} \\
&= \arg\min_{\Omega_x \in \mathbb{S}_+^p} \left\{ tr\left(S_x\Omega_x\right) - \log|\Omega_x| + \frac{1}{n}\left\|\left(\mathbb{X}\Omega_x\Sigma_{xy} - \mathbb{Y}\right)\Omega_{y|x}^{1/2}\right\|_F^2 \right\}
\end{aligned}
$$

If it is the case that, given X, each of the r responses are pairwise independent with equal variance so that $\Omega_{y|x}^{1/2} = \sigma_{y|x} I_r$ and we let $\lambda = 2\sigma_{y|x}^2/n$, then we have that

$$
\hat{\Omega}_x = \arg\min_{\Omega_x \in \mathbb{S}_+^p} \left\{ tr\left(S_x\Omega_x\right) - \log|\Omega_x| + \frac{\lambda}{2}\left\|\mathbb{X}\Omega_x\Sigma_{xy} - \mathbb{Y}\right\|_F^2 \right\}
$$

This is exactly the estimator conceived of previously. Of course, throughout this derivation there were several assumptions that were made and in practice we do not know the true values of neither $\Omega_{y|x}$ nor Σ_{xy}. However, we can see that this estimator is solving a very similar problem to that of optimizing the joint log-likelihood with respect to Ω_x. We think this estimator and related ones deserve attention in future work.

The algorithm for solving this optimization problem is below. Note that no closed-form solution exists and so we must resort to some iterative algorithm similar to the SCPME augmented ADMM algorithm[6]. The only difference here is that in step four we no longer require elementwise soft-thresholding.

[6] Proof in section A.0.9.

Theorem 3.2 (Modified Augmented ADMM Algorithm for Shrinking Characteristics of Precision Matrix Estimators with Frobenius Norm)**.** *Set* $k = 0$ *and initialize* Z^0, Λ^0, Ω^0*, and* ρ *and repeat steps 1-5 until convergence.*

1. *Compute* $G^k = \rho A' \left(A\Omega^k B - Z^k - C + \rho^{-1}\Lambda^k\right) B'$
2. *Decompose* $S + \left(G^k + (G^k)'\right)/2 - \rho\tau\Omega^k = VQV'$ *(via the spectral decomposition).*
3. *Set* $\Omega^{k+1} = V\left(-Q + \left(Q^2 + 4\rho\tau I_p\right)^{1/2}\right) V'/(2\rho\tau)$
4. *Set* $Z^{k+1} = \left[\rho\left(A\Omega^{k+1}B - C\right) + \Lambda^k\right]/(\lambda + \rho)$
5. *Set* $\Lambda^{k+1} = \rho\left(A\Omega^{k+1}B - Z^{k+1} - C\right)$

Appendix A

Appendix

A.0.1 Proof of (2.5)

Witten and Tibshirani (2009) and Price et al. (2015) showed that the following optimization problem can be solved in closed-form. We outline their steps here.

$$\Omega^{k+1} = \arg\min_{\Omega \in \mathbb{S}_+^p} \left\{ tr\left(S\Omega\right) - \log|\Omega| + tr\left[\Lambda^k\left(\Omega - Z^k\right)\right] + \frac{\rho}{2}\left\|\Omega - Z^k\right\|_F^2 \right\}$$

First start by taking the gradient with respect to Ω:

$$\begin{aligned} &\nabla_\Omega \left\{ tr\left(S\Omega\right) - \log|\Omega| + tr\left[\Lambda^k\left(\Omega - Z^k\right)\right] + \frac{\rho}{2}\left\|\Omega - Z^k\right\|_F^2 \right\} \\ &\quad = S - \Omega^{-1} + \Lambda^k + \rho\left(\Omega - Z^k\right) \end{aligned}$$

Note that because all of the variables are symmetric, we can ignore the symmetric constraint when deriving the gradient. Next set the gradient equal to zero and decompose $\Omega^{k+1} = VDV'$ using spectral decomposition so that D is a diagonal matrix with diagonal elements equal to the eigen values of Ω^{k+1} and V is the matrix with corresponding eigen vectors as columns so that

$$S + \Lambda^k - \rho Z^k = (\Omega^{k+1})^{-1} - \rho\Omega^{k+1} = VD^{-1}V' - \rho VDV' = V\left(D^{-1} - \rho D\right)V'$$

This equivalence implies that

$$\phi_j\left(S+\Lambda^k-\rho Z^k\right)=\frac{1}{\phi_j(\Omega^{k+1})}-\rho\phi_j(\Omega^{k+1})$$

where $\phi_j(\cdot)$ is the jth eigen value.

$$\begin{aligned}
&\Rightarrow \rho\phi_j^2(\Omega^{k+1})+\phi_j\left(S+\Lambda^k-\rho Z^k\right)\phi_j(\Omega^{k+1})-1=0\\
&\Rightarrow \phi_j(\Omega^{k+1})=\frac{-\phi_j(S+\Lambda^k-\rho Z^k)\pm\sqrt{\phi_j^2(S+\Lambda^k-\rho Z^k)+4\rho}}{2\rho}
\end{aligned}$$

In summary, if we decompose $S+\Lambda^k-\rho Z^k=VQV'$ then

$$\Omega^{k+1}=\frac{1}{2\rho}V\left[-Q+(Q^2+4\rho I_p)^{1/2}\right]V' \tag{A.1}$$

A.0.2 Proof of (2.6)

Solve the optimization problem

$$Z^{k+1}=\arg\min_{Z\in\mathbb{S}^p}\left\{\lambda\left[\frac{1-\alpha}{2}\left\|Z\right\|_F^2+\alpha\left\|Z\right\|_1\right]+tr\left[\Lambda^k\left(\Omega^{k+1}-Z\right)\right]+\frac{\rho}{2}\left\|\Omega^{k+1}-Z\right\|_F^2\right\}$$

Start by taking the subgradient with respect to Z:

$$\begin{aligned}
&\partial\left\{\lambda\left[\frac{1-\alpha}{2}\left\|Z\right\|_F^2+\alpha\left\|Z\right\|_1\right]+tr\left[\Lambda^k\left(\Omega^{k+1}-Z\right)\right]+\frac{\rho}{2}\left\|\Omega^{k+1}-Z\right\|_F^2\right\}\\
&=\partial\left\{\lambda\left[\frac{1-\alpha}{2}\left\|Z\right\|_F^2+\alpha\left\|Z\right\|_1\right]\right\}+\nabla_\Omega\left\{tr\left[\Lambda^k\left(\Omega^{k+1}-Z\right)\right]+\frac{\rho}{2}\left\|\Omega^{k+1}-Z\right\|_F^2\right\}\\
&=\lambda(1-\alpha)Z+\text{sign}(Z)\lambda\alpha-\Lambda^k-\rho\left(\Omega^{k+1}-Z\right)
\end{aligned}$$

where sign is the elementwise sign operator. By setting the gradient/sub-differential equal to zero, we have that

$$Z_{ij}^{k+1}=\frac{1}{\lambda(1-\alpha)+\rho}\left(\rho\Omega_{ij}^{k+1}+\Lambda_{ij}^k-\text{sign}(Z_{ij}^{k+1})\lambda\alpha\right)$$

for all $i = 1, ..., p$ and $j = 1, ..., p$. We observe two scenarios:

- If $Z_{ij}^{k+1} > 0$ then

$$\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k} > \lambda\alpha$$

- If $Z_{ij}^{k+1} < 0$ then

$$\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k} < -\lambda\alpha$$

This implies that $\text{sign}(Z_{ij}) = \text{sign}(\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k})$. Putting all the pieces together, we arrive at

$$\begin{aligned} Z_{ij}^{k+1} &= \frac{1}{\lambda(1-\alpha)+\rho}\text{sign}\left(\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k}\right)\left(\left|\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k}\right| - \lambda\alpha\right)_{+} \\ &= \frac{1}{\lambda(1-\alpha)+\rho}\text{soft}\left(\rho\Omega_{ij}^{k+1} + \Lambda_{ij}^{k}, \lambda\alpha\right) \end{aligned}$$

where soft is the soft-thresholding function.

A.0.3 Proof of the dual residual

Here we want to show that $0 \in \rho\left(Z^{k+1} - Z^{k}\right)$ which is a direct result of the fact that Ω^{k+1} is the minimizer of the augmented lagrangian and $0 \in f\left(\Omega^{k+1}\right) + \Lambda^{k+1}$.

$$\begin{aligned} 0 &\in \partial\left\{f\left(\Omega^{k+1}\right) + tr\left[\Lambda^{k}\left(\Omega^{k+1} - Z^{k}\right)\right] + \frac{\rho}{2}\left\|\Omega^{k+1} - Z^{k}\right\|_{F}^{2}\right\} \\ &= \partial f\left(\Omega^{k+1}\right) + \Lambda^{k} + \rho\left(\Omega^{k+1} - Z^{k}\right) \\ &= \partial f\left(\Omega^{k+1}\right) + \Lambda^{k} + \rho\left(\Omega^{k+1} + Z^{k+1} - Z^{k+1} - Z^{k}\right) \\ &= \partial f\left(\Omega^{k+1}\right) + \Lambda^{k} + \rho\left(\Omega^{k+1} - Z^{k+1}\right) + \rho\left(Z^{k+1} - Z^{k}\right) \\ &= \partial f\left(\Omega^{k+1}\right) + \Lambda^{k+1} + \rho\left(Z^{k+1} - Z^{k}\right) \\ \Rightarrow 0 &\in \rho\left(Z^{k+1} - Z^{k}\right) \end{aligned}$$

A.0.4 Proof of second dual optimality condition

Here we use the primal optimality condition $\Omega^{k+1} - Z^{k+1} = 0$ to show that the second dual optimality condition $0 \in \partial g\left(Z^{k+1}\right) - \Lambda^{k+1}$ is always satisfied.

$$\begin{aligned} 0 &\in \partial \left\{ g\left(Z^{k+1}\right) + tr\left[\Lambda^k\left(\Omega^{k+1} - Z^{k+1}\right)\right] + \rho \left\|\Omega^{k+1} - Z^{k+1}\right\|_F^2 \right\} \\ &= \partial g\left(Z^{k+1}\right) - \Lambda^k - \rho\left(\Omega^{k+1} - Z^{k+1}\right) \\ &= \partial g\left(Z^{k+1}\right) - \Lambda^{k+1} \end{aligned}$$

A.0.5 Explanation of majorizing function (3.8)

To see why this particular function was used, consider the Taylor's expansion of $\rho \left\|A\Omega B - Z^k - C\right\|_F^2 /2$:

$$\begin{aligned} \frac{\rho}{2}\left\|A\Omega B - Z^k - C\right\|_F^2 \approx \frac{\rho}{2}&\left\|A\Omega^k B - Z^k - C\right\|_F^2 \\ &+ \frac{\rho}{2} vec\left(\Omega - \Omega^k\right)' \left(A'A \otimes BB'\right) vec\left(\Omega - \Omega^k\right) \\ &+ \rho vec\left(\Omega - \Omega^k\right)' vec\left(BB'\Omega^k A'A - B(Z^k)'A - BC'A\right) \end{aligned}$$

Note that the gradient and hessian, respectively, are

$$\nabla_\Omega \left\{ \frac{\rho}{2}\left\|A\Omega B - Z - C\right\|_F^2 \right\} = \rho BB'\Omega A'A - \rho BZ'A - \rho BC'A$$

$$\nabla_\Omega^2 \left\{ \frac{\rho}{2}\left\|A\Omega B - Z - C\right\|_F^2 \right\} = \rho\left(A'A \otimes BB'\right)$$

This implies that

$$
\begin{aligned}
\frac{\rho}{2}\left\|A\Omega B - Z^k - C\right\|_F^2 + \frac{\rho}{2} vec\left(\Omega - \Omega^k\right)' Q \left(\Omega - \Omega^k\right) \\
&\approx \frac{\rho}{2}\left\|A\Omega^k B - Z^k - C\right\|_F^2 + \frac{\rho}{2} vec\left(\Omega - \Omega^k\right)' Q \left(\Omega - \Omega^k\right) \\
&+ \frac{\rho}{2} vec\left(\Omega - \Omega^k\right)' \left(A'A \otimes BB'\right) vec\left(\Omega - \Omega^k\right) \\
&+ \rho vec\left(\Omega - \Omega^k\right)' vec\left(BB'\Omega^k A'A - B(Z^k)'A - BC'A\right) \\
&= \frac{\rho}{2}\left\|A\Omega^k B - Z^k - C\right\|_F^2 + \frac{\rho\tau}{2}\left\|\Omega - \Omega^k\right\|_F^2 \\
&+ \rho tr\left[\left(\Omega - \Omega^k\right)\left(BB'\Omega^k A'A - B(Z^k)'A - BC'A\right)\right]
\end{aligned}
$$

Let us now plug in this equality into our optimization problem that includes the augmented lagrangian:

$$
\begin{aligned}
\hat{\Omega}^{k+1} &= \arg\min_{\Omega \in \mathbb{S}_+^p} \tilde{L}_\rho(\Omega, Z^k, \Lambda^k) \\
&= \arg\min_{\Omega \in \mathbb{S}_+^p} \Big\{ tr\left(S\Omega\right) - \log|\Omega| + tr\left[(\Lambda^k)'(A\Omega B - Z^k - C)\right] + \rho\left\|A\Omega B - Z^k - C\right\|_F^2/2 \\
&+ vec\left(\Omega - \Omega^k\right)' \rho Q \left(\Omega - \Omega^k\right)/2 \Big\} \\
&= \arg\min_{\Omega \in \mathbb{S}_+^p} \Big\{ tr\left(S\Omega\right) - \log|\Omega| + tr\left[(\Lambda^k)'(A\Omega B - Z^k - C)\right] + \rho\left\|A\Omega^k B - Z^k - C\right\|_F^2/2 \\
&+ \rho\tau\left\|\Omega - \Omega^k\right\|_F^2/2 + tr\left[\rho\left(\Omega - \Omega^k\right)\left(BB'\Omega^k A'A - B(Z^k)'A - BC'A\right)\right] \Big\} \\
&= \arg\min_{\Omega \in \mathbb{S}_+^p} \Big\{ tr\left[\left(S + \rho A'(A\Omega^k B - Z^k - C + \Lambda^k/\rho)B'\right)\Omega\right] \\
&- \log|\Omega| + \rho\tau\left\|\Omega - \Omega^k\right\|_F^2/2 \Big\} \\
&= \arg\min_{\Omega \in \mathbb{S}_+^p} \Big\{ tr\left[\left(S + G^k\right)\Omega\right] - \log|\Omega| + \rho\tau\left\|\Omega - \Omega^k\right\|_F^2/2 \Big\}
\end{aligned}
$$

where $G^k = \rho A'(A\Omega^k B - Z^k - C + \Lambda^k/\rho)B'$.

A.0.6 Proof of (3.9)

We show that the following optimization problem can be solved in closed-form similar to (A.1).

$$\Omega^{k+1} = \arg\min_{\Omega \in \mathbb{S}_+^p} \left\{ tr\left[\left(S + G^k\right)\Omega\right] - \log|\Omega| + \frac{\rho\tau}{2}\left\|\Omega - \Omega^k\right\|_F^2 \right\}$$

First start by taking the gradient with respect to Ω:

$$\begin{aligned}
&\nabla_\Omega \left\{ tr\left[\left(S + G^k\right)\Omega\right] - \log|\Omega| + \frac{\rho\tau}{2}\left\|\Omega - \Omega^k\right\|_F^2 \right\} \\
&= 2S - S \circ I_p + G^k + (G^k)' - G^k \circ I_p - 2\Omega^{-1} + \Omega^{-1} \circ I_p \\
&+ \frac{\rho\tau}{2}\left[2\Omega - 2(\Omega^k)' + 2\Omega' - 2\Omega^k - 2(\Omega - \Omega^k)' \circ I_p\right]
\end{aligned}$$

Note that we need to honor the symmetric constraint given by Ω. By setting the gradient equal to zero and multiplying all off-diagonal elements by $1/2$, this simplifies to

$$S + \frac{1}{2}\left(G^k + (G^k)'\right) - \rho\tau\Omega^k = (\Omega^{k+1})^{-1} - \rho\tau\Omega^{k+1}$$

We can then decompose using spectral decomposition $\Omega^{k+1} = VDV'$ where D is a diagonal matrix with diagonal elements equal to the eigen values of Ω^{k+1} and V is the matrix with corresponding eigen vectors as columns.

$$S + \frac{1}{2}\left(G^k + (G^k)'\right) - \rho\tau\Omega^k = VD^{-1}V' - \rho\tau VDV' = V\left(D^{-1} - \rho\tau D\right)V'$$

This equivalence implies that

$$\phi_j\left(D^k\right) = \frac{1}{\phi_j(\Omega^{k+1})} - \rho\tau\phi_j(\Omega^{k+1})$$

where $\phi_j(\cdot)$ is the jth eigen value and $D^k = S + \left(G^k + (G^k)'\right)/2 - \rho\tau\Omega^k$. Therefore

$$\begin{aligned}
&\Rightarrow \rho\tau\phi_j^2(\Omega^{k+1}) + \phi_j\left(D^k\right)\phi_j(\Omega^{k+1}) - 1 = 0 \\
&\Rightarrow \phi_j(\Omega^{k+1}) = \frac{-\phi_j(D^k) \pm \sqrt{\phi_j^2(D^k) + 4\rho\tau}}{2\rho\tau}
\end{aligned}$$

In summary, if we decompose $S + \left(G^k + (G^k)'\right)/2 - \rho\tau\Omega^k = VQV'$ then

$$\Omega^{k+1} = \frac{1}{2\rho\tau} V \left[-Q + (Q^2 + 4\rho\tau I_p)^{1/2} \right] V'$$

A.0.7 Proof of (3.10)

Solve the optimization problem

$$Z^{k+1} = \arg\min_{Z \in \mathbb{R}^{n \times r}} \left\{ \lambda \|Z\|_1 + tr\left[(\Lambda^k)' \left(A\Omega^{k+1} B - Z - C \right) \right] + \frac{\rho}{2} \left\| A\Omega^{k+1} B - Z - C \right\|_F^2 \right\}$$

Start by taking the subgradient with respect to Z:

$$\begin{aligned}
&\partial \left\{ \lambda \|Z\|_1 + tr\left[(\Lambda^k)' \left(A\Omega^{k+1} B - Z - C \right) \right] + \frac{\rho}{2} \left\| A\Omega^{k+1} B - Z - C \right\|_F^2 \right\} \\
&= \partial \left\{ \lambda \|Z\|_1 \right\} + \nabla_\Omega \left\{ tr\left[(\Lambda^k)' \left(A\Omega^{k+1} B - Z - C \right) \right] + \frac{\rho}{2} \left\| A\Omega^{k+1} B - Z - C \right\|_F^2 \right\} \\
&= \text{sign}(Z)\lambda - \Lambda^k - \rho \left(A\Omega^{k+1} B - Z - C \right)
\end{aligned}$$

where sign(Z) is the elementwise sign operator. By setting the gradient/sub-differential equal to zero, we arrive at the following equivalence:

$$Z_{ij}^{k+1} = \frac{1}{\rho} \left(\rho \left(A\Omega_{ij}^{k+1} B - C \right) + \Lambda_{ij}^k - Sign\left(Z_{ij}^{k+1} \right) \lambda \right)$$

for all $i = 1, ..., p$ and $j = 1, ..., p$. We observe two scenarios:

- If $Z_{ij}^{k+1} > 0$ then

$$\rho \left(A\Omega_{ij}^{k+1} B - C \right) + \Lambda_{ij}^k > \lambda\alpha$$

- If $Z_{ij}^{k+1} < 0$ then

$$\rho \left(A\Omega_{ij}^{k+1} B - C \right) + \Lambda_{ij}^k < -\lambda\alpha$$

This implies that $\text{sign}\left(Z_{ij}^{k+1}\right) = \text{sign}\left(\rho\left(A\Omega_{ij}^{k+1}B - C\right) + \Lambda_{ij}^{k}\right)$. Putting all the pieces together, we arrive at

$$\begin{aligned} Z_{ij}^{k+1} &= \frac{1}{\rho}\text{sign}\left(\rho\left(A\Omega_{ij}^{k+1}B - C\right) + \Lambda_{ij}^{k}\right)\left(\left|\rho\left(A\Omega_{ij}^{k+1}B - C\right) + \Lambda_{ij}^{k}\right| - \lambda\right)_{+} \\ &= \frac{1}{\rho}\text{soft}\left(\rho\left(A\Omega_{ij}^{k+1}B - C\right) + \Lambda_{ij}^{k}, \lambda\right) \end{aligned}$$

where soft is the soft-thresholding function.

A.0.8 Proof of (3.11)

Here we want to show that $0 \in \rho\left(B(Z^{k+1} - Z^k)'A + A'(Z^{k+1} - Z^k)B'\right)/2$ which is a direct result of the fact that $0 \in \partial f\left(\Omega^{k+1}\right) + \left(B(\Lambda^{k+1})'A + A'\Lambda^{k+1}B'\right)/2$.

$$\begin{aligned} 0 &\in \partial\left\{f\left(\Omega^{k+1}\right) + tr\left[\Lambda^k\left(A\Omega^{k+1}B - Z^k - C\right)\right] + \rho\left\|A\Omega^{k+1}B - Z^k - C\right\|_F^2/2\right\} \\ &= \partial f\left(\Omega^{k+1}\right) + \left(B(\Lambda^k)'A + A'\Lambda^k B'\right)/2 + \rho\left(BB'\Omega^{k+1}A'A + A'A\Omega^{k+1}BB'\right)/2 \\ &- \rho\left(A'(Z^k + C)B' + B(Z^k + C)'A\right)/2 \\ &= \partial f\left(\Omega^{k+1}\right) + \left(B(\Lambda^k)'A + A'\Lambda^k B'\right)/2 \\ &+ \rho\left(B(B'\Omega^{k+1}A' - (Z^k)' - C')A + A'(A\Omega^{k+1}B - Z^k - C)B'\right)/2 \\ &= \partial f\left(\Omega^{k+1}\right) + \left(B(\Lambda^k)'A + A'\Lambda^k B'\right)/2 + \rho\left(A'(A\Omega^{k+1}B - Z^{k+1} + Z^{k+1} - Z^k - C)B'\right)/2 \\ &+ \rho\left(B(B'\Omega^{k+1}A' - (Z^{k+1})' + (Z^{k+1})' - (Z^k)' - C')A\right)/2 \\ &= \partial f\left(\Omega^{k+1}\right) + \left[B\left((\Lambda^k)' + \rho(B'\Omega^{k+1}A' - (Z^{k+1})' - C')\right)A\right]/2 \\ &+ \left[A'\left(\Lambda^k + \rho(A\Omega^{k+1}B - Z^{k+1} - c)B\right)B'\right]/2 + \rho\left(B(Z^{k+1} - Z^k)'A + A'(Z^{k+1} - Z^k)B'\right)/2 \\ &= \partial f\left(\Omega^{k+1}\right) + \left(B(\Lambda^{k+1})'A + A'\Lambda^{k+1}B'\right)/2 + \rho\left(B(Z^{k+1} - Z^k)'A + A'(Z^{k+1} - Z^k)B'\right)/2 \\ \Rightarrow 0 &\in \rho\left(B(Z^{k+1} - Z^k)'A + A'(Z^{k+1} - Z^k)B'\right)/2 \end{aligned}$$

A.0.9 Proof of no closed-form solution

Here we want to show that the following optimization problem cannot be solved in closed-form.

$$\hat{\Omega} = \arg\min_{\Omega \in \mathbb{S}_+^p} \left\{ tr\left(S\Omega\right) - \log|\Omega| + \frac{\lambda}{2} \left\| \mathbb{X}\Omega\Sigma_{xy} - \mathbb{Y} \right\|_F^2 \right\}$$

First start by taking the gradient with respect to Ω:

$$\begin{aligned}
&\nabla_\Omega \left\{ tr\left(S\Omega\right) - \log|\Omega| + \frac{\lambda}{2} \left\| \mathbb{X}\Omega\Sigma_{xy} - \mathbb{Y} \right\|_F^2 \right\} \\
&\quad = \nabla_\Omega \left\{ tr\left(S\Omega\right) - \log|\Omega| + \frac{\lambda}{2} tr\left[\left(\mathbb{X}\Omega\Sigma_{xy} - \mathbb{Y} \right)' \left(\mathbb{X}\Omega\Sigma_{xy} - \mathbb{Y} \right) \right] \right\} \\
&\quad = \nabla_\Omega \left\{ tr\left(S\Omega\right) - \log|\Omega| + \frac{\lambda}{2} tr\left(\Omega\mathbb{X}'\mathbb{X}\Omega\Sigma_{xy}\Sigma_{xy}' - 2\Omega\Sigma_{xy}\mathbb{Y}'\mathbb{X} \right) \right\} \\
&\quad = 2S - S \circ I_p - 2\Omega^{-1} + \Omega^{-1} \circ I_p + \lambda\mathbb{X}'\mathbb{X}\Omega\Sigma_{xy}\Sigma_{xy}' + \lambda\Sigma_{xy}\Sigma_{xy}'\Omega\mathbb{X}'\mathbb{X} \\
&\quad - \lambda\mathbb{X}'\mathbb{X}\Omega\Sigma_{xy}\Sigma_{xy}' \circ I_p - \Sigma_{xy}\mathbb{Y}'\mathbb{X} - \mathbb{X}'\mathbb{Y}\Sigma_{xy} + \Sigma_{xy}\mathbb{Y}'\mathbb{X} \circ I_P \\
\Rightarrow 0 &= S - \hat{\Omega}^{-1} + \frac{\lambda}{2} \left(\mathbb{X}'\mathbb{X}\hat{\Omega}\Sigma_{xy}\Sigma_{xy}' + \Sigma_{xy}\Sigma_{xy}'\hat{\Omega}\mathbb{X}'\mathbb{X} \right) - \lambda\Sigma_{xy}\mathbb{Y}'\mathbb{X}
\end{aligned}$$

Note that we cannot isolate $\hat{\Omega}$.

Appendix B

`ADMMsigma` R Package

`ADMMsigma` is an R package that estimates a penalized precision matrix, often denoted Ω, via the alternating direction method of multipliers (ADMM) algorithm. Though not the fastest estimation method, the ADMM algorithm is easily adaptable and allows for rapid experimentation by the user, which is the primary goal of this package.

The package currently supports a general elastic-net penalty that allows for both ridge and lasso-type penalties as special cases. In particular, the algorithm solves the following optimization problem:

$$\hat{\Omega} = \arg\min_{\Omega \in S_+^p} \left\{ tr(S\Omega) - \log\det(\Omega) + \lambda \left[\frac{1-\alpha}{2} \|\Omega\|_F^2 + \alpha \|\Omega\|_1 \right] \right\}$$

where $\lambda > 0$, $0 \leq \alpha \leq 1$ are tuning parameters, $\|\cdot\|_F^2$ is the Frobenius norm, and we define $\|A\|_1 = \sum_{i,j} |A_{ij}|$.

A list of functions contained in the package can be found below:

- `ADMMsigma()` computes the estimated precision matrix via the ADMM algorithm (ridge, lasso, and elastic-net regularization optional)
- `RIDGEsigma()` computes the estimated ridge penalized precision matrix via closed-form solution
- `plot.ADMMsigma()` produces a heat map or optional line graph for cross validation errors

- `plot.RIDGEsigma()` produces a heat map or optional line graph for cross validation errors

B.1 Installation

```
# The easiest way to install is from CRAN
install.packages("ADMMsigma")

# You can also install the development version from GitHub:
# install.packages('devtools')
devtools::install_github("MGallow/ADMMsigma")
```

This package is hosted on Github at github.com/MGallow/ADMMsigma. The project website is located at mattxgalloway.com/ADMMsigma.

B.2 Tutorial

By default, `ADMMsigma` will estimate a penalized precision matrix, Ω, using the elastic-net penalty and choose the optimal λ and α tuning parameters. The primary function is simply `ADMMsigma()`. The input value X is an $n \times p$ data matrix so that there are n rows each representing an observation and p columns each representing a unique feauture or variable.

Here, we will use a 100×5 data matrix that is generated from a multivariate normal distribution with mean zero and tapered oracle covariance matrix S. A tapered covariance matrix has an inverse - or precision matrix - that is tri-diagonal, which is useful for illustration purposes because the object is sparse (many zero entries) and the shrinkage penalty should prove useful.

```
# elastic-net type penalty (set tolerance to 1e-8)
ADMMsigma(X, tol.abs = 1e-08, tol.rel = 1e-08)
```

```
##
## Call: ADMMsigma(X = X, tol.abs = 1e-08, tol.rel = 1e-08)
```

```
##
## Iterations: 48
##
## Tuning parameters:
##       log10(lam)  alpha
## [1,]      -1.599      1
##
## Log-likelihood: -108.41003
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  2.15283 -1.26902  0.00000  0.00000  0.19765
## [2,] -1.26902  2.79032 -1.32206 -0.08056  0.00925
## [3,]  0.00000 -1.32206  2.85470 -1.17072 -0.00865
## [4,]  0.00000 -0.08056 -1.17072  2.49554 -1.18959
## [5,]  0.19765  0.00925 -0.00865 -1.18959  1.88121
```

We can see that the optimal α value selected is one. This selection corresponds with a lasso penalty – a special case of the elastic-net penalty.

We can also explicitly assume sparsity in our estimate rather than let the package decide by restricting the class of penalties to the lasso. We do this by simply setting `alpha = 1` in our function:

```r
# lasso penalty (default tolerance)
ADMMsigma(X, alpha = 1)
```

```
##
## Call: ADMMsigma(X = X, alpha = 1)
##
## Iterations: 24
##
```

```
## Tuning parameters:
##       log10(lam)  alpha
## [1,]      -1.599      1
##
## Log-likelihood: -108.41193
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  2.15308 -1.26962  0.00000  0.00000  0.19733
## [2,] -1.26962  2.79103 -1.32199 -0.08135  0.00978
## [3,]  0.00000 -1.32199  2.85361 -1.16953 -0.00921
## [4,]  0.00000 -0.08135 -1.16953  2.49459 -1.18914
## [5,]  0.19733  0.00978 -0.00921 -1.18914  1.88096
```

`ADMMsigma` also has the capability to provide plots for the cross validation errors. This allows the user to analyze and manually select the appropriate tuning parameters. In the heatmap plot below, the bright, white areas of the heat map correspond to a better tuning parameter selection (low validation error). In the line graph, each line corresponds to a different α tuning parameter.

```
# produce CV heat map for ADMMsigma
ADMM = ADMMsigma(X, tol.abs = 1e-08, tol.rel = 1e-08)
plot(ADMM, type = "heatmap")
```

```
# produce line graph for CV errors for ADMMsigma
plot(ADMM, type = "line")
```

`ADMMsigma` has a number of more advanced options such as cross validation criteria, regularization path, and parallel CV that are explained in more detail on the project website at mattxgalloway.com/ADMMsigma.

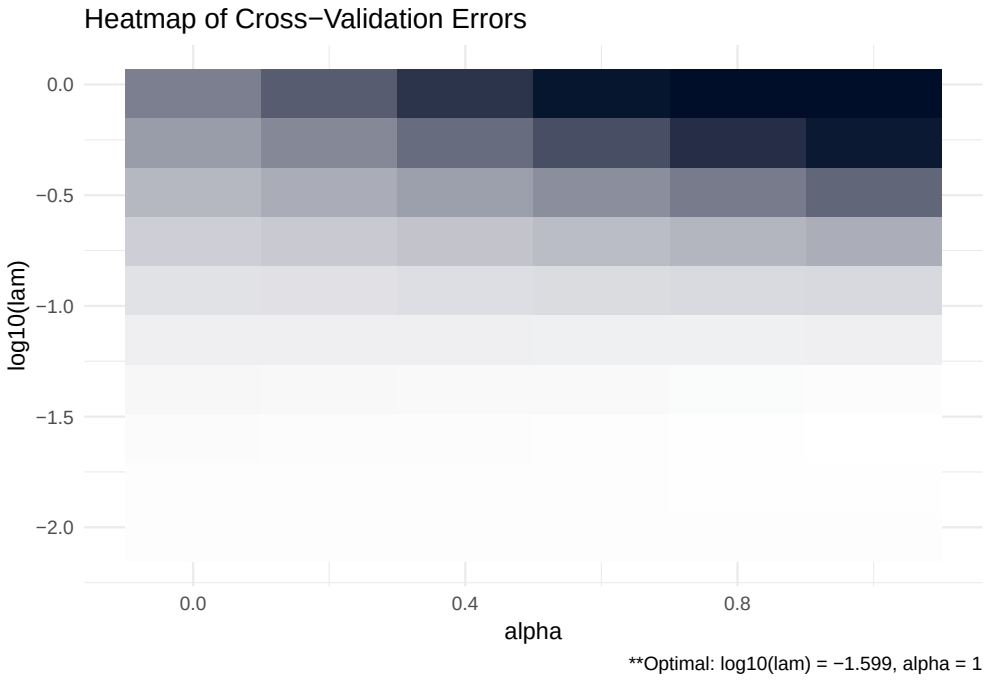

Figure B.1: CV heatmap for ADMMsigma tutorial.

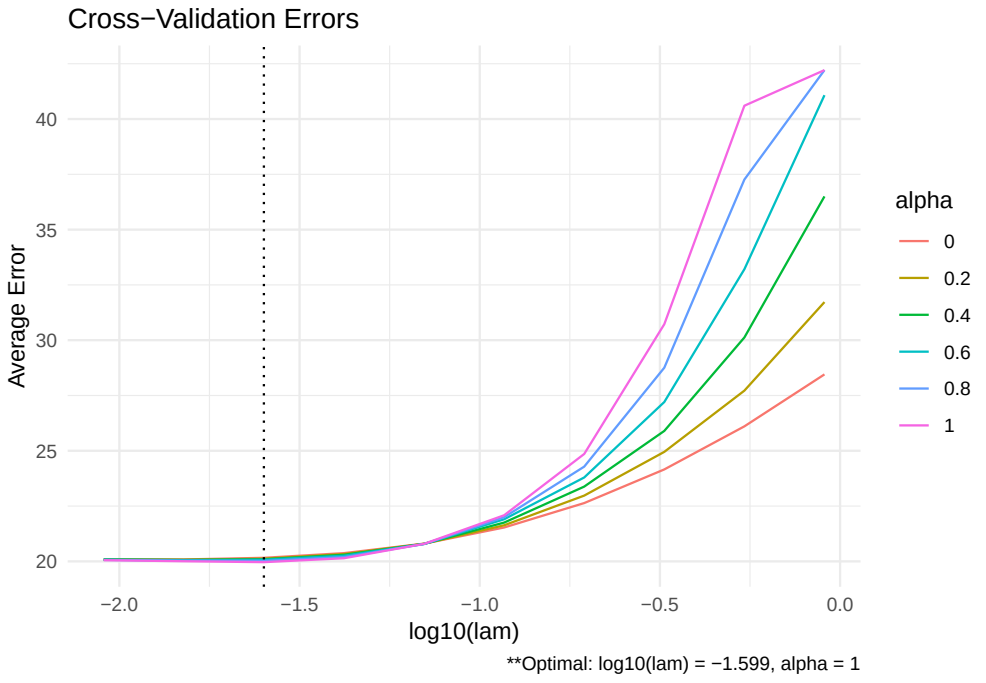

Figure B.2: CV line graph for ADMMsigma tutorial.

Appendix C

SCPME R Package

SCPME is an R package that estimates a penalized precision matrix via a modified alternating direction method of multipliers (ADMM) algorithm as described in Molstad and Rothman (2017). Specifically, the modified ADMM algorithm solves the following optimization problem:

$$\hat{\Omega} = \arg\min_{\Omega \in S_+^p} \left\{ tr\left(S\Omega\right) - \log\det\left(\Omega\right) + \lambda \left\| A\Omega B - C \right\|_1 \right\}$$

where $\lambda > 0$ is a tuning parameter, $A, B,$ and C are known, user-specified matrices, and we define $\|A\|_1 = \sum_{i,j} |A_{ij}|$.

This form of penalty leads to many new, interesting, and novel estimators for the precision matrix Ω. Users can construct matrices $A, B,$ and C so that emphasis is placed on the sum, absolute value of a *characteristic* of Ω. We will explore a few of these estimators in the tutorial section.

A list of functions contained in the package can be found below:

- `shrink()` computes the estimated precision matrix
- `data_gen()` data generation function (for convenience)
- `plot.shrink()` produces a heat map or optional line graph for the cross validation errors

C.1 Installation

```r
# The easiest way to install is from CRAN
install.packages("SCPME")

# You can also install the development version from GitHub:
# install.packages('devtools')
devtools::install_github("MGallow/SCPME")
```

This package is hosted on Github at github.com/MGallow/SCPME. The project website is located at mattxgalloway.com/SCPME.

C.2 Tutorial

The primary function in the `SCPME` package is `shrink()`. The input values X is an $n \times p$ data matrix so that there are n rows each representing an observation and p columns each representing a unique variable and Y is an $n \times r$ response matrix where r is the dimension of the response vector. By default, `SCPME` will estimate Ω using a lasso penalty ($A = I_p, B = I_p,$ and $C = 0$) and choose the optimal λ tuning parameter that minimizes the mean squared prediction error for the regression of the variable Y on X (here I_p denotes a p-dimension identity matrix). If Y is not provided, then tuning parameter selection will be based on the validation likelihood. Note that Ω (perhaps better denoted here as Ω_x) will only have meaningful shrinkage unless the data vector $X \in \mathbb{R}^p$ is multi-dimensional ($p > 1$).

In this example, the data matrix is 100×5 and the response is generated according to the following model:

$$Y_i = \beta' X_i + E_i$$

where $E_i \sim N(0, 1)$ and X_i is generated from a multivariate normal distribution with mean zero and tapered oracle covariance matrix S. A tapered covariance matrix has the property that its

inverse - the precision matrix - is tri-diagonal. Estimating this oracle precision matrix well and efficiently will be our primary interest. In addition, β is randomly generated and sparse. The data will be generated using the `data_gen()` function contained in the package.

```
library(SCPME)
set.seed(123)

# generate 100 x 5 X data matrix and 100 x 1 Y data matrix
data = data_gen(p = 5, n = 100, r = 1)

# the oracle regression coefficients are sparse
data$betas
```

```
##              [,1]
## [1,] -0.25065233
## [2,]  0.00000000
## [3,]  0.69707555
## [4,]  0.03153231
## [5,]  0.00000000
```

```
# shrink sum absolute entries in omega
shrink(X = data$X, Y = data$Y)
```

```
##
## Call: shrink(X = data$X, Y = data$Y)
##
## Iterations: 37
##
## Tuning parameters:
##       log10(lam)    lam
## [1,]      -1.163  0.069
```

```
##
## Log-likelihood: -178.20154
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  1.60847 -0.73553 -0.14094 -0.04329 -0.11730
## [2,] -0.73553  1.66045 -0.52579 -0.03576 -0.03342
## [3,] -0.14094 -0.52579  1.73410 -0.85121 -0.07332
## [4,] -0.04329 -0.03576 -0.85121  2.02541 -0.93612
## [5,] -0.11730 -0.03342 -0.07332 -0.93612  1.62397
```

Notice here that the estimated precision matrix is *not* sparse. This is due to the fact that our cross validation criteria is the mean-squared prediction error. We can estimate a new precision matrix using the validation likelihood as the cross validation criteria with the following command:

```
# shrink sum absolute entries in omega
shrink(X = data$X, Y = data$Y, crit.cv = "loglik")
```

```
##
## Call: shrink(X = data$X, Y = data$Y, crit.cv = "loglik")
##
## Iterations: 51
##
## Tuning parameters:
##       log10(lam)    lam
## [1,]      -2.163  0.007
##
## Log-likelihood: -120.02858
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
```

```
## [1,]  2.11926 -1.17294 -0.13784 -0.00678 -0.20014
## [2,] -1.17294  2.28420 -0.81629  0.00009 -0.00001
## [3,] -0.13784 -0.81629  2.45520 -1.42117  0.01650
## [4,] -0.00678  0.00009 -1.42117  3.09526 -1.56839
## [5,] -0.20014 -0.00001  0.01650 -1.56839  2.24703
```

All of the estimators so far have used a lasso penalty that penalizes the sum of the absolute value of all the entries in Ω ($A = I_p, B = I_p,$ and $C = 0$). In effect, this penalty embeds an assumption in our estimate that the true population precision matrix, Ω, is sparse. The flexibility of the penalty described in Molstad and Rothman (2017) allows us to make other assumptions as well. For instance, in the penalty we could set $A = I_p, B = \Sigma_{xy}$ where Σ_{xy} is the covariance matrix of X and Y, and $C = 0$. In which case our penalty function

$$P_\lambda(\Omega) = \lambda \|A\Omega B - C\|_1 = \lambda \left\|\Omega\Sigma_{xy}\right\|_1 = \lambda \|\beta\|_1$$

This objective function estimates an Ω via the marginal log-likelihood of X under the assumption that the forward regression coefficient β is sparse (recall that $\beta \equiv \Omega\Sigma_{xy}$). Of course, in practice, we do not know the true covariance matrix Σ_{xy} but we might consider using the sample estimate $\hat{\Sigma}_{xy} = \sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})' / n$.

```
# assume sparsity in beta
lam_max = max(abs(crossprod(data$X, data$Y)))
(shrink = shrink(X = data$X, Y = data$Y, B = cov(data$X, data$Y),
    lam.max = lam_max, nlam = 20))
```

```
##
## Call: shrink(X = data$X, Y = data$Y, B = cov(data$X, data$Y), nlam = 20,
##     lam.max = lam_max)
##
## Iterations: 84
##
```

```
## Tuning parameters:
##       log10(lam)    lam
## [1,]      -0.167  0.681
##
## Log-likelihood: -133.98097
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  2.12467 -1.20016 -0.01149  0.01660 -0.20424
## [2,] -1.20016  2.28202 -0.70370  0.03047 -0.01211
## [3,] -0.01149 -0.70370  2.09284 -1.47505  0.01020
## [4,]  0.01660  0.03047 -1.47505  2.86829 -1.45784
## [5,] -0.20424 -0.01211  0.01020 -1.45784  2.18752
```

Note that we specified the maximum `lam` value in the previous function to expand the tuning parameter grid. With these settings, the augmented ADMM algorithm also solves for the estimated β coefficient matrix simultaneously:

```
# print estimated beta matrix
shrink$Z
```

```
##            [,1]
## [1,] 0.00000000
## [2,] 0.00000000
## [3,] 0.42221120
## [4,] 0.04782093
## [5,] 0.00000000
```

Another possible penalty is to set $B = \left[\Sigma_{xy}, I_p\right]$ so that the identity matrix is appended to the covariance matrix of X and Y. That is, the penalty P, is constructed as

$$P_\lambda(\Omega) = \lambda \left\| A\Omega B - C \right\|_1 = \lambda \left\| \Omega \left[\Sigma_{xy}, I_p\right] \right\|_1 = \lambda \left\| \beta \right\|_1 + \lambda \left\| \Omega \right\|_1$$

In this case, we are equally penalizing the sum, absolute values of entries in β *and* Ω which embeds an assumption that β and Ω are both sparse.

```
# assume sparsity in beta AND omega
(shrink = shrink(X = data$X, Y = data$Y, B = cbind(cov(data$X,
    data$Y), diag(ncol(data$X))), lam.max = 10, lam.min.ratio = 1e-04,
    nlam = 20))
```

```
##
## Call: shrink(X = data$X, Y = data$Y, B = cbind(cov(data$X, data$Y),
##     diag(ncol(data$X))), nlam = 20, lam.max = 10, lam.min.ratio = 1e-04)
##
## Iterations: 46
##
## Tuning parameters:
##       log10(lam)    lam
## [1,]       0.368  2.336
##
## Log-likelihood: -624.54758
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  0.26376 -0.00003 -0.00015 -0.00010 -0.00005
## [2,] -0.00003  0.24002 -0.00017 -0.00012 -0.00006
## [3,] -0.00015 -0.00017  0.19066 -0.00516 -0.00020
## [4,] -0.00010 -0.00012 -0.00516  0.20362 -0.00014
## [5,] -0.00005 -0.00006 -0.00020 -0.00014  0.22750
```

```
# print estimated beta
shrink$Z[, 1, drop = FALSE]
```

```
##            [,1]
```

```
## [1,] 0.06389361
## [2,] 0.08542992
## [3,] 0.14200713
## [4,] 0.12357129
## [5,] 0.09958374
```

SCPME also has the capability to provide plots (heatmaps and line graphs) for the cross validation errors. In the heatmap plot below, the more bright, white areas correspond to a better tuning parameter selection (lower cross validation error).

```
# produce CV heat map
plot(shrink, type = "heatmap")
```

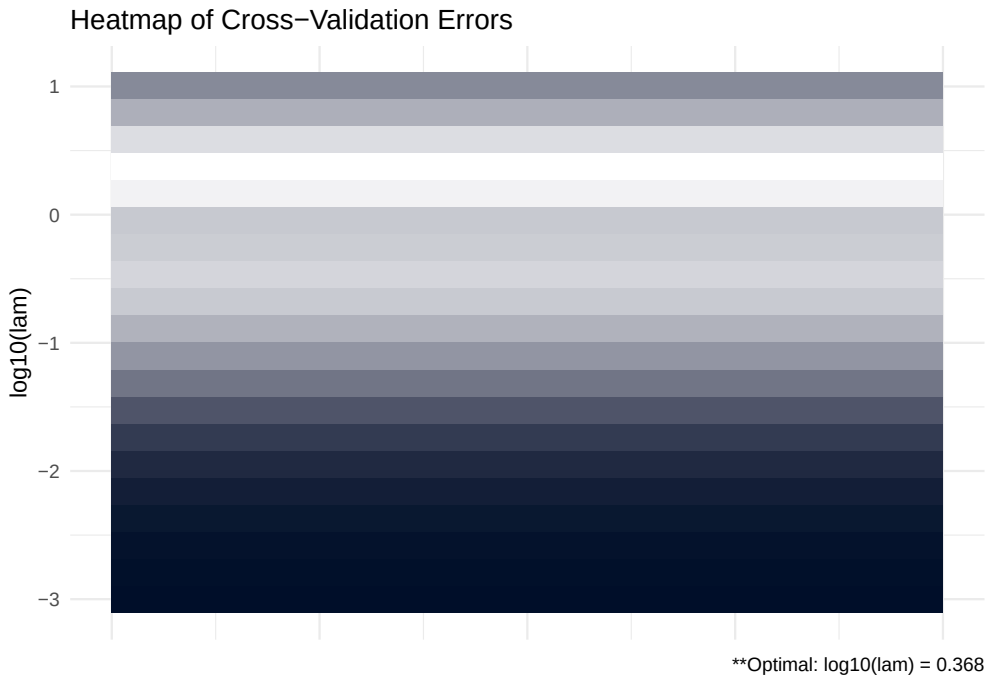

Figure C.1: CV heatmap for SCPME tutorial

```
# produce line graph
plot(shrink, type = "line")
```

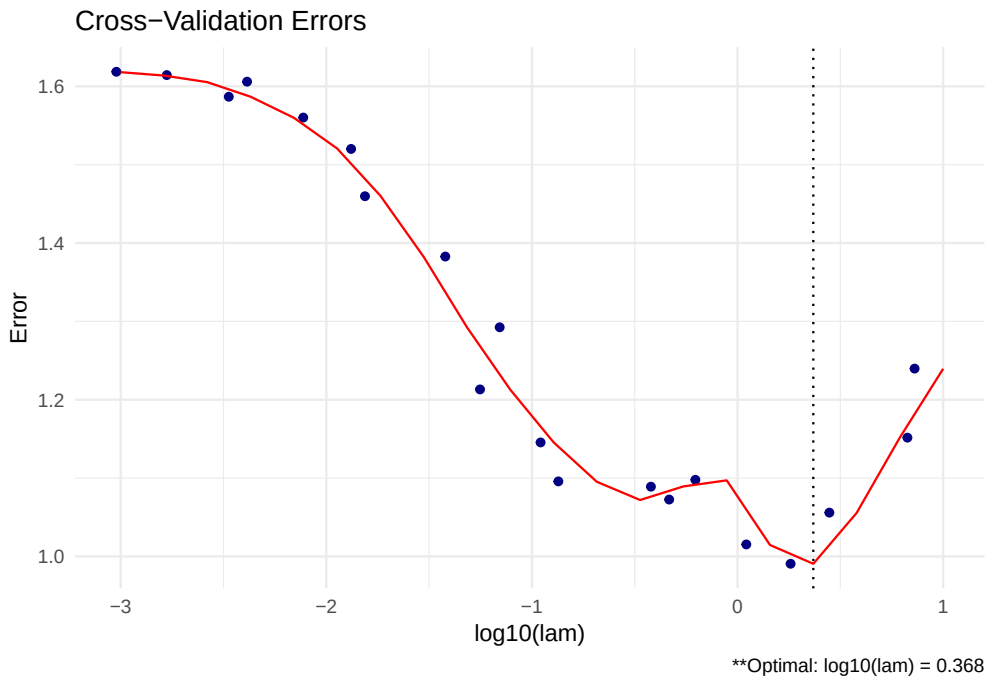

Figure C.2: CV line graph for SCPME tutorial

`SCPME` has a number of more advanced options including alternative convergence criteria and parallel CV that are explained in detail on the project website at mattxgalloway.com/SCPME.

Bibliography

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Cai, T. and Liu, W. (2011). A direct estimation approach to sparse linear discriminant analysis. *Journal of the American statistical association*, 106(496):1566–1577.

Dalal, O. and Rajaratnam, B. (2017). Sparse gaussian graphical model estimation via alternating minimization. *Biometrika*, 104(2):379–395.

Fan, J., Feng, Y., and Tong, X. (2012). A road to classification in high dimensional space: the regularized optimal affine discriminant. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(4):745–771.

Fan, J., Feng, Y., and Wu, Y. (2009). Network exploration via the adaptive lasso and scad penalties. *The annals of applied statistics*, 3(2):521.

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

Lam, C. and Fan, J. (2009). Sparsistency and rates of convergence in large covariance matrix estimation. *Annals of statistics*, 37(6B):4254.

Lange, K. (2016). *MM optimization algorithms*, volume 147. SIAM.

Mai, Q., Zou, H., and Yuan, M. (2012). A direct approach to sparse discriminant analysis in ultra-high dimensions. *Biometrika*, 99(1):29–42.

Molstad, A. J. and Rothman, A. J. (2016). Indirect multivariate response linear regression. *Biometrika*, 103(3):595–607.

Molstad, A. J. and Rothman, A. J. (2017). Shrinking characteristics of precision matrix estimators. *Biometrika*.

Molstad, A. J. and Rothman, A. J. (2018). A penalized likelihood method for classification with matrix-valued predictors. *Journal of Computational and Graphical Statistics*, pages 1–12.

Price, B. S., Geyer, C. J., and Rothman, A. J. (2015). Ridge fusion in statistical learning. *Journal of Computational and Graphical Statistics*, 24(2):439–454.

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

Rothman, A. J., Forzani, L., et al. (2014). On the existence of the weighted bridge penalized gaussian likelihood precision matrix estimator. *Electronic Journal of Statistics*, 8(2):2693–2700.

Witten, D. M. and Tibshirani, R. (2009). Covariance-regularized regression and classification for high dimensional problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):615–636.

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

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.