Oral Manuscript

Matt Galloway

2018-06-20

Contents

Chapter 1

Prerequisites

Chapter 2

Introduction

This is the introduction for Matt's oral manuscript. (UPDATED 8)

Chapter 3

Tutorial

```r
# 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")
```

If there are any issues/bugs, please let me know: github. You can also contact me via my website. Pull requests are welcome!

A (possibly incomplete) list of functions contained in the package can be found below:

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

3.1 Usage

We will first generate data from a sparse, tri-diagonal precision matrix and denote it as Omega.

```
library(ADMMsigma)

# generate data from a sparse matrix first compute covariance matrix
S = matrix(0.7, nrow = 5, ncol = 5)
for (i in 1:5) {
    for (j in 1:5) {
        S[i, j] = S[i, j]^abs(i - j)
    }
}

# print oracle precision matrix (shrinkage might be useful)
(Omega = round(qr.solve(S), 3))
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]
## [1,]  1.961 -1.373  0.000  0.000  0.000
## [2,] -1.373  2.922 -1.373  0.000  0.000
## [3,]  0.000 -1.373  2.922 -1.373  0.000
## [4,]  0.000  0.000 -1.373  2.922 -1.373
## [5,]  0.000  0.000  0.000 -1.373  1.961
```

```
# generate 100 x 5 matrix with rows drawn from iid N_p(0, S)
set.seed(123)
Z = matrix(rnorm(100 * 5), nrow = 100, ncol = 5)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5) %*% t(out$vectors)
X = Z %*% S.sqrt

# snap shot of data
head(X)
```

```
##            [,1]         [,2]         [,3]       [,4]        [,5]
## [1,] -0.4311177 -0.217744186  1.276826576 -0.1061308 -0.02363953
## [2,] -0.0418538  0.304253474  0.688201742 -0.5976510 -1.06758924
## [3,]  1.1344174  0.004493877 -0.440059159 -0.9793198 -0.86953222
## [4,] -0.0738241 -0.286438212  0.009577281 -0.7850619 -0.32351261
## [5,] -0.2905499 -0.906939891 -0.656034183 -0.4324413  0.28516534
## [6,]  1.3761967  0.276942730 -0.297518545 -0.2634814 -1.35944340
```

As described earlier in the report, the maximum likelihood estimator (MLE) for Omega is the inverse of the sample precision matrix $S^{-1} = \left[\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T/n\right]^{-1}$:

```
# print inverse of sample precision matrix (perhaps a bad estimate)
round(qr.solve(cov(X) * (nrow(X) - 1)/nrow(X)), 5)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  2.32976 -1.55033  0.22105 -0.08607  0.24309
## [2,] -1.55033  3.27561 -1.68026 -0.14277  0.18949
## [3,]  0.22105 -1.68026  3.19897 -1.25158 -0.11016
## [4,] -0.08607 -0.14277 -1.25158  2.76790 -1.37226
## [5,]  0.24309  0.18949 -0.11016 -1.37226  2.05377
```

However, because Omega (known as the *oracle*) is sparse, a shrinkage estimator will perhaps perform better than the sample estimator. Below we construct various penalized estimators:

```
# 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: 162
##
## 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
```

LASSO:

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

```
##
## Call: ADMMsigma(X = X, alpha = 1)
##
## Iterations: 66
##
## Tuning parameters:
##       log10(lam)  alpha
## [1,]      -1.599      1
##
## Log-likelihood: -108.41022
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  2.15228 -1.26841  0.00000  0.00000  0.19744
## [2,] -1.26841  2.78830 -1.31943 -0.08246  0.01018
## [3,]  0.00000 -1.31943  2.84979 -1.16708 -0.01015
## [4,]  0.00000 -0.08246 -1.16708  2.49277 -1.18844
## [5,]  0.19744  0.01018 -0.01015 -1.18844  1.88069
```

ELASTIC-NET:

```
# elastic-net penalty (alpha = 0.5)
ADMMsigma(X, alpha = 0.5)
```

```
##
## Call: ADMMsigma(X = X, alpha = 0.5)
##
## Iterations: 67
```

```
##
## Tuning parameters:
##       log10(lam)  alpha
## [1,]      -1.821    0.5
##
## Log-likelihood: -101.13595
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  2.20031 -1.32471  0.01656 -0.00334  0.21798
## [2,] -1.32471  2.90659 -1.37599 -0.19084  0.13651
## [3,]  0.01656 -1.37599  2.92489 -1.12859 -0.12033
## [4,] -0.00334 -0.19084 -1.12859  2.56559 -1.23472
## [5,]  0.21798  0.13651 -0.12033 -1.23472  1.94528
```

RIDGE:

```
# ridge penalty
ADMMsigma(X, alpha = 0)
```

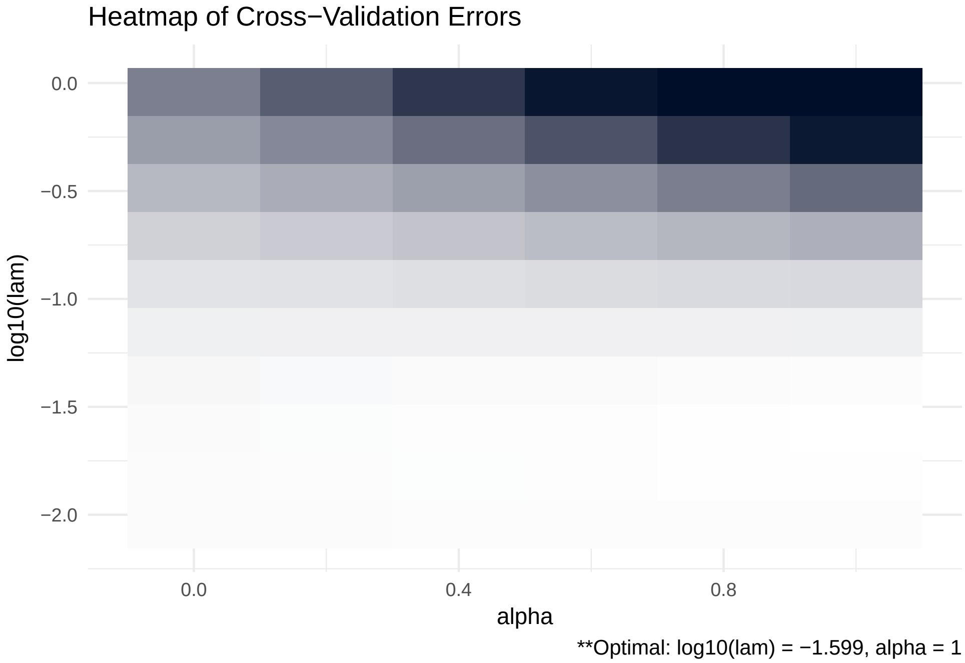
```
##
## Call: ADMMsigma(X = X, alpha = 0)
##
## Iterations: 65
##
## Tuning parameters:
##       log10(lam)  alpha
## [1,]      -1.821      0
##
## Log-likelihood: -99.19746
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  2.18979 -1.31533  0.04515 -0.04090  0.23511
## [2,] -1.31533  2.90019 -1.37049 -0.22633  0.17808
## [3,]  0.04515 -1.37049  2.89435 -1.07647 -0.17369
## [4,] -0.04090 -0.22633 -1.07647  2.55026 -1.22786
## [5,]  0.23511  0.17808 -0.17369 -1.22786  1.95495
```

```
# ridge penalty no ADMM
RIDGEsigma(X, lam = 10^seq(-8, 8, 0.01))
```

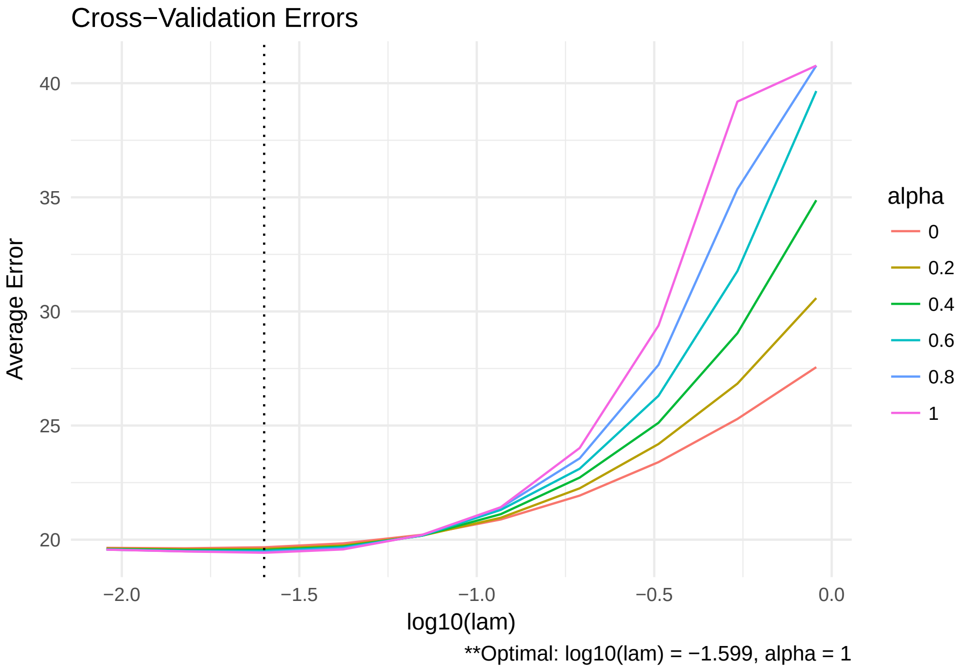
```
##
## Call: RIDGEsigma(X = X, lam = 10^seq(-8, 8, 0.01))
##
## Tuning parameter:
##       log10(lam)    lam
## [1,]       -2.17  0.007
##
## Log-likelihood: -109.18156
##
## Omega:
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  2.15416 -1.31185  0.08499 -0.05571  0.22862
## [2,] -1.31185  2.85605 -1.36677 -0.19650  0.16880
## [3,]  0.08499 -1.36677  2.82606 -1.06325 -0.14946
## [4,] -0.05571 -0.19650 -1.06325  2.50721 -1.21935
## [5,]  0.22862  0.16880 -0.14946 -1.21935  1.92871
```

This package also has the capability to provide heat maps for the cross validation errors. The more bright (white) areas of the heat map pertain to more optimal tuning parameters.

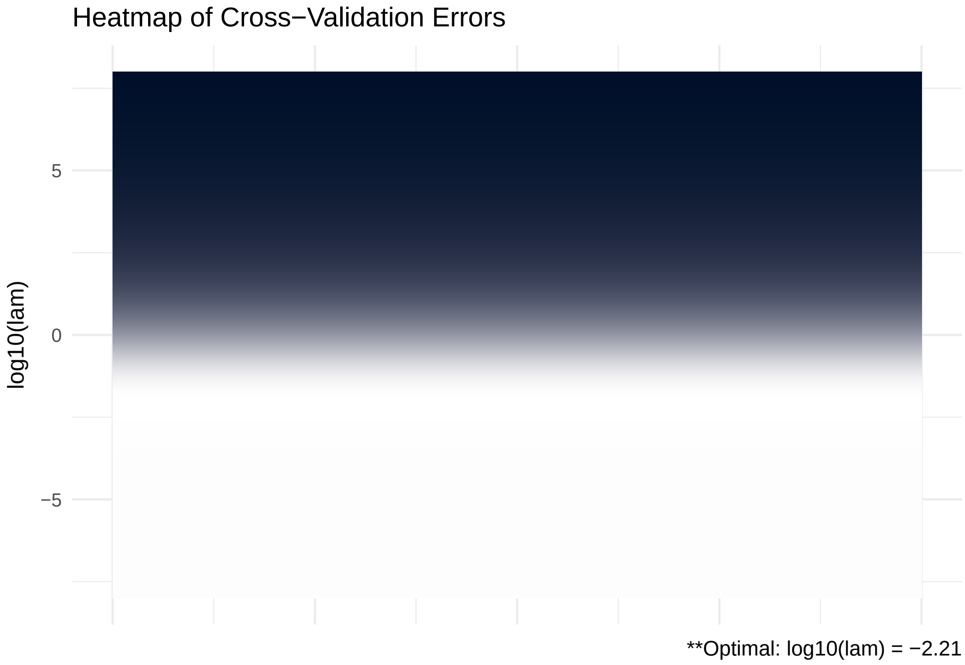
```
# produce CV heat map for ADMMsigma
ADMM = ADMMsigma(X, tol.abs = 1e-08, tol.rel = 1e-08)
plot(ADMM, type = "heatmap")
```



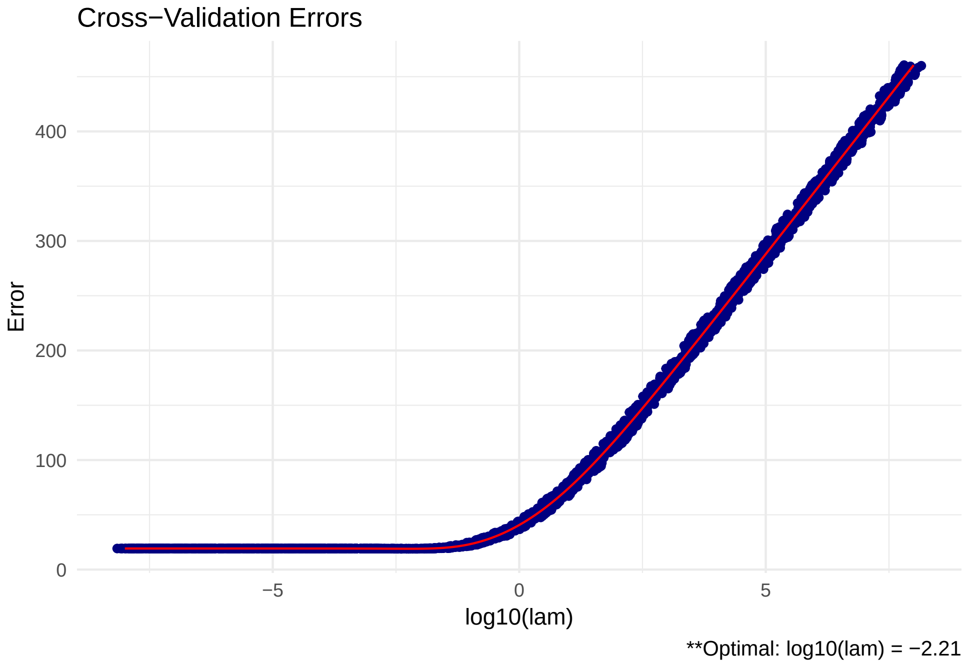
```r
# produce line graph for CV errors for ADMMsigma
plot(ADMM, type = "line")
```



```r
# produce CV heat map for RIDGEsigma
RIDGE = RIDGEsigma(X, lam = 10^seq(-8, 8, 0.01))
plot(RIDGE, type = "heatmap")
```



```r
# produce line graph for CV errors for RIDGEsigma
plot(RIDGE, type = "line")
```



Chapter 4

Details

Suppose we want to solve the following optimization problem:

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

where $x \in \mathbb{R}^n, z \in \mathbb{R}^m, A \in \mathbb{R}^{p \times n}, B \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$. Following Boyd et al. (2011), the optimization problem will be introduced in vector form though we will later consider cases where x and z are matrices. We will also assume f and g are convex functions. Optimization problems like this arise naturally in several statistics and machine learning applications – particularly regularization methods. For instance, we could take f to be the squared error loss, g to be the l_2-norm, c to be equal to zero and A and B to be identity matrices to solve the ridge regression optimization problem.

The *augmented lagrangian* is constructed as follows:

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

where $y \in \mathbb{R}^p$ is the lagrange multiplier and $\rho > 0$ is a scalar. Clearly any minimizer, p^*, under the augmented lagrangian is equivalent to that of the lagrangian since any feasible point (x, z) satisfies the constraint $\rho\left\|Ax + Bz - c\right\|_2^2/2 = 0$.

$$p^* = inf\left\{f(x) + g(z) | Ax + Bz = c\right\}$$

The alternating direction method of multipliers (ADMM) algorithm consists of the following repeated iterations:

$$x^{k+1} := \arg\min_{x} L_{\rho}(x, z^{k}, y^{k}) \tag{4.1}$$

$$z^{k+1} := \arg\min_{z} L_{\rho}(x^{k+1}, z, y^{k}) \tag{4.2}$$

$$y^{k+1} := y^{k} + \rho(Ax^{k+1} + Bz^{k+1} - c) \tag{4.3}$$

A more complete introduction to the algorithm – specifically how it arose out of *dual ascent* and *method of multipliers* – can be found in Boyd et al. (2011).

4.1 ADMM Algorithm

Now consider the case where $X_1, ..., X_n$ are iid $N_p(\mu, \Sigma)$ random variables and we are tasked with estimating the precision matrix, denoted $\Omega \equiv \Sigma^{-1}$. The maximum likelihood estimator for Ω is

$$\hat{\Omega}_{MLE} = \arg\min_{\Omega \in S_{+}^{p}} \left\{ Tr\left(S\Omega\right) - \log\det\left(\Omega\right) \right\}$$

where $S = \sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T/n$ and $\bar{X}$ is the sample mean. By setting the gradient equal to zero, we can show that when the solution exists, $\hat{\Omega}_{MLE} = S^{-1}$.

As in regression settings, we can construct a *penalized* likelihood estimator by adding a penalty term, $P\left(\Omega\right)$, to the likelihood:

$$\hat{\Omega} = \arg\min_{\Omega \in S_{+}^{p}} \left\{ Tr\left(S\Omega\right) - \log\det\left(\Omega\right) + P\left(\Omega\right) \right\}$$

$P\left(\Omega\right)$ is often of the form $P\left(\Omega\right) = \lambda\|\Omega\|_F^2$ or $P\left(\Omega\right) = \|\Omega\|_1$ where $\lambda > 0$, $\left\|\cdot\right\|_F^2$ is the Frobenius norm and we define $\left\|A\right\|_1 = \sum_{i,j}\left|A_{ij}\right|$. These penalties are the ridge and lasso, respectively. We will, instead, take $P\left(\Omega\right) = \lambda\left[\frac{1-\alpha}{2}\left\|\Omega\right|_F^2 + \alpha\left\|\Omega\right\|_1\right]$ so that the full penalized likelihood is

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

where $0 \leq \alpha \leq 1$. This *elastic-net* penalty was explored by Hui Zou and Trevor Hastie (Zou and Hastie, 2005) and is identical to the penalty used in the popular penalized regression package glmnet. Clearly, when $\alpha = 0$ the elastic-net reduces to a ridge-type penalty and when $\alpha = 1$ it reduces to a lasso-type penalty.

By letting f be equal to the non-penalized likelihood and g equal to $P\left(\Omega\right)$, our goal is to minimize the full augmented lagrangian where the constraint is that $\Omega - Z$ is equal to zero:

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

The ADMM algorithm for estimating the penalized precision matrix in this problem is

$$\Omega^{k+1} = \arg\min_{\Omega} \left\{ Tr\left(S\Omega\right) - \log\det\left(\Omega\right) + Tr\left[\Lambda^k\left(\Omega - Z^k\right)\right] + \frac{\rho}{2}\left\|\Omega - Z^k\right\|_F^2 \right\} \tag{4.4}$$

$$Z^{k+1} = \arg\min_{Z} \left\{ \lambda \left[\frac{1-\alpha}{2} \left\|Z\right\|_F^2 + \alpha \left\|Z\right\|_1 \right] + Tr\left[\Lambda^k\left(\Omega^{k+1} - Z\right)\right] + \frac{\rho}{2}\left\|\Omega^{k+1} - Z\right\|_F^2 \right\} \tag{4.5}$$

$$\Lambda^{k+1} = \Lambda^k + \rho\left(\Omega^{k+1} - Z^{k+1}\right) \tag{4.6}$$

4.2 Scaled-Form ADMM

An alternate form of the ADMM algorithm can constructed by scaling the dual variable (Λ^k). Let us define $R^k = \Omega - Z^k$ and $U^k = \Lambda^k/\rho$.

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

Therefore, the condensed-form ADMM algorithm can now be written as

$$\Omega^{k+1} = \arg\min_{\Omega}\left\{Tr\left(S\Omega\right) - \log\det\left(\Omega\right) + \frac{\rho}{2}\left\|\Omega - Z^k + U^k\right\|_F^2\right\} \tag{4.7}$$

$$Z^{k+1} = \arg\min_{Z}\left\{\lambda\left[\frac{1-\alpha}{2}\left\|Z\right\|_F^2 + \alpha\left\|Z\right\|_1\right] + \frac{\rho}{2}\left\|\Omega^{k+1} - Z + U^k\right\|_F^2\right\} \tag{4.8}$$

$$U^{k+1} = U^k + \Omega^{k+1} - Z^{k+1} \tag{4.9}$$

And more generally (in vector form),

$$x^{k+1} = \arg\min_{x}\left\{f(x) + \frac{\rho}{2}\left\|Ax + Bz^k - c + u^k\right\|_2^2\right\} \tag{4.10}$$

$$z^{k+1} = \arg\min_{z}\left\{g(z) + \frac{\rho}{2}\left\|Ax^{k+1} + Bz - c + u^k\right\|_2^2\right\} \tag{4.11}$$

$$u^{k+1} = u^k + Ax^{k+1} + Bz^{k+1} - c \tag{4.12}$$

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 without making further adjustments to U^k. It has been shown in the literature that a dynamic step size can significantly reduce the number of iterations required for convergence.

4.3 Algorithm

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

Initialize Z^0, Λ^0, and ρ. Iterate the following three steps until convergence:

1. Decompose $S + \Lambda^k - \rho Z^k = VQV^T$ via spectral decomposition.

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

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

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

3. Update Λ^{k+1}.

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

4.3.1 Proof of (1):

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

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

Set the gradient equal to zero and decompose $\Omega = VDV^T$ where D is a diagonal matrix with diagonal elements equal to the eigen values of Ω and V is the matrix with corresponding eigen vectors as columns.

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

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 $S + \Lambda^k - \rho Z^k = VQV^T$ then

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

4.3.2 Proof of (2)

$$Z^{k+1} = \arg\min_Z \left\{\lambda\left[\frac{1-\alpha}{2}\left\|Z\right\|_F^2 + \alpha\left\|Z\right\|_1\right] + Tr\left[\Lambda^k\left(\Omega^{k+1} - Z\right)\right] + \frac{\rho}{2}\left\|\Omega^{k+1} - Z\right\|_F^2\right\}$$

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

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

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

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

- If $Z_{ij} > 0$ then

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

- If $Z_{ij} < 0$ then

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

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

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

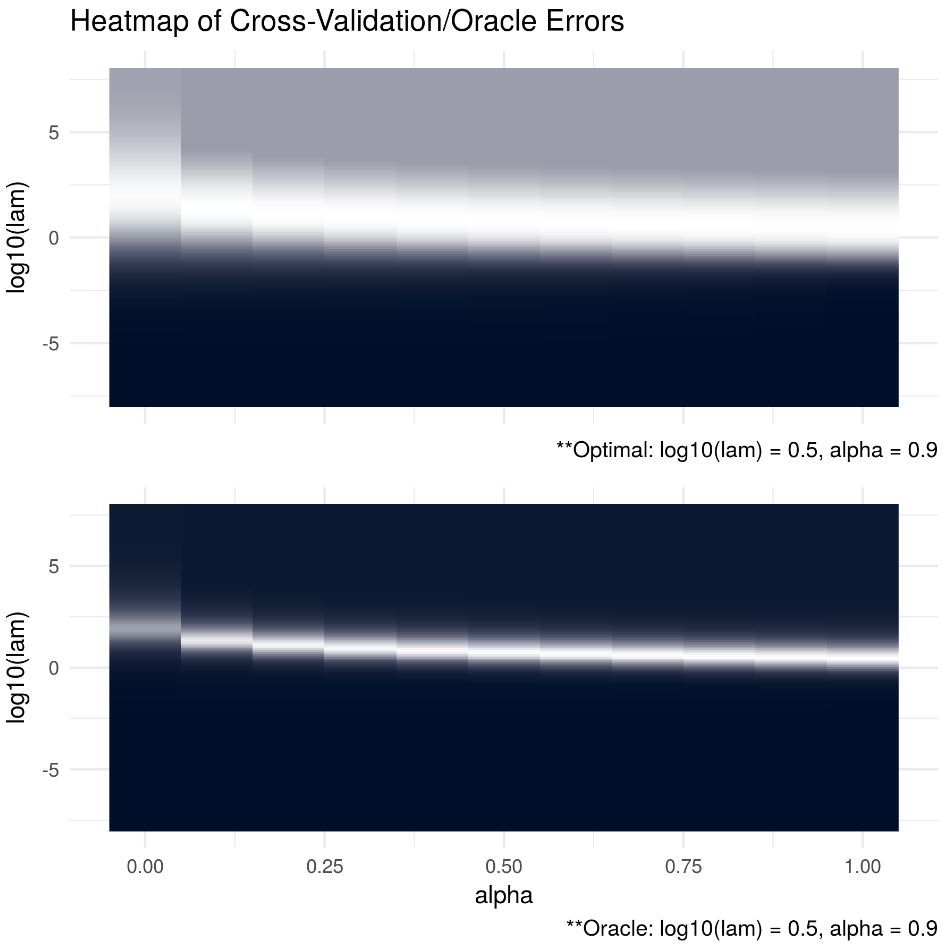
where $Soft$ is the soft-thresholding function.

Chapter 5

Simulations

In the simulations below we generated data from a number of oracle precision matrices with various structures. For each data-generating procedure, the `ADMMsigma()` function was run using 5-fold cross validation. After 20 replications, the cross validation errors were totalled and the optimal tuning parameters were selected (results in the top figure). These results are compared with the Kullback Leibler (KL) losses between the estimates and the oracle precision matrix (bottom figure). We can see below that our cross validation procedure choosing tuning parameters close to the optimal parameters.

5.1 Compound Symmetric: P = 100, N = 50



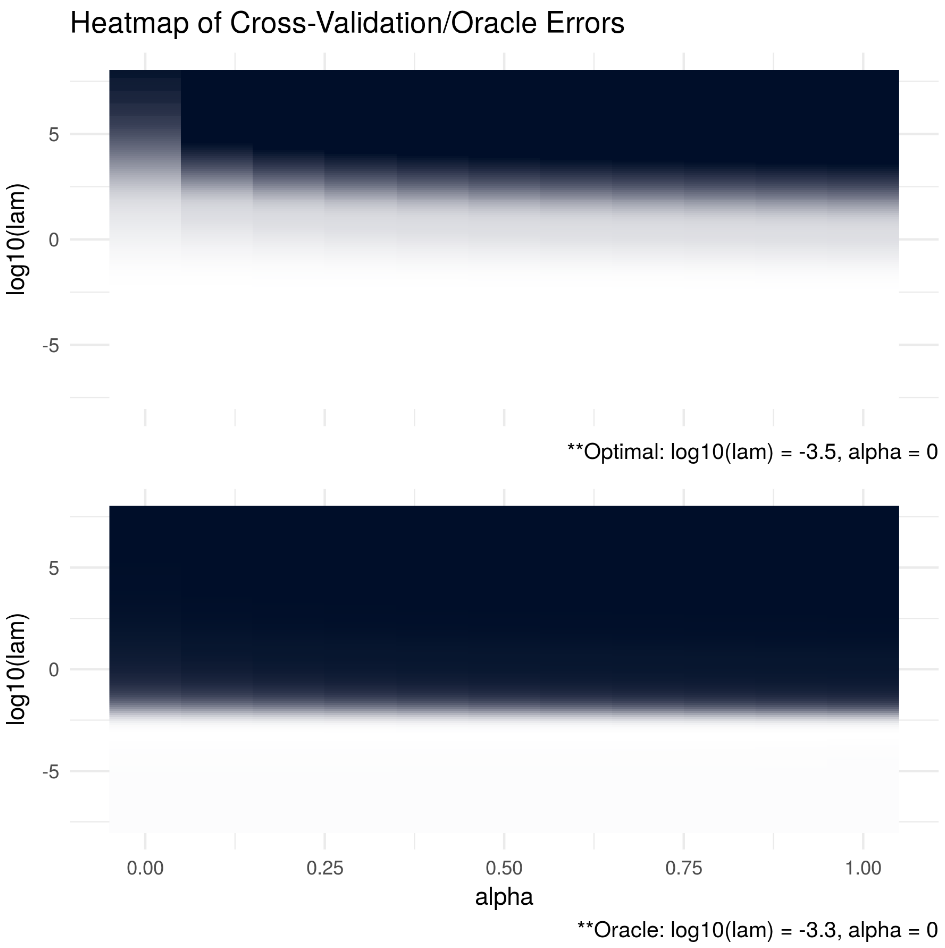
```r
# oracle precision matrix
Omega = matrix(0.9, ncol = 100, nrow = 100)
diag(Omega = 1)

# generate covariance matrix
S = qr.solve(Omega)

# generate data
Z = matrix(rnorm(100 * 50), nrow = 50, ncol = 100)
```

```r
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5) %*% t(out$vectors)
X = Z %*% S.sqrt
```

5.2 Compound Symmetric: P = 10, N = 1000



Heatmap of Cross-Validation/Oracle Errors

**Optimal: log10(lam) = -3.5, alpha = 0

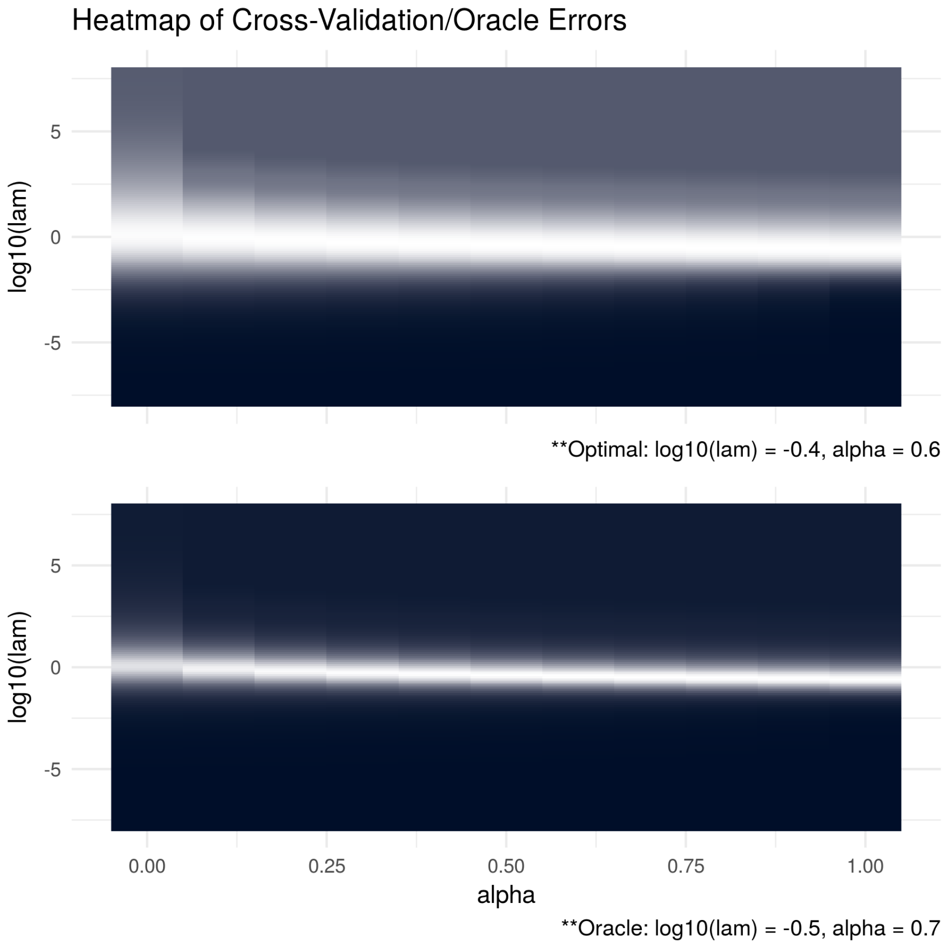
**Oracle: log10(lam) = -3.3, alpha = 0

```r
# oracle precision matrix
Omega = matrix(0.9, ncol = 10, nrow = 10)
diag(Omega = 1)
```

```
# generate covariance matrix
S = qr.solve(Omega)

# generate data
Z = matrix(rnorm(10 * 1000), nrow = 1000, ncol = 10)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5) %*% t(out$vectors)
X = Z %*% S.sqrt
```

5.3 Dense: P = 100, N = 50



Heatmap of Cross-Validation/Oracle Errors

**Optimal: log10(lam) = -0.4, alpha = 0.6

**Oracle: log10(lam) = -0.5, alpha = 0.7

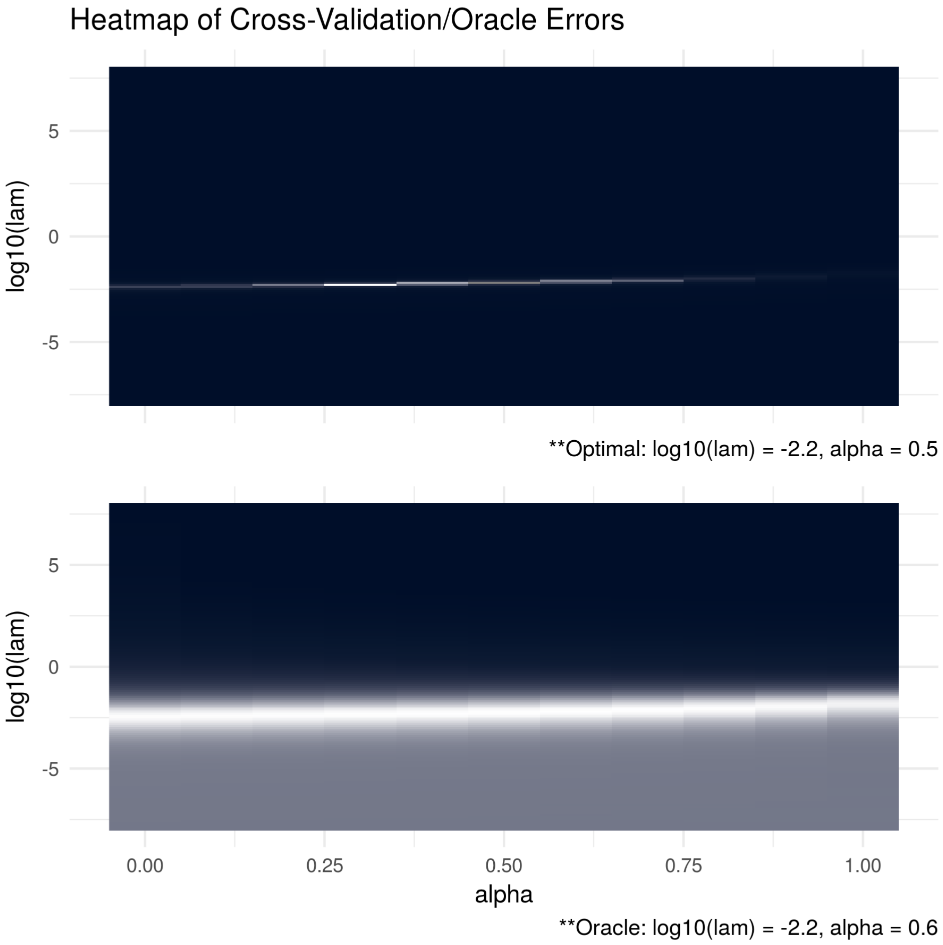
```r
# generate eigen values
eigen = c(rep(1000, 5, rep(1, 100 - 5)))

# randomly generate orthogonal basis (via QR)
Q = matrix(rnorm(100*100), nrow = 100, ncol = 100) %>% qr %>% qr.Q

# generate covariance matrix
S = Q %*% diag(eigen) %*% t(Q)
```

```
# generate data
Z = matrix(rnorm(100*50), nrow = 50, ncol = 100)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5) %*% t(out$vectors)
X = Z %*% S.sqrt
```

5.4 Dense: P = 10, N = 50


Heatmap of Cross-Validation/Oracle Errors

**Optimal: log10(lam) = -2.2, alpha = 0.5

**Oracle: log10(lam) = -2.2, alpha = 0.6

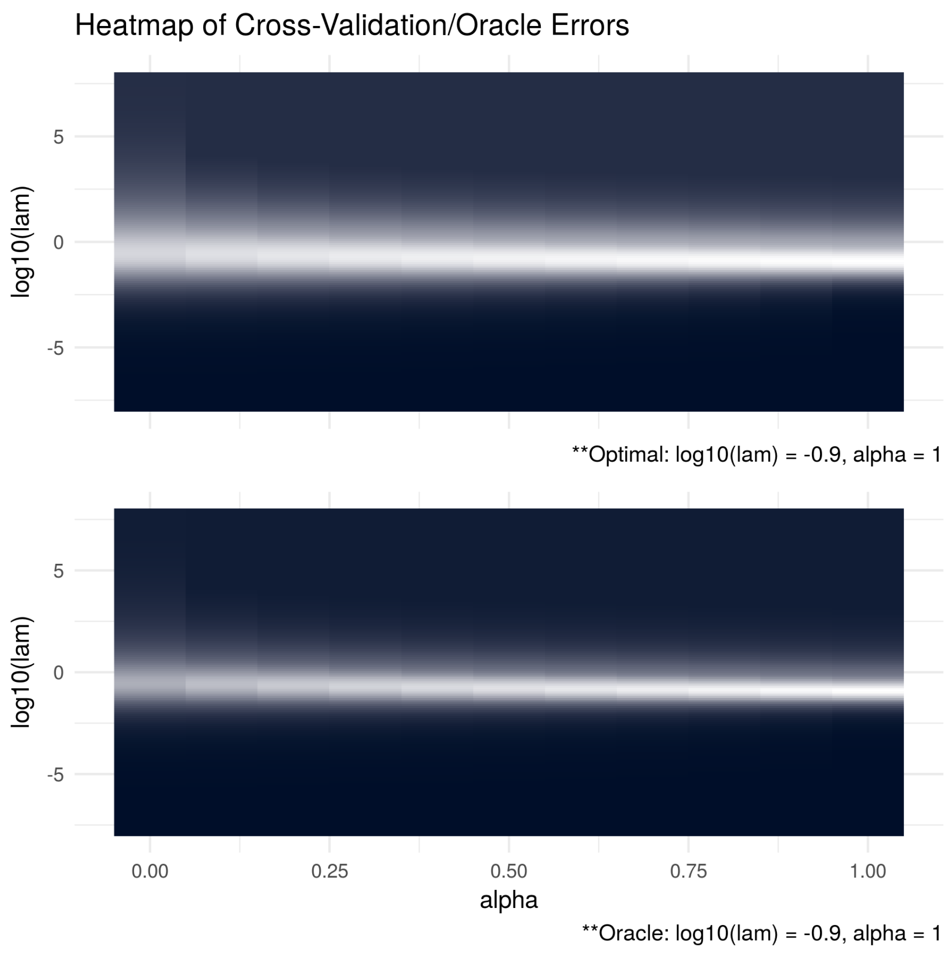
```r
# generate eigen values
eigen = c(rep(1000, 5, rep(1, 10 - 5)))

# randomly generate orthogonal basis (via QR)
Q = matrix(rnorm(10*10), nrow = 10, ncol = 10) %>% qr %>% qr.Q

# generate covariance matrix
S = Q %*% diag(eigen) %*% t(Q)

# generate data
Z = matrix(rnorm(10*50), nrow = 50, ncol = 10)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5) %*% t(out$vectors)
X = Z %*% S.sqrt
```

5.5 Tridiagonal: P = 100, N = 50



Heatmap of Cross-Validation/Oracle Errors

**Optimal: log10(lam) = -0.9, alpha = 1

**Oracle: log10(lam) = -0.9, alpha = 1

```
# generate covariance matrix
# (can confirm inverse is tri-diagonal)
S = matrix(0, nrow = 100, ncol = 100)
for (i in 1:100){
  for (j in 1:100){
    S[i, j] = 0.7^abs(i - j)
  }
}
```

```r
# generate data
Z = matrix(rnorm(10*50), nrow = 50, ncol = 10)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5) %*% t(out$vectors)
X = Z %*% S.sqrt
```

Chapter 6

Benchmark

Below we benchmark the various functions contained in `ADMMsigma`. We can see that `ADMMsigma` (at the default tolerance) offers comparable computation time to the popular `glasso` R package.

6.0.1 Computer Specs:

- MacBook Pro (Late 2016)
- Processor: 2.9 GHz Intel Core i5
- Memory: 8GB 2133 MHz
- Graphics: Intel Iris Graphics 550

```r
library(ADMMsigma)
library(microbenchmark)

# generate data from tri-diagonal (sparse) matrix compute covariance matrix
# (can confirm inverse is tri-diagonal)
S = matrix(0, nrow = 100, ncol = 100)

for (i in 1:100) {
    for (j in 1:100) {
        S[i, j] = 0.7^(abs(i - j))
    }
}
```

```
# generate 1000 x 100 matrix with rows drawn from iid N_p(0, S)
set.seed(123)
Z = matrix(rnorm(1000 * 100), nrow = 1000, ncol = 100)
out = eigen(S, symmetric = TRUE)
S.sqrt = out$vectors %*% diag(out$values^0.5) %*% t(out$vectors)
X = Z %*% S.sqrt


# glasso (for comparison)
microbenchmark(glasso::glasso(s = S, rho = 0.1))
```

```
## Unit: milliseconds
##                               expr      min       lq     mean   median
##  glasso::glasso(s = S, rho = 0.1) 49.46673 53.37757 58.35196 55.90935
##        uq      max neval
##  60.77517 92.68082   100
```

```
# benchmark ADMMsigma - default tolerance
microbenchmark(ADMMsigma(S = S, lam = 0.1, alpha = 1, tol.abs = 1e-04, tol.rel =
    trace = "none"))
```

```
## Unit: milliseconds
##
##  ADMMsigma(S = S, lam = 0.1, alpha = 1, tol.abs = 1e-04, tol.rel = 1e-04,
##      min       lq     mean   median       uq      max neval
##  77.2134 83.51351 89.71565 85.37738 94.37814 126.0049   100
```

```
# benchmark ADMMsigma - tolerance 1e-8
microbenchmark(ADMMsigma(S = S, lam = 0.1, alpha = 1, tol.abs = 1e-08, tol.rel =
    trace = "none"))
```

```
## Unit: milliseconds
##
##  ADMMsigma(S = S, lam = 0.1, alpha = 1, tol.abs = 1e-08, tol.rel = 1e-08,
##       min       lq     mean   median       uq      max neval
##  258.5389 261.5402 272.3741 267.8829 274.7276 353.4726   100
```

```
# benchmark ADMMsigma CV - default parameter grid
microbenchmark(ADMMsigma(X, trace = "none"), times = 5)
```

```
## Unit: seconds
##                           expr      min       lq     mean   median       uq
##  ADMMsigma(X, trace = "none") 8.338241 8.341611 8.536446 8.472933 8.515822
##       max neval
##  9.013621     5
```

```
# benchmark ADMMsigma parallel CV
microbenchmark(ADMMsigma(X, cores = 3, trace = "none"), times = 5)
```

```
## Unit: seconds
##                                     expr      min       lq     mean
##  ADMMsigma(X, cores = 3, trace = "none") 9.285137 9.315978 9.470666
##   median       uq      max neval
##  9.40943 9.426579 9.916207     5
```

```
# benchmark ADMMsigma CV - likelihood convergence criteria
microbenchmark(ADMMsigma(X, crit = "loglik", trace = "none"), times = 5)
```

```
## Unit: seconds
##                                           expr      min       lq     mean
##  ADMMsigma(X, crit = "loglik", trace = "none") 7.028816 7.121599 7.185927
##    median       uq      max neval
##  7.181803 7.277249 7.320166     5
```

```
# benchmark RIDGEsigma CV
microbenchmark(RIDGEsigma(X, lam = 10^seq(-8, 8, 0.01), trace = "none"), times = 5)
```

```
## Unit: seconds
##                                                      expr      min
##  RIDGEsigma(X, lam = 10^seq(-8, 8, 0.01), trace = "none") 12.22374
##        lq     mean   median       uq      max neval
##  12.37326 12.93705 13.01892 13.04356 14.02577     5
```

Chapter 7

Penalized Regression Estimators

This document explores a number of regression estimators subject to various assumptions. We will assume n samples of

$$Y_i = \mu_y + \beta^T (X_i - \mu_x) + \epsilon_i$$

where $\epsilon_i \sim N_r\left(0, \Sigma_{y|x}\right)$ and $X_i \sim N_p\left(0, \Sigma_x\right)$ so that by taking $\theta = \left(\mu_y, \mu_x, vec\left(\beta\right), vec\left(\Sigma_{y|x}^{-1}\right), vec\left(\Sigma_{xx}^{-1}\right)\right)^T$ the joint log-likelihood is of the following form:

$$\begin{aligned} l(\theta) = \frac{nr}{2} log(2\pi) - \frac{n}{2} log\left|\Sigma_{y|x}^{-1}\right| + \frac{1}{2}\sum_{i=1}^{n}\left(Y_i - \mu_y - \beta^T\left(X_i - \mu_x\right)\right)^T \Sigma_{y|x}^{-1}\left(Y_i - \mu_y - \beta^T\left(X_i - \mu_x\right)\right) \\ + \frac{np}{2} log(2\pi) - \frac{n}{2} log\left|\Sigma_{xx}^{-1}\right| + \frac{1}{2}\sum_{i=1}^{n}\left(X_i - \mu_x\right)^T \Sigma_{xx}^{1}\left(X_i - \mu_x\right) \end{aligned}$$

For convenience, we will later denote $\mathbb{X}$ as the $n \times p$ matrix with rows $X_i - \bar{X}$ where $\bar{X} = \sum_{i=1}^{n} X_i/n$ and $\mathbb{Y}$ as the $n \times r$ matrix with rows $Y_i - \bar{Y}$ and $\bar{Y}$ defined similarly. Note that $\bar{X}$ and $\bar{Y}$ are the maximum likelihood esitmators of μ_x and μ_y, respectively.

7.1 Formulas

Below we outline a few important formulas that will be used throughout this report:

1. Forward regression coefficient: $\beta = \Sigma_{xx}^{-1}\Sigma_{xy} = \Sigma_{x|y}^{-1}\alpha^T\left(\Sigma_{yy}^{-1} + \alpha\Sigma_{x|y}^{-1}\alpha^T\right)^{-1}$

2. Inverse regression coefficient: $\alpha = \Sigma_{yy}^{-1}\Sigma_{xy}^T$

3. $\Sigma_{y|x} = \Sigma_{yy} - \beta^T\Sigma_{xx}\beta = \Sigma_{yy} - \Sigma_{xy}^T\Sigma_{xx}^{-1}\Sigma_{xy}$

4. $\Sigma_{y|x}^{-1} = \Sigma_{yy}^{-1} + \Sigma_{y|x}^{-1}\Sigma_{xy}^T\left(\Sigma_{yy} + \Sigma_{xy}\Sigma_{y|x}^{-1}\Sigma_{xy}^T\right)^{-1}\Sigma_{xy}\Sigma_{y|x}^{-1} = \Sigma_{yy}^{-1} + \Sigma_{y|x}^{-1}\beta\left(\Sigma_{xx}^{-1} + \beta\Sigma_{y|x}^{-1}\beta^T\right)^{-1}\beta\Sigma_{y|x}^{-1}$

5. $\Sigma_{x|y} = \Sigma_{xx} - \alpha^T\Sigma_{yy}\alpha = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{xy}^T$

6. $\Sigma_{x|y}^{-1} = \Sigma_{xx}^{-1} + \Sigma_{x|y}^{-1}\Sigma_{xy}\left(\Sigma_{yy} + \Sigma_{xy}^T\Sigma_{x|y}^{-1}\Sigma_{xy}\right)^{-1}\Sigma_{xy}^T\Sigma_{x|y}^{-1} = \Sigma_{xx}^{-1} + \Sigma_{x|y}^{-1}\alpha\left(\Sigma_{yy}^{-1} + \alpha\Sigma_{x|y}^{-1}\alpha^T\right)^{-1}\alpha\Sigma_{x|y}^{-1}$

7. $\Sigma_{yy}^{-1} = \Sigma_{y|x}^{-1} - \Sigma_{y|x}^{-1}\Sigma_{xy}^T\left(\Sigma_{xx} + \Sigma_{xy}\Sigma_{y|x}^{-1}\Sigma_{xy}^T\right)^{-1}\Sigma_{xy}\Sigma_{y|x}^{-1} = \Sigma_{y|x}^{-1} - \Sigma_{y|x}^{-1}\beta\left(\Sigma_{xx}^{-1} + \beta\Sigma_{y|x}^{-1}\beta^T\right)^{-1}\beta\Sigma_{y|x}^{-1}$

8. $\Sigma_{xx}^{-1} = \Sigma_{x|y}^{-1} - \Sigma_{x|y}^{-1}\Sigma_{xy}\left(\Sigma_{yy} + \Sigma_{xy}^T\Sigma_{x|y}^{-1}\Sigma_{xy}\right)^{-1}\Sigma_{xy}^T\Sigma_{x|y}^{-1} = \Sigma_{x|y}^{-1} - \Sigma_{x|y}^{-1}\alpha\left(\Sigma_{yy}^{-1} + \alpha\Sigma_{x|y}^{-1}\alpha^T\right)^{-1}\alpha\Sigma_{x|y}^{-1}$

7.2 Estimators

7.2.1 Assume ridge β, optimize in-sample prediction error

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2} \|\mathbb{Y} - \mathbb{X}\beta\|_F^2 + \frac{\lambda}{2} \|\beta\|_F^2 \right\}$$

$$\Rightarrow -\mathbb{X}^T \left(\mathbb{Y} - \mathbb{X}\hat{\beta} \right) + \lambda\hat{\beta} = 0$$

$$\Rightarrow \hat{\beta} = \left(\mathbb{X}^T\mathbb{X} + \lambda I_p \right)^{-1} \mathbb{X}^T\mathbb{Y} = \mathbb{X}^T \left(\mathbb{X}\mathbb{X}^T + \lambda I_n \right)^{-1} \mathbb{Y}$$

7.2.2 Assume ridge β, optimize the likelihood for β

$$\hat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{n} Tr \left[(\mathbb{Y} - \mathbb{X}\beta)^T (\mathbb{Y} - \mathbb{X}\beta) \Sigma_{y|x}^{-1} \right] - \frac{1}{2} log \left| \Sigma_{y|x}^{-1} \right| + \frac{\lambda}{n} \|\beta\|_F^2 \right\}$$

$$\Rightarrow -\frac{2}{n}\mathbb{X}^T \left(\mathbb{Y} - \mathbb{X}\hat{\beta} \right) \hat{\Sigma}_{y|x}^{-1} + \frac{2}{n}\lambda\hat{\beta} = 0 \Rightarrow \left(\hat{\Sigma}_{y|x}^{-1} \otimes \mathbb{X}^T\mathbb{X} \right) vec\left(\hat{\beta}\right) + \lambda vec\left(\hat{\beta}\right) = vec\left(\mathbb{X}^T\mathbb{Y}\hat{\Sigma}_{y|x}^{-1} \right)$$

$$\Rightarrow vec\left(\hat{\beta}\right) = \left(\hat{\Sigma}_{y|x}^{-1} \otimes \mathbb{X}^T\mathbb{X} + \lambda I_{pr} \right)^{-1} vec\left(\mathbb{X}^T\mathbb{Y}\hat{\Sigma}_{y|x}^{-1} \right)$$

where $\hat{\Sigma}_{y|x} = (\mathbb{Y} - \mathbb{X}\hat{\beta})^T(\mathbb{Y} - \mathbb{X}\hat{\beta})/n$. In order to solve for $\hat{\beta}$, we need to iterate between $\hat{\beta}$ and $\hat{\Sigma}_{y|x}^{-1}$ until convergence.

Note that if $r = 1$ then σ_y^2 can be absorbed into $\tilde{\lambda}$ and

$$\hat{\beta} = \left(\mathbb{X}^T\mathbb{X} + \tilde{\lambda} I_p \right)^{-1} \mathbb{X}^T\mathbb{Y} = \mathbb{X}^T \left(\mathbb{X}\mathbb{X}^T + \tilde{\lambda}_n I_n \right)^{-1} \mathbb{Y}$$

7.2.3 Assume ridge Σ_{xx}^{-1}, optimize in-sample prediction error

$$\hat{\Sigma}_{xx}^{-1} = \arg\min_{\Sigma_{xx}^{-1}} \left\{ \frac{1}{2} \left\| \mathbb{Y} - \mathbb{X}\beta \right\|_F^2 + \frac{\lambda}{2} \left\| \Sigma_{xx}^{-1} \right\|_F^2 \right\}$$

$$= \arg\min_{\Sigma_{xx}^{-1}} \left\{ \frac{1}{2} \left\| \mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right\|_F^2 + \frac{\lambda}{2} \left\| \Sigma_{xx}^{-1} \right\|_F^2 \right\}$$

$$\nabla_{\Sigma_{xx}^{-1}} \left\{ \frac{1}{2} Tr \left[\left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right)^T \left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right) \right] + \frac{\lambda}{2} Tr \left[\Sigma_{xx}^{-1}\Sigma_{xx}^{-1} \right] \right\}$$

$$= -\mathbb{X}^T\mathbb{Y}\Sigma_{xy}^T + \mathbb{X}^T\mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy}\Sigma_{xy}^T + \lambda\Sigma_{xx}^{-1}$$

$$\Rightarrow vec\left(\hat{\Sigma}_{xx}^{-1} \right) = \left(\hat{\Sigma}_{xy}\hat{\Sigma}_{xy}^T \otimes \mathbb{X}^T\mathbb{X} + \lambda I_{pp} \right)^{-1} vec\left(\mathbb{X}^T\mathbb{Y}\hat{\Sigma}_{xy}^T \right)$$

If we use the sample estimate $\hat{\Sigma}_{xy} = \mathbb{X}^T\mathbb{Y}/n$ then

$$vec\left(\hat{\Sigma}_{xx}^{-1} \right) = \left(\mathbb{X}^T\mathbb{Y}\mathbb{Y}^T\mathbb{X} \otimes \frac{1}{n}\mathbb{X}^T\mathbb{X} + \tilde{\lambda} I_{pp} \right)^{-1} vec\left(\mathbb{X}^T\mathbb{Y}\mathbb{Y}^T\mathbb{X} \right)$$

so that $\hat{\beta} = \hat{\Sigma}_{xx}^{-1}\hat{\Sigma}_{xy} = \hat{\Sigma}_{xx}^{-1}\mathbb{X}^T\mathbb{Y}/n$.

7.2.4 Assume ridge Σ_{xx}^{-1}, optimize likelihood for β

$$\hat{\Sigma}_{xx}^{-1} = \arg\min_{\Sigma_{xx}^{-1}} \left\{ \frac{1}{n} Tr \left[\left(\mathbb{Y} - \mathbb{X}\beta \right)^T \left(\mathbb{Y} - \mathbb{X}\beta \right) \Sigma_{y|x}^{-1} \right] - \frac{1}{2} log \left| \Sigma_{y|x}^{-1} \right| + \frac{\lambda}{n} \left\| \Sigma_{xx}^{-1} \right\|_F^2 \right\}$$

$$= \arg\min_{\Sigma_{xx}^{-1}} \left\{ \frac{1}{n} Tr \left[\left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right)^T \left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right) \Sigma_{y|x}^{-1} \right] - \frac{1}{2} log \left| \Sigma_{y|x}^{-1} \right| + \frac{\lambda}{n} \left\| \Sigma_{xx}^{-1} \right\|_F^2 \right\}$$

$$\nabla_{\Sigma_{xx}^{-1}} \left\{ \frac{1}{n} Tr \left[\left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right)^T \left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right) \Sigma_{y|x}^{-1} \right] + \frac{\lambda}{n} Tr \left[\Sigma_{xx}^{-1}\Sigma_{xx}^{-1} \right] \right\}$$

$$= -\frac{2}{n}\mathbb{X}^T\mathbb{Y}\Sigma_{y|x}^{-1}\Sigma_{xy}^T + \frac{2}{n}\mathbb{X}^T\mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy}\Sigma_{y|x}^{-1}\Sigma_{xy}^T + \frac{2\lambda}{n}\Sigma_{xx}^{-1}$$

$$\Rightarrow vec\left(\hat{\Sigma}_{xx}^{-1}\right) = \left(\hat{\Sigma}_{xy}\hat{\Sigma}_{y|x}^{-1}\hat{\Sigma}_{xy}^T \otimes \mathbb{X}^T\mathbb{X} + \lambda I_{pp} \right)^{-1} vec\left(\mathbb{X}^T\mathbb{Y}\hat{\Sigma}_{y|x}^{-1}\hat{\Sigma}_{xy}^T \right)$$

If we use the sample estimate $\hat{\Sigma}_{xy} = \mathbb{X}^T\mathbb{Y}/n$ then

$$vec\left(\hat{\Sigma}_{xx}^{-1}\right) = \left(\mathbb{X}^T\mathbb{Y}\hat{\Sigma}_{y|x}^{-1}\mathbb{Y}^T\mathbb{X} \otimes \frac{1}{n}\mathbb{X}^T\mathbb{X} + \tilde{\lambda} I_{pp} \right)^{-1} vec\left(\mathbb{X}^T\mathbb{Y}\hat{\Sigma}_{y|x}^{-1}\mathbb{Y}^T\mathbb{X} \right)$$

where $\hat{\Sigma}_{y|x} = (\mathbb{Y} - \mathbb{X}\hat{\beta})^T(\mathbb{Y} - \mathbb{X}\hat{\beta})/n$ so that in order to solve for $\hat{\beta} = \hat{\Sigma}_{xx}^{-1}\hat{\Sigma}_{xy} = \hat{\Sigma}_{xx}^{-1}\mathbb{X}^T\mathbb{Y}/n$, we need to iterate between $\hat{\beta}$ and $\hat{\Sigma}_{y|x}^{-1}$ until convergence.

7.2.5 Assume ridge Σ_{xx}^{-1}, optimize likelihood for Σ_{xx}^{-1}

$$\hat{\Sigma}_{xx}^{-1} = \arg\min_{\Sigma_{xx}^{-1}} \left\{ Tr\left(S_{xx}\Sigma_{xx}^{-1}\right) - log\left|\Sigma_{xx}^{-1}\right| + \frac{\lambda}{2}\left\|\Sigma_{xx}^{-1}\right\|_F^2 \right\}$$

$$\nabla_{\Sigma_{xx}^{-1}} \left\{ Tr\left(S_{xx}\Sigma_{xx}^{-1}\right) - \log\det\left(\Sigma_{xx}^{-1}\right) + \frac{\lambda}{2}\left\|\Sigma_{xx}^{-1}\right\|_F^2 \right\} = S_{xx} - \Sigma_{xx} + \lambda\Sigma_{xx}^{-1}$$

where $S_{xx} = \mathbb{X}^T\mathbb{X}/n$. Set the gradient equal to zero and decompose $\Sigma_{xx}^{-1} = VDV^T$ where D is a diagonal matrix with diagonal elements equal to the eigen values of Σ_{xx}^{-1} and V is the matrix with corresponding eigen vectors as columns.

$$S_{xx} = \Sigma_{xx} - \lambda\Sigma_{xx}^{-1} = VD^{-1}V^T - \lambda VDV^T = V\left(D^{-1} - \lambda D\right)V^T$$

This equivalence implies that

$$\phi_j\left(S_{xx}\right) = \frac{1}{\phi_j(\Sigma_{xx}^{-1})} - \lambda\phi_j\left(\Sigma_{xx}^{-1}\right)$$

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

$$\Rightarrow \lambda\phi_j^2\left(\Sigma_{xx}^{-1}\right) + \phi_j\left(S_{xx}\right)\phi_j\left(\Sigma_{xx}^{-1}\right) - 1 = 0$$

$$\Rightarrow \phi_j\left(\Sigma_{xx}^{-1}\right) = \frac{-\phi_j\left(S_{xx}\right) \pm \sqrt{\phi_j^2\left(S_{xx}\right) + 4\lambda}}{2\lambda}$$

In summary, if we decompose $S_{xx} = VQV^T$ then

$$\hat{\Sigma}_{xx}^{-1} = \frac{1}{2\lambda}V\left[-Q + \left(Q^2 + 4\lambda I_p\right)^{1/2}\right]V^T$$

so that $\hat{\beta} = \hat{\Sigma}_{xx}^{-1}\Sigma_{xy} = \hat{\Sigma}_{xx}^{-1}\mathbb{X}^T\mathbb{Y}/n$.

7.2.6 Assume ridge Σ_{xy}, optimize in-sample prediction error

$$\hat{\Sigma}_{xy} = \arg\min_{\Sigma_{xy}}\left\{\frac{1}{2}\left\|\mathbb{Y} - \mathbb{X}\beta\right\|_F^2 + \frac{\lambda}{2}\left\|\Sigma_{xy}\right\|_F^2\right\}$$

$$= \arg\min_{\Sigma_{xy}}\left\{\frac{1}{2}\left\|\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy}\right\|_F^2 + \frac{\lambda}{2}\left\|\Sigma_{xy}\right\|_F^2\right\}$$

$$\nabla_{\Sigma_{xy}} \left\{ \frac{1}{2} Tr \left[\left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right)^T \left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right) \right] + \frac{\lambda}{2} Tr \left[\Sigma_{xy}\Sigma_{xy} \right] \right\}$$

$$= -\Sigma_{xx}^{-1}\mathbb{X}^T\mathbb{Y} + \Sigma_{xx}^{-1}\mathbb{X}^T\mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} + \lambda\Sigma_{xy}$$

$$\Rightarrow \hat{\Sigma}_{xy} = \left(\hat{\Sigma}_{xx}^{-1}\mathbb{X}^T\mathbb{X}\hat{\Sigma}_{xx}^{-1} + \lambda I_p \right)^{-1} \hat{\Sigma}_{xx}^{-1}\mathbb{X}^T\mathbb{Y}$$

If we use the sample estimate $\hat{\Sigma}_{xx} = \mathbb{X}^T\mathbb{X}/n$ (assuming $\mathbb{X}^T\mathbb{X}$ is positive definite) then

$$\hat{\Sigma}_{xy} = \left[\left(\mathbb{X}^T\mathbb{X} \right)^{-1} + \tilde{\lambda} I_p \right]^{-1} \left(\mathbb{X}^T\mathbb{X} \right)^{-1} \mathbb{X}^T\mathbb{Y}/n = \left(I_p + \tilde{\lambda}\mathbb{X}^T\mathbb{X} \right)^{-1} \mathbb{X}^T\mathbb{Y}/n$$

so that

$$\hat{\beta} = \left(\mathbb{X}^T\mathbb{X} \right)^{-1} \left[\left(\mathbb{X}^T\mathbb{X} \right)^{-1} + \tilde{\lambda} I_p \right]^{-1} \left(\mathbb{X}^T\mathbb{X} \right)^{-1} \mathbb{X}^T\mathbb{Y}$$

$$= \left[\mathbb{X}^T\mathbb{X} \left(I_p + \tilde{\lambda}\mathbb{X}^T\mathbb{X} \right) \right]^{-1} \mathbb{X}^T\mathbb{X} = \left(\mathbb{X}^T\mathbb{X} + \tilde{\lambda}\mathbb{X}^T\mathbb{X}\mathbb{X}^T\mathbb{X} \right)^{-1} \mathbb{X}^T\mathbb{Y}$$

7.2.7 Assume ridge Σ_{xy}, optimize likelihood for β

$$\hat{\Sigma}_{xy} = \arg\min_{\Sigma_{xy}} \left\{ \frac{1}{n} Tr \left[\left(\mathbb{Y} - \mathbb{X}\beta \right)^T \left(\mathbb{Y} - \mathbb{X}\beta \right) \Sigma_{y|x}^{-1} \right] - \frac{1}{2} log \left| \Sigma_{y|x}^{-1} \right| + \frac{\lambda}{n} \left\| \Sigma_{xy} \right\|_F^2 \right\}$$

$$= \arg\min_{\Sigma_{xy}} \left\{ \frac{1}{n} Tr \left[\left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right)^T \left(\mathbb{Y} - \mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy} \right) \Sigma_{y|x}^{-1} \right] - \frac{1}{2} log \left| \Sigma_{y|x}^{-1} \right| + \frac{\lambda}{n} \left\| \Sigma_{xy} \right\|_F^2 \right\}$$

$$\nabla_{\Sigma_{xy}}\left\{\frac{1}{n}Tr\left[\left(\mathbb{Y}-\mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy}\right)^T\left(\mathbb{Y}-\mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy}\right)\Sigma_{y|x}^{-1}\right]+\frac{\lambda}{n}Tr\left[\Sigma_{xy}^T\Sigma_{xy}\right]\right\}$$

$$=-\frac{2}{n}\Sigma_{xx}^{-1}\mathbb{X}^T\mathbb{Y}\Sigma_{y|x}^{-1}+\frac{2}{n}\Sigma_{xx}^{-1}\mathbb{X}^T\mathbb{X}\Sigma_{xx}^{-1}\Sigma_{xy}\Sigma_{y|x}^{-1}+\frac{2\lambda}{n}\Sigma_{xy}$$

If we use the sample estimate $\hat{\Sigma}_{xx}=\mathbb{X}^T\mathbb{X}/n$ (assuming $\mathbb{X}^T\mathbb{X}$ is positive definite) then

$$vec\left(\hat{\Sigma}_{xy}\right)=\left[\hat{\Sigma}_{y|x}^{-1}\otimes n\left(\mathbb{X}^T\mathbb{X}\right)^{-1}+\tilde{\lambda}I_{rp}\right]^{-1}vec\left[\left(\mathbb{X}^T\mathbb{X}\right)^{-1}\mathbb{X}^T\mathbb{Y}\hat{\Sigma}_{y|x}^{-1}\right]$$

$$=\left[\Sigma_{y|x}^{-1}\otimes\left(\tilde{\lambda}\mathbb{X}^T\mathbb{X}+I_p\right)\right]^{-1}vec\left[\mathbb{X}^T\mathbb{Y}\Sigma_{y|x}^{-1}\right]$$

where $\hat{\Sigma}_{y|x}=(\mathbb{Y}-\mathbb{X}\hat{\beta})^T(\mathbb{Y}-\mathbb{X}\hat{\beta})/n$ so that in order to solve for $\hat{\beta}=\hat{\Sigma}_{xx}^{-1}\hat{\Sigma}_{xy}=n\left(\mathbb{X}^T\mathbb{X}\right)^{-1}\hat{\Sigma}_{xy}$, we need to iterate between $\hat{\beta}$ and $\hat{\Sigma}_{y|x}^{-1}$ until convergence.

7.2.8 Assume ridge $\Sigma_{x|y}^{-1}$, optimize likelihood for $\Sigma_{x|y}^{-1}$

$$\hat{\Sigma}_{x|y}^{-1}=\arg\min_{\Sigma_{x|y}^{-1}}\left\{Tr\left(S_{x|y}\Sigma_{x|y}^{-1}\right)-log\left|\Sigma_{x|y}^{-1}\right|+\frac{\lambda}{2}\left\|\Sigma_{x|y}^{-1}\right\|_F^2\right\}$$

$$\nabla_{\Sigma_{x|y}^{-1}}\left\{Tr\left(S_{x|y}\Sigma_{x|y}^{-1}\right)-\log\det\left(\Sigma_{x|y}^{-1}\right)+\frac{\lambda}{2}\left\|\Sigma_{x|y}^{-1}\right\|_F^2\right\}=S_{x|y}-\Sigma_{x|y}+\lambda\Sigma_{x|y}^{-1}$$

where $S_{x|y}=\left(\mathbb{X}-\mathbb{Y}\hat{\alpha}\right)^T\left(\mathbb{X}-\mathbb{Y}\hat{\alpha}\right)/n$ and $\hat{\alpha}=\left(\mathbb{Y}^T\mathbb{Y}\right)^{-1}\mathbb{Y}^T\mathbb{X}$. Following the derivations in section (3.5), if we decompose $S_{x|y}=VQV^T$ then

$$\hat{\Sigma}_{x|y}^{-1}=\frac{1}{2\lambda}V\left[-Q+\left(Q^2+4\lambda I_p\right)^{1/2}\right]V^T$$

Using the Woodbury Identity, we find that

$$\hat{\beta} = \hat{\Sigma}_{x|y}^{-1}\Sigma_{xy}\left(\Sigma_{yy} + \Sigma_{xy}^{T}\hat{\Sigma}_{x|y}^{-1}\Sigma_{xy}\right)^{-1}\Sigma_{yy} = \hat{\Sigma}_{x|y}^{-1}\hat{\alpha}^{T}\left(\Sigma_{yy}^{-1} + \hat{\alpha}\hat{\Sigma}_{x|y}^{-1}\hat{\alpha}\right)^{-1}$$

7.2.9 Assume ridge α, optimize in-sample prediction error for α

$$\hat{\alpha} = \arg\min_{\alpha}\left\{\frac{1}{2}\left\|\mathbb{X} - \mathbb{Y}\alpha\right\|_{F}^{2} + \frac{\lambda}{2}\left\|\alpha\right\|_{F}^{2}\right\}$$

$$\Rightarrow -\mathbb{Y}^{T}\left(\mathbb{X} - \mathbb{Y}\hat{\alpha}\right) + \lambda\hat{\alpha} = 0$$

$$\Rightarrow \hat{\alpha} = \left(\mathbb{Y}^{T}\mathbb{Y} + \lambda I_{r}\right)^{-1}\mathbb{Y}^{T}\mathbb{X} = \mathbb{Y}^{T}\left(\mathbb{Y}\mathbb{Y}^{T} + \lambda I_{n}\right)^{-1}\mathbb{X}$$

Using the Woodbury Identity, we find that

$$\hat{\beta} = \Sigma_{x|y}^{-1}\hat{\alpha}^{T}\left(\Sigma_{yy}^{-1} + \hat{\alpha}\Sigma_{x|y}^{-1}\hat{\alpha}^{T}\right)^{-1}$$

7.2.10 Assume ridge α, optimize likelihood for α

$$\hat{\alpha} = \arg\min_{\alpha}\left\{\frac{1}{n}Tr\left[\left(\mathbb{X} - \mathbb{Y}\alpha\right)^{T}\left(\mathbb{X} - \mathbb{Y}\alpha\right)\Sigma_{x|y}^{-1}\right] - \frac{1}{2}log\left|\Sigma_{x|y}^{-1}\right| + \frac{\lambda}{n}\left\|\alpha\right\|_{F}^{2}\right\}$$

$$\Rightarrow -\frac{2}{n}\mathbb{Y}^{T}\left(\mathbb{X} - \mathbb{Y}\hat{\alpha}\right)\hat{\Sigma}_{x|y}^{-1} + \frac{2}{n}\lambda\hat{\alpha} = 0 \Rightarrow \left(\hat{\Sigma}_{x|y}^{-1} \otimes \mathbb{Y}^{T}\mathbb{Y}\right)vec\left(\hat{\alpha}\right) + \lambda vec\left(\hat{\alpha}\right) = vec\left(\mathbb{Y}^{T}\mathbb{X}\hat{\Sigma}_{x|y}^{-1}\right)$$

$$\Rightarrow vec\left(\hat{\alpha}\right) = \left(\hat{\Sigma}_{x|y}^{-1} \otimes \mathbb{Y}^{T}\mathbb{Y} + \lambda I_{pr}\right)^{-1} vec\left(\mathbb{Y}^{T}\mathbb{X}\hat{\Sigma}_{x|y}^{-1}\right)$$

where $\hat{\Sigma}_{x|y} = (\mathbb{X} - \mathbb{Y}\hat{\alpha})^T(\mathbb{X} - \mathbb{Y}\hat{\alpha})/n$. In order to solve for $\hat{\alpha}$, we need to iterate between $\hat{\alpha}$ and $\hat{\Sigma}_{x|y}^{-1}$ until convergence. Again, using the Woodbury Identity, we find that

$$\hat{\beta} = \hat{\Sigma}_{x|y}^{-1}\hat{\alpha}^T\left(\Sigma_{yy}^{-1} + \hat{\alpha}\hat{\Sigma}_{x|y}^{-1}\hat{\alpha}^T\right)^{-1}$$

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.

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.