Assignment 1

David Fleischer – 260396047

Last Update: 18 January, 2018

DOUBLE CHECK 2 (e) FOR WHEN p > n... noninvertible X^T X...

Question 1

From our definitions of $\tilde{X}$ and $\tilde{Y}$

$$\begin{aligned}
\tilde{X} &= X_{-1} - \mathbf{1}_n \bar{x}^T \\
\tilde{Y} &= Y - \mathbf{1}_n^T \bar{Y},
\end{aligned}$$

we find

$$\begin{aligned}
\hat{\beta}_{-1} &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \, ||\tilde{Y} - \tilde{X}\beta||_2^2 \\
&= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \, ||Y - \mathbf{1}_n \bar{Y} - \left(X_{-1} - \mathbf{1}_n \bar{x}^T\right) \beta_{-1}||_2^2 \\
&= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \, ||Y - X_{-1}\beta_{-1} - \mathbf{1}_n \left(\bar{Y} - \bar{x}^T \beta_{-1}\right)||_2^2 \\
&= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \, ||Y - X_{-1}\beta_{-1} - \mathbf{1}_n \beta_1||_2^2 \quad \text{(by definition of } \beta_1 \text{ above)} \\
&= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \, ||Y - [\mathbf{1}_n, \, X_{-1}] \, [\beta_1, \, \beta_{-1}]||_2^2 \\
&\equiv \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min} \, ||Y - X\beta||_2^2.
\end{aligned}$$

Therefore, if $\hat{\beta} = \left(\hat{\beta}_1, \, \hat{\beta}_{-1}^T\right)^T \in \mathbb{R}^p$ and

$$\hat{\beta}_1 = \bar{Y} - \bar{x}^T \hat{\beta}_{-1},$$

then $\hat{\beta}$ also solves the uncentered problem

$$\hat{\beta} \equiv \left(\hat{\beta}_1, \, \hat{\beta}_{-1}^T\right)^T = \underset{\beta \in \mathbb{R}^p}{\arg\min} \, ||Y - X\beta||_2^2,$$

as desired.

Question 2

Consider the (centered) ridge regression problem of estimating β_* with the ℓ_2 penalized least squares regression coefficients $\hat{\beta}^{(\lambda)} = \left(\hat{\beta}_1^{(\lambda)}, \hat{\beta}_{-1}^{(\lambda)\,T}\right)^T$ defined by

$$\begin{aligned}
\hat{\beta}_{-1}^{(\lambda)} &= \underset{\beta \in \mathbb{R}^{p-1}}{\arg\min}\ \|\tilde{Y} - \tilde{X}\beta\|_2^2 + \lambda\|\beta\|_2^2 \\
\hat{\beta}_1^{(\lambda)} &= \bar{Y} - \bar{x}^T \hat{\beta}_{-1}^{(\lambda)}.
\end{aligned}$$

(a)

We define our objective function $f : \mathbb{R}^p \to \mathbb{R}$ by

$$\begin{aligned}
f(\beta) &= \|\tilde{Y} - \tilde{X}\beta\|_2^2 + \lambda\|\beta\|_2^2 \\
&= \left(\tilde{Y} - \tilde{X}\beta\right)^T \left(\tilde{Y} - \tilde{X}\beta\right)^T + \lambda\beta^T\beta \\
&= \tilde{Y}^T\tilde{Y} - \tilde{Y}^T\tilde{X}\beta - \beta^T\tilde{X}^T\tilde{Y} + \beta^T\tilde{X}^T\tilde{X}\beta + \lambda\beta^T\beta \\
&= \tilde{Y}^T\tilde{Y} - 2\beta^T\tilde{X}^T\tilde{Y} + \beta^T\tilde{X}^T\tilde{X}\beta + \lambda\beta^T\beta.
\end{aligned}$$

Therefore, taking the gradient of our function $\nabla f(\beta)$ we find

$$\nabla f(\beta) = -2\tilde{X}^T\tilde{Y} + 2\tilde{X}^T\tilde{X}\beta + 2\lambda\beta,$$

as desired.

(b)

We find the Hessian $\nabla^2 f(\beta)$ to be

$$\nabla^2 f(\beta) = 2\tilde{X}^T\tilde{X} + 2\lambda\mathbb{I}_{p-1},$$

where $\mathbb{I}_{p-1}$ is the $(p-1)\times(p-1)$ identity matrix. Note that $2\tilde{X}^T\tilde{X} \in \mathbb{S}_+^{p-1}$ is positive semi-definite and, with $\lambda > 0$, our scaled identity matrix $2\lambda\mathbb{I}_{p-1}$ is also positive semi-definite, $2\lambda\mathbb{I}_{p-1} \in \mathbb{S}_+^{p-1}$. Therefore, since a sum of positive semi-definite matrices is also positive semi-definite, we find

$$\nabla^2 f(\beta) = 2\tilde{X}^T\tilde{X} + 2\lambda\mathbb{I}_{p-1} \in \mathbb{S}_+^{p-1},$$

and so f must be strictly convex in β.

(c)

Strict convexity implies that the global minimizer must be unique, and so for $\lambda > 0$, we are guaranteed that the above solution will be the unique solution to our penalized least squares problem.

(d)

To write our function computing the ridge coefficients we first note that setting $\nabla f(\beta) = 0$ yields

$$\hat{\beta}^{(\lambda)}_{-1} = \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y}.$$

For the purpose of computational efficiency we make use of the singular value decomposition of $\tilde{X}$

$$\tilde{X} = UDV^T,$$

for $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{(p-1)\times(p-1)}$ both orthogonal matrices, $U^TU = \mathbb{I}_n$, $V^TV = \mathbb{I}_{p-1}$, and $D \in \mathbb{R}^{n\times(p-1)}$ a diagonal matrix with entries $\{d_j\}_{j=1}^{\min(n,\,p-1)}$ along the main diagonal. Hence,

$$\begin{aligned}
\hat{\beta}^{(\lambda)}_{-1} &= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y} \\
&= \left(\left(UDV^T\right)^T UDV^T + \lambda VV^T\right)^{-1}\left(UDV^T\right)^T\tilde{Y} \\
&= \left(VD^TU^TUDV^T + \lambda VV^T\right)^{-1} VD^TU^T\tilde{Y} \\
&= \left(V\left(D^TD + \lambda\mathbb{I}_{p-1}\right)V^T\right)^{-1} VD^TU^T\tilde{Y} \\
&= V\left(D^TD + \lambda\mathbb{I}_{p-1}\right)^{-1} V^TVD^TU^T\tilde{Y} \\
&= V\left(D^TD + \lambda\mathbb{I}_{p-1}\right)^{-1} D^TU^T\tilde{Y}.
\end{aligned}$$

Note that $D^TD + \lambda\mathbb{I}_{p-1}$ is a diagonal $(p-1)\times(p-1)$ matrix with entries $d_j^2 + \lambda$, $j = 1, ..., p-1$, and so the inverse $\left(D^TD + \lambda\mathbb{I}_{p-1}\right)^{-1}$ must also be diagonal with entries $\left(d_j^2 + \lambda\right)^{-1}$, $j = 1, ..., p-1$. We exploit this to avoid performing a matrix inversion in our code. For brevity we let

$$D^* = \left(D^TD + \lambda I_{p-1}\right)^{-1} D^T,$$

so that

$$\hat{\beta}^{(\lambda)} = VD^*U^T\tilde{Y}.$$

We present a function written in `R` performing such calculations below.

```
ridge_coef <- function(X, y, lam) {
  Xm1 <- X[,-1] # remove leading column of 1's marking the intercept

  ytilde <- y - mean(y) # center response
  xbar <- colMeans(Xm1) # find predictor means
  Xtilde <- sweep(Xm1, 2, xbar) # center each predictor according to its mean

  # compute the SVD on the centered design matrix
  Xtilde_svd <- svd(Xtilde)
  U <- Xtilde_svd$u
  d <- Xtilde_svd$d
  V <- Xtilde_svd$v

  # compute the inverse (D^T D + lambda I_{p-1})^{-1} D^T
  Dstar <- diag(d/(d^2 + lam))
```

```
  b <- V %*% (Dstar %*% crossprod(U, ytilde))
  b1 <- mean(y) - crossprod(xbar, b)
  return (list(b1 = b1, b = b))
}
```

Note the choice to use `V %*% (Dstar %*% crossprod(U, ytilde))` to compute the matrix product $VD^*U^T\tilde{Y}$ as opposed to (the perhaps more intuitive) `V %*% Dstar %*% t(U) %*% ytilde`. Such a choice is empirically justified in an appendix.

(e)

We first take the expectation of $\hat{\beta}_{-1}^{(\lambda)}$

$$\begin{aligned}
\mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda)}\right] &= \mathbb{E}\left[\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y}\right] \\
&= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\mathbb{E}\left[\tilde{Y}\right] \\
&= \left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{X}\beta_{-1} \\
&= \left(\tilde{X}^T\tilde{X} + \lambda\tilde{X}^T\tilde{X}\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1}\tilde{X}^T\tilde{X}\beta_{-1} \\
&= \left(\tilde{X}^T\tilde{X}\left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)\right)^{-1}\tilde{X}^T\tilde{X}\beta_{-1}.
\end{aligned}$$

Recall from elementary linear algebra that, if A and B are invertible matrices and if AB is defined, then

$$(AB)^{-1} = B^{-1}A^{-1}.$$

Hence

$$\begin{aligned}
\mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda)}\right] &= \left(\tilde{X}^T\tilde{X}\left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)\right)^{-1}\tilde{X}^T\tilde{X}\beta_{-1} \\
&= \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1}\left(\tilde{X}^T\tilde{X}\right)^{-1}\tilde{X}^T\tilde{X}\beta_{-1} \\
&= \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1}\beta_{-1},
\end{aligned}$$

as desired. We next compute the variance of our centered ridge estimates

$$\begin{aligned}
\mathrm{Var}\left(\hat{\beta}_{-1}^{(\lambda)}\right) &= \mathrm{Var}\left(\left(\tilde{X}^T\tilde{X} + \lambda\mathbb{I}_{p-1}\right)^{-1}\tilde{X}^T\tilde{Y}\right) \\
&= \mathrm{Var}\left(\left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1}\left(\tilde{X}^T\tilde{X}\right)^{-1}\tilde{X}^T\tilde{Y}\right) \quad \text{(by the same argument as above)} \\
&= \mathrm{Var}\left(\left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1}\hat{\beta}_{-1}^{\mathrm{OLS}}\right) \quad \text{(definition of } \hat{\beta}_{-1}^{\mathrm{OLS}}\text{)} \\
&= \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1}\mathrm{Var}\left(\hat{\beta}_{-1}^{\mathrm{OLS}}\right)\left(\left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1}\right)^T \\
&= \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1}\mathrm{Var}\left(\hat{\beta}_{-1}^{\mathrm{OLS}}\right)\left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1},
\end{aligned}$$

where the final line was achieved via the symmetry of $\mathbb{I}_{p-1}$ and $\tilde{X}^T\tilde{X}$. Recall that if our response $Y \sim \mathcal{N}(X\beta, \sigma^2)$ then our ordinary least squares estimates $\hat{\beta}^{\text{OLS}}$ of β have variance (conditional on X)

$$\text{Var}\left(\hat{\beta}^{\text{OLS}}\right) = \sigma^2 \left(X^T X\right)^{-1}.$$

So, if $\hat{\beta}_{-1}^{\text{OLS}}$ are our (centered) OLS estimates of β_{-1} for (centered) responses $\tilde{Y} \sim \mathcal{N}(\tilde{X}\beta_{-1}, \sigma_*^2)$ then

$$\text{Var}\left(\hat{\beta}_{-1}^{\text{OLS}}\right) = \sigma_*^2 \left(\tilde{X}^T \tilde{X}\right)^{-1}.$$

Hence

$$\begin{aligned}
\text{Var}\left(\hat{\beta}_{-1}^{(\lambda)}\right) &= \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1} \text{Var}\left(\hat{\beta}_{-1}^{\text{OLS}}\right) \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1} \\
&= \sigma_*^2 \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1} \left(\tilde{X}^T\tilde{X}\right)^{-1} \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1},
\end{aligned}$$

as desired. For computational considerations[1] we once again apply the SVD on $\tilde{X}$ as we had done before so that $\tilde{X} = UDV^T$. Then,

$$\begin{aligned}
\left(\tilde{X}^T\tilde{X}\right)^{-1} &= V\Delta V^T \\
\left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1} &= V\Delta^* V^T,
\end{aligned}$$

for diagonal matrices Δ and Δ^* given by

$$\begin{aligned}
\Delta &= \begin{bmatrix} d_1^{-2} & & \\ & \ddots & \\ & & d_{p-1}^{-2} \end{bmatrix} \\
\Delta^* &= \begin{bmatrix} \left(1 + \frac{\lambda}{d_1^2}\right)^{-1} & & \\ & \ddots & \\ & & \left(1 + \frac{\lambda}{d_{p-1}^2}\right)^{-1} \end{bmatrix}.
\end{aligned}$$

saving us a number of matrix inversions. Therefore, we can express our expectation by

$$\begin{aligned}
\mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda)}\right] &= \left(\mathbb{I}_{p-1} + \lambda\left(\tilde{X}^T\tilde{X}\right)^{-1}\right)^{-1} \beta_{-1} \\
&= V\Delta^* V^T \beta_{-1},
\end{aligned}$$

and variance by

[1]It turns out that the following method I proposed for speeding up the calculations are not very effective when $n >> p$ (in fact, it can be sometimes marginally slower), but very effective for $n \sim p$.

$$
\begin{aligned}
\mathrm{Var}\left(\hat{\beta}_{-1}^{(\lambda)}\right) &= \sigma_*^2 V\Delta^* V^T V \Delta V^T V \Delta^* V^T \\
&= \sigma_*^2 V \Delta^* \Delta \Delta^* V^T \\
&= \sigma_*^2 V \Delta^{**} V^T,
\end{aligned}
$$

where Δ^{**} is a $(p-1)\times(p-1)$ diagonal matrix with diagonal elements $\left(d_j + \frac{\lambda}{d_j}\right)^{-2}$, $j = 1, ..., p-1$. We now wish to perform a simulation study to estimate our theoretical values $\mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda)}\right]$ and $\mathrm{Var}\left(\hat{\beta}_{-1}^{(\lambda)}\right)$. For readability we first define functions computing the theoretical mean and variance according to our above expressions.

```
ridge_coef_params <- function(X, lam, beta, sigma) {
  n <- nrow(X); p <- ncol(p)
  betam1 <- beta[-1] # remove intercept term
  Xm1 <- X[,-1] # remove leading column of 1's in our design matrix

  xbar <- colMeans(Xm1) # find prector means
  Xtilde <- sweep(Xm1, 2, xbar) # center each predictor according to its mean

  # compute SVD on the centered design matrix
  Xtilde_svd <- svd(Xtilde)
  d <- Xtilde_svd$d
  V <- Xtilde_svd$v

  Delta_star <- diag(1/(1 + lam/d^2))
  Delta_star2 <- diag(1/(d + lam/d)^2)

  b <- V %*% (Delta_star %*% crossprod(V, betam1))
  vcv <- sigma^2 * V %*% tcrossprod(Delta_star2, V)
  return (list(b = b, vcv = vcv))
}
```

We may now perform our simulation.

```
set.seed(124)

# set parameters
nsims <- 1e3
n <- 1e2
p <- 6
lam <- 4
beta_star <- 1:p
sigma_star <- 1

# generate fixed design matrix
X <- cbind(1, matrix(rnorm(n * (p - 1)), nrow = n))

# compute theoretical mean and variance
par_true <- ridge_coef_params(X, lam, beta_star, sigma_star)
b_true <- as.vector(par_true$b)
vcv_true <- par_true$vcv

# simulate ridge coefficients nsims times
```

```r
# outputs a matrix with rows corresponding to coefficients
# and columns correspond to simulation number
pt <- proc.time()
b_hat <- replicate(nsims, {
  y <- X %*% beta_star + rnorm(n, 0, sigma_star)
  return (as.vector(ridge_coef(X, y, lam)$b))
})
proc.time() - pt
```

```
##    user  system elapsed
##   0.220   0.006   0.229
```

```r
# estimate variance of b1, ..., b_p estimates
vcv_hat <- var(t(b_hat))

# print estimated fused ridge coefficients vs. expected values
b <- rbind(rowMeans(b_hat), b_true)
rownames(b) <- c("b_hat", "b_true")
round(b, 4)
```

```
##          [,1]   [,2]   [,3]   [,4]   [,5]
## b_hat  1.9145 2.8529 3.8017 4.8510 5.6930
## b_true 1.9207 2.8527 3.8043 4.8506 5.6961
```

```r
# print absolute error between estimated and true fused ridge variances
round(abs(vcv_true - vcv_hat), 4)
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 0e+00 7e-04 0e+00 2e-04 8e-04
## [2,] 7e-04 3e-04 4e-04 5e-04 2e-04
## [3,] 0e+00 4e-04 1e-04 1e-04 1e-04
## [4,] 2e-04 5e-04 1e-04 7e-04 2e-04
## [5,] 8e-04 2e-04 1e-04 2e-04 1e-04
```

We see that the empirical sample estimates are very close to their theoretical values, as expected.

Question 3

```r
ridge_cv <- function(X, y, lam.vec, K) {

}
```

Question 4

For this problem we first define some additional functions and set some global parameters which remain constant across (a)-(d)

```r
set.seed(124)

# global parameters
nsims <- 50
lams <- 10^seq(-8, 8, 0.5)
sigma_star <- sqrt(1/2)
```

(a)

```r
# set parameters
n <- 100
p <- 50
theta <- 0.5

# generate data
beta_star <- rnorm(p, 0, sigma_star)
Z <- matrix(rnorm(n * (p - 1)), nrow = n, ncol = p - 1) # indep. normal deviates
SIGMA <- outer(1:(p - 1), 1:(p - 1), FUN = function(a, b) theta^abs(a - b))
C <- chol(SIGMA)
X <- cbind(rep(1, n), Z %*% C) # correlated normal deviates

# simulate noise and response
sim <- replicate(nsims, {
  eps <- rnorm(n, 0, sigma_star)
  y <- X %*% beta_star + eps

})
```

(b)

(c)

(d)

Question 5

(a)

Taking the gradient of our objective function g with respect to coefficient vector β yields

$$\begin{aligned}\nabla_\beta g(\beta, \sigma^2) &= \nabla_\beta \left(\frac{n}{2} \left(\log \sigma^2 \right) + \frac{1}{2\sigma^2} \|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right) \\ &= -\frac{1}{\sigma^2} \left(\tilde{X}^T \tilde{Y} + \tilde{X}^T \tilde{X} \beta \right) + \lambda \beta,\end{aligned}$$

while the gradient of g with respect to σ^2 is given by

$$\begin{aligned}\nabla_{\sigma^2} g(\beta, \sigma^2) &= \nabla_\beta \left(\frac{n}{2} \left(\log \sigma^2 \right) + \frac{1}{2\sigma^2} \|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda}{2} \|\beta\|_2^2 \right) \\ &= \frac{n}{2\sigma^2} - \frac{1}{\sigma^4} \|\tilde{Y} - \tilde{X}\beta\|_2^2.\end{aligned}$$

(b)

(c)

(d)

(e)

(f)

Question 6

(a)

Consider our objective function

$$f(\beta) = \frac{1}{2}\|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda_1}{2}\|\beta\|_2^2 + \frac{\lambda_2}{2}\sum_{j=2}^{p}(\beta_j - \beta_{j-1})^2$$

To show convexity we wish to show $\nabla^2 f(\beta) \in \mathbb{S}_+^{p-1}$. However, it's not immediately obvious how to take such a gradient with our fused sum terms $(b_j - \beta_{j-1})^2$. One way to get around this is to define vector $B \in \mathbb{R}^{p-1}$ given by

$$B = \begin{bmatrix} \beta_2 - \beta_1 \\ \vdots \\ \beta_p - \beta_{p-1} \end{bmatrix}$$

Then

$$\sum_{j=2}^{p}(\beta_j - \beta_{j-1})^2 = B^T B$$

In order to achieve our task of expressing the fused sum in terms of the vector β we must next decompose B into a product of β and some matrix. To this end we define matrix $A \in \mathbb{R}^{(p-2)\times(p-1)}$ with entries -1 along the main diagonal and 1 along the upper diagonal, i.e.,

$$A = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}$$

Then

$$\begin{aligned} \sum_{j=2}^{p}(\beta_j - \beta_{j-1})^2 &= B^T B \\ &= \beta^T A^T A \beta \\ &\equiv \|A\beta\|_2^2 \end{aligned}$$

Therefore, our objective function can be expressed as

$$\begin{aligned} f(\beta) &= \frac{1}{2}\|\tilde{Y} - \tilde{X}\beta\|_2^2 + \frac{\lambda_1}{2}\|\beta\|_2^2 + \frac{\lambda_2}{2}\|A\beta\|_2^2 \\ &\equiv \frac{1}{2}\tilde{Y}^T\tilde{Y} - \beta^T\tilde{X}^T\tilde{Y} + \frac{1}{2}\beta^T\tilde{X}^T\tilde{X}\beta + \frac{\lambda_1}{2}\beta^T\beta + \frac{\lambda_2}{2}\beta^T A^T A\beta \end{aligned}$$

Hence

$$\nabla f(\beta) = -\tilde{X}^T\tilde{Y} + \tilde{X}^T\tilde{X}\beta + \lambda_1\beta + \lambda_2 A^T A\beta$$

admitting the Hessian

$$\nabla^2 f(\beta) = \tilde{X}^T\tilde{X} + \lambda_1\mathbb{I}_{p-1} + \lambda_2 A^T A$$

Recalling that a matrix multiplied with its transpose must always be positive semi-definite, we find $\tilde{X}^T X$ and $A^T A$ must be positive semi-definite. Thus, since $\lambda_1 > 0$, we find that our sum $\tilde{X}^T\tilde{X} + \lambda_1\mathbb{I}_{p-1} + \lambda_2 A^T A = \nabla^2 f(\beta)$ is positive semi-definite, and so $f(\beta)$ must be strictly convex, as desired.

(b)

We first solve for $\hat{\beta}_{-1}^{(\lambda_1,\,\lambda_2)}$ in (a) by setting $\nabla f(\beta) = 0$

$$\begin{aligned} 0 &= -\tilde{X}^T\tilde{Y} + \tilde{X}^T\tilde{X}\beta + \lambda_1\beta + \lambda_2 A^T A\beta \\ \tilde{X}^T\tilde{Y} &= \left(\tilde{X}^T\tilde{X} + \lambda_1\mathbb{I}_{p-1} + \lambda_2 A^T A\right)\beta \\ \implies \hat{\beta}_{-1}^{(\lambda_1,\,\lambda_2)} &= M\tilde{X}^T\tilde{Y} \end{aligned}$$

where we have set $M = \left(\tilde{X}^T\tilde{X} + \lambda_1\mathbb{I}_{p-1} + \lambda_2 A^T A\right)^{-1}$ for brevity. Therefore

$$\begin{aligned} \mathbb{E}\left[\hat{\beta}_{-1}^{(\lambda_1,\,\lambda_2)}\right] &= \mathbb{E}\left[M\tilde{X}^T\tilde{Y}\right] \\ &= M\tilde{X}^T\mathbb{E}\left[\tilde{Y}\right] \\ &= M\tilde{X}^T\beta_{*,\,-1} \end{aligned}$$

and

$$\begin{aligned} \mathrm{Var}\left(\hat{\beta}_{-1}^{(\lambda_1,\,\lambda_2)}\right) &= \mathrm{Var}\left(M\tilde{X}^T Y\right) \\ &= M\tilde{X}^T\mathrm{Var}\left(\tilde{Y}\right)\tilde{X}M^T \\ &= \sigma_*^2 M\tilde{X}^T\tilde{X}M^T \end{aligned}$$

as desired. We now perform our fused ridge simulation study to test the theoretical values with some empirical estimates. We first define our fused ridge coefficient estimation function (as well as functions permitting us to easily compute the theoretical means and variances of the fused ridge problem)

```r
fused_ridge_coef <- function(X, y, lam1, lam2) {
  n <- nrow(X); p <- ncol(X)
  Xm1 <- X[,-1] # remove leading column of 1's marking the intercept

  ytilde <- y - mean(y) # center response
  xbar <- colMeans(Xm1) # find predictor means
  Xtilde <- sweep(Xm1, 2, xbar) # center each predictor according to its mean

  I <- diag(p - 1)
  UD <- cbind(rep(0, p - 2), diag(p - 2)) # upper diagonal matrix
  J <- -1 * cbind(diag(p - 2), rep(0, p - 2)) # diag (p - 2)*(p - 1) matrix
  A <- J + UD

  M <- solve(crossprod(Xtilde) + lam1 * I + lam2 * crossprod(A))
  b <- M %*% crossprod(Xtilde, y)
  b0 <- mean(y) - crossprod(xbar, b)
  return(list(b0 = b0, b = b))
}
fused_ridge_coef_params <- function(X, lam1, lam2, beta, sigma) {
  # omits intercept term b0
  # returns theoretical means and variances for the fused ridge problem
  n <- nrow(X); p <- ncol(X)
  Xm1 <- X[,-1] # remove leading column of 1's marking the intercept
  betam1 <- beta[-1] # remove intercept term

  xbar <- colMeans(Xm1) # find predictor means
  Xtilde <- sweep(Xm1, 2, xbar) # center each predictor according to its mean

  I <- diag(p - 1)
  UD <- cbind(rep(0, p - 2), diag(p - 2)) # upper diagonal matrix
  J <- -1 * cbind(diag(p - 2), rep(0, p - 2)) # diag (p - 2)*(p - 1) matrix
  A <- J + UD

  M <- solve(crossprod(Xtilde) + lam1 * I + lam2 * crossprod(A))
  b <- M %*% crossprod(Xtilde, (Xtilde %*% betam1))

  vcv <- matrix(0, nrow = p - 1, ncol = p - 1)
  if (n > p) { # when n > p this matrix multiplication routine is quicker
    vcv <- sigma^2 * M %*% tcrossprod(crossprod(Xtilde), M)
  } else { # when p > n this matrix multiplication routine is quicker
   vcv <- sigma^2 * tcrossprod(M, Xtilde) %*% tcrossprod(Xtilde, M)
  }

  return (list(b = b, vcv = vcv))
}
```

We now simulate some data to test our estimates:

```r
set.seed(124)

# set parameters
nsims <- 1e4
n <- 1e2
p <- 5
```

```r
lam1 <- 1
lam2 <- 1
sigma_star <- 1
beta_star <- rnorm(p)

# generate (fixed) design matrix
X <- cbind(rep(1, n), matrix(rnorm(n * (p - 1)), nrow = n, ncol = p - 1))

# compute expected parameter values
par_true <- fused_ridge_coef_params(X, lam1, lam2, beta_star, sigma_star)
b_true <- as.vector(par_true$b)
vcv_true <- par_true$vcv

# simulate our fused ridge coefficients nsims times
# outputs a matrix with rows corresponding to coefficients
# and columns correspond to simulation number
pt <- proc.time()
b_hat <- replicate(nsims, {
  y <- X %*% beta_star + rnorm(n, 0, sigma_star) # generate response
  return (as.vector(fused_ridge_coef(X, y, lam1, lam2)$b))
})
proc.time() - pt
```

```
##    user  system elapsed
##   2.199   0.027   2.523
```

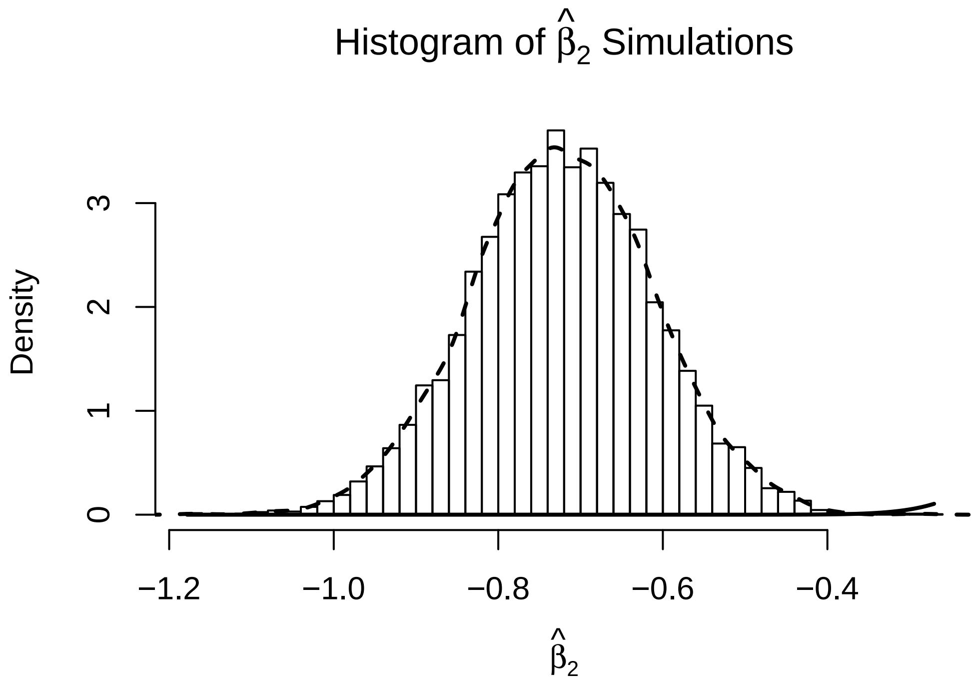
```r
# estimate variance of b2, ..., b_p estimates
vcv_hat <- var(t(b_hat))

# print estimated fused ridge coefficients vs. expected values
b <- rbind(rowMeans(b_hat), b_true)
rownames(b) <- c("b_hat", "b_true")
round(b, 4)
```

```
##          [,1]    [,2]   [,3]   [,4]
## b_hat  0.0316 -0.7226 0.2226 1.3899
## b_true 0.0313 -0.7240 0.2235 1.3920
```

```r
# print absolute error between estimated and true fused ridge variances
round(abs(vcv_true - vcv_hat), 4)
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] 2e-04 1e-04 1e-04 1e-04
## [2,] 1e-04 1e-04 1e-04 2e-04
## [3,] 1e-04 1e-04 0e+00 1e-04
## [4,] 1e-04 2e-04 1e-04 3e-04
```

As a case study, we may look at the simulations of $\hat{\beta}_2^{(\lambda_1,\lambda_2)}$ and compare it with it's theoretical distribution. Note that the estimates $\hat{\beta}^{(\lambda_1,\lambda_2)} = M\tilde{X}^T\tilde{Y}$ are normally distributed because they are a linear combination of $\tilde{Y} \sim \mathcal{N}(\tilde{X}\beta, \sigma^2)$ (when our noise terms $\epsilon \sim \mathcal{N}(0, \sigma^2)$). We visualize the histogram of the $\hat{\beta}_2^{(\lambda_1,\lambda_2)}$ simulations with its empirical and theoretical densities overlaid (dashed, solid), along with its expected value (vertical line) below.



Histogram of $\hat{\beta}_2$ Simulations
Density
0
1
2
3
−1.2
−1.0
−0.8
−0.6
−0.4
$\hat{\beta}_2$

Appendix

Matrix Multiplication Timing

Consider the following matrix multiplication benchmarks (for the cases of $n >> p$ and $p >> n$).

```
library(microbenchmark)

#===== Large n case =====#
set.seed(124)

# set parameters
n <- 1e3
p <- 1e2
lam <- 1

# generate data
X <- matrix(rnorm(n * p), nrow = n)
beta <- rnorm(p)
eps <- rnorm(n)
y <- X %*% beta + eps

ytilde <- y - mean(y)
xbar <- colMeans(X)
Xtilde <- sweep(X, 2, xbar)

# compute decomposition
Xtilde_svd <- svd(Xtilde)
U <- Xtilde_svd$u
d <- Xtilde_svd$d
V <- Xtilde_svd$v
Dstar <- diag(d/(d^2 + lam))

# define multiplication functions
f1 <- function() V %*% Dstar %*% t(U) %*% ytilde
f2 <- function() V %*% Dstar %*% (t(U) %*% ytilde)
f3 <- function() V %*% (Dstar %*% (t(U) %*% ytilde))
f4 <- function() V %*% (Dstar %*% crossprod(U, ytilde))
f5 <- function() V %*% crossprod(Dstar, crossprod(U, ytilde))

# test speed
microbenchmark(f1(), f2(), f3(), f4(), f5(), times = 100, unit = "us")
```

```
## Unit: microseconds
##   expr      min       lq       mean     median        uq       max neval
##   f1() 8577.004 9818.477 12627.1034 10603.4305 12829.470 55410.373   100
##   f2() 1108.271 1236.065  1701.7573  1408.3960  2161.596  4806.444   100
##   f3()  372.457  451.897  1623.4061   562.3735   753.077 48340.309   100
##   f4()  130.438  138.466   180.1480   156.2790   176.267  1162.350   100
##   f5()  126.630  132.799   215.3654   154.3505   173.159  4227.418   100
```

```
#===== Large p case =====#
set.seed(124)

# set parameters
```

```r
n <- 1e2
p <- 1e3
lam <- 1

# generate data
X <- matrix(rnorm(n * p), nrow = n)
beta <- rnorm(p)
eps <- rnorm(n)
y <- X %*% beta + eps

# define multiplication functions
f1 <- function() V %*% Dstar %*% t(U) %*% ytilde
f2 <- function() V %*% Dstar %*% (t(U) %*% ytilde)
f3 <- function() V %*% (Dstar %*% (t(U) %*% ytilde))
f4 <- function() V %*% (Dstar %*% crossprod(U, ytilde))
f5 <- function() V %*% crossprod(Dstar, crossprod(U, ytilde))

# test speed
microbenchmark(f1(), f2(), f3(), f4(), f5(), times = 100, unit = "us")
```

```
## Unit: microseconds
##   expr      min        lq       mean     median         uq       max neval
##   f1() 8957.257 10439.7035 12399.8537 11340.7225 13330.7770 45850.697   100
##   f2() 1100.864  1381.4075  1771.8768  1552.9405  2056.9075  6150.713   100
##   f3()  359.171   475.4390  1149.6884   571.6775   746.1980 40504.652   100
##   f4()  132.024   153.9875   190.2180   176.3490   206.2130   806.032   100
##   f5()  126.438   140.9070   171.2517   159.8620   182.7235   382.560   100
```