SDS 385 Ex 05:
Sparsity in Covariates (LASSO)

October 3, 2016

Jennifer Starling

Problem 1 - Penalized Regression & Soft Thresholding

Part A

Proof 1

Show the quadratic term in the objective is the negative loglikelihood of a Gaussian with mean θ, var = 1.

The $Gaussian(\theta, 1)$ distribution has pdf $f(y|\theta, 1) = (2\pi\sigma^2)^{-1/2}exp[-\frac{(y-\theta)^2}{2\sigma^2}] = (2\pi)^{-1/2}exp[-\frac{(y-\theta)^2}{2}]$

Then the likelihood function is $L(\theta|y, 1) = (2\pi)^{-1/2}exp[-\frac{(y-\theta)^2}{2}]$ (looking at a single y)

Then the log-likelihood function is $logL = log(L(\theta|y, 1) = \frac{-1}{2}log(2\pi) - \frac{1}{2}(y-\theta)^2$, and the first term is not dependent on θ, so we drop it out.

Then the negative log-likelihood function is $-\frac{1}{2}(y-\theta)^2$.

Therefore, the quadratic term in the objective is the negative log-likelihood of a Gaussian with mean θ and variance 1. ■

Proof 2

Prove that $S_\lambda(y) = sign(y)\dot{(}|y| - \lambda)_+$ where $a_+ = max(a, 0)$, the positive part of a.

Take the derivate of $S_\lambda(y)$ to obtain:

$$\frac{\delta S_\lambda(y)}{\delta\theta} = \frac{\delta}{\delta\theta}\left(\frac{1}{2}(y-\theta)^2 + \lambda|\theta|\right) = -(y-\theta) + \lambda\frac{|\theta|}{\theta} = -(y-\theta) + \lambda * sign(\theta)$$

Break the problem into three cases:

(1) $\theta > 0$
(2) $\theta < 0$
(3) $\theta = 0$

Case 1: If $\theta > 0$, the objective function set equal to zero can be rewritten as $-(y-0) + \lambda = 0 \rightarrow \theta = y - \lambda$

Constraint: $\theta > 0$, and $\theta = y - \lambda$, so $y - \theta > 0 \rightarrow y > \lambda$

Case 2: If $\theta < 0$, the objective function set equal to zero can be rewritten as $-(y-0) - \lambda = 0 \rightarrow \theta = y + \lambda$

Constraint: $\theta < 0$, and $\theta = y + \lambda$, so $y + \lambda < 0 \rightarrow y < -\lambda$

Case 3: The above cases cover $y > \lambda$ and $y < -\lambda$, leaving $|y| < \lambda$ as the $\theta = 0$ constraint.

These three cases can be summarized in a single function:

$S_\lambda(y) = sign(y)(|y| - \lambda)_+$ ■

Part B

1 through 3

The plotted grid of $\hat{\theta}(y_i)$ versus θ_i across varyingλ values shows how certain θ_i's are selected (i.e. set to zero), while the non-zero θ_i's are shrunk towards zero.

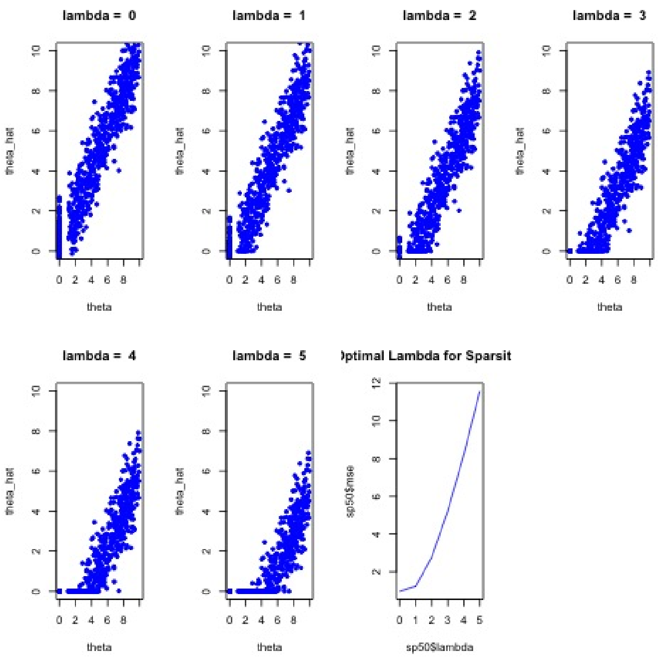

Figure 1: $\hat{\theta}(y_i)$ vs θ_i for varying λ at 50% sparsity

Below is a similar figure, for a smaller group of lambdas (0.0, 0.2, 0.4, 0.6, 0.8, 1.0), which better illustrations finding an optimal lambda value. This plot is also done using 50% sparsity.

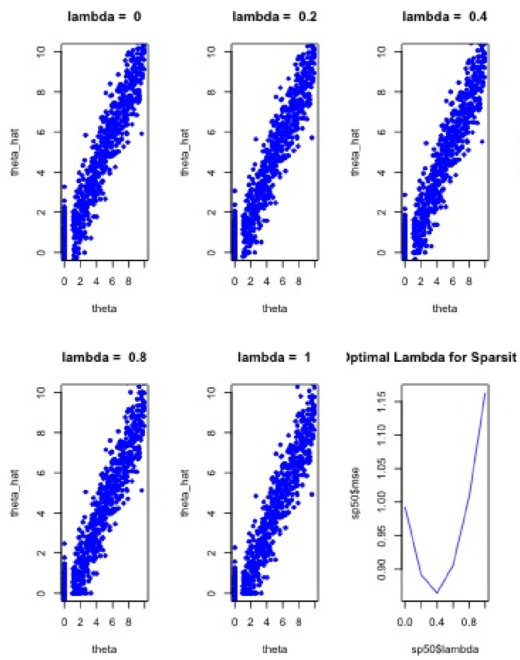

Figure 2: $\hat{\theta}(y_i)$ vs θ_i for varying λ at 50% sparsity

4 The plot of the MSE of the estimate as a function of θ is as follows. Horizontal lines represent the location of the minimum MSE for each sparsity level. Notice that increasing sparsity has the effect of increasing the optimal lambda value.

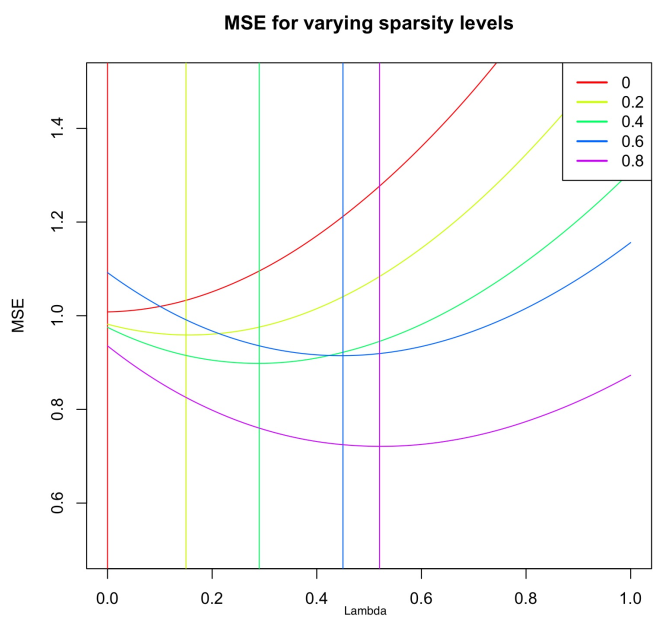

Figure 3: MSE as a function of λ for varying levels of sparsity

Problem 2 - The LASSO

Part A

The plot of the solution path of $\hat{\beta}_\lambda$ as a function of λ is as follows. Note that the glmnet plot function plots based on $log(\lambda)$ instead of λ so that the λ values are evenly spaced. The values across the top of the plot represent degrees of freedom, ie non-zero coefficients, at each λ value.

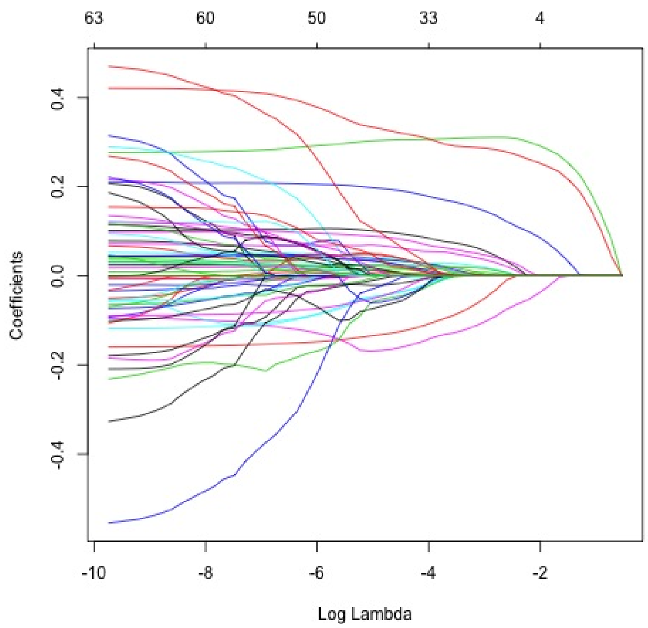

Figure 4: solution path of $\hat{\beta}_\lambda$ as a function of λ

The optimal lambda in this case was zero, which makes sense because the model is being optimally fit to the $\hat{\beta}$'s.
See section C for the plot of the MSE, MOOSE (CV) and Cp errors together.

Part B

The optimal lambda selected by my custom cross-validation function was 0.03493409. (This matched the lambda using the glmnet built-in cv functionality.)

See section C for the plot of the MSE, MOOSE (CV) and Cp errors together.

Part C

The optimal lambda selected by Mallow's Cp was 0.03493409. This matched the optimal lambda selected by cross-validation in part B.

The plots of the MSE, CV (MOOSE) and Cp errors by lambda is below. The second plot is using log(lambda) on the x-axis. The optimal lambda occurs at the same value for CV and Cp.

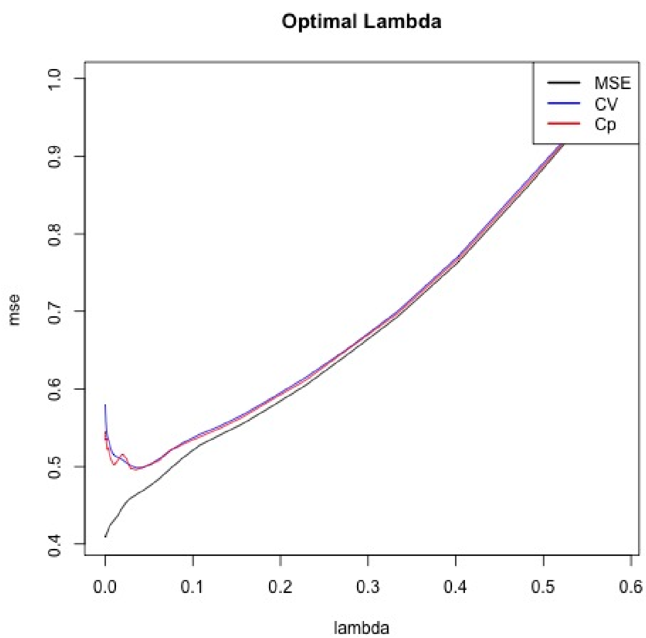

Figure 5: Various errors by λ

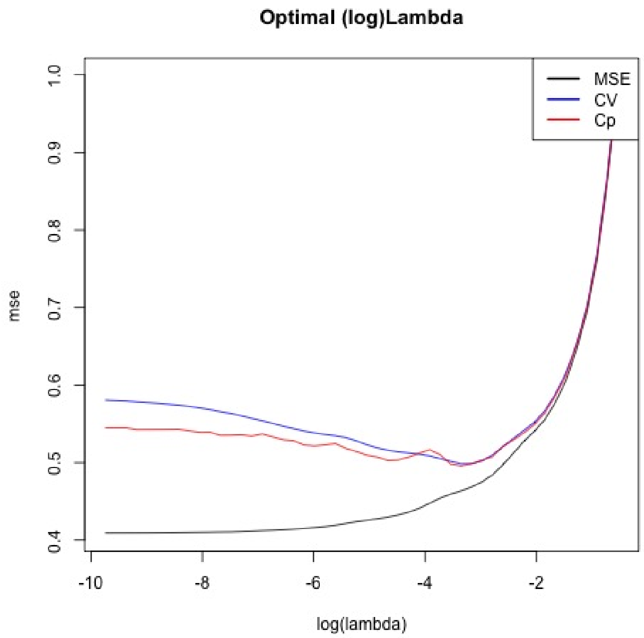

Figure 6: Various errors by $log(\lambda)$

Appendix: R Code

```
#Big Data Exercise 5
#Jennifer Starling
#27 Sept 2016

rm(list=ls())

####################################################
###   PENALIZED LHOOD & SOFT THRESHOLDING        ###
####################################################

#-------------------------------------------------------------------------------
#Functionalize the toy example, so that is easy to call for various sparsity
   levels.
toy_example <- function(n,sd,sparsity,lambda){
#Function inputs:
    #n = sample size
    #sd = n-length vector of standard deviations.
    #sparsity = percent of theta value sparse; 0 to 1.
    #lambda = vector of lambda values to test.
#Function output:
    #mse = Vector of MSE values for each theta.
    #theta = 'True values' of generated thetas.
    #theta_hat = Matrix of estimated theta values.  Each col = a different lambda.

    #Generate "true" theta values; different for each obs.
    theta = sample(seq(1,10,by=.01),n,replace=T)
    theta[sample(1:n,sparsity*n,replace=T)]=0

    #Simulate n y-values, using yi ~ N(theta_i,sd_i)
```

```
    y = rep(0,n)
    for (i in 1:n){
        y[i] = rnorm(1,theta[i],sd[i])
    }

    #Initialize theta_hat matrix.
    theta_hat <- matrix(0,nrow=n,ncol=length(lambda))
    colnames(theta_hat) = paste('lambda=',lambda)

    #Initialize vector to hold MSE for each lambda.
    mse <- rep(0,length(lambda))

    #Calculate theta_hat values for each lambda.
    for (j in 1:length(lambda)){
        #Calculate the Sy function; takes a few steps.
        Sy <- abs(y)-lambda[j]
        Sy[which(Sy<0)]=0   #Take only the positive part.
        Sy <- sign(y)*Sy    #Multiply by sign of y.

        #Assign Sy=theta_hat to its column.
        theta_hat[,j] = Sy

        #Calculate mse for lambda_j.
        mse[j] <- (1/n) * sum((theta_hat[,j] - theta)^2)
    }

    return(list(theta=theta,lambda=lambda,theta_hat=theta_hat,mse=mse,sparsity=
        sparsity))

} #end function.

#
    ------------------------------------------------------------------------------------------

#EXAMPLE 1: With exaggerated (large) lambdas, to show drastic shrinking/sparsity
    in coeffs.

#Try out function with 50% sparsity, n=1000, sd=1.
n=1000
sp50 = toy_example(n=1000,sd <- rep(1,n), sparsity=.5,lambda=c(0,1,2,3,4,5))

#Plot results for varying lambda values.
par(mfrow=c(2,4))
for (j in 1:length(sp50$lambda)){
    plot(sp50$theta,sp50$theta_hat[,j],col='blue',pch=20,xlim=c(0,10),ylim=c(0,10)
        ,
    main=paste('lambda = ',sp50$lambda[j]),xlab='theta',ylab='theta_hat')
}

#Plot MSE for this sparsity level.
plot(sp50$lambda,sp50$mse,type='l',col='blue',main=paste('Optimal Lambda for
    Sparsity ',sp50$sparsity))

#
    ------------------------------------------------------------------------------------------

#EXAMPLE 2: With more realistic lambdas, sparsity 50%

sp50 = toy_example(n=1000,sd <- rep(1,n), sparsity=.5,lambda=seq(0,1,by=.2))
```

```
#Plot results for varying lambda values.
par(mfrow=c(2,4))
for (j in 1:length(lambda)){
    plot(sp50$theta,sp50$theta_hat[,j],col='blue',pch=20,xlim=c(0,10),ylim=c(0,10)
        ,
    main=paste('lambda = ',sp50$lambda[j]),xlab='theta',ylab='theta_hat')
}

#Plot MSE for this sparsity level.
plot(sp50$lambda,sp50$mse,type='l',col='blue',main=paste('Optimal Lambda for
    Sparsity ',sp50$sparsity))

#
    ----------------------------------------------------------------------------------------------------

#PROBLEM B-4: Plot MSE for several configurations of theta (sparsity levels)
#and observe how optimal lambda changes.

#Initialize a vector of sparsity levels.
sp_levels <- seq(0,.8,by=.2)
mse <- list()
lambda=seq(0,1,by=.01)

#Loop through sparsity levels.
for (s in 1:length(sp_levels)){

    #Run toy example for given sparsity level. Save mse.
    temp = toy_example(n=1000,sd <- rep(1,n), sparsity=sp_levels[s],lambda=lambda)
    mse[[s]] = temp$mse
}

#Plot MSE for each sparsity level.
colors <- rainbow(length(sp_levels))
plot(lambda,mse[[1]],col = colors[1],type='l',xlim=c(0,1),ylim=c(.5,1.5),
    main='MSE for varying sparsity levels',xlab='lambda',ylab='MSE')
abline(v=lambda[which(mse[[1]]==min(mse[[1]]))],col=colors[1])

for (j in 2:length(sp_levels)){
    lines(lambda,mse[[j]],col = colors[j],type='l')
    abline(v=lambda[which(mse[[j]]==min(mse[[j]]))],col=colors[j])
}
labels = paste(sp_levels)
legend('topright',legend=labels,lwd=2,col=colors, bty != "n", bg='white')


##############################
###   THE LASSO            ###
##############################
library(glmnet)

#Read in Diabetes.csv data.
X <- read.csv(file='/Users/jennstarling/UTAustin/2016_Fall_SDS 385_Big_Data/
    Exercise 05 R Code/DiabetesX.csv',header=T)
y <- read.csv(file='/Users/jennstarling/UTAustin/2016_Fall_SDS 385_Big_Data/
    Exercise 05 R Code/DiabetesY.csv',header=F)

#Scale X and y.
X = scale(X)
```

```
y = scale(y)

#-----------------------------------------------------
#Part A:

#Fit lasso model across a range of lambda values (which glmnet does automatically)
    .
#Plot the solution path beta_hat_lambda as a funciton lambda.
myLasso <- glmnet(X,y,family='gaussian',nlambda=50)

#Plot of beta_hat as function of lambda.
jpeg(file='/Users/jennstarling/UTAustin/2016_Fall_SDS 385_Big_Data/Exercise05
    LaTeX Files/LassoBetaPaths.jpg')
plot(myLasso,xvar="lambda", main='Lasso: Solution path of Beta_hat as function of
    lambda')
dev.off()

#Track in-sample MSE prediction error of the fit across the solution path:

lambda = myLasso$lambda
betas = myLasso$beta
n = nrow(X)

#Initialize vector to hold MSE for each beta.
MSE_betas = rep(0,length(lambda))

for (i in 1:length(lambda)){
    MSE_betas[i] = (1/n) * sum((y-X %*% betas[,i])^2)
}

jpeg(file='/Users/jennstarling/UTAustin/2016_Fall_SDS 385_Big_Data/Exercise05
    LaTeX Files/LassoMSE.jpg')
par(mfrow=c(1,2))
plot(log(lambda),MSE_betas,type='l')
plot(lambda,MSE_betas,type='l')
dev.off()

#-----------------------------------------------------
#Part B:

#Run cross-validation using vector of lambdas from model fit in part A.
lassoCV = myCV(X,y,lambda,cv_folds=10)

#Plot CV results to visualize optimal lambda.
plot(lassoCV$lambda,lassoCV$pred_err_by_lambda,type='l',col='blue')

#Output minimum lambda.
paste('Optimal lambda: ',
    lassoCV$lambda[which(lassoCV$pred_err_by_lambda==min(lassoCV$pred_err_by_
        lambda))])

#Test results against the built-in cross-val functionality in glmnet.
cvfit = cv.glmnet(X,y,lambda=lambda)
cvfit$lambda.min

#RESULTS:
#> paste('Optimal lambda: ',
#+  lassoCV$lambda[which(lassoCV$pred_err_by_lambda==min(lassoCV$pred_err_by_
    lambda))])
```

```
#[1] "Optimal lambda:  0.0349340915741447"
#> cvfit$lambda.min
#[1] 0.03493409
#

#My cross-validation function.
myCV <- function(X,y,lambda,cv_folds=10){
    #data = holds predictors and response.
    #lambda = vector of lambdas to include in the model.
    #folds = number of cross-val folds.

    #Randomly shuffle data.
    data = cbind(y,X)

    data = data[sample(nrow(data)),]

    #Create 'folds' number of equally sized folds.
    folds <- cut(seq(1,nrow(data)),breaks=cv_folds,labels=F)

    #Initialize vector to hold prediction error for each cv fold iteration.
    pred_test_err = matrix(0,nrow=cv_folds,ncol=length(lambda))
    #pred_test_error <- rep(0,cv_folds)

    #Perform cross-validation.
    for (i in 1:cv_folds){

        #Split up data using folds.
        testIndices <- which(folds==i,arr.ind=T)
        testData <- data[testIndices, ]
        trainData <- data[-testIndices, ]

        #Fit glmnet lasso model for the data excluding the current fold.
        trainLasso = glmnet(x=trainData[,-1],y=trainData[,1],family='gaussian',
            nlambda=50,lambda=lambda)

        #Predict values on test data.
        predLasso = predict(trainLasso,newx=testData[,-1],s=lambda)

        #Calculate and save prediction error.
        predErr = apply(predLasso,2,function(yhat) sum((yhat-testData[,1])^2))/
            nrow(testData)
        pred_test_err[i,] = predErr
    }

    #Return average predicted test error for each k value.
    return(list(lambda=lambda,
        pred_err_by_lambda=colMeans(pred_test_err),
        pred_err_var = apply(pred_test_err,2,var)))
}

#------------------------------------------------------------
#Part C: Compute and plot the Cp statistic (Mallow's Cp) as a function of lambda.

#Use the Part A glm lasso model, fit using whole data set for the same vector of
    50 lambdas.
#Also use the MSE calculated in part A.
lambda = myLasso$lambda
mse = MSE_betas
df = myLasso$df
```

```
n = nrow(X)

#Fit an OLS model to obtain estimate of sigma2.
mylm = lm(y~X-1)     #Fit model with no intercept
sigma2_hat = summary(mylm)$sigma^2

#Calculate Mallow's cp.
Cp = mse + 2 * (df/n) * sigma2_hat

#Plot Mallow's Cp as a function of lambda.
plot(lambda,Cp,type='l',main='Mallows Cp as a function of lambda',xlab='lambda',
    ylab='Cp')

#Output optimal lambda based on smallest Mallow's Cp.
paste('Optimal lambda based on Cp: ',lambda[which(Cp==min(Cp))])

#PLOTTING:
jpeg(file='/Users/jennstarling/UTAustin/2016_Fall_SDS 385_Big_Data/Exercise05
    LaTeX Files/LassoOptimalLambda.jpg')
#Plot MSE, CV error and Cp on the same plot, to compare the optimal values of
    lambda they each yield.
plot(lambda,mse,type='l',col='black',main='Optimal Lambda')
lines(lambda,lassoCV$pred_err_by_lambda,col='blue')
lines(lambda,Cp,col='red')
labels = c('MSE','CV','Cp')
legend('topright',legend=labels,lwd=2,col=c('black','blue','red'), bty != "n", bg=
    'white')
dev.off()

jpeg(file='/Users/jennstarling/UTAustin/2016_Fall_SDS 385_Big_Data/Exercise05
    LaTeX Files/LassoOptimalLogLambda.jpg')
plot(log(lambda),mse,type='l',col='black',main='Optimal (log)Lambda')
lines(log(lambda),lassoCV$pred_err_by_lambda,col='blue')
lines(log(lambda),Cp,col='red')
labels = c('MSE','CV','Cp')
legend('topright',legend=labels,lwd=2,col=c('black','blue','red'), bty != "n", bg=
    'white')
dev.off()
```