Ridge Regression, LASSO and Smoothing

Sam Bowyer

2023-03-16

Task 1

For this first task we will be using the Communities and Crime dataset from the `R` package `mogavs`. This contains 1595 datapoints each with data on $p = 122$ variables and one target variable. We use a random 80%/20% split to obtain distinct training and test sets with 1595 and 399 datapoints respectively.

```
library(mogavs)  # for crime & communities dataset
data("crimeData")

p = ncol(crimeData) - 1

propTrain = 0.8
trainIdx  = sample(1:nrow(crimeData), nrow(crimeData)*0.8)

XTrain = crimeData[trainIdx, -(p+1)]
yTrain = crimeData[trainIdx, p+1]

XTest = crimeData[-trainIdx, -(p+1)]
yTest = crimeData[-trainIdx, p+1]

dim(XTrain); length(yTrain)
```

```
## [1] 1595  122
```

```
## [1] 1595
```

```
dim(XTest); length(yTest)
```

```
## [1] 399 122
```

```
## [1] 399
```

Next we center and scale our data.

```
# center the X data around the training mean
XTrain_mean = colMeans(XTrain)
XTrain = XTrain - rep(XTrain_mean, rep.int(nrow(XTrain), p))
XTest = XTest - rep(XTrain_mean, rep.int(nrow(XTest), p))

# scale the data so that variables have unit variance
D = diag(cov(XTrain))
XTrain = XTrain * (D^(-1/2))
Xest   = XTest * (D^(-1/2))
```

Ridge Regression

For both ridge and LASSO regression we use the package `glmnet`, which imposes a penalty given by:

$$(1-\alpha)/2||\beta||_2^2 + \alpha||\beta||_1.$$

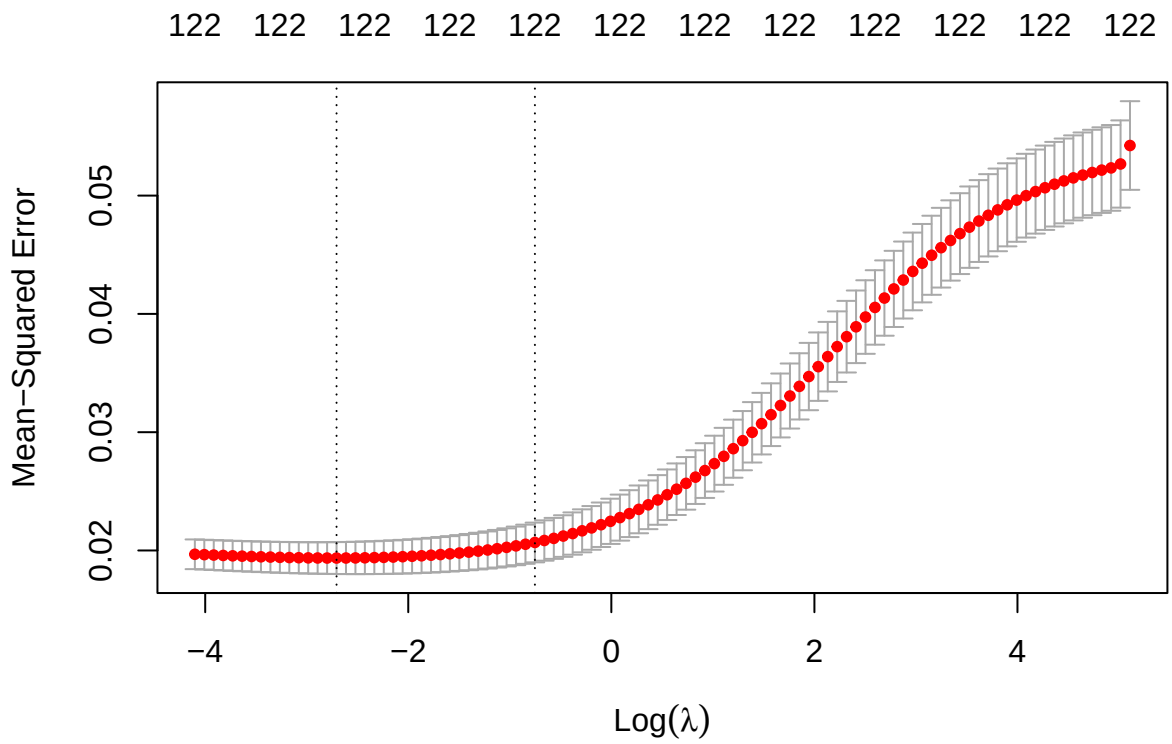
Clearly this gives us the ridge penalty for $\alpha = 0$ and the LASSO penalty for $\alpha = 1$, hence we proceed with $\alpha = 0$ and use the default 10-fold cross validation to find the best value for λ by which to multiply the penalty.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
ridgeCV = cv.glmnet(as.matrix(XTrain), yTrain, alpha=0) #k=10-fold cv by default
plot(ridgeCV)
```



This leads to an optimum lambda of 0.06677693.

```
optLambda = ridgeCV$lambda.min
optLambda
```

```
## [1] 0.06677693
```

Using this value of lambda we can then obtain predicted target values $\hat{y}$ and see that the MSE of these compared the the true target values y is roughly 0.0456.

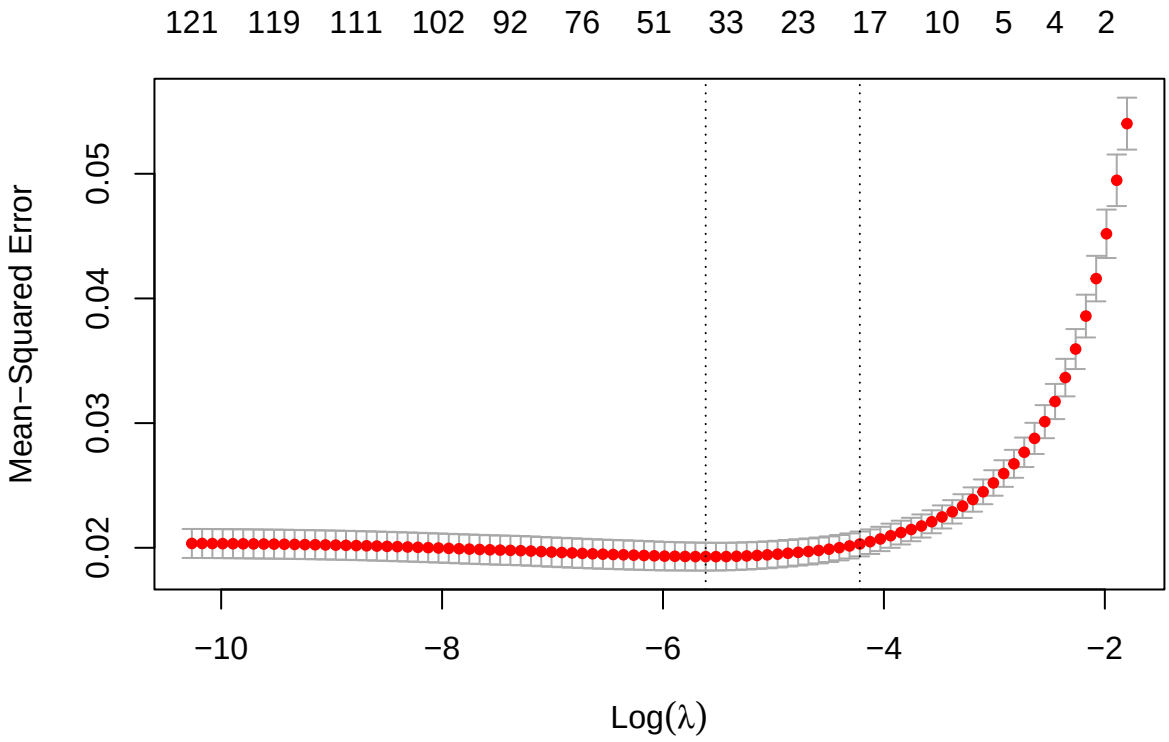
```
ridge = glmnet(as.matrix(XTrain), yTrain, alpha=0, lambda=optLambda)

yRidge = predict(ridge, as.matrix(XTest))
sum((yTest - yRidge)^2)/length(yTest)
```

```
## [1] 0.04559334
```

LASSO Regression

We now repeat the above but with $\alpha = 1$ to use LASSO regression instead of ridge regression.

```
lassoCV = cv.glmnet(as.matrix(XTrain), yTrain, alpha=1) # k=10-fold cv by default
plot(lassoCV)
```



(The ticks along the top of the plot denotes the number of non-zero coefficients for the corresponding value of λ.)

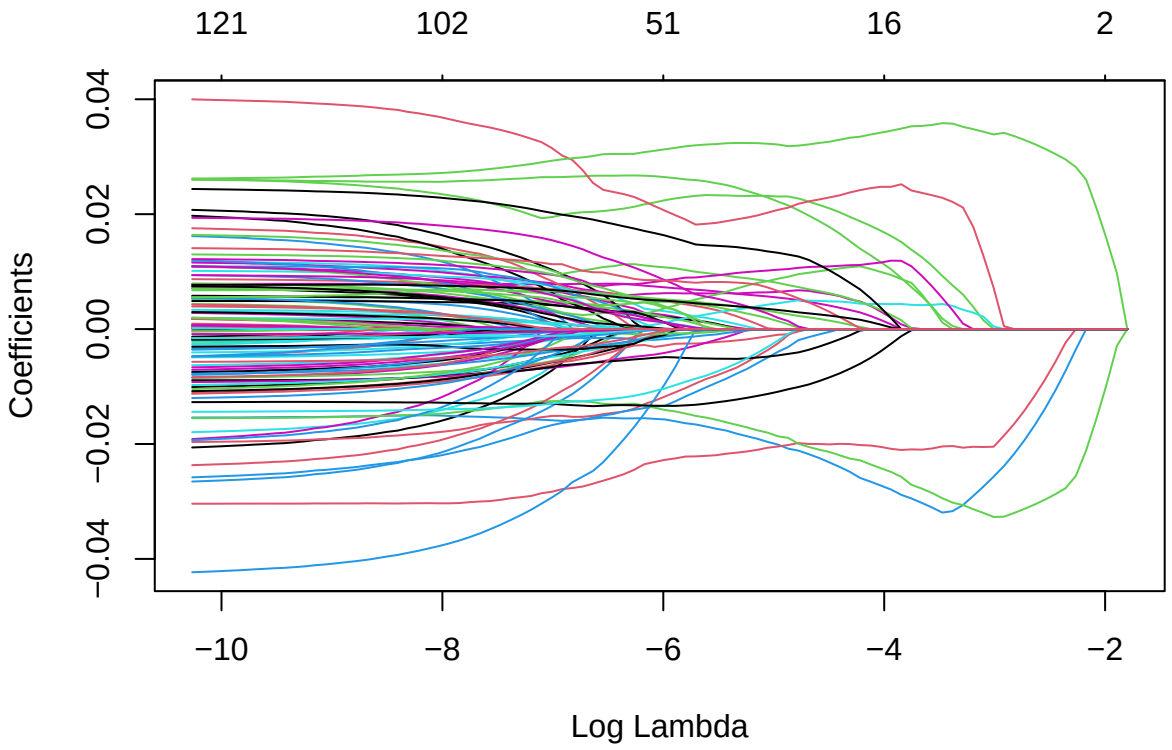
This then leads to the optimum lambda value of 0.003647541.

```
optLambda = lassoCV$lambda.min
optLambda
```

```
## [1] 0.003647541
```

We can also plot the LASSO path. Note that we are actually plotting the *scaled* values of our β estimate (given as γ in the notes), which we will rescale to obtain our actual β estimates later.

```
lasso = glmnet(XTrain, yTrain, alpha=1, lambda=optLambda)
plot(glmnet(XTrain, yTrain, alpha=1), xvar='lambda')
```



We see that many of the 122 coefficients are sent to zero when $\log \lambda < -4$, at which point the remaining 16 nonzero coefficients require more significant increases in $\log \lambda$ to be sent to zero, culminating in just 2 nonzero coefficients when $\log \lambda = -2$.

As with ridge regression, we now calculate the MSE of the LASSO model's predictions. This leads to a slightly better MSE of 0.0435 compared to ridge's 0.0456.

```
yLasso = predict(lasso, as.matrix(XTest))
sum((yTest - yLasso)^2)/length(yTest)
```

```
## [1] 0.04354691
```

Finally, we can rescale the coefficients to obtain our optimal β estimates for both ridge and LASSO regression as below. Here we see that ridge regression leads to no zero-valued coefficients, whilst LASSO leads to 83 zero-valued coefficients (this makes sense as an increased number of zero-valued coefficients is one of the main motivations behind LASSO regression).

```
ridgeBeta = D^(-1/2) * ridge$beta
lassoBeta = D^(-1/2) * lasso$beta

sum(ridgeBeta == 0); sum(lassoBeta == 0)
```

```
## [1] 0
```

```
## [1] 83
```

(But as mentioned before, the number of zero-coefficients would be the same if we checked the unscaled estimates too.)

```
sum(ridge$beta == 0); sum(lasso$beta == 0)
```

```
## [1] 0
```

```
## [1] 83
```

Task 2

In this task we will use the Bone Mineral Density dataset[1] and the `R` package `mgcv` to fit our model. This dataset contains 485 recordings of spinal bone mineral density for male and female subjects of varying ages.

```
X0 = read.table("boneMineralDensity.dat", header=T)
head(X0)
```

```
##   idnum   age gender      spnbmd
## 1     1 11.70   male 0.018080670
## 2     1 12.70   male 0.060109290
## 3     1 13.75   male 0.005857545
## 4     2 13.25   male 0.010263930
## 5     2 14.30   male 0.210526300
## 6     2 15.30   male 0.040843210
```

```
dim(X0)
```

```
## [1] 485   4
```

First we split our dataset into male and female observations since we will fit separate models for each.

```
maleX = X0[X0$gender=="male",]
femaleX = X0[X0$gender=="female",]
```

To fit out models we use the function `gam` with the formula `spnbmd ~ s(age, bs="cr")`: `spnbmd` gives us the variable of interest, the relative change in spinal BMD, whilst `s(age, bs="cr")` tells us that we want to fit a spline using the `age` variables over a basis (`bs`) of cubic regression splines (`cr`). This function uses generalised cross validation by default.

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
maleSpline = gam(spnbmd ~ s(age, bs="cr"), data=maleX) # uses gcv by default
femaleSpline = gam(spnbmd ~ s(age, bs="cr"), data=femaleX)

summary(maleSpline)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## spnbmd ~ s(age, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.039625   0.002748   14.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df     F p-value
## s(age) 4.567  5.556 15.97  <2e-16 ***
## ---
```

[1] https://hastie.su.domains/ElemStatLearn/

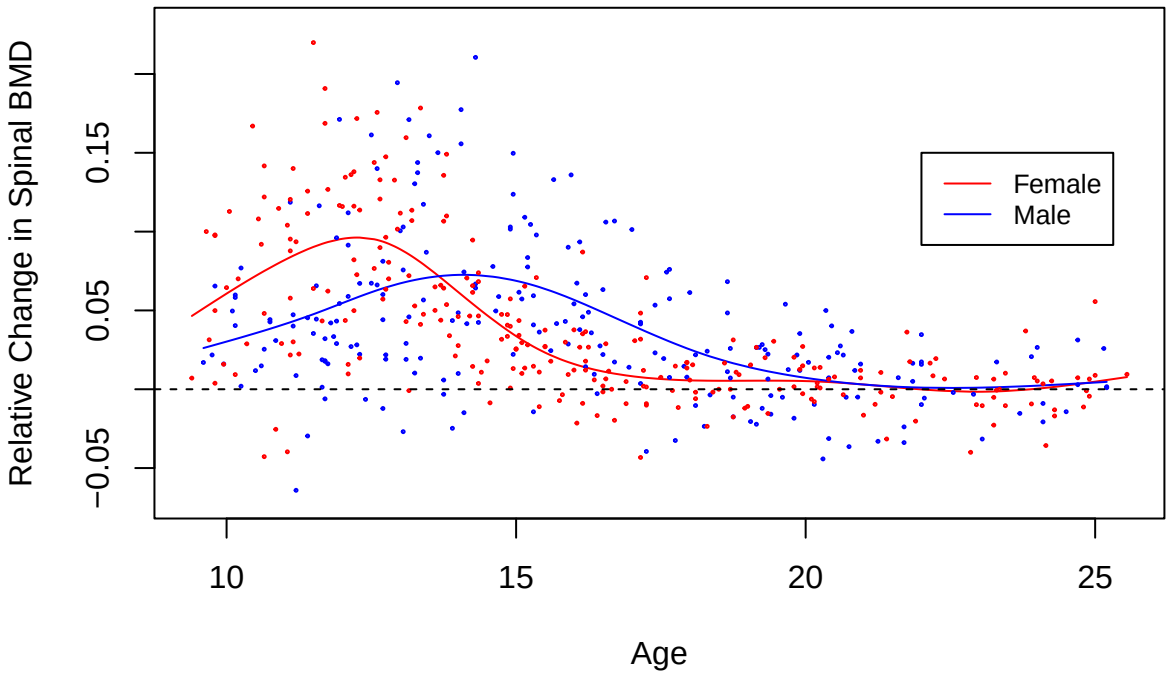
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.282   Deviance explained = 29.6%
## GCV = 0.0017492  Scale est. = 0.0017061  n = 226
```

```
summary(femaleSpline)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## spnbmd ~ s(age, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.038926   0.002176   17.89   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df     F p-value
## s(age) 6.354   7.47 37.29  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.519   Deviance explained = 53.1%
## GCV = 0.0012617  Scale est. = 0.0012258  n = 259
```

Below we then plot the two models along with the datapoints with female data in red and male in blue in order to recreate Figure 5.6 from *The Elements of Statistical Learning*. However, note that unlike Figure 5.6, we have different values of λ for the male and female splines, as found through generalised cross validation, rather than setting $\lambda \approx 0.00022$ for both models. This has led to a slightly different plot here, though one showing largely the same behaviour as Figure 5.6.

```
# Get the increasing-age order of rows so that we can plot the splines properly
maleAgesOrder = order(maleX$age)
femaleAgesOrder = order(femaleX$age)

# Plot the splines
plot(femaleX$age[femaleAgesOrder], femaleSpline$fitted.values[femaleAgesOrder],
     col="red", type="l", xlab="Age",
     ylab="Relative Change in Spinal BMD", ylim=c(-0.07,0.23))
lines(maleX$age[maleAgesOrder] ,maleSpline$fitted.values[maleAgesOrder], col="blue")

# Plot the data points
points(maleX$age, maleX$spnbmd, col="blue", cex=0.2)
points(femaleX$age, femaleX$spnbmd, col="red", cex=0.2)

# Add a dotted x=0 line and a legend
abline(0,0, lty=2)
legend(22, 0.15, legend=c("Female", "Male"),
       col=c("red", "blue"), lty=1, cex=0.8)
```


Relative Change in Spinal BMD
−0.05
0.05
0.15
10
15
20
25
Age
Female
Male