Stat 6021 R Tutorial: Shrinkage Methods & PCA

For this tutorial, you will see two examples, one on shrinkage methods, and another on Principal Component Analysis (PCA). For both examples, we will use the `mtcars` data set in R. The data were taken from the 1974 Motor Trend US magazine, and compares fuel efficiency along with 10 characteristics of automobile design and performance for 32 automobiles.

Given that this is a small data set with 32 observations, and 10 predictors, we are concerned that the variance associated with the regression model will be large. Furthermore, creating a scatterplot matrix of the predictors show that a number of predictors are highly correlated with one another, for example, the engine displacement is correlated with the number of cylinders, the horsepower, rear axle ratio, and weight, so multicollinearity is a concern.

1. In this first example, we will use ridge regression

   (a) Ridge regression. Ridge regression allows us to introduce some bias into our model while possibly reducing the variance of the model substantially. The `glmnet()` function from the library `glmnet` allows us to fit either a ridge or lasso regression. To do so, use the following code.

   ```
   library(glmnet)
   x<-model.matrix(mpg~.,mtcars)[,-1]
   y<-mtcars$mpg
   ridge.r<-glmnet(x,y,alpha=0, lambda=0, thresh = 1e-14)
   coefficients(ridge.r)
   ```

   It is important to note the following:

   - The `glmnet()` function cannot handle categorical variables that are not dummy coded. The function `model.matrix()` should be used to specify the predictor variables. `model.matrix()` will form the design matrix for the specified regression and convert the categorical predictors into dummy codes.
   - Notice that with the `model.matrix()` function, the first column is removed. The first column of a design matrix is a column of 1s, needs to be removed to use the `glmnet()` function.

- In the `glmnet()` function, the argument `alpha` is used to specify if we want to use ridge or lasso. Use 0 for ridge and 1 for lasso.
- In this example, I put the argument `lambda` as 0; we know this will fit a OLS regression.
- The argument `thresh` must be set to be a very small value, especially if multicollinearity is present. Try changing this argument to another value, for example `1e-04` and see what happens to the coefficients.

Compare the coefficients from this ridge regression from OLS regression. They should be the same, theoretically.

(b) Choosing the tuning parameter. The value of λ is chosen by cross-validation (CV). We want to find the value of λ does minimizes the mean-squared error (MSE) of the model.

i. First, we randomly split the data into a training and testing set.

```
set.seed(12)
train<-sample(1:nrow(x), nrow(x)/2)
test<-(-train)
y.test<-y[test]
```

ii. Based on the training set, use CV to find the optimal value of λ, the value that minimizes the MSE.

```
set.seed(12)
cv.out<-cv.glmnet(x[train,],y[train],alpha=0)
bestlam<-cv.out$lambda.min
bestlam
plot(cv.out)
```

What is the optimal value of λ? What is the plot produced from `plot(cv.out)` telling us?

(c) Comparing MSEs. Next, we compare the test MSEs of the model using the optimal value of λ with the model using OLS.

```
ridge.mod<-glmnet(x[train,],y[train],alpha=0,lambda=bestlam, thresh = 1e-14)

##Test MSE with best lambda
ridge.pred<-predict(ridge.mod,s=bestlam,newx=x[test,])
mean((ridge.pred-y.test)^2)

##fit OLS by setting lambda=0
ridge.mod.0<-glmnet(x[train,],y[train],alpha=0,lambda=0, thresh = 1e-14)

##test MSE with lambda=0
ridge.pred.0<-predict(ridge.mod.0,newx=x[test,])
mean((ridge.pred.0-y.test)^2)
```

(d) Comparing estimated coefficients from ridge and OLS regression. We can make this comparison with the following code. Note that I used all the observations in this next block of code.

```
out.ridge<-glmnet(x,y,alpha=0,lambda=bestlam,thresh = 1e-14)
out.ols<-glmnet(x,y,alpha=0, lambda=0, thresh = 1e-14)
cbind(coefficients(out.ridge), coefficients(out.ols))
```

Notice how the coefficients using ridge regression are generally smaller. Hence the term "shrinkage methods".

(e) Ridge plot. We can create a plot to show how the values of the coefficients change as we change the value of λ.

```
grid<-10^seq(10,-2,length=100)
out.all<-glmnet(x,y,alpha=0,lambda=grid,thresh = 1e-14)
plot(out.all, xvar = "lambda")
abline(v=log(bestlam), lty=2)
legend("bottomright", lwd = 1, col = 1:6, legend = colnames(x), cex = .7)
```

2. In this second example, we will use PCA on the quantitative variables.

(a) PCA. The function `prcomp()` carries out PCA in R. Note that categorical variables must be removed in PCA. The argument `scale=TRUE` indicates we want the function to scale the variables for us. This should be done especially if the variables are all on different scales.

```
pr.out<-prcomp(mtcars[,c(-8,-9)], scale=TRUE)
```

(b) Extracting information from `prcomp()`.

i. Typing `pr.out$center` and `pr.out$scale` gives the means and standard deviations of each variable.

ii. Typing `pr.out$rotation` gives the loading vector for each of the Principal Components (PCs). How would you interpret the first two PCs?

iii. Typing `pr.out$sdev` gives the standard deviation of each PC. To find the proportion of variance in our data that can be explained by each PC, we can type

```
##variance of each PC
pr.var<-pr.out$sdev^2
pr.var
##proportion of variance in data explained by each PC
pve<-pr.var/sum(pr.var)
pve
```

How would you interpret these values?

(c) Graphical summaries from PCA. These plots are generally useful in visualizing the output from PCA.

i. Plotting the PCs. The `biplot()` function plots the first two PCs.

```
biplot(pr.out, scale=0)
```

How do we interpret this plot?

ii. Scree plots. A scree plot plots the proportion of variation explained by each PC.

```
plot(pve, ylim=c(0,1))
```

By convention, the y-axis ranges from 0 to 1, since the y-axis represents a proportion. How do we interpret this plot?
Another plot that is similar is a plot that shows the cumulative proportion of variance explained by the PCs.

```
plot(cumsum(pve), ylim=c(0,1))
```