Ridge Regression Lab

Group 2-07

Dataset Hitters (from ISLR)

- Hitters is a dataset with 263 observations of 20 variables. Here, we take the last variable, salary, as our response and the other 19 variables as predictors.

```
library("ISLR", "glmnet")
Hitters <- na.omit(Hitters)
x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary
```

- Split the sample by 50/50:

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

glmnet()

- *x* & *y*
- elasticnet mixing parameter $\frac{1-\alpha}{2}||\beta||_2^2 + \alpha||\beta||_1$ where $\alpha \in [0, 1]$
  ($\alpha = 0$ for Ridge; $\alpha = 1$ for Lasso)

```r
ridge.mod <- glmnet(x,y,alpha=0)
```

- Note that the *glmnet()* function standardizes the variables so that they are on the same scale. To turn off this default setting, use the argument **standardize=FALSE**.

Choose λ by cross-validation

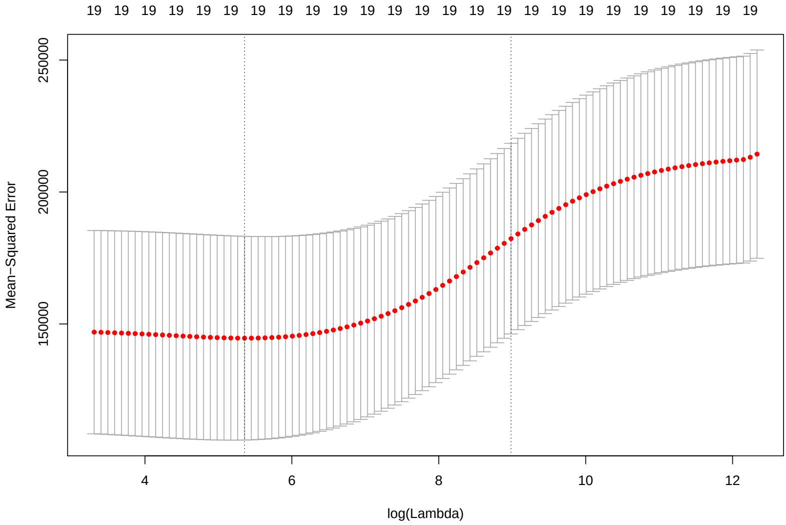
- By default the function performs 10-fold cross-validation, though this can be changed using the argument *n*folds.
- the *lambda.min* is the lambda that gives *min(MSE - length(lambda))*

```
set.seed(1)
cv.out <- cv.glmnet(x[train,],y[train],alpha=0)
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 211.7416
```

Plot of cv.out

```
plot(cv.out)
```



Final Ridge Regression Model

```r
ridge.pred <- predict(ridge.mod,s=bestlam,newx=x[test,])
predict(ridge.mod,type="coefficients",s=bestlam)[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun
##   9.88487157   0.03143991   1.00882875   0.13927624   1.
##          RBI        Walks        Years       CAtBat
##   0.87318990   1.80410229   0.13074381   0.01113978   0.
##       CHmRun        CRuns         CRBI       CWalks
##   0.45158546   0.12900049   0.13737712   0.02908572  27.
##    DivisionW      PutOuts      Assists       Errors   Ne
## -91.63411299   0.19149252   0.04254536  -1.81244470   7.
```

- Ridge regression does **not** perform variable selection!

Benefit to performing ridge regression with bestlam

```
ridge.train <- glmnet(x[train,],y[train],alpha = 0)
```

- Remember that when $\lambda = 0$, we are not doing shrinkage regression but least square regression.

```
ridge.pred <- predict(ridge.train,s=0,newx=x[test,],
                      x=x[train,],y=y[train])
#mean((ridge.pred-y.test)^2)
```

- With bestlam

```
ridge.pred <- predict(ridge.train,s=bestlam,newx=x[test,],
                      x=x[train,],y=y[train])
#mean((ridge.pred-y.test)^2)
```

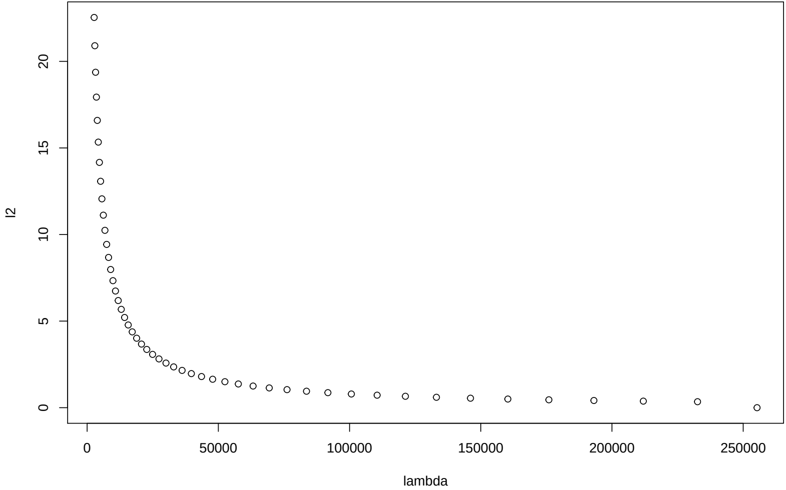
cont.

- In general, if we want to fit a (unpenalized) least squares model, then we should use the lm() function, since that function provides more useful outputs, such as standard errors and p-values for the coefficients.

```
coef(lm(y~x, subset=train))
predict(ridge.mod,s=0,exact=T,type="coefficients",x=x[train,],y=y[train])
```

Comparing different λ and their l2 norm

```
plot(lambda,l2)
```



What's the difference between coefficients?

```
lambda_large<- coef(ridge.mod)[,1]
lambda_small<- coef(ridge.mod)[,50]
cbind(lambda_large,lambda_small)
```

```
##              lambda_large  lambda_small
## (Intercept)  5.359259e+02 213.066443434
## AtBat        1.221172e-36   0.090095728
## Hits         4.429736e-36   0.371252756
## HmRun        1.784944e-35   1.180126956
## Runs         7.491019e-36   0.596298287
## RBI          7.912870e-36   0.594502390
## Walks        9.312961e-36   0.772525466
## Years        3.808598e-35   2.473494238
## CAtBat       1.048494e-37   0.007597952
## CHits        3.858759e-37   0.029272172
## CHmRun       2.910036e-36   0.217335716
## CRuns        7.741531e-37   0.058705097
## CRBI         7.989430e-37   0.060722036
```

Another example in classification

- Setting up

```
yclass <- rep("Yes", length(y))
yclass[y < median(y)] <- "No"
yclass <- factor(yclass)
yclass.test <- yclass[test]
```

- Use Logistic Regression to compare coefficients
  Recall glm() uses **family="binomial"**, and call **type = "response"** in predict();
  coef_glm <- logistic.mod.class$coefficients

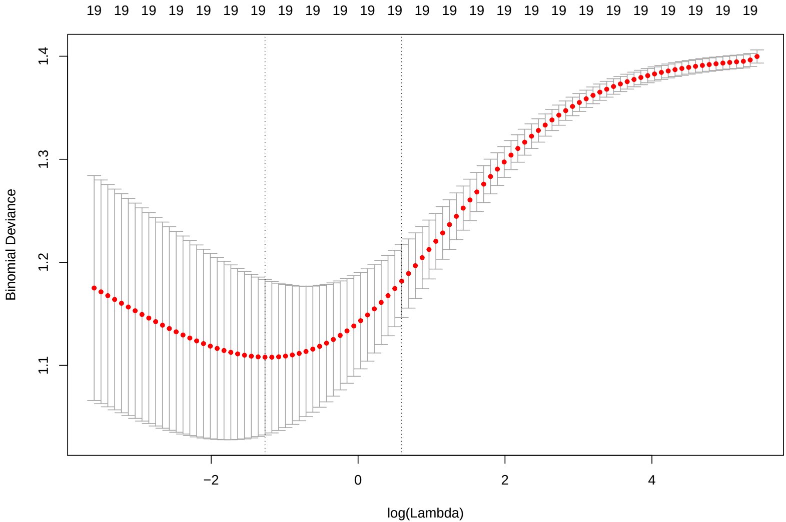
Ridge regression (classification)

- Again, override family as binomial in both glmnet() and cv.glmnet()

```
ridge.mod.class <- glmnet(x, yclass, alpha=0,
                          family="binomial")
cv.out.class <- cv.glmnet(x[train, ], yclass[train],
                          alpha=0, family="binomial")
bestlam <- cv.out.class$lambda.min
ridge.pred.class <- predict(ridge.mod.class, s=bestlam,
                            newx=x[test, ], type="class")
error.rate.ridge <- mean(ridge.pred.class != yclass.test)
ridge.coefficients <- predict(ridge.mod.class,
                              type="coefficients",
                              s=bestlam)[1:20, ]
```

Plot for cv.out.class

```r
plot(cv.out.class)
```



Comparison

- Error rate

```
cbind(error.rate.logistic,error.rate.ridge)
```

```
##      error.rate.logistic error.rate.ridge
## [1,]           0.2348485        0.1590909
```

- Coefficients

```
cbind(coef_glm,ridge.coefficients)
```

```
##                 coef_glm ridge.coefficients
## (Intercept) -3.551364583      -2.822169e+00
## AtBat       -0.006591710       8.156105e-04
## Hits         0.036790282       3.767020e-03
## HmRun        0.043504900       2.264340e-03
## Runs        -0.064130104       3.922014e-03
## RBI          0.007675413       3.281934e-03
```