STAT432_HW6

Taiga Hasegawa(taigah2)

2019/3/4

Question1

```
#fix scale
data(Boston, package="MASS")
useLog = c(1,3,5,6,8,9,10,14)
Boston[,useLog] = log(Boston[,useLog])
Boston[,2] = Boston[,2] / 10
Boston[,7] = Boston[,7]^2.5 / 10^4
Boston[,11] = exp(0.4 * Boston[,11])/1000
Boston[,12] = Boston[,12] / 100
Boston[,13] = sqrt(Boston[,13])
```

```
numcol=dim(Boston)[1]
#sequence k values such that the degrees of freedom is within the range of 1 to 14
#degree of freedom of knn is N/k
k=numcol:(numcol/14)
```

```
X=Boston[,1:13]
#centering
cx <- sweep(X, 2, colMeans(X), "-")
#SVD
svd=svd(cx)
#d_j^2
d=svd$d^2
#when lambda=0, degree of freedom is 13+1 (intercept)
sum(d/(d+0))
```

```
## [1] 13
```

```
#when lambda=8583, degree of freedom is 0+1 (intercept)
sum(d/(d+8583))
```

```
## [1] 0.9999942
```

```
#This is the range of lambda in which degree of freedom takes 1 to 14
lambda=0:8583
```

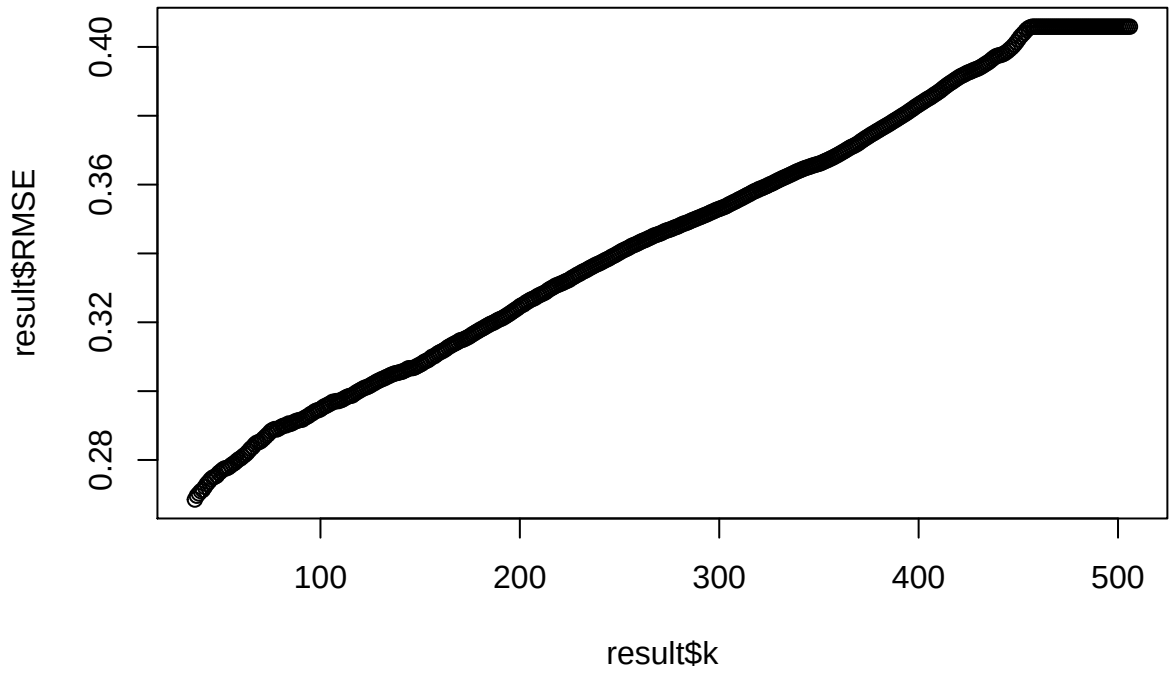
```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
y=Boston[,14]
X$chas=as.factor(X$chas)
TrainData = data.frame(X)
#cross validation to find the best k
knnFit <- train(TrainData, y,
                method = "knn",
                tuneGrid=data.frame("k"=k),
```

```
                  trControl = trainControl(method = "cv",number = 10))
result=knnFit$results
#k vs RMSE
plot(result$k,result$RMSE)
```




```
#Best k
knnFit$bestTune
```

```
##    k
## 1 37
```

It turned out k=37 is the best and in this case, the degree of freedom is 14.

```
library(glmnet)
```

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

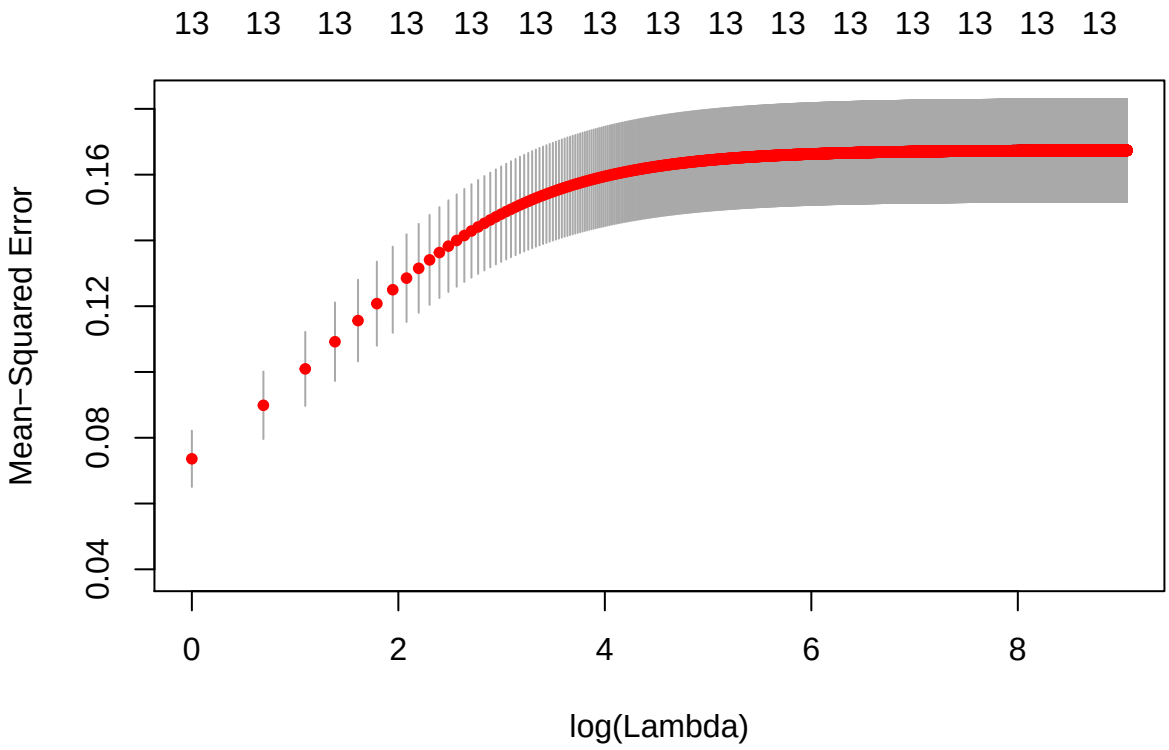
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
set.seed(100)
#cross validation to find best lambda
fit1 = cv.glmnet(data.matrix(X), y, nfolds = 10, alpha = 0, lambda = lambda)
#the best lambda
fit1$lambda.min
```

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

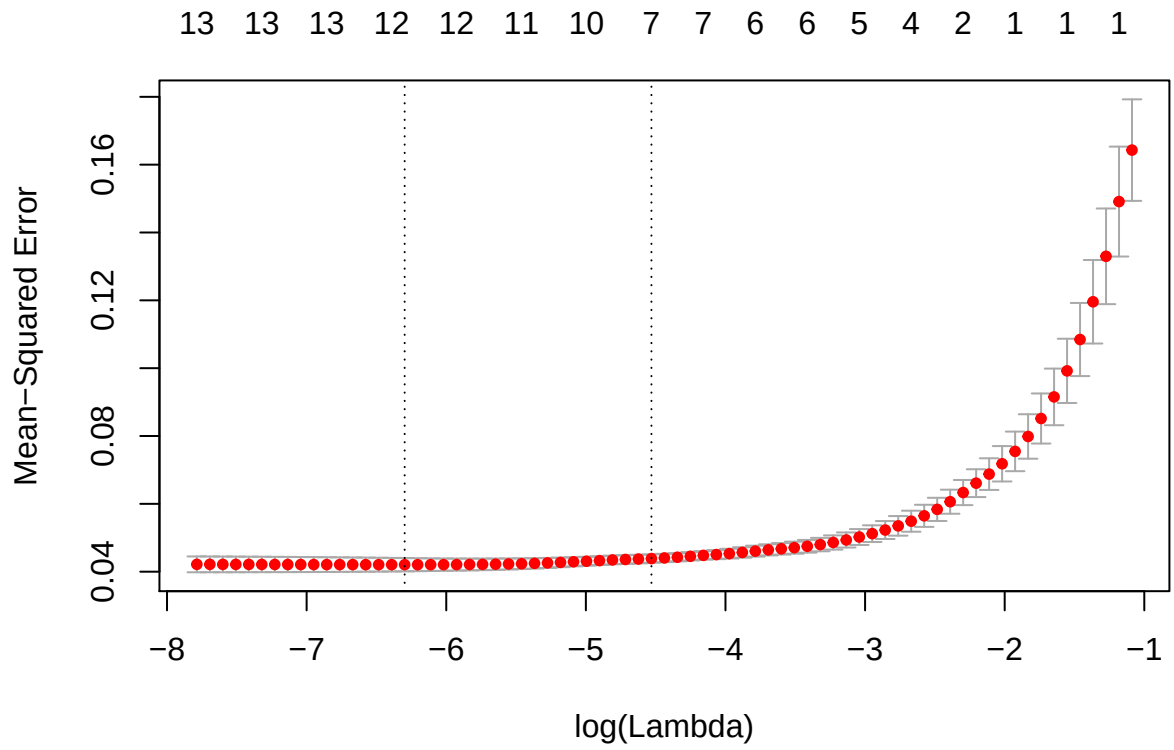
```
#lambda vs cvm
plot(fit1)
```



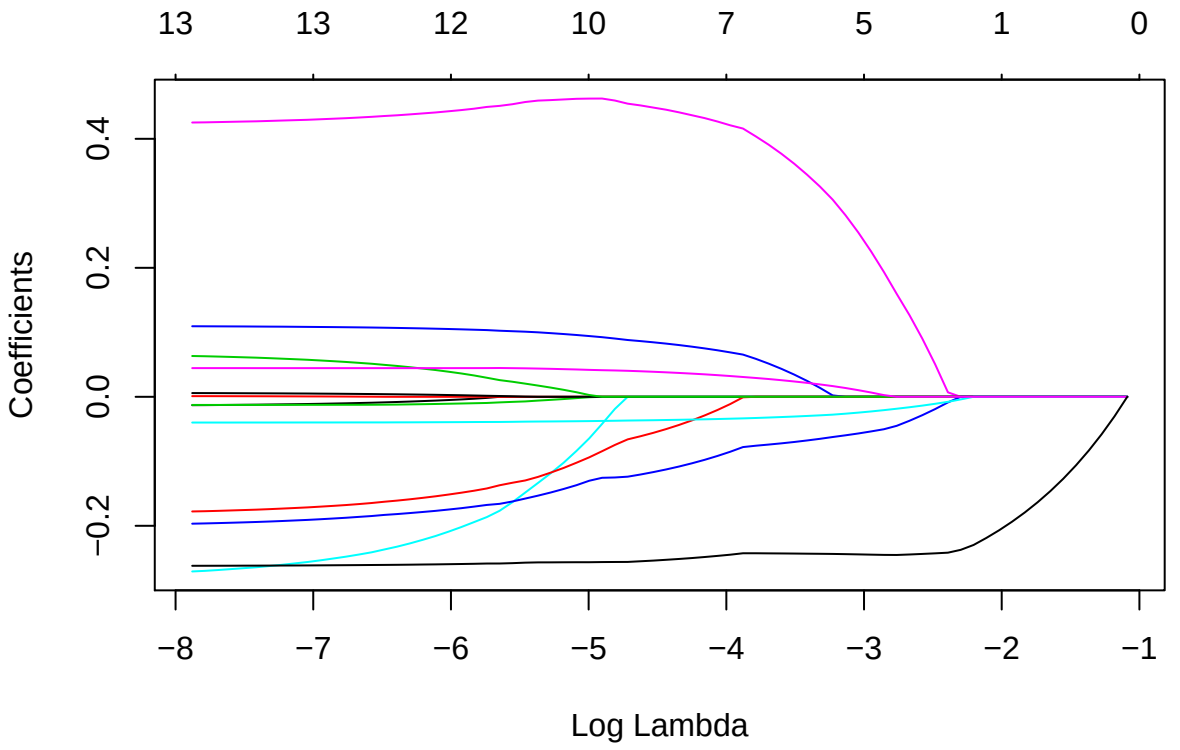
It turned out the best lambda is 0 and in this case, the degree of freedom is 14. So both knn and ridge regression prefer 14 degree of freedom.

Question2

```
set.seed(100)
fit2 = cv.glmnet(data.matrix(X), y, nfolds = 3, alpha = 1,family = "gaussian")
#plotting the cross-validation errors
plot(fit2)
```




```
#how the estimated parameters change as a function of
plot(fit2$glmnet.fit, "lambda")
```




```
#the selected variables
coef(fit2, s = "lambda.min")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                        1
## (Intercept)  3.969795664
## crim        -0.007380514
## zn           .
## indus       -0.011768357
## chas         0.106429348
## nox         -0.227285251
## rm           0.437888034
## age          0.003627727
## dis         -0.158903808
## rad          0.045865054
## tax         -0.180270263
## ptratio     -0.039572115
## black        0.044422323
## lstat       -0.260280792
```

```
lambdamin=fit2$lambda.min
```

As you can see in the plotting, the best tuning was 0.0018402. The varaibles other than zn were selected by the lasso regression.

```
lmfit = lm(medv~., Boston)
#subset selection with AIC penalty
stepaic<-step(lmfit, direction="backward", trace=0)
#the selected variables
paste(variable.names(stepaic),collapse = ' + ')
```

```
## [1] "(Intercept) + chas + nox + rm + dis + rad + tax + ptratio + black + lstat"
```

On the other hand, the above variables were selected based on AIC penalty.

Trade of bias-variance in lasso regression was done by just shrinken toward zero but the one in AIC was done by adding 2p (the double of the number of predictors). So when the number of predictors is increasing, AIC tends to select smaller model.

Question3

```
#the first case
#binomial likelihood
binom.test(3, 12, p=0.5, alternative="less")
```

```
##
##  Exact binomial test
##
## data:  3 and 12
## number of successes = 3, number of trials = 12, p-value = 0.073
## alternative hypothesis: true probability of success is less than 0.5
## 95 percent confidence interval:
##  0.0000000 0.5273266
## sample estimates:
## probability of success
##                   0.25
```

```
#second case
#negative binomial likelihood
negative_binomial=function(n,r,p){
```

```
  gamma(n)/(gamma(r)*gamma(n-r+1))*0.5^n
}
#P(n>=12|r=3,p=0.5)
sum(negative_binomial(12:50,r=3,p=0.5))
```

```
## [1] 0.03271484
```

The p-value was affected by the irrelevant information of stopping rule because the likelihood was changed from first case to second case but in fact stopping rule is not relevant to p-values.