JieTang_Lab4.R

tjj

2020-07-21

```r
#Name:Jie Tang
#Course:Machine learning using R 374815
#Quarter:Summer
#Insturctor name : Michael Chang
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
#Quiz part
library(ISLR)
library(leaps)
fix(Hitters)
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"
## [19] "Salary"    "NewLeague"
```

```r
dim(Hitters)
```

```
## [1] 322  20
```

```r
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

```r
Hitters=na.omit(Hitters)
dim(Hitters)
```

```
## [1] 263  20
```

```r
sum(is.na(Hitters))
```

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

```r
#Q1 compared best 7 variabels and best 8 variables
regfit.full=regsubsets(Salary~.,Hitters)
coef(regfit.full,7)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits       CHmRun
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073    1.4420538
##    DivisionW      PutOuts
## -129.9866432    0.2366813
```

```r
coef(regfit.full,8)
```

```
##  (Intercept)        AtBat         Hits        Walks       CHmRun        CRuns
##  130.9691577   -2.1731903    7.3582935    6.0037597    1.2339718    0.9651349
##       CWalks    DivisionW      PutOuts
##   -0.8323788 -117.9657795    0.2908431
```

```r
#Q2
reg.summary=summary(regfit.full)
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```r
#check ADJUSTED R-squared and BIC(missed adjusted at first attempt)
reg.summary$adjr2
```

```
## [1] 0.3188503 0.4208024 0.4450753 0.4672734 0.4808971 0.4972001 0.5007849
## [8] 0.5137083
```

```r
which.max(reg.summary$adjr2)
```

```
## [1] 8
```

```r
reg.summary$bic
```

```
## [1]  -90.84637 -128.92622 -135.62693 -141.80892 -144.07143 -147.91690 -145.25594
## [8] -147.61525
```

```r
which.min(reg.summary$bic)
```

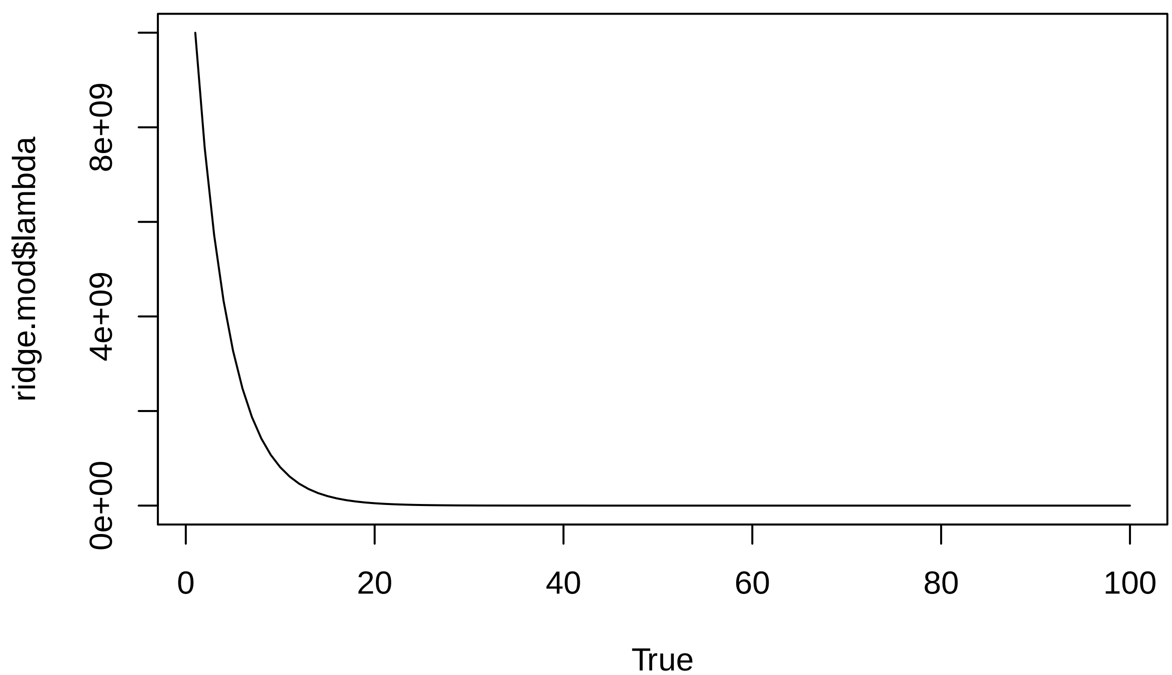
```
## [1] 6
```

```r
#Q3
#If alpha=0 then a ridge regression model is fit
#if alpha = 1 then a lasso model is fit.
#Q4
x=model.matrix(Salary~.,Hitters)[,-1]
y=Hitters$Salary
#check graph
library(glmnet)
```

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

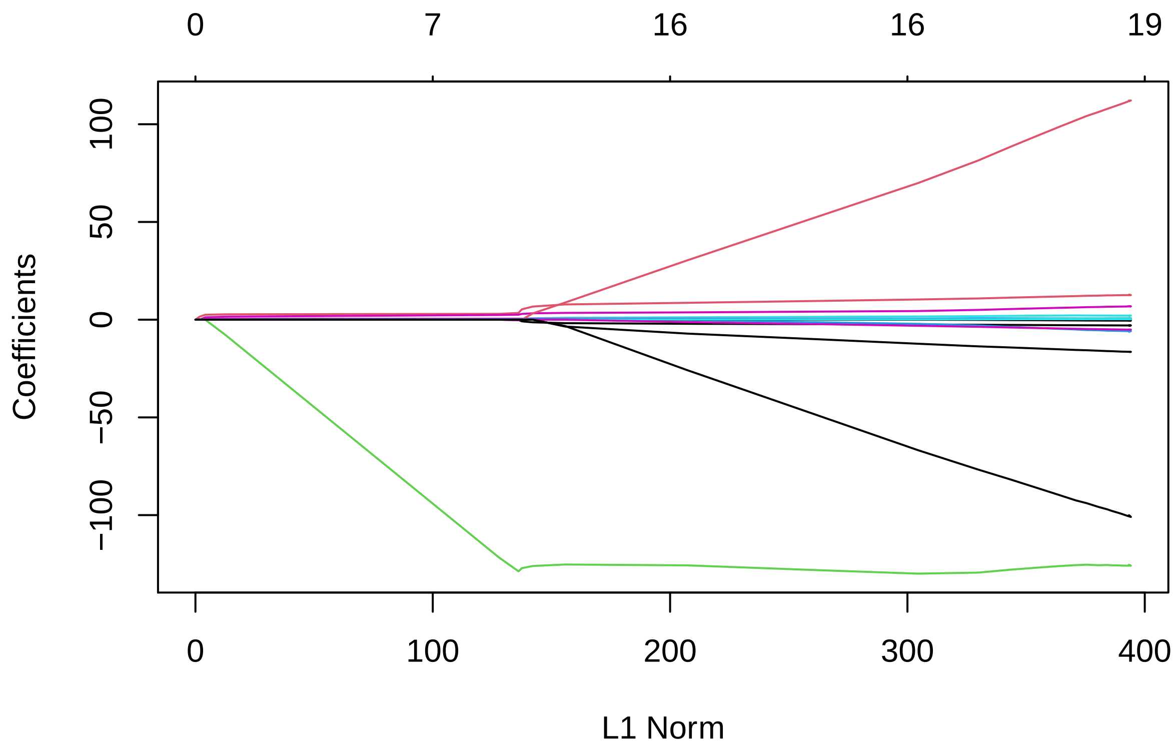
```
## Loaded glmnet 4.0-2
```

```
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
plot(ridge.mod$lambda,xlab="True",type="l")
```




```
#Q5
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]
lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
plot(lasso.mod)
```

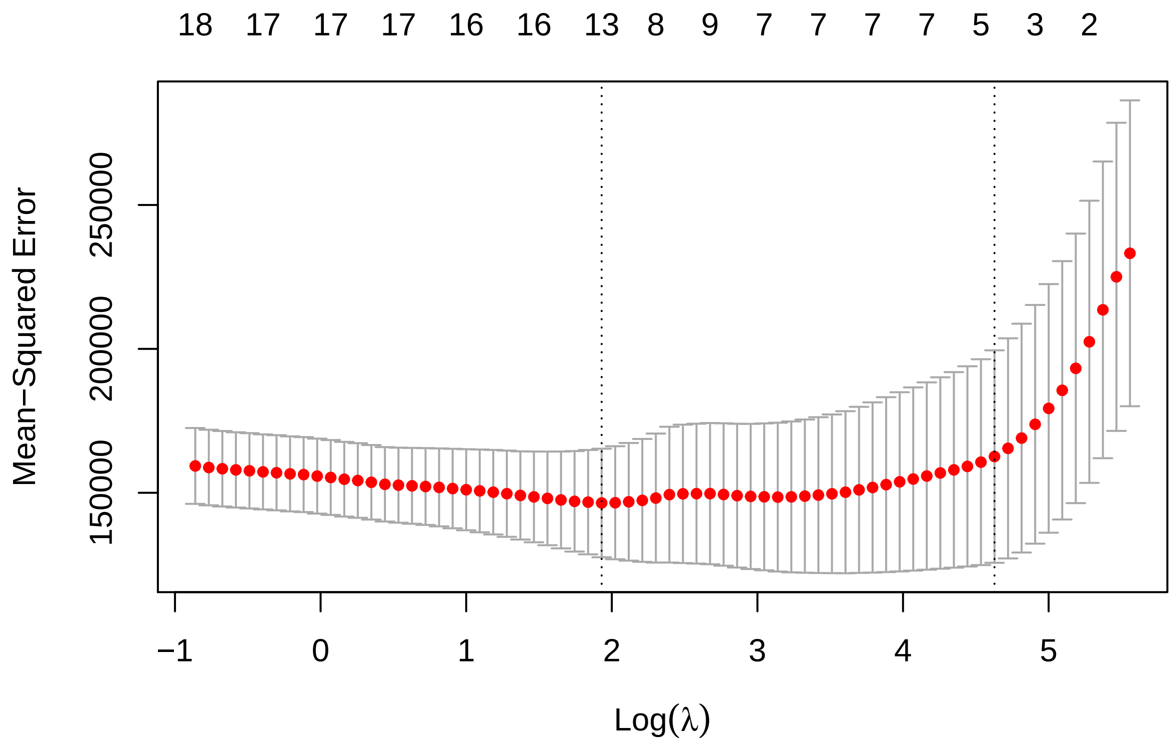
```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
RNGkind(sample.kind = "Rounding")
```

```
## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
```




```r
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 6.891469
```

```r
#Q6
#Check how many elements in that vector
lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda = grid)
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred - y.test)^2) # 100743.4
```

```
## [1] 104513.6
```

```r
out <- glmnet(x, y, alpha = 1,lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:20,]
lasso.coef
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##   26.40141879   -0.40525834    3.09714299    0.00000000    0.00000000
##           RBI         Walks         Years        CAtBat         CHits
##    0.00000000    2.73465290   -2.79191507    0.00000000    0.00000000
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##    0.10043903    0.29607379    0.41898361   -0.08181301   25.16447958
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -118.21661332    0.24452197    0.00000000   -0.77307111    0.00000000
```

```r
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)         AtBat          Hits         Walks         Years
##   26.40141879   -0.40525834    3.09714299    2.73465290   -2.79191507
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##    0.10043903    0.29607379    0.41898361   -0.08181301   25.16447958
##     DivisionW       PutOuts        Errors
## -118.21661332    0.24452197   -0.77307111
```

```r
#7
#By the result, we can see after 4 comps, the variance reaches 79.03
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
set.seed(2)
pcr.fit=pcr(Salary~., data=Hitters,scale=TRUE,validation="CV")
summary(pcr.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    348.9    352.2    353.5    352.8    350.1    349.1
## adjCV          452    348.7    351.8    352.9    352.1    349.3    348.0
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       349.6    350.9    352.9     353.8     355.0     356.2     363.5
## adjCV    348.5    349.8    351.6     352.3     353.4     354.5     361.6
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        355.2     357.4     347.6     350.1     349.2     352.6
## adjCV     352.8     355.2     345.5     347.6     346.7     349.8
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
## Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
##         16 comps  17 comps  18 comps  19 comps
## X          99.89     99.97     99.99    100.00
## Salary     53.01     53.85     54.61     54.61
```

```
# Chapter 6 Lab 1: Subset Selection Methods

# Best Subset Selection
#
# library(ISLR)
# fix(Hitters)
# names(Hitters)
# dim(Hitters)
# sum(is.na(Hitters$Salary))
# Hitters=na.omit(Hitters)
# dim(Hitters)
# sum(is.na(Hitters))
# library(leaps)
# regfit.full=regsubsets(Salary~.,Hitters)
# summary(regfit.full)
# regfit.full=regsubsets(Salary~.,data=Hitters,nvmax=19)
# reg.summary=summary(regfit.full)
# names(reg.summary)
# reg.summary$rsq
# par(mfrow=c(2,2))
# plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
# plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
# which.max(reg.summary$adjr2)
# points(11,reg.summary$adjr2[11], col="red",cex=2,pch=20)
# plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
# which.min(reg.summary$cp)
# points(10,reg.summary$cp[10],col="red",cex=2,pch=20)
# which.min(reg.summary$bic)
# plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
# points(6,reg.summary$bic[6],col="red",cex=2,pch=20)
# plot(regfit.full,scale="r2")
# plot(regfit.full,scale="adjr2")
# plot(regfit.full,scale="Cp")
# plot(regfit.full,scale="bic")
# coef(regfit.full,6)
#
# # Forward and Backward Stepwise Selection
#
# regfit.fwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="forward")
# summary(regfit.fwd)
# regfit.bwd=regsubsets(Salary~.,data=Hitters,nvmax=19,method="backward")
# summary(regfit.bwd)
# coef(regfit.full,7)
# coef(regfit.fwd,7)
# coef(regfit.bwd,7)
#
# # Choosing Among Models
#
# set.seed(1)
# train=sample(c(TRUE,FALSE), nrow(Hitters),rep=TRUE)
# test=(!train)
# regfit.best=regsubsets(Salary~.,data=Hitters[train,],nvmax=19)
# test.mat=model.matrix(Salary~.,data=Hitters[test,])
```

```
# val.errors=rep(NA,19)
# for(i in 1:19){
#   coefi=coef(regfit.best,id=i)
#   pred=test.mat[,names(coefi)]%*%coefi
#   val.errors[i]=mean((Hitters$Salary[test]-pred)^2)
# }
# val.errors
# which.min(val.errors)
# coef(regfit.best,10)
# predict.regsubsets=function(object,newdata,id,...){
#   form=as.formula(object$call[[2]])
#   mat=model.matrix(form,newdata)
#   coefi=coef(object,id=id)
#   xvars=names(coefi)
#   mat[,xvars]%*%coefi
# }
# regfit.best=regsubsets(Salary~.,data=Hitters,nvmax=19)
# coef(regfit.best,10)
# k=10
# set.seed(1)
# folds=sample(1:k,nrow(Hitters),replace=TRUE)
# cv.errors=matrix(NA,k,19, dimnames=list(NULL, paste(1:19)))
# for(j in 1:k){
#   best.fit=regsubsets(Salary~.,data=Hitters[folds!=j,],nvmax=19)
#   for(i in 1:19){
#     pred=predict(best.fit,Hitters[folds==j,],id=i)
#     cv.errors[j,i]=mean( (Hitters$Salary[folds==j]-pred)^2)
#   }
# }
# mean.cv.errors=apply(cv.errors,2,mean)
# mean.cv.errors
# par(mfrow=c(1,1))
# plot(mean.cv.errors,type='b')
# reg.best=regsubsets(Salary~.,data=Hitters, nvmax=19)
# coef(reg.best,11)
#
#
# # Chapter 6 Lab 2: Ridge Regression and the Lasso
#
# x=model.matrix(Salary~.,Hitters)[,-1]
# y=Hitters$Salary
#
# # Ridge Regression
#
# library(glmnet)
# grid=10^seq(10,-2,length=100)
# ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
# dim(coef(ridge.mod))
# ridge.mod$lambda[50]
# coef(ridge.mod)[,50]
# sqrt(sum(coef(ridge.mod)[-1,50]^2))
# ridge.mod$lambda[60]
# coef(ridge.mod)[,60]
```

```
# sqrt(sum(coef(ridge.mod)[-1,60]^2))
# predict(ridge.mod,s=50,type="coefficients")[1:20,]
# set.seed(1)
# train=sample(1:nrow(x), nrow(x)/2)
# test=(-train)
# y.test=y[test]
# ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)
# ridge.pred=predict(ridge.mod,s=4,newx=x[test,])
# mean((ridge.pred-y.test)^2)
# mean((mean(y[train])-y.test)^2)
# ridge.pred=predict(ridge.mod,s=1e10,newx=x[test,])
# mean((ridge.pred-y.test)^2)
# ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T,x=x[train,],y=y[train])
# mean((ridge.pred-y.test)^2)
# lm(y~x, subset=train)
# predict(ridge.mod,s=0,exact=T,type="coefficients",x=x[train,],y=y[train])[1:20,]
# set.seed(1)
# cv.out=cv.glmnet(x[train,],y[train],alpha=0)
# plot(cv.out)
# bestlam=cv.out$lambda.min
# bestlam
# ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
# mean((ridge.pred-y.test)^2)
# out=glmnet(x,y,alpha=0)
# predict(out,type="coefficients",s=bestlam)[1:20,]
#
# # The Lasso
#
# lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
# plot(lasso.mod)
# set.seed(1)
# cv.out=cv.glmnet(x[train,],y[train],alpha=1)
# plot(cv.out)
# bestlam=cv.out$lambda.min
# lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])
# mean((lasso.pred-y.test)^2)
# out=glmnet(x,y,alpha=1,lambda=grid)
# lasso.coef=predict(out,type="coefficients",s=bestlam)[1:20,]
# lasso.coef
# lasso.coef[lasso.coef!=0]
#
#
# # Chapter 6 Lab 3: PCR and PLS Regression
#
# # Principal Components Regression
#
# library(pls)
# set.seed(2)
# pcr.fit=pcr(Salary~., data=Hitters,scale=TRUE,validation="CV")
# summary(pcr.fit)
# validationplot(pcr.fit,val.type="MSEP")
# set.seed(1)
# pcr.fit=pcr(Salary~., data=Hitters,subset=train,scale=TRUE, validation="CV")
```

```r
# validationplot(pcr.fit,val.type="MSEP")
# pcr.pred=predict(pcr.fit,x[test,],ncomp=7)
# mean((pcr.pred-y.test)^2)
# pcr.fit=pcr(y~x,scale=TRUE,ncomp=7)
# summary(pcr.fit)
#
# # Partial Least Squares
#
# set.seed(1)
# pls.fit=plsr(Salary~., data=Hitters,subset=train,scale=TRUE, validation="CV")
# summary(pls.fit)
# validationplot(pls.fit,val.type="MSEP")
# pls.pred=predict(pls.fit,x[test,],ncomp=2)
# mean((pls.pred-y.test)^2)
# pls.fit=plsr(Salary~., data=Hitters,scale=TRUE,ncomp=2)
# summary(pls.fit)
```