STOR 767 Spring 2019 Hw2: Computational Part

Due on 02/06/2019 in Class

Zhenghan Fang

Instruction.

- Please use **RMarkdown** to create a formatted report for the **Computational Part** of this homework.

Exercise 1

Read data.

```
data.prost <- read.table('prostate.data.txt')
pred.name <- c("lcavol","lweight","age","lbph","svi","lcp","gleason","pgg45")
resp.name <- "lpsa"
train.idx <- which(data.prost$train)
test.idx <- which(!data.prost$train)
pred.N <- length(pred.name)
formula.full <- as.formula(paste(resp.name, paste(pred.name, collapse=" + "), sep=" ~ "))
```

Standardize the predictors to unit variance and zero mean.

```
for (i in pred.name){
  t <- data.prost[, i]
  data.prost[ ,i] <- (t-mean(t)) / sqrt(var(t))
}
```

Split the data into training and test set.

```
data.train <- data.prost[train.idx, ]
data.test <- data.prost[test.idx, ]
```

Least squares (LS)

The coefficients are

```
model.ls <- lm(formula.full, data.train)
coef(model.ls)
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##  2.46493292  0.67952814  0.26305307 -0.14146483  0.21014656  0.30520060
##         lcp     gleason       pgg45
## -0.28849277 -0.02130504  0.26695576
```

The test error is

```
test.pred <- predict(model.ls, data.test)
err <- abs(data.test[,resp.name] - test.pred) ^ 2
mean(err)
```

```
## [1] 0.521274
```

Best-subset selection

```
regfit.best = regsubsets(formula.full, data = data.train, nvmax=8)
```

The coefficients of the best model that contains 2 variables are

```
coef(regfit.best, id = 2)
```

```
## (Intercept)      lcavol     lweight
##   2.4773573   0.7397137   0.3163282
```

The test error of the above model is

```
test.mat = model.matrix(formula.full, data = data.test)
coefi= coef(regfit.best, id = 2)
pred=test.mat[,names(coefi)]%*%coefi
err=(data.test$lpsa-pred)^2
mean(err)
```

```
## [1] 0.4924823
```

Ridge regression

Fit with cross validation.

```
x <- data.matrix(data.train[,pred.name])
y <- data.train[,resp.name]
fit <- cv.glmnet(x,y,alpha=0,nlambda=100)
```

Select the λ which gives minimum mean cross-validated error.

```
fit$lambda.min
```

```
## [1] 0.09645702
```

The test error is

```
pred <- predict(fit,newx=data.matrix(data.test[,pred.name]),  s = "lambda.min")
err = mean((data.test$lpsa-pred)^2)
err
```

```
## [1] 0.4932193
```

The coefficients are

```
coef(fit, s = "lambda.1se")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  2.459332992
## lcavol       0.313013858
## lweight      0.192110518
## age         -0.007296873
## lbph         0.136046822
## svi          0.194866328
## lcp          0.063274211
## gleason      0.052644865
## pgg45        0.107661820
```

LASSO

Fit with cross validation.

```
fit <- cv.glmnet(x,y,alpha=1)
```

Select the λ which gives the most regularized model such that error is within one standard error of the minimum.

```
fit$lambda.1se
```

```
## [1] 0.1646861
```

The test error is

```
pred <- predict(fit,newx=data.matrix(data.test[,pred.name]),  s = "lambda.1se")
err = mean((data.test$lpsa-pred)^2)
err
```

```
## [1] 0.4601393
```

The coefficients are

```
coef(fit, s = "lambda.1se")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 2.46719612
## lcavol      0.53807071
## lweight     0.18474896
## age         .
## lbph        0.04402323
## svi         0.12484079
## lcp         .
## gleason     .
## pgg45       0.02530337
```

PCR

Fit with cross validation.

```
fit <- pcr(formula.full, data = data.train, validation = "CV")
```

Select 7 as the number of components.

The test error is

```
pred <- predict(fit, ncomp = 7, newdata = data.test)
err = mean((data.test$lpsa-pred)^2)
err
```

```
## [1] 0.44936
```

The coefficients are

```
coef(fit, ncomp = 7)
```

```
## , , 7 comps
##
##                 lpsa
## lcavol    0.55087266
```

```
## lweight  0.28876032
## age     -0.15471478
## lbph     0.21411395
## svi      0.31461483
## lcp     -0.06229606
## gleason  0.22754818
## pgg45   -0.04782207
```

PLS

Fit with cross validation.

```r
fit <- plsr(formula.full, data = data.train, validation = "CV")
summary(fit)
```

```
## Data:    X dimension: 67 8
##  Y dimension: 67 1
## Fit method: kernelpls
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.217   0.8401   0.8188   0.7967   0.7868   0.7725   0.7745
## adjCV        1.217   0.8373   0.8133   0.7913   0.7796   0.7667   0.7685
##        7 comps  8 comps
## CV      0.7756   0.7756
## adjCV   0.7695   0.7695
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       41.27    58.49    71.38    79.88    85.79    90.06    94.93
## lpsa    57.02    64.67    67.56    69.09    69.38    69.43    69.44
##       8 comps
## X      100.00
## lpsa    69.44
```

Select 5 as the number of components based on the cross validation result.

The test error is

```r
pred <- predict(fit, ncomp = 5, newdata = data.test)
err=mean((data.test$lpsa-pred)^2)
err
```

```
## [1] 0.505514
```

The coefficients are

```r
coef(fit, ncomp = 5)
```

```
## , , 5 comps
##
##                lpsa
## lcavol   0.68332019
## lweight  0.26726531
## age     -0.13895979
```

```
## lbph     0.19967507
## svi      0.31112648
## lcp     -0.28657308
## gleason  0.01519539
## pgg45    0.22347932
```

Exercise 2

Read data from *zip.train.gz*. Parse 3's and 8's from the data.

```
data <- read.table('zip.train')
data <- data[which(data$V1 == 3 | data$V1 == 8), ]
```

Split the data into training (60%) and test set.

```
N.tr <- round(length(data[,1])*0.6)
N.te <- length(data[,1]) - N.tr
data.tr <- data[1:N.tr,]
data.te <- data[(N.tr+1):length(data[,1]),]
```

Define function *knn* to perform K-Nearest Neighbors Classifier using L-p distances. Use matrix computation to accelerate calculation of distance.

```
knn <- function(x, tr.pred, tr.resp, k, p){
  # x: test data point (predictors).
  # tr.pred: training data points (predictors).
  # tr.resp: training data points (reponses).
  # k: # of neighbors.
  # p: L-p distance.

  N.tr <- length(tr.resp)

  x = data.matrix(x)
  x = x[rep(1,N.tr), ]
  y = data.matrix(tr.pred)
  dist = rowSums( abs(x - y) ^ p )

  # slow implementation
  #for(i in c(1:N.tr)){
  #  print(i)
  #  dist[i] <- get.dist(x, tr.pred[,])
  #}

  idx <- order(dist)[1:k]
  n.3 <- sum(tr.resp[idx] == 3)
  n.8 <- sum(tr.resp[idx] == 8)
  x.resp <- 3*(n.3>n.8) + 8*(n.3<=n.8)
  return(x.resp)
}
```

Define function *apply.knn* to apply *knn* to the data.

```
apply.knn <- function(data.te, data.tr, k, p){
  # data.te: test data.
  # data.tr: training data.
```

```r
  # k: # of neighbors.
  # p: L-p distance.
  te.pred = rep(NA, N.te)
  for(i in c(1:N.te)){
    te.pred[i] = knn(data.te[i,2:257], data.tr[,2:257], data.tr[,1], k, p)
  }
  te.error = mean((data.te[,1]!=te.pred))

  tr.pred = rep(NA, N.tr)
  for(i in c(1:N.tr)){
    tr.pred[i] = knn(data.tr[i,2:257], data.tr[,2:257], data.tr[,1], k, p)
  }
  tr.error = mean((data.tr[,1]!=tr.pred))
  c(te.error,tr.error)
}
```

Calcualte training and test errors for $K = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and L-p distances with p $= \{1, 1.5, 2\}$.

```r
k.knn = c(1,2,3,4,5,6,7,8,9,10)
p.dist = c(1, 1.5, 2)
te.err = matrix(nrow = length(k.knn), ncol = length(p.dist))
tr.err = matrix(nrow = length(k.knn), ncol = length(p.dist))
for(i in 1:length(k.knn)){
  for(j in 1:length(p.dist)){
    err = apply.knn(data.te, data.tr, k=k.knn[i], p=p.dist[j])
    te.err[i,j] = err[1]
    tr.err[i,j] = err[2]
  }
}
```

Calculate the least squares estimate.

```r
fit.ls = lm(V1~., data.tr)
```

Calculate the test and training errors of least squares.

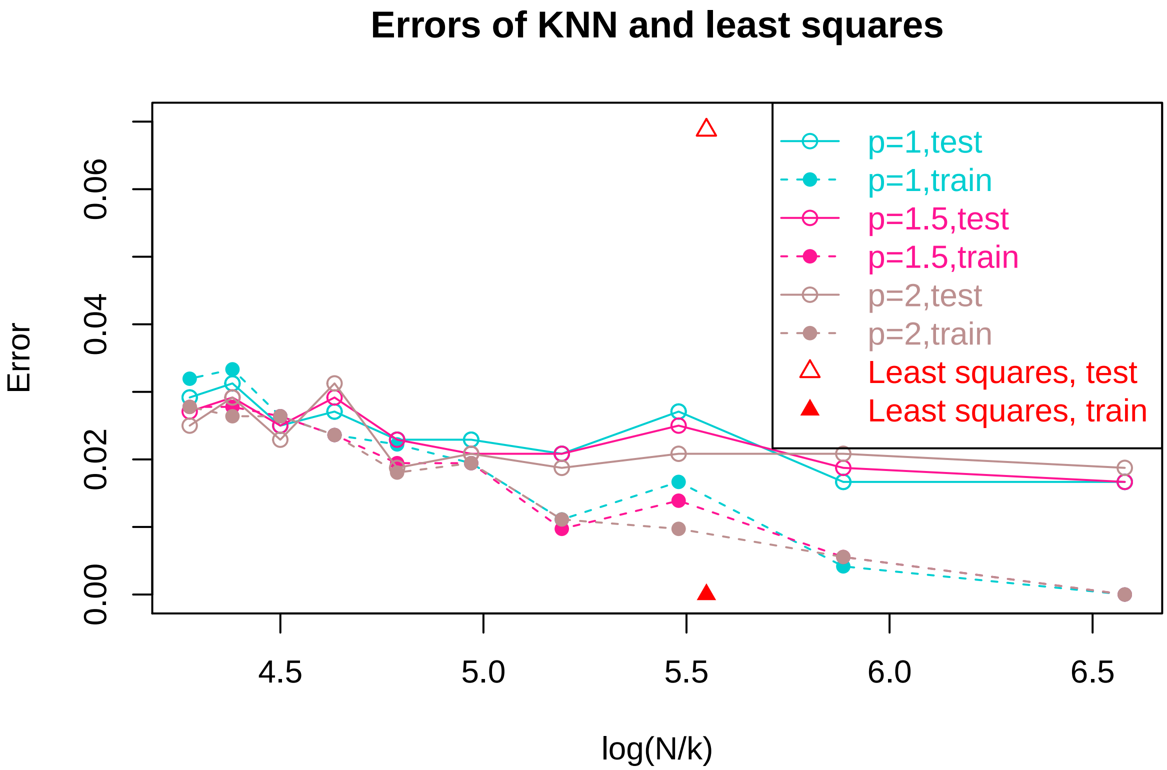
```r
pred = predict(fit.ls, data.te)
pred = (abs(pred-3) < abs(pred-8) ) * 3 + (abs(pred-3) >= abs(pred-8) ) * 8
te.err.ls = mean((data.te$V1!=pred))

pred = predict(fit.ls, data.tr)
pred = (abs(pred-3) < abs(pred-8) ) * 3 + (abs(pred-3) >= abs(pred-8) ) * 8
tr.err.ls = mean((data.tr$V1!=pred))
```

Plot the training and test errors of least squares and KNN with various K's and p's. Error = number of misclassified samples / total number of samples.

```r
cols = c("DarkTurquoise","DeepPink","RosyBrown","Red")
x = log(N.tr/k.knn)
plot(x, te.err[,1],col=cols[1], pch=1,xlab="log(N/k)",ylab="Error", ylim=c(0,0.07))
for(i in c(1:3)){
  lines(x, te.err[,i], col=cols[i], lty=1)
  points(x, te.err[,i], col=cols[i], pch=1)
  lines(x, tr.err[,i], col=cols[i], lty=2)
  points(x, tr.err[,i], col=cols[i], pch=16)
}
points(log(257), te.err.ls, col=cols[4], pch=2)
```

```r
points(log(257), tr.err.ls, col=cols[4], pch=17)
legend("topright", c("p=1,test","p=1,train","p=1.5,test","p=1.5,train","p=2,test","p=2,train","Least sq
title("Errors of KNN and least squares")
```



Exercise 3

(a) Variable Selection

Load the dataset. Remove rows with missing data. Take log transform of Salary.

```r
data(Hitters)
Hitters = na.omit(Hitters)
Hitters$Salary = log(Hitters$Salary)
```

Prepare data.

```r
# get the response as a vector
resp <- Hitters[,"Salary"]

# get the predictors
N.pred <- 19
pred <- Hitters[,c(1:18,20)]
```

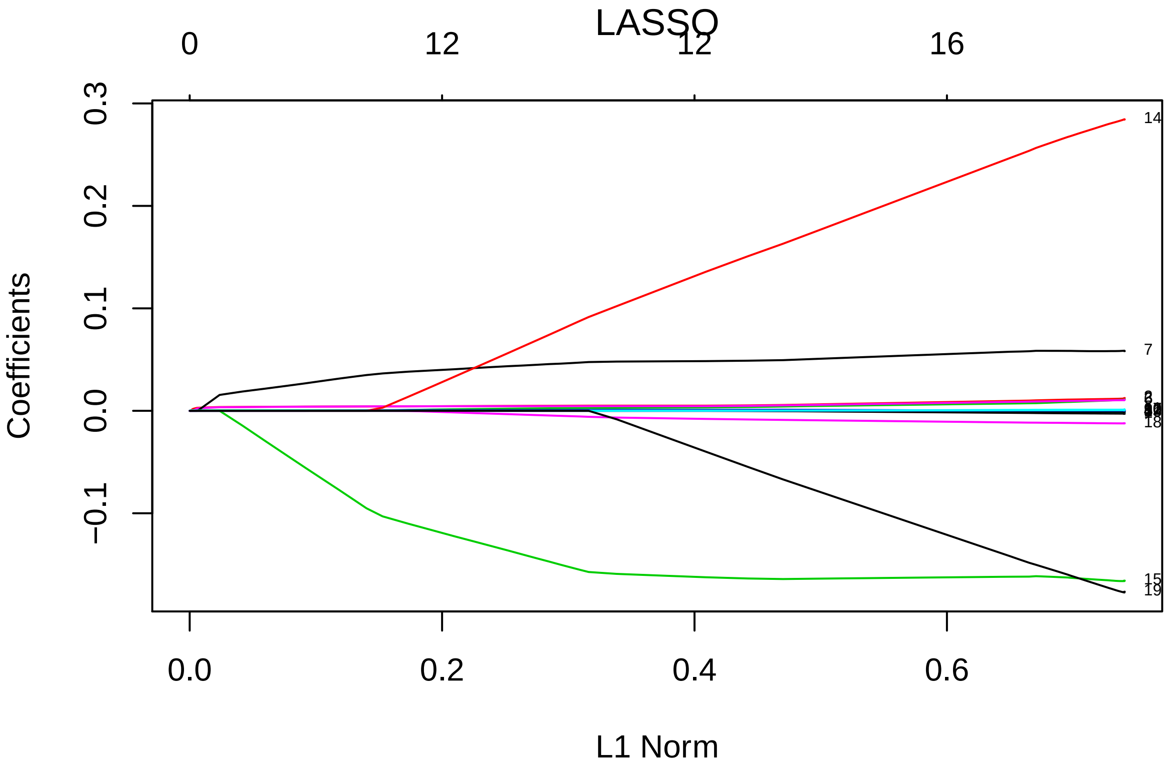
Fit and Visualize regularization paths for LASSO, elastic net, adaptive LASSO, SCAD.

```r
# conver the predictors to a matrix
pred <- data.matrix(pred)

x <- pred
```

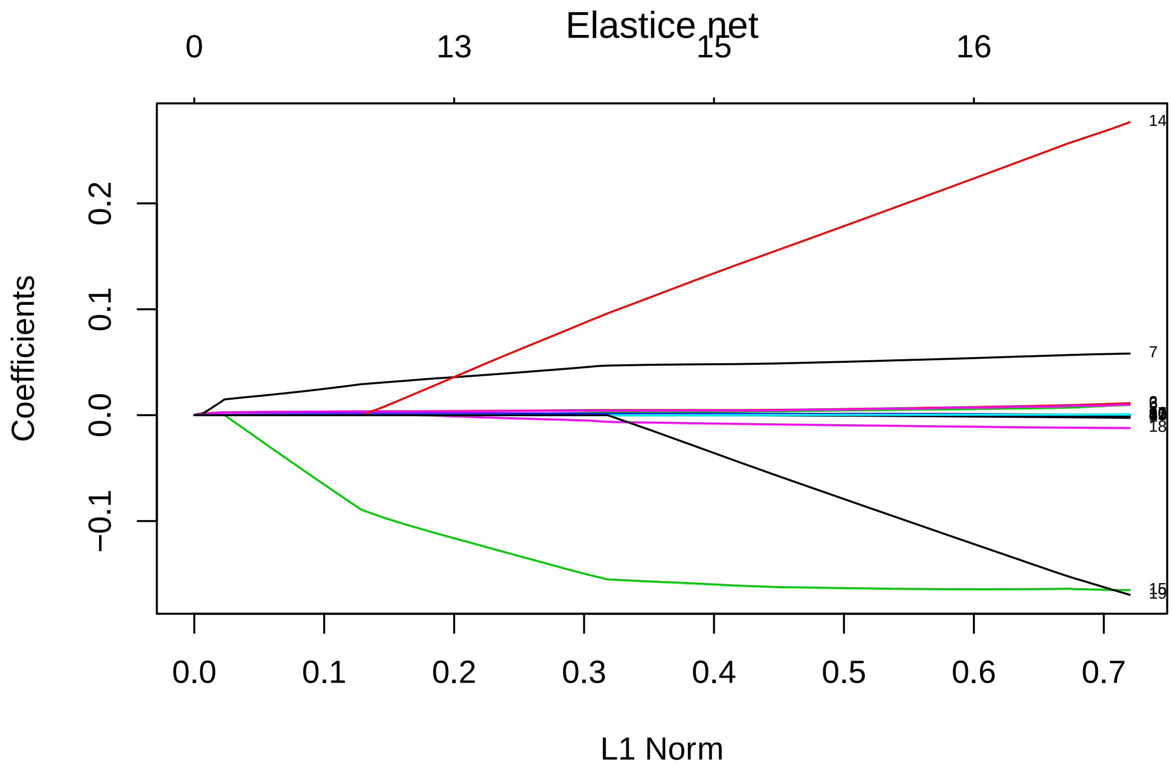
```r
y <- resp

# lasso
fit.lasso <- glmnet(x,y,family='gaussian',alpha=1)
plot(fit.lasso, main=expression(paste("LASSO")), label=TRUE)
```



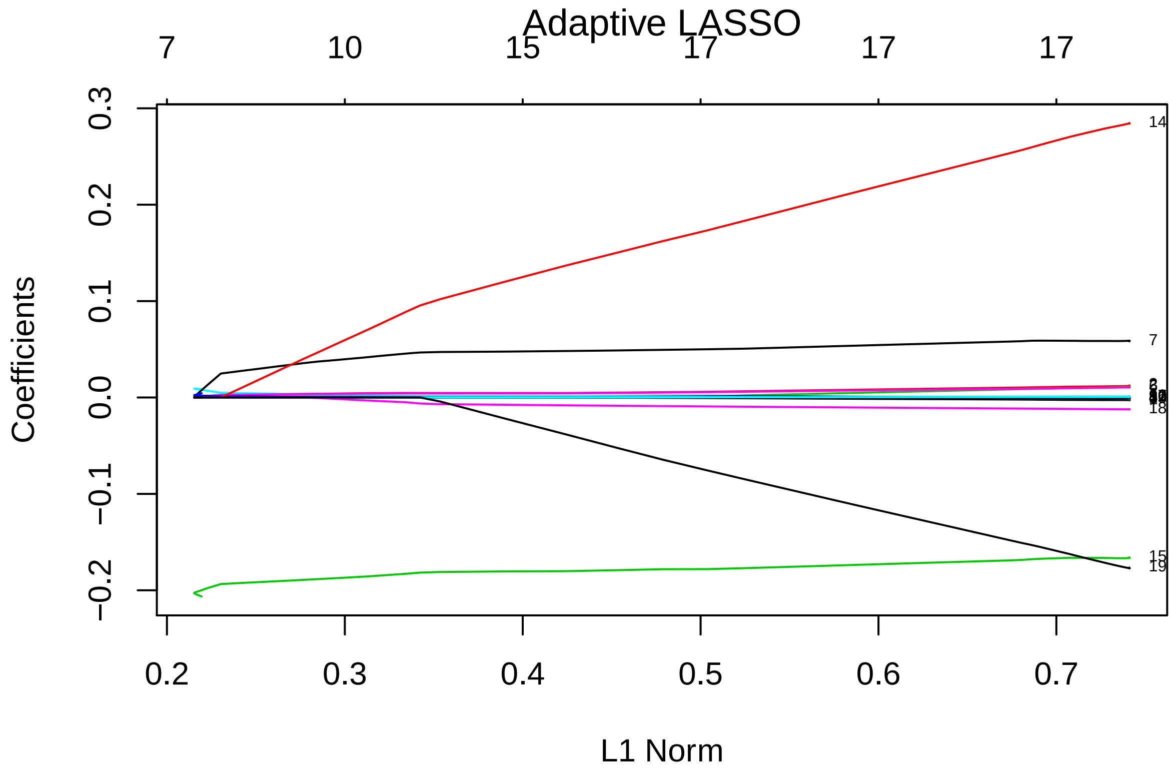

```r
#predict(fit.lasso,newx=x,s=c(0.01,0.005))

# elastic net
alpha = 0.5
fit.elastic <- glmnet(x,y,family='gaussian',alpha=alpha)
plot(fit.elastic, main=expression(paste("Elastice net")), label=TRUE)
```



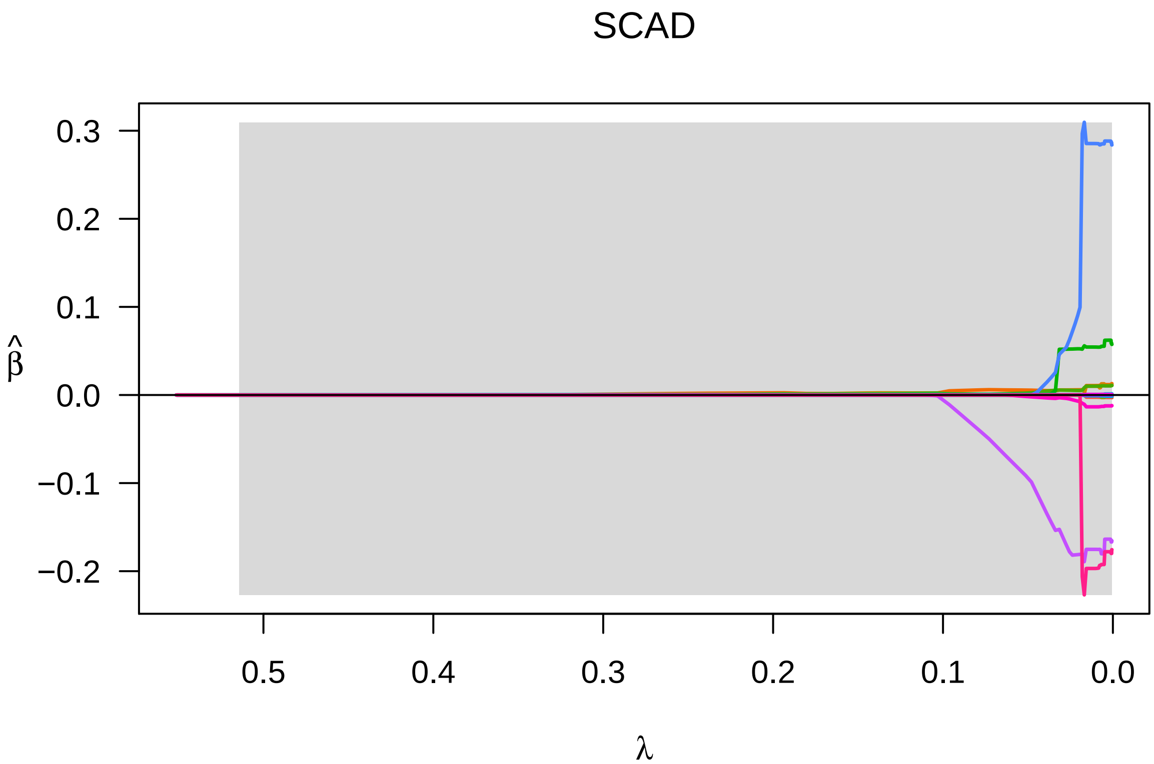

```
#predict(fit.elastic,newx=x,s=c(0.01,0.005))

# adaptive LASSO
p.fac <- rep(1, N.pred)
p.fac[c(5, 10, 15)] <- 0
fit.adpLasso <- glmnet(x,y,family='gaussian',alpha=1,penalty.factor=p.fac)
plot(fit.adpLasso, main=expression(paste("Adaptive LASSO")), label=TRUE)
```




```
#predict(fit.adpLasso,newx=x,s=c(0.01,0.005))

# SCAD
library('ncvreg')
fit.SCAD <- ncvreg(x, y, penalty="SCAD")
plot(fit.SCAD, main=expression(paste("SCAD")))
```



The top predictor selected by LASSO is "CRuns".

```
coefi = coef(fit.lasso)
coefi[,2]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
## 5.8737122910 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##          RBI        Walks        Years       CAtBat        CHits
## 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##       CHmRun        CRuns         CRBI       CWalks       League
## 0.0000000000 0.0001481346 0.0000000000 0.0000000000 0.0000000000
##     Division      PutOuts      Assists       Errors    NewLeague
## 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```

The top predictors selected by elastic net are "CAtBat", "CHits", and "CRuns".

```
coefi = coef(fit.elastic)
coefi[,2]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
## 5.884426e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##          RBI        Walks        Years       CAtBat        CHits
## 0.000000e+00 0.000000e+00 0.000000e+00 1.203762e-06 2.631984e-05
##       CHmRun        CRuns         CRBI       CWalks       League
## 0.000000e+00 5.699681e-05 0.000000e+00 0.000000e+00 0.000000e+00
##     Division      PutOuts      Assists       Errors    NewLeague
## 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

The top predictors selected by adaptive LASSO are "RBI", "CHits", "CHmRun", and "Division".

```
coefi = coef(fit.adpLasso)
coefi[,2]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  5.458325e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
##           RBI         Walks         Years        CAtBat         CHits
##  8.806189e-03  0.000000e+00  0.000000e+00  0.000000e+00  7.107665e-05
##        CHmRun         CRuns          CRBI        CWalks        League
##  3.966969e-03  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
##      Division       PutOuts       Assists        Errors     NewLeague
## -2.057007e-01  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
```

The top predictor selected by SCAD is CRuns".

```r
coefi = coef(fit.SCAD)
coefi[,2]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
## 5.8866265810 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##          RBI        Walks        Years       CAtBat        CHits
## 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
##       CHmRun        CRuns         CRBI       CWalks       League
## 0.0000000000 0.0001123828 0.0000000000 0.0000000000 0.0000000000
##     Division      PutOuts      Assists       Errors    NewLeague
## 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```

The top predictors selected by different methods are different, because different methods have different regularizations on fitting parameters.

(b) Prediction

Split data into training (50%) and test set.

```r
set.seed(1)
train = sample(c(TRUE, FALSE), nrow(Hitters), rep = TRUE)
test = (!train)
```

Least squares (LS)

```r
model.ls <- lm(Salary ~., Hitters[train, ])
```

The test MSE is

```r
test.pred <- predict(model.ls, Hitters[test, ])
err <- abs(Hitters$Salary[test] - test.pred) ^ 2
err.ls.te <- mean(err)
err.ls.te
```

```
## [1] 0.535157
```

The training MSE is

```r
train.pred <- predict(model.ls, Hitters[train, ])
err <- abs(Hitters$Salary[train] - train.pred) ^ 2
err.ls.tr <- mean(err)
err.ls.tr
```

```
## [1] 0.2715661
```

The coefficients are

```r
coef(model.ls)
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  4.3026034408 -0.0025265381  0.0127380742  0.0022373233  0.0011440948
##           RBI         Walks         Years        CAtBat         CHits
##  0.0027375245  0.0092989898  0.0384669300  0.0001087536  0.0021093310
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  0.0042361546 -0.0017379234 -0.0020926499 -0.0015526067  0.5555255631
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -0.1036815660  0.0002874883  0.0001216390 -0.0126210594 -0.3953590657
```

Ridge regression

Fit with cross validation.

```r
x <- data.matrix(Hitters[train,c(1:18,20)])
y <- data.matrix(Hitters[train,19])
fit <- cv.glmnet(x,y,alpha=0,nlambda=100)
```

The test error is

```r
pred <- predict(fit,newx=data.matrix(Hitters[test,c(1:18,20)]),  s = "lambda.min")
err.rid.te = mean((Hitters$Salary[test]-pred)^2)
err.rid.te
```

```
## [1] 0.4729224
```

The training error is

```r
pred <- predict(fit,newx=data.matrix(Hitters[train,c(1:18,20)]),  s = "lambda.min")
err.rid.tr = mean((Hitters$Salary[train]-pred)^2)
err.rid.tr
```

```
## [1] 0.2925044
```

The coefficients are

```r
coef(fit, lambda = fit$lambda.min)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  5.057463e+00
## AtBat        2.752852e-04
## Hits         1.159731e-03
## HmRun        2.739750e-03
## Runs         1.666092e-03
## RBI          1.538619e-03
## Walks        1.239343e-03
## Years        8.459138e-03
## CAtBat       2.052308e-05
## CHits        8.094478e-05
## CHmRun       2.542679e-04
## CRuns        1.402280e-04
## CRBI         1.153741e-04
## CWalks       1.070957e-04
## League       3.513257e-02
## Division    -4.037416e-02
```

```
## PutOuts      7.504399e-05
## Assists      4.322130e-05
## Errors       3.018090e-04
## NewLeague    1.400326e-02
```

LASSO

Fit with cross validation.

```
fit <- cv.glmnet(x,y,alpha=1,nlambda=100)
```

The test error is

```
pred <- predict(fit,newx=data.matrix(Hitters[test,c(1:18,20)]),  s = "lambda.min")
err.las.te = mean((Hitters$Salary[test]-pred)^2)
err.las.te
```

```
## [1] 0.4623307
```

The training error is

```
pred <- predict(fit,newx=data.matrix(Hitters[train,c(1:18,20)]),  s = "lambda.min")
err.las.tr = mean((Hitters$Salary[train]-pred)^2)
err.las.tr
```

```
## [1] 0.3173605
```

The coefficients are

```
coef(fit, lambda = fit$lambda.min)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 4.9190813545
## AtBat       .
## Hits        0.0054921536
## HmRun       .
## Runs        .
## RBI         0.0001834590
## Walks       .
## Years       .
## CAtBat      .
## CHits       0.0005259238
## CHmRun      .
## CRuns       .
## CRBI        .
## CWalks      .
## League      .
## Division    .
## PutOuts     .
## Assists     .
## Errors      .
## NewLeague   .
```

Elastic net

```r
fit <- cv.glmnet(x,y,alpha=0.5,nlambda=100)
```

The test error is

```r
pred <- predict(fit,newx=data.matrix(Hitters[test,c(1:18,20)]),  s = "lambda.min")
err.ela.te = mean((Hitters$Salary[test]-pred)^2)
err.ela.te
```

```
## [1] 0.4987902
```

The training error is

```r
pred <- predict(fit,newx=data.matrix(Hitters[train,c(1:18,20)]),  s = "lambda.min")
err.ela.tr = mean((Hitters$Salary[train]-pred)^2)
err.ela.tr
```

```
## [1] 0.2774543
```

The coefficients are

```r
coef(fit, lambda = fit$lambda.min)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 4.933563e+00
## AtBat       .
## Hits        3.912543e-03
## HmRun       .
## Runs        1.477522e-03
## RBI         1.713729e-03
## Walks       .
## Years       6.800443e-03
## CAtBat      3.449591e-05
## CHits       2.429012e-04
## CHmRun      .
## CRuns       1.588759e-04
## CRBI        .
## CWalks      .
## League      .
## Division    .
## PutOuts     .
## Assists     .
## Errors      .
## NewLeague   .
```

Adaptive LASSO

```r
p.fac <- rep(1, N.pred)
p.fac[c(5, 10, 15)] <- 0
fit <- cv.glmnet(x,y,alpha=1,penalty.factor=p.fac)
```

The test error is

```
pred <- predict(fit,newx=data.matrix(Hitters[test,c(1:18,20)]),  s = "lambda.min")
err.adp.te = mean((Hitters$Salary[test]-pred)^2)
err.adp.te
```

```
## [1] 0.4994871
```

The training error is

```
pred <- predict(fit,newx=data.matrix(Hitters[train,c(1:18,20)]),  s = "lambda.min")
err.adp.tr = mean((Hitters$Salary[train]-pred)^2)
err.adp.tr
```

```
## [1] 0.2792487
```

The coefficients are

```
coef(fit, lambda = fit$lambda.min)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  4.7632647319
## AtBat        .
## Hits         0.0023083285
## HmRun        .
## Runs         .
## RBI          0.0114903681
## Walks        .
## Years        0.0120710625
## CAtBat       .
## CHits        0.0005756477
## CHmRun      -0.0006541021
## CRuns        .
## CRBI         .
## CWalks       .
## League       0.0512864769
## Division    -0.1414889323
## PutOuts      .
## Assists      .
## Errors       .
## NewLeague    .
```

SCAD

Fit with cross validation.

```
fit.SCAD <- cv.ncvreg(x, y, penalty="SCAD")
```

The test error is

```
pred <- predict(fit.SCAD,data.matrix(Hitters[test,c(1:18,20)]), lambda = fit.SCAD$lambda.min)
err.scad.te = mean((Hitters$Salary[test]-pred)^2)
err.scad.te
```

```
## [1] 0.5046771
```

The training error is

```r
pred <- predict(fit.SCAD,data.matrix(Hitters[train,c(1:18,20)]), lambda = fit.SCAD$lambda.min)
err.scad.tr = mean((Hitters$Salary[train]-pred)^2)
err.scad.tr
```

```
## [1] 0.343659
```

The coefficients are

```r
coef(fit.SCAD, lambda = fit.SCAD$lambda.min)
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
## 4.3076162752 0.0000000000 0.0091136543 0.0000000000 0.0000000000
##          RBI        Walks        Years       CAtBat        CHits
## 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0007733268
##       CHmRun        CRuns         CRBI       CWalks       League
## 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0456446544
##     Division      PutOuts      Assists       Errors    NewLeague
## 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```

Best subset selection

Choose among models using cross validation. (Reference: https://rpubs.com/davoodastaraky/subset)

```r
k = 10
set.seed(1)
Hitters.train = Hitters[train, ]
folds = sample(1:k,nrow(Hitters.train),replace=TRUE)
# table(folds)

cv.errors=matrix(NA,k,19, dimnames=list(NULL, paste(1:19)))

for(j in 1:k){
        best.fit = regsubsets(Salary ~., data=Hitters.train[folds != j,], nvmax = 19)
        test.mat = model.matrix(Salary~., data = Hitters.train[folds == j,])

        for (i in 1:19){
          coefi= coef(best.fit, id = i)
          pred=test.mat[,names(coefi)]%*%coefi
          cv.errors[j, i] = mean((Hitters.train$Salary[folds == j] - pred)^2)
        }
}

mean.cv.errors = apply(cv.errors ,2,mean)
```

The mean cross-validation errors for different numbers of variables are

```r
mean.cv.errors
```

```
##         1         2         3         4         5         6         7
## 0.5211516 0.3971851 0.4367048 0.4615058 0.4667831 0.4664319 0.4531661
##         8         9        10        11        12        13        14
## 0.4606001 0.4254916 0.4303366 0.4344454 0.4356314 0.4539237 0.4554794
##        15        16        17        18        19
## 0.4534422 0.4527108 0.4456892 0.4453084 0.4458934
```

The 2-variable model gives the minimum mean cross-validation error.

```r
which.min(mean.cv.errors)
```

```
## 2
## 2
```

Apply best subset selection on full training set and select the 2-variable model. The test error is

```r
reg.best=regsubsets (Salary~.,data=Hitters[train, ] , nvmax=19)
coefi = coef(reg.best ,2)
test.mat = model.matrix(Salary~., data = Hitters[test,])
pred=test.mat[,names(coefi)]%*%coefi
err.bsub.te = mean((Hitters$Salary[test] - pred)^2)
err.bsub.te
```

```
## [1] 0.5042485
```

The training error is

```r
train.mat = model.matrix(Salary~., data = Hitters[train,])
pred=train.mat[,names(coefi)]%*%coefi
err.bsub.tr = mean((Hitters$Salary[train] - pred)^2)
err.bsub.tr
```

```
## [1] 0.3482641
```

The coefficients are

```r
coef(reg.best ,2)
```

```
##  (Intercept)         Hits        CHits
## 4.3759891872 0.0090545713 0.0007749659
```

Discussion

Among the methods, ridge regression and LASSO give the best prediction errors on test set. These methods perform well probably because their regularizations are more suitable for the data.

The least squares method seems to overfit the training set. The reason is that it has no regularization on the fitting parameters.

Not all methods select the same subset of variables. However, “Hits”, “CHits”, and “RBI” are selected by most methods, which suggests that these variables are the most predictive of the response “Salary”.