Feature selection

STAT5003

Justin Wishart

Libraries to load

```
library(mlbench)
```

Feature selection and classification on Sonar dataset

This is the data set used by Gorman and Sejnowski in their study of the classification of sonar signals using a neural network [1]. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp. The label associated with each record contains the letter “R” if the object is a rock and “M” if it is a mine (metal cylinder). The numbers in the labels are in increasing order of aspect angle, but they do not encode the angle directly.

References Gorman, R. P., and Sejnowski, T. J. (1988). “Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets” in Neural Networks, Vol. 1, pp. 75-89.

Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html (http://www.ics.uci.edu/~mlearn/MLRepository.html)]. Irvine, CA: University of California, Department of Information and Computer Science.

```
# load data
library(mlbench)
library(class)
data(Sonar)
dim(Sonar)
```

```
## [1] 208  61
```

partition the data into training and testing sets

```
library(caret)
```

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

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

```
set.seed(123)
inTrain <- createDataPartition(Sonar$Class, p = 0.5)[[1]]
SonarTrain <- Sonar[inTrain, ]
SonarTest  <- Sonar[-inTrain, ]
```

Wrapper feature selection

Forward stepwise selection

```
selectFeature <- function(train, test, cls.train, cls.test, features) {
  ## identify a feature to be selected
  current.best.accuracy <- -Inf
  selected.i <- NULL
  for (i in seq(ncol(train))) {
    current.f <- colnames(train)[i]
    if (!current.f %in% features) {
      model <- knn(train = cbind(train[, features], train[, current.f]),
                   test = cbind(test[, features], test[, current.f]), cl = cls.train,
k = 3)
      test.acc <- sum(model == cls.test) / length(cls.test)

      if (test.acc > current.best.accuracy) {
        current.best.accuracy <- test.acc
        selected.i <- colnames(train)[i]
      }
    }
  }
  return(selected.i)
}


##
library(caret)
set.seed(1)
inTrain <- createDataPartition(Sonar$Class, p = .6)[[1]]
allFeatures <- colnames(Sonar)[-61]
train <- Sonar[inTrain, -61]
test  <- Sonar[-inTrain, -61]
cls.train <- Sonar$Class[inTrain]
cls.test <- Sonar$Class[-inTrain]

features <- NULL
# select the 1 to 10 best features using knn as a wrapper classifier
for (j in 1:10) {
  selected.i <- selectFeature(train, test, cls.train, cls.test, features)
  print(selected.i)

  # add the best feature from current run
  features <- c(features, selected.i)
}
```

```
## [1] "V48"
## [1] "V1"
## [1] "V34"
## [1] "V45"
## [1] "V36"
## [1] "V43"
## [1] "V51"
## [1] "V50"
## [1] "V52"
## [1] "V53"
```

Classify on the two types of samples using the full dataset compared to using top 10 wrapper selected features

```
# fitting the classifier on top 10 wrapper selected features
knn.fit3 <- knn(train = SonarTrain[, features], test = SonarTest[, features],
                cl = SonarTrain$Class, k = 3, prob = TRUE)
table(knn.fit3, SonarTest$Class)
```

```
##
## knn.fit3  M  R
##        M 41 10
##        R 14 38
```

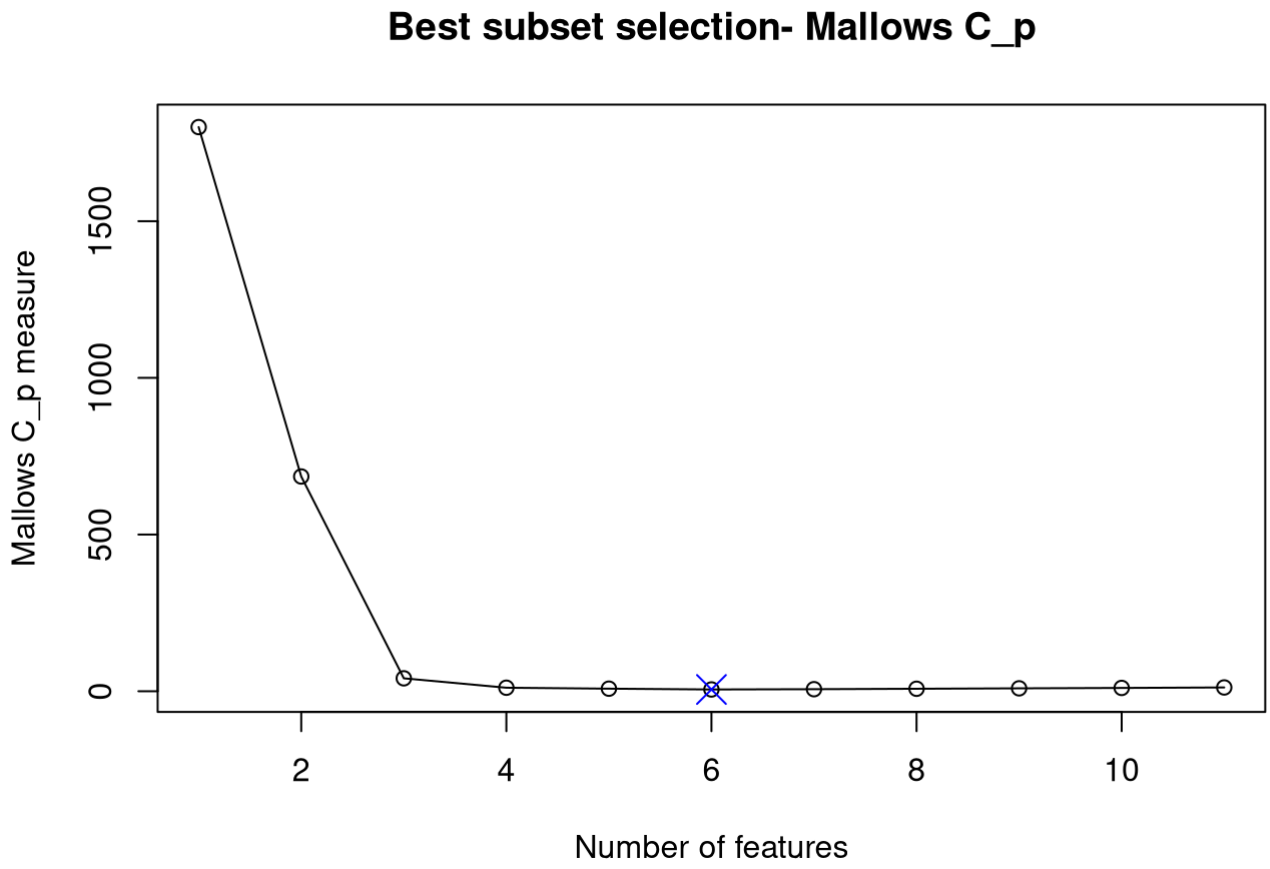
`leaps` the helper package for subset selection and stepwise search

Consider the example plot given in the lecture but only on the training data. Starting with all possible subsets (exhaustive search).

```
library(leaps)
library(ISLR)
leaps.credit <- regsubsets(Balance ~ . - ID, data = Credit,
                           method = "exhaustive", nvmax = 11)
summary.leaps.credit <- summary(leaps.credit)
summary.leaps.credit
```

```
## Subset selection object
## Call: regsubsets.formula(Balance ~ . - ID, data = Credit, method = "exhaustive",
##     nvmax = 11)
## 11 Variables  (and intercept)
##                    Forced in Forced out
## Income                 FALSE      FALSE
## Limit                  FALSE      FALSE
## Rating                 FALSE      FALSE
## Cards                  FALSE      FALSE
## Age                    FALSE      FALSE
## Education              FALSE      FALSE
## GenderFemale           FALSE      FALSE
## StudentYes             FALSE      FALSE
## MarriedYes             FALSE      FALSE
## EthnicityAsian         FALSE      FALSE
## EthnicityCaucasian     FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: exhaustive
##           Income Limit Rating Cards Age Education GenderFemale StudentYes
## 1  ( 1 )  " "    " "   "*"    " "   " " " "       " "          " "
## 2  ( 1 )  "*"    " "   "*"    " "   " " " "       " "          " "
## 3  ( 1 )  "*"    " "   "*"    " "   " " " "       " "          "*"
## 4  ( 1 )  "*"    "*"   " "    "*"   " " " "       " "          "*"
## 5  ( 1 )  "*"    "*"   "*"    "*"   " " " "       " "          "*"
## 6  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       " "          "*"
## 7  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 8  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 9  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 10  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 11  ( 1 ) "*"    "*"   "*"    "*"   "*" "*"       "*"          "*"
##           MarriedYes EthnicityAsian EthnicityCaucasian
## 1  ( 1 )  " "        " "            " "
## 2  ( 1 )  " "        " "            " "
## 3  ( 1 )  " "        " "            " "
## 4  ( 1 )  " "        " "            " "
## 5  ( 1 )  " "        " "            " "
## 6  ( 1 )  " "        " "            " "
## 7  ( 1 )  " "        " "            " "
## 8  ( 1 )  " "        "*"            " "
## 9  ( 1 )  "*"        "*"            " "
## 10  ( 1 ) "*"        "*"            "*"
## 11  ( 1 ) "*"        "*"            "*"
```

```
min.cp <- which.min(summary.leaps.credit$cp)
plot(summary.leaps.credit$cp, type = "l",
     main = "Best subset selection- Mallows C_p",
     ylab = "Mallows C_p measure",
     xlab = "Number of features")
points(summary.leaps.credit$cp)
points(min.cp, summary.leaps.credit$cp[min.cp], pch = 4, col = "blue", cex = 2)
```

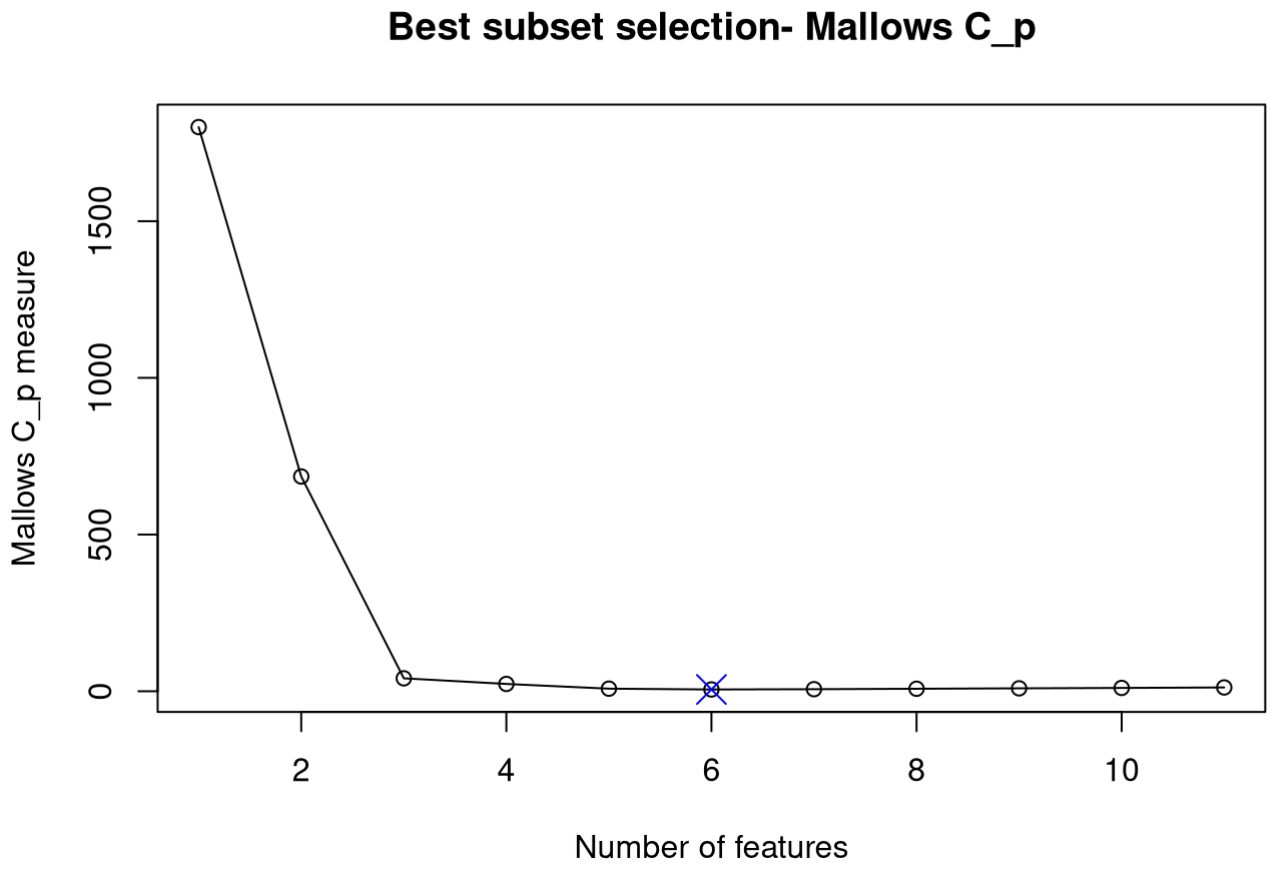


Consider also the forward stepwise search

```
forward.credit <- regsubsets(Balance ~ . - ID, data = Credit,
                             method = "forward", nvmax = 11)
summary.forward.credit <- summary(forward.credit)
summary.forward.credit
```

```
## Subset selection object
## Call: regsubsets.formula(Balance ~ . - ID, data = Credit, method = "forward",
##     nvmax = 11)
## 11 Variables  (and intercept)
##                    Forced in Forced out
## Income                 FALSE      FALSE
## Limit                  FALSE      FALSE
## Rating                 FALSE      FALSE
## Cards                  FALSE      FALSE
## Age                    FALSE      FALSE
## Education              FALSE      FALSE
## GenderFemale           FALSE      FALSE
## StudentYes             FALSE      FALSE
## MarriedYes             FALSE      FALSE
## EthnicityAsian         FALSE      FALSE
## EthnicityCaucasian     FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: forward
##           Income Limit Rating Cards Age Education GenderFemale StudentYes
## 1  ( 1 )  " "    " "   "*"    " "   " " " "       " "          " "
## 2  ( 1 )  "*"    " "   "*"    " "   " " " "       " "          " "
## 3  ( 1 )  "*"    " "   "*"    " "   " " " "       " "          "*"
## 4  ( 1 )  "*"    "*"   "*"    " "   " " " "       " "          "*"
## 5  ( 1 )  "*"    "*"   "*"    "*"   " " " "       " "          "*"
## 6  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       " "          "*"
## 7  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 8  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 9  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 10  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"          "*"
## 11  ( 1 ) "*"    "*"   "*"    "*"   "*" "*"       "*"          "*"
##           MarriedYes EthnicityAsian EthnicityCaucasian
## 1  ( 1 )  " "        " "            " "
## 2  ( 1 )  " "        " "            " "
## 3  ( 1 )  " "        " "            " "
## 4  ( 1 )  " "        " "            " "
## 5  ( 1 )  " "        " "            " "
## 6  ( 1 )  " "        " "            " "
## 7  ( 1 )  " "        " "            " "
## 8  ( 1 )  " "        "*"            " "
## 9  ( 1 )  "*"        "*"            " "
## 10  ( 1 ) "*"        "*"            "*"
## 11  ( 1 ) "*"        "*"            "*"
```

```
min.cp <- which.min(summary.forward.credit$cp)
plot(summary.forward.credit$cp, type = "l",
     main = "Best subset selection- Mallows C_p",
     ylab = "Mallows C_p measure",
     xlab = "Number of features")
points(summary.forward.credit$cp)
points(min.cp, summary.forward.credit$cp[min.cp], pch = 4, col = "blue", cex = 2)
```



Ridge and Lasso regressions.

Ridge and Lasso regression can be achieved using the `glmnet` package which computes both as part of the generalised method of the elastic net. The elastic net includes both the ridge regression and lasso as special cases of the elastic net model below which minimizes the following equation.

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2N}\sum_{i=1}^{N}(y_i-\beta_0-\boldsymbol{\beta}^T\boldsymbol{x}_i)^2+\lambda\left((1-\alpha)\|\beta\|_2^2/2+\alpha\|\beta\|_1\right).$$

where l is the negative log likelihood. The Gaussian likelihood will give the residual sum of squares in the summation term and the choice of $\alpha=1$ or $\alpha=0$ will reduce the penalty term to the Lasso penalty (ℓ_1 penalty) or the Ridge regression penalty (ℓ_2 penalty).

Load the example data “Credit” from “ISLR” package

```
library(ISLR)
names(Credit)
```

```
##  [1] "ID"        "Income"    "Limit"     "Rating"    "Cards"
##  [6] "Age"       "Education" "Gender"    "Student"   "Married"
## [11] "Ethnicity" "Balance"
```

```
dim(Credit)
```

```
## [1] 400  12
```

```
# create learning matrix X and regression response variable Y
x <- model.matrix(Balance ~ ., data = Credit[-1])[, -1]
y <- Credit$Balance

# partition the data into training and test sets (50% each)
set.seed(1)
train <- sample(1:nrow(x), nrow(x) / 2)
test <- -train
y.test <- y[test]
```

Ridge regression

`glmnet` package implements Ridge and Lasso regressions and anything in between

```
library(glmnet)
```

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

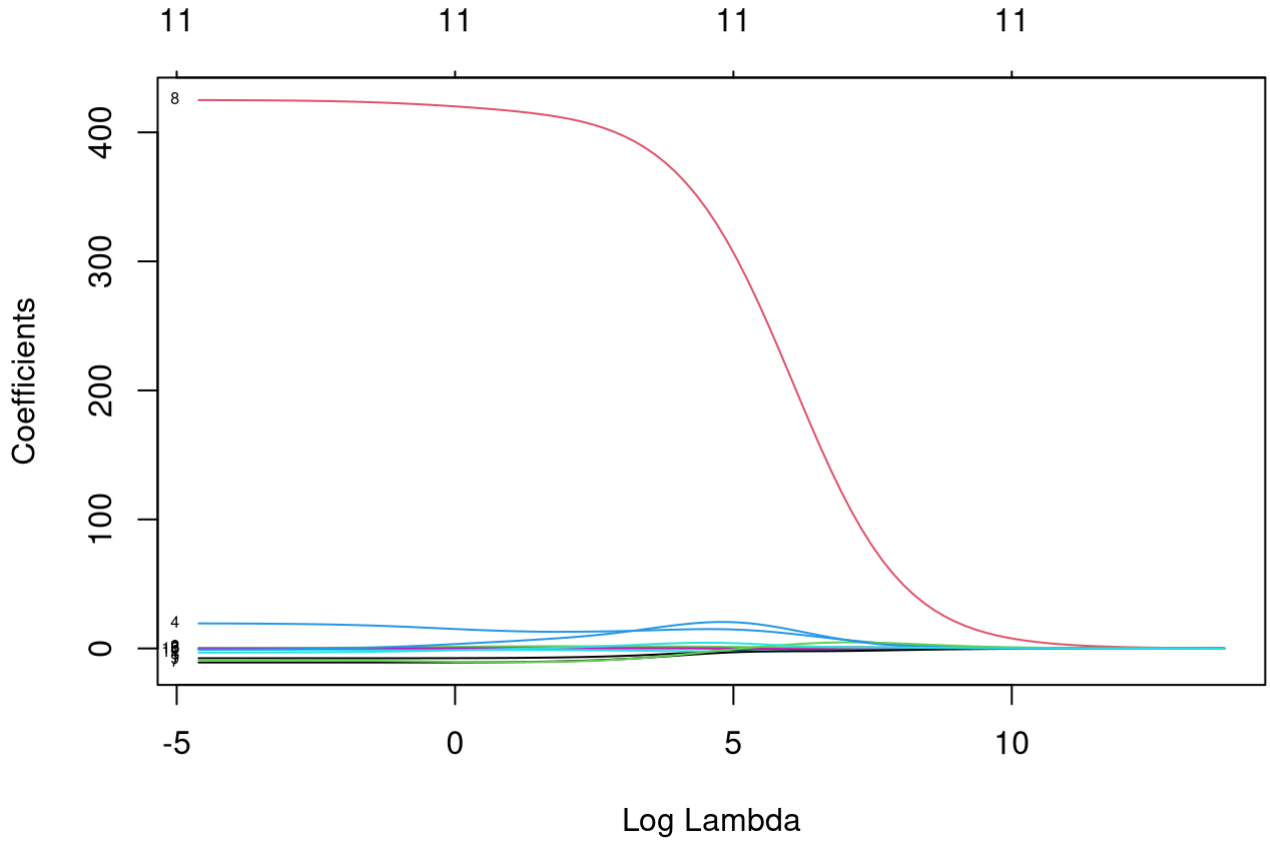
```
## Loaded glmnet 4.1-4
```

```
# set the range of lambda values to be tested.
grid <- 10^seq(6, -2, length = 100)

# alpha is the elastic net mixing parameter with 0 correspond to Ridge regression
# and 1 correspond to Lasso and anything in between correspond to elastic net
ridge.mod <- glmnet(x[train, ], y[train], alpha = 0, lambda = grid, standardize = TRU
E)
dim(coef(ridge.mod))
```

```
## [1]  12 100
```

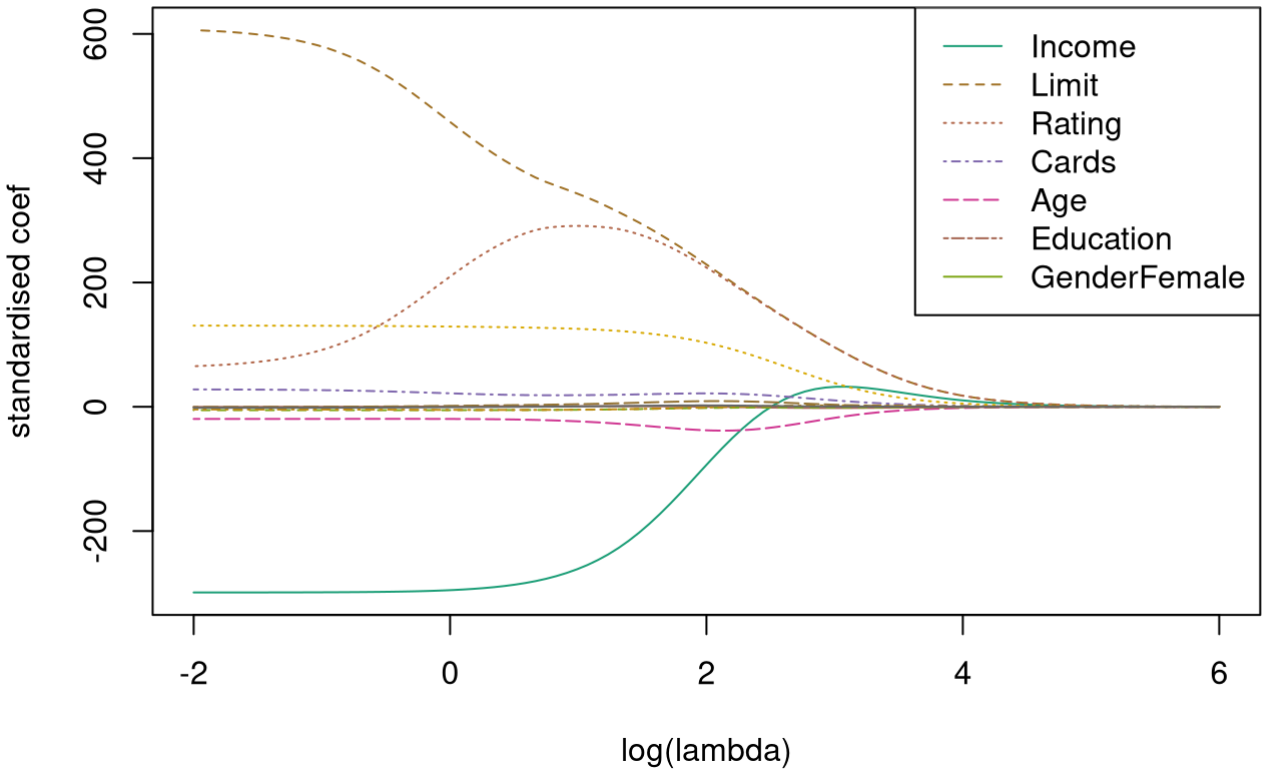
```
plot(ridge.mod, xvar = "lambda", label = TRUE)
```



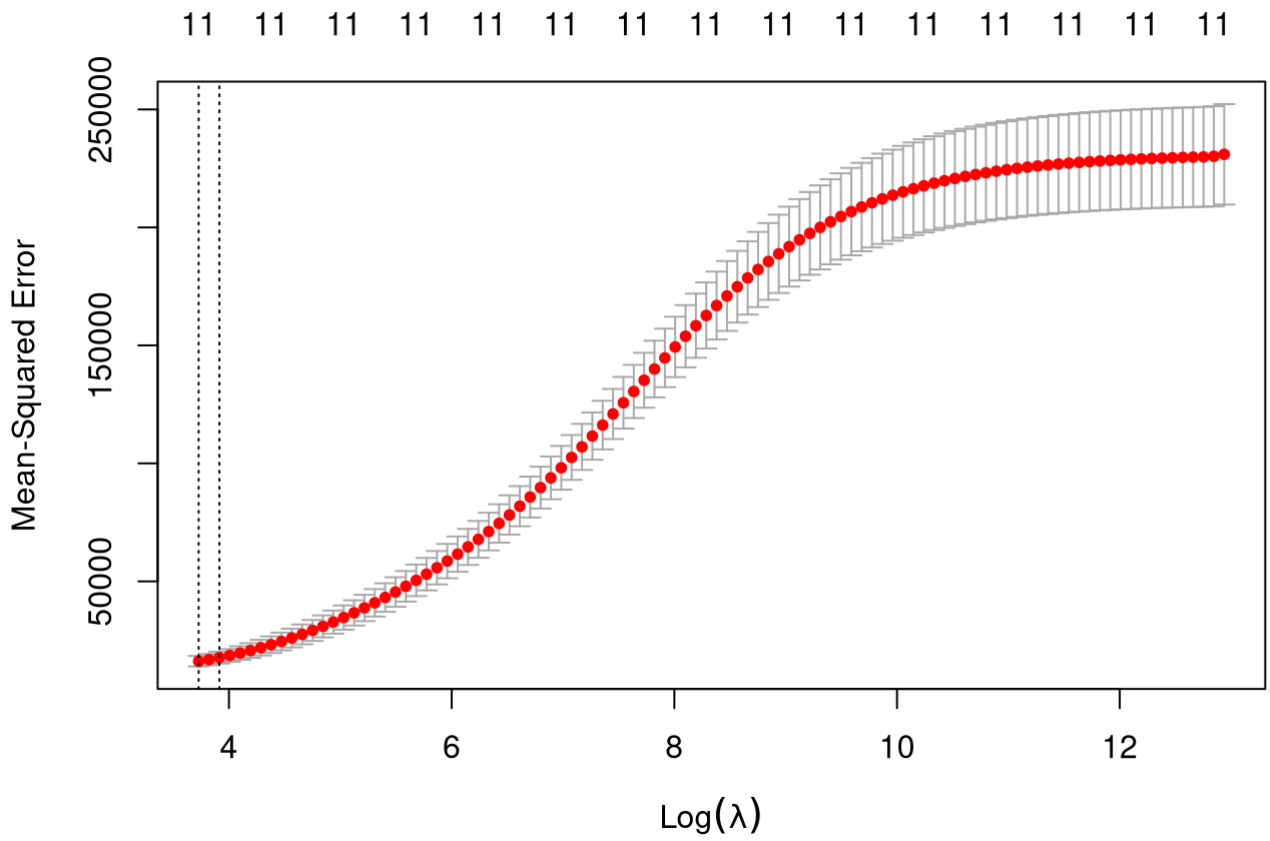

```
# Plot again with normalised coefficients
feature.sd <- apply(x[train, ], 2, sd)
ridge.coef <- coef(ridge.mod)[-1, ] # Extract the coefficients without intercept

ridge.coef.norm <- ridge.coef * feature.sd

library(RColorBrewer)
cols <- colorRampPalette(brewer.pal(8, "Dark2"))(11)
matplot(x = matrix(log10(grid), nrow = ncol(ridge.coef)),
        y = t(ridge.coef.norm), type = "l",
        ylim = range(ridge.coef.norm[, ncol(ridge.coef.norm)]),
        xlab = "log(lambda)", ylab = "standardised coef",
        col = cols)
legend("topright", legend = rownames(ridge.coef.norm)[1:7], col = cols[1:7], lty = 1:
7)
```




```
# we can use cross-validation to determine optimal lambda value. This is
# implemented as a function in the glmnet package.
set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 41.60385
```

```
# we then predict on the test data using optimal lambda determined by CV
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test, ])
# and compute the MSE
mean((ridge.pred - y.test)^2)
```

```
## [1] 16105.44
```

```
# For feature selection?
ridge.coef <- predict(ridge.mod, type = "coefficients", s = bestlam)[1:5, ]
ridge.coef
```

```
## (Intercept)      Income       Limit      Rating       Cards
## -351.272640   -4.492977    0.112235    1.591557   14.315970
```

```
which(abs(ridge.coef) > 5)
```

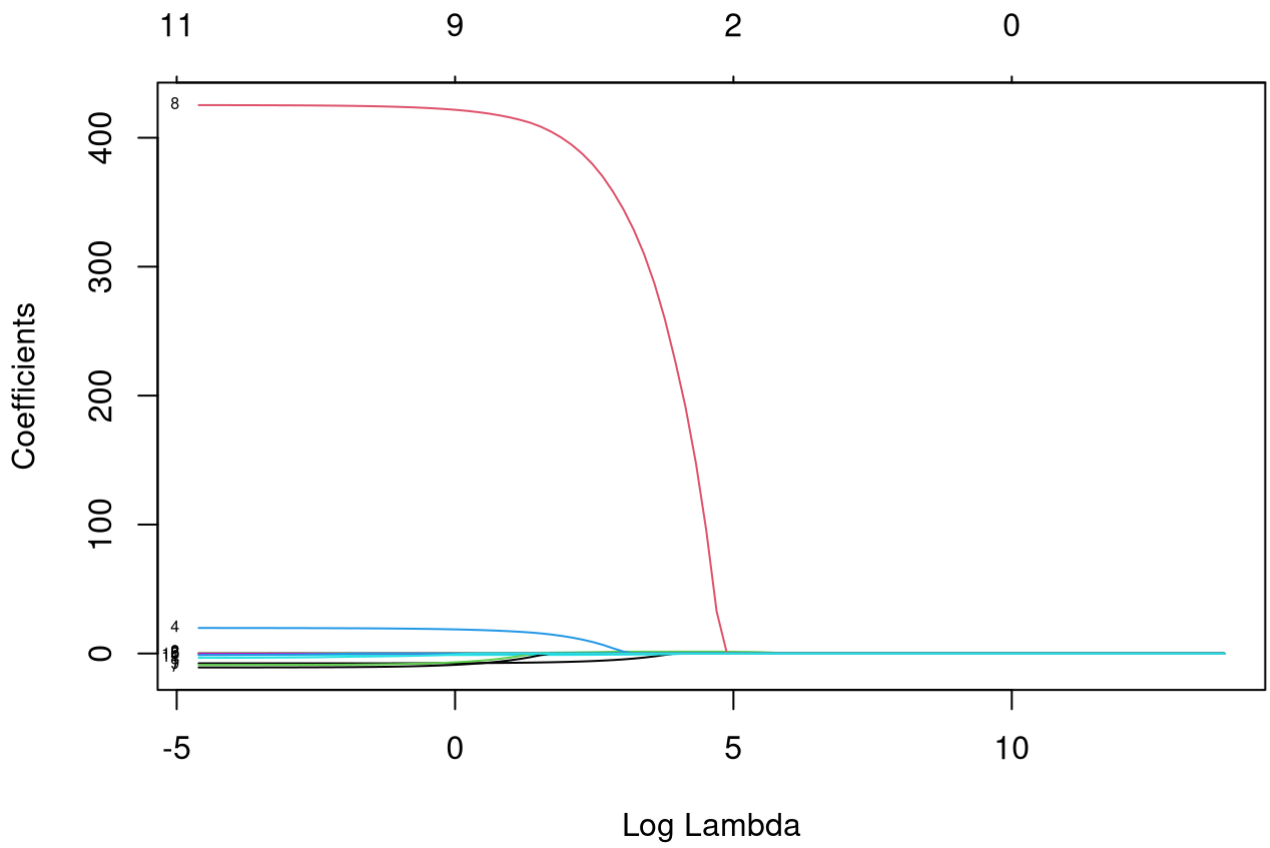
```
## (Intercept)       Cards
##           1           5
```

Lasso regression

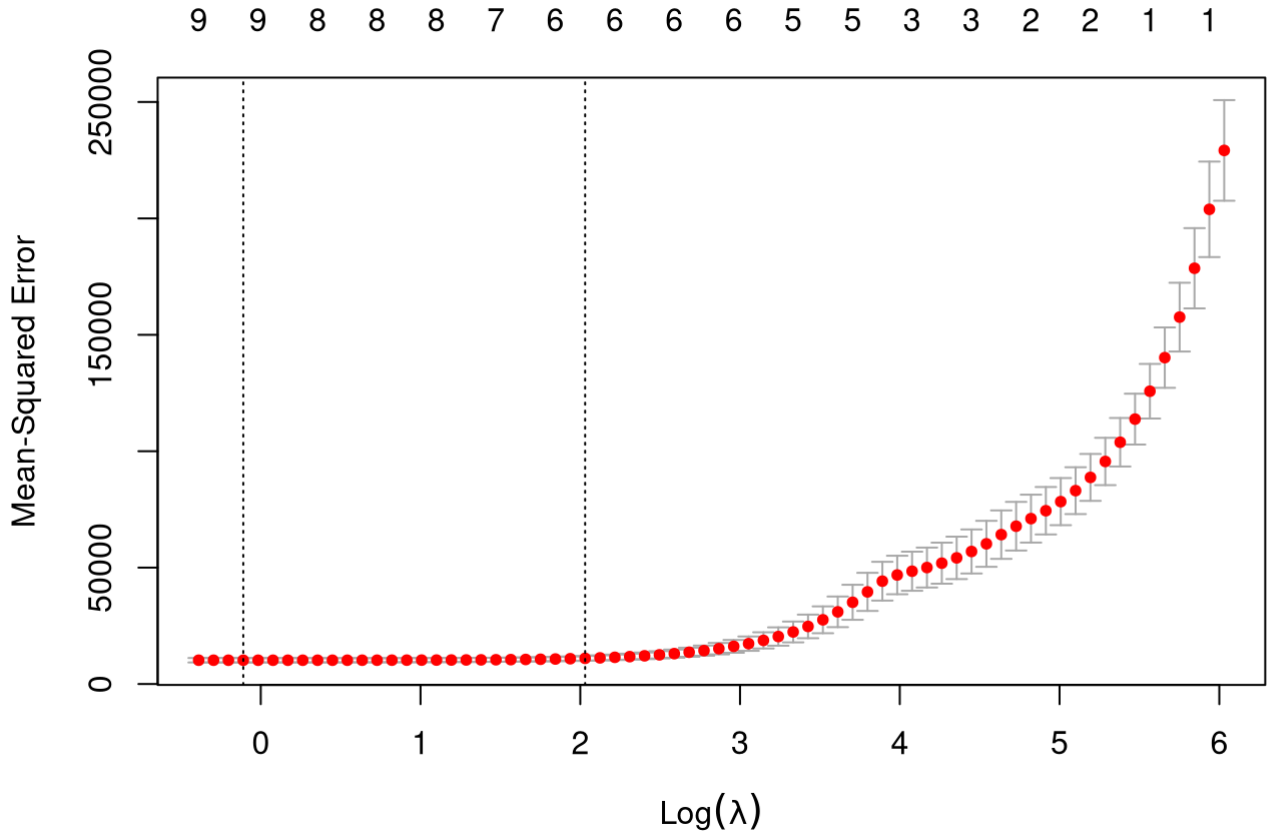
```
## Lasso model
lasso.mod <- glmnet(x[train, ], y[train], alpha = 1, lambda = grid)
dim(coef(lasso.mod))
```

```
## [1]  12 100
```

```
plot(lasso.mod, "lambda", label = TRUE)
```




```
set.seed(1)
# Using cross-validation for Lasso to find the best lambda (based on cvm "mean cross-validated error")
cv.lasso <- cv.glmnet(x[train, ], y[train], alpha = 1)
plot(cv.lasso)
```




```
bestlam <- cv.lasso$lambda.min
# predict on test set using optimal lambda value estimated by CV
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test, ])
# compute MSE
mean((lasso.pred - y.test)^2)
```

```
## [1] 10531.43
```

```
# Lasso for feature selection
lasso.coef <- predict(lasso.mod, type = "coefficients", s = bestlam)[1:5, ]
lasso.coef
```

```
##  (Intercept)       Income        Limit       Rating        Cards
## -449.9599280   -7.4846668    0.2397090    0.4097832   18.8071902
```