SLLD - Module 2

Feature Selection

S.Tonini, F.Chiaromonte

Sant'Anna School of Advanced Study - Pisa

7/3/2024

Libraries

We are going to use

```r
library(caret) # statistical learning techniques
library(leaps) # BSS
library(glmnet)
library(mvtnorm)
```

Data

We simulate data as follows

```r
n<-100
p <- 10
set.seed(1)
x <- rmvnorm(n=n, replicate(p,0), diag(p))
alpha <- c(4,1,3,0.5,2)
y <- x[,1:5]%*%alpha + rnorm(n, 0, 2)
df <- cbind(y,x)
```

LASSO

We can start this practicum with one of the methods we discovered during the last lecture: the LASSO. As you have learnt, the LASSO implicitly performs feature selection.

```
cv_lasso <- cv.glmnet(as.matrix(x), y, alpha = 1)
cv_lasso
```

```
##
## Call:  cv.glmnet(x = as.matrix(x), y = y, alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure     SE Nonzero
## min 0.0574    47   4.464 0.8385       8
## 1se 0.3687    27   5.278 0.7655       5
```

```r
se_lasso <- glmnet(as.matrix(x), y, alpha=1, lambda=
                     cv_lasso$lambda.1se)
lasso_coefs <- coef(se_lasso)
lasso_coefs
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept) 0.02052538
## V1          3.39671914
## V2          0.69268840
## V3          2.36186268
## V4          0.37305158
## V5          1.74391295
## V6          .
## V7          .
## V8          .
## V9          .
## V10         .
```

Best Subset Selection (Linear Regression)

We will use the **regsubsets()** function (part of the **leaps** library). It performs best subset selection by identifying the best model that contains a given number of predictors, where the notion of “best” is based on the in-sample RSS. No cross-validation is performed. The summary() command outputs the best set of variables for each model size.

```
regfit.full = regsubsets(y ~ ., data = data.frame(x),
                         nvmax = 10, method="exhaustive")
summary(regfit.full)$outmat
```

```
##           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10
## 1  ( 1 ) "*" " " " " " " " " " " " " " " " " " "
## 2  ( 1 ) "*" " " "*" " " " " " " " " " " " " " "
## 3  ( 1 ) "*" " " "*" " " "*" " " " " " " " " " "
## 4  ( 1 ) "*" "*" "*" " " "*" " " " " " " " " " "
## 5  ( 1 ) "*" "*" "*" "*" "*" " " " " " " " " " "
## 6  ( 1 ) "*" "*" "*" "*" "*" " " " " "*" " " " "
## 7  ( 1 ) "*" "*" "*" "*" "*" " " " " "*" " " "*"
## 8  ( 1 ) "*" "*" "*" "*" "*" " " " " "*" "*" "*"
## 9  ( 1 ) "*" "*" "*" "*" "*" " " "*" "*" "*" "*"
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

This logical matrix indicates which elements are in each model

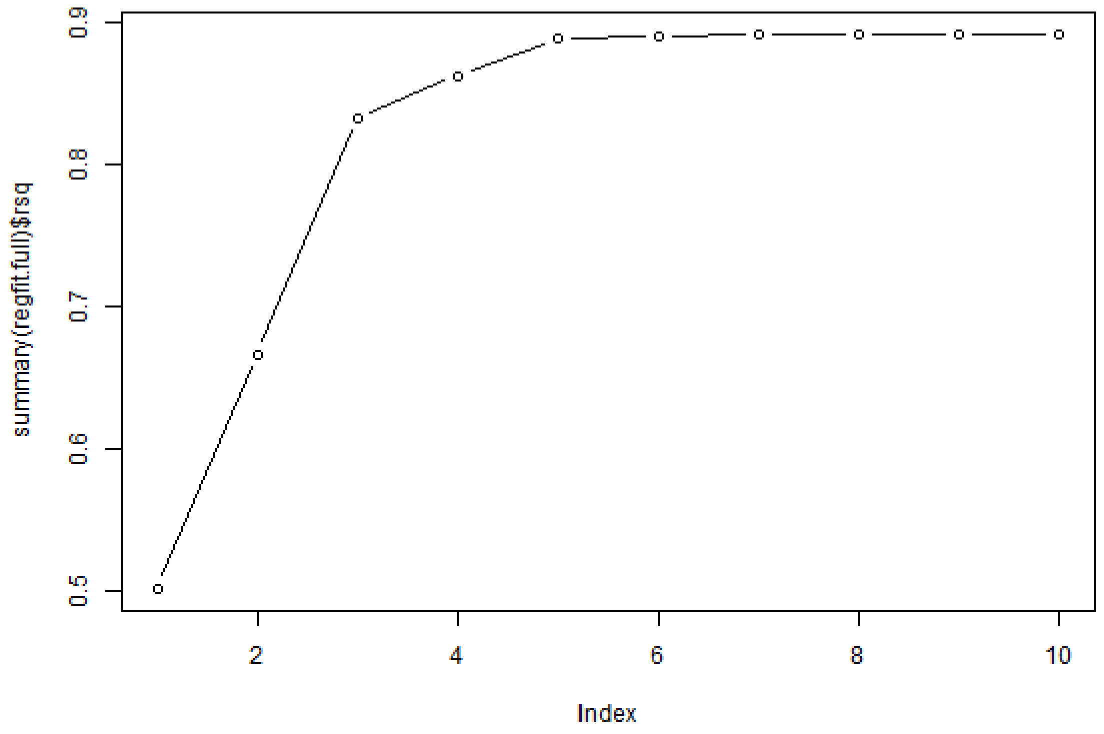
The **summary()** function also returns R2, RSS R2 adj , Cp, and BIC. We can examine these to try to select the best overall model.

```
names(summary(regfit.full))
```

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

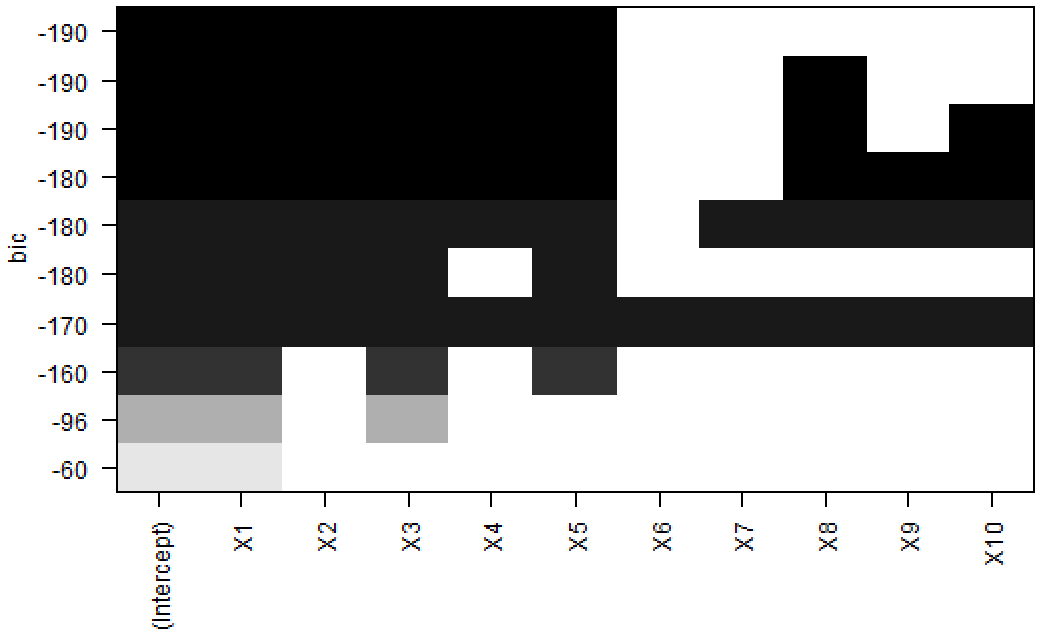
- **which:** A logical matrix indicating which elements are in each model
- **rsq:** The r-squared for each model
- **rss:** Residual sum of squares for each model
- **adjr2:** Adjusted r-squared
- **cp:** Mallows' Cp
- **bic:** Schwartz's information criterion, BIC
- **outmat:** A version of the which component that is formatted for printing
- **obj:** A copy of the regsubsets object

```
#plot rss
plot(summary(regfit.full)$rsq, type="b")
```



Here is another visualization tool which is available in leaps and that you may find useful. It plots a table of models showing which variables are in each model. The models are ordered by the specified model selection statistic. This plot is particularly useful when there are more than ten or so models and the simple table produced by **summary.regsubsets** is too big to read.

```
plot(regfit.full,scale="bic") #for "bic"
```



Tuning strategies

We split the observations into a training set and a test set (of size 70 vs 30%) to select the best model.

```
set.seed(1)
train = sample(1:nrow(df), round(nrow(df)*0.7), rep=F)
test = which(!(1:nrow(df) %in% train))
length(test)
```

```
## [1] 30
```

Now, we apply regsubsets() on the training set:

```
regfit.best <- regsubsets(y[train] ~ ., data = data.frame(x[train,]),
                          nvmax = 10, method="exhaustive")
```

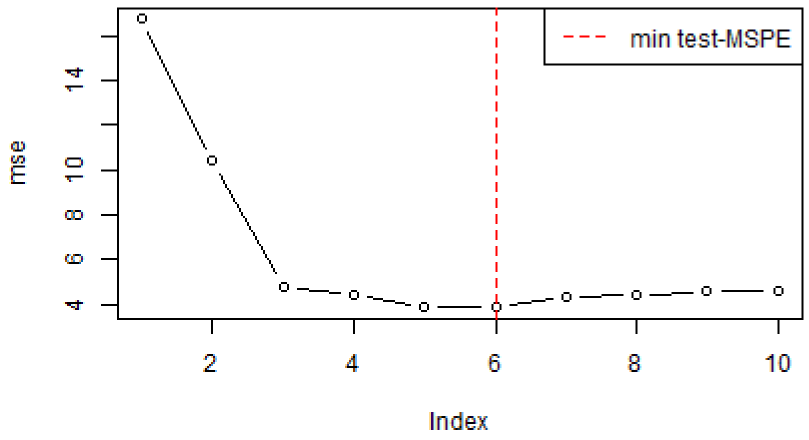
We then compute the test set error for the best model of each size.

```
test.mat = model.matrix(y[test] ~ ., data = data.frame(x[test,]))
dim(test.mat)
```

```
## [1] 30 11
```

Within a for loop, for each size i, we extract the coefficients for the best model. We need to compute all of this by hand, since the leaps library does not provide a predict function (as it is customary for R packages). Thus, we first make a model matrix containing test data:

```r
mse <- rep(NA, p) # out-of-sample MSE
for(i in 1:p){coefi=coef(regfit.best,id=i)
pred=test.mat[,names(coefi)]%*%coefi
mse[i]=mean((y[test]-pred)^2)}
plot(mse, type='b')
abline(v=which.min(mse), col="red", lty=2)
legend(x = "topright",
legend = "min test-MSPE",
lty = 2,
col = "red")
```



The best model is the one with 6 parameters (plus the intercept term). Let us see the coefficients

```
coef(regfit.best, which.min(mse))
```

```
## (Intercept)          X1          X2          X3          X4          X5 
##   0.2916780   3.9074943   1.3464891   2.9435304   1.1781320   2.1356339 
##          X8 
##  -0.3150792
```

Now we perform best subset selection on the entire dataset and select the 6 best variables. By focusing on the entire dataset, we expect to get the same 6 variables as we get from the training set alone. If this does not happen, it means that we are using data that does not allow stable selection.

```
regfit.best.full <- regsubsets(y ~ ., data=data.frame(x) ,nvmax=10)
rescompare <- rbind(coef(regfit.best, which.min(mse)),
coef(regfit.best.full, which.min(mse)))
rownames(rescompare) <- c("train", "full")
rescompare
```

```
##       (Intercept)       X1       X2       X3       X4       X5         X8
## train  0.29167804 3.907494 1.346489 2.943530 1.178132 2.135634 -0.3150792
## full   0.07893518 3.867236 1.242023 2.851341 1.070876 2.081128 -0.2548971
```

Now it's your turn!!!