Subset Selection and Shrinkage Methods

Yuan Bian

University of Western Ontario

2022/11/17

Motivations

- Provided that the true relationship between the response and the predictors is approximately linear, the least squares estimates will have low bias.
- If $n >> p$, then the least squares estimates tend to also have low variance, and hence will perform well on test observations.
- However, if n is not much larger than p, then there can be a lot of variability in the least squares fit, resulting in overfitting and consequently poor predictions on future observations not used in model training.
- If $p > n$, then there is no longer a unique least squares coefficient estimate.
- In practice, some or many of the variables are in fact not associated with the response. Including such irrelevant variables leads to unnecessary complexity in the resulting model. By removing these variables, we can obtain a model that is more easily interpreted.

Objective of Today's Session

- Subset Selection Methods
  - Best Subset Selection
  - Forward Stepwise Selection
  - Backward Stepwise Selection
- Shrinkage Methods
  - Ridge Regression
  - Lasso Regression

Subset Selection Methods

This approach involves identifying a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.

```
library(ISLR2)
options(digits=3)
Hitters <- na.omit(Hitters)
dim(Hitters)
```

```
[1] 263  20
```

```
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"
```

Best Subset Selection

- The regsubsets() function performs best subset selection by identifying the best model that contains a given number of predictors, where best is quantified by minimizing

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

- The summary() command outputs the best set of variables for each model size.

```
library(leaps)
regfit.full <- regsubsets(Salary ~ ., Hitters)
#summary(regfit.full)
```

- By default, regsubsets() only reports results up to the best eight-variable model. But the nvmax option can be used to return as many variables as are desired.

```
regfit.full <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19)
reg.summary <- summary(regfit.full)
names(reg.summary)
```

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

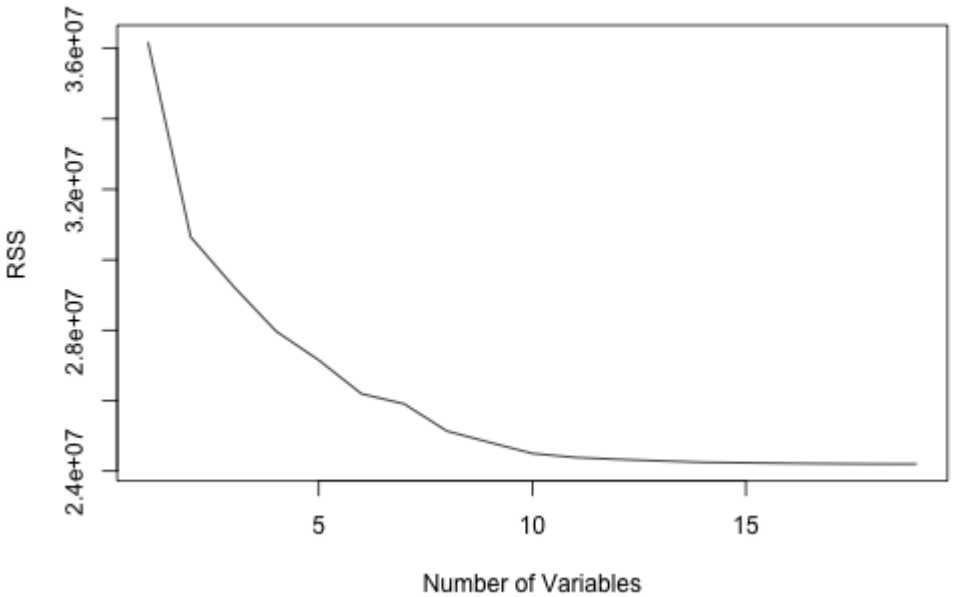
- R^2 statistic increases from 32%, when only one variable is included in the model, to almost 55%, when all variables are included.
- As expected, the R^2 statistic increases monotonically as more variables are included.

```
reg.summary$rsq
```

```
 [1] 0.321 0.425 0.451 0.475 0.491 0.509 0.514 0.529 0.535 0.540 0.543 0.544
[13] 0.544 0.545 0.545 0.546 0.546 0.546 0.546
```

Plotting RSS, adjusted R^2, and BIC for all of the models at once will help us decide which model to select.

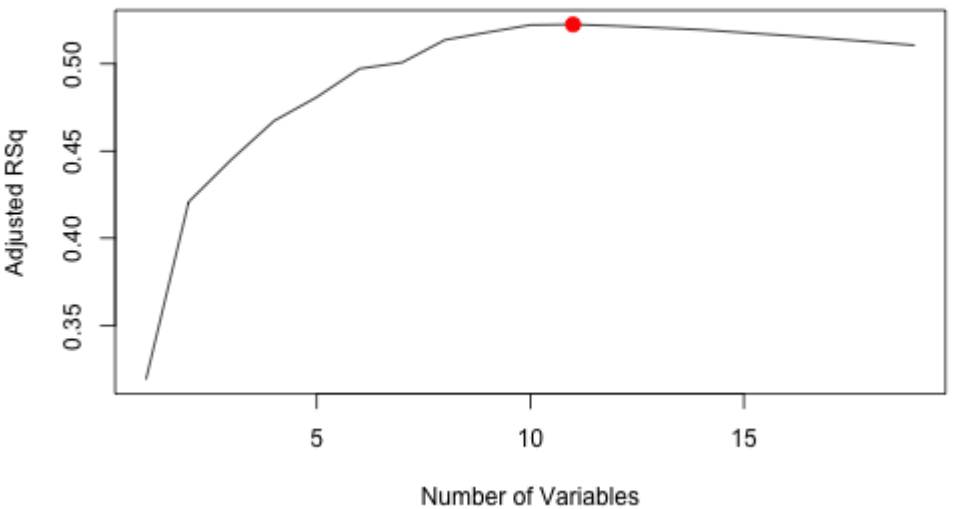
```
plot(reg.summary$rss, xlab = "Number of Variables",
    ylab = "RSS", type = "l")
```



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

```
[1] 11
```

```
plot(reg.summary$adjr2, xlab = "Number of Variables",
    ylab = "Adjusted RSq", type = "l")
points(11, reg.summary$adjr2[11], col = "red", cex = 2,
    pch = 20)
```

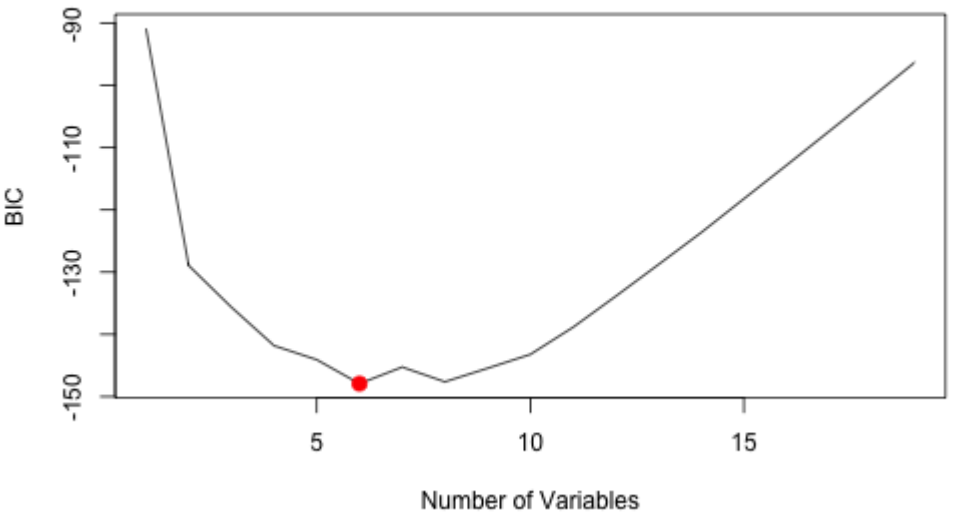


In a similar fashion we can plot the BIC statistics.

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

```
[1] 6
```

```
plot(reg.summary$bic, xlab = "Number of Variables",
    ylab = "BIC", type = "l")
points(6, reg.summary$bic[6], col = "red", cex = 2,
    pch = 20)
```



Forward and Backward Stepwise Selection

- We can also use the regsubsets() function to perform forward stepwise or backward stepwise selection, using the argument method = "forward" or method = "backward".

```
regfit.fwd <- regsubsets(Salary ~ ., data = Hitters,
    nvmax = 19, method = "forward")
regfit.bwd <- regsubsets(Salary ~ ., data = Hitters,
    nvmax = 19, method = "backward")
```

- For this data, the best one-variable through six-variable models are each identical for best subset and forward selection.
- However, the best seven-variable models identified by forward stepwise selection, backward stepwise selection, and best subset selection are different.

```
coef(regfit.full, 7)
```

```
(Intercept)        Hits       Walks      CAtBat       CHits      CHmRun
     79.451       1.283       3.227      -0.375       1.496       1.442
  DivisionW     PutOuts
   -129.987       0.237
```

```
coef(regfit.fwd, 7)
```

```
(Intercept)       AtBat        Hits       Walks        CRBI      CWalks
    109.787      -1.959       7.450       4.913       0.854      -0.305
  DivisionW     PutOuts
   -127.122       0.253
```

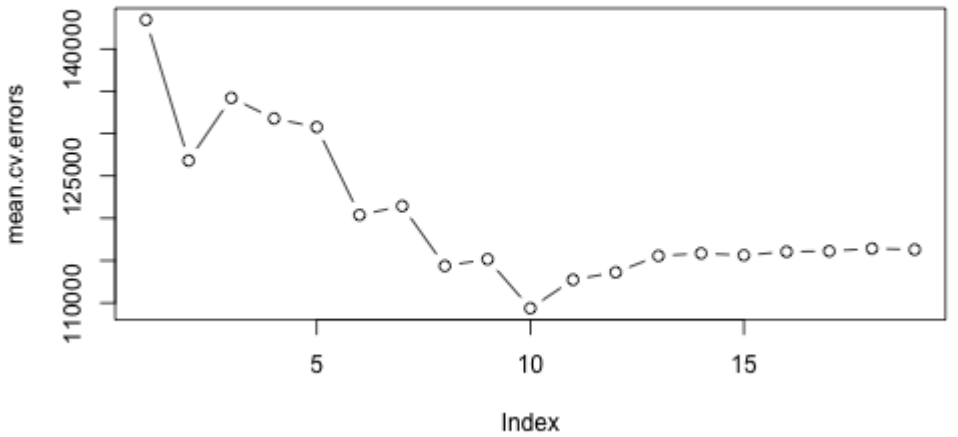
```
coef(regfit.bwd, 7)
```

```
(Intercept)       AtBat        Hits       Walks       CRuns      CWalks
    105.649      -1.976       6.757       6.056       1.129      -0.716
  DivisionW     PutOuts
   -116.169       0.303
```

Choosing Among Models Using Cross-Validation

```
predict.regsubsets <- function(object, newdata, id, ...) {
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id = id)
  xvars <- names(coefi)
  mat[, xvars] %*% coefi
}
k <- 10
n <- nrow(Hitters)
set.seed(1)
folds <- sample(rep(1:k, length = n))
cv.errors <- matrix(NA, k, 19, dimnames = list(NULL, paste(1:19)))
for (j in 1:k) {
  best.fit <- regsubsets(Salary ~ ., data = Hitters[folds != j, ],
       nvmax = 19)
  for (i in 1:19) {
    pred <- predict(best.fit, Hitters[folds == j, ], id = i)
    cv.errors[j, i] <-
         mean((Hitters$Salary[folds == j] - pred)^2)
   }
}
```

```
mean.cv.errors <- apply(cv.errors, 2, mean)
plot(mean.cv.errors, type = "b")
```




```
regfit.best <- regsubsets(Salary ~ ., data = Hitters,
    nvmax = 19)
coef(regfit.best, 10)
```

```
(Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
    162.535       -2.169        6.918        5.773       -0.130        1.408
       CRBI       CWalks    DivisionW      PutOuts      Assists
      0.774       -0.831     -112.380        0.297        0.283
```

Shrinkage Methods

- It involves fitting a model involving all p predictors. However, the estimated coefficients are shrunken towards zero relative to the least squares estimates.
- It has the effect of reducing variance.
- **Ridge Regression**: the ridge regression coefficient estimates will approach zero.
- **Lasso Regression**: some of the lasso regression coefficients may be estimated to be exactly zero. Hence, it can also perform variable selection.

```
library(glmnet)
x <- model.matrix(Salary ~ ., Hitters)[, -1]
y <- Hitters$Salary
```

Ridge Regression

- Ridge regression will always generate a model involving all p predictors.
- Ridge regression is very similar to least squares, except that the coefficients are estimated by minimizing

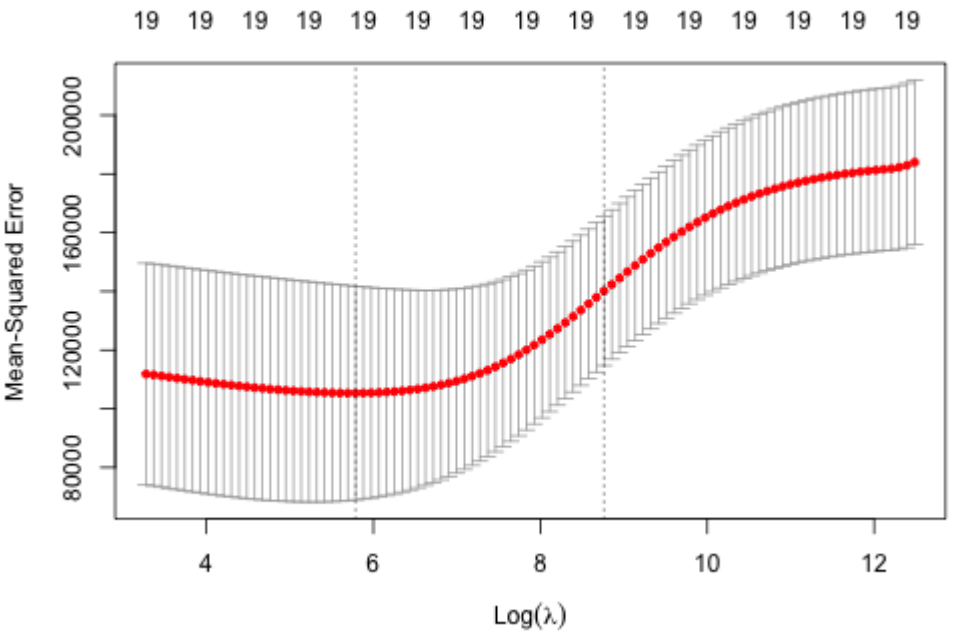
$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

```
grid <- 10^seq(10, -2, length = 100)
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

- By default the glmnet() function performs ridge regression for an automatically selected range of λ values.
- Here we have chosen to implement the function over a grid of values ranging from $\lambda = 10^{10}$ to 10^{-2}.
- Note that by default, the glmnet() function standardizes the variables so that they are on the same scale. To turn off this default setting, use the argument 'standardize = FALSE'.

- In general, it would be better to use cross-validation to choose the tuning parameter λ.

```
set.seed(1)
train <- sample(1:nrow(x), nrow(x) / 2)
test <- (-train)
y.test <- y[test]
set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0)
plot(cv.out)
```



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

```
[1] 326
```

```
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test, ])
mean((ridge.pred - y.test)^2)
```

```
[1] 119114
```

```
out <- glmnet(x, y, alpha = 0)
predict(out, type = "coefficients", s = bestlam)[1:20, ]
```

```
(Intercept)       AtBat        Hits       HmRun        Runs         RBI
    15.4438      0.0772      0.8591      0.6010      1.0637      0.8794
      Walks       Years      CAtBat       CHits      CHmRun       CRuns
     1.6244      1.3525      0.0113      0.0575      0.4068      0.1146
       CRBI      CWalks     LeagueN   DivisionW     PutOuts     Assists
     0.1212      0.0530     22.0914    -79.0403      0.1662      0.0294
     Errors  NewLeagueN
    -1.3609      9.1249
```

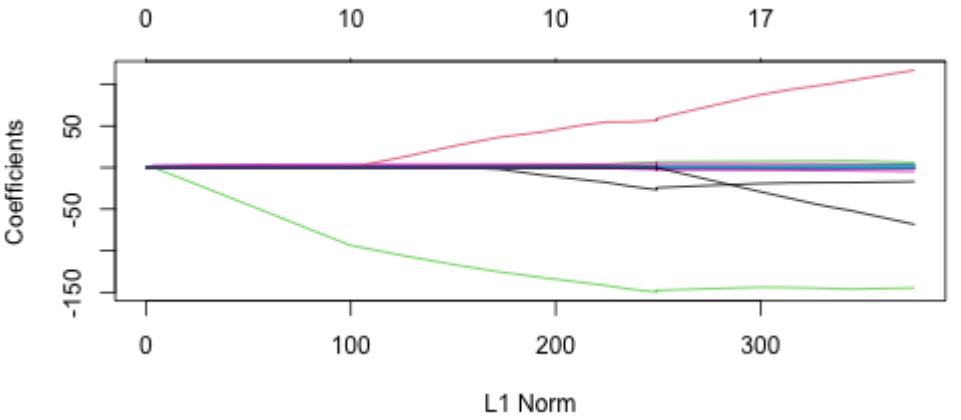
- As expected, none of the coefficients are zero — ridge regression does not perform variable selection!

Lasso Regression

- Lasso regression is a relatively recent alternative to ridge regression that perform variable selection. The lasso coefficients minimize

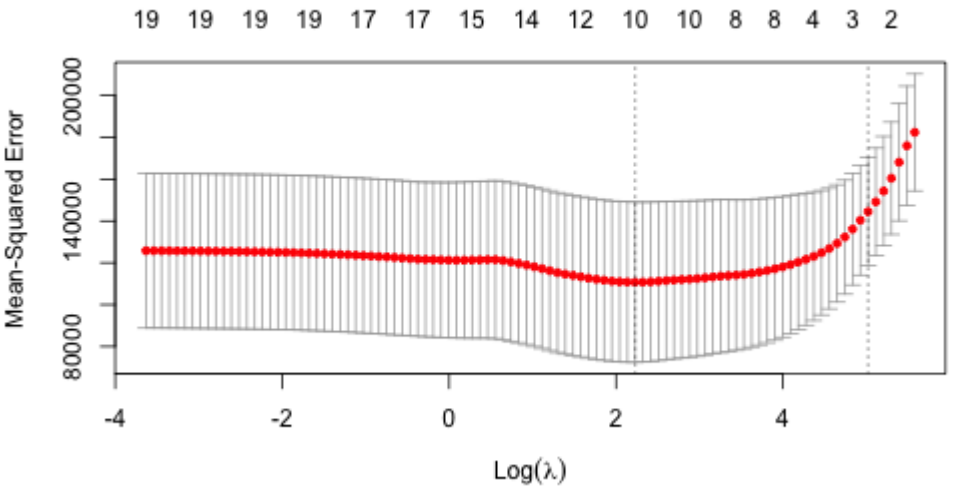
$$RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

```
lasso.mod <- glmnet(x[train, ], y[train], alpha = 1, lambda = grid)
plot(lasso.mod)
```



- perform cross-validation and compute the associated test error

```
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)
```

```
[1] 143674
```

```
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         RBI
     1.2748     -0.0550      2.1803      0.0000      0.0000      0.0000
      Walks       Years      CAtBat       CHits      CHmRun       CRuns
     2.2919     -0.3381      0.0000      0.0000      0.0283      0.2163
       CRBI      CWalks     LeagueN   DivisionW     PutOuts     Assists
     0.4171      0.0000     20.2862   -116.1676      0.2375      0.0000
     Errors  NewLeagueN
    -0.8563      0.0000
```

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

```
(Intercept)       AtBat        Hits       Walks       Years      CHmRun
     1.2748     -0.0550      2.1803      2.2919     -0.3381      0.0283
      CRuns        CRBI     LeagueN   DivisionW     PutOuts      Errors
     0.2163      0.4171     20.2862   -116.1676      0.2375     -0.8563
```

Reference

James, G, Witten, D, Hastie, T, Tibshirani, R (2013) *An Introduction to Statistical Learning*. Springer, New York, Second edition.