Regression and Regularization

Olena Smotrova

22/02/2018

- Multiple Linear Regression
- Ridge Regression
- The Lasso Regression
- Refefences

Multiple Linear Regression

Linear regression is a very simple approach for supervised learning. Y is a response variable, $X_1, X_2, \ldots, X_p$ are p distinct predictors. Multiple linear regression model takes the form

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p + \epsilon, \tag{1}$$

$\beta_0, \beta_1, \ldots, \beta_p$ are unknown and must be estimated. The parameters are estimated using least square approach. We choose $\beta_0, \beta_1, \ldots, \beta_p$ to minimize the *sum of square residuals* (RSS)

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \ldots - \hat{\beta}_p x_{ip})^2 \tag{2}$$

The values $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ that minimize RSS are the multiple least squares regression coefficient estimates.

Mean square error (MSE)

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3}$$

The MSE will be small if the predicted response are very close tj the true response, and will be large if some of the observations the predicted and true responses differ substantially. The quality of a linear regression fit is typically assessed using two related quantities: the *regular standart error* (RSE) and R^2 statistics. The RSE is an estimate of the standard deviation of ϵ.

$$RSE = \sqrt{\frac{1}{n - p - 1} RSS} \tag{4}$$

$$R^2 = 1 - \frac{RSS}{TSS} \tag{5}$$

$$TSS = \sum_{i=1}^{n} (y_i - \overline{y}_i)^2 \tag{6}$$

The TSS is *total sum of squares*, $\overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$. The TTS measures the total variance in the response Y. R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression.

```
diabets <- read.csv('Data/diabetes-data.csv', header = FALSE )

names(diabets) = c('age', 'sex', 'body mass index', 'blood pressure',
                   'serum1', 'serum2', 'serum3', 'serum4', 'serum5', 'serum6', 'y')
str(diabets)
```

```
## 'data.frame':    442 obs. of  11 variables:
##  $ age            : int  59 48 72 24 50 23 36 66 60 29 ...
##  $ sex            : int  2 1 2 1 1 1 2 2 2 1 ...
##  $ body mass index: num  32.1 21.6 30.5 25.3 23 22.6 22 26.2 32.1 30 ...
##  $ blood pressure : num  101 87 93 84 101 89 90 114 83 85 ...
##  $ serum1         : int  157 183 156 198 192 139 160 255 179 180 ...
##  $ serum2         : num  93.2 103.2 93.6 131.4 125.4 ...
##  $ serum3         : num  38 70 41 40 52 61 50 56 42 43 ...
##  $ serum4         : num  4 3 4 5 4 2 3 4.55 4 4 ...
##  $ serum5         : num  4.86 3.89 4.67 4.89 4.29 ...
##  $ serum6         : int  87 69 85 89 80 68 82 92 94 88 ...
##  $ y              : int  151 75 141 206 135 97 138 63 110 310 ...
```

To fit linear regression divide data set into training and test subsets.

```
index = sample(nrow(diabets), size = 100)
Train = diabets[-index, ]
Test = diabets[index, ]
```

Fit linear regression model on training set

```
lm.mod = lm(y ~ ., data = Train)
summary(lm.mod)
```

```
##
## Call:
## lm(formula = y ~ ., data = Train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -151.548  -39.021   -0.421   41.239  157.097
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -300.03616   80.15936  -3.743 0.000214 ***
## age                  0.06745    0.25236   0.267 0.789411
## sex                -24.12663    6.77392  -3.562 0.000423 ***
## `body mass index`    5.42423    0.83430   6.502 2.93e-10 ***
## `blood pressure`     1.15884    0.26911   4.306 2.19e-05 ***
## serum1              -0.86285    0.65942  -1.308 0.191614
## serum2               0.53412    0.61401   0.870 0.384997
## serum3               0.18711    0.92023   0.203 0.838998
## serum4               8.75179    7.40503   1.182 0.238105
## serum5              59.82133   18.77516   3.186 0.001579 **
## serum6               0.10341    0.31399   0.329 0.742108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 55.4 on 331 degrees of freedom
## Multiple R-squared:  0.4938, Adjusted R-squared:  0.4785
## F-statistic: 32.28 on 10 and 331 DF,  p-value: < 2.2e-16
```

Predict values of y on test set and assessing accuracy

```
lm.pred = predict(lm.mod, newdata = Test)

TSS = sum((Test$y - mean(Test$y))**2)
RSS = sum((Test$y - lm.pred)**2)

R_2 = 1 - RSS/TSS
paste("Test R2 lm:", as.character(R_2))
```

```
## [1] "Test R2 lm: 0.586235983466942"
```

```
MSE = mean((Test$y - lm.pred)**2)
paste("Test MSE lm:", as.character(MSE))
```

```
## [1] "Test MSE lm: 2529.45942463138"
```

Ridge Regression

The ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{7}$$

where $\lambda \geq 0$ is a tuning parameter. Second term $\lambda \sum_{j=1}^{p} \beta_j^2$ is called a shrinkage penalty. Shrinkage penalty is small, when $\beta_0, \beta_1, \ldots, \beta_p$ are close to 0. It has effect of shrinking the estimates β_j towards 0. At $\lambda = 0$ penalty has no effect. $\lambda \to \infty$ hence $\hat{\beta}^R \to 0$. Selecting a good value for λ is critical. See [1] for more details.

For ridge and Lasso `glmnet` package is used. Parameter `alpha=1` is the lasso penalty, and `alpha=0` the ridge penalty.

```
library(glmnet)
```

```
x = model.matrix(y~., diabets)[, -1]
y = diabets$y

ridge.mod = glmnet(x[-index, ], y[-index], alpha = 0)
```

We use cross-validation to choose the tunning parameter λ. By default, function `cv.glmnet()` performs ten-fold cross-validation.

```
cv.out = cv.glmnet(x[-index, ], y[-index], alpha = 0)
best.lam = cv.out$lambda.min
best.lam
```

```
## [1] 6.298059
```

```
ridge.pred = predict(ridge.mod, s = best.lam, newx = x[index, ])
```

```
MSE = mean((diabets$y[index] - ridge.pred)**2 )
MSE
```

```
## [1] 2606.32
```

```
TSS = sum((Test$y - mean(Test$y))**2)
RSS = sum((Test$y - ridge.pred)**2)

R_2 = 1 - RSS/TSS
paste("Test R2 cv-ridge:", as.character(R_2))
```

```
## [1] "Test R2 cv-ridge: 0.573663352974165"
```

The Lasso Regression

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j| \tag{8}$$

The lasso penalty is $\lambda \sum_{j=1}^{p} |\beta_j|$. At $\lambda = 0$ penalty has no effect. $\lambda \to \infty$ hence $\hat{\beta}^L \to 0$. The lasso performs variable selection. We say that the lasso yields sparse models - that is, models that involves only a subset of the variables. Selection of λ is performed by cross=validation. See [1] for more details.

Note $n \gg p$ Least square estimates good performs. $p > n$ no unique least squares coefficient estimate. $n \geq p$ poor prediction with min RSS.

Parameter `alpha=1`

```
lasso.mod = glmnet(x[-index, ], y[-index], alpha = 1)
cv.out = cv.glmnet(x[-index, ], y[-index], alpha = 1)
best.lam = cv.out$lambda.min
best.lam
```

```
## [1] 1.835918
```

```
lasso.pred = predict(lasso.mod, s = best.lam,  newx = x[index, ])

MSE = mean((diabets$y[index] - lasso.pred)**2 )
MSE
```

```
## [1] 2626.346
```

```
TSS = sum((Test$y - mean(Test$y))**2)
RSS = sum((Test$y - lasso.pred)**2)

R_2 = 1 - RSS/TSS
paste("Test R2 cv-lasso:", as.character(R_2))
```

```
## [1] "Test R2 cv-lasso: 0.570387493411837"
```

Refefences

[1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, *An Introduction to Statistical Learning with Applications in R*, **2013**.