Handmade Linear Regression

Álvaro Orgaz Expósito

Theory

First, let's see the mathematical formulas for the *gradient descendent* used in the optimization code. Since we will predict a continuous variable (then it is a *regression* problem), the cost function J to optimize will be the *MSE* or *Mean Square Error*:

$$J = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip}))^2$$

where n is the number of samples or observations in the dataset and p is the number of explanatory features.

Then, the formula for the gradient of J is:

$$\frac{\partial J}{\partial w} = \begin{pmatrix} \frac{\partial J}{\partial w_0} \\ \cdots \\ \frac{\partial J}{\partial w_p} \end{pmatrix} = \begin{pmatrix} r\frac{1}{n}\sum_{i=1}^{n} 2(y_i - \hat{y_i})(\frac{\partial (y_i - \hat{y_i})}{\partial w_0}) \\ \cdots \\ \frac{1}{n}\sum_{i=1}^{n} 2(y_i - \hat{y_i})(\frac{\partial (y_i - \hat{y_i})}{\partial w_p}) \end{pmatrix} = \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n} 2(y_i - (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip}))(-1) \\ \cdots \\ \frac{1}{n}\sum_{i=1}^{n} 2(y_i - (w_0 + w_1 x_{i1} + \cdots + w_p x_{ip}))(-x_{ip}) \end{pmatrix}$$

and we will use the gradient of J to update the weights in each iteration of the optimization process:

$$w^{new} = \begin{pmatrix} w_0^{new} \\ \cdots \\ w_p^{new} \end{pmatrix} = \begin{pmatrix} w_0 - \eta\frac{\partial J}{\partial w_0} \\ \cdots \\ w_p - \eta\frac{\partial J}{\partial w_p} \end{pmatrix}$$

where η is the learning rate parameter.

Code

The data used is the popular dataset *iris*. Let's predict the variable *Sepal.Length* with the explanatory features: *Sepal.Width*, *Petal.Length* and *Petal.Width*.

```
x1 <- iris$Sepal.Width
x2 <- iris$Petal.Length
x3 <- iris$Petal.Width
y <- iris$Sepal.Length
n <- nrow(iris)
```

Set initial values for the weights:

```
w0 <- 0
w1 <- 0
w2 <- 0
w3 <- 0
```

Start the weights optimization with *gradient descent*:

```
learning <- 0.001
rounds <- 1000000
for(i in 1:rounds){
  y_predicted <- w0+w1*x1+w2*x2+w3*x3
  # Calculate the J gradient by weights
```

```r
  w0_gradient <- 1/n*sum(2*(y-y_predicted)*-1)
  w1_gradient <- 1/n*sum(2*(y-y_predicted)*-x1)
  w2_gradient <- 1/n*sum(2*(y-y_predicted)*-x2)
  w3_gradient <- 1/n*sum(2*(y-y_predicted)*-x3)
  # Update the weights
  w0 <- w0-w0_gradient*learning
  w1 <- w1-w1_gradient*learning
  w2 <- w2-w2_gradient*learning
  w3 <- w3-w3_gradient*learning
}
```

Let's see the optimal estimated weights:

```r
c(w0,w1,w2,w3)
```

```
## [1]  1.8559975  0.6508372  0.7091320 -0.5564827
```

Now, compare our optimal weights with the implemented function in R for linear models:

```r
lm(y~1+x1+x2+x3)$coefficients
```

```
## (Intercept)          x1          x2          x3
##   1.8559975   0.6508372   0.7091320  -0.5564827
```