Handmade Linear Regression

Álvaro Orgaz Expósito

THEORY

The aim of the *linear regression* is to predict a continuous variable y (then it is a *regression* problem) knowing the explanatory variables $x = \{x_1, \dots, x_p\}$.

Firstly, let's see the mathematical formulas for the *gradient descent* used optimizing the loss function. Firstly, we will predict a continuous variable with the *linear predictor* $z = w_0 + w_1 x_1 + ... + w_p x_p$ and we assume that z predicts the expected value of y for a given x whichs in probability terms is $\hat{y} = E[y|x]$. Now, assuming a Gaussian distribution for $y \sim Normal(\mu = E[y|x], \sigma^2)$, we will use the *likelihood function* of our data with n independent samples for defining a function to optimze respect to weights.

$$L = \prod_i^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y_i - E[y_i|x_i])^2}{2\sigma^2}} = \prod_i^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y_i - w\cdot x_i)^2}{2\sigma^2}} \quad and \quad since \quad \frac{\partial L}{\partial w} \equiv \frac{\partial log(L)}{\partial w}$$

$$then \quad log(L) = \sum_i^n log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y_i - w\cdot x_i)^2}{2\sigma^2}}\right) = \sum_i^n log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{(y_i - w\cdot x_i)^2}{2\sigma^2}$$

Thus, the loss function J to optimize (minimize) will be $-log(L)$ which corresponds to the *MSE* or *Mean Square Error* (after deleting constants to the previous function that do not affect the gradient)

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

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

Finally, the formula for the gradient of J is

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

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

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

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){
  # Predict the data
  y_predicted <- w0+w1*x1+w2*x2+w3*x3
  # Calculate the J gradient by weights
  error <- y-y_predicted
  w0_gradient <- 1/n*sum(2*error*-1)
  w1_gradient <- 1/n*sum(2*error*-x1)
  w2_gradient <- 1/n*sum(2*error*-x2)
  w3_gradient <- 1/n*sum(2*error*-x3)
  # Update the weights
  w0 <- w0-learning*w0_gradient
  w1 <- w1-learning*w1_gradient
  w2 <- w2-learning*w2_gradient
  w3 <- w3-learning*w3_gradient
}
```

Let's see the optimal estimated weights.

```
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 regression* and see how it provides the same estimation for the weights.

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

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