Handmade Logistic Regression

Álvaro Orgaz Expósito

THEORY

The aim of the *logistic regression* is to predict the probability of positive value in the binary variable y (then it is a *classification* problem) knowing the explanatory variables $x = \{x_1, \dots, x_p\}$.

Firstly, you need to understand how the *logistic regression* use the *linear predictor* $z = w_0 + w_1 x_1 + ... + w_p x_p$ to predict $P(y = 1|x) = \pi$. As $-\infty \leq z \leq \infty$ but we want to predict $0 \leq \pi \leq 1$, we need a *link function* $g(z) = \pi$. Using the log-odds $-\infty \leq log(\frac{\pi}{1-\pi}) \leq \infty$ we can isolate π from the expression

$$log(\frac{\pi}{1-\pi}) = z = w_0 + w_1 x_1 + \cdots + w_p x_p$$

getting the popular *sigmoid link function*

$$\frac{\pi}{1-\pi} = e^z \quad \rightarrow \quad \pi = e^z - \pi e^z \quad \rightarrow \quad \pi + \pi e^z = e^z \quad \rightarrow \quad \pi = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}} = sigmoid(z)$$

Secondly, you need to understand the *likelihood function* for the distribution of the target variable y which is a collection of n *Bernoulli* independent trials $y = \{y_1, ..., y_n\}$ where $y_i = \{0, 1\}$ with respective probability of positive value $P(y_i = 1|x_i) = \pi_i$.

$$L = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i}$$

The *likelihood function* L will be used as the cost function in the *gradient descent* optimization. The logic under this product of probabilities is: when $y_i = 1$ we should predict $\pi_i \approx 1$ and when $y_i = 0$ we should predict $\pi_i \approx 0$. The perfect prediction makes $L = 1$, but if the prediction is poor L will go down.

Thirdly, let's see the mathematical formula for the *gradient descendent* used in the optimization code. Since we will predict a binary variable, the cost function J to optimize (minimize) will be the negative logarithm (just to simplify the gradient calculation) of the *likelihood function*.

$$J = -log(L) = -\sum_{i=1}^{n} log(\pi_i^{y_i}) + log((1-\pi_i)^{1-y_i}) = -\sum_{i=1}^{n} y_i log(\pi_i) + (1-y_i) log(1-\pi_i)$$

where as defined above

$$\pi_i = P(y_i = 1|x_i) = sigmoid(z_i) = \frac{1}{1+e^{-z_i}} = \frac{1}{1+e^{-(w_0 + w_1 x_{i1} + \cdots + w_p x_{ip})}}$$

and n is the number of samples or observations with p explanatory features.

Then, assuming $\pi_i = \hat{y}_i$ for simplicity, the formula for the gradient of J is

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

where using the *chain rule* for derivatives

$$\frac{\partial \hat{y}_i}{\partial w_p} = \frac{\partial sigmoid(z_i)}{\partial z_i} \frac{\partial z_i}{\partial w_p} = \frac{e^{-z_i}}{(1+e^{-z_i})^2} x_i p$$

and

$$\frac{\partial(1-\hat{y_i})}{\partial w_p} = -\frac{\partial \hat{y_i}}{\partial w_p}$$

But this expression gives a lot of computational errors in the optimization, then let's simplify it as

$$\frac{\partial \hat{y_i}}{\partial w_p} = \frac{1}{(1+e^{-z_i})}\frac{e^{-z_i}}{(1+e^{-z_i})}x_ip = \frac{1}{1+e^{-z_i}}\left(\frac{1+e^{-z_i}}{1+e^{-z_i}} - \frac{1}{(1+e^{-z_i})}\right)x_ip = \hat{y_i}(1-\hat{y_i})x_ip$$

Then, substituting in the previous formula, the final gradient of J is

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

$$= \begin{pmatrix} -\sum_{i=1}^n y_i - y_i\hat{y_i} - \hat{y_i} + y_i\hat{y_i} \\ \cdots \\ -\sum_{i=1}^n y_ix_{ip} - y_i\hat{y_i}x_{ip} - \hat{y_i}x_{ip} + y_i\hat{y_i}x_{ip} \end{pmatrix} = \begin{pmatrix} -\sum_{i=1}^n (y_i - \hat{y_i}) \\ \cdots \\ -\sum_{i=1}^n (y_i - \hat{y_i})x_{ip} \end{pmatrix}$$

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 *titanic_train* in R library *titanic*. Let's predict the binary variable *Survived* with the explanatory features: *Fare* and *Age*.

```
library(titanic)
x1 <- titanic_train[complete.cases(titanic_train[,c("Fare","Age","Survived")]),"Fare"]
x2 <- titanic_train[complete.cases(titanic_train[,c("Fare","Age","Survived")]),"Age"]
y <- titanic_train[complete.cases(titanic_train[,c("Fare","Age","Survived")]),"Survived"]
```

Note: The function *complete.cases* help us to select only the observations without missing values in the needed variables.

Set initial values for the weights.

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

Start the weights optimization with *gradient descent*.

```
sigmoid <- function(z){
  1/(1+exp(-z))
}
learning <- 0.000001
rounds <- 1000000
for(i in 1:rounds){
  z <- w0+w1*x1+w2*x2
  y_predicted <- sigmoid(z)
```

```r
  # Calculate the J gradient by weights
  w0_gradient <- -sum((y-y_predicted)*1)
  w1_gradient <- -sum((y-y_predicted)*x1)
  w2_gradient <- -sum((y-y_predicted)*x2)
  # Update the weights
  w0 <- w0-w0_gradient*learning
  w1 <- w1-w1_gradient*learning
  w2 <- w2-w2_gradient*learning
}
```

Let's see the optimal estimated weights.

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

```
## [1] -0.41705506  0.01725837 -0.01757841
```

Now, compare our optimal weights with the implemented function in R for *logistic regression*.

```r
glm(y~x1+x2,family="binomial")$coefficients
```

```
## (Intercept)          x1          x2
## -0.41705506  0.01725837 -0.01757841
```