Logistic Regression

Mohammed Luqmaan Akhtar

October 2, 2025

Suppose that we are given the weights of several mice and whether they are obese or not. Our task is to predict whether a mouse (whose data was not provided) is obese or not based on its weight. We achieve this using logistic regression.
In logistic regression, we predict the probability of a certain event being true.

Mathematics behind Logistic Regression

- The data is in the form of a table. One column contains the value which needs to be classified as true or false (for example: obese or not obese) and the other columns contain the parameters the predicted value depends on.
- First we take linear regression of the data:

$$z = w \cdot x + b \tag{1}$$

  where w is the set of weights, x is the value of the parameters and b is the bias.

- Now we map z to the **Sigmoid Function**. This function converts z to a probability between 0 and 1.

$$p = \frac{1}{1 + e^{-z}} \tag{2}$$

  Here, we can clearly observe that if $\lim z \to \infty$ then $p \approx 1$ and if $\lim z \to -\infty$ then $p \approx 0$

- To optimize the value of the weights and bias, we need to define a function that we can either maximize or minimize.
  If the actual value of the data is 1 then we want the probability (of prediction being true) to be close to 1. $P(y = 1|p) = p$.
  Similarly, if the actual value of the data is 0 then we want the probability (of prediction being true) to be close to 0. $P(y = 0|p) = 1 - p$.

Combining these two conditions, we get the likelihood function.

$$L = \prod_{i=1}^{N} p_i^{y_i}(1-p_i)^{1-y_i} \tag{3}$$

where N is the total number of data points.
if y=1 it equals p
if y=0 it equals 1-p
We need to maximise this function, but the product will give a very small value; thus, we take log.

- Taking the log of this likelihood function, we get:

$$logL = \sum_{i=1}^{N} [y_i log p_i + (1-y_i) log(1-p_i)] \tag{4}$$

We could maximize this function, or we could we could minimize the negative mean of it. Since it will be easier to use gradient descent to minimize the negative mean of it, we will do so:

$$J = -\frac{1}{N} \sum_{i=1}^{N} [y_i log p_i + (1-y_i) log(1-p_i)] \tag{5}$$

- The value of J is minimized using the gradient descent method. Gradient Descent is an iterative optimization algorithm used to minimize a function by adjusting model parameters in the direction of the steepest descent of the function's gradient. In simple terms, it finds the optimal values of weights and biases by gradually reducing the error between predicted and actual outputs. The equation for gradient descent are:

$$w = w - \gamma \frac{\partial J}{\partial w} \tag{6}$$

where w is any weight and γ is the learning rate.

To calculate the gradient with respect to our model, first, we simplify the function J (for the sake of simplicity of calculation, we will consider only one parameter x):

$$\begin{aligned} J &= -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\frac{1}{1+e^{-}wx+b}) + (1-y_i)log(1-\frac{1}{1+e^{-}wx+b})] \\ &= -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\frac{e^{wx+b}}{1+e^{wx+b}}) + (1-y_i)log(\frac{1}{1+e^{wx+b}})] \\ &= -\frac{1}{N}\sum_{i=1}^{N}[y_i(wx+b) - y_i log(1+e^{wx+b}) + (1-y_i)(-log(1+e^{wx+b})] \\ &= -\frac{1}{N}\sum_{i=1}^{N}[y_i(wx+b) - log(1+e^{wx+b})] \end{aligned} \tag{7}$$

Now, calculating the gradient:

$$\begin{aligned} \frac{\partial J}{\partial w} &= \frac{\partial(-\frac{1}{N}\sum_{i=1}^{N}[y_i(wx+b) - log(1+e^{wx+b})])}{\partial w} \\ &= -\frac{1}{N}\sum_{i=1}^{N}[y_i x_i - \frac{e^{wx_i+b}x_i}{1+e^{wx_i+b}}] \\ &= -\frac{1}{N}\sum_{i=1}^{N}[y_i - p_i]x_i \end{aligned} \tag{8}$$

Similarly for $w_1, w_2, \ldots, w_m$ replace x_i with $x_{i1}, x_{i2}, \ldots, x_{im}$

$$\frac{\partial J}{\partial b} = -\frac{1}{N}\sum_{i=1}^{N}[y_i - p_i] \tag{9}$$

Now, the values of w and b are updated as:

$$w_j = w_j - \gamma\frac{\partial J}{\partial w_j} \tag{10}$$

$$b = b - \gamma\frac{\partial J}{\partial b} \tag{11}$$

They are updated until the convergence condition of the regression is reached.

- The learning rate γ chosen by me is 0.001 and the value of the weights and parameters are updated until the change in J in the current and previous iteration is less than $10^{-}8$. The learning rate ensures that no overstepping occurs and the convergence condition ensures that the parameters are optimized.

- The data, before processing, is also normalized and after processing the parameters are converted to what it should be for non-normalized data. The normalization is done by the following equation:

$$x_{norm,j} = \frac{x_j - mean(x_j)}{std(x_j)} \tag{12}$$

And the parameters are converted back in the following way:

$$\begin{aligned} z &= w_{norm,1}x_{norm,1} + w_{norm,2}x_{norm,2} + \cdots + w_{norm,m}x_{norm,m} + b_{norm} \\ &= \sum_{j=1}^{m} w_{norm,j}x_{norm,j} + b_{norm} \\ &= \sum_{j=1}^{m} w_{norm,j}\frac{x_j - mean(x_j)}{std(x_j)} + b_{norm} \\ &= \sum_{j=1}^{m} \frac{w_{norm,j}x_j}{std(x_j)} + b_{norm} - \sum_{j=1}^{m} \frac{w_{norm,j}mean(x_j)}{std(x_j)} \end{aligned} \tag{13}$$

Comparing the coefficients we get:

$$w_j = \frac{w_{norm,j}}{std(x_j)} \tag{14}$$

$$b = b_{norm} - \sum_{j=1}^{m} \frac{w_{norm,j}mean(x_j)}{std(x_j)} \tag{15}$$

Comparing the results

Scratch code values	SKLearn Values	Difference
-1.031682	-1.082384	0.050702
-2.703552	-2.732904	0.029352
-0.037852	-0.039598	0.001746
-0.318363	-0.327501	0.009138
-0.107537	-0.099200	-0.008337
0.002574	0.001924	0.000650
0.219966	0.223595	-0.003629
4.422252	4.615264	-0.193011

Table 1: Comparison

References

[1] *Logistic Regression*

[2] *Gradient Descent*