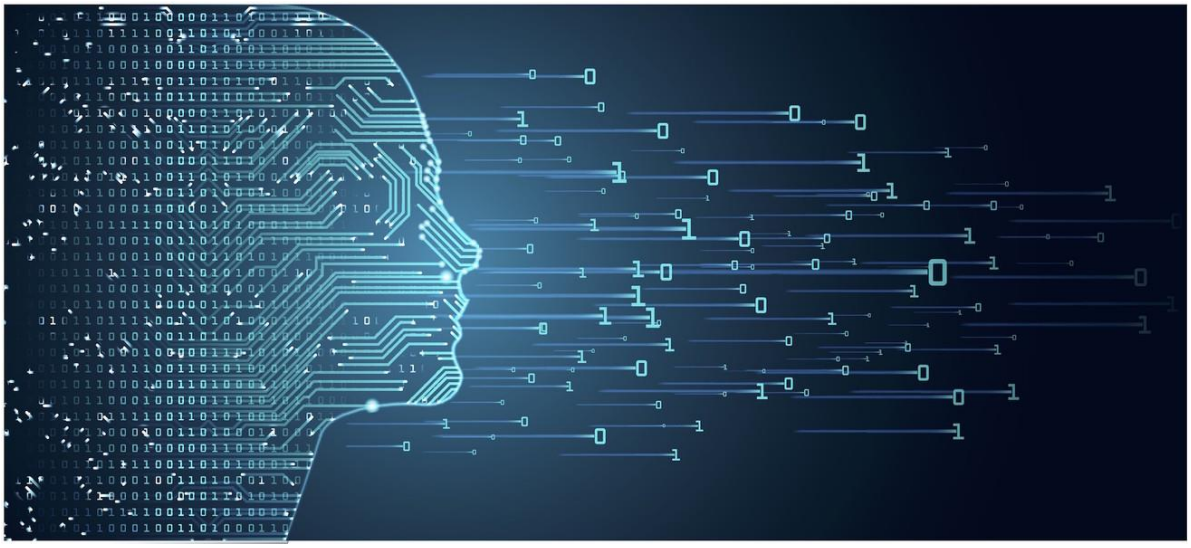


LinkedIn version : https://www.linkedin.com/pulse/linear-vs-logistic-regression-nicolas-hubert/

Linear vs. Logistic Regression

Nicolas Hubert

The title of this article suggests there is not the slightest reason to consider these techniques as similar. They are used for totally different purposes. But the occurrence of the word “regression” in these two expressions may be one of their few common characteristics. So, what is a Regression? Regression is a technique used to estimate/predict the value of a dependent variable, from one or more predictors, that are basically independent variables “explaining” how your value to predict is fluctuating. These independent variables are generally numeric, which is not necessarily the case of the variable you want to predict. I guess you see my point : if Linear and Logistic Regression differ, it is mainly due to the type of output variable. If Linear Regression is only about estimating continuous output variable, Logistic Regression is what is called a “classification” method, in the sense it is used for predicting the belonging of objects to a certain class, on the basis of explanatory quantitative and/or qualitative variables. Please note that there are various forms of regression such as linear, logistic, polynomial, Poisson, non-parametric, etc. This article will only revolve around the first two ones.

Linear Regression

Though it may seem somewhat dull compared to some of the fancier machine learning algorithms such as Support Vector Machine (SVM) or Naive Bayes, in reality Linear regression really is a useful and widely used statistical learning method. In addition, having a good knowledge of this tool prove to be a prerequisite before looking to more complex statistical learning algorithms. In fact, a lot of modern statistical learning algorithms strongly rely on Linear Regression, and some of them can even been

seen as generalizations or extensions of Linear Regression. In other words, mastering the fundamentals first !

a) Simple Linear Regression

Let me no longer keep it as a secret : there are two types of Linear Regression : Simple and Multiple Linear Regression. In this subsection, we will focus on the first one.

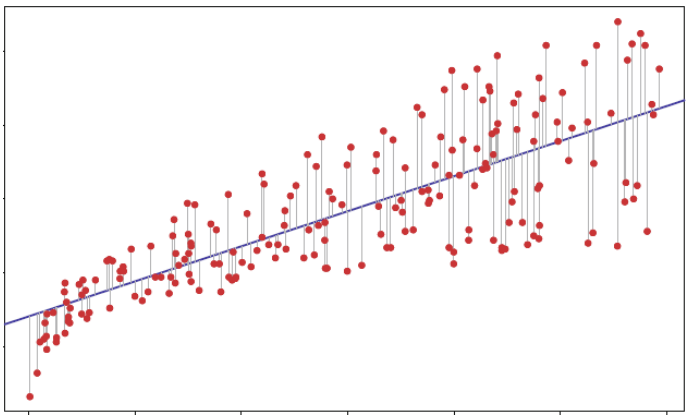
As you may have already guessed, Simple Linear Regression is used for predicting a continuous output variable Y on the basis of a *single predictor* variable X. This linear model works under the assumption that there is approximately a linear relationship between X and Y. Mathematically, this relationship would look like this :

$$Y \approx \beta_0 + \beta_1 X$$

Data Scientists often say they are "*regressing Y on X".*

In the above-mentioned formula, β0 and β1 are the model coefficients/parameters of our Linear Regression model. They respectively refer to the *intercept* term and *slope* term. Note that these two coefficients are unknown. So, before we can make any prediction, we must collect and use some training data to estimate these coefficients as precisely as possible. If we note (x1, y1), (x2, y2), . . . , (xn, yn) n observations, each one gives us both a measure of X (the input) and Y (the output). With a given number of data points n (as any as observations), our objective here is to infer what may be the approximate values of β0 and β1. These estimations are written as $\hat{\beta}_0$ and $\hat{\beta}_1$. But I want to draw your attention on the fact that these coefficients – as accurate as they might be – will only be approximations of β0 and β1. For sure, if you are lucky you may find the good ones, but in real-world data-oriented problems, better to play the lottery.

If we are able to draw a line that well fits our data points, then we have found good approximations of $\hat{\beta}_0$ and $\hat{\beta}_1$. One common way of measuring how well your estimation is using the least squares error method.



Least squares method. The red dots are the real observations. The vertical black lines show how much the real values differ from our predictions.

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

will be the prediction for Y based on the ith value of X.

$$e_i = y_i - \hat{y}_i$$

will be the residual.

We can then define the Residual Sum of Squares as :

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$$

All the challenge here is to find the values of $\hat{\beta}_0$ and $\hat{\beta}_1$ for which the RSS reach its minimum.

b) Multiple Linear Regression

In most real-world examples, we often consider a lot more than just one predictor. Sometimes, you will even find problems where you have more predictors than data sample (think of Bioinformatics), i.e you may collect data on 400 patients, but for each and every one of them, you consider more than 400 features (height, weight, age, etc). One possible way of dealing with multiple predictors would be to fit a simple linear regression model for each predictor. But we would lose a lot of information in the process : many features are correlated or are dependent. By estimating them separately, one could even take very badly informed decisions.

Then, a better idea is to extent the simple linear regression model we discussed above. By 'adding' new predictors to the formula and assigning them their own coefficient, our linear regression model would now look like this :

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

In the above-mentioned formula, there are some key points to understand :

- indices range from 0 to p because we basically assume here that we have p predictors to estimate;
- The intercept coefficient is always associated with the constant X0=1;
- Ɛ is an error term. You can just interpret it as some "noise" that would add some inaccuracy to our model. Although Ɛ has a very low value, it is really important and assuming some mathematical properties associated to Ɛ (ex : mean-zero random term, normally distributed, etc) will clearly help when it comes to properly define our model and make some predictions.

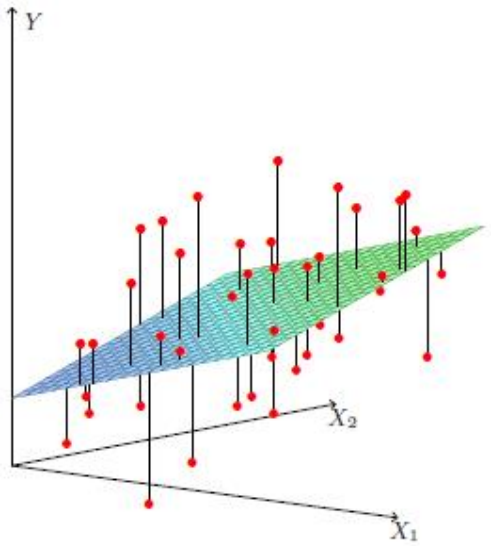
Talking about predictions, you may have already guessed what our formula for making predictions would be :

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$$

And then we choose our parameters the same way as we did with the simple linear regression :

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2 \end{aligned}$$

Remember what our least square distances looked like in our Simple Regression model ? Assume now that $X \in (\mathbb{R})^2$ (i.e we have two independent features). This would now look like this :



If you just feel curious about how Data Scientists use these tools in their everyday work, here are some questions they always consider when implementing a Linear Regression model :

- Do all the predictors help to explain Y, or can I just extract a subset of predictors that are relevant considering my goal ? This is commonly called “model selection”. All the difficulty here is determining which predictors you can drop without risk. Selecting the wrong predictors in your model can cause you to come to a flawed or distorted analysis;
- How well does my model fit the data ? This question is essential when it comes to make new predictions. Remember that you build your own model with a set of existing data points, but the ultimate purpose is obviously to make wise decisions based on your model;

Logistic Regression

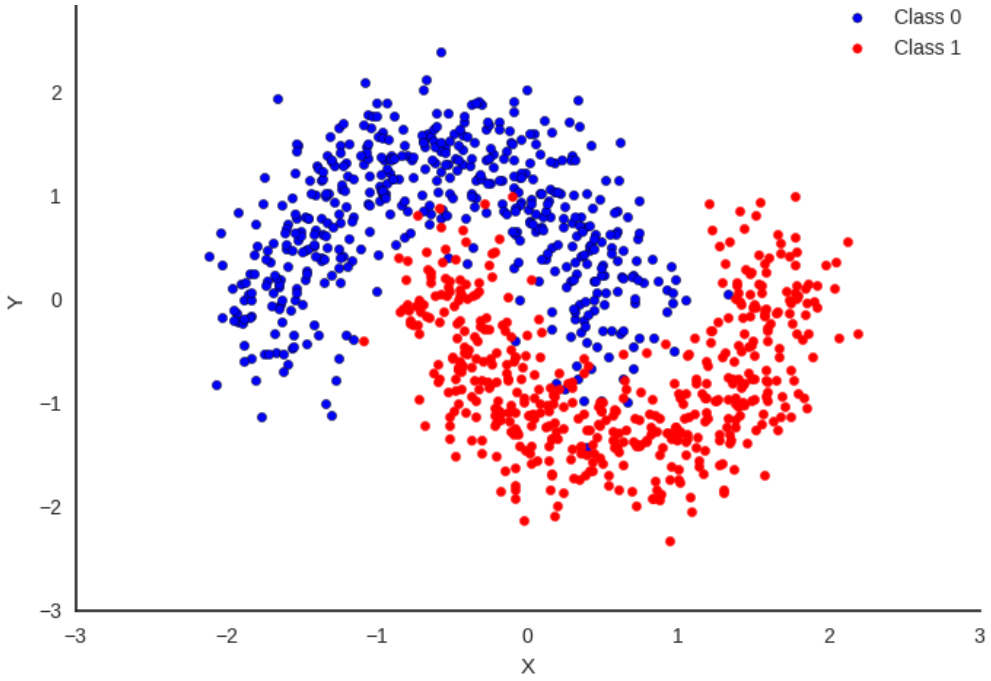
We previously introduced the Linear Regression model as a robust method for predicting the value of a continuous output. On the contrary, Logistic Regression is meant to be used in “Classification” problems. A lot of real-world data science projects involve Classification issues; the goal is to predict the belonging of objects to a certain class. Let’s discuss an example :

Imagine you are a biologist or physician, and you want to know whether the patients you help fight against cancer are likely to be permanently cured. So, you do not want to predict any continuous

value. Instead, you just want to know if Mr.Smith (your favorite patient) is definitely saved from disease or if his cancer may come back. A very intuitive and meaningful way to model this situation would be to define two classes : “Recurrent” and “Non-recurrent”. As with Multiple Linear Regression, you will certainly take several variables into account. That is not a problem. It is rather a sound choice as it will make your model more precise and robust. The only difference here is that the value of Y (your output variable which value you want to predict) can basically takes two possible values : either 0 or 1. Y is then a **categorical** variable. In fact, the right way to implement your model is to convert your character variable Y as a numerical variable. Let’s say that :

“N” → 0 (i.e. “N” will be replaced by the value 0)

“R” → 1 (i.e. “R” will be replaced by the value 1)



In the plot drawn above, one can imagine that initially, all our data points (accounting for our patients) were black. After using Logistic Regression, we defined some rules and came up with these two classes : one for each kind of patients (Recurrent vs. Non-recurrent). When new data points (i.e. data about new patients) are provided, we will want to know in which group classify them. Graphically speaking, it will really depend on where your data points will be located in this plot.

Now you can algorithmically work with this variable. We then expect a binary response for Y. It must be clear now that Linear Regression is not the right model for such a problem : even if you would be able to convert a continuous output into either 0 or 1 (for example, values between 0 and 0.5 would be automatically assigned the value 0, and *vice versa*) measuring how well your model is accurate using least squares regression would just make no sense.

In other words, what Logistic Regression does, is that rather than modeling Y

directly, it models the probability that Y belongs to a particular category. By the way, problems may be more complex than the one we discussed above : you can end up with more than two classes ! (as we will see later on in this article) If you want to classify hundreds of different animals, you may want

to know into what race your observations fall into. And thus, you may have a class for cats, another one for dogs, birds, etc.

But for the time being, let's say we just want to classify instances into two possible groups. To do so, we need to use some basic probability tools. More precisely, it will be required to convert every output into a range from 0 to 1. In Logistic Regression, the Logistic function is used (no kidding !) :

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

And the quantity introduced below is called the *odds*, and can take on any value odds between 0 and ∞.

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X}$$

Finally, we take the logarithm of this expression, and we end up with the *log-odds* or *logit* function below :

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

Once again, the coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ are unknown. We need to estimate them considering our training data set. Although we could use (non-linear) least squares to fit the model, which would be using the same tools as we saw for Linear Regression in a non-linear framework, Data Scientists prefer using the *maximum likelihood* method. The basic intuition behind is as follows: we try to find $\hat{\beta}_0$ and $\hat{\beta}_1$ such that plugging these estimates into the model for $p(X)$, we can classify as smoothly as possible, and without risk, our data points into one of the two groups. Taking our example on cancer patients, that means we want to set a rule for classifying, as we do not have too much doubt about whether a patient must be in group 1 or 2. We find this very basic idea in the *likelihood function* introduced below :

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'}))$$

The estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to *maximize* this likelihood function.

Just for you to know, as in Linear Regression we discussed earlier, there is a multiple version of Logistic Regression. If you have p different predictors, the formulas introduced above just need to be rewritten in this form :

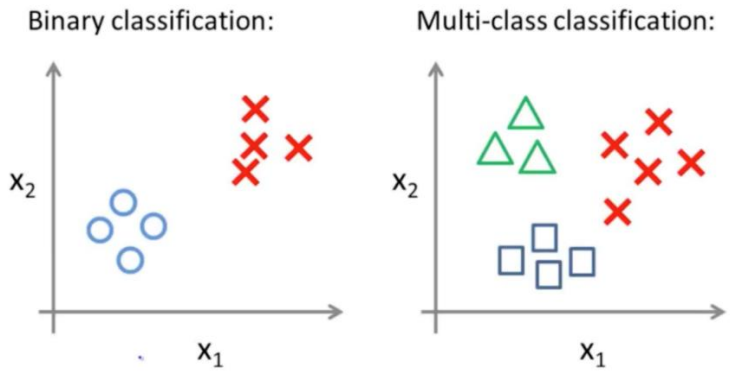
$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

And

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

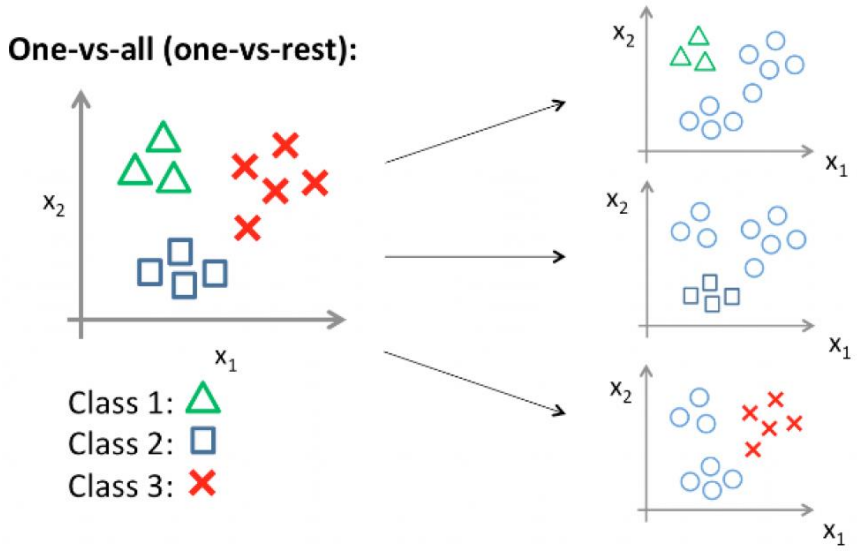
We basically use maximum likelihood method the same way as we would do with simple Logistic Regression, just with additional terms in our formula.

For the time being, we just had a grasp of Binary Logistic Regression. If you consider more than two groups, you must use Multinomial Logistic Regression. Graphically, the difference is obvious and just by taking a quick look at these plots, one might consider the problem is not the same at all :



Different methods may be implemented for Multinomial Logistic Regression. One of the most used is the *Softmax Regression*. Considering you have 4 groups, the *Softmax* function will output 4 probabilities related to the likelihood of belonging to each group. But we won't go in further detail for the *Softmax* Regression. There are loads of great articles and books explaining what the *Softmax Regression* is meant for, and it would be beyond the purpose of this article. We will rather navigate through another algorithm for Multinomial Logistic Regression : the *One-Versus-All* algorithm.

This algorithm aims at slicing the whole problem into smaller, binary ones. Taking a look at the plot below :



Here green triangles to be linked to class 1, blue squares to class 2 and red crosses to class 3, the One-Versus-All algorithm proceeds as follows :

- Step 1 : We consider that green triangles represent our “positive group” (i.e label “1”) et every other form belongs to the “negative class” (i.e label “0”). Logistic Regression is thus trained with this configuration, which will create a prediction function;
- Step 2 : The same method is repeated but this time, only blue squares are part of the positive class. Then, we end up with a second prediction function;
- Step 3 : This time, red crosses belong to the positive class and the rest is put into the negative class.

Finally, each one of the prediction functions will give us the probability that a certain data point *x* belongs to these three groups. The right way to classify x is to choose the class for which it obtained the greater probability.

Sources :

https://www.listendata.com/2014/11/difference-between-linear-regression.html

https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

An Introduction to Statistical Learning with Applications in R - G.James

Elements of Statistical Learning – Hastie, Tibshirani, Friedman