Logistic Regression

Logistic regression is the primary algorithm used for classification. Linear regression and Logistic regression share certain things; however the biggest difference is what they are used for. Linear regression algorithms are used to predict/forecast real values, but logistic regression is used for classification tasks in two or more categories/classes.

NotePlease do not be confused by the word 'regression' in the name. This is mainly just a convention Logistic regression (classification) is not regression at all.

In order to understand logistic regression, we need to understand the logistic function that it is based on.

The Logistic Function

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. We need the output of the algorithm to be class variable, i.e. 0 representing NO, 1 representing YES. Therefore, we are squashing the output of the linear equation into a range of [0,1]. To squash the predicted value between 0 and 1. To achieve this We use the non-linear sigmoid function visualized below, which maps predictions made by the algorithm to probabilities for the output classes.

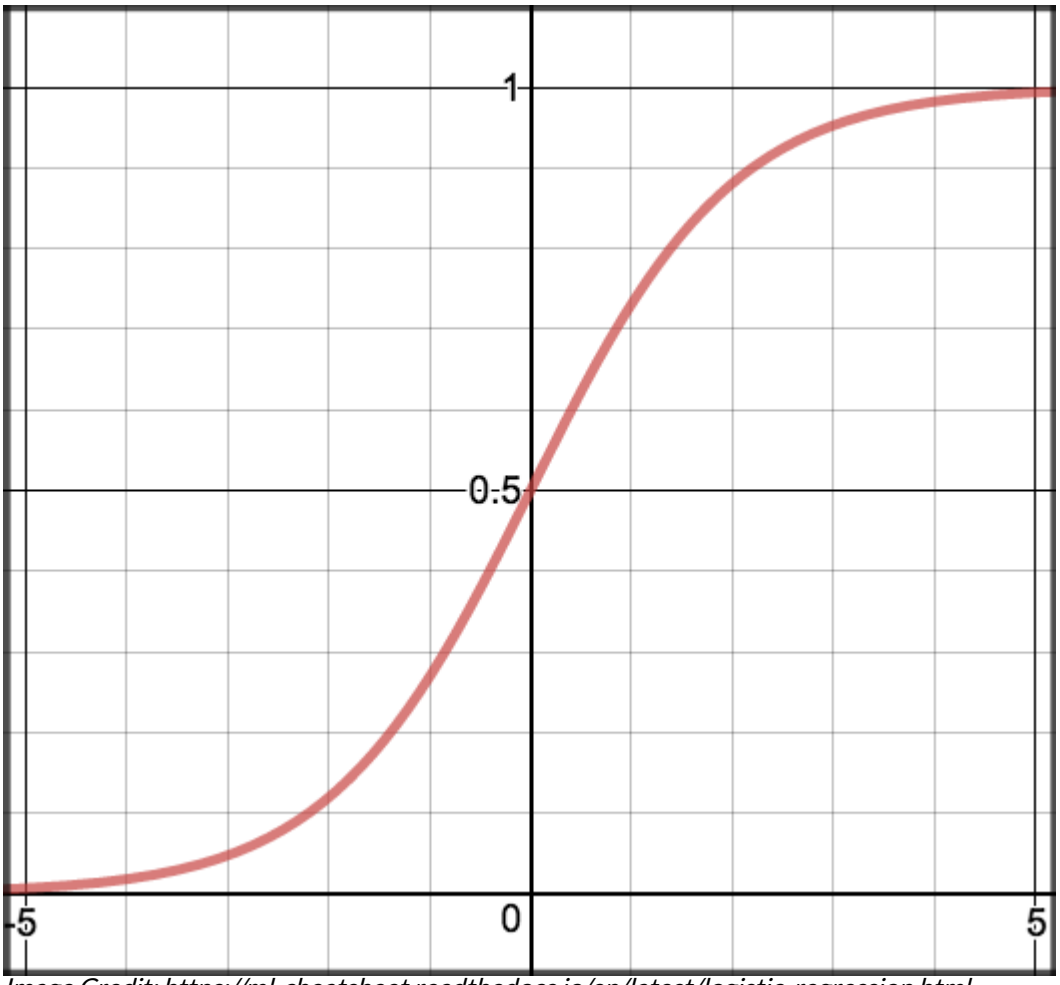

Image Credit: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

Apart from Sigmoid function, it is also referred to as logistic function and logit function.

We take the output(z) of the linear equation:

$$z = \theta_0 + \theta_1 \cdot x_1 + \theta \cdot x_2 + \cdots$$

and give to the function as z:

$$S(z) = \frac{1}{1 + e^{-z}}$$

Such as:

$$\frac{1}{1 + e^{-(\theta_0 + \theta_1 * x_1 + \theta_2 * x_2)}}$$

This returns a squashed value h or Sigmoid of z – S(z), the value h will lie in the range of 0 to 1.

The model itself simply models probability of output in terms of input and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class and below the cutoff as the other.

Our current prediction function returns a probability score between 0 and 1. In order to map this to a discrete class (true/false, cat/dog), we select a threshold value or tipping point above which we will classify values into class 1 and below which we classify values into class 2:

$$p \geq 0.5, class = 1$$
$$p < 0.5, class = 0$$

For example, if our threshold was .5 and our prediction function returned .6, we would classify this observation as positive. If our prediction was .4 we would classify the observation as negative:

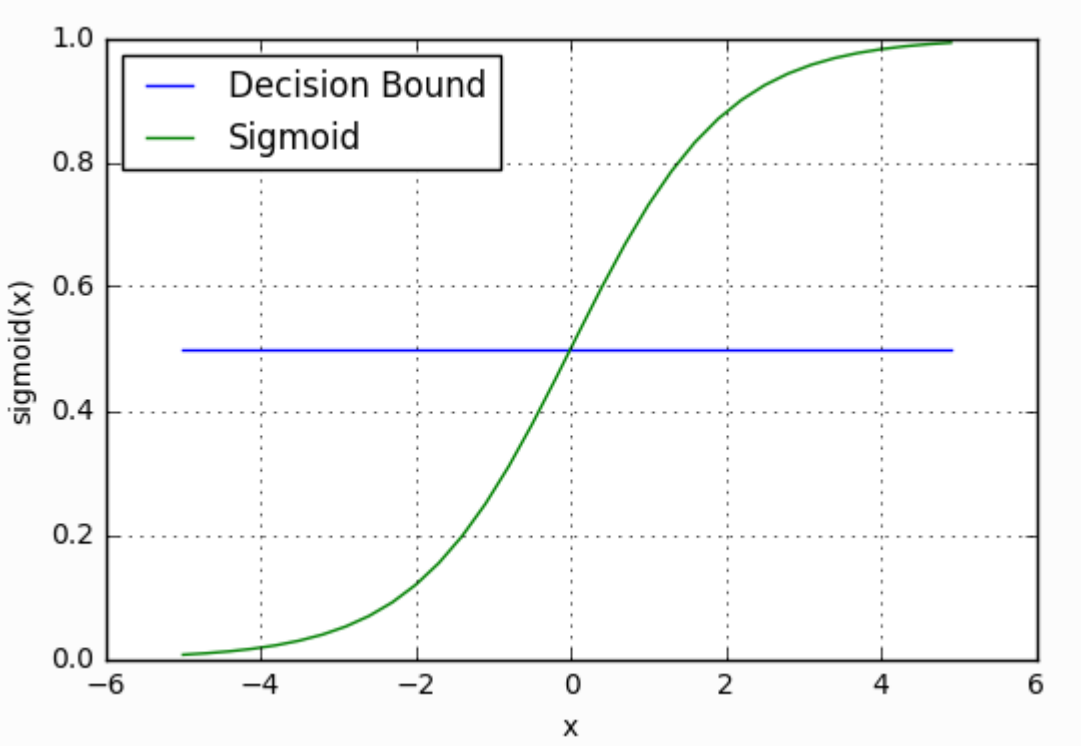

Image Credit: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

Although we are mainly interested in getting a predicted class out of our model, we can use the probability obtained as a confidence score for a class being predicted.

So far so good, however when it comes to cost function, we cannot use the same cost function used in linear regression algorithm as we are trying to predict classes. For this reason, we use a logarithmic loss function (also known as Cross-entropy) to calculate the cost for misclassifying.

Cross-entropy loss can be divided into two separate cost functions: one for y=1 and one for y=0:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if y} = 1$$
$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if y} = 0$$

To minimize our cost, Gradient Descent is a popular choice just like before in Linear Regression.

Regularization in Logistic Regression

Regularization is extremely important in logistic regression modeling as without it Logistic regression is prone to overfitting similar to Linear Regression.

Also without regularization, the asymptotic (limiting, approaching towards zero) nature of logistic regression would keep driving loss towards 0 in when there are large number of features. Consequently, most logistic regression models use one of the following two strategies to dampen model complexity:

- L2 regularization. (Already discussed in Linear regression)

- Early stopping, that is, limiting the number of training steps or the learning rate.

Conclusion

The reason Logistic Regression is widely used despite availability of the state of the art algorithms such as deep neural networks is because logistic regression is very efficient and does not require too much computational resources which makes it suitable for production environments.

Model Assumptions

- Data is free of missing values
- The target variable is binary, in other words it only accepts two values, or it could be ordinal, a categorical variable with ordered values.
- All predictors are independent of each other
- That there are at least 50 observations per predictor variable to ensure reliable results

Watch the following video for more information on logistic regression:

Direct Video Link: Logistic Regression (https://www.lynda.com/Python-tutorials/Logistic-regression-model/520233/601977-4.html?org=sheridancollege.ca)