3. Logistic Regression

Linear vs Logistic Regression

- Linear Regression is well suited, when the response variable y is CONTINUOUS.
- What about when y is discrete ?
  Example : y is binary. (ie) $y \in \{0, 1\}$
  This is precisely handled by LOGISTIC REGRESSION.
  For instance, let us consider
  (i) Image / Video : Person is present or absent
  (ii) Medical imaging : Disease is present or absent

Logistic function

- The Logistic function is given below.

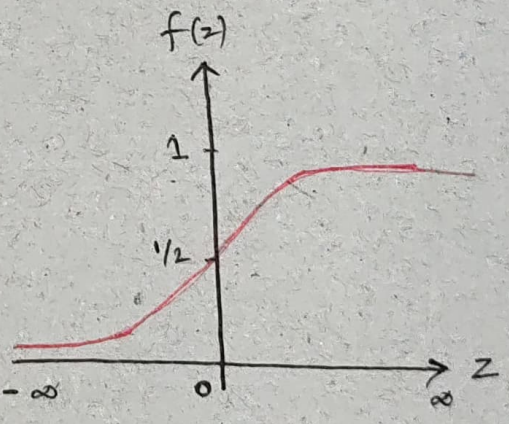
$$f(z) = \frac{1}{1+e^{-z}}$$

- This is also referred to as the SIGMOID Function
- The Logistic function is used in the Logistic Regression.
- Observe :

$$\lim_{z \to \infty} \left(\frac{1}{1+e^{-z}} \right) \to 1$$

$$\lim_{z \to -\infty} \left(\frac{1}{1+e^{-z}} \right) \to 0$$

$$z = 0, \quad f(z) = \frac{1}{1+1} = \frac{1}{2}$$



Plot of Logistic Function

$e^{\infty} = \infty$

$e^{-\infty} = \frac{1}{e^{\infty}} = \frac{1}{\infty} = 0$

Probability

⊙ In Logistic Regression, the Probabilities are given as

$$P(y=1|\bar{x}) = \frac{1}{1+e^{-\bar{x}^T\bar{h}}} = g(\bar{x})$$

where, $y \rightarrow$ Response variable (0/1)

$\bar{h} \rightarrow$ Regression coefficient vector, which is unknown, to be determined from the training data

$\bar{x} \rightarrow$ Regressors

$$P(y=0|\bar{x}) = 1 - P(y=1|\bar{x})$$

$$= 1 - \frac{1}{1+e^{-\bar{x}^T\bar{h}}}$$

$$= \frac{1+e^{-\bar{x}^T\bar{h}} - 1}{1+e^{-\bar{x}^T\bar{h}}}$$

$$= \frac{e^{-\bar{x}^T\bar{h}}}{1+e^{-\bar{x}^T\bar{h}}} = 1 - g(\bar{x})$$

Thus, we are modeling the Probability for the Response variable $y = 0/1$, for the given Regressors $\bar{x}$.

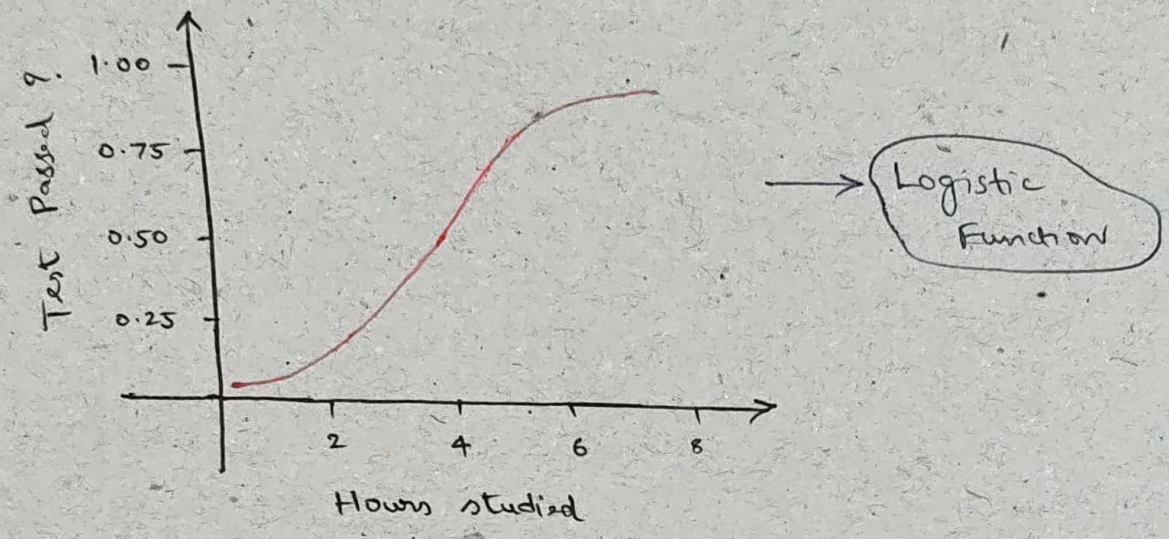
$$P(y=1|\bar{x}) = \frac{1}{1+e^{-\bar{x}^T\bar{h}}} = g(x)$$

$$P(y=0|\bar{x}) = \frac{e^{-\bar{x}^T\bar{h}}}{1+e^{-\bar{x}^T\bar{h}}} = 1 - g(x)$$

Example:

$P(\text{Result} \mid \text{Hours studied})$

- Hours studied → Regressor, x
- Result → Binary / Discrete Response, y
  - $y = 1$ → PASS
  - $y = 0$ → FAIL



How to determine the Regression parameter $\bar{\beta}$ from the training data in this case?

Note that, the technique which requires training data is called SUPERVISED Learning.

The particular algorithm that we use in this case is known as Maximum Likelihood algorithm.

"Given the training data $(y, \bar{x})$, what is the Regression coefficient vector $\bar{\beta}$ that has the Maximum Likelihood"

The Maximum likelihood is one of the techniques to efficiently estimate or determine unknown parameter in many applications / scenarios.

Wkt, $P\left(y(k)=1 \mid \bar{x}(k)\right) = g\left(\bar{x}(k)\right)$

$$P\left(y(k)=0 \mid \bar{x}(k)\right) = 1 - g\left(\bar{x}(k)\right)$$

Thus, the Likelihood of $\left(y(k), \bar{x}(k)\right)$ (Respons, Regressor; Training pair) can be written as

$$P\left(y(k) \mid \bar{x}(k)\right) = \left[g\left(\bar{x}(k)\right)\right]^{y(k)} \cdot \left[1 - g\left(\bar{x}(k)\right)\right]^{1-y(k)}$$

The Joint Likelihood of all outputs/Responses, considering all the outputs/Responses are Independent is given as

$$L(\bar{h}) = \prod_{k=1}^{M} P\left(y(k) \mid \bar{x}(k)\right)$$

$\Sigma \rightarrow$ Summation

$\prod \rightarrow$ Product

$$= \prod_{k=1}^{M} \left[g\left(\bar{x}(k)\right)\right]^{y(k)} \cdot \left[1 - g\left(\bar{x}(k)\right)\right]^{1-y(k)}$$

where, $M \rightarrow$ # Training points.

The Log Likelihood is given as

$$\ln L(\bar{h}) = l(\bar{h})$$

$$l(\bar{h}) = \ln \prod_{k=1}^{M} \left[g\left(\bar{x}(k)\right)\right]^{y(k)} \cdot \left[1 - g\left(\bar{x}(k)\right)\right]^{1-y(k)}$$

$$= \sum_{k=1}^{M} y(k) \cdot \ln\left[g\left(\bar{x}(k)\right)\right] + \left(1-y(k)\right) \cdot \ln\left[1 - g\left(\bar{x}(k)\right)\right]$$

To maximize the log-likelihood $l(\bar{h})$, one can employ the gradient ascent technique.

$$\bar{h}(k+1) = \bar{h}(k) + \frac{\eta}{2} \nabla l(\bar{h})$$

Gradient of Log Likelihood

Recall, the Gradient gives the direction of the greatest increase. So, we step in the direction of the Gradient (ii) We adjust/update the Regression parameter vector $\bar{h}$ in the direction of the gradient, because we want to maximize the Log Likelihood.

So, we take the Gradient of log likelihood, and we move in the direction of the gradient. This is basically the Gradient ascent technique.

Thus, the update rule reduces to

$$\bar{h}(k+1) = \bar{h}(k) + \eta\, e(k+1)\cdot \bar{x}(k+1)$$

which is very similar to the LMS Rule.

$$\Rightarrow \bar{h}(k+1) = \bar{h}(k) + \eta \left[y(k+1) - g\left(\bar{x}(k+1)\right) \right] \bar{x}(k+1)$$

where,

$$\underbrace{e(k+1)}_{\text{Error}} = \underbrace{y(k+1)}_{\text{Observed Response}} - \underbrace{g\left(\bar{x}(k+1)\right)}_{\text{Predicted Response}}$$

Thus,

$$\bar{h}(k+1) = \bar{h}(k) + \eta \left[y(k+1) - g\left(\bar{x}(k+1)\right) \right] \bar{x}(k+1)$$

is the Logistic Regression update rule, which is identical to LMS Rule (AKA. Adaptive filter / Online Learning)

We have, the update Rule as below.

$$\bar{h}(k+1) = \bar{h}(k) + \eta\, e(k+1)\, \bar{x}(k+1)$$

This is identical to LMS Rule, which means the complexity is very low and this is an Online algorithm, so we can keep updating this, as and when the data are arriving (ie) we can keep running this rule sample by sample rather than batch wise or block wise.

Now, if we look at the Log-likelihood expression,

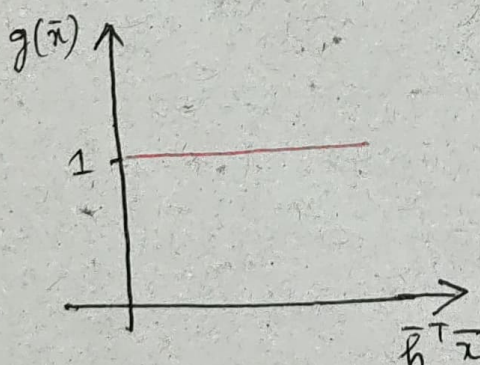
$$l(\bar{h}) = \sum_{k=1}^{N} y(k) \cdot \ln\left[g(\bar{x}(k))\right] + (1-y(k)) \cdot \ln\left[1-g(\bar{x}(k))\right]$$

this is a very complicated function. If we try to maximize this directly, there is no closed form solution, unlike the Least Squares (LS). But interestingly, this can be shown to be convex.

Since, direct maximization is challenging, we have derived a Stochastic gradient ascent algorithm to maximize the Logic.

Perceptron Learning Algorithm.

The Perceptron Learning Algorithm basically replaces the Logistic function by Unit Step function.

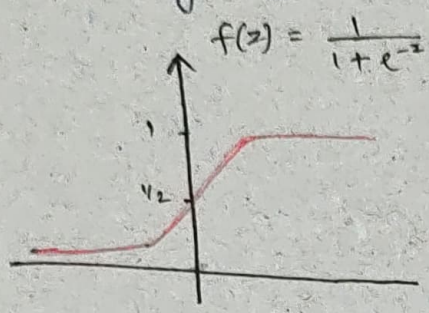


$$g(\bar{x}) = \begin{cases} 1, & \text{if } \bar{h}^T\bar{x} \geq 0 \\ 0, & \text{if } \bar{h}^T\bar{x} < 0 \end{cases}$$

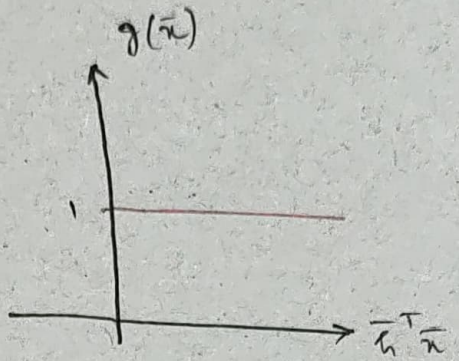
where,

$g \rightarrow$ Threshold function.

If we compare the Logistic function (Sigmoid) with the Unit step function, we see that the Sigmoid is Smooth (ie.) infinitely differentiable.



LOGISTIC FUNCTION

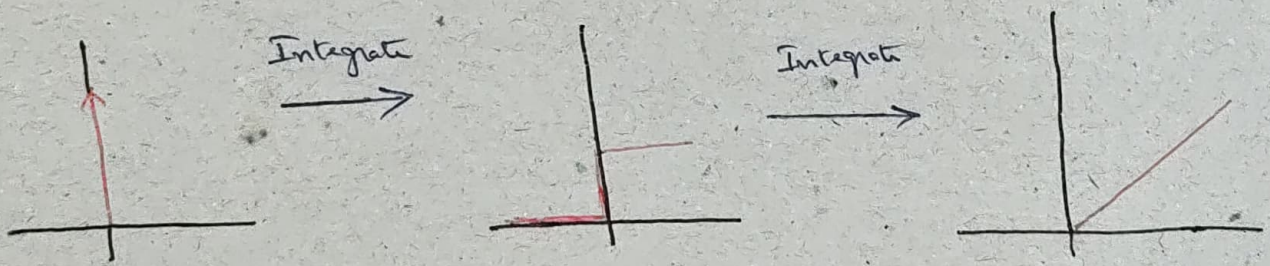


UNIT STEP FUNCTION

Therefore, we can take gradient ascent and move in the direction of it.

But the problem in case of Unit step function is that it is not differentiable at zero. (ii) we'll get impulse. (ii) NOT Smooth. Thus, we cannot derive the gradient ascent in a conventional sense. But we can use other interesting principles and optimizations known as Sub-Gradients and so on.

Note:



This fact is used in Deep Learning, which is known as ReLU activation function (Rectified Linear Unit). It is used in inner layers of the Neural Networks.

WKT, Unit Step function is not differentiable. So we cannot derive the update rule using the conventional gradient ascent, because one cannot take the gradient as it is not differentiable at 0. But, what is interesting is, one can derive the update rule again which is exactly similar.

The update rule once again is given as

$$\bar{h}(k+1) = \bar{h}(k) + \eta\, e(k+1)\, \bar{x}(k+1)$$

where,

$$e(k+1) = y(k+1) - g\left(\bar{x}(k+1)\right)$$

Thus, once again, we get the same update rule as LMS.

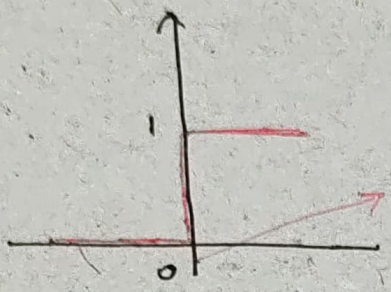
Brief Theory.

In Machine Learning theory, a broad set of cost functions will have the same update rule. For instance, we have an exponential structure (Gaussian) in the Least Squares / Maximum Likelihood. Here also, we have an exponential structure (Sigmoid). So, broadly speaking, when we have such an exponential involved, the update rule reduces similar to the LMS algorithm.

This is the Learning rule for the Logistic regression and the algorithm of replacing the logistic function by Unit step function, which in turn reduces to LMS learning rule, is termed as Perceptron Learning Algorithm.

The Perceptron Learning Algorithm is the first model for the Human Brain, where the activation function is Unit Step function.

$$g(\bar{x}) = \begin{cases} 1, & \bar{w}^T \bar{x} \geq 0 \\ 0, & \bar{w}^T \bar{x} < 0 \end{cases}$$



UNIT STEP FUNCTION

This Unit step function is the Activation function of a neuron. (ie) This is how the neuron fires, when the stimulus exceeds a threshold. This is also called as Neuron Firing Model.

If the threshold is < 0, Neuron is inactive.

If the threshold is ≥ 0, Neuron Firing

Like Sigmoid, this is also highly non-linear. Interestingly, the ReLU (Ramp function) is also non-linear though it seems like linear (coz, after zero only, it is linearly increasing, until zero, its not linear)

ReLU is also a non-linear model, which is why it is used in the inner layers of a Deep learning neural network. Thus, Unit Step / ReLU both are non-linear activation function, and was developed as an initial model for a Neuron in the Human Brain.

So, essentially, this is termed as Perceptron Learning Algorithm and this is a Learning Model for neurons, to model the Learning capabilities of the neurons in the Human brain.

And, for a broad set of these models / activation functions, the learning rule reduces to the LMS update rule, which is very advantageous coz its an Online algorithm which keeps updating / Learning continuously and the complexity / computational cost is extremely low. So, it can be employed in any practical systems with great ease.