संगणक विज्ञान एवं अभियांत्रिकी विभाग

Department of Computer Science & Engineering

National Institute of Technology Sikkim, Ravangla, South Sikkim-737139

(AN INSTITUTE OF NATIONAL IMPORTANCE, MINISTRY OF EDUCATION (MoE), GOVT. OF INDIA)

Machine Learning (CS22144/CS16104) **B.Tech, 6^{th} Semester**

Logistic Regression

Now suppose the dependent (target) variable *y* is **Categorical.** Therefore, it may take on either two values "Success" (1) or "Failure" (0) or three or more values like Veg, Non-veg, Vegan, etc. We are interested in predicting a *y* from a continuous dependent variable *x*. This is the situation in which **Logistic Regression** is used. There are three types of Logistic Regression as follows.

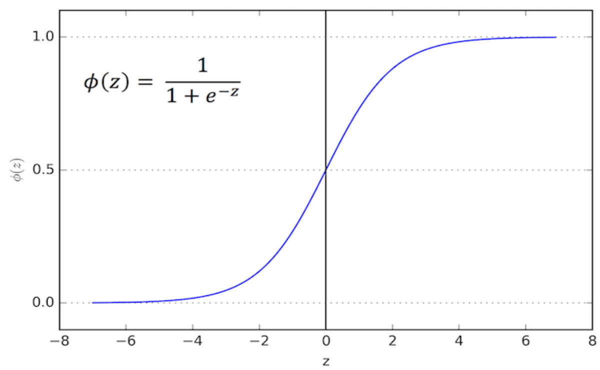
- ***Binary Logistic Regression***: the categorical response has only two 2 possible outcomes, e.g., Veg or Non-veg
- ***Multinomial Logistic Regression***: three or more categories without ordering, e.g., predicting which food is preferred more (Veg, Non-Veg, Vegan).
- ***Ordinal Logistic Regression***: three or more categories with ordering, e.g., movie rating from 1 to 5.

The goal of *binary logistic regression* is to train a classifier that can make a binary decision about the class of a new input observation. Here we introduce the *sigmoid classifier* that will help us make this decision.

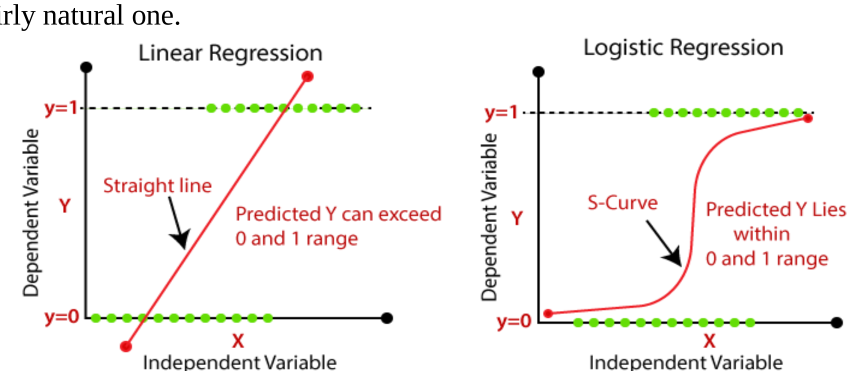
The Logistic (or logit) Function

The ***sigmoid function*** is also called the ***logistic*** **(*or logit*)** ***function***, has the following equation.

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



The *sigmoid* has a number of advantages; it takes a real-valued number and maps it into the range [0,1]. Notice that $\sigma(z)$ tends towards 1 as $z \to \infty$, and $\sigma(z)$ tends towards 0 as $z \to -\infty$. Other functions that smoothly increase from 0 to 1 can also be used, but for a couple of reasons, the choice of the logistic function is a fairly natural one.



If we apply the sigmoid to the sum of the weighted features, we get a number between 0 and 1. Here, let us assume $z = b_0 + b_1x$. The logistic regression predicts probabilities, rather than just classes. We can fit it using likelihood. To make it a probability, we just need to make sure that the two cases, $P(y = 1)$ and $P(y = 0)$, sum to 1. We can do this as follows.

$$\begin{aligned} P(y=1\,|\,x) &= \sigma(z) \\ &= \sigma(b_0 + b_1x) \\ &= \frac{1}{1+e^{-(b_0+b_1x)}} \end{aligned}$$

$$\begin{aligned} P(y=0\,|\,x) &= 1-\sigma(z) \\ &= 1-\sigma(b_0 + b_1x) \\ &= \frac{e^{-(b_0+b_1x)}}{1+e^{-(b_0+b_1x)}} \end{aligned}$$

Now we have an algorithm that given an instance x_i computes the probability $P(y_i = 1|\, x_i)$. How do we make a decision? For a test instance x_i, we say yes if the probability $P(y_i = 1|x_i)$ is more than 0.5, and no otherwise. We call 0.5 the decision boundary:

$$y_i = \begin{cases} 1 & \text{If } P(y_i = 1|X_i) > 0.5 \\ 0 & \text{Otherwise.} \end{cases}$$

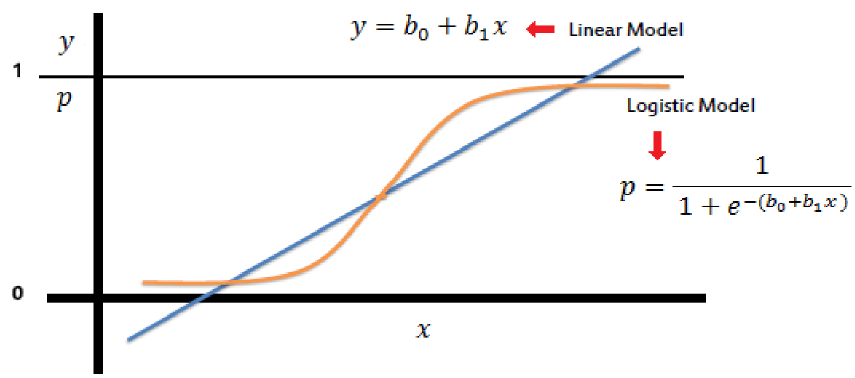
Remark: *Before moving on, here is a useful property of the derivative of the sigmoid function*:

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) = \frac{e^{-z}}{(1+e^{-z})^2} = \left(\frac{1}{1+e^{-z}}\right)\left(1-\frac{1}{1+e^{-z}}\right) = \sigma(z)(1-\sigma(z))$$

Linear vs. Logistic Regression

A linear regression is not appropriate for predicting the value of a categorical variable as a linear regression will predict values outside the acceptable range (e.g., predicting probabilities outside the range 0 to 1).

Binary logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). A logistic regression produces a logistic curve, which is limited to values between 0 and 1.



Learning in Logistic Regression

How the parameters of the model, i.e., the weights $B = [b_0, b_1, b_2, \ldots, b_n]^T$ are learned? Logistic regression is an instance of supervised classification in which we know the correct label y_i (either 0 or 1) for each observation x_i.

- First, we will introduce the loss function that is commonly used for logistic regression, the ***cross-entropy loss***.
- The second thing we need is an optimization algorithm for iteratively updating the weights so as to minimize this loss function. The standard algorithm for this is ***gradient descent***.

The Cross-entropy Loss Function

Let, $z = H(X, B) = b_0+b_1x_1+b_2x_2+ \ldots + b_nx_n$. We need a loss function that expresses, how close the classifier output ($\sigma(z)$) is to the correct output (y, which is 0 or 1) for an observation X_i. Here, the loss function is the negative log likelihood loss, generally called the **cross-entropy** loss.

$$L(H(X_i,B),y_i) = \begin{cases} -\log(P(y_i=1|X_i)) & \text{If } y_i = 1 \\ -\log(1-P(y_i=1|X_i)) & \text{If } y_i = 0 \end{cases}$$

$$= \begin{cases} -\log(\sigma(H(X_i,B))) & \text{If } y_i = 1 \\ -\log(1-\sigma(H(X_i,B))) & \text{If } y_i = 0 \end{cases}$$

From the above equation, it can be observed that,

- If $y_i = 1$ and $P(y_i = 1|X_i) = \sigma(H(X_i,B)) = 1$, $L(H(X_i,B)) = 0$; but for $P(y_i = 1|X_i) \rightarrow 0$, $L(H(X_i,B)) \rightarrow \infty$.
- If $y_i = 0$ and $P(y_i = 1|X_i) = \sigma(H(X_i,B)) = 0$, $L(H(X_i,B)) = 0$; but for $P(y_i = 1|X_i) \rightarrow 1$, $L(H(X_i,B)) \rightarrow \infty$.

This corresponds to intuition: if prediction is $P(y_i = 1|X_i) = 0$ but actual value was $y_i = 1$, learning algorithm will be penalized by large cost. As we always have $y_i = 0$ or $y_i = 1$, we can simplify the cost function definition to,

$$L(H(X_i,B),y_i) = -y_i\log(P(y_i=1|X_i)) - (1-y_i)\log(1-P(y_i=1|X_i))$$

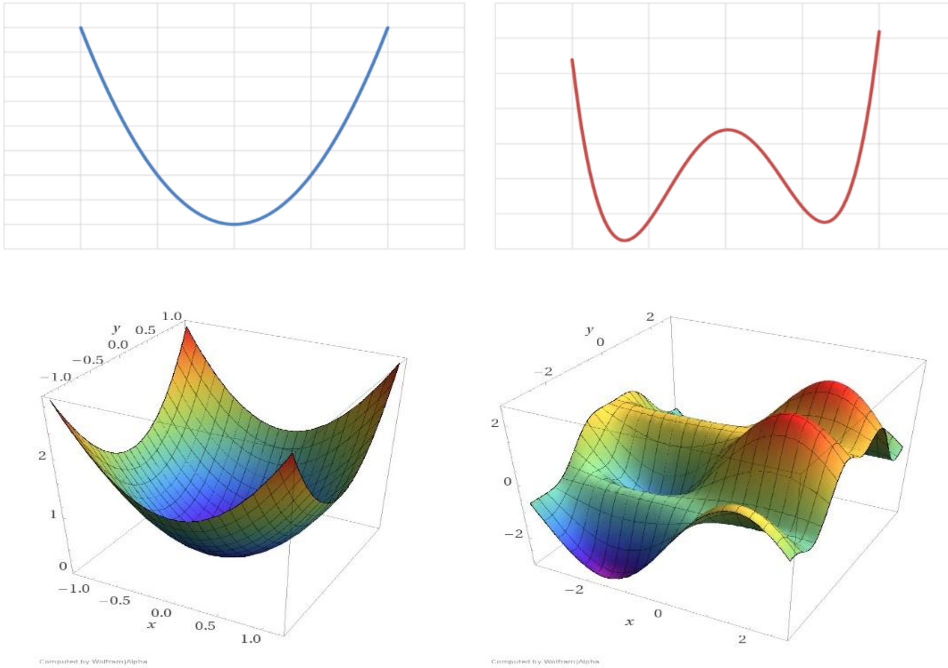
$$= -y_i\log(\sigma(H(X_i,B))) - (1-y_i)\log(1-\sigma(H(X_i,B)))$$

Multiplying by y_i and $(1- y_i)$ in the above equation is a sneaky trick that let us use the same equation to solve for both $y_i = 1$ and $y_i = 0$ cases. If $y_i = 0$, the first side cancels out. If $y_i = 1$, the second side cancels out. In both cases we only perform the operation we need to perform. Therefore, the overall cost function can be derived as follow:

$$J(B) = \frac{1}{m}\sum_{i=1}^{m} L(H(X_i,B),y_i) = \frac{1}{m}\sum_{i=1}^{m}\{-y_i\log(P(y_i=1|X_i)) - (1-y_i)\log(1-P(y_i=1|X_i))\}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\{-y_i\log(\sigma(H(X_i,B))) - (1-y_i)\log(1-\sigma(H(X_i,B)))\}$$

Remark: *Why does the cost function which has been used for linear regression cannot be used for logistic regression?* Linear regression uses mean squared error as its cost function. If this is used for logistic regression, then it will be a non-convex function of the parameters (b). *Gradient descent* will converge into global minimum only if the function is convex. Therefore, a new cost function that is suitable for the gradient descent is required. For other optimization techniques, any other suitable cost function can be used.



Covex function **Non-convex function**

Gradient Descent Algorithm

The ***gradient descent*** algorithm, which starts with some initial b_j, and repeatedly performs the following update:

$$b_j = b_j - \eta \frac{\partial J(B)}{\partial b_j}$$

This update is simultaneously performed for all values of $j = 1, \ldots, n$. Here, η is called the *learning rate*. In order to implement this algorithm, we have to work out what is the partial derivative term on the right-hand side. Let us first work it out.

Here, $z = H(X, B) = b_0+b_1x_1+b_2x_2+ \ldots + b_nx_n$. In order to update the parameters $B = [b_0, b_1, b_2, \ldots, b_n]^T$, we need a definition for the gradient. The derivative of this function based on b_1 for the overall training data X_i, $i = 1,2,\ldots, m$ is,

$$\frac{\partial J(B)}{\partial b_1} = \frac{\partial J(B)}{\partial u} \times \frac{\partial u}{\partial z} \times \frac{\partial z}{\partial b_1}$$

where, $u = P(y_i = 1|X_i) = \sigma(H(X_i,B))$ and $z = H(X_i,B) = b_0+b_1x_1+b_2x_2+ \ldots + b_nx_n$. The first part can be derived as follows.

$$\frac{\partial J(B)}{\partial u} = \frac{\partial}{\partial u}\frac{1}{m}\sum_{i=1}^{m}\{-y_i \log\, u - (1-y_i)\log(1-u)\}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left\{-y_i\frac{1}{u} - (-1)\frac{(1-y_i)}{(1-u)}\right\} = \frac{1}{m}\sum_{i=1}^{m}\left\{-\frac{y_i}{u} + \frac{(1-y_i)}{(1-u)}\right\}$$

The second and third parts are as follows.

$\frac{\partial u}{\partial z} = u(1-u)$ and $\frac{\partial z}{\partial b_1} = x_1$

Therefore,

$$\frac{\partial J(B)}{\partial b_1} = \frac{1}{m}\sum_{i=1}^{m}\left\{-\frac{y_i}{u} + \frac{(1-y_i)}{(1-u)}\right\} \times u(1-u) \times x_1$$

$$= \frac{1}{m}\sum_{i=1}^{m}\{(u-y_i) \times x_1\}$$

Hence, the b_1 can be updated as follows.

$$b_1 = b_1 - \eta\frac{\partial J(B)}{\partial b_1}$$

$$= b_1 - \eta\frac{1}{m}\sum_{i=1}^{m}\{(u-y_i) \times x_1\}$$

Similarly, all parameters can be updated as follows.

$$b_j = b_j - \eta\frac{\partial J(B)}{\partial b_j}, \forall j \in \{0, 1, 2, ..., n\}$$

(A) Batch Gradient Descent Algorithm: With the identical algorithm as in linear regression, the batch gradient descent algorithm is as follow.

Repeat until convergence
{

$$\boldsymbol{b_j = b_j - \eta\frac{1}{m}\sum_{i=1}^{m}\{(\sigma(H(X_i, B)) - y_i) \times x_j\}, \forall j \in \{0, 1, 2, \dots, n\}}$$

}

This method looks at every example in the entire training set on every step.

(B) Stochastic Gradient Descent Algorithm: With the identical algorithm as in linear regression, the stochastic gradient descent algorithm is as follow.

Repeat until convergence
{
for (_i_=1 to _m_)
{

$$\boldsymbol{b_j = b_j - \eta\{(\sigma(H(X_i, B)) - y_i) \times x_j\}, \forall j \in \{0, 1, 2, \dots, n\}}$$

}
}

In this algorithm, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single training example only.

Odds and log odds

Logistic regression works with **odds** rather than proportions. The odds are the ratio of the proportions for the two possible outcomes. If p is the probability of a success, then 1 - p is the probability of a failure, and

$$\text{Odds} = \frac{P}{1-P} = \frac{\text{probability of success}}{\text{probability of failure}}$$

The quantity $\ln(\frac{P}{1-P})$ is called the log odds.

Example: odds ratio, log odds ratio

Suppose a dice is rolled:
Success = "roll a six", P = 1/6

The **odds ratio** $\frac{P}{1-P} = \frac{\frac{1}{6}}{1-\frac{1}{6}} = \frac{\frac{1}{6}}{\frac{5}{6}} = \frac{1}{5}$

The **log odds ratio** $\ln\left(\frac{P}{1-P}\right) = \ln(0.2) = -1.69044$

The **log odds ratio** is linearly related to x, i.e.,

$$\text{In}\left(\frac{P}{1-P}\right) = b_0 + b_1 x$$

$$\frac{P}{1-P} = e^{b_0+b_1 x}$$

$$Or,\ P = \frac{e^{b_0+b_1 x}}{1+e^{b_0+b_1 x}} = \frac{1}{1+e^{-(b_0+b_1 x)}}$$

***Exercise* 1:** A study was designed to compare two energy drink commercials. Each participant was shown the commercials, *A* and *B*, in random order and asked to select the better one. There were 150 women and 140 men who participated in the study. Commercial *A* was selected by 71 women and by 87 men. Find the odds of selecting Commercial *A* for the men. Do the same for the women.

***Exercise* 2:** Refer to the previous exercise. Find the odds of selecting Commercial *B* for the men. Do the same for the women. Find the log odds for the men and the log odds for the women choosing Commercial *A* and *B*.

Q. *Why the Mean-Squared-Error evaluation metric cannot be applied in case of logistic regression*?
Ans: Logistic regression is a classification algorithm so it's output cannot be real time value so mean squared error cannot be used for evaluating it.