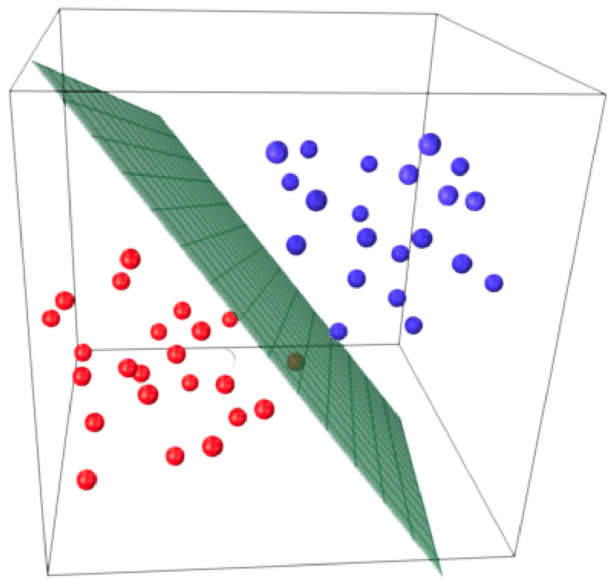
The main focus of logistic regression is to seperate the two classes using a hyperplane and then get values as 1 or 0 apprpximately when put into sigmoid function.



The following is the hyperplane which seperates the classes giving positive and negative values respectively. The plane gives positive for one side and negative for other as its a hyperplane of that particular space. The equation of the comes out to be:

$$W_0 + W_1 x_1 + \ldots + W_n x_n = 0$$

where W_0,W_1,.. are weights which has to be trained using the gradient decent method, and x_1,x_2,.. are features of the dependent variable.

Then we can represent the features and weights as vectors for easy representation and manipulation. Let:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \end{bmatrix} \quad and \quad W = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \end{bmatrix} \quad then\ z(x) = W^T X + W_0$$

then we get the hypothesis function as:

$$h(x) = \sigma(z(x)) = \frac{1}{1 + e^{-\left(W^T X + W_0\right)}} = \frac{1}{1 + e^{-(W_0 + W x_1 + \ldots + W_n x_n)}}$$

from bernoulli's distribution equation we get:

$$B(y_p) = (y_p)^y(1-y_p)^{1-y} \qquad where,\ y_p\ is\ predicted\ value$$
$$y\ is\ the\ actual\ value$$

we define cost function as $-log(B(x))$ for one sample, as it penalises more even for small deviation from the actual value(can be infered from the graph). Then for entire set it is,

$$cost(y_p) = -\frac{1}{m}\sum_{i=1}^{m}[(y)log(y_p) + (1-y)log(1-y_p)] \tag{1}$$

then,

$$cost(y_p) = \begin{cases} y=1 & \sum log\left(1+e^{-z(x)}\right) \\ y=0 & \sum log\left(1+e^{z(x)}\right) \end{cases}$$

then differentiating we get,

$$\frac{\partial cost(y_p)}{\partial W_0} = \frac{1}{m}\begin{cases} y=1 & \sum \frac{-e^{-z(x)}}{\left(1+e^{-z(x)}\right)} = \sigma(z(x))-1 \\ y=0 & \sum \frac{e^{z(x)}}{\left(1+e^{z(x)}\right)} = \sigma(z(x))-0 \end{cases}$$

$$\frac{\partial cost(y_p)}{\partial W_j} = \frac{1}{m}\begin{cases} y=1 & \sum \frac{-x_je^{-z(x)}}{\left(1+e^{-z(x)}\right)} = x_j(\sigma(z(x))-1) \\ y=0 & \sum \frac{x_je^{z(x)}}{\left(1+e^{z(x)}\right)} = x_{j(}\sigma(z(x))-0) \end{cases}$$

then we can write it just as ,

$$\frac{\partial cost(y_p)}{\partial W_0} = \frac{1}{m}\sum_{i=1}^{m}[h(X)-y_i]$$

$$\frac{\partial cost(y_p)}{\partial W_j} = \frac{1}{m}\sum_{i=1}^{m}x_j^i[h(X)-y_i]$$

In vector form the approximate algorithm would be,

$$W_0 = W_0 - \frac{\alpha}{m}\sum_{i=1}^{m}\left[\sigma\left(W^TX+W_0\right)-y_i\right]$$

$$W_j = W_j - \frac{\alpha}{m}\sum_{i=1}^{m} x_j^i\left[\sigma\left(W^T X + W_0\right) - y_i\right]$$

$$where\ y_i \in \{0, 1\}$$
$$where\ j = 1, 2, .. n$$

Alpha is a constant which determine the rate of covergence of the process, it shouldn't be too small or too big. If its too small the convergence rate will be very slow and on other hand if its too big then it may never hit the minimum and instead deviate far away.

These equations should be iterated several times to get the desired minimum-error weights. .