CLASSIFICATION

Classification is the problem of identifying to which set of categories (sub-populations) a new observation belongs, based on a training set of data containing observations (or instances) whose category membership is known.

Binary Classification

- It is a type of classification which involves only two categories. i.e all the observations can be split into two categories.

- Examples

1. Checking if email is spam or not
2. Whether a tumour is malignant or not

Multiclass classification

- Here the observations can be split into more than two categories.

- Examples

1. Match prediction (Win/loss/draw)
2. Predicting the cover type in a forest

How to solve such classification problems ?

- Can we use Linear regression ?

1. We can threshold the classifier output (i.e. anything over some value is yes, else no)

2. For example, if we have a binary classification we can set threshold as 0.5 and depending upon the classifier output we can predict

Issues

1. Few outliers can spoil the model.
2. The value of the classifier output may be more than 1 or less than 0. But it should be between 0 and 1.
3. It leads to erroneous outputs.

Logistic Regression

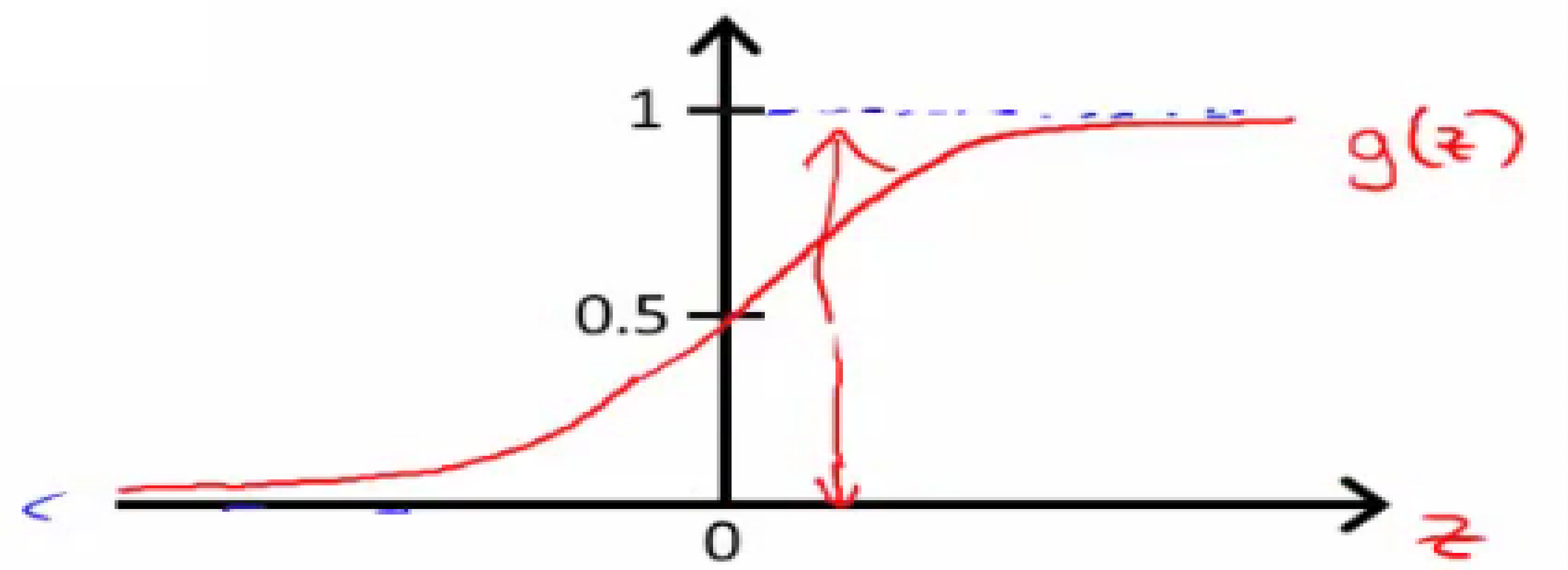
- So we need an algorithm which gives value between 0 or 1(or It can generate only discrete values 0 and 1).
- Logistic regression is a classification algorithm.
- It is used to solve binary as well as multi class classification problems.

Hypothesis Representation

- When using linear regression we used $h_\theta(x) = (\theta^T x)$
- For classification hypothesis representation we do $h_\theta(x) = g((\theta^T x))$
    - Where we define g(z)
        - z is a real number
    - $g(z) = 1/(1 + e^{-z})$
        - This is the **sigmoid function**, or the **logistic function**

Why did we use a different hypothesis ?

- The graph of the above hypothesis looks as below



Contd..

- It crosses 0.5 at the origin and flattens out (i.e it has asymptodes when the y value is at 1 or 0)
- So basically it gives 0 at negative infinity and 1 at positive infinity
- So we can be assured that the value returned by the function is between 0 and 1.
- Then we can use some threshold for Predicting the value.

Understanding what hypothesis is computing

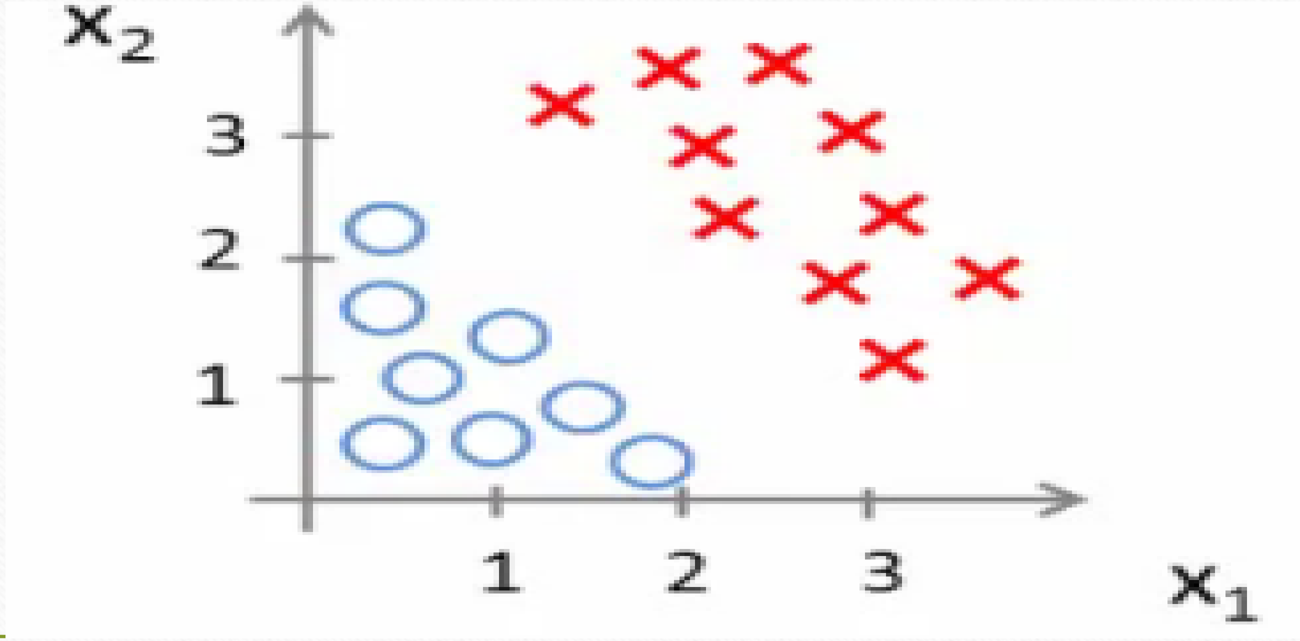
- The sigmoid function g(z) gives a value greater than 0.5 when the value of z is positive.
- $\theta^T x >= 0$

- So if we set the threshold as 0.5 then we can say that

1. If value of z is positive then it predicts 1
2. Or else it will give you 0.

Decision Boundary with Example

- Consider the following diagram



Contd

- Assume that

  $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

So if we have set the threshold as 0.5 then we can say

1. The model outputs the value 1 if

   $\theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0$

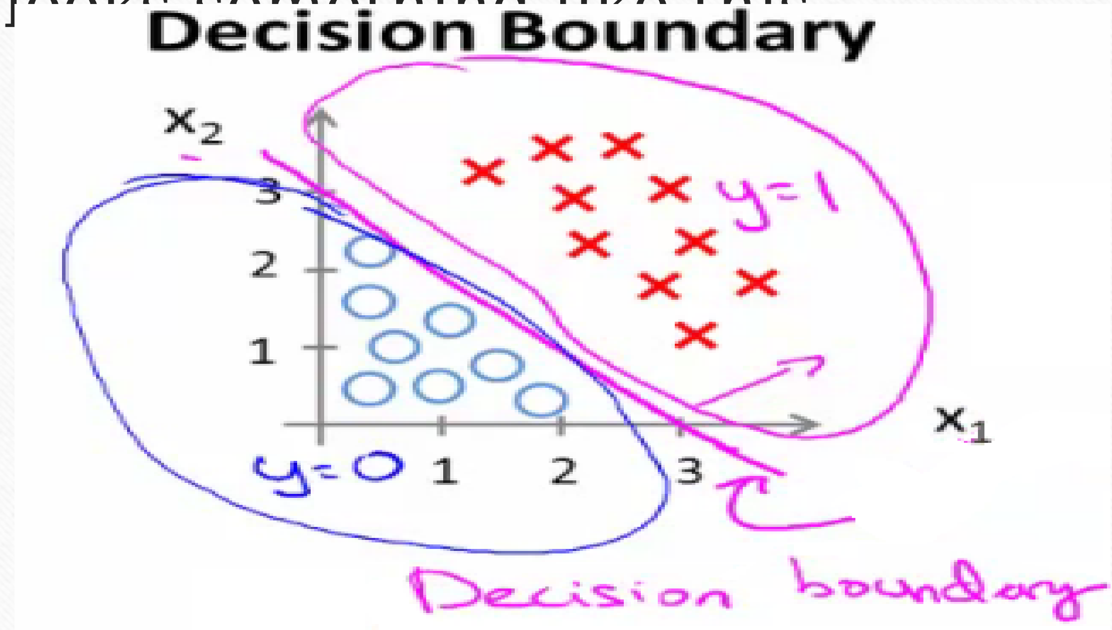
2. It will output 0 in the other case.

Decision Boundary

- The decision boundary is a property of the hypothesis
- Means we can create the boundary with the hypothesis and parameters without any data
- Later, we use the data to determine the parameter values
- We use decision boundary to split the observations into different categories

Linear Decision boundary

- For the above example using the same h(x) we will get decision boundary which looks something like this



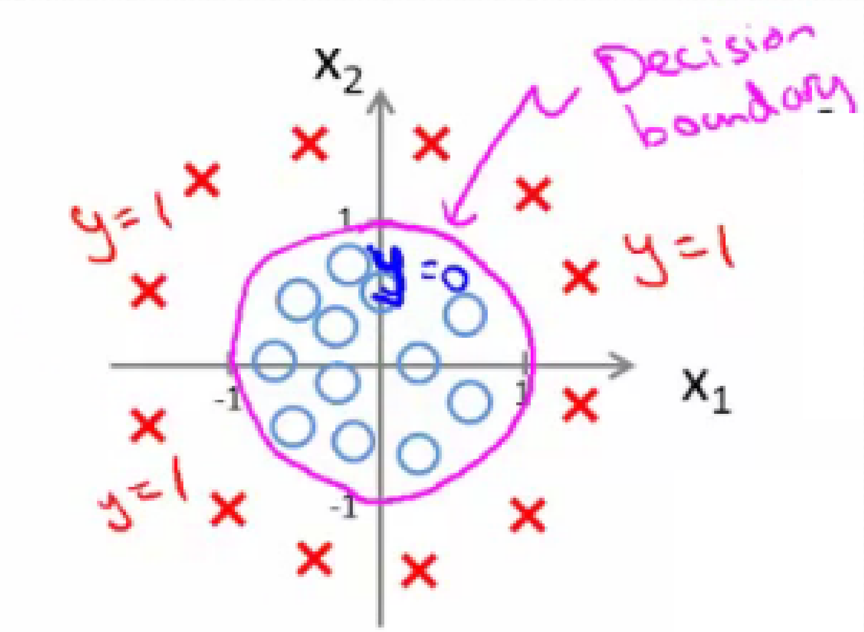
Non linear Decision boundary

- Suppose our classifier has higher order terms
- Used logistic regression to fit a complex non-linear data set
- Because just a linear fit won't be able to fit all models
- It may require some function like this

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Contd..

- In such cases the decision boundary won't be linear.
- One such example is given h



Cost Function

- For linear regression we use the following the cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

This is the cost you want the learning algorithm to pay if the outcome is $h_\theta(x)$ and the actual outcome is y

- If we use this function for logistic regression this is a **non-convex function** for parameter optimization

- Our hypothesis function has a non-linearity (sigmoid function of $h_\theta(x)$). This is a complicated non-linear function

- If you take $h_\theta(x)$ and plug it into the Cost() function, and them plug the Cost() function into $J(\theta)$ and plot $J(\theta)$ we find many local optimum because it's a *non convex function*

- Lots of local minima mean gradient descent may not find the global optimum - may get stuck in a global minimum

- We would like a convex function so if you run gradient descent you converge to a global minimum

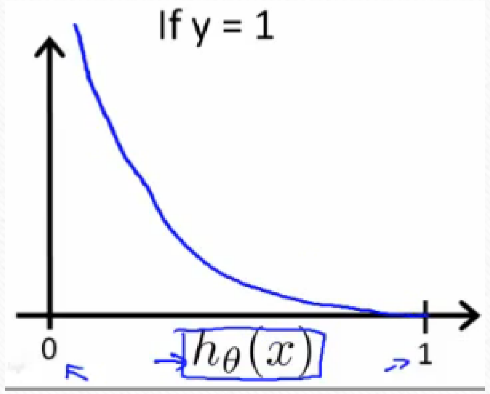
So what do we do ?

- We need a different convex cost function
- So our new cost function which is given below

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

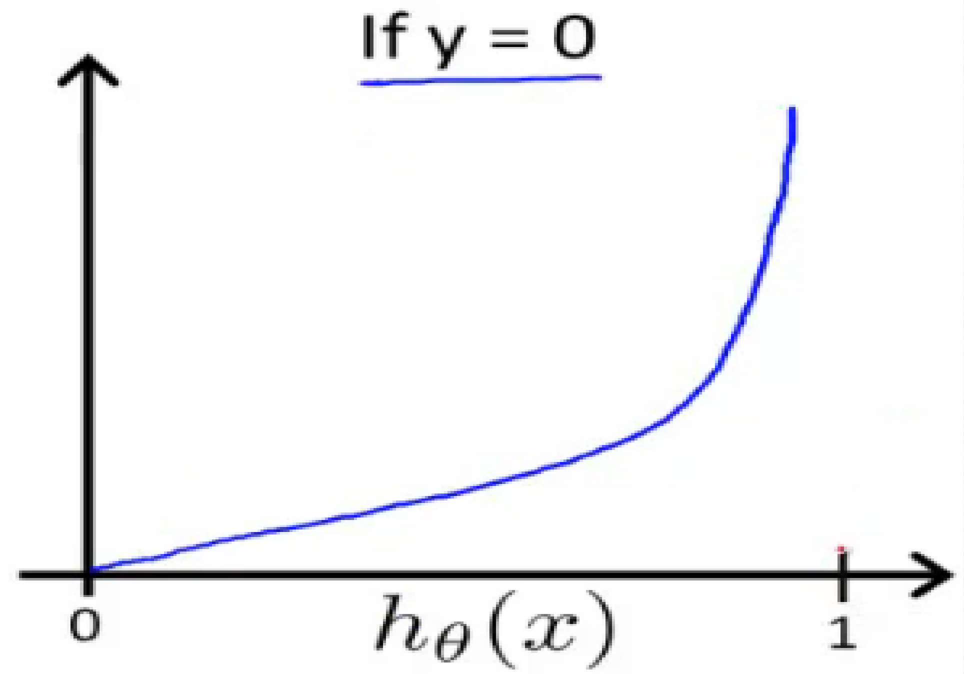
How exactly the cost function is penalizing ?

- If y = 1, the cost function looks as below



As you can see the function is tending to infinity(penalising heavily) as h(x) Is tending to 0 and the penality is tending to zero when h(x) is tending to 1.

- For y= 0 the cost function looks like this



As you can see the function is tending to infinity(penalising heavily) as h(x) Is tending to 1 and the penality is tending to zero when h(x) is tending to 0.