Binary Classification:

In this our dependent variable (y) is a discrete variable where it takes two values i.e. $y \in \{0,1\}$ where 0 represents negative class and 1 represents a positive class.

Examples:

Email : Spam or not ?

Online Transactions : Fraud or not ?

Tumor : malignant or Benign ?

Threshold classifier output : We use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually a linear function.

What if we use linear reg for classification problem ?

If we use linear regression model then out predicted output might be <0 or sometimes it might be >1, both the cases are not suitable for classification problem.

We can use classification algorithms such as **logistic Regression** for the binary classification.

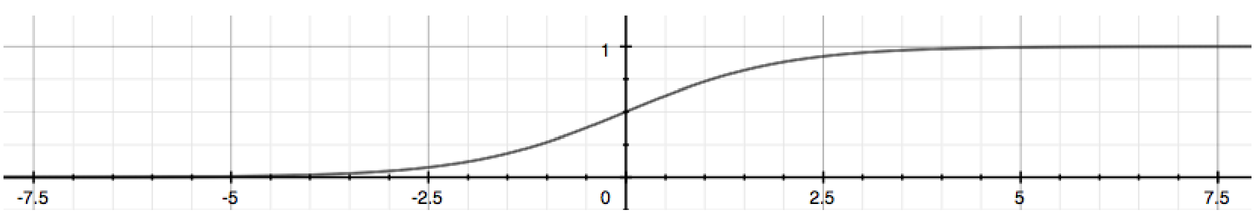
Logistic Regression :

Our goal is to have our hypothesis function between 0 to 1, i.e. $0 <= h_\theta(x) <= 1$.

θ

For Logistic regression we modify our hypothesis function as $h_\theta(x) = g(h_\theta(x))$, i.e. $h_\theta(\theta^T X) = \frac{1}{1+e^{-\theta^T X}}$

where g(z) is a logistic/sigmoid function, where $g(z) = \frac{1}{1+e^{-z}}$, the graph looks like below.



Interpretation of Hypothesis Output:

$h_\theta(x)$ = estimated probability that y=1 on input x.

Example of cancer prediction : $h_\theta(x) = 0.7$ → it tells that 70% chance of tumor being malignant tumor.

Another way of notation is $h_\theta(x) = P(y=1/x; \theta)$ → " *probability that y=1, given x, parameterized by* θ"

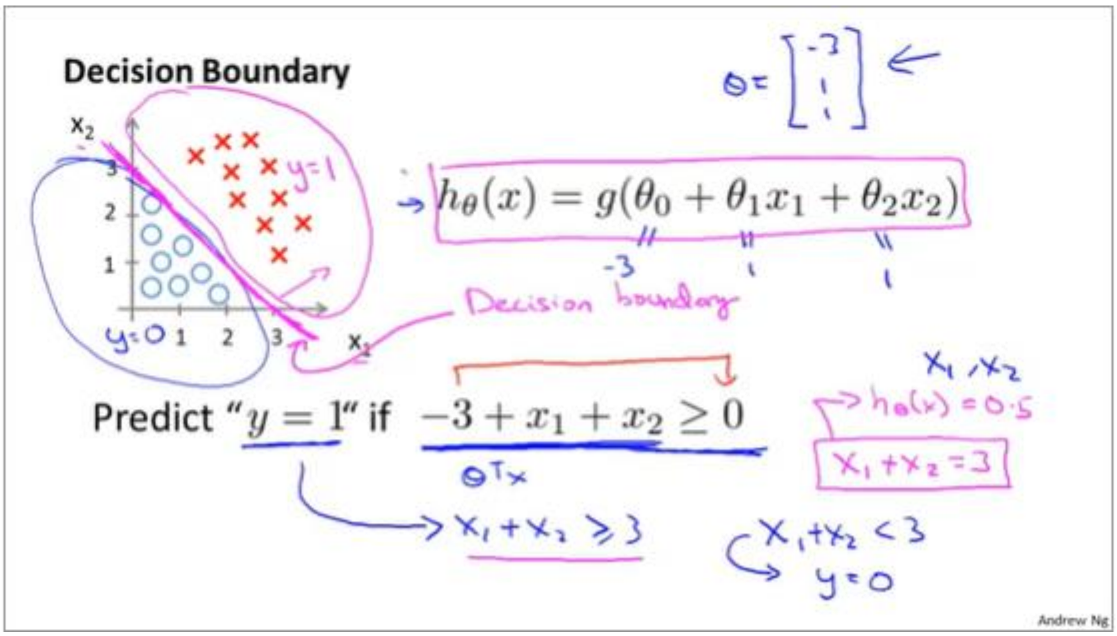
Also, P(y=1/x; θ) + P(y=0/x; θ) =1.

<u>Decision Boundary</u> :

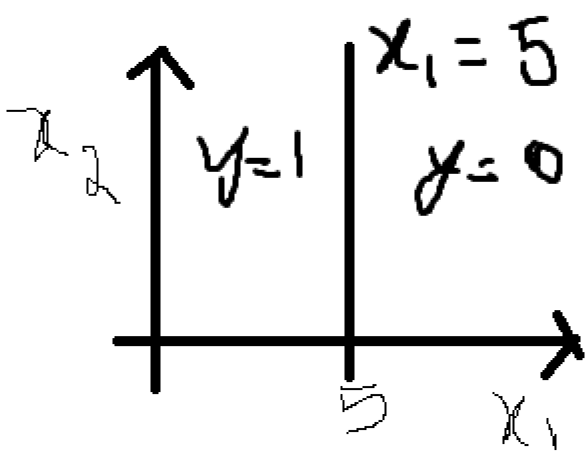
From sigmoid graph, we can say that g(z) >= 0.5 when z>=0, suppose if z=0 →g(z) =1/2

That is

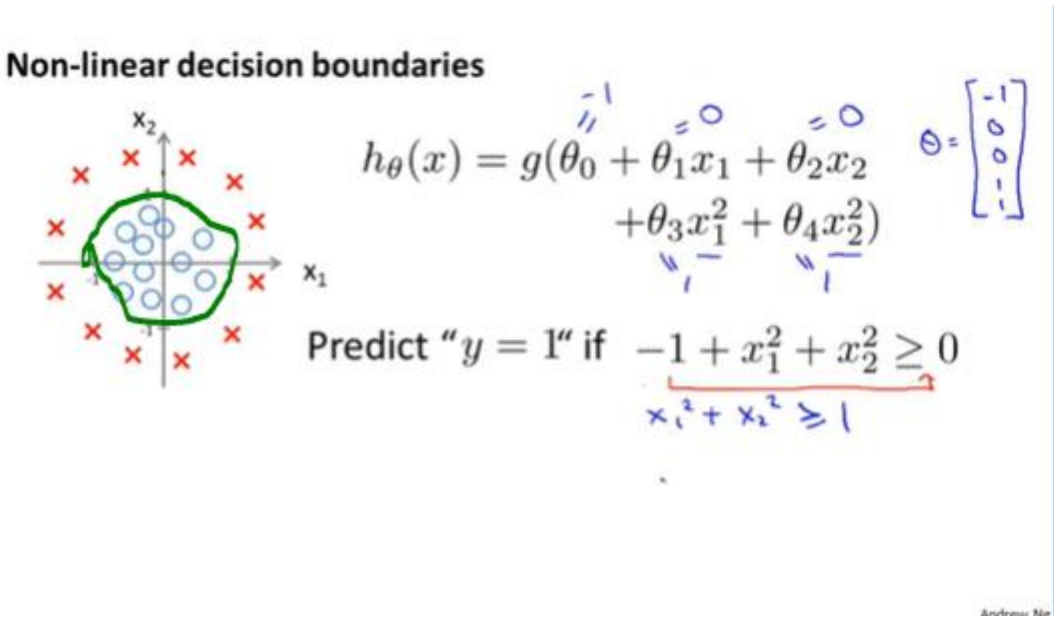
- for y=1, → g($θ^T$X) >=0.5 i.e. ($θ^T$X) >= 0. Similarly
- for y = 0 → g($θ^T$X) < 0.5 i.e. ($θ^T$X) < 0.



- Consider logistic regression with two features *x*1 and *x*2. Suppose theta_0 = 5 , theta_1 = -1 and theta_2=0, so that *hϑ*(*x*)=*g*(5−*x*1). Which of these shows the decision boundary of *hϑ*(*x*)?
- For y=1, $θ^T$X i.e. 5-x1 >= 0 → x1 <= 5 is the required decision Boundary equation.



- The decision boundaries might not always be linear, it can also be non-linear as well.



<u>Cost Function for Logistic Regression</u>:

Training set : $\{(x^1, y^1), (x^2, y^2), \ldots\ldots, (x^m, y^m)\}$, m examples and we have our X $\in \begin{vmatrix} x^0 \\ \vdots \\ x^n \end{vmatrix}$ i.e. a (n+1) dimensional vector, x0 =1 and y $\in$ {0,1}.

Our hypothesis/Objective function, $h_\theta(\theta^T X) = \frac{1}{1+e^{-\theta^T X}}$

Now how to choose parameters theta(θ) ?

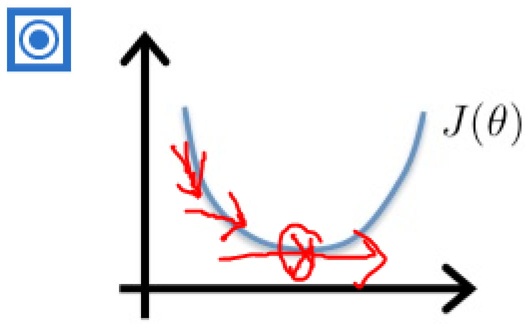
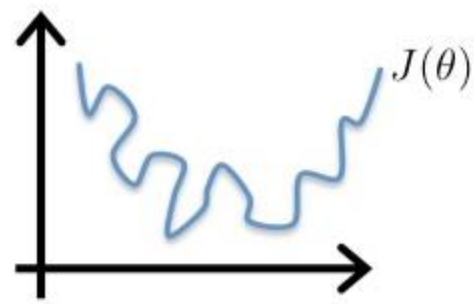
Cost function :

For Linear Regression cost function → $J(\theta) = \frac{1}{2m} * \sum_{i=1}^{m} (h_{theta}(x^i) - y^i)^2$

$$= \frac{1}{m} * \sum_{i=1}^{m} cost(h_{theta}(x^i), y^i)$$

Where → $Cost(h_{theta}(x), y) = \frac{1}{2}(h_{theta}(x) - y)^2$,for logistic regression this cofunction will be a non-convex (it will have lot of local minimums).

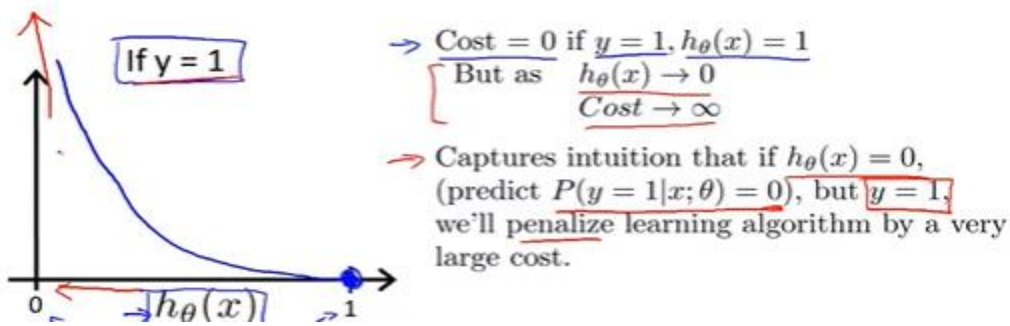
Because of squared cost function and the sigmoid functions non-linearity the graph will be non-convex, so it will be impossible to find global minimum.

Convex	Non-Convex
	

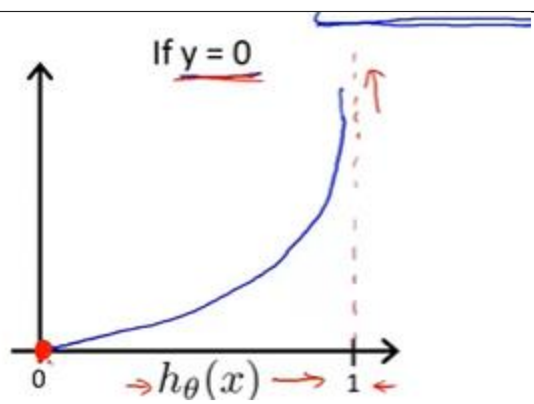
So, we use a different cost function for logistic regression.

$$Cost(h\_theta\ (x), y) = \begin{cases} -\log\left(h_{theta}(x)\right)\ ,\ \ if\ y = 1 \\ -\log\left(1 - h_{theta}(x)\right),\ \ if\ y = 0 \end{cases}$$

<u>Case 1</u>: If y =1



<u>Case 2</u> : If y =0.

	If our hypothesis predicts as 1 but the value of y is zero, then we get large cost value to be penalized that shows that our hypothesis is predicting the wrong class label. If hypothesis =0 and y =0 then the cost value is zero.

Why log1 =0 ?

- A logarithm is defined as the exponent or power to which a base must be raised to get some new number. It is a convenient approach to express large numbers.

  Example :$10^2 = 100$ which can be written as $2 = \log_{10}100$, Common or Briggsian logarithm is the logarithm with base 10.

  So, we know anything to the power of zero is one → $10^0 = 1$

  $= \log 10^0$

  $= 0\log_{10}10$

  $= 0$. (since $\log_{10}10 = 1$)

Why anything to power 0 is one i.e. x^0=1 ?

- We know $a^m/a^n = a^{m-n}$
- if m=n:
  - ➔ $a^m/a^m = a^{m-m}$
  - ➔ $1 = a^0$
  - ➔ $a^0 = 1$

  also remember 0^0 !=1

<u>**Simplified cost function and gradient descent**</u>:

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

$$= \frac{1}{m} * \sum_{i=1}^{m} cost(h_\theta(x^i), y^i)$$

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

Note : y =0 or 1 always

➔ Cost = $-y\log(h_\theta(x)) - (1-y)\log(1 - h_\theta(x))$

Logistic regression Cost Function : (based on maximum likelihood)

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

$$= -\frac{1}{m} * \left[\sum_{i=1}^{m} y\log(h_\theta(x)) + (1-y)\log(1 - h_\theta(x))\right]$$

To fit parameter θ : that minimize J(θ)

Gradient descent :

Goal to minimize J(θ)

Repeat {

$$\theta j \; = \; \theta j \; - \alpha * \frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta\left(x^i\right) - y^i\right) x_j^i\right)$$

}

$$J(\theta) \; = -\frac{1}{m} * \left[\sum_{i=1}^{m} y \log\left(h_\theta(x)\right) + (1-y)\log\left(1-h_\theta(x)\right)\right]$$

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial\theta_j}g(\theta^T x) \\
&= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial\theta_j}\theta^T x \\
&= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right)x_j \\
&= \left(y - h_\theta(x)\right)x_j
\end{aligned}$$

Vectorized Implementation :

- ➔ $h_\theta(\mathrm{x}) = g(\theta \mathrm{x})$
- ➔ cost = $1/\mathrm{m} * [-y^T * \log(h_\theta(\mathrm{x})) - (1-\mathrm{y})^T * \log(1-h_\theta(\mathrm{x}))]$
- ➔ $\theta := \theta - \frac{\alpha}{m}X^T(g(X\theta) - \mathrm{y})$

Out Scope for now : Usage of 1x1 convolution ?

We can shrink the number of channels, based on the number of 1x1 convolutional filters.