# * Logistic Regression *

Binary classification.

we will learn binary class classification problem / multiclass we will learn later

** calling logistic regression → 'regression' isn't correct it actually a classification problem not regression one.

Linear regression

Polynomial regression

Logistic classification.

## * Hypothesis Representation *

Denote

$P(y=0 \mid x;\theta)$ ⇒ Probability that $y=0$ for given input values of $x$ which is ~~that~~ parametrised by $\theta$.

Let's choose of function for our ; {There are some}

A function which is such that for ~~x~~ all range of input values yeilds result $\in (0,1)$ And it is calculus friendly.

The underlying assumption behind the sigmoid function is :→

$$\ln\left(\frac{P}{1-P}\right) = \sum_{0}^{K} b_i x_i = \theta^T x$$

odds ratio

$$\Downarrow$$

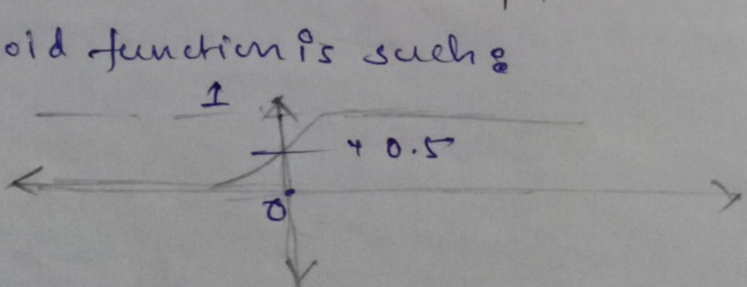
$$P = \frac{1}{1+\exp(\theta^T x)}$$

$h(x^i)$

} The answer of sigmoid function is expressed in the form of probility.

## * Desicion Boundary *

Every sigmoid function is such:



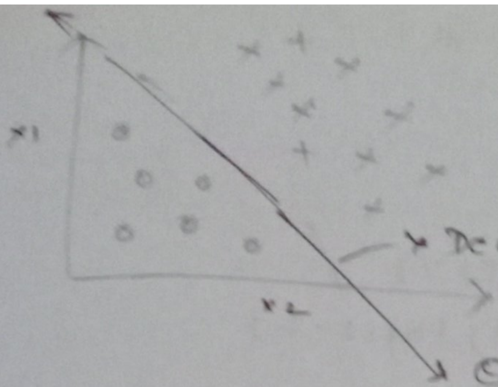

∴ when $h(x_i) = 0.5$

$\Downarrow$

$x = 0$.

Predict :→ $y = \begin{cases} 0 & \theta^T x < 0 \\ 1 & \theta^T x \geq 0. \end{cases}$

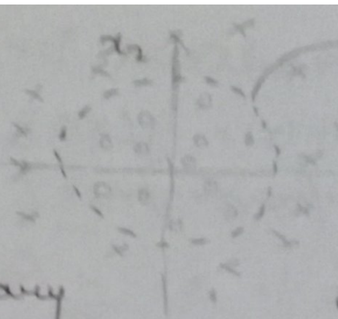
} This theory only applies when you final $\theta$ {parameters}




→ Decision boundary

→ $\theta^T x$

$= \theta_0 + x_1\theta_1 + x_2\theta_2$




→ Decision boundary

$\theta^T x$

↓

$\theta_0 + x_1\theta_1 + x_2\theta_2 + \underbrace{x_1^2\theta_3 + x_2^2\theta_4}$

Polynomial function forced to linear function

"Decision boundary is a property, not of training set, but of the hypothesis under the parameters"

---

## * Finding The Parameter Value ($\theta$) * for the most appropriate fit.

* Gradient Descent approach *

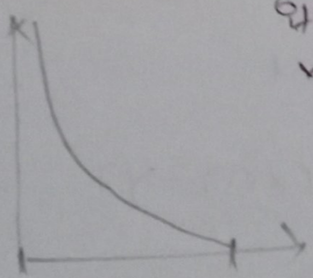
Minimise the cost of error.

$$J(\theta) = \underbrace{\sum_{1}^{m} \left( g( \cdots \right)}_{\text{what to use ***}} * \frac{1}{2m} = \frac{1}{m}\sum \boxed{\text{cost}(h, y)}$$

cost(h, y) ‖ probability prediction
$$\begin{cases} -\log(h(x)) & y = 1 \\ -\log(1 - h(x)) & y = 0 \end{cases}$$
} It is the product of some superior stastical maths

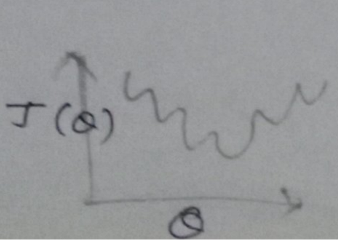
But let's concern ourselves only with intitution :→

when y=1



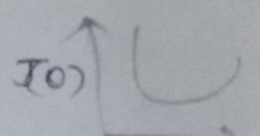

If you say no cancer when there actually a cancer then you have chosen death

** not $(h-y)^2$ because it will yeild




But when cost is

$$\text{cost} = y * (-\log P) - (1-y)\log(1-P)$$



* Gradient Descent *

** note that $h(x) = g(X\Theta)$

Repeat until convergence

{

$\cancel{\Theta := \Theta - \frac{\alpha}{m} \Sigma \, cost(P, Y)}$

} % $-y^i \log h - (1 - y^i) \log(1 - h)$

vectorised form

Repeat until convergence

{

$\cancel{\Theta = \Theta - y^T \log h - (1 - y)^T \log(1 - h)}$

}

The vectorised cost function:

$$J(\Theta) = \frac{1}{m} \Sigma \, cost(h, y)$$

$$\boxed{J(\Theta) = \frac{1}{m} \{ -y^T \log h - (1 - y)^T \log(1 - h) \}}$$

The gradient descent:

Repeat until convergence

{

$\Theta_j = \Theta_j - \frac{\alpha}{m} \frac{\partial}{\partial \Theta_j} J(\Theta)$ Foundation

$\Theta_j = \Theta_j - \frac{\alpha}{m} \Sigma (h - y^i) x_j^i$ } yes in case of log R also it comes out to be this

% But remember that

$h(X\Theta) \Rightarrow G(X\Theta)$

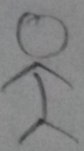
}

vectorised form of gradient descent

{

$\Theta = \Theta - \frac{\alpha}{m} X^T (g(X\Theta) - y)$

}



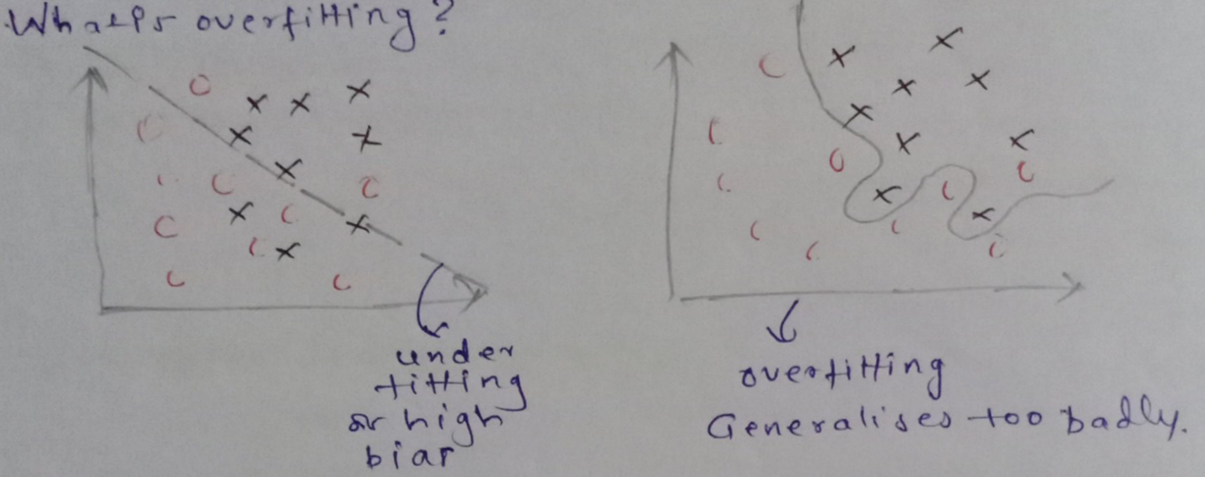
feature Scaling and Normalisation

Don't forget me

# * Overfitting *

↳ " It makes accurate prediction for the example in training set, but it may not generalise the hypothesis which give good result while predicting new or unknown queries "

What's overfitting?




How to adress overfitting?

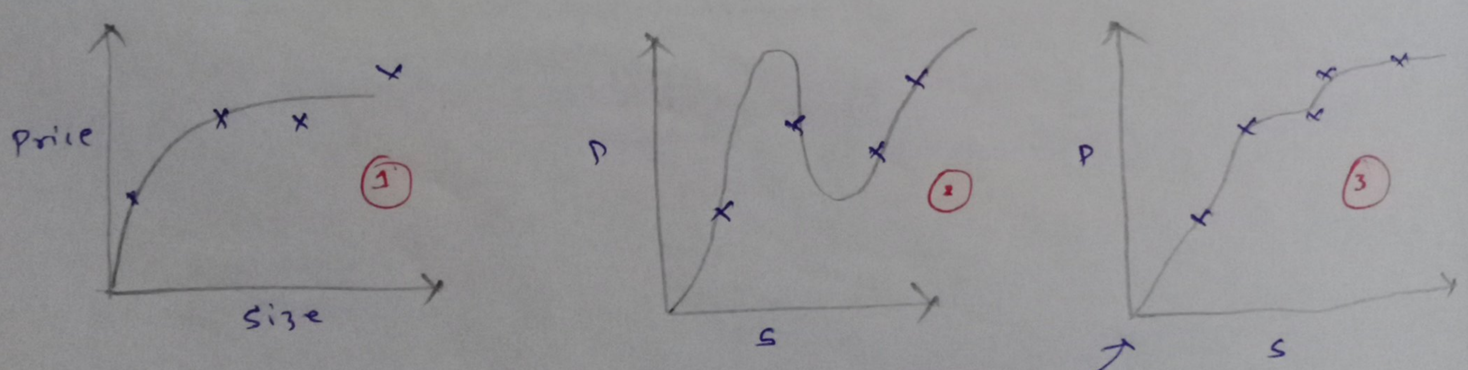
1. Reduce number of features { Also inc $x_3 = x_1 x_2^2$ }
   → Select which features to keep yourself
   → Model selection Algorithms
2. Regularisation.......

## * When to Use Regularisation? *

↳ There are many featurer { including the made up ones }

Effect of regularisation

1 Perfect fit




$p = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$

Overfitting → Regularisation

$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

Technically can extra term

How to reduce the effect of unnecessary term?
It is to reduce the value of parameter.

So we make cost function pay greatly for each parameter

$$\therefore \; J(\theta) = \frac{1}{2m} \sum_{1}^{m} (h-y)^2 + \underbrace{\sum_{1}^{n} (\lambda * \theta_i^2)}$$ regularisation parameter

Here $\theta_0$ is not included. because it's not amplifying any parameter

* We want to reduce some parameter terms ($\theta$) so that their amplification of undecisive ~~par~~ input's/features decreases. But / But / But their unwanted presence
* improves the quality of ~~dec~~ regression / Decision-boundry { see ③ >>> ① }

* Gradient Descent equation *

{ %. But before that note $J(\theta)$ of ~~for~~ is different for $\theta_0$ and rest.

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta}$$
}

$\Downarrow$

$$\{ \; \theta_0 := \theta_0 - \frac{\alpha}{m} \sum (h-y^{i}) x_0$$

$$\theta_j := \theta_j \underbrace{\left(1 - \alpha \frac{\lambda}{m}\right)}_{<1} - \frac{\alpha}{m} \sum_{1}^{n} (h-y) x_j^{(i)} \; \}$$

intuition. ⟸ so gradually the parameter is shrinking.

Regularisation:

Trade off between small error and simpler fit.

By adding more and more parameters I can actually reduce the squared error but at the same time is it actually worth it when compared to the penalty I am paying for it

Finally:

Improvised Normal Equations ⟶

$$\theta = (X^T X + \lambda . L)^{-1} X^T y$$

* Regularised Logistic Regression *

Again,

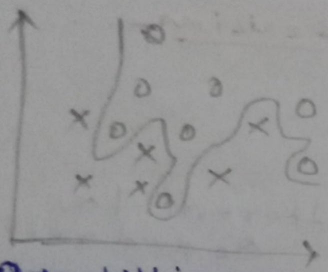
$$J(\theta) = \frac{1}{m}\sum_{1}^{m} \text{cost}(h,y) + \sum_{1}^{n+1} \theta_i^2 * (\lambda)$$

Regularization parameter.

Gradient Descent,

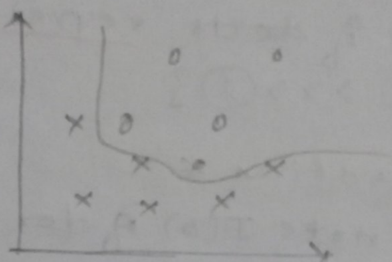
$$\left\{ \begin{array}{l} \theta_0 := \theta_0\left(1 - \cancel{\alpha\frac{\lambda}{m}}\right) - \alpha\frac{1}{m}\sum(h-y) \\ \theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum(h-y)x_j \end{array} \right\}$$

Same as before for linear regression



Overfitting.

$\Rightarrow$



Regularised

Found A NEW NAME for parameter $\theta$, "WEIGHTS"