Logistic Regression

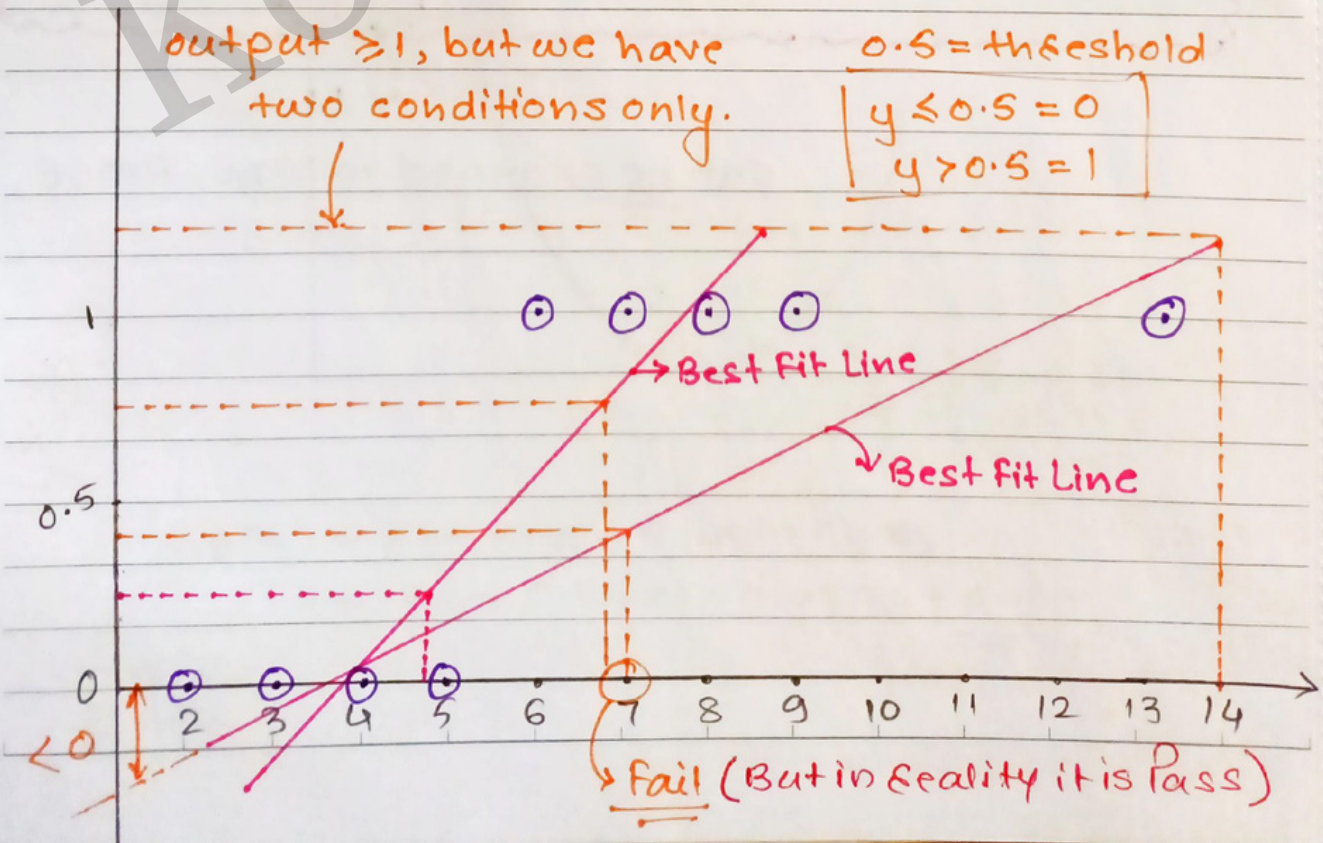
- classification problem

Eg.

dataset:

Study hours	Play hours	O/P (Pass/Fail)
2	8	Fail
3	7	Fail
6	3	Pass
outliers → 1	4	Pass

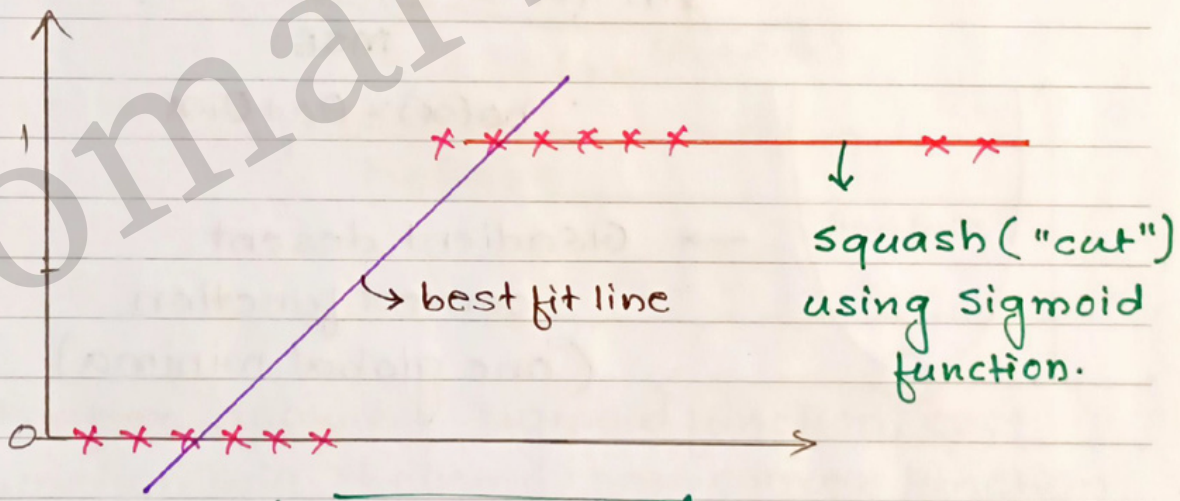
* we cannot perform regression, we need to perform classification into Pass / Fail.



so, even if a person study for 7hrs
$y < 0.5$
∴ the model will show Fail.
∵ we cannot used Linear Regression in this type of problem statement.

n we can
sing
s change and

- we cannot remove outliers always.
- Threshold can't be changed, once fixed.
- In logistic we squash ("cut") the line. we will not change the line.



$$h\theta(x) = \theta_0 + \theta_1 x$$

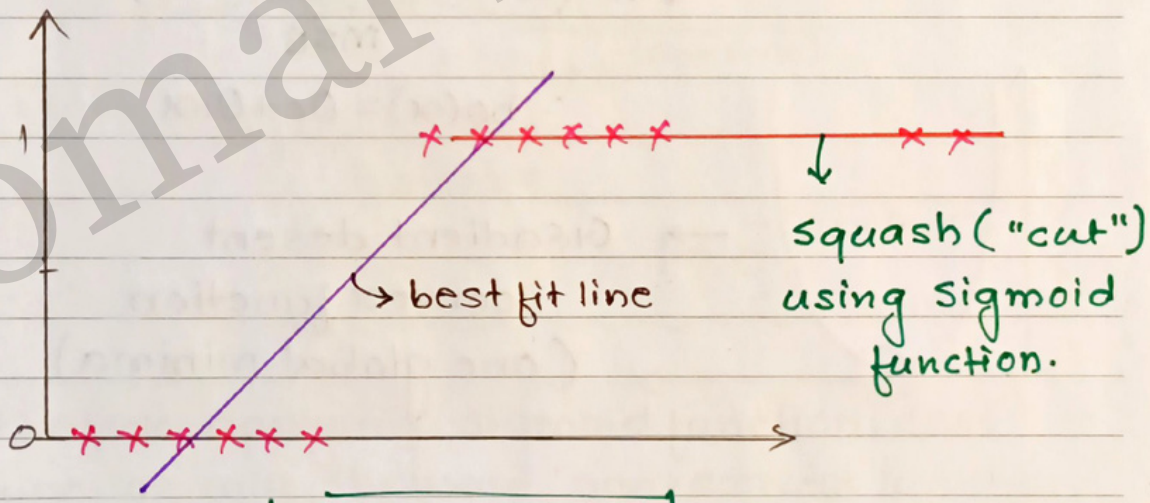
⇓

sigmoid function → output will range betn 0 and 1.

why we use logistic Regression when we can solve classification problem using Linear Regression?

→ • due to outliers best fit line gets change and results will be wrong.

- we cannot remove outliers always.
- Threshold can't be changed, once fixed.
- In logistic we squash ("cut") the line. we will not change the line.



$h_\theta(x) = \theta_0 + \theta_1 x$

⇓

Sigmoid function → output will range betn 0 and 1.

Sigmoid Function:

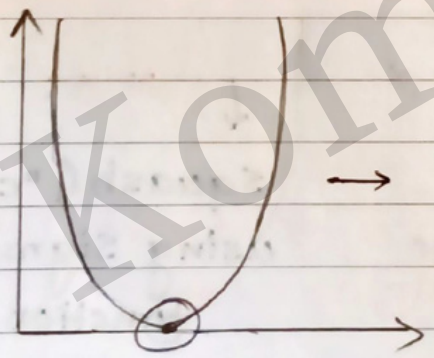
$$\text{sigmoid function} = \frac{1}{1+e^{-z}} \rightarrow \text{betn } 0 \text{ \& } 1.$$

1. create a best fit line.
2. squashing → sigmoid function

Linear Regression cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \underbrace{\left\{ h_\theta(x)^{(i)} - y^{(i)} \right\}^2}_{\text{MSE}}$$

$\therefore h_\theta(x) = \theta_0 + \theta_1 x$



→ Gradient desent
convex function
(one global minima)

Logistic Regression cost function:

steps:

① create best fit line
② apply squashing using sigmoid function.

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left\{ h_\theta(x)^{(i)} - y^{(i)} \right\}^2$$

↗ sigmoid function

$$h_\theta(x) = \sigma(\theta_0 + \theta_1 x)$$

~~let $z = h_\theta(x) = \theta_0 + \theta_1 x$~~

let $z = \theta_0 + \theta_1 x$

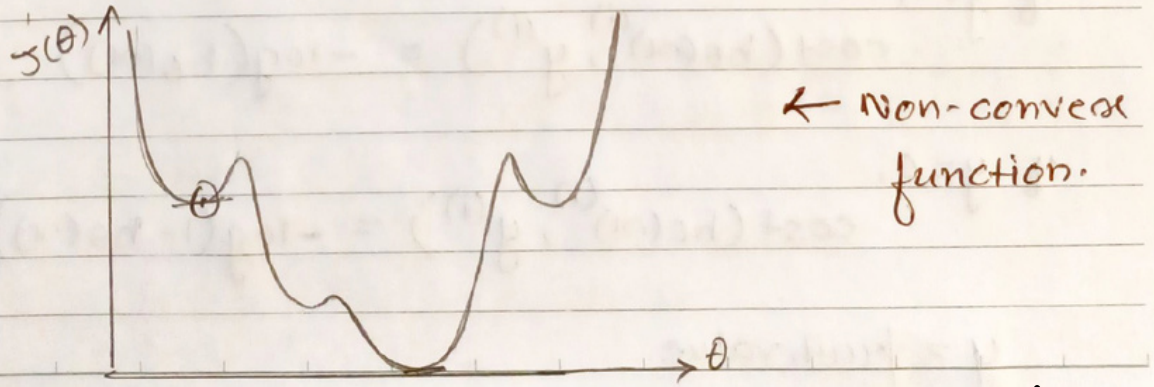
$$h_\theta(x) = \sigma(z)$$

$$h_\theta(x) = \frac{1}{1 + e^{-z}}$$

$$\therefore \quad h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

$$\therefore \quad \boxed{h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}}$$

but, after applying sigmoid function, cost function will become non-convex function and have a chances to get local minima.



← Non-convex function.

- change the cost function to solve the convexity problem.

- **Log Loss Cost Function**

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

→ convex function

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}$$

$$\text{cost}\left(h_\theta(x)^{(i)}, y^{(i)}\right) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

→ This will never give local minima.

If $y=1$,

$$\text{cost}\left(h_\theta(x)^{(i)}, y^{(i)}\right) = -\log(h_\theta(x))$$

If $y=0$,

$$\text{cost}\left(h_\theta(x)^{(i)}, y^{(i)}\right) = -\log(1-h_\theta(x))$$

y = truth value

minimize cost function $J(\theta_0, \theta_1)$ by changing θ_0, θ_1.

Convergence Algorithm:

Repeat Convergence

{

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}

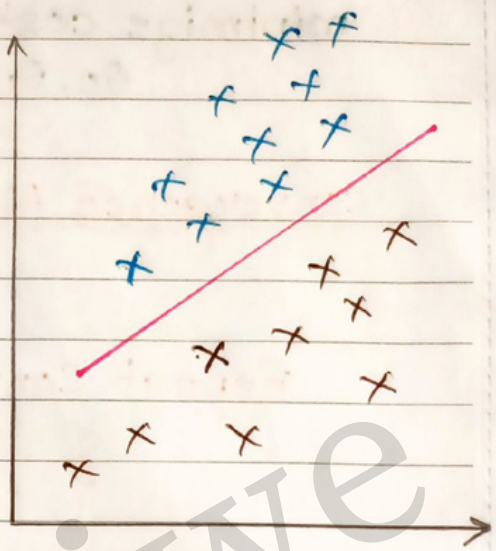
for $j = 0$ and $j = 1$

- by default take threshold = 0.5
  using ROC and TOC curve, we can define threshold.

Komal Diwe

Performance Metrics

1. Confusion matrix
2. Accuracy
3. Precision
4. Recall
5. F-Beta score



dataset.

f_1	f_2	o/p	$P = \hat{y}$ (model predict)
-	-	0	1
-	-	1	1
-	-	0	0
-	-	1	1
-	-	1	1
-	-	0	1
-	-	1	0

Confusion matrix

1 → correct prediction
0 → wrong prediction

predicted value (ŷ) \ Actual value (y)	1	0
1	3	2
0	1	1

← Actual value

predicted value (ŷ) \ Actual value (y)	1	0
1	TP	FP
0	FN	TN

correct match { TP: True Positive, TN: True Negative }

wrong match { FP: False Positive, FN: False Negative }

Accuracy

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

Eg. $\text{accuracy} = \frac{3+1}{3+2+1+1} = \frac{4}{7} = 57\%$

dataset → Binary classification

↓

1000 datapoints

↙ ↘

900 (1) 100 (0) ← Imbalanced dataset

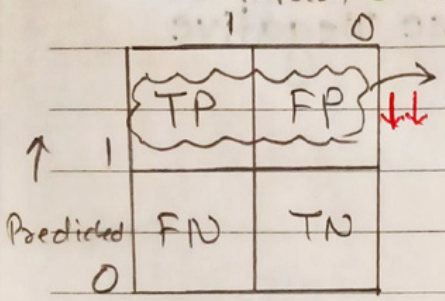
dumb model → 1 ⇒ we get 90% accuracy.

- If the accuracy is 90%, it is not sufficient. model is not good. To overcome this problem we can use Recall and Precision.

Komal Diwe

· Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$



out of all the predicted values how many are correctly predicted.

· our aim is to reduce False Positive (FP)

Eg. ① mail → spam or harm

TP: actual → spam and predicted → spam (Good)

FN actual → spam and predicted → harm (not good)

FP: actual → harm and ~~actual~~ predicted → spam (mail is not spam) (critical problem)

focus on reducing FP.

② Diabetes or Not diabetes

Actual → diabetes and Predicted → Not diabetes

critical problem → reduce FN.

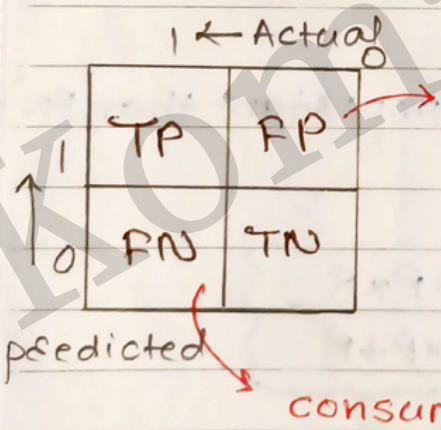
Recall :

$$\text{Recall} = \frac{TP}{TP + FN}$$

→ out of all the actual values how many are correctly predicted.

Eg. Tommorow the stock market is going to crash.

Two points →
- consumers → FN↓
- companies → FP↓↓ (can take certain decision)



for FP companies can take action.
company (action: sale shares and discounted price)

F-Beta score

$$\text{F-Beta score} = \frac{(1+\beta^2)\,\text{Precision} \times \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

① If FP and FN are both important

$\beta = 1$

$$\therefore \text{F1 score} = \frac{2 \cdot P \times R}{P + R}$$

② If FP is more important than FN

$\beta = 0.5$

$$\text{F-0.5 score} = \frac{(1 + 0.25) P \times R}{0.25 \times P + R}$$

③ If FN >> FP, (FP is less important than FN),

$\beta = 2$

$$\text{F2 score} = \frac{(1 + 4) P \times R}{4 \times P + R}$$

In-depth Insights

Logistic Regression

Komal Diwe

Classification

Classification problem, is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values.

$y \in \{0, 1\}$ — Binary class

0 : " -ve class"

1 : " +ve class"

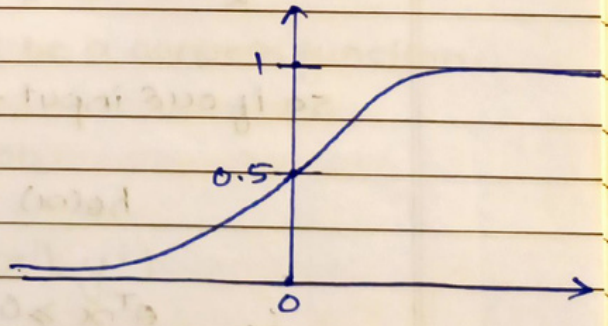
Hypothesis Representation

Intuitively, it doesn't make sense for $h_\theta(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$. To fix this, let's change the form for our hypothesis $h_\theta(x)$ to satisfy $0 \leq h_\theta(x) \leq 1$.

This accomplished by plugging $\theta^T x$ into the logistic function.

our new form uses the "Sigmoid Function" also called the "Logistic Function".

$$h_\theta(x) = g(\theta^T x)$$
$$z = \theta^T x$$
$$g(z) = \frac{1}{1 + e^{-z}}$$



Function $g(z)$, shown here, maps any real number to the $(0, 1)$ interval, making it useful for transforming an arbitary-valued function into a function better suited for classification.

$h_\theta(x)$ will give us the probability that our output is 1.

Eg. If $h_\theta(x) = 0.7 \longrightarrow$ output : 1

thus $h_\theta(x) = 0.3 \longrightarrow$ 0

$h_\theta(x) = P(y=1|x;\theta) = 1 - P(y=0|x;\theta)$
$P(y=0|x;\theta) + P(y=1|x;\theta) = 1$

Decision Boundary

In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows.

$h_\theta(x) \geq 0.5 \rightarrow y=1$
$h_\theta(x) \leq 0.5 \rightarrow y=0$

The way our logistic function g behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:

$g(z) \geq 0.5$ when $z \geq 0$

Remember:

$z=0, e^0=1 \Rightarrow g(x)=1/2$
$z \rightarrow \infty, e^{-\infty} \rightarrow 0 \Rightarrow g(x)=1$
$z \rightarrow -\infty, e^{\infty} \rightarrow \infty \Rightarrow g(x)=0$

So if our input to g is $\theta^T x$, then that means

$h_\theta(x) = g(\theta^T x) \geq 0.5$ when $\theta^T x \geq 0$

$\therefore \quad \theta^T x \geq 0 \rightarrow y=1$
$\theta^T x \leq 0 \rightarrow y=0$

The decision boundary is the line that separates the area where $y=0$ & $y=1$. It is created by our hypothesis function.

Eg. $\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$

$y = 1$ if $5 + (-1)x_1 + 0 \cdot x_2 \geq 0$

$5 - x_1 \geq 0$

$-x_1 \geq -5$

$x_1 \leq 5$.

In this case, our decision boundary is a straight vertical line placed on the graph where $x_1 = 5$, and everything to the left of that denotes $y = 1$, while everything to the right denotes $y = 0$.

again, the input to the sigmoid function $g(z)$ (e.g. $\theta^T x$ doesn't need to be linear, could be a function that describes a circle (e.g. $z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$) or any shape to fit our data.

Cost Function

We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima.

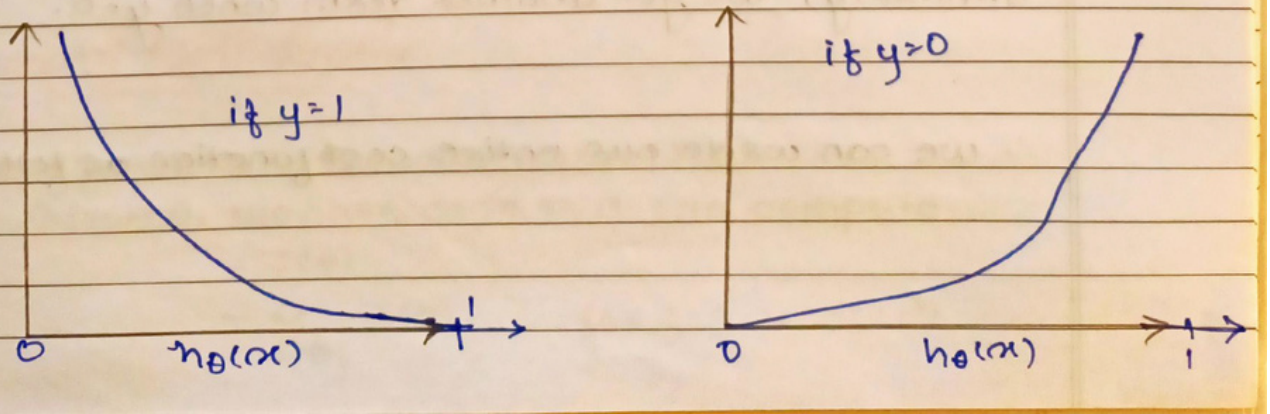
In other words, it will not be a convex function.

Cost function for logistic function:

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

$$\text{cost}(h_\theta(x^i), y) = -\log(h_\theta(x)) \quad \text{if } y = 1$$

$$\text{cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad \text{if } y = 0$$



$\text{cost}(h_\theta(x), y) = 0$ if $h_\theta(x) = y$
$\text{cost}(h_\theta(x), y) \to \infty$, if $y = 0$ and $h_\theta(x) \to 1$
$\text{cost}(h_\theta(x), y) \to \infty$ if $y = 1$ and $h_\theta(x) \to 0$

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.

Note that writing the cost function in this way guarantee that $J(\theta)$ is convex for logistic ~~function~~ regression.

Simplified Cost function and Gradient Descent

We can compress our cost function's two conditional cases into one case.

$$\text{Cost}(h_\theta(x), y) = -y \cdot \log(h_\theta(x)) - (1-y) \cdot \log(1 - h_\theta(x))$$

Thus, when we substitute $y = 1$ in abv eqn,
$\text{cost}(h_\theta(x), y) = -y \cdot \log(h_\theta(x))$
similarly, we get another term when $y = 0$.

∴ we can write our entire cost function as follows:

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

A vectorized implementation is:

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \cdot \log(1-h) \right)$$

Gradient Descent:

General form of gradient descent is:

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

we can workout the derivative part using calculus to get:

Repeat {
$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right) \cdot x_j^{(i)}$$
}

Notice that this algorithm is identical to the one we used in linear regression. we still have to simultaneously update all values in theta. ($h_\theta(x)$ is different)

A vectorized implementation is:

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

Advanced Optimization

cost function $J(\theta)$. want $\min_\theta J(\theta)$.

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ for $j = 0, 1, \ldots, n$

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS.

advantages :- (1) No need to manually pick α.
(2) often faster than gradient descent

disadv^n :- more complex to implement.

Multiclass classifications

Now we will approach the classification of data when we have more than two categories.

Instead of $y = [0,1]$ we will expand our definition so that $y = [0, 1, \ldots, n]$

since $y = [0,1,\ldots,n]$, we divide our problem into $n+1$ ($+1$ bcz the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a ~~member~~ member of one of our classes.

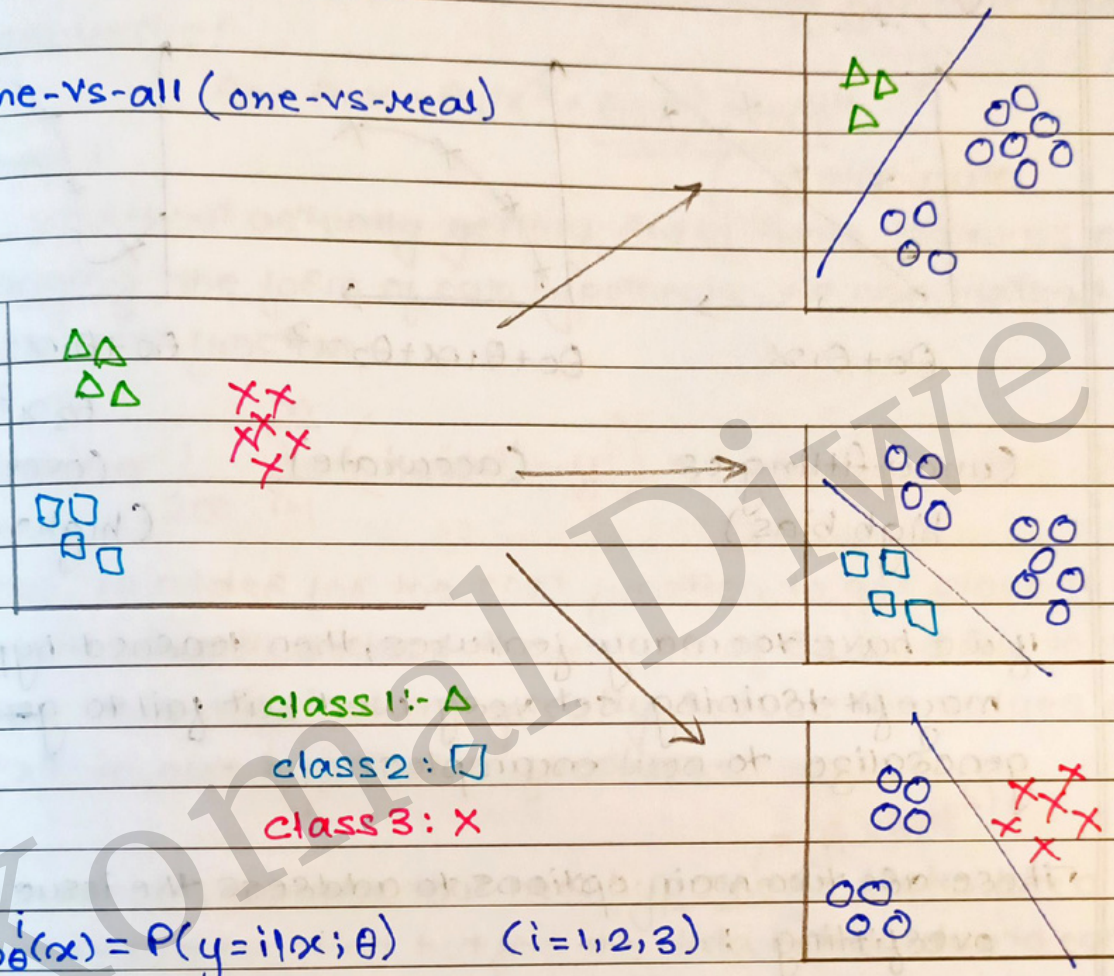
$$y \in \{0, 1, \ldots, n\}$$
$$h_\theta^0(x) = P(y=0|x;\theta)$$
$$h_\theta^1(x) = P(y=1|x;\theta)$$
$$\vdots$$
$$h_\theta^n(x) = P(y=n|x;\theta)$$
$$\text{prediction} = \max_i\left(h_\theta^i(x)\right)$$

we are basically choosing one class and then lumping all the others into a single second class. we do this repeatedly;

Date: / /

applying binary logistic regression to each case, and thus use the hypothesis that returned the highest value as our prediction.

one-vs-all (one-vs-rest)



class 1:- △

class 2: □

class 3: X

$h_\theta^{(i)}(x) = P(y=i|x;\theta) \quad (i=1,2,3)$

To summarise:

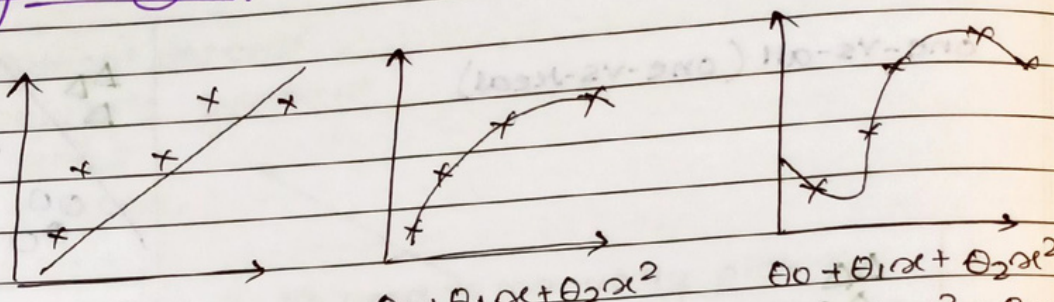
Train a logistic regression classifier $h_\theta(x)$ for each class i to predict the probability that $y=i$

on a new input x, to make a prediction, pick the class that maximizes

$$\max_i h_\theta^{(i)}(x)$$

Date: / /

Regularization



$\theta_0 + \theta_1 x$ (underfitting or high bias)

$\theta_0 + \theta_1 x + \theta_2 x^2$ (accurate)

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ (overfit) (high variance)

If we have too many features, then learned hypothesis may fit training set very well but fail to ~~generalize~~ generalize to new examples.

There are two main options to address the issue of overfitting:

1. Reduce the number of features:
   - manually select which features to keep.
   - use a model selection algorithm
2. Regularization
   - keep all the features, but reduce the magnitude of parameters θ_j.
   - Regularization works well when we have a lot of slightly useful ~~resource~~ features.

Cost function

If we have overfitting from our hypothesis function, we can reduce the weight that some of the terms in our

function easily by increasing their cost.

Eg.

we wanted to make the following function more quadratic:

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

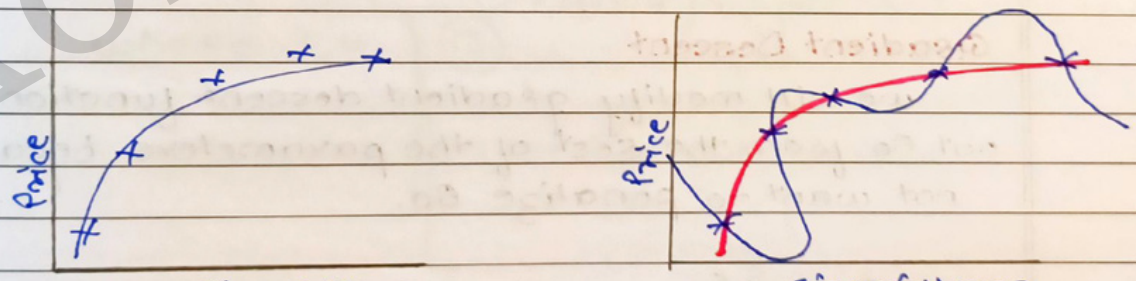
we will eliminate

without actually getting rid of these features or changing the form of our hypothesis, we can instead modify our cost function.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$$

Now, in order for the cost function to get close to zero, we will have to reduce the values of θ_3 and θ_4 to near zero. This will in turn greatly reduce the values of $\theta_3 x^3$, $\theta_4 x^4$ in our hypothesis function.

As a result, we see that new hypothesis (Pink Graph) looks like a quadratic function but fits the data better due to extra small items $\theta_3 x^3$ and $\theta_4 x^4$.



$\theta_0 + \theta_1 x + \theta_2 x^2$

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2 + 1000\theta_3^2 + 1000\theta_4^2$$

pink graph shows this complete equation.

we could also regularize all of our theta parameters in a single summations as:

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2$$

The λ, or lambda, is the **Regularization parameter**. It determines how much the costs of our theta parameters are inflated.

- using the above cost function with the extra summation, we can smooth the output of our hypothesis function to reduce overfitting.

- If lambda is choosen to be too large, it may smooth out the function too much and cause underfitting.

Regularized Linear Regression

Gradient Descent

we will modify gradient descent function to separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0.

Repeat {

$$\theta_0 := \theta_0 - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right) \cdot x_0^i$$

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i \right) \cdot x_j^i \right) + \frac{\lambda}{m} \theta_j \right]$$

$j \in \{1, 2, \ldots, n\}$

The term $\frac{\lambda}{m}\theta_j$ performs our regularization. With some manipulation our update rule can also be represented as:

$$\theta_j := \theta_j\left(1-\alpha\cdot\frac{\lambda}{m}\right) - \alpha\cdot\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i)-y^i)\cdot x_j^i$$

↳ always will be less than 1

Intuitively we can see it as reducing the value of θ_j by some amount on every update.

The 2nd term is exactly same as it was before.

Normal Equation:-

To add in regularization, the equation is the same as our original, except that we add another term inside the parantheses:

$$\theta = (X^TX + \lambda\cdot L)^{-1}X^Ty$$

where $L =$ → always zero

$$\begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix}$$

L is a matrix with 0 at the top left and 1's down the diagonal, with 0's everywhere else.

It should have dimension $(n+1)\times(n+1)$. Intuitively, this is the ~~identity~~ m identity matrix (though we are not including x_0), multiplied with a single real no. λ.

If $m<n$, then X^TX is non-invertible. However, when we add the term $\lambda\cdot L$, then $X^TX + \lambda\cdot L$ becomes invertible.

Regularized Logistic Regression

cost function:

cost function for logistic regression was:

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

we can regularize this equation by adding a term to the end:

$$J(\theta) = \frac{-1}{m} \sum_{i=1}^{m} \left[y^i \cdot \log(h_\theta(x^i)) + (1-y^i) \cdot \log(1-h_\theta(x^i)) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

means to explicitly exclude the bias term, θ_0. i.e. the θ vector is indexed from 0 to n (holding n+1 values, θ_0 through θ_n), and this sum explicitly skips θ_0, by running from 1 to n, skipping 0.

Thus, when computing the equation, we should continuously update the two following equations.

Gradient Descent

repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) \cdot x_0^i$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i + \frac{\lambda}{m} \theta_j \right]$$

$$\frac{\partial}{\partial \theta_j} J(\theta) \qquad j = (1, 2, \ldots n)$$