Chapter 1 :

Machine learning:

Field of study that gives computers the ability to learn without being explicitly programmed.

—Arthur Samuel.

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

— Tom Mitchell.

Machine learning is an application of Artificial intelligence (AI) that provides the systems the ability to automatically learn and improve from experience without being explicitly programmed.

Machine learning allows software applications to become more accurate in predicting outcomes without being explicitly programmed.

Example

Definition

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E

1) LP-6

→ Suppose your email watches which email you do or do not mark as spam and based on that learns how to better filter spam.

- Classifying emails as spam or not spam - Task(T)
- Watching you label emails as spam or not - Exp(E)
- The no. of emails correctly classified. - Performance(P)

2) A Robot driving learning problem:

T: Driving on public four lane highway using vision sensor.

P: Avg. distance traveled before an error (as judged by human overseen)

E: A sequence of images and steering commands recorded while observing a human driver

3) A checkers learning Problem:

T: playing checkers

P: percent of games won against opponents

E: Playing practice games against itself.

Sunidhi

Machine learning Algorithms
→ Supervised learning
→ Unsupervised learning

Supervised learning:
In machine learning its a task of inferring function from labeled training data. Trainging data consist of a set of training examples.
Here, each example is a pair consisting of a input object (vector) and a desired output.
A supervised learning algorithm analyzes the training data and produces an inferred function which can be used to map new examples like wise.
Supervised learning algorithm produces more right answers and often reffered as "Regression" problem

[Regression: Predict continuous valued output]

Supervised learning → Regression
→ Classification

Ex: You're running a company and you want to develop learning algorithms to address each of two problems

Problem 1: You have a large inventory of identical items. You want to predict how many of these items will sell over next 3 months.

Problem 2: You'd like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised.

Problem 1 is a Regression problem
bcoz suppose we have 1000's of value & u can treat it as real/continous value

Problem 2 is a classification problem
set values 0 — Not hacked
1 — Hacked. and predict the ~~via~~ algorithm

→ Approach steps to solve a supervised learning problem
1. Determine the type of training examples
2. Gather a training set. ie gathering a set of input objects and corresponding output
3. Determine the input feature representation of the learned function
4. Determine the structure/flow of the learning algorithm
5. Complete the design. ie Run the learning algorithm on the gathered training set
6. Evaluate the accuracy of the learned function.

## Unsupervised learning:

Its a machine learning task of inferring a function to describe unlabled data (classification or categorization is not included)

Since the ex given are unlabeled there is no evaluation of accuracy of the structure

Approaches to solve unsupervised learning may be clustering, Neural networks etc.

## Examples

→ A highly developed AI that serves as a house keeping Robot develops a theory that there is usually dust under sofa. Each week, the theory is confirmed as the robot often finds dust under the sofa. Nobody explicitly tell the robot the theory is correct but it develops on its own.

→ Given a set of news article found on the web, group them into set of articles of the same story

→ Given a database of customer data, automatically discover market segments and group customers into different market segments.

Semi Supervised learning:

Its a class of supervised learning. It lies between supervised and unsupervised learning.

- It has small amount of labled data and a huge amount of unlabeled data
- labled and unlabeled data when used in conjunction with labled data, considerabelly improves learning accuracy.

Reinforcement learning:

This differs from standard supervised learning in which correct input/output pair are never present, nor sub optimal action explicitly corrected. Instead the focus is on performance which involves finding balance betn parameters

Reinforcement learning allows machines & software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance.

LP-2

Application of Machine learning

1. Virtual Personal Assistants

Siri, Google, Cortona are some popular examples As the name suggests they assist in finding information when asked over voice.

For answering the personal assistant look out for the information stored or recalls your related queries or send a command to other resources to collect info.

# ML is an important part of these personal assistants as they collect & refine the info on the basis of your previous involvement with them

2. Video Surveillance

These systems now a day make it possible to detect crime before they happen. They track unusual behaviour of people. The sym can thus gives an alert to human attendants.

3. Social Media Services.

Social Media platforms utilize machine learning for their own and user benifits.

A few example include.

→ People You May know. ML works on simple concept understanding with experience. Facebook continuously notices the friend you connect, the often profile you visit, your interest work place or group that you share, and on this ~~bases~~ basis suggest frndz

→ Face Recognition: When uploaded a picture with a friend, the Facebook instantly recognizes that friend and check then match with the people in your friend list

→ Similar Pins: ML is the core element of Computer vision. Pinterest uses computer vision to identify the objects or pins in the image and recommend similar pins

4. Email Spam & Malware filtering

5. Search Engine Result Refining

6. Product Recommendations

# Chapter - 2 Supervised Learning

*Linear Regression: Linear Regression.

In supervised learning there is basically a data set where 'x' is the input and 'y's are the corresponding output, 'm' is the no. of training examples, and the job is to predict the output which is also known as hypothesis function

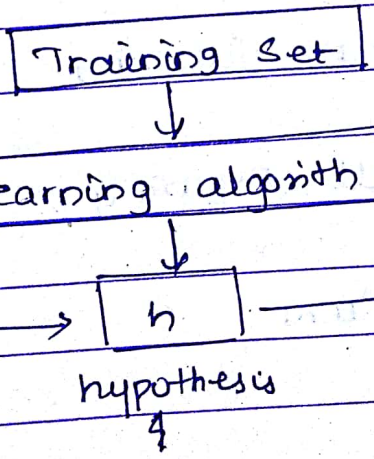
(x, y) - denotes single training example

$(x^{(i)}, y^{(i)})$ - denotes $i^{th}$ training example
index

| Ex: Size in $feet^2$ (x) | Price in 1000's $ (y) |
|---|---|
| 2104 | 460 |
| 1416 | 232 |
| 1534 | 315 |
| 852 | 178 |
| ⋮ | ⋮ |

$x^1 = 2104$ $y^1 = 460$
⋮

## working of supervised learning algorithm




h - function that maps x's to y's

Nidhi

## Hypothesis & Representing hypothesis :

Its an estimate or model or function with limited inputs/ training examples / features to which an inference is drawn at that instant.

Its a proposed explanation made on the basis of limited evidence as a starting point for further investigation

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

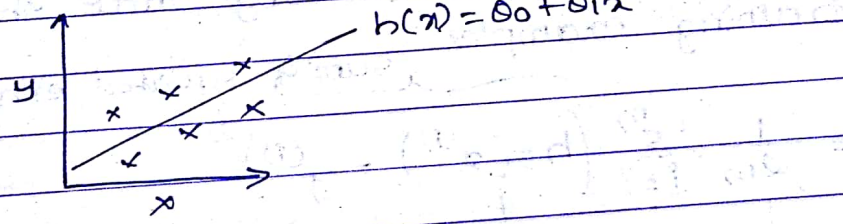
$\theta_0, \theta_1 \rightarrow$ parameters/weight

$h_\theta(x) \rightarrow$ predicted O/P.

$\parallel$

$h(x)$

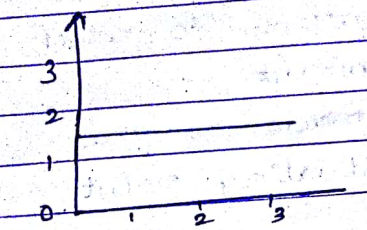
$h_\theta(x)$ means that we are predicting y some straight line function. ~~straight~~ or linear function



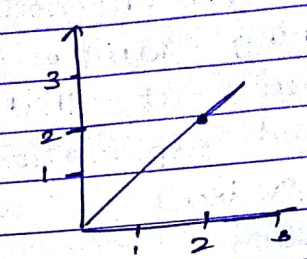

So this regression is having only one variable x there is no ~~ith~~ i'th index as such hence the name linear regression with one variable or also known as Univariate linear regression

Cost functions

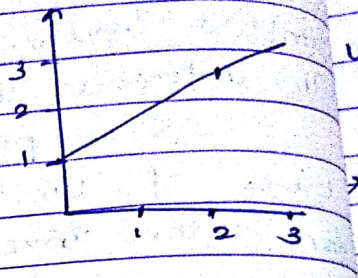
$h_\theta(x) = \theta_0 + \theta_1 x$



$\theta_0 = 1.5$
$\theta_1 = 0$

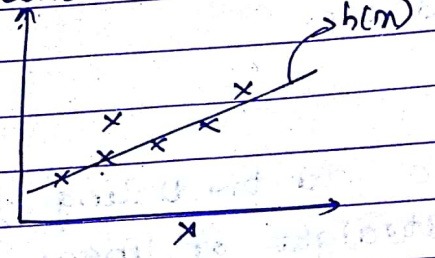


$\theta_0 = 0$
$\theta_1 = 0.5$



$\theta_0 = 1$
$\theta_1 = 0.5$

The whole




$\theta_0$ and $\theta_1$ values as choosen so that $h_\theta(x)$ is close to y for ~~the~~ training ex (x, y)

minimizing $\theta_0$ and $\theta_1$ considering that there are m training examples.

$$\underset{\theta_0,\theta_1}{\text{minimize}} = \frac{1}{2m} \sum_{i=1}^{m} \underbrace{\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}_{\text{sum of squared errors}}$$

$\frac{1}{2m}$ → avg

$h_\theta(x^{(i)}) \cdot \theta_0 + \theta_1 x^{(i)}$

re writting

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

$$\underset{\theta_0,\theta_1}{\text{minimize}} = \underbrace{J(\theta_0, \theta_1)}_{\text{cost function}}$$

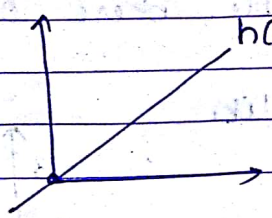
Cost function also called as "squared error function"

The goal is to minimize the cost function ie $J(\theta_0, \theta_1)$ where $\theta_0$ is the bias function.

→ Understanding Cost function using simplied hypothesis function

$h_\theta(x) = \theta_1 x$ → function of $x$

$\theta_0 = 0$



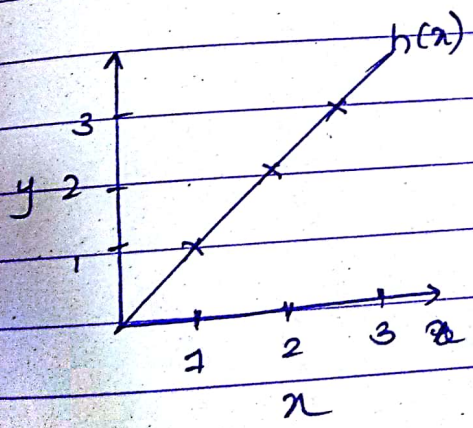

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$J(\theta_1)$ ↓ function of parameter $\theta_1$

$h_\theta(x^{(i)}) \leftarrow \theta_1(x^{(i)})$

→ $h_\theta(x)$

Assume $\theta_1(x) = 1$




$\theta_1 = 1$

$h_\theta(x^{(i)}) = y^{(i)}$

writing cost function

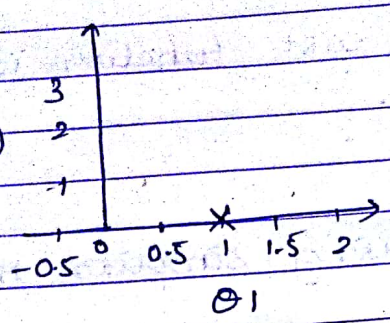
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

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

$$= \frac{1}{2m} (0^2 + 0^2 + 0^2)$$

$$J(1) = 0$$

plotting J(θ1)

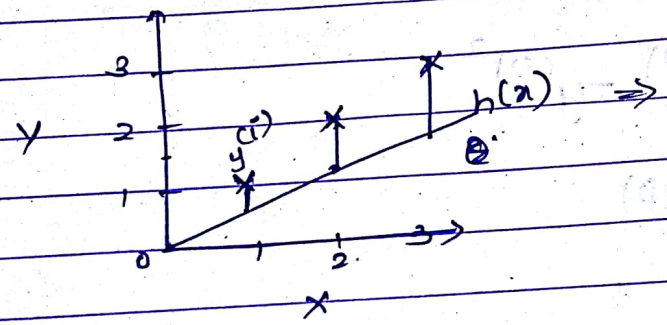


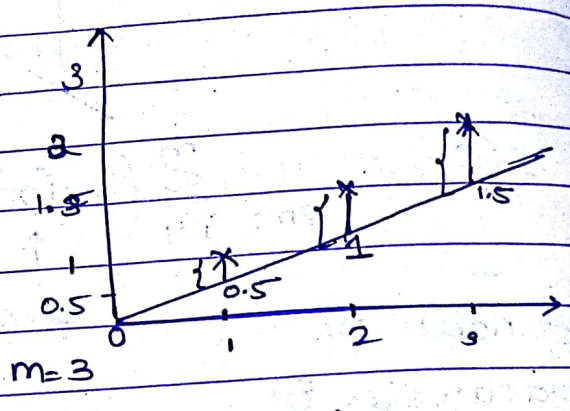

θ1 = 1

J(θ1) = 0

consider another example where θ1 = 0.5

∴ the hypothesis looks like



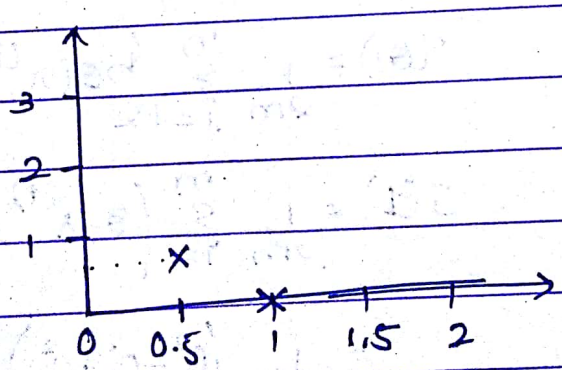




$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

$$= \frac{1}{6}\left[3.5\right]$$

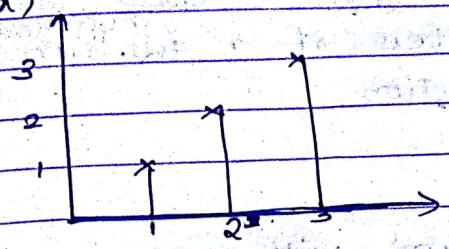
$J(0.5) \approx 0.58$

$\{ h_\theta(x^{(i)}) - y^{(i)}$



Consider $\theta_1 = 0$

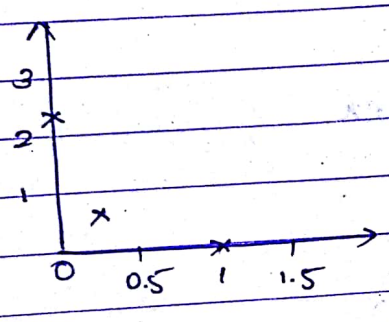
$\therefore h_\theta(x)$



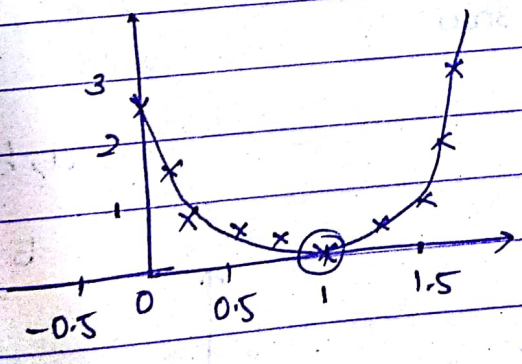

$$J(0) = \frac{1}{2m}\left[1^2 + 2^2 + 3^2\right.$$

$$= \frac{1}{6}\left[14\right]$$

$$= 2.3$$




the $J(\theta_1)$ can be computed for different values and that ~~graph~~ will look like




→ always the same for linear function

Since the goal ~~was~~ is to minimize the $J(\theta_1)$ the cost function is minimum at $\theta_1 = 1$

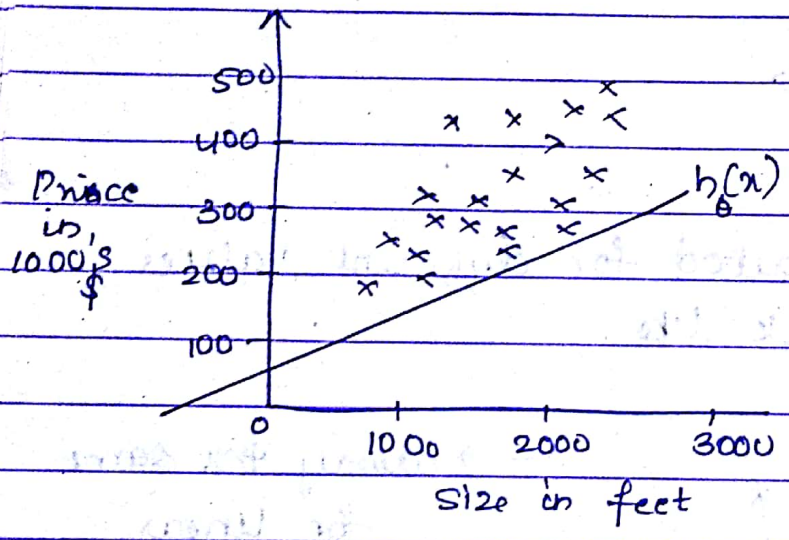
Gradient descent

## Understanding cost function with cost function having more than 1 $\theta$.

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

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

considering a training ex.

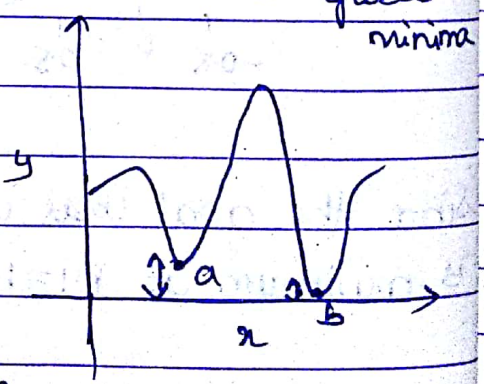



$h_\theta(x) = 50 + 0.06x$.

$\theta_0 = 50$

$\theta_1 = 0.06$

plotting corresponding $J(\theta_0, \theta_1)$




distance bet$^n$ x and point of with point b gives the $J(\theta_0)$ and $J(\theta_1)$ respectively

## Gradient Descent

Its an optimization algorithm used to find the values of parameters of a function/hypothesis that minimizes the cost function

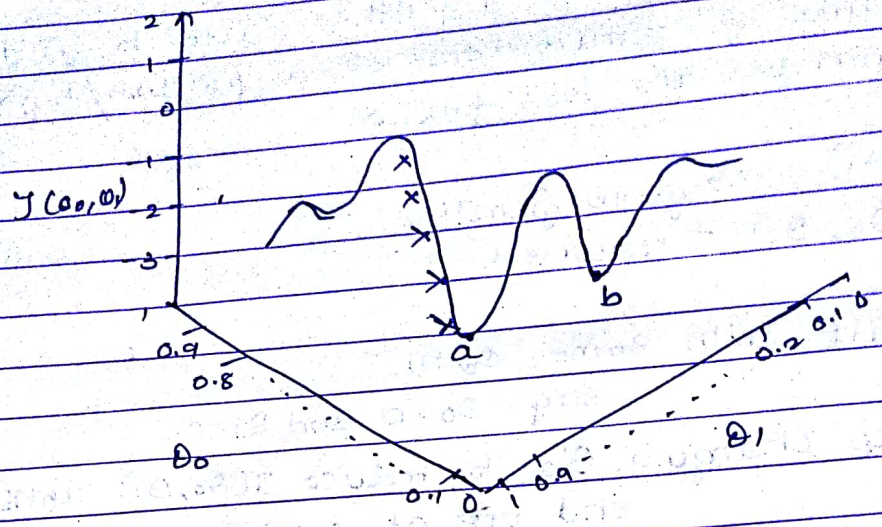
$J(\theta_0, \theta_1)$ → given function

$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ – required.

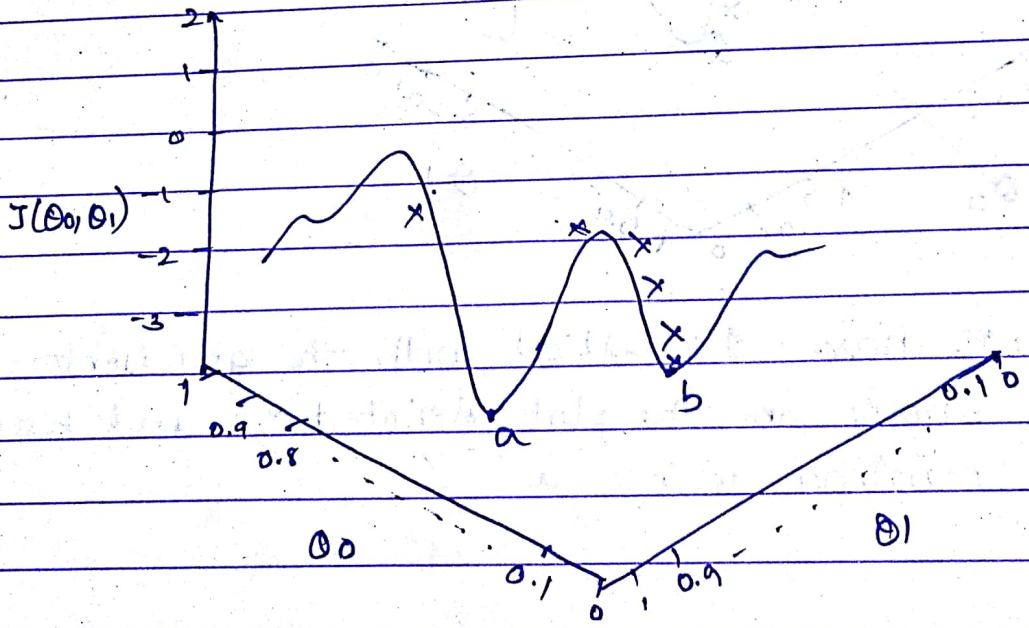
∴ start with some $\theta_0, \theta_1$

say $\theta_0 = 0$ and $\theta_1 = 0$

and keep changing $\theta$'s to reduce $J(\theta_0, \theta_1)$ until you end up at a minimum




Imagine we have started with a gradient descent as shown in the above plot (denoted x) and reach a local minima point 'a'




Consider another gradient descent slightly away from the previous one and taking certain steps ends up reaching at another local minima point 'b'

Gradient descent algorithm { := Assignment

Repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \{ j=0 \text{ and } j=1$$

}

// Simultaneous update (correct Method)

$$temp0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$temp1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$\theta_0 := temp0$

$\theta_1 := temp1$

$\alpha$ – learning rate / decides step size

$\alpha$ is large the diff bet$^n$ two $\theta$ values is more larger steps & vice versa

$\frac{\partial}{\partial}$

Incorrect ⟹ $temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\theta_0 := temp0$

$$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$\theta_1 := temp1$

* Action of derivative term on $\theta$

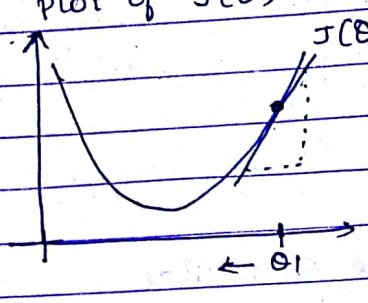
Considering the gradient descent algorithm.

$$\theta_j := \theta_j - \alpha \underbrace{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}_{\text{derivative}} \quad \{ j=0 \text{ \& } j=1$$

To understand the derivative considering a simple example

$$\min_{\theta_1} J(\theta_1)$$

Assuming plot of $J(\theta_1)$




$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

∴ the derivative indicate minimum

the tangent drawn to the J(θ1) function is a slope

the line drawn has +ve slope

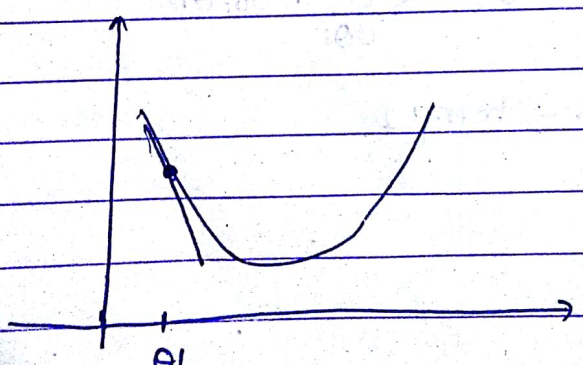
i.e $\frac{\partial}{\partial \theta_1} J(\theta) \geq 0$

∴ the update θ value will be

$\theta_1 := \theta_1 - \alpha$ (+ve number)

here the θ values are decreased to reach a local minima

* Considering another $\theta_1$ for the same function $J(\theta_1)$




the derivative term changes

the tangent drawn is -ve slope

i.e $\frac{d}{d\theta_1} J(\theta) \leq 0$

the update θ value will be

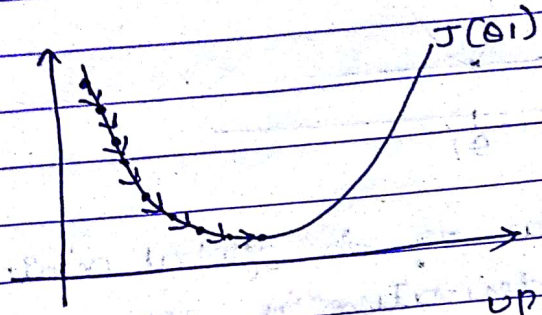
$\theta_1 := \theta_1 - \alpha$ (-ve number)

The θ value is increased to reach a local minima

Action of Learning Rate ($\alpha$) on the $\theta$.

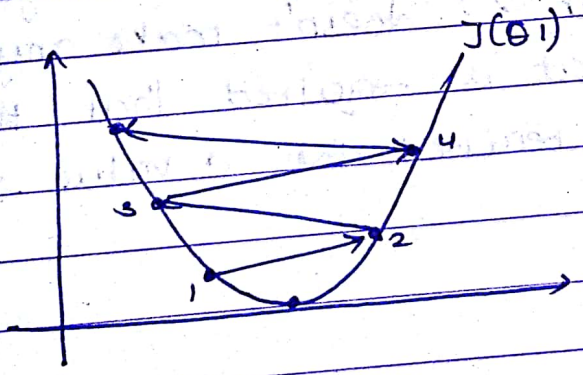
i) $\alpha$ is too small

$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$




If the $\alpha$ is too small the update derivative is multipled by a very small no. so the next point is very near to the previous step as shown in the plot This requires lots of step to reach the global minimum

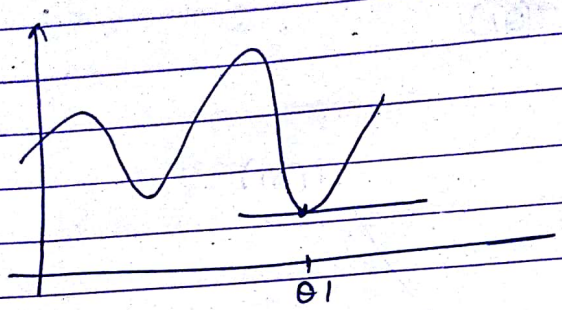
ii) $\alpha$ is too large




If the $\alpha$ is too large the gradient descent may even overshoot the global minima and fail to converge or even diverge

★

Suppose if the $\theta_1$ is already initialized at the global minima




If a slope is draw to the global minima the slope is 0 ie the derivative is zero as the equation infers.

$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{0}$$

$$\theta_1 := \theta_1$$

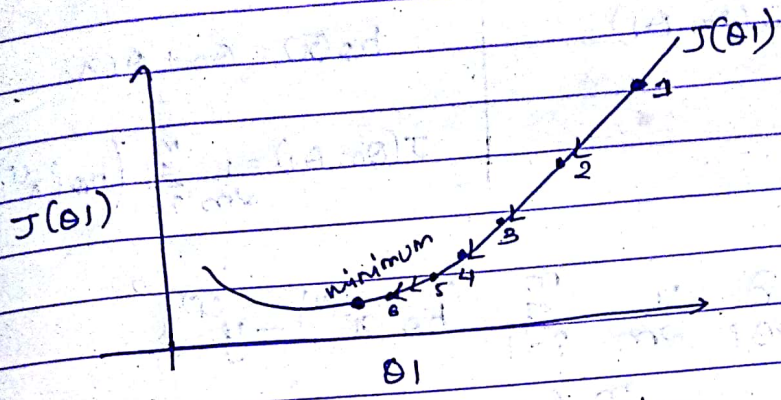
This infers that if we already at the global minima any step taken ($\alpha$) doesn't make any change This is actually what is required that ~~the~~ once the global minima is reached the $\theta$ value should remain unchanged.

* Gradient
even wit

$\theta_1 :=$

* Gradient descent can converge to a local minimum, even with the learning rate $\alpha$ fixed.

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$




As we approach a local minimum, gradient descent will automatically take smaller steps. So no need to decrease $\alpha$ over time, This infers that if we start $\theta$ value from point 1 in the plot & the derivative is larger while & $\alpha$ is constant throughout. So the next step point 2 is larger slope at point is less & derivative is less when multiplied with $\alpha$ it is a smaller value this continues * till the local minima is reached. Once the local minima is reached the slope is zero that is derivative is zero and the $\theta$ value is fixed. Hence change in $\alpha$ doesn't make any major difference.

* Applying gradient descent algorithm for linear Regression Model.

| Gradient descent Algorithm | Linear Regression Model |
|---|---|
| $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ | $h_\theta(x) = \theta_0 + \theta_1 x$ <br> $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$ |

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

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$$

$\theta_0$ $j = 0$ :

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

$\theta_1$ $j = 1$ :

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

∴ Gradient descent algorithm will be

repeat until convergence {

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

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

}

cost function for linear regression is always going to be a bowl shaped function (convex function)

Batch Gradient Descent :-

↓

Each step of gradient descent uses all of the training examples.

## * Linear Regression with multiple variables

Multiple features - n

Here the ~~features~~ notation may be

n - no. of features

$x^{(i)}$ - input (feature) of $i^{th}$ training example.

$x_j^{(i)}$ - value of feature j in $i^{th}$ " "

ex :

| Size (feet$^2$) | No. of bedroom | No. of floors | Age of home | Price. |
|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
| 2104 | 5 | 1 | 45 | 460 |
| 1416 | 3 | 2 | 40 | 232 |
| 1534 | 3 | 2 | 30 | 315 |
| 852 | 2 | 1 | 36 | 178 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_3^{(2)} = 2$$

previously hypothesis was

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

now for multiple variates

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

~~for the ex~~

in general

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$x_0 = 1$$

~~x0~~ parameter vector

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$\therefore h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots \theta_n x_n$$

$$= \theta^T x.$$

## * Gradient descent for Multiple variants

$h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \cdots \theta_n x_n$

$\theta_0, \theta_1, \cdots \theta_n$ - parameters $\Rightarrow$ n+1 divisional vector

Cost function

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

$\Downarrow$

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

## Gradient descent

Repeat {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0 \cdots \theta_n)$

}

Simultaneously update for every $j = 0 \cdots n$

OR

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ ← learning rate.

$\Rightarrow$ gradient descent changes as follows for linear regression for multiple variate

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

$\frac{\partial}{\partial \theta_j} J(\theta)$

Considering more than 2 features then we have the following simultaneous update for the gradient descent

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

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_2^{(i)}$$

⋮

## Feature Scaling

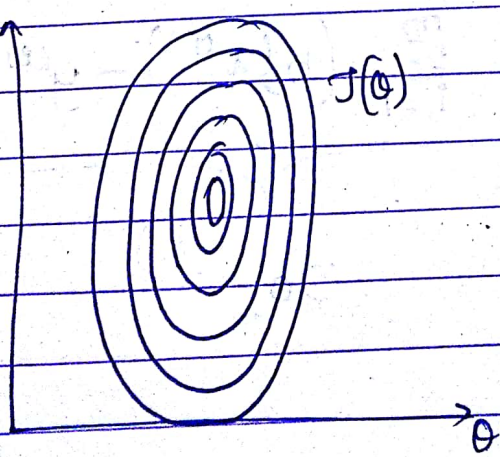
If there are multiple feature and are on similar scale then feature scaling can be used to arrive at a gradient descent.

Consider an example with two feature

$x_1$ = size (0-2000 feet$^2$)

$x_2$ = number of bedrooms (1-5)

plotting for the above features ignoring $\theta_0$ we have (plotting contours)




As the contours are steep the gradient descent may end up taking lot of time to reach the global minima.

In these se
scale the f
redefining
$x_1$ = si

$x_2$ = n

the countor
and when
direct path
rather than

Feature sca
in the
this
value li

Normaliza

∴ In these setting the useful thing to do is scale the features that is redefining the $x_1$ and $x_2$ as

$x_1 = \frac{\text{Size (feet}^2\text{)}}{2000}$

$x_2 = \frac{\text{number of bedrooms}}{5}$ then

the countors may look much more like circles and when gradient descent is run over this a direct path is found to get the global minima rather than complicating the path

Feature scaling is getting feature approximately in the range of $-1 \leq x_i \leq 1$

this can also be some relatively closes value like $0 \leq x_1 \leq 3$ ✓

$-2 \leq x_2 \leq 0.5$ ✓

$-0.0001 \leq x_4 \leq 0.0001$ ✗

$-100 \leq x_3 \leq 100$ ✗

Normalization:

To bring data to a same scale

{ # no. of

## Mean normalization:

Replace $x_i$ with $x_i - \mu_i$ to make feature have approximately zero mean (doesn't apply to $x_0 = 1$)

Eg: $x_1 = \frac{\text{Size} - 1000}{2000}$

$x_2 = \frac{\#\text{bedrooms} - 2}{5}$

$-0.5 \leq x_1 \leq 0.5$

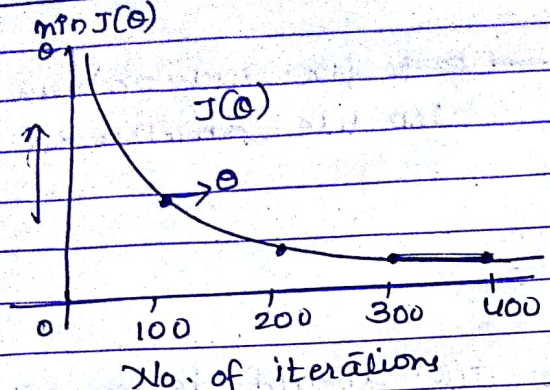
$-0.5 \leq x_2 \leq 0.5$

$x_1 = \frac{x_1 - \mu_1}{s_1}$ → avg value of $x$ in training set.

$s_1$ ⟶ range

$x_2 = \frac{x_2 - \mu_2}{s_2}$

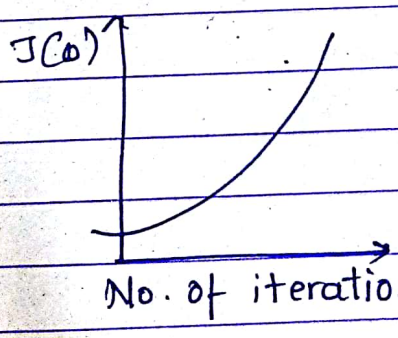
## Gradient descent II learning rate :



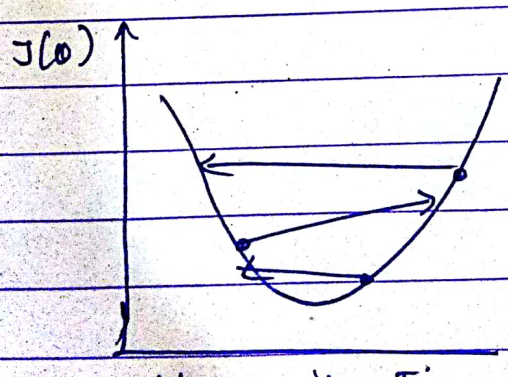

This graph shows the value of ~~θ~~ cost function after certain iterations and J(θ) should decrease after iterations

It can be noticed that from 300 - 400 there is no much decrease in the gradient this implies that there is no ~~shan~~ major change in the θ and ~~this~~ at ~~may~~ these iteration there may be convergence of the gradient descent.

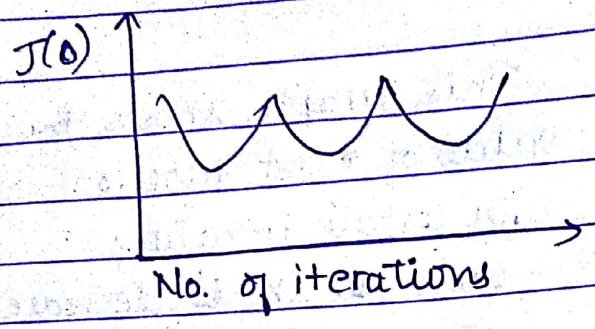
## *Making sure the gradient descent is working properly




here J(θ) is increasing this notifies that the gradient descent is not working
∴ α should be smaller




if α is too large the gradient descent keeps on over shooting.
∴ use ~~a~~ smaller α in such cases.

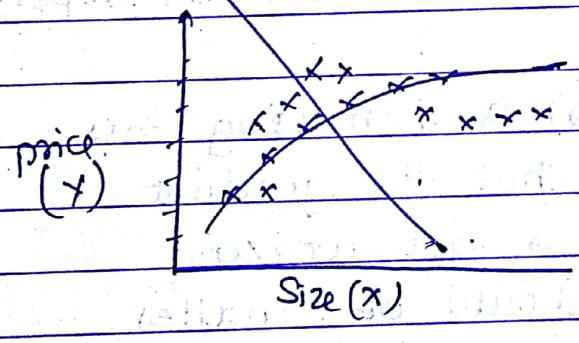



→ to fix such problem also use smaller α

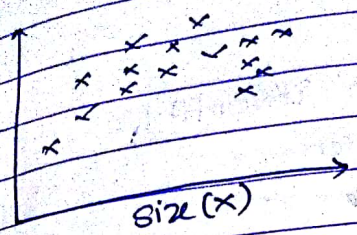
inference from the above
- For sufficiently small α J(θ) should decrease on every iteration
- But if α is too small, gradient descent can be slow to converge

## Features & Polynomial Regression

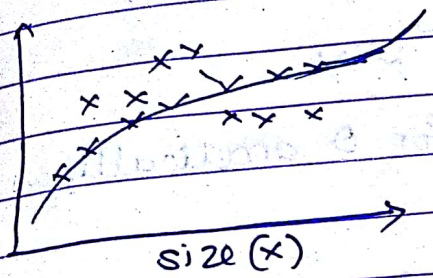
Consider a ex



# Features and polynomial regression




A straight line cannot fit into the data and a quadratic function such as $\theta_0 + \theta_1 x + \theta_2 x^2$ but eventually goes down therefore choosing 3rd order function



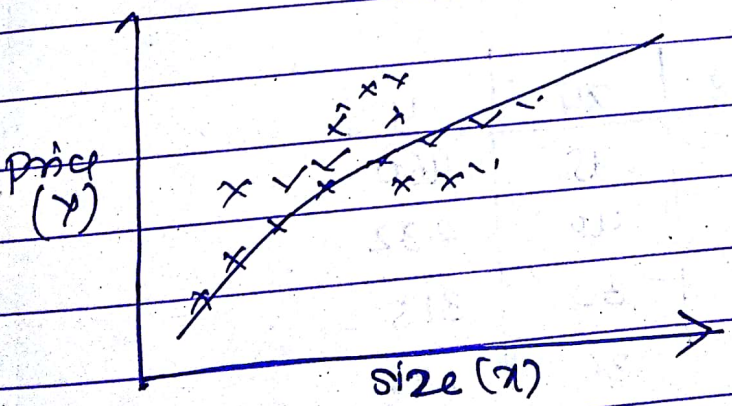

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

$$h_\theta(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3$$

$x_1$ $x_2$ $x_3$

Another choice to fit the data is

$$h_\theta(x) = \theta_0 + \theta_1 \text{size} + \theta_2 \sqrt{\text{size}}$$



Normal Equation: Non Invertibility

$\theta = (X^T X)^{-1} X^T y$

$X^T X$ is some times non invertible
to avoid or overcome them
→ Redundant features
→ If there many features ($m < n$)
delete some features or regularize

Normal Equation : $\theta = (X^T X)^{-1} X^T y$
gives a method to solve for $\theta$ analytically

consider a ex.

$J(\theta) = a\theta^2 + b\theta + c \Rightarrow J(\theta_0, \theta_1 \ldots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

taking derivative of the cost function and solving for the derivative we arrive at $\theta$ values that take to global minima

Ex:-

$m = 4$

| $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|---|
| 1 | 2104 | 5 | 1 | 45 | 460 |
| 1 | 1416 | 3 | 2 | 40 | 232 |
| 1 | 1534 | 3 | 2 | 30 | 315 |
| 1 | 852 | 2 | 1 | 36 | 178 |

↑ adding $n+1^{th}$ training sample.

forming matrix for x and y and solving for $\theta = (X^T X)^{-1} X^T y$ give the $\theta$ value that ends up reaching the global minima.

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad Y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$X = m \times (n+1)$

$Y$ → n dimension

Generalizing the equations

m examples $(x^{(1)}, y^{(1)}), \ldots (x^{(m)}, y^{(m)})$

n features

$$x^{(i)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(2)} \\ \vdots \\ x_n^{m} \end{bmatrix}$$

$$(X) \text{ design matrix} = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}$$

$X = m \times (n+1)$

LP-5

| Gradient Descent | Normal Equation |
| --- | --- |
| → disadvt: • need to choose $\alpha$<br>Needs many iteration | adv: No need to choose $\alpha$<br>Don't need to iterate |
| → advantage:<br>: work well even when n is large | disadvantage:<br>• Need to compute $(X^T X)^{-1}$<br>• slow if n is large |

## Logistic Regression

Earlier classification (linear classification) the prediction y may have two values i.e $y \in \{0, 1\}$

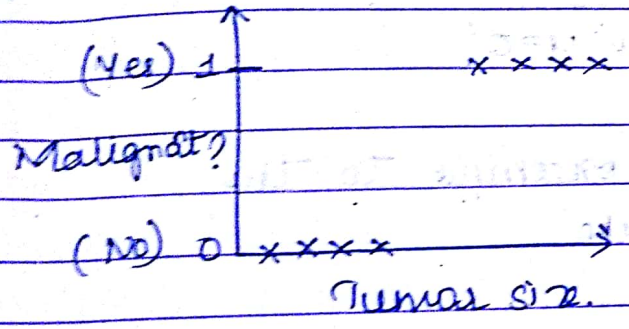
0 - Negative class (ex: benign tumor)

1 - +ve class (ex: malignant tumor)

(Logistic Regression has binary classification and multivariate classification as well)

drawback of linear Regression

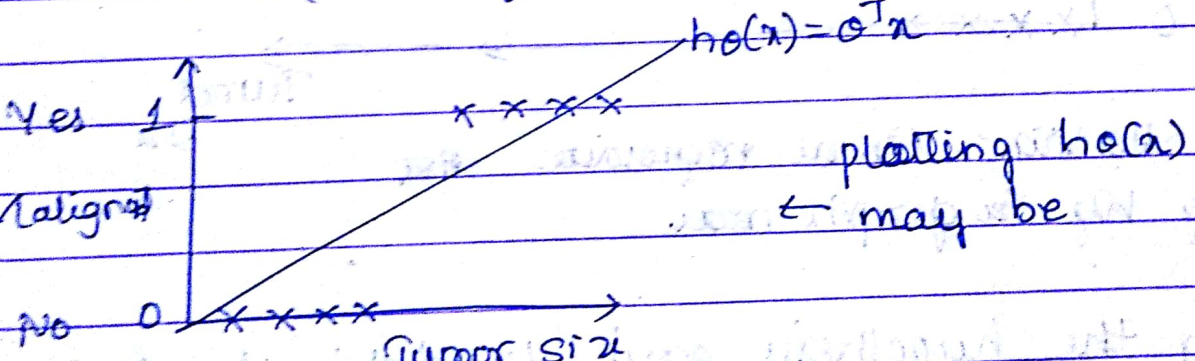
Ex:



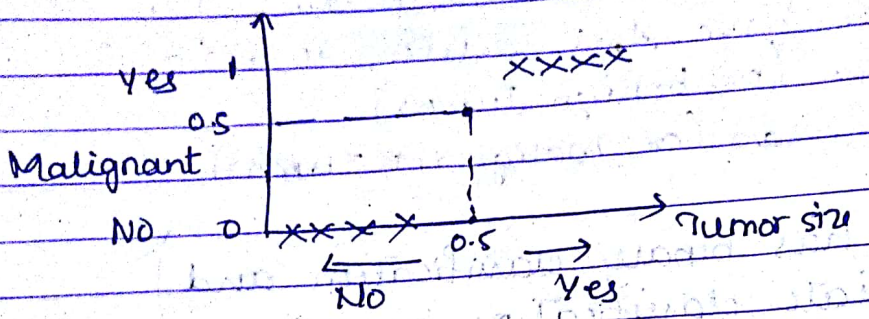

Considering a hypothesis function (used earlier)

$h_\theta(x) = \theta^T x$

(linear function)



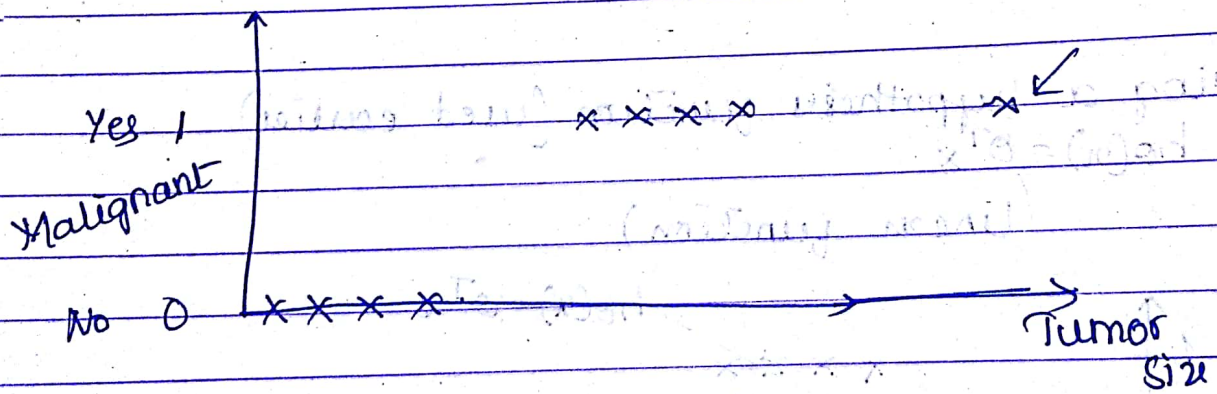
Considering threshold.




Threshold classification output $h_\theta(x)$ @ 0.5

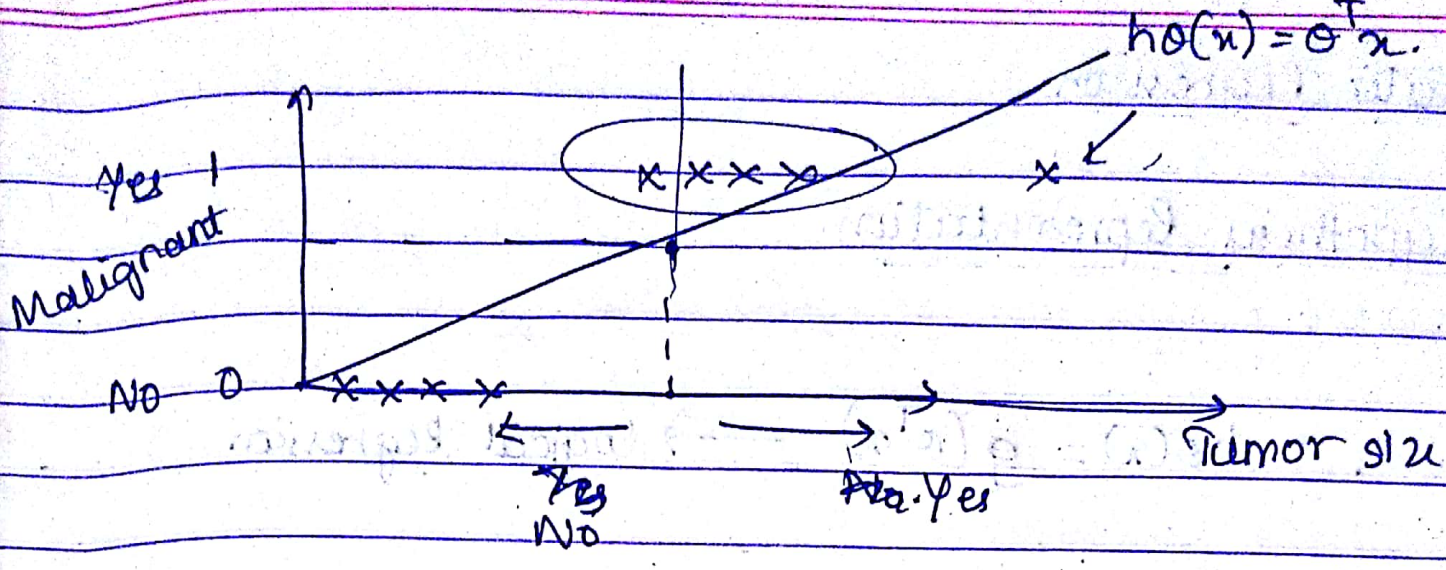
if $h_\theta(x) \geq 0.5$ predict '$y = 1$'
if $h_\theta(x) < 0.5$ predict '$y = 0$'

→ Adding an extra training example to the data the graph may look like




After running linear regression the fitting for the graph may

fitting the hypothesis and threshold classification to this data set may vary the whole prediction




The training sets on Yes(1) line are all the ex- with Malignant tumor but when the linear regression is run on this data set the ~~bag~~ fitting of hypothesis changes and the threshold classification also changes ~~th~~ as it can be seen from the above diagrams that few of the malignat tumor data ex ~~wks~~ lie on the ~~the~~ either side ie begnin tumor. ∴ linear Regression is not prefered for classification problem.

* The y takes values 0 and 1
but $h_\theta(x)$ can be $>1$ or $<0$
this is constraint for fitting the curve.
Where as ~~by~~ ~~Eas~~ ~~bae~~ $h_\theta(x)$ is 0 or 1 is
logistic Regression

# Logistic Regression

## Hypothesis Representation

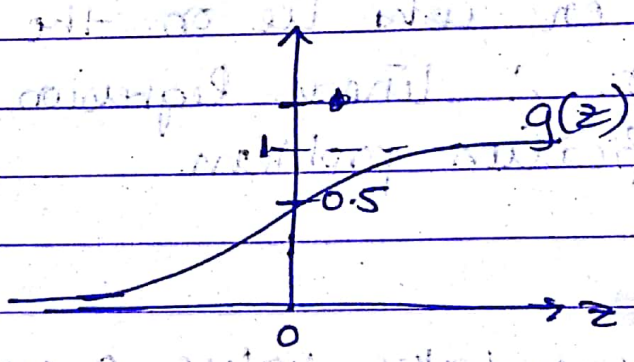
$h_\theta(x) = g(\theta^T x)$ ⟶ Logical Regression

for logistic.

$h_\theta(x) = g(\theta^T x)$

$g(z) = \frac{1}{1+e^{-z}}$ — sigmoid function

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



## Interpretation of Hypothesis Output

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

ex: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumor size} \end{bmatrix}$

$h_\theta(x) = 0.7$.

Suppose the patient have some tumor size and their feature hypothesis is 0.7. This tells that for the patient with features $x$ the probability that the patient is having tumor is 70% (0.7)

Mathematically writing.

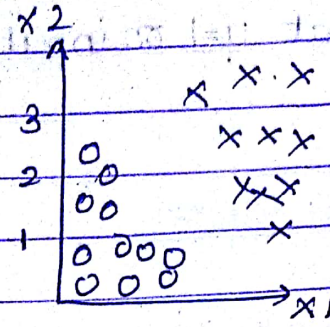
$h_\theta(x) = P(y=1|x;\theta)$ "probability that $y=1$, give $x$, parameterized by $\theta$"

y can take either 0 or 1.

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

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

# Decision Boundary




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

$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$\theta_0 = -3$

$\theta_1 = 1$

$\theta_2 = 1$

Predict $y = 1$ if $-3 + x_1 + x_2 \geq 0$

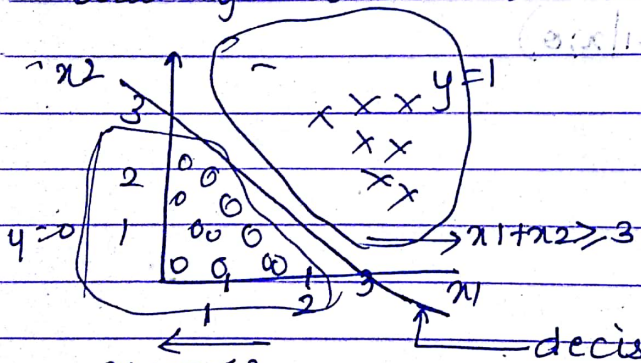
or

$x_1 + x_2 \geq 3$

considering $x_1 + x_2 = 3$

this is a straight line

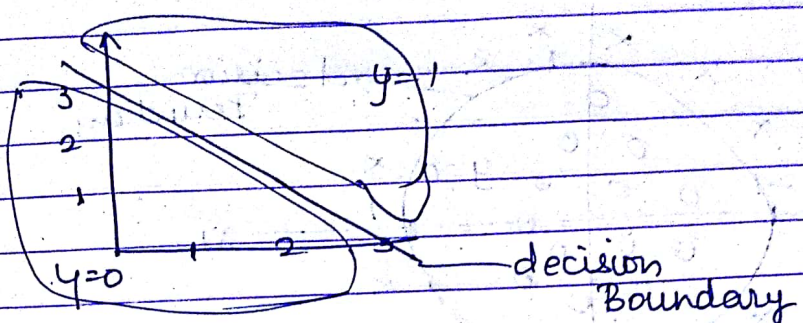
drawing this on the data set



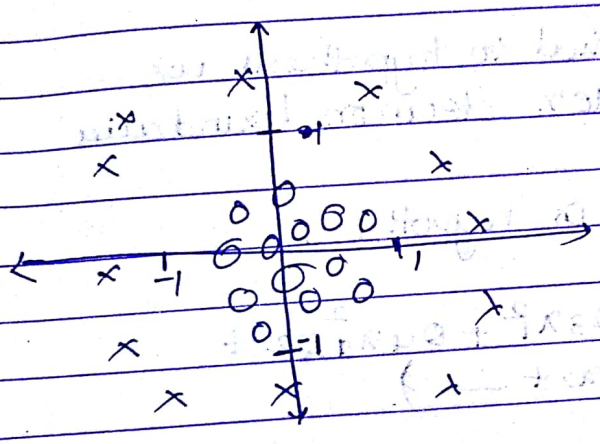

$x_1 + x_1 < 3 \Rightarrow y = 0$

Decision boundary is the line that seperates where the hypothesis predicts y is exactly equal to 0 and exactly equal to 1

Decision boundary is the property of hypothesis and the parameter and not the property of the data set




Non linear decision Boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

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

$\theta_0 = -1$

$\theta_1 = 0$

$\theta_2 = 0$

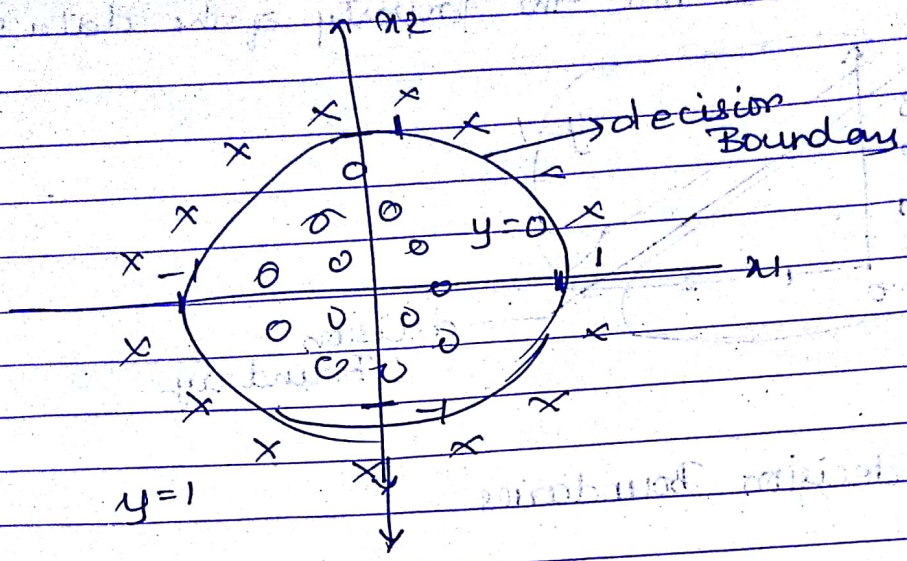
$\theta_3 = 1$

$\theta_4 = 1$

Predict $y=1$ if $-1 + x_1^2 + x_2^2 \geq 0$

$$x1_2^2 + x_2^2 \geq 1$$

$x_1^2 + x_2^2 = 1$ ⟶ equation of circle.



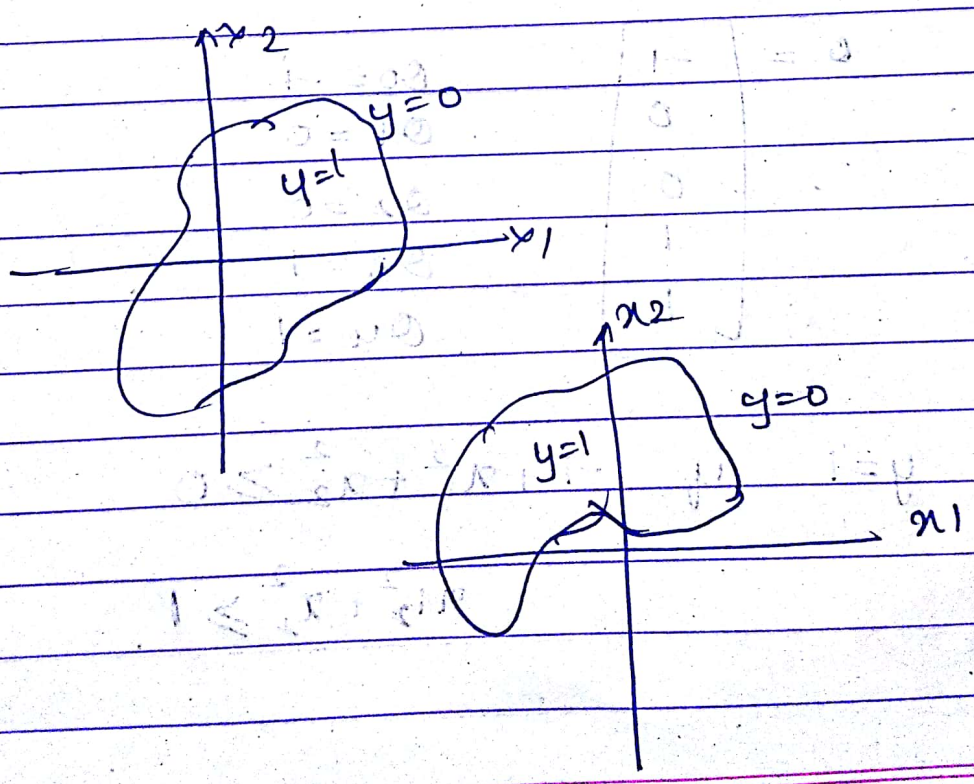

adding complex polynomial to hypothesis we can arrive at more complex decision boundaries

Considering more parameter in hypothesis

ex:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 \cdot x_2 + \theta_5 x_1^2 + x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

The decision boundary may end up looking like




Fitting Data

ex:

Training se

n example

$h_\theta(x)$

choos

# Fitting Data to Logistic Regression

Ex:

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(m)})\}$

m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad x_0 = y \quad y \in \{0, 1\}$$

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

choosing parameter $\theta$ ie (cost function)

Cost function

Linear Regression →

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

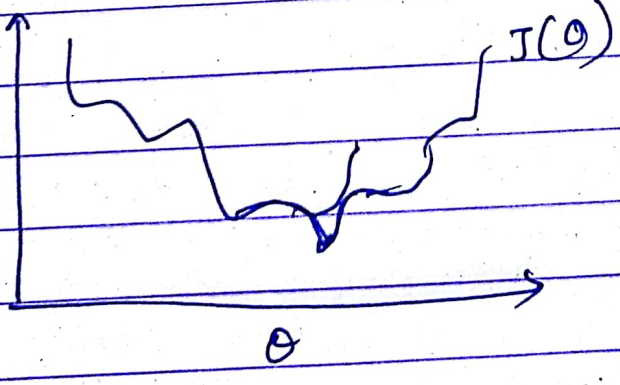
$$\text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right) = \frac{1}{2} \left( h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Considering same cost function for logistic Regression

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

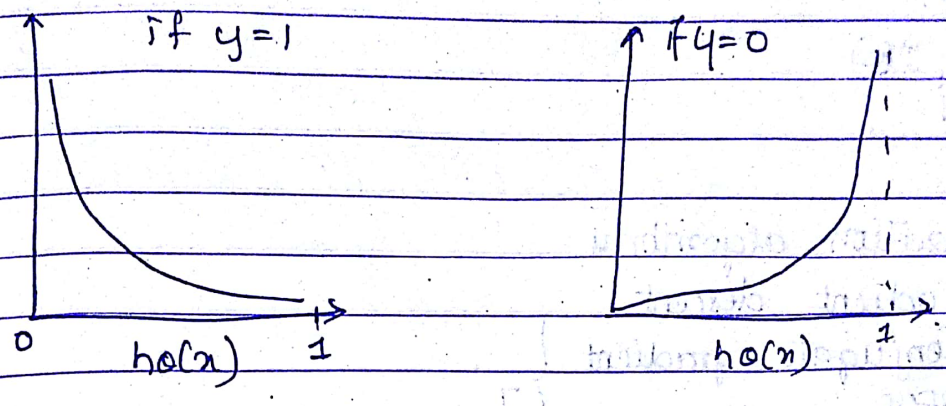
(non linear function)

The resulting gradient may look like a non-convex curve



## Cost function for logistic regression

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




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

Can also be written as

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

To fit parameters:

$\min_\theta J(\theta)$ ⟶ get $\theta$

Repeat {

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

(simultaneously update all $\theta_j$)

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

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

## Optimization Algorithms

given $\theta$ we code that can compute

$J(\theta)$

$\frac{\partial}{\partial \theta_j} J(\theta)$

Optimization algorithms
- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages
- No need to manually pick $\alpha$
- Often faster than gradient descent

disadvantages
- More complex.

Multi class Classification:

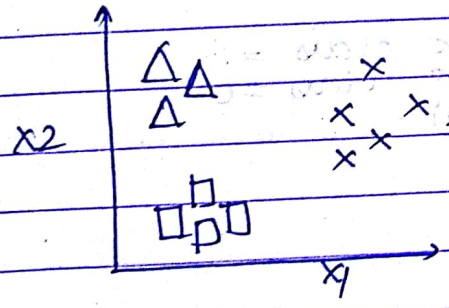
Ex:

Email foldering / tagging: Work, Friends, Family, Hobby
y=1, y=2, y=3, y=4

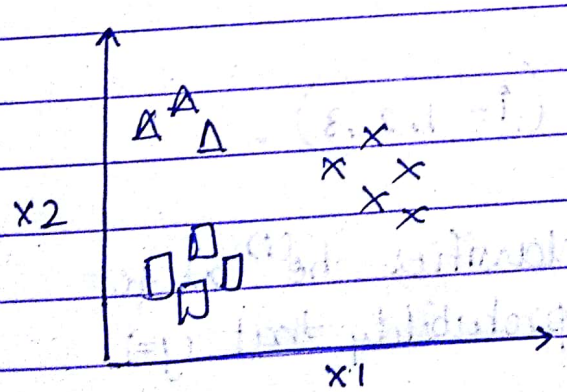
Medical diagrams: Not ill, Cold, Flu
y=1, y=2, y=3

Weather: Sunny, Cloudy, Rain, Snow
y=1, y=2, y=3, y=4

Ex for data set

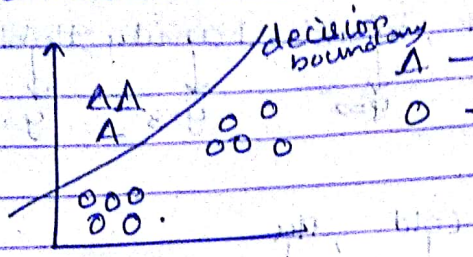



One-vs-all (one-vs-rest)




Class 1 = Δ
2 = □
3 x

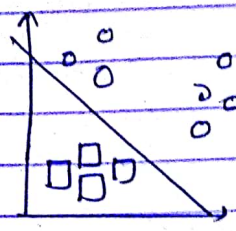
Considering class we create new training set where were class 2 & 3 are assigned to -ve class and class 1 is +ve class




Δ — +ve class = 1
O — -ve class = 0

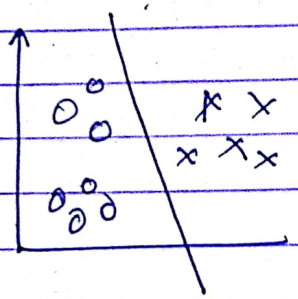
$h_\theta^{(1)}(x)$.

Ny for other class



□ +ve class = 1
O -ve class = 0
$h_\theta^{(2)}(x)$

Ny for 3rd class



× +ve class = 1
O -ve class = 0
$h_\theta^{(3)}(x)$

to summarize we fit 3 classifier then find the probability that $y=1$ give for any $x$ values

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

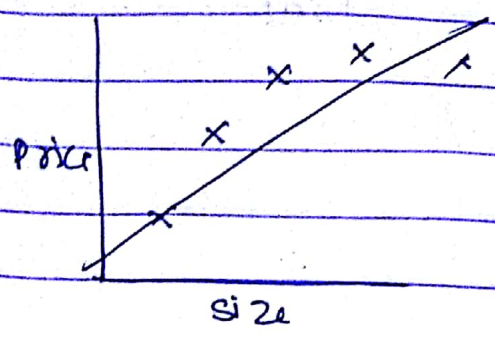
Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class $i$ to predict the probability that $y=i$.

On a new input $x$ to make a prediction, pick the class $i$ that maximizes

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

# The Problem of overfitting

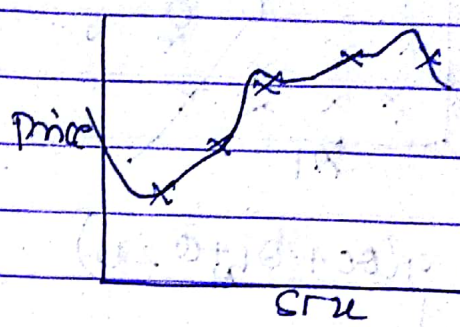
Linear Regression
Ex: (Housing Prices)




$\theta_0 + \theta_1 x$

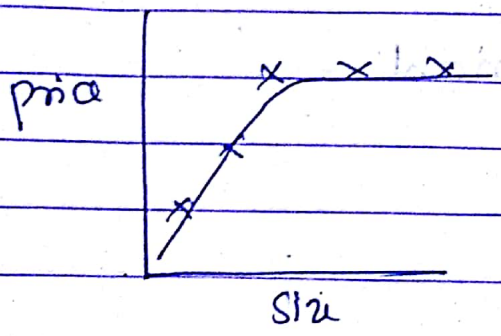
Underfit

High bias




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

Overfit High variance

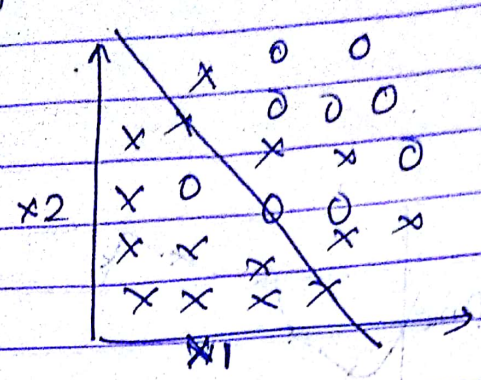



$\theta_0 + \theta_1 x + \theta_2 x^2$

just right

Overfitting: if we have too many features, the learned hypothesis may fit the training set very well but fail to generalize to new examples

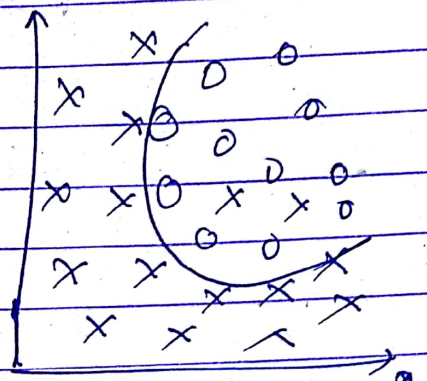
Logistic Regression




Under fit/high bias

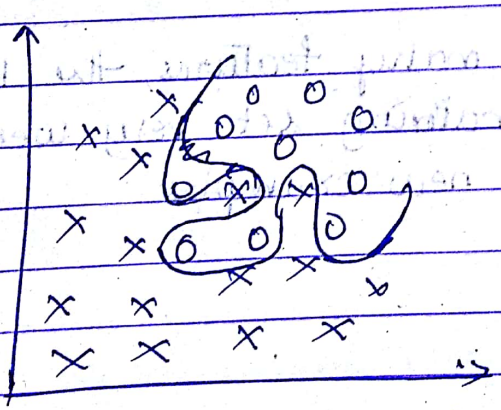
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

↓ sigmoid function



just right

$g(\theta_0 + \theta_1 x + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_3)$



Overfitting
high variance

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_4^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \cdots)$

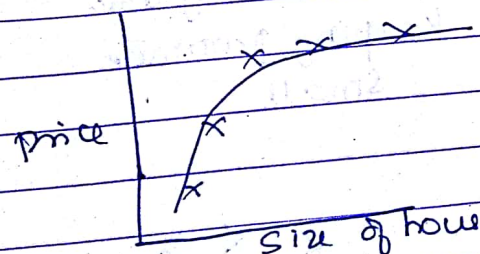
## Addressing Overfitting:-

1. Reduce Number of feature
   - Manually select which features to keep
   - Model selection algorithm

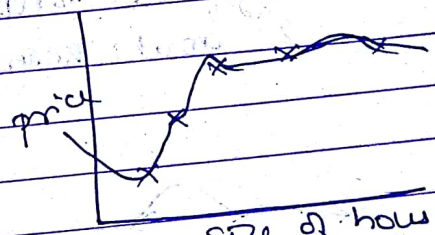
   disadvantage: ignoring of few important features

2. Regularization
   - Keep all the features but reduce magnitude/values of parameters $\theta_j$
   - Works well when we have a lot of features each of which contributes a bit to predicting y

### Regularization




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




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

Suppose $\theta_3$ & $\theta_4$ are made very small.

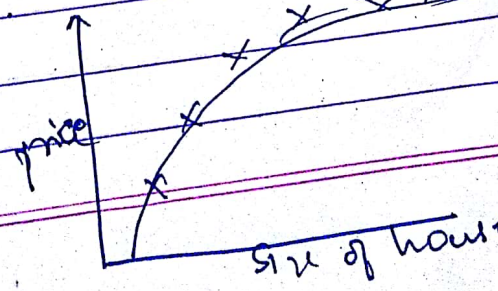
$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + 1000\theta_3^2 + 10000\theta_4^2$$

some Random huge value.

$\theta_3 \approx 0$ and $\theta_4 \approx 0$.

∴ we end up having quadratic eqn with negligible value of $\theta_3$ & $\theta_4$.

∴



Small value for parameter
- "simpler" hypothesis
- less prone to overfitting

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

↳ Regularized cost function

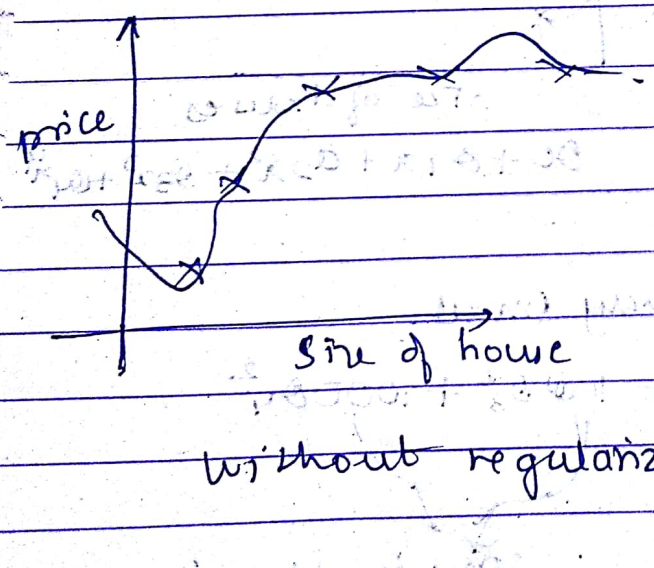
Regularization term

Regularization Parameter

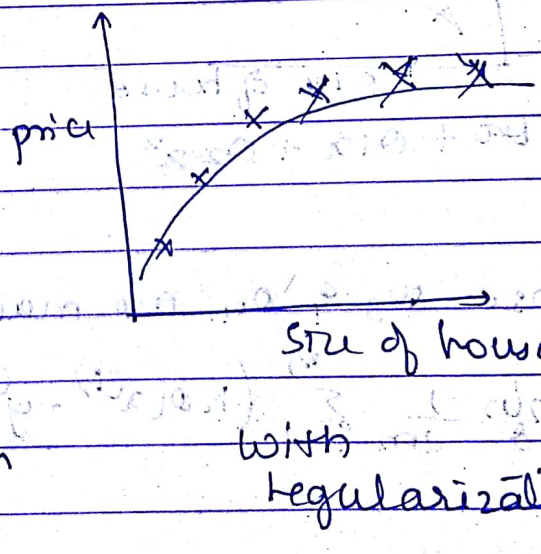
Controls trade of bet^n two goals

1st goal: train data fit all training data

2nd goal: regularize keeping parameter small.




without regularization




with regularization

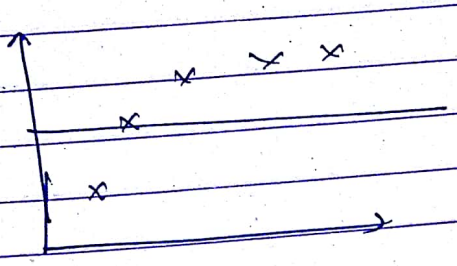
if $\lambda$ is set too large then

$\therefore$ ~~to 0~~

$\therefore \quad h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \dots$

($\theta_1, \theta_2, \theta_3, \dots$ become $\cong 0$)

$\therefore h_\theta(x) = \theta_0.$




Regularized linear regression

gradient descent

Repeat {

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

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

}

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

$$1 - \alpha \frac{\lambda}{m} < 1$$