Date: / /
Page No.

INTRODUCTION.....

* What is Machine Learning?

Informally, machine learning is the field of study that gives computers, the ability to learn without being explicitly programmed.
- By Arthur Samuel.

A more modern definition by Tom Mitchell:

A computer program is said to 'learn' from experience E with respect to some class of task T and performance measure P, if its performance P at task(s) in T improves with experience E.

Eg If you develop a program that is able to play chess and learn the moves that lead to victory, then the program is said to be using ML to improve its P at playing chess (T) and winning over time as it plays more and more game (E).

Date: / /
Page No.

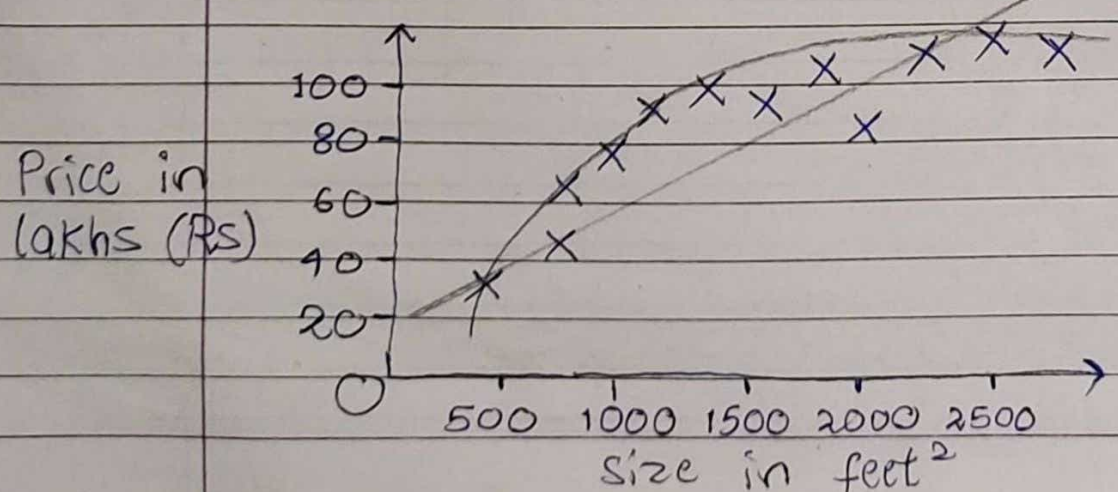
* TYPES OF MACHINE LEARNING ALGOS.

(1) SUPERVISED ML ALGORITHMS

If you want to predict the price of houses in a given neigbourhood given that you have some pre-collected and labelled data about the pricing of the houses for / V.S. the area of the house you can establish a relation / function between the area and the price of the houses for the given data and then use this relation to predict prices for some more houses whose area is given.

The above is an example of supervised Learning.

The fact that we gave the algorithm some right answers. That is we gave it a data set of houses in which for every example in this data set we told it what is the right price for which the house got sold and the task of the algorithm was to just produce more of these right answers



Date: / /
Page No.

This particular problem is a Regression Problem which is a type of supervised Learning.

→ So, Supervised Learning is the machine learning task of learning a function that maps an input to an output based on example input - output pairs which are the 'labelled training data' consisting of a set of correct input - output pairs. Thus, the model can be trained and then used to predict the output for given unlabelled data / input.

(i) Regression:

Regression is a statistical method relating two or more quantities / mapping quantities out of which, one is a 'Dependent variable' and the other is the 'Independent Variable'. By mapping them, it creates a continuous valued output between the variables.

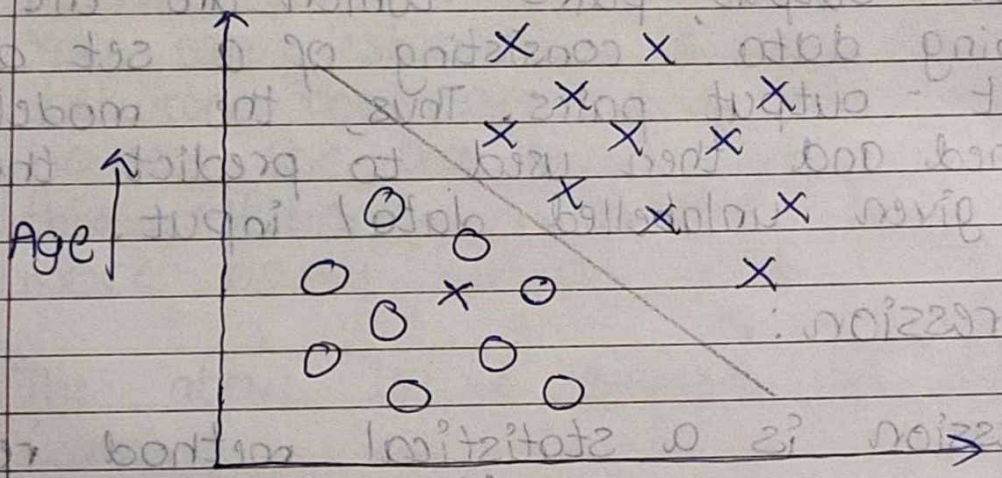
(ii) Classification:

The Classification type of problem is another statistical measure that helps in classifying a given input data belonging to a given category / class. So, we might be trying to predict a discrete value output say 0 or 1 for some given data. This can only be done if the model has been fed / trained with training data examples i.e. 'correct data'

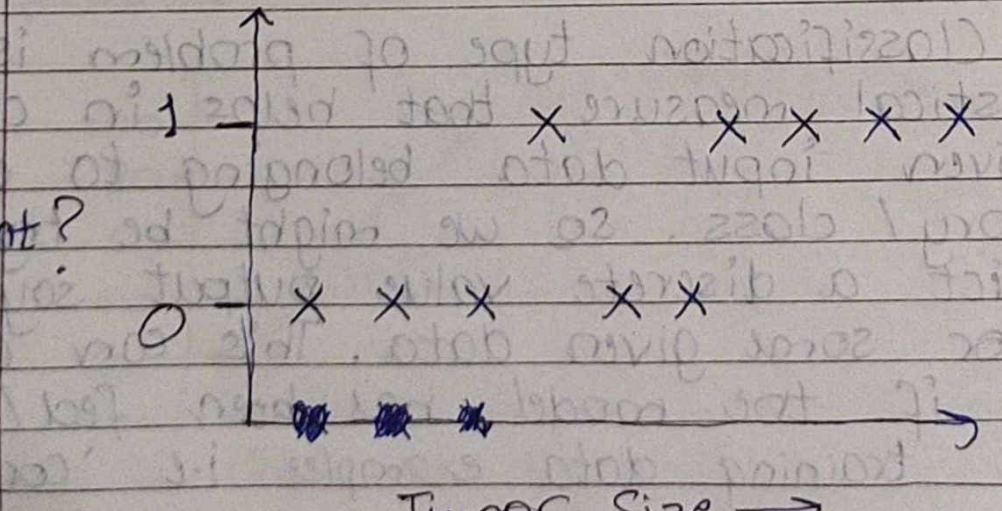
Date: / /
Page No.

A classification problem can have more than 2 classes depending on the problem. So, we can take constants like 0, 1, 2, 3, etc depicting the different categories of our output.

If there are 2 parameters and two classes, say as shown below:



The learning algorithm might try to fit a line that tries to separate out the malignant tumors from the benign ones and this might help us in deciding whether a tumor is benign or malignant.



Date: / /
Page No.

(2) Unsupervised ML Algorithms

In Unsupervised learning we have little/no labelled data i.e. the correct input/output pairs and all we have is an unlabelled data set with some possible structure in it. The Unsupervised Learning Algorithm has to find a structure in the data set which can be done by quite a few methods. One of such is Clustering.

So, given a data set the ML Algo. might 'sit and decide' that the data lives in two or more clusters. This is the Clustering Algorithm.

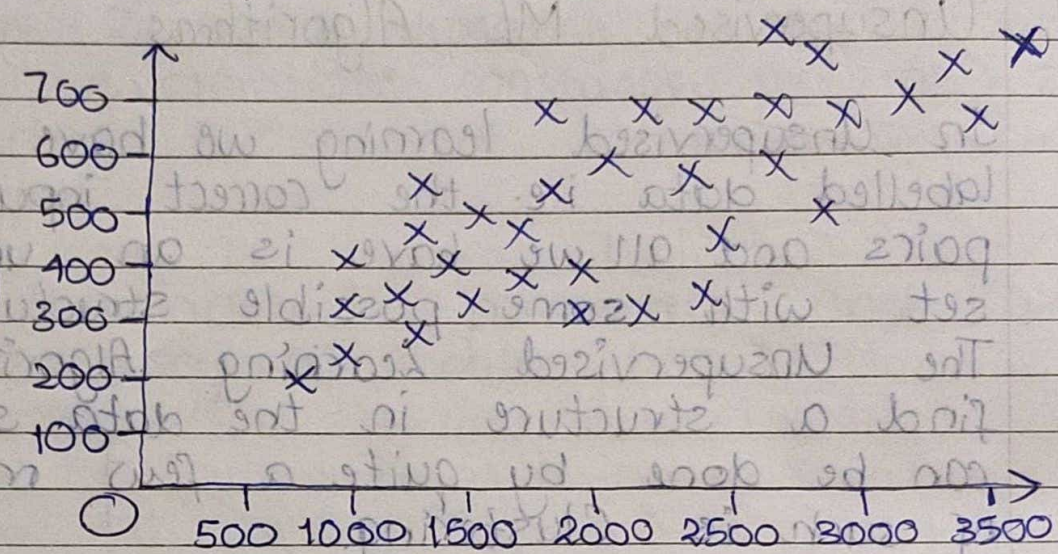
Eg. A company might try to use Clustering Algo to organize its customers into clusters based on some Natural Similarities that the algo finds in them. This will group customers into different segments and target them accordingly.

Eg Cocktail Party Problem

Here, 2 or more sources of sound play together and then the Unsupervised Algo. has to separate them into clusters by finding some natural similarities.

Date: / /
Page No.

* LINEAR REGRESSION WITH ONE VARIABLE



Example of Regression / Supervised Learning problem.

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1536	315
852	178
...	...

Notation

m = Total no. of training examples
x's = "input" features
y's = "output" / "target" feature

Date: / /
Page No.

So, (x_1, y_1), (x_2, y_2), ... are the training examples which can be generalised to $(x^{(i)}, y^{(i)})$ for the i^{th} term

$\therefore$ $x^{(1)} = 2104$ $\quad y^{(1)} = 460$
$x^{(2)} = 1416$ $\quad y^{(2)} = 232$

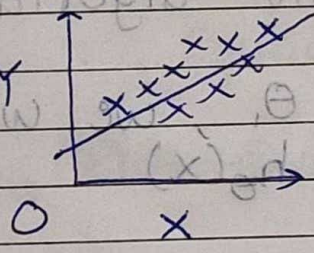
This training set is used to train the learning algorithm and it in turn develops a hypothesis (h) which is a function and it maps the x's to the Y's.

- How to Represent h?

$\underbrace{h_\theta(x)} = \theta_0 + \theta_1 x$

Shorthand: $h(x)$

$\Rightarrow$



So, this is linear regression with one variable also called univariate linear regression.

Date: / /
Page No.

Training Set:

Size in feet^2 (x)	Price ($) in 1000's ($y$)
2104	460
1416	232
1536	315
852	178
...	...

$m = 47.$

Hypothesis:

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

Here, θ's are the parameters which need to be determined by the algorithm.

Depending on θ_0 and θ_1, we will get different curves for $h_\theta(x)$.

Our aim is to come up with parameters θ_0 & θ_1 so that our $h_\theta(x)$ is close enough to our Y for the training examples $(x^{(i)}, y^{(i)})$

$\therefore$ We want to minimize the difference

Date: / /
Page No.

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

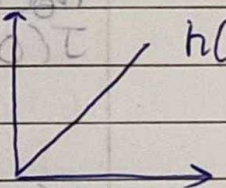
In words: Minimize the squared difference (squared distance) between the hypothesis and the output values y for all the given training examples from $i=1$ to $i=m$, average it out by dividing this by m and for the ease of calculation, divide by 2 again.

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

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

Squared error function

EXAMPLE $h_\theta(x) = \theta_1 x \quad (\theta_0 = 0)$

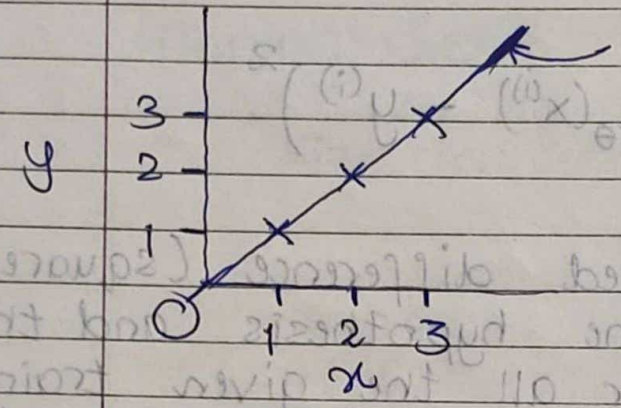


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$\underset{\theta_1}{\text{minimize}} \; J(\theta_1)$$

where $h_\theta(x^{(i)})$ is a function of x

and $J(\theta_1)$ is a function of θ_1.

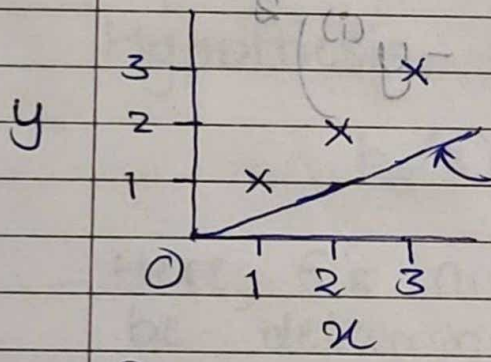


The points are $y^{(i)}$

Let $\theta_1 = 1$

$\Rightarrow h_\theta(x) = x$

$\Rightarrow J(\theta_1) = J(1) = \frac{1}{6}\{0^2 + 0^2 + 0^2\} = 0$



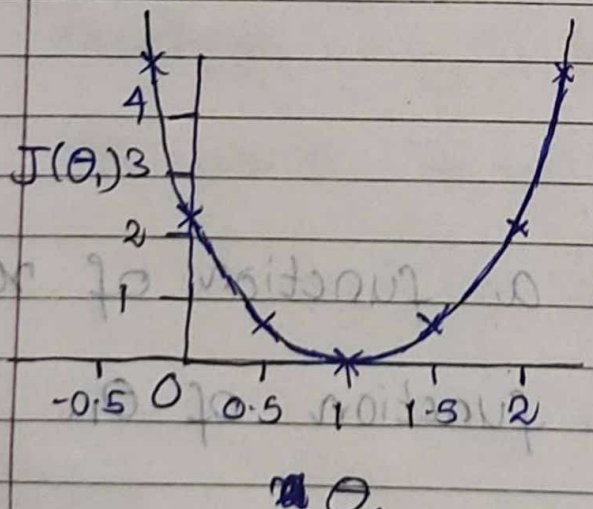
$\theta_1 = 0.5$

$\theta_1 = 0.5$

$h_\theta(x) = 0.5x$

$J(0.5) = \frac{1}{6}\{(0.5)^2 + (1)^2 + (1.5)^2\}$

$J(0.5) = 0.58$



Date: / /
Page No.

We had to minimize $J(\theta_1)$ and we can see $\min(J(\theta_1)) = 0$ as seen from the graph. Thus our function $h_\theta(x)$ is basically

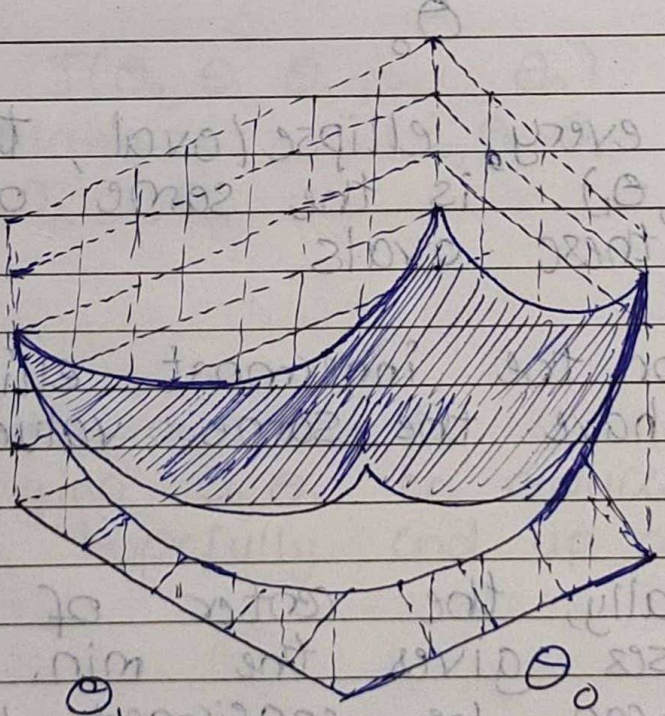
$h_\theta(x) = 1.x = x.$

CONTOUR PLOTS:

We figured out the hypothesis and the cost function when $h_\theta(x) = \theta_1 x$ and $J(\theta_1)$ depends on θ_1. What if $h(x)$ depends on both θ_0 and θ_1?

We saw that for the first case, the graph for the cost function is bow-shaped.

Now, we may get a plot something like this -

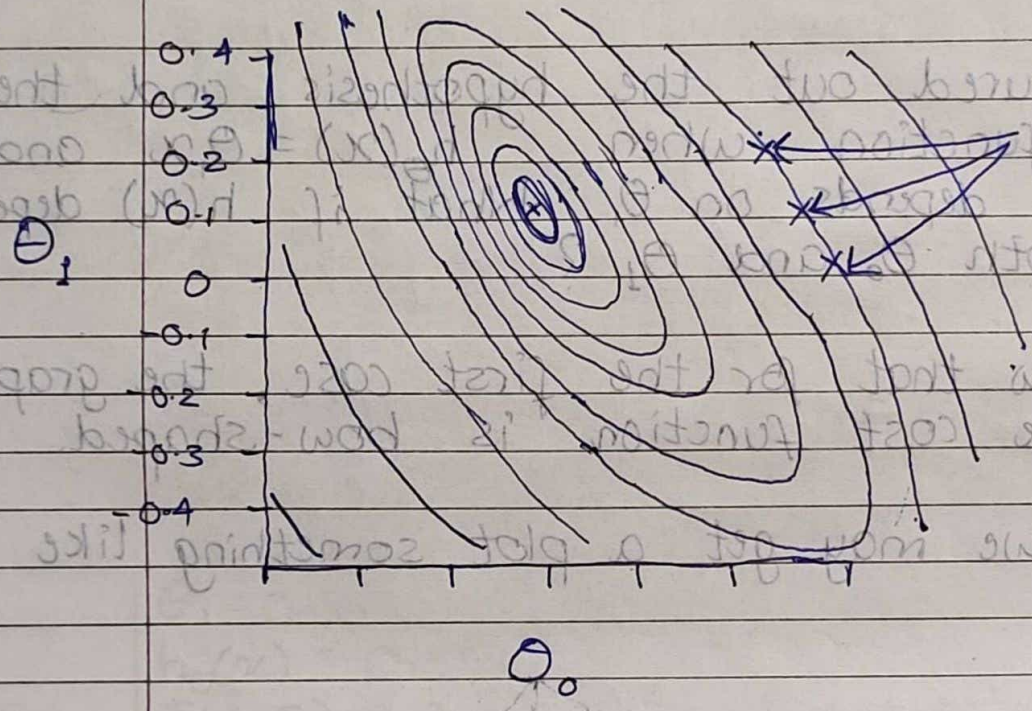


3D Plot

Date: / /
Page No.

So, in this graph also, a bow shaped figure is obtained a thus, we may assume that the cost function always has a bow-shaped graph

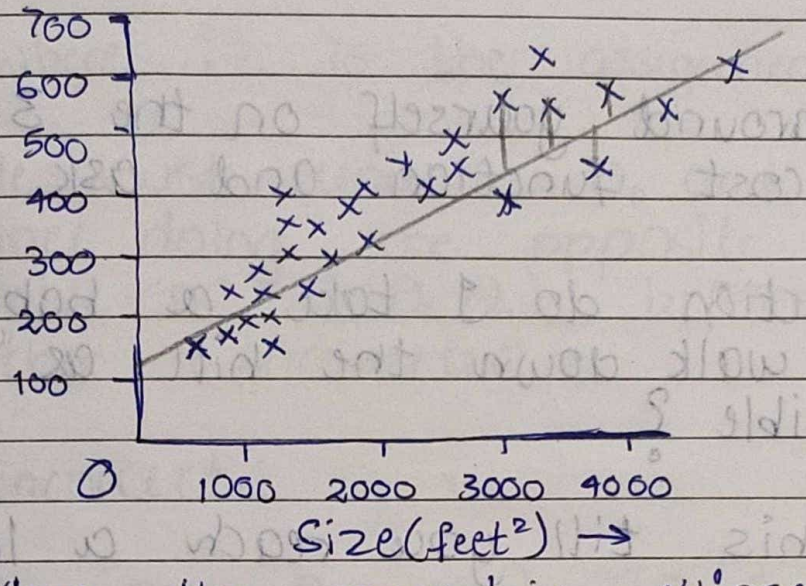
We may not use such 3-D figures and would rather use contour plots which look somewhat like this -



For every ellipse/oval, the value of $J(\theta_0, \theta_1)$ is the same on the bound of these ovals.

So, for the innermost ellipse, $J(\theta_0, \theta_1)$ will have the same value on the bou dary.

Generally, the center of these concentr ellipses gives the min. value of $J(\theta_0, \theta$ and can be confirmed by drawing the cost function



When the concentric ellipses' centre is taken.

* GRADIENT DESCENT:

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

This applies to all functions:

Function: $J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

Want: $\min_{\theta_0, \theta_1, \ldots, \theta_n} J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

- Outline:

① Start with some θ_0, θ_1

② Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at minimum

Look all around yourself on the 3-D plot of the cost function and ask yourself,

What direction do I take a baby step in, to walk down the hill as rapidly as possible?

Repeat this till you reach a local minima.

It is a local property. Changing the starting point, you may end up at some other local minima.

- Gradient Descent Algo.

repeat until convergence {

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

}

Correct Simultaneous update

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

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

$$\theta_0 := temp0$$
$$\theta_1 := temp1$$

where := is the assignment operator.

We must update θ_0 and θ_1 simultaneously since doing the opposite will change the value of $J(\theta_0, \theta_1)$ before we update the other parameter.

Incorrect:

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

$$\theta_0 := temp0$$

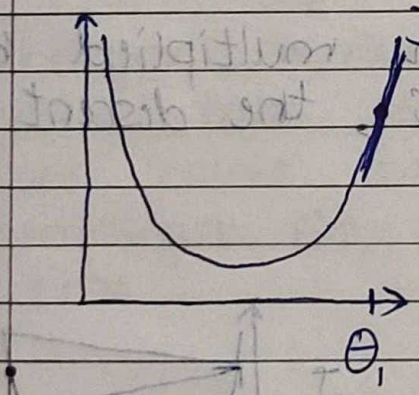
Here we have already updated θ_0.

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

$$\theta_1 := temp1.$$

Say we have only one parameter θ_1

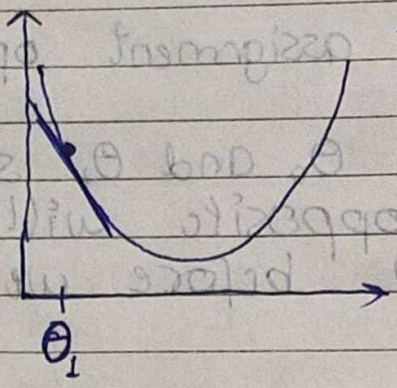
$J(\theta_1)$ is our cost function



$(\theta_1 \in \mathbb{R})$

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

So, at this point $\frac{d}{d\theta_1} J(\theta_1)$ is +ve & α is always +ve, so θ_1 will decrease and we will approach the minima.

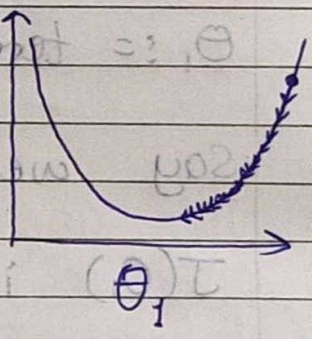


Here, the slope is negative, so $-\alpha \frac{d}{d\theta_1} J(\theta_1)$ will be positive and thus θ_1 will increase till we reach the minima.

- Variations in α

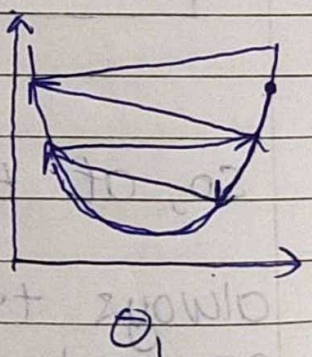
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

① If α is too small, the gradient descent can be too slow. This is because $\frac{d}{d\theta_1} J(\theta_1)$ will keep



on decreasing and when multiplied by the very small value of α, the descent will be too slow.

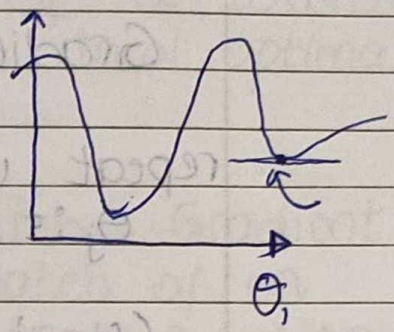
② If α is too large, we might miss the optima and might not be able to converge or even diverge.



So, if $\frac{d}{d\theta_1} J(\theta_1)$ is multiplied with the

huge value of α, it might be still too big when it approaches the optima and thus, might not converge.

③ If θ_1 is already at the local minima.



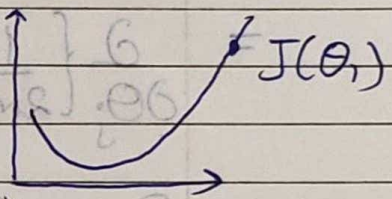
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

α is non-zero +ve

$\frac{d}{d\theta_1} J(\theta_1) = 0$

$\therefore$ θ_1 remains constant.

④ Learning Rate α is constant.



Even with α being constant Gradient Descent can converge to a local minima. This is because, $\frac{d}{d\theta_1} J(\theta_1)$ will automatically become smaller and thus we will take smaller steps as we move towards the local minima. So, no need to decrease α over time.

GRADIENT DESCENT FOR LINEAR REGRESSION

Gradient Desc.	Linear Regression

repeat until conv. {
$\theta_j := \theta_j - \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
(for $j=0$ and $j=1$)
}
	$h_\theta(x) = \theta_0 + \theta_1 x$
	$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

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

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

For $j = 0$,

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

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

Gradient Descent Algo.

repeat until convergence {

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

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

Date: / /
Page No.

NOTE: It turns out that the cost function for a linear regression problem is always a bow-shaped curve. Technically, a convex function. So it only has one local optima. i.e. the global minima.

This all is also called "Batch" Gradient Descent, since we take a batch of m training examples into our cost function and eventually in our gradient descent.

Date: / /
Page No.

MULTIVARIATE LINEAR REGRESSION

Previously, we had

Size (feet^2) x	Price ($1000) y
2104	460
1416	232
1534	315
852	178
	...

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

with just one feature x, the size of the house.

Now, let's say we have not just the size of the house, but also other features like Number of bedrooms, Number of floors, Age of house, etc.

Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($) (1000s) Y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...
...	...	...	...	...

Date: / /
Page No.

Notation:

n = no. of features
$x^{(i)}$ = input features of i^{th} training example
$x_j^{(i)}$ = value of feature j in i^{th} training ex.

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

$x_1^{(4)} = 852$

Hypothesis:

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

Now,

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

for 4 features in a given data set.

For n features,

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

For convinience, define $x_0 = 1$.

This means, for every feature i, there will a new value $x_0^{(i)} = 1$.

So, we sort of have a new feature and X becomes a $(n+1)$-dimensional vector

Date: / /
Page No.

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n.$$

$$\boxed{h_\theta(x) = \theta^T X}$$

* GRADIENT DESCENT :

Hypothesis : $h_\theta(x) = \theta^T X = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n$

$\hookrightarrow x_0 = 1$

Parameters : $\theta_0, \theta_1, \theta_2, \ldots, \theta_n$. $-(\theta)$

Cost Function :

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

Gradient Descent :

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$$

}

(simultaneously update for every $j = 0, 1, 2, \ldots, n$)

Date: / /
Page No.

New Algorithm ($n \geq 1$):

Repeat {

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

(simultaneously update for $j = 0, 1, \ldots, n$)

}

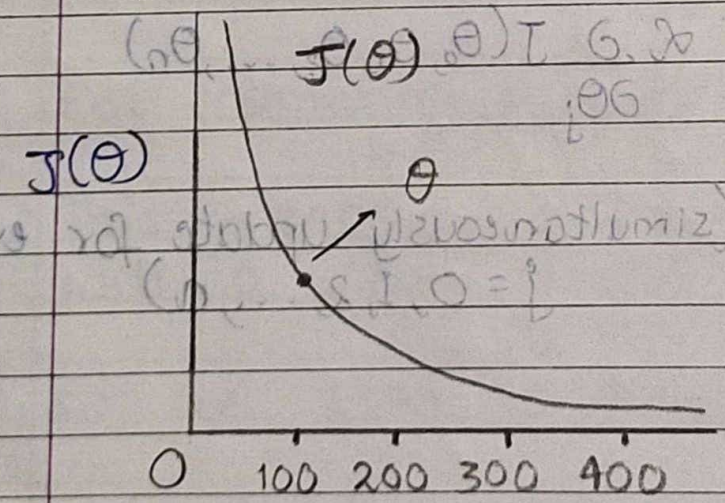
$$\theta_0 := \theta_0 - \frac{\alpha}{m} \sum_{i=1}^{m} \{(h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}\}$$

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

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

* Making Sure Gradient Descent is Working Correctly.

$\min_\theta J(\theta)$



Date: / /
Page No.

- The job of g.d. is to find the value of θ for you that hopefully minimizes the cost function

- Plot $J(\theta)$ as g.d. runs. x-axis is the no. of iterations. Previously, we had our x-axis as the value of θ

- So, concretely, what a point, say (100, ~) means is that after the gd algo is has run for 100 times, each time simultaneously updating θ, the value of $J(\theta)$ obtained is ~.

→ Hence, if gradient descent is working properly, then the value of $J(\theta)$ must decrease after every iteration

→ It also helps in determining that by how many iterations, the value of $J(\theta)$ remains more or less same and thus the cost function has converged.

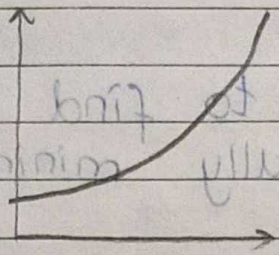
AUTOMATIC CONVERGENCE TEST :

Declare convergence if $J(\theta)$ decreases by less than some specified value, say $\epsilon > 0$ in one iteration.

Eg $\epsilon = 10^{-3}$.

But this is ambiguous. because choosing ϵ can be difficult.

→ If,

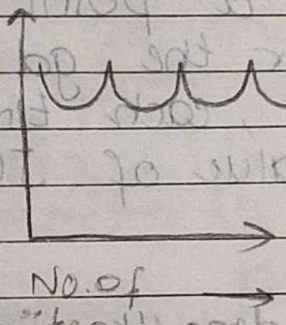


Gradient Descent not working

Use smaller α.

It is overshooting the optima as discussed before.

→ Also,



G.D not working.

Use smaller α.

→ But, if α is too small, g.d can be very slow while converging.

Hint: To choose α, try

$\ldots, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, \ldots$

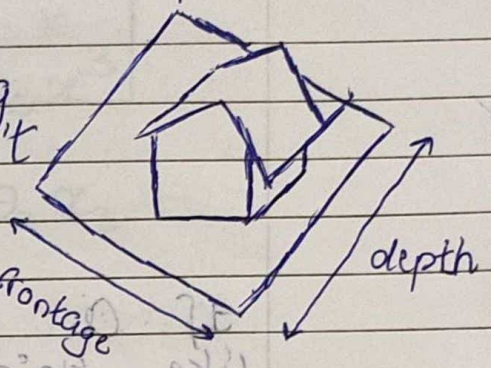
3x 3x 3x 3x 3x

Date: / /
Page No.

* FEATURES AND POLYNOMIAL REGRESSION

• Housing Prices Prediction

$h_\theta(x) = \theta_0 + \theta_1 \times \text{frontage} + \theta_2 \times \text{depth}$



But, when you are applying linear regression, you don't necessarily have to use just the given features x_1 and x_2. You can create new features by yourself.

So, I might decide that what really determines the size of house is the area of the plot. So, feature x is created such that

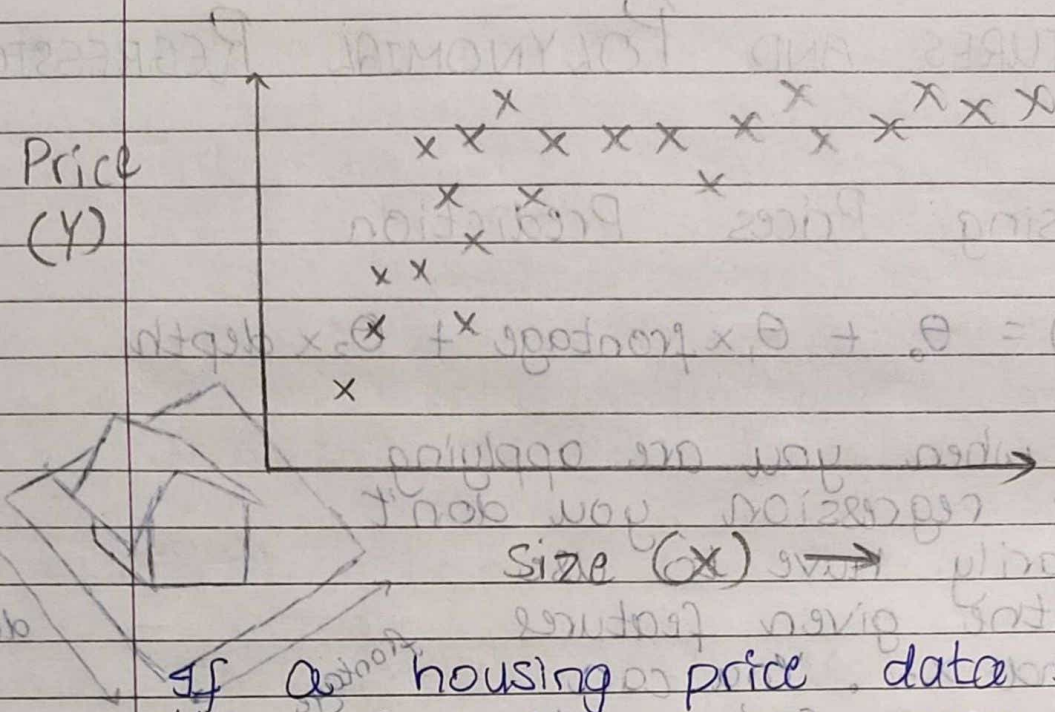
$x = \text{frontage} * \text{depth}$

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

(x ← Area)

Closely, related to the idea of choosing your features is this idea called polynomial regression.

Date: / /
Page No.



If a housing price data set looks like this, then you may come up with different models to fit this

One thing you could do, is fit a quadratic model like this since is looks like straight line model does not fit this quite well.

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

But then this will eventually go down and housing prices don't go down so, we might try a cubic model.

Eg $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

Now, we know the machinery of multivariante linear regression we can convert our hypothesis with the help of a very simple modification.

Date: / /
Page No.

We have seen how does a multivariate linear regression algo works, where

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

We can use this expression,

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

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

$x_1 = (\text{size})$
$x_2 = (\text{size})^2$
$x_3 = (\text{size})^3$

and now, since x_1, x_2 & x_3 take very different range of values, thus we must use feature scaling in order to bring them down to a desired range.

Eg
$(\text{size}) : 1 - 1000$
$(\text{size})^2 : 1 - 10^6$
$(\text{size})^3 : 1 - 10^9$

Date: / /
Page No.

* NORMAL EQUATION

Till now, the algorithm we were using for linear regression is G.D. which involves simultaneously updating all the parameters in the direction of the steepest descend thus reaching a local minima. It's an iterative algorithm that involves a large no. of steps to minimize the $J(\theta)$.

Normal Eq^n : Method to solve for θ analytically.

Intuition:

If 1D ($\theta \in \mathbb{R}$)

$$J(\theta) = a\theta^2 + b\theta + c$$

To minimize $J(\theta)$, $\frac{d}{d\theta} J(\theta) \overset{set}{=} 0$

Solve for θ.

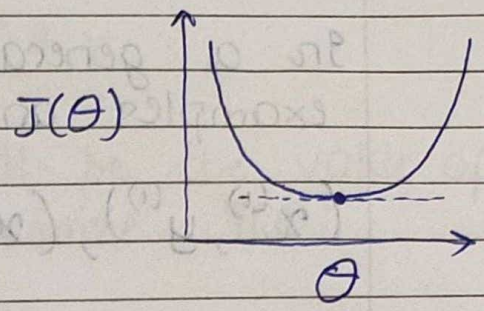
Our Case: $\theta \in \mathbb{R}^{n+1}$

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

$$\frac{\partial}{\partial \theta_j} J(\theta) = 0 \quad \text{(for every } j\text{)}$$

Solve for $\theta_1, \theta_2, \theta_3, \ldots, \theta_n$.

Date: / /
Page No.



Ex Let $n = 4$.

x_0	Size x_1	No. of bedr. x_2	No. of floors x_3	Age of home x_4	Price ($1000) 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

① Add an extra feature $x_0 = 1$

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \quad m \times (n+1)$$

(4×5)

$$y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \quad (4 \times 1)$$

m-dimensional

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

In a general case, let there be m training examples and n features.

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

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$(x^{(i)})^T = \begin{bmatrix} x_0^{(i)} & x_1^{(i)} & \cdots & x_n^{(i)} \end{bmatrix}$$

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

$(m \times (n+1))$ matrix

Eg: If $x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x \\ & \\ & \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

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

Date: / /
Page No.

Octave : pinv(X'*X)*X'*Y

$\theta \Rightarrow (X^T X)^{-1} X^T Y$ will be the value of θ that will minimize $J(\theta)$

NOTE: There is no need to scale down / up the parameters when you're using normal eq^n.

GRADIENT DESCEND	NORMAL EQUATION
• Need to choose α	• No need to choose α
• Needs many iterations	• Don't need to iterate.
• Works well even when n is large. $n = 10^6$.	Need to compute $(X^T X)^{-1}$ and is slow when n is large. $O(n^3)$ $n = (100), (1000), (10,000)$.

Q What if $X^T X$ is non-invertible ?

- Redundant features (linearly dependent)
  Eg. x_1 = size in feet^2
  x_2 = size in m^2

- Too many features ($m \leq n$)
  - Delete some features, or use regularization