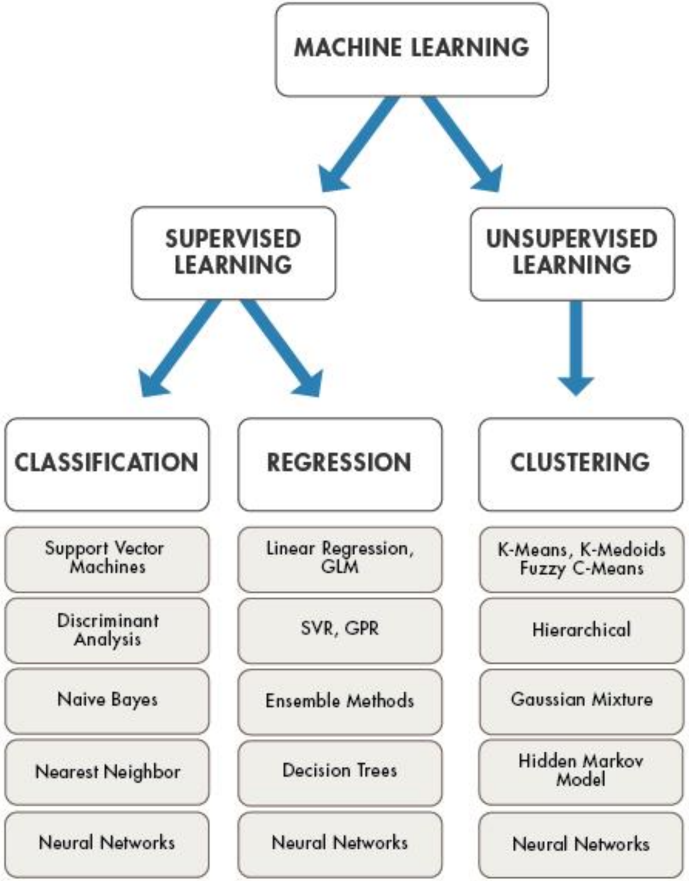
Hands-On Session on Linear Regression

By- Bhuvanjeet Singh Gandhi


MACHINE LEARNING
SUPERVISED LEARNING
UNSUPERVISED LEARNING
CLASSIFICATION
REGRESSION
CLUSTERING
Support Vector Machines
Discriminant Analysis
Naive Bayes
Nearest Neighbor
Neural Networks
Linear Regression, GLM
SVR, GPR
Ensemble Methods
Decision Trees
Neural Networks
K-Means, K-Medoids Fuzzy C-Means
Hierarchical
Gaussian Mixture
Hidden Markov Model
Neural Networks

Overview:

Today's webinar is divided into 2 parts :

Part I - Linear Regression Understanding

Part II- Hands-On Session on Jupyter Notebook

PART - I

What is Linear Regression?

1- Linear Regression is a machine learning algorithm based on supervised learning (the machine learning task of learning a function that maps an input to an output based on example input-output pairs.)

2- It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

3- Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used -

=> When there is only 1 independent variable - Univariate or Simple Linear Regression ,
=> When there are many independent variables - Multiple Linear Regression

Let’s Dive Deeper :

Here , we are having a data set which has one independent variable/feature (x) - Size and a dependent/target variable (y):

Training set of housing prices (Portland, OR)

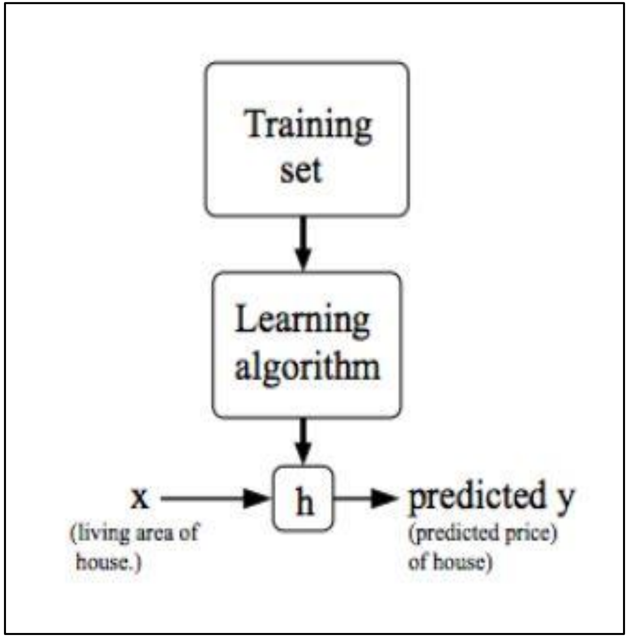
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

Notation:

m = Number of training examples
x’s = “input” variable / features
y’s = “output” variable / “target” variable

Model Representation:

Univariate Linear Regression:

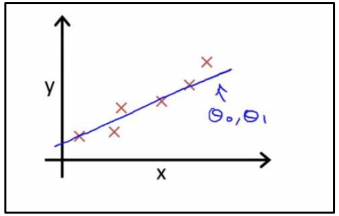


Hypothesis Function:

- So, the h in the flowchart represents ‘hypothesis function’ which maps from x’s to y’s.

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

Here, θ_i‘s: Parameters i.e. theta0 and theta1 are the parameters of this hypothesis function.



=> h(x) is the predicted output value for the input x.

- Also, we can see that the hypothesis function is of the form y=mx+c which is the equation of straight i.e. a linear line. So basically hypothesis function is nothing but a straight line.

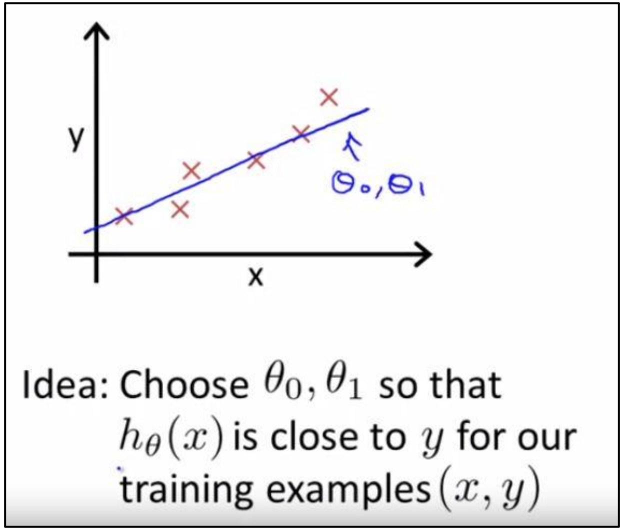
So now the question arises: **How to choose θ_i's ?**

Here comes the role of **Cost function** : -

Cost Function:-

Cost function helps us to figure out how to fit the best possible straight line to the data.

We want to come up with such values of parameters theta0 & theta1 so that the straight line with which we come up is the line of best fit.



i.e. $$\underset{\theta_0, \theta_1}{\text{minimize}} \quad h_\theta(x) - y$$

We want to find such values of theta0 & theta1 which minimizes the difference between h(x) and y i.e we want the predicted value to be as close as possible to the actual target value.

Since there are m training examples,

i represents the ith training example i.e. for first training example, i=1 and so on.

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

The cost function is defined as: -

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

So, now we have understood that we want to find such values of theta0 & theta1 so as to minimize the cost function J(theta0,theta1).

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

So , our overall optimization objective is to minimize the cost function.

Let's do a quick recap:

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

Cost Function:

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

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Now the question arises:

How to find such values of parameters theta0 and theta1 which minimizes the cost function J.

Here, Gradient Descent comes to rescue.

Gradient Descent

So, to find(update) the parameters, we will be using of an optimization algorithm known as gradient descent.

Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) .

This is always the case in machine learning problems since the real-life dataset has too many features and a large number of training examples which makes it almost impossible to calculate the parameters analytically.

Basic Idea of Gradient Descent:

- Start with some initial guesses of theta0 and theta1 ,

  Say theta0 = 0 & theta1 = 0

- Keep changing theta0 & theta1 a little bit to reduce the cost function J(theta0,theta1) until we hopefully end up at a local minimum.

For the ease of understanding, let theta0 = 0

	Simplified
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$	$h_\theta(x) = \theta_1 x$ $\theta_0 = 0$
Parameters: θ_0, θ_1	θ_1
Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$	$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$	$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

And let m=3 :

(x1,y1) = (1,1) ; (x2,y2) = (2,2) ; (x3,y3) = (3,3)

<u>Case 1:</u> Let theta1=0 :

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

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

$$= \frac{1}{2m}\left((-1)^2 + (-2)^2 + (-3)^2\right)$$

$$= \frac{1}{\cancel{2}\times 3}\left(\overset{7}{\cancel{14}}\right) = \frac{7}{3} = 2.33$$

$$(\because m=3)$$

<u>Case 2:</u> Let theta1=0.5 :

$$J(0.5) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

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

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

$$= \frac{1}{2\times 3} (3.5) = \frac{3.5}{6} = 0.58$$

Case 3: Let theta1=1

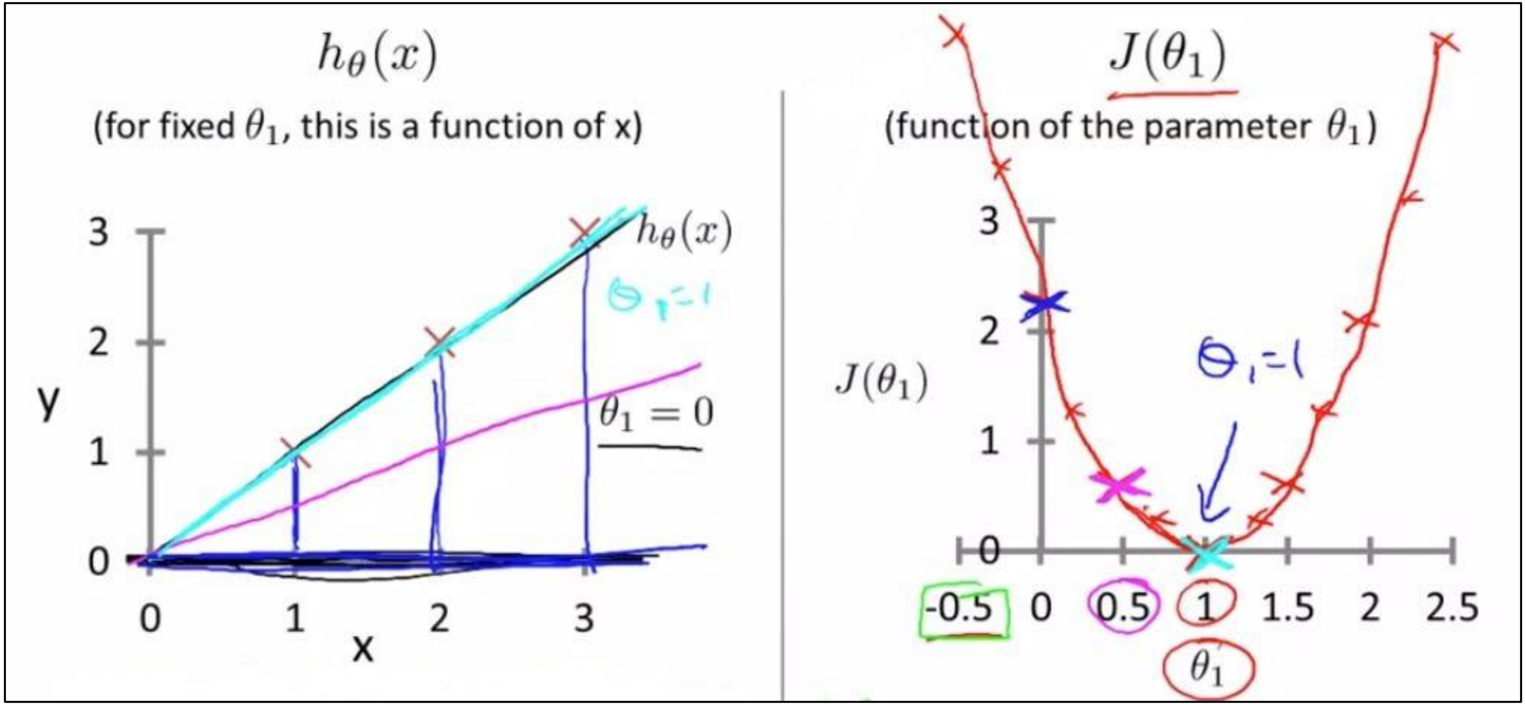
$$J(1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

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

$1 \cdot 1 = 1 ; 1 \cdot 2 = 2 ; 1 \cdot 3 = 3$

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

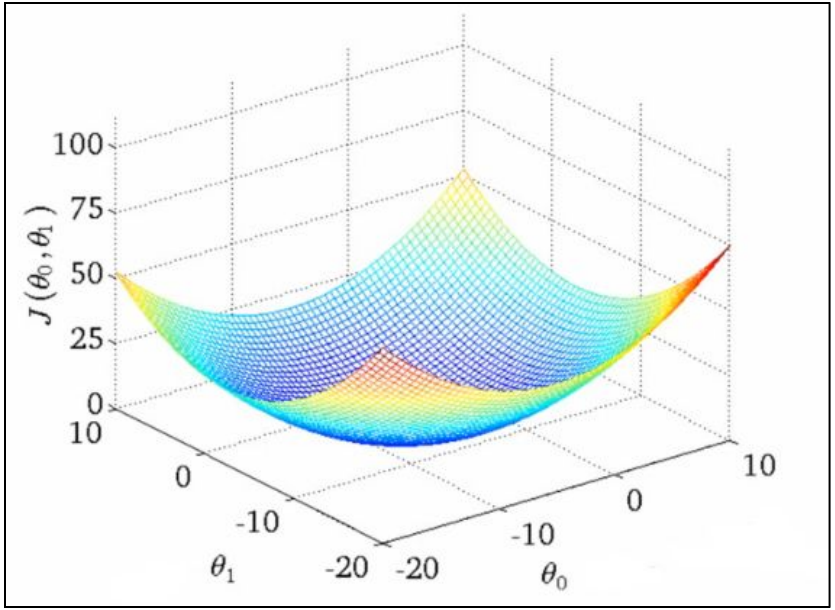
$$\Rightarrow J(1) = 0$$



$h_\theta(x)$
(for fixed θ_1, this is a function of x)
3
2
1
0
y
0
1
2
3
x
$h_\theta(x)$
$\theta_1=1$
$\theta_1 = 0$
$J(\theta_1)$
(function of the parameter θ_1)
3
2
1
0
$J(\theta_1)$
$\theta_1=1$
-0.5
0
0.5
1
1.5
2
2.5
θ_1

Now, suppose we now want to find out the values of both the parameters theta0 and theta1 , then:

The plot of cost function will look like:



Remember:

Cost function for linear regression is convex shaped i.e. it has no local optima , only 1 global optima, so it will always converge to global optima.

Summing it all up:

1. Hypothesis function:
2. Cost Function:
3. Gradient Descent
4. From gradient descent we will find such values of theta0 and theta1 which minimizes the cost function.
5. On placing these parameter values in hypothesis function, we will get the predicted output value.

PART - II

Hands-On Session

Dataset :- https://www.kaggle.com/vikrishnan/boston-house-prices

If you want to follow along :-

Github repo :- https://github.com/Bhuvanjeet/Boston-House-Price-Prediction

** Head Over to Jupyter Notebook **

Thank You for being such an amazing audience

Connect with me -

LinkedIn: www.linkedin.com/in/bhuvanjeet

Email: bhuvanjeet_08@ieee.org