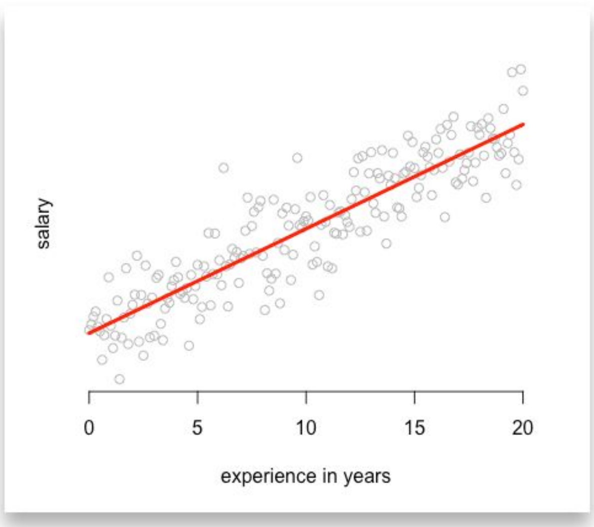
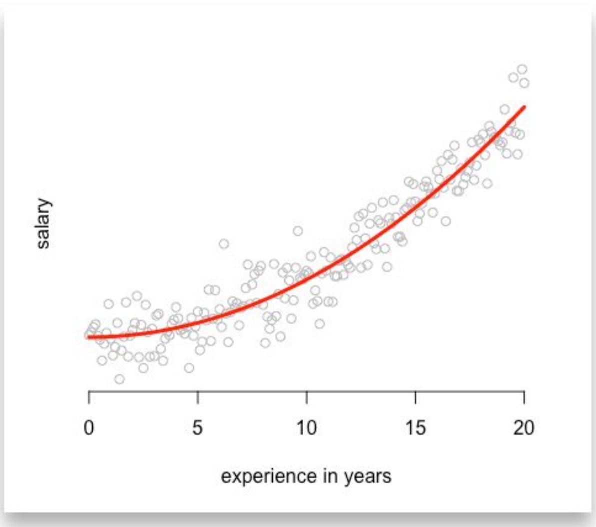
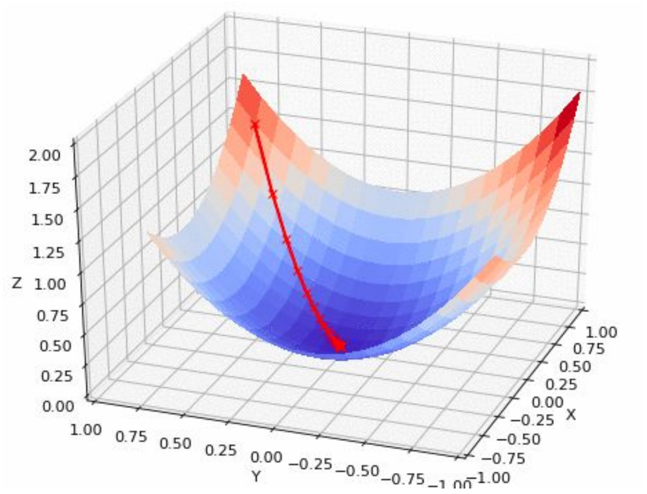
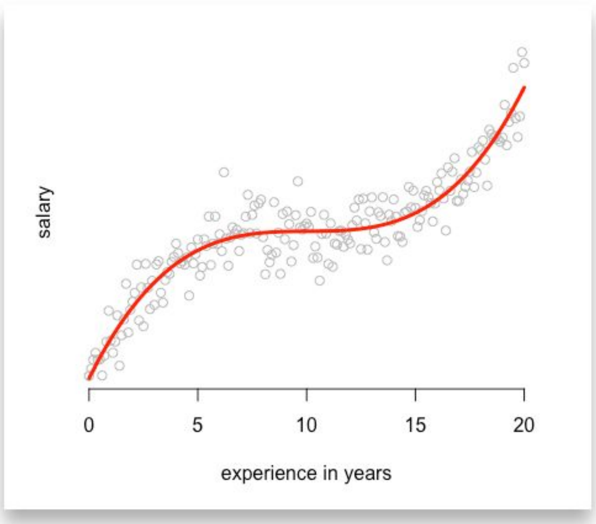
Machine Learning









Kaushal Kishore
October 7, 2018
Data Analysis Club, IITPKD

Contents

Chapter 1

Introduction to Machine Learning

1.1 Definitions

"Field of study that gives computers the ability to learn without being explicitly programmed." — Arthur Samuel (1959)

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience." — Tom M. Mitchell

Machine Learning is defined in various ways, but the Mitchell's definition is most popular among them.

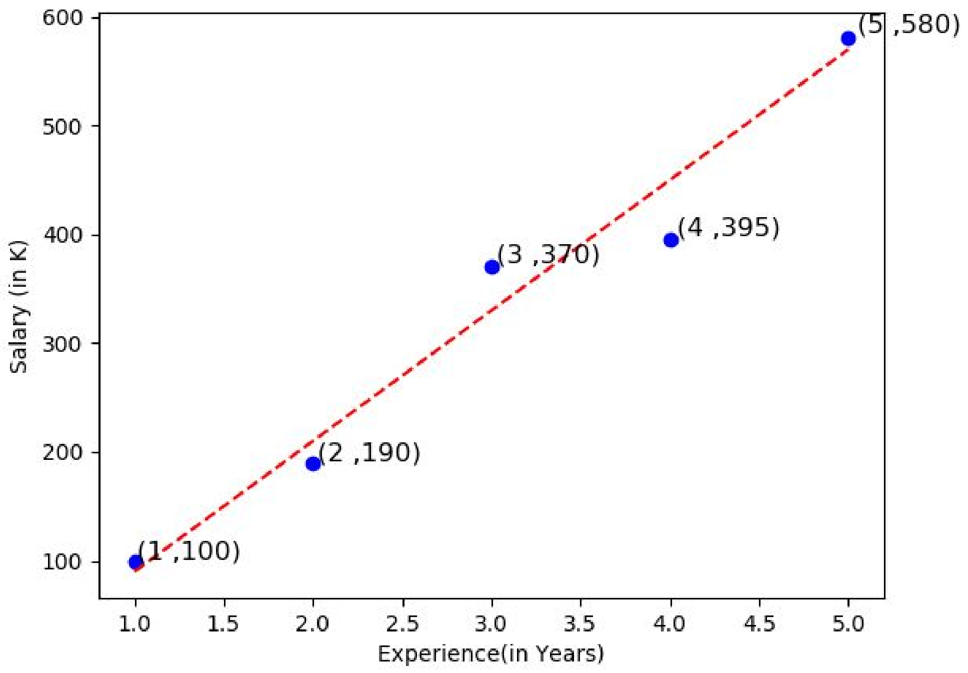
1.2 ML Goals

While doing ML, you will be provided some data, for instance let's consider a dataset consisting of experience (in years) and salary. You might have already guessed by now that we want the Machine(a computer program) to **study the data** and come up with **a model**(a function) such that it will be able **to predict** the salary(approximately) of a person given the amount of the experience. Ofcourse this prediction is done on unseen data (i.e., the data which is not used for training the model).

Let's consider the following dataset:

Experience (in Years)	Salary
1	100 K
2	190 K
3	370 K
4	395 K
5	580 K

Let's visualize the above data:



Intuitively, we can observe that a model function which can predict the salary can be similar to what is shown by the red line in the above figure. What I did in the above image is just tried to make a curve (line in this case) which seems most plausible (ie., an **optimal fit**). Let's denote this model function by $h(experience)$.

For a moment, let assume I give you this model function and ask you to tell me the salary for a person who has experience of 2.5 years. You can easily predict the salary by using the model function i.e., $h(experience = 2.5)$.

So the goal of the ML(roughly) is to find this model function with some study of the available data(we call this training the Machine on training dataset) and then use this model function to predict the values of some arbitrary input.

Dataset → Training → Model Function → Prediction

Rest of the discussion will be on how to find this model function $h(experience)$.

1.3 Model Function

Considering the same training dataset (experience vs. salary) and let this dataset be denoted by,

$$D = \{ (x_i , y_i) \}_{i=1}^{m} = \{ (x_1 , y_1) , (x_2 , y_2) , ... , (x_m , y_m) \}$$

Where m is the number of the training data points, hence for our current scenario, i.e., experience vs. salary dataset, m is 5 and each (x_i , y_i) denotes a training data point where x_i is the experience and y_i is the salary of i^{th} data point.

Consider a set $H = \{h_1 , h_2 , h_3 , ... , h_N \}$ of functions such that $h_i : (experience \in \mathbb{R}) \longrightarrow (salary \in \mathbb{R})$, $\forall\ h_i \in H$.

Let's assume an imaginary function h_{ideal} which can accurately determine the salary of the a person given the experience. What I mean by accurately is that:

$$\forall\ (x_i , y_i) \in D , \ y_i = h_{ideal}(x_i)$$

Intuitively, we can see that no function can perform achieve better accuracy on training data points than this imaginary h_{ideal} function, however it can be as good as this function but not better.

Next section defines the metrics of how to compare two model functions. In machine learning jargon, the model functions of set H are called hypothesis function. From now on we will be using these terms interchangeably.

1.4 Concept of Error Function

In the previous section we have defined a set H which consists of many hypothesis function. The ML problem wants the Machine to select the best hypothesis function from this set. What I mean by the best function is that it's accuracy on the training set is better than all other set elements. In this section we will define a way in which we can compare any two hypothesis function.

Let's define the error function:

$$E(h, D) = \sum_{(x,y) \in D} (h(x) - y)^2$$

Where h is an hypothesis function and D is the training dataset(defined in previous section).

Hence, our imaginary function $h_{best}(x)$ in principle have an error 0.

$$E(h, D) = \sum_{(x,y) \in D} (h_{ideal}(x) - y)^2 = \sum_{(x,y) \in D} (y - y)^2 = 0$$

Hence, $h_{ideal}(x)$ is an ideal function for the ML problem. Unfortunately, we don't know this $h_{ideal}(x)$ completely, if we would, then there is no need for the machine to find this hypothesis, and hence no machine learning. All we know about $h_{ideal}(x)$ is that for a data point (x, y) the value of $h_{ideal}(x)$ is exactly equal to y.

Intuitively, we want find an $h(x)$ which can perform as good as $h_{ideal}(x)$.

We say a hypothesis function h_1 is better than h_2 if the following condition is satisfied:

$$E(h_1, D) \leq E(h_2, D)$$

I.e., the error generated by h_1 is less than h_2.

To select the most suitable hypothesis function from the set H:

$$h_{best}(x) = arg\ min_{h \in H} E(h, D),$$

i.e., choose the $h \in H$ such that the error is minimum.

To summarise, we defined an error function to compare hypothesis functions and then devised a procedure to select the best hypothesis function h_{best}(x) from the set H. Intuitively, our aim was to perform as good as $h_{ideal}(x)$ function. To do this we considered a set H consisting of many hypothesis functions. Hence we gave the machine a limited (finite) number of hypothesis functions in form of set H to choose from. But a situation can arise when there exist functions not belonging to H which are better than the h_{best}(x). In the next section we will discuss how to allow the machine to explore an infinite number of hypothesis functions (of a certain form) to choose from, in the hope of finding a function which can perform as good as $h_{ideal}(x)$.

1.5 Linear Regression

Before we move forward, let's discuss the notations we are going to follow:

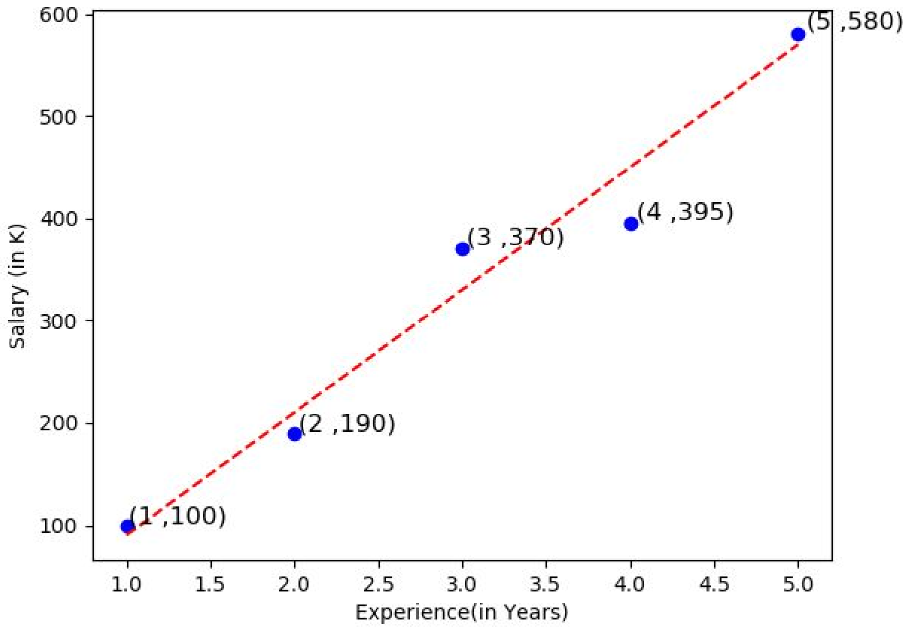
- **Dataset (D) :** Set of training data, where each element is of the form (x, y), where x denotes the **feature** and y denotes the value associated with it.

$$D = \{ (x_i, y_i) \}_{i=1}^{m} = \{ (x_1, y_1), (x_2, y_2), \ldots\ldots, (x_m, y_m) \}$$

- **Total number of training examples** is denoted by m.

- **Error Function,** $E(h, D)$: error value of hypothesis function h on dataset D.
- **Number of Features (n):** In this case the number of features is 1 i.e., the experience in years.

In the previous section, we defined a set H containing some finite number of hypothesis functions. And we discussed the limitations of considering the finite number of hypothesis functions, i.e., the most ideal function may not belong to the set H. We need to extend this idea to a set H such that it contains infinitely many possibilities of hypothesis function.



Considering the same data set of experience vs. salary we can easily argue that the red line is one such optimal line, that has a reasonably good accuracy. But the question is 'How we are supposed to find that line or in general the hypothesis function?'

The procedure is as follows:

Let assume an arbitrary function $h_\theta(x) = \theta_1 x + \theta_0$, where $\theta_1, \theta_0 \in \mathbb{R}$. Now the task is to find that θ_1, θ_0 such that $E(h, D)$ is minimum.

Hence minimize, $E(h, D) = \sum_{(x,y)\in D} (h_\theta(x) - y)^2 = \sum_{(x,y)\in D} (\theta_1 x + \theta_0 - y)^2$

Rewriting the above equations in the following form:

$$E(h, D) = \sum_{i=1}^{m} (\theta_1 x_i + \theta_0 - y_i)^2 = \sum_{i=1}^{m} (y_i - \theta_1 x_i + \theta_0)^2$$

Now let's represent the $\theta's$ in form of a column matrix.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

And similarly y_i's and x_i's in form of a column matrix:

$$Y = \begin{bmatrix} y_0 \\ y_1 \\ .. \\ .. \\ y_m \end{bmatrix} \qquad X = \begin{bmatrix} x_0 \\ x_1 \\ .. \\ .. \\ x_m \end{bmatrix}$$

But instead of representing X in the above form, let's represent it in the following form:

$$X = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ .. & .. \\ .. & .. \\ 1 & x_m \end{bmatrix}$$

We just inserted a column containing 1's in the previous X matrix. It will be clear in a moment why we are representing in this form.

We can rewrite the equation: $E(h,D) = \sum_{i=1}^{m} (y_i - \theta_1 x_i + \theta_0)^2$ by using the above matrices as follows:

$$E(h,D) = (Y - X\theta)^T (Y - X\theta)$$

Please make sure that you understand the above matrix equation.

Proceeding further, we want to minimize error E. Hence differentiating E with respect to parameter vector θ (for now assume vector as just column matrix). Since, the above equation involves matrix differentiation most of you might face difficulty at present. Hence, I am going to do this calculation on behalf of you (just following along with me).

$$\tfrac{\partial E}{\partial \theta} = \tfrac{\partial}{\partial \theta} (Y - X\theta)^T (Y - X\theta)$$

Use the following matrix differentiation identity to simplify the above equation:

$$\frac{\partial}{\partial \mathbf{s}} (\mathbf{x} - \mathbf{A}\mathbf{s})^T \mathbf{W} (\mathbf{x} - \mathbf{A}\mathbf{s}) = -2\mathbf{A}^T \mathbf{W} (\mathbf{x} - \mathbf{A}\mathbf{s})$$

where W is symmetric matrix.

$$\tfrac{\partial E}{\partial \theta} = \tfrac{\partial}{\partial \theta} (Y - X\theta)^T (Y - X\theta) = \tfrac{\partial}{\partial \theta} (Y - X\theta)^T I (Y - X\theta)$$

where I is identity matrix.

$$\tfrac{\partial E}{\partial \theta} = -2X^T I (Y - X\theta) = -2X^T (Y - X\theta)$$

Equating above expression to zero <u>matrix</u>(why?) we get,

$$-2X^T (Y - X\theta) = 0 \Rightarrow X^T Y - X^T X\theta = 0 \Rightarrow X^T X\theta = X^T Y$$

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

If X is invertible matrix then,

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

The following result: $\theta = (X^T X)^{-1} X^T Y$ is known by the name '**Normal Equation**'. And if the feature matrix X (this is the first time I am using this name to denote X) is invertible then the normal equation reduces to $\theta = X^{-1} Y$.

In this way we can find the optimal parameters (θ_i). Remember θ is a column matrix containing all parameters. In the current scenario where there is only one feature i.e., the experience in years the size of θ column matrix is 2.

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

Remember the hypothesis function $h_\theta(x) = \theta_1 x + \theta_0$, where θ_1 *and* θ_0 can be determined by normal equation.

Remember the previous section, where we were selecting the h_{best} from a finite set of hypothesis functions H and we discussed the limitations of it. Recall and summarise what we did in this section. We gave machine a template of a function of the form $h_\theta(x) = \theta_1 x + \theta_0$ and then applied the application of derivative to find the function for which the error E is minimum. If you are familiar with applications of derivative then you already know that when we apply this method the first-order differential on equating to 0, it returns us a set of points where a local optima(minima or maxima) can occur, and the optima in this case is a minima(since the error function is parabola with open mouth upwards). The cool feature of using this method is that it searches through the whole range of values of θ_1 *and* θ_0 and returns the optima, which in principle equivalent to selecting h_{best} from a set H which contains all the functions of the form $h_\theta(x) = \theta_1 x + \theta_0$.

In the next section we will discuss a another method of finding the optimal parameters and then we will compare which one is better.

1.6 Gradient Descent

The algorithm for our experience vs. salary example where a hypothesis function is of the form $h_\theta(x) = \theta_1 x + \theta_0$.

```
Repeat Until Convergence {
        Simultaneous Update{
                $\theta_1 := \theta_1 - \alpha \frac{\partial E}{\partial \theta_1}$
                $\theta_0 := \theta_0 - \alpha \frac{\partial E}{\partial \theta_0}$
        }
}
```

Explanation:

First, let's consider the innermost statements:

$\theta_1 := \theta_1 - \alpha \frac{\partial E}{\partial \theta_1}$ and $\theta_0 := \theta_0 - \alpha \frac{\partial E}{\partial \theta_0}$

These statements describes how the parameters θ_i is to be updated in each step. These equations can be understood as follows:

$(\textit{new value of } \theta) := (\textit{current value of } \theta) - \alpha\,(\textit{partial differentiation of E wrt to } \theta)$

Where α is a very small positive number known as **learning rate**.

Remember the statements $\theta_1 := \theta_1 - \alpha \frac{\partial E}{\partial \theta_1}$ and the other one is not an equation, the symbol ":=" in mathematics is called **assignment** operator, it means evaluate the right hand side of the expression and then assign the variable on the left hand side with the evaluation value. Hence the value parameters θ_i will be updated in the each step. The partial differentiation term can be considered as slope of the function at a point (θ_0, θ_1) w.r.t. The parameter by which we are differentiating the E. By simultaneous update we mean that all the right hand sides of the assignment operations are evaluated first and then the parameters are updated simultaneously. In present scenario, by simultaneous update we mean first evaluate the following terms:

$\theta_1 - \alpha \frac{\partial E}{\partial \theta_1}$ & $\theta_0 - \alpha \frac{\partial E}{\partial \theta_0}$, and store the following terms in temporary variables say $temp1$ & $temp0$, and then in the next step update the value of (θ_0 , θ_1) by the value of $(temp0 , temp1)$. Hence, the complete procedure what we mean by simultaneous update is given below:

$$temp1 := \theta_1 - \alpha \frac{\partial E}{\partial \theta_1}$$

$$temp0 := \theta_0 - \alpha \frac{\partial E}{\partial \theta_0}$$

$$\theta_1 := temp1$$

$$\theta_0 := temp0$$

Why simultaneous update?

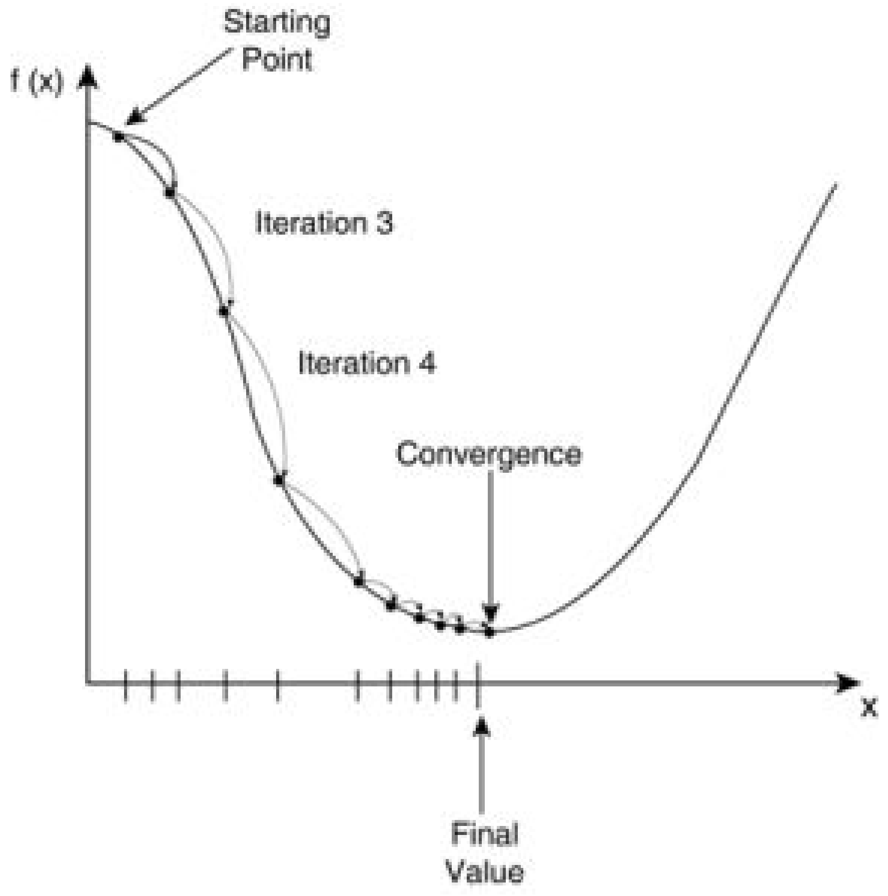
Because if we don't do this the value of θ_1 will change in the first step and then in the second step while calculation of θ_0 we will use this update value of θ_1 to compute the value of $\frac{\partial E}{\partial \theta_0}$ which is not the current gradient and data point (θ_0 , θ_1).

Please go through the above explanation multiple times to make sure you understand this.

Now the only question left is what's happening in the update rule:

The update can be simply written as :

$$(next\ value\ of\ \theta) = (current\ value\ of\ \theta) - (small\ +ve\ number) * (slope)$$



Consider the above figure where the optima is denoted by '**final value**'. Consider the following equation (in accordance with the figure given above) :

$$(\textit{new value of } x) \ := \ (\textit{current value of } x) \ - \ \alpha \ * \ (\textit{slope})$$

There arise two case: 1.) your current value of x is to the left of the final value 2.) or it is on the right of the final value

In first case the slope will be negative and hence your update rule will be :

$$(\textit{new value of } x) \ := \ (\textit{current value of } x) + \ (\textit{some positive value})$$

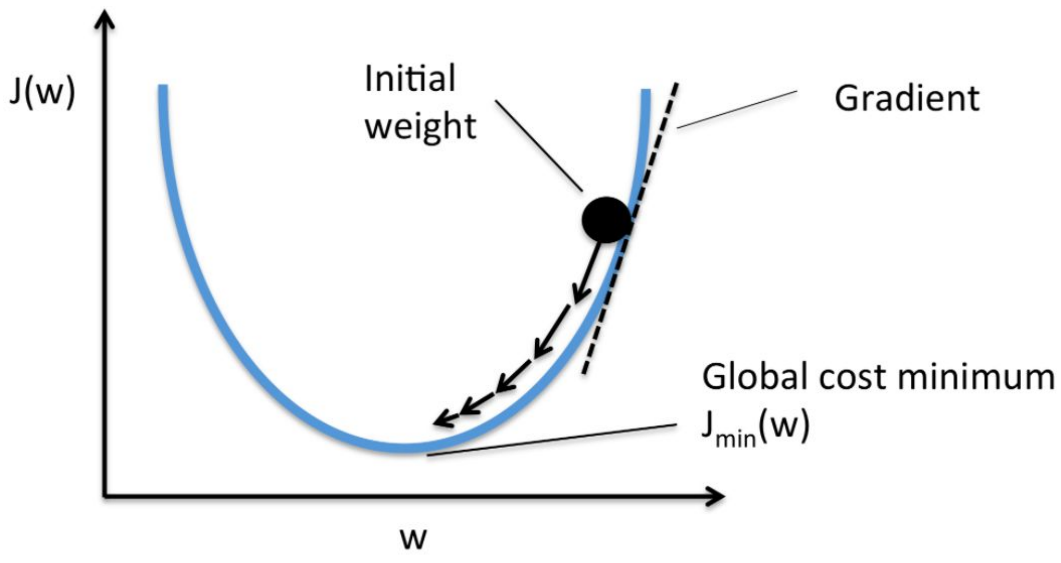
And hence your x will increase which will make it approach the final value.

Similarly, in the second case the slope is positive real number and hence the update rule will be:

$$(\textit{new value of } x) := (\textit{current value of } x) - (\textit{some positive value})$$

And hence the value of the x in this case decreases and approaches the final value.

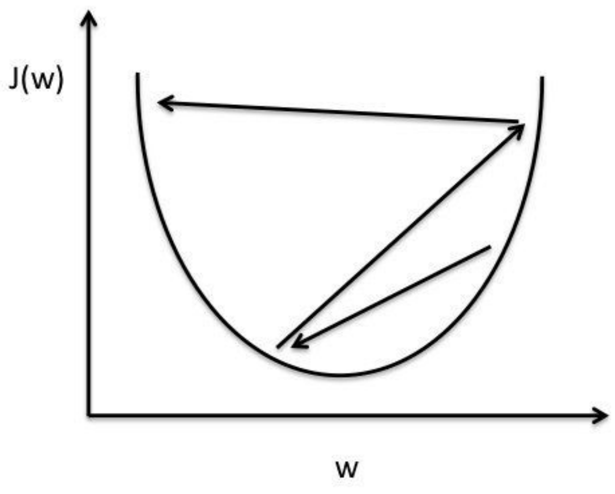
So, we conclude that whatever be the case the gradient descent will make sure that the parameter value approaches to minima and converge. The following image gives a brief intuition of the gradient descent. In the image below replace **w** with θ and **J(w)** with $E(\theta)$ [= $E(h_\theta)$]. Hence when we reach the optima we have reached to a state where the value of error function $E(\theta)$ is minimum.



Also remember the effect of the value of α on the gradient descent algorithm. In the update statement:

$$(\textit{new value of } x) := (\textit{current value of } x) - \alpha * (\textit{slope})$$

The magnitude of $\alpha * (\textit{slope})$ determines the step value, hence if α is very small the gradient descent algorithm will take very small steps to reach the convergence point and if α is very large quantity then the gradient descent algorithm will diverge instead of converging (shown in the figure below).



Large learning rate: Overshooting.

This concludes our discussion on Gradient Descent and hence the Linear regression.

Slack Link for Signup: https://dac-iitpkd.slack.com/signup

Upcoming Sessions: Once we are finished with the programming sessions on regression we will start with the classification problem. What is this classification problem? Instead of definition, I would give example: “given some images of cats and dogs, the machine should be able to tell whether it a cat or a dog” , “handwritten digit recognition”, etc.

If any Query: Ask on #general channel on slack group.