**Gradient Descent**

Gradient Descent is the most optimized technique used in machine learning.

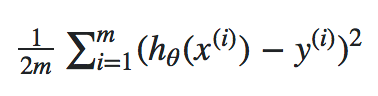
**What is Gradient Descent?**

Gradient Descent is a technique used to find out the best parameters (θ1) and (θ2) for our learning algorithm.

Eg: So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function

**h(x) = θ1+ θ2x**

**Cost function: (**Mean Square Error**)**



**J(θ1, θ2) =**



We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, i.e. when its value is the minimum. The red arrows show the minimum points in the graph.

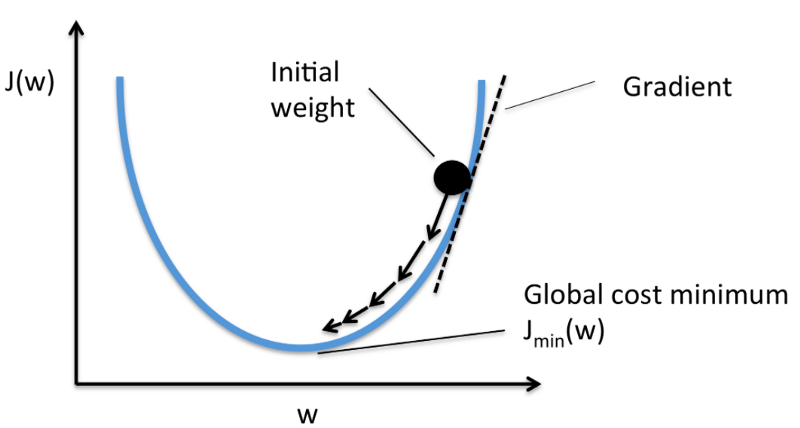
Gradient Descent is a general function for **minimizing a function, in this case the Mean Squared Error cost function.**

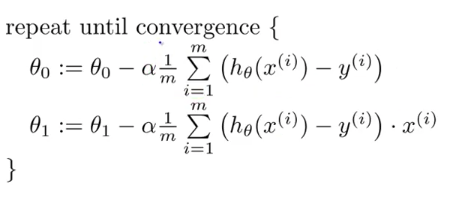
Gradient Descent basically just does what we were doing by hand — change the theta values, or parameters, bit by bit, until we hopefully arrived a minimum.

We start by initializing theta0 and theta1 to any two values, say 0 for both, and go from there. Formally, the algorithm is as follows:

https://cdn-images-1.medium.com/max/800/1*QKHtyn4Rr-0R-s0an1eSsA.png

Where **α**, alpha, is the **learning** **rate**, or how quickly we want to move towards the minimum. If α is too large, however, we can overshoot or too small it’s takes long time to reach the minimum point.



The gradient descent algorithm is:

https://cdn-images-1.medium.com/max/800/1*QKHtyn4Rr-0R-s0an1eSsA.png

**Note:** The gradient moves as in positive or negative directions depends in the

If the slope or gradient values is positive is moves towards the origin direction and if the slope or gradient value is negative it moves away to the origin direction till it reaches the minimum cost function

There are many types of gradient descent algorithms. They can be classified by two methods mainly:

* **On the basis of data ingestion**
  1. **Full Batch Gradient Descent Algorithm**: Uses whole data at once to compute the gradient
  2. **Stochastic Gradient Descent Algorithm:** Take a sample while computing the gradient

**Ref:** [**https://www.analyticsvidhya.com/blog/2017/03/introduction-to-gradient-descent-algorithm-along-its-variants/**](https://www.analyticsvidhya.com/blog/2017/03/introduction-to-gradient-descent-algorithm-along-its-variants/)

**Gradient Descent in Practice I - Feature Scaling:**

* To work gradient Decent more effectively and fast, feature scaling is to done if features are in diff scale

Eg: House size(sqft):2000 , No of Bedrooms=3

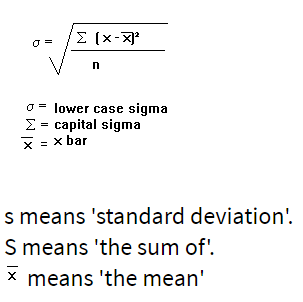
The way to prevent this is to modify the ranges of our input variables so that they are all roughly the same.

Ideally: **−1 ≤  *x*(*i*)​ ≤ 1 or 0.5≤  *x*(*i*)​ ≤ 0.5**

Two techniques to help with this are **feature scaling** and **mean normalization**.

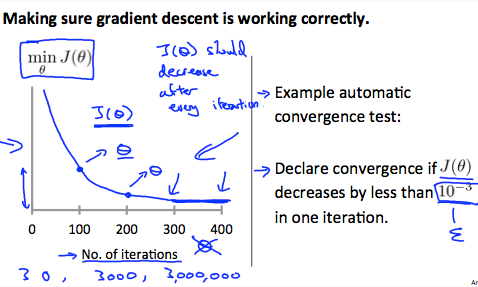
* **Feature scaling** involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just
* **Mean normalization** involves subtracting the average value for an input variable from the values for that input variable resulting in a new average value for the input variable of just zero. To implement both of these techniques, adjust your input values as shown in this formula:



* Where *μi* ​ is the **average** of all the values for feature (i) and *si*​  is the range of values (max - min), or *si*​  is the standard deviation
* Standard deviation:

# Gradient Descent in Practice II - Learning Rate

**Debugging gradient descent.** Make a plot with cost function value J(θ) on the x-axis. Now plot the cost function, J(θ) over the number of iterations of gradient descent. If J(θ) ever increases, then you probably need to decrease α.



It has been proven that if learning rate α is sufficiently small, then J(θ) will decrease on every iteration.\

To summarize:

If *α* is too small: slow convergence.

If *α* is too large: may not decrease on every iteration and thus may not converge.

**Normal Equation:**

Normal Equation" method, we will minimize J by explicitly taking its derivatives with respect to the θj ’s, and setting them to zero. This allows us to find the optimum theta without iteration.

The normal equation formula is given below:



