Gradient Descent

JAHAN GHOFRANIHA, PH.D.

Gradient

Gradient is simply another word for the derivative or the rate of change of a function but in the context of a multivariable functions.

The gradient is a vector (multiple derivatives) that has the following properties:

- The maximum rate of change in the direction of one variable when the rest of the variables are kept constant.
- is equal to zero at the maximum or minimum of the function.

A function with a single variable has only one derivative such F(x) will have only dF/dx.

Gradient

If the function is multi-variate then there is a derivative for very single variable, i.e. if F(x,y,z) then we will have dF/dx, dF/dy and dF/dz.

One question that can come up is if we are considering the change of F in the direction of x (dF/dx) then what is happening with y and z at this time?

The answer to this is that we keep the other two variables constant and only consider the changes of F with respect to x only.

We repeat this process with every independent variable (y and z in this example). For this reason we use the partial derivative notation: $\frac{\partial F}{\partial x}$

Gradient

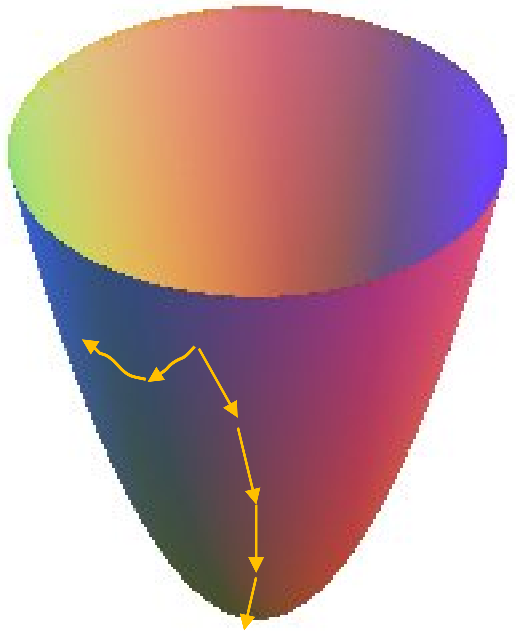
The gradient in this case will have three component, hence the vector representation: $(\frac{\partial F}{\partial x}\ \frac{\partial F}{\partial y}\ \frac{\partial F}{\partial z})$.

A vector in three dimensions can move in all directions, and the function F will change as x, y and z are changed, however, the function changes the maximum amount in the direction of the gradient.

Gradient is direction to move and is different from the coordinates x, y and z. The gradient gives direction from one coordinate to the next in the direction where F changes the most.

Gradient Descent

- Gradient descent is the process of step-wise movement towards a minimum or a maximum.
- There are two scenarios when we are looking for minimum or maximum of a function.
- We either know what the function is explicitly or we are given sample points of the function (discrete sample points)



https://www.mathcurve.com/surfaces.gb/paraboloidrevolution/paraboloidrevolution.shtml

Gradient Descent

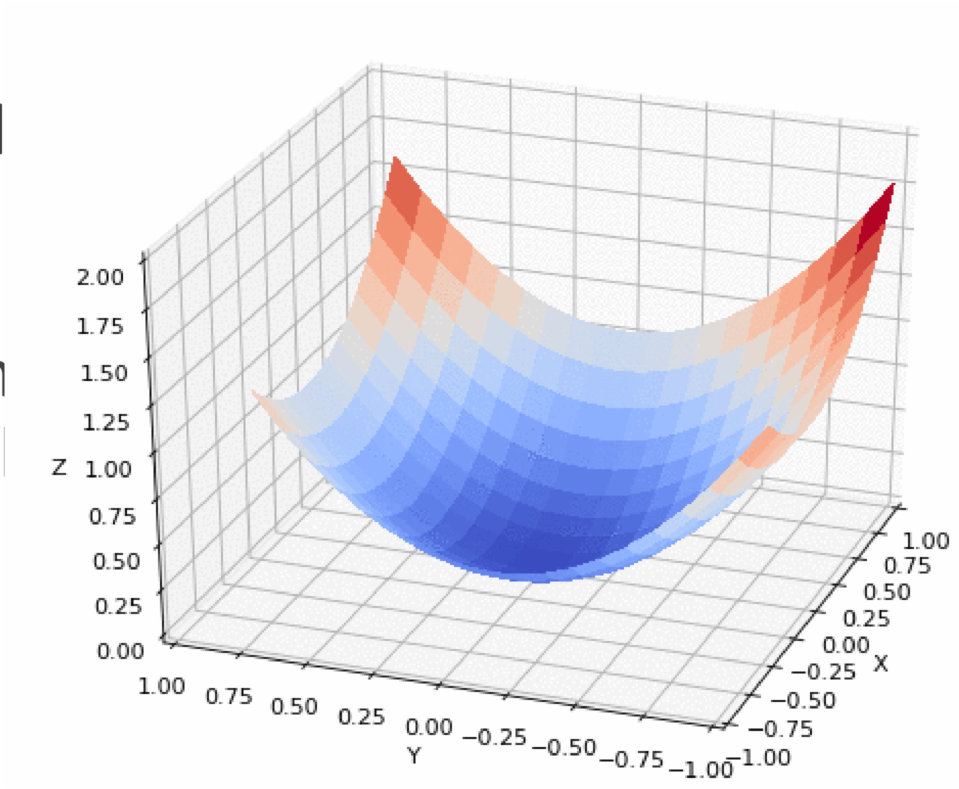
In the first case, we can calculate the gradient analytically and set it to zero to find out the minimum (maximum is also implied as maximization can always be formulated as a minimization problem).

In the second case, i.e. in most cases, we need to search for the min/max numerically.

This means that we need to calculate the gradient vector one step at a time to move closer to the minimum since the function is not known explicitly ahead of time but at every point, we should move in the direction of gradient descent to move closer to the minimum.

Gradient Descent

Depending on the smoothness of the function formed by the data points and the speed of movement towards the minimum the search for the minimum may fail or instead of a global minimum the optimization could end up at a local minimum.



Finding the minimum through gradient descent: (image from: https://github.com/pvigier/gradient-descent)

Gradient Descent Math (objective function)

The following example formulates the optimization problem once as an explicit function where we can calculate the minimum analytically and the second method is to solve numerically with gradient descent.

Revisiting the EOQ example:

EOQ Formula

The Economic Order Quantity formula is calculated by minimizing the total cost per order by setting the first order derivative to zero. The components of the formula that make up the total cost per order are the cost of holding inventory and the cost of ordering that inventory. The key notations in understanding the EOQ formula are as follows:

Gradient Descent (EOQ regression)

Components of the EOQ Formula:

D: Annual Quantity Demanded

Q: Volume per Order

S: Ordering Cost (Fixed Cost)

C: Unit Cost (Variable Cost)

H: Holding Cost (Variable Cost)

i: Carrying Cost (Interest Rate)

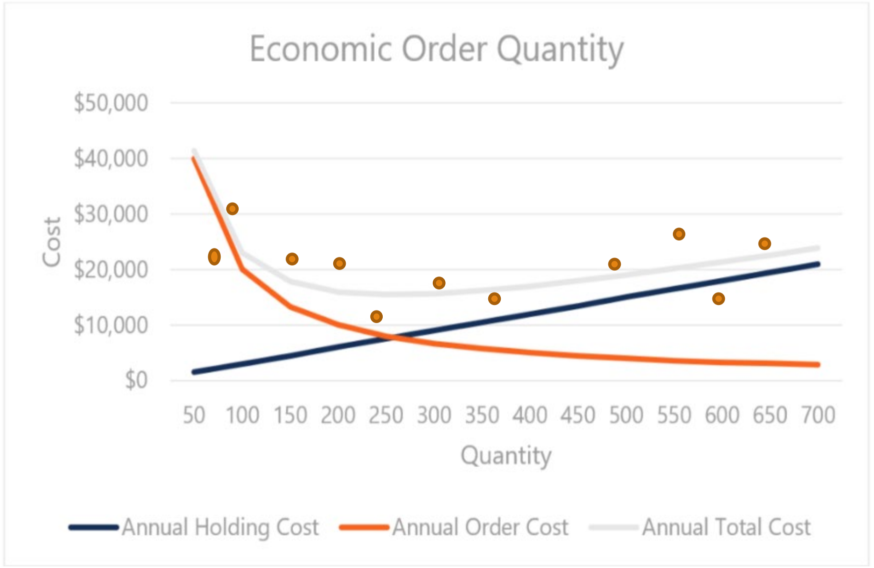
$$\# orders = \frac{D}{Q}$$

$$\text{ordering cost } = \frac{D}{Q} S$$

$$\text{holding cost } = \frac{Q}{2} * H$$

$$\text{total cost } = \text{TC} = \frac{D}{Q} S + \frac{Q}{2} H$$

$$EOQ = \frac{dTC}{dQ} = \sqrt{\frac{2SD}{H}}$$



https://corporatefinanceinstitute.com/resources/knowledge/finance/what-is-eoq-formula/

Gradient Descent

The derivation of EOQ formula is based on some basic assumptions. If we take a data-driven approach to this problem and look at the actual number of orders and the cost of inventory etc, we will be dealing with a nonlinear regression problem.

In this case our cost function will be to calculate the error between the actual model and the historical data.

The cost function/objective function is the function that we want to minimize. If we use a parametric model, the objective function based on the sum of the squared error is:

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

$$gradient:$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i}^{m} \left(y_i - \hat{y}_i \right) . \frac{\partial \hat{y}_i}{\partial \theta_j}$$

Gradient descent update formula

$$\theta_{new} = \theta_{old} - \eta \underbrace{\frac{\partial J(\theta)}{\partial \theta_j}}_{gradient}$$

$$\eta = \text{ Learning rate}$$

Gradient Descent

Learning rate is a constant that determines how much we should move in the direction of gradient vector towards the minimum.

If the value of learning rate is large the algorithm will overshoot and could oscillate around the minimum.

Too small a value can cause slow convergence. This is normally a tuning parameter (hyper parameter) that needs to be determined at the design time.

This flavor of the gradient descent that is also the baseline method is called the batch gradient descent since for every step towards the minimum all the samples are used to calculate the gradient.

This is evident from the summation over all samples as shown in the gradient descent formula.

Gradient descent is a first order optimization algorithm as the gradient is the equivalent of a first order derivative. There are also modified versions of gradient descent that use second order derivatives.

Gradient Descent Variations

Batch gradient descent(BGD): uses all the sample in each step to calculate the next step towards minimum.

This could increase the complexity of the computation when the number of samples are large.

Stochastic gradient descent (SGD): Only uses one sample for each step.

SGD is normally very noisy as only uses the information in a single sample to move towards the minimum. This could be a minor problem as long the algorithm converges fast.

Mini-batch gradient descent uses n-samples instead of one sample as in SGD.

Mini-batch is a compromise between the two (BGD and SGD) and is used quite often in optimization.

As mentioned second order algorithms and use of momentum such as Adams algorithm are the dominant variations of gradient descent in practice.