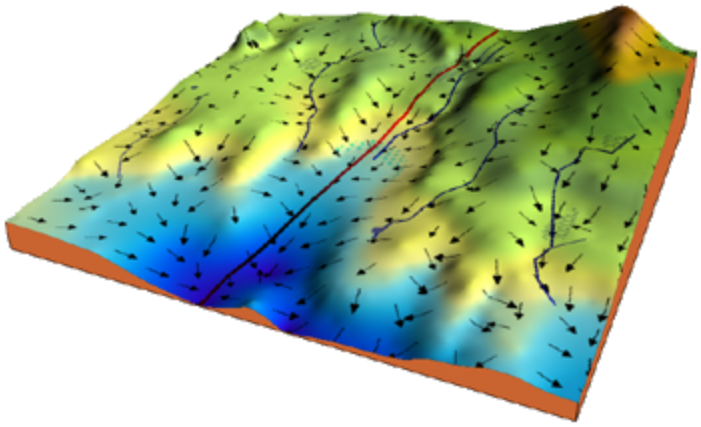
What is Gradient descent :

Gradient descent is an optimization algorithm that's used to find a local minimum of a differentiable function when training a machine learning model. It's based on a cost function and tweaks its parameters iteratively moving in the direction of steepest descent as defined by the negative of the gradient to minimize a given function to its local minimum.

You start by defining the initial parameter's values and from there gradient descent uses calculus to iteratively adjust the values so they minimize the given cost-function.

Consider the 3-dimensional graph below in the context of a cost function. Our goal is to move from the mountain in the top right corner (high cost) to the dark blue sea in the bottom left (low cost). The arrows represent the direction of steepest descent (negative gradient) from any given point–the direction that decreases the cost function as quickly as possible.



Starting at the top of the mountain, we take our first step downhill in the direction specified by the negative gradient. Next we recalculate the negative gradient (passing in the coordinates of our new point) and take another step in the direction it specifies. We continue this process iteratively until we get to the bottom of our graph, or to a point where we can no longer move downhill–a local minimum. imagesource.

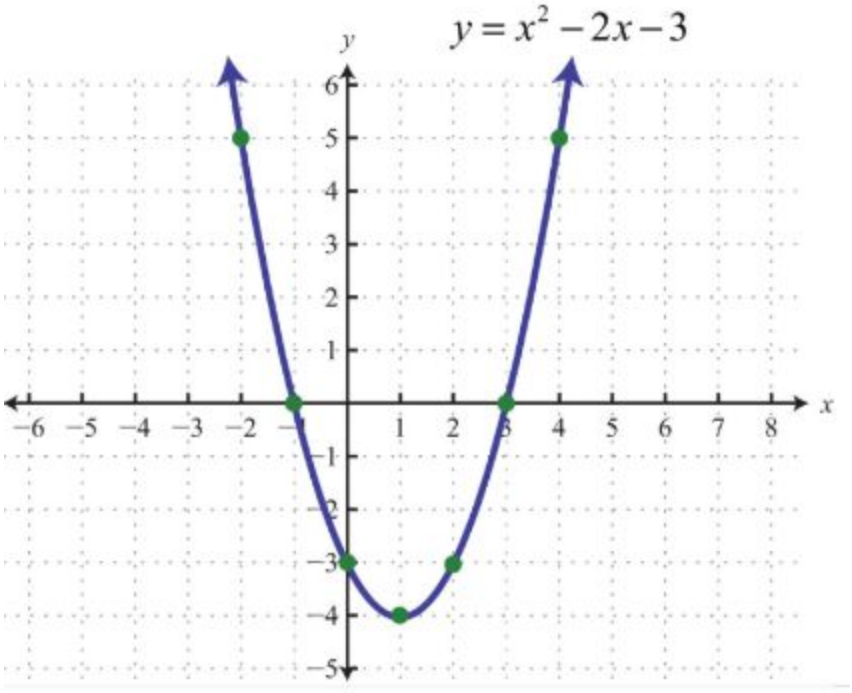
Purpose of gradient descent :

Gradient descent is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible using traditional slope-intercept form

y= mx+b

where m and b are the variables our algorithm will try to “learn” to produce the most accurate predictions.
x represents our input data and
y represents our prediction.

Gradient, in plain terms means slope or slant of a surface. So gradient descent literally means descending a slope to reach the lowest point on that surface. Let us imagine a two dimensional graph, such as a parabola in the figure below.



A parabolic function with two dimensions (x,y)

In the above graph, the lowest point on the parabola occurs at x = 1. The objective of gradient descent algorithm is to find the value of “x” such that “y” is minimum. “y” here is termed as the objective function that the gradient descent algorithm operates upon, to descend to the lowest point.
It is important to understand the above before proceeding further.

Steps to implement Gradient Descent

Gradient descent uses derivatives to actually decide whether to increase or decrease the weights in order to increase or decrease any objective function.

Primarily we shall be dealing with two concepts from calculus :

- Power Rule
- Chain Rule

Here we will be using Power rule and Partial derivative to calculate the derivative of a variable raised to a power.

If we have a function like

$$f(x) = x^n$$

, then

$$\frac{\partial f(x)}{\partial x} = nx^{n-1}$$

Example : Find the derivative of the function f(x) w.r.t. x where

$$f(x) = 3x^5$$

$$\frac{\partial f(x)}{\partial x} = 15x^4$$

Partial derivatives which says that if there is a function of two variables, then to find the partial derivative of that function w.r.t to one variable, treat the other variable as constant. This will be more clear with an example:

Partial Derivatives

Say for instance, we have a function:

$$f(x, y) = x^4 + y^7$$

partial derivative of the function w.r.t 'x' will be :

$$\frac{\partial f}{\partial x} = 4x^3 + 0$$

treating 'y' as a constant

And partial derivative of the function w.r.t 'y' will be :

$$\frac{\partial f}{\partial y} = 0 + 7y^6$$

treating 'x' as a constant

The Gradient Descent steps :

1. First, we take a function we would like to minimize, and very frequently it will be Mean Squared Errors function.
2. We identify parameters, such as m and b in the regression function and we take partial derivatives of MSE with respect to these parameters. This is the most crucial and hardest part. Each derived function can tell which way we should tune parameters and by how much.
3. We update parameters by iterating through our derived functions and gradually minimizing MSE. In this process, we use an additional parameter learning rate which helps us define the step we take towards updating parameters with each iteration. By setting a smaller learning rate we make sure our model wouldn't jump over a minimum point of MSE and converge nicely.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad \text{where} \quad \hat{y}_i = mx_i + b$$

$$\frac{\partial f}{\partial m} = \frac{1}{n}\sum_{i=1}^{n} -2x_i(y_i - (mx_i + b))$$

Partial derivative with respect to m

4. Update the gradient function by plugging in the parameter values.
5. Calculate the step sizes for each feature as : step size = gradient * learning rate.
6. Calculate the new parameters as : new params = old params -step size
7. Repeat steps 3 to 6 until gradient is almost 0.

The “learning rate” mentioned above is a flexible parameter which heavily influences the convergence of the algorithm. Larger learning rates make the algorithm take huge steps down the slope and it might jump across the minimum point thereby missing it. So, it is always good to stick to low learning rate such as 0.01. It can also be mathematically shown that gradient descent algorithm takes larger steps down the slope if the starting point is high above and takes baby steps as it reaches closer to the destination to be careful not to miss it and also be quick enough.

Conclusion :

Gradient descent is one of the most commonly used algorithms in Machine Learning on continuous data like (sales, prices etc) to reduce the error thereby minimizing cost function to its local minimum.

Glossary Terms:

Gradient :

"A gradient measures how much the output of a function changes if you change the inputs a little bit." — Lex Fridman (MIT)

A gradient simply measures the change in all parameters/weights with regard to the change in error. You can also think of a gradient as the slope of a function. The higher the gradient, the steeper the slope and the faster a model can learn. But if the slope is zero, the model stops learning. In

mathematical terms, a gradient is a partial derivative with respect to its inputs.

Cost Function :

It is a function that measures the performance of a Machine Learning model for given data. Cost Function quantifies the error between predicted values and expected values and presents it in the form of a single real number. Depending on the problem Cost Function can be formed in many different ways. The purpose of Cost Function is to be either:

- Minimized - then returned value is usually called cost, loss or error. The goal is to find the values of model parameters for which Cost Function return as small number as possible.
- Maximized - then the value it yields is named a reward. The goal is to find values of model parameters for which returned number is as large as possible.

For algorithms relying on Gradient Descent to optimize model parameters, every function has to be differentiable.

$$Cost = \frac{1}{N}\sum_{i=1}^{N}(Y' - Y)^2$$

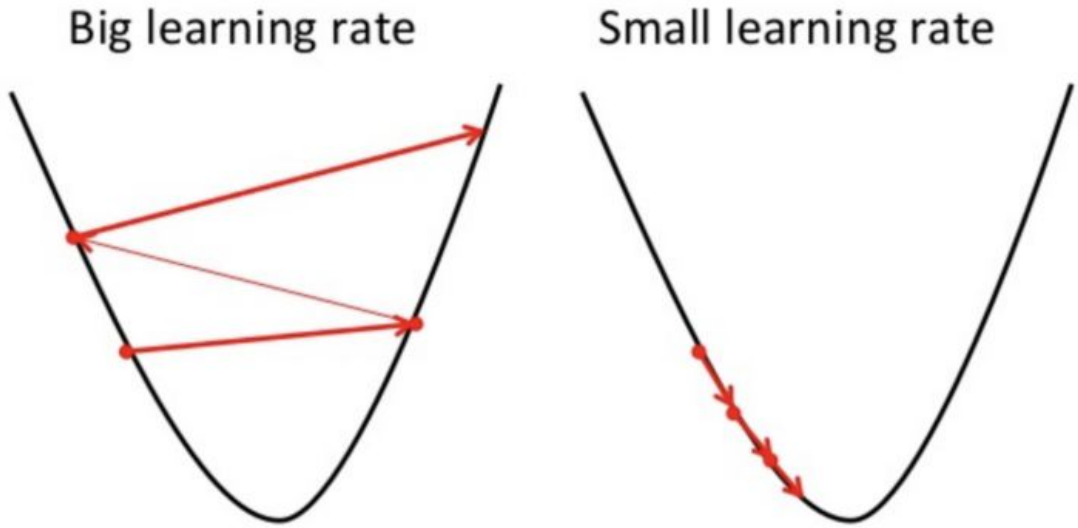
Learning Rate

The size of these steps is called the learning rate. With a high learning rate we can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. With a very low learning rate, we can confidently move in the direction of the negative gradient since we are recalculating it so frequently. A low learning rate is more precise, but calculating the gradient is time-consuming, so it will take us a very long time to get to the bottom.

Importance of the Learning Rate

How big the steps are gradient descent takes into the direction of the local minimum are determined by the learning rate, which figures out how fast or slow we will move towards the optimal weights.

For gradient descent to reach the local minimum we must set the learning rate to an appropriate value, which is neither too low nor too high. This is important because if the steps it takes are too big, it may not reach the local minimum because it bounces back and forth between the convex function of gradient descent (see left image below). If we set the learning rate to a very small value, gradient descent will eventually reach the local minimum but that may take a while (see the right image).

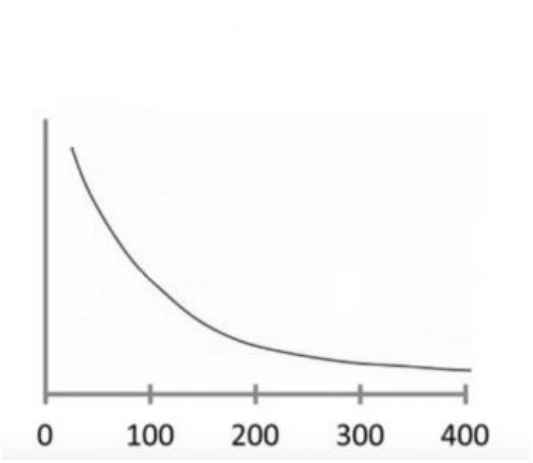


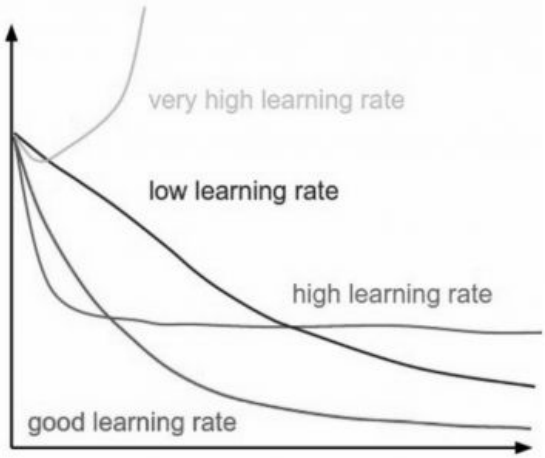
So, the learning rate should never be too high or too low for this reason. You can check if you're learning rate is doing well by plotting it on a graph.

HOW TO MAKE SURE IT WORKS PROPERLY

A good way to make sure gradient descent runs properly is by plotting the cost function as the optimization runs. Put the number of iterations on the x-axis and the value of the cost-function on the y-axis. This helps you see

the value of your cost function after each iteration of gradient descent, and provides a way to easily spot how appropriate your learning rate is. You can just try different values for it and plot them all together. The left image below shows such a plot, while the image on the right illustrates the difference between good and bad learning rates.





If gradient descent is working properly, the cost function should decrease after every iteration.

When gradient descent can't decrease the cost-function anymore and remains more or less on the same level, it has converged. The number of iterations gradient descent needs to converge can sometimes vary a lot. It can take 50 iterations, 60,000 or maybe even 3 million, making the number of iterations to convergence hard to estimate in advance.

There are some algorithms that can automatically tell you if gradient descent has converged, but you must define a threshold for the convergence beforehand, which is also pretty hard to estimate. For this reason, simple plots are the preferred convergence test.

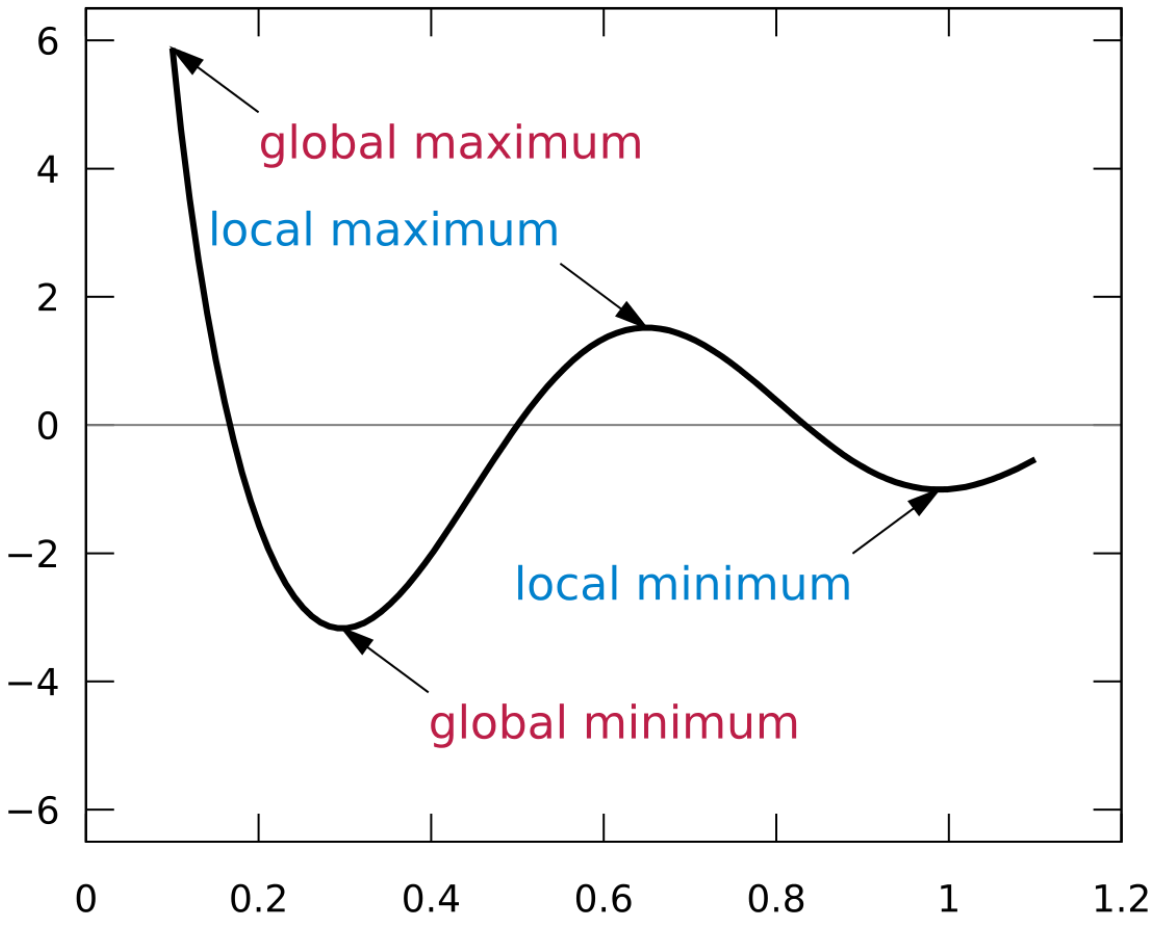
Another advantage of monitoring gradient descent via plots is it allows us to easily spot if it doesn't work properly, for example if the cost function is increasing. Most of the time the reason for an increasing cost-function when using gradient descent is a learning rate that's too high.

If the plot shows the learning curve just going up and down, without really reaching a lower point, try decreasing the learning rate. Also, when starting out with gradient descent on a given problem, simply try 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, etc., as the learning rates and look at which one performs the best.

Derivative

is a term that comes from calculus and is calculated as the slope of the graph at a particular point. The slope is described by drawing a tangent line to the graph at the point. So, if we are able to compute this tangent line, we might be able to compute the desired direction to reach the minima.

Local and Global Minima



To understand the difference between local minima and global minima, take a look at the figure above. The global minimum is the least value of a function while a local minimum is the least value of a function in a certain neighbourhood.