Gradient Descent

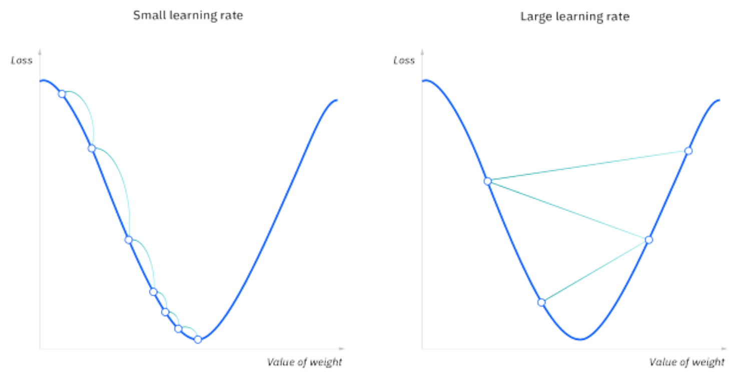
Gradient descent is a first order optimisation algorithm used to find a local minimum / maximum of a given function , it is most commonly used in machine learning models and neural networks . It is widely used as it can be used for minimizing   cost/loss functions.

Basic Idea-
1.Starts with an initial set of parameter value -are randomly assigned weights to the neural network .

2.Calculate the gradient of the cost function with respect to each parameter . The gradient indicates th direction of the cost function with respect to ach parameter and shows which parameter needs to be adjusted.

3.It then updates the parameters with new values by calculating them, each parameter is subtracted by a small step in the direction of the negative gradient [this size of the step i known as the learning rate]

4.the steps 2-3 are repeated until the cos function is minimized to the lowest achievable value or when some stopping criterion is met



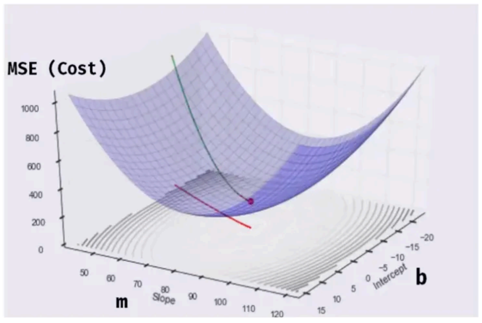
Basic Math behind Gradient Descent

1. Take some random values for the coefficients m and b and calculate the MSE (cost function).
2. Calculate the partial derivatives of MSE with respect to m and b
3. Set a value for the learning rate. And calculate the change in m and b using the following formula.
    - m = m — learning rate * ∂/∂m = m — α * ∂/∂m
    - b = b — learning rate * ∂/∂b = b — α * ∂/∂b
4. Use these values of m and b to calculate the new MSE.
5. Repeat steps 2, 3 and 4 until the changes in m and b do not significantly reduce the MSE (cost).

$$mse = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (mx_i + b)\right)^2$$

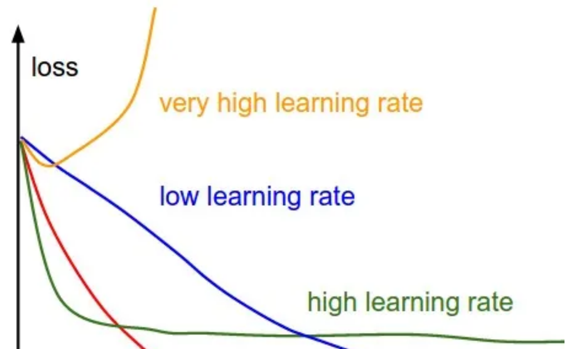
$$\partial/_{\partial m} = \frac{2}{\mathrm{n}}\sum_{i=1}^{n} -x_i\left(y_i - (mx_i + b)\right)$$

$$\partial/_{\partial b} = \frac{2}{\mathrm{n}}\sum_{i=1}^{n} -\left(y_i - (mx_i + b)\right)$$



Gradient descent graph as we go down the loss decreases

Loss graph
Closer to the x axis the better



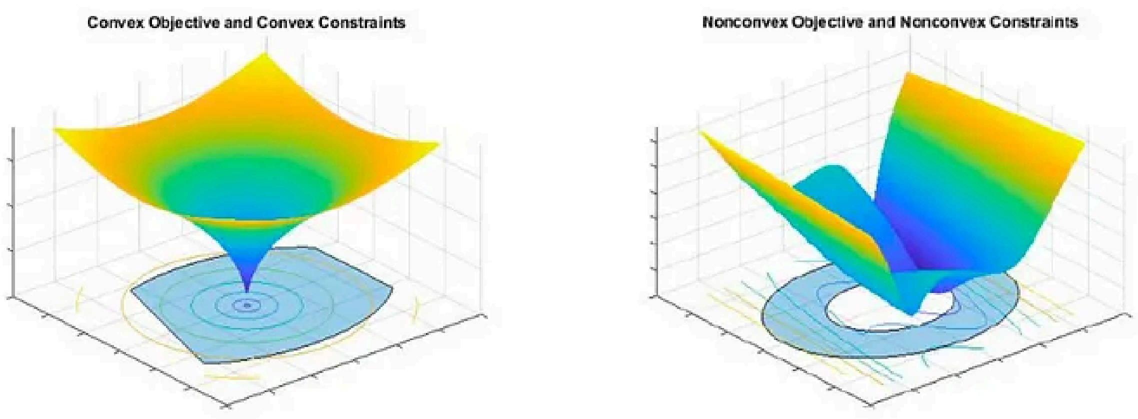
There are three types of gradient descent learning algorithms: batch gradient descent, stochastic gradient descent and mini-batch gradient descent.

1.Batch Gradient Descent
When we have n observations in a dataset and we use all of the n observations to update the values of m an b , it is called batch gradient descent , the whole dataset has to be available in the memory

2.Stochastic GD
In sgd the values of m and b are updates for wach observation in the dataset , due to the frequent updates of the coefficients it provides a good rate of improvement - more computationally expensive (from BGD)

3.Mini -BGD
Mini-batch gradient descent is a combination of the SGD and batch gradient descent. It splits the dataset into batches and the coefficients are updated at the end of each of these batches.

Convex functions have a unique global minimum. This means that if we want to optimize a convex function, we can be sure that we will always find the best solution by searching for the minimum value of the function. This makes optimization easier and more reliable.. Non-convex functions, on the other hand, can have multiple local minima, making optimization more challenging.


Convex Objective and Convex Constraints
Nonconvex Objective and Nonconvex Constraints

Convex & non-convex Functions in Machine Learning

When to use and where

For convex optimization problems, where the objective function is convex, gradient descent is guaranteed to find the global minimum. Examples of convex functions include:

- Linear functions: f(x) = ax + b
- Quadratic functions: f(x) = ax^2 + bx + c, where a > 0
- Exponential functions: f(x) = e^x
- Logarithmic functions: f(x) = log(x)

In these cases, gradient descent will reliably converge to the unique global minimum of the function**.**

For non-convex optimization problems, where the objective function has multiple local minima, gradient descent may struggle to find the global minimum. Examples of non-convex functions include:

Sinusoidal functions: f(x) = sin(x)

Absolute value functions: f(x) = |x|
Rational functions: f(x) = 1/x

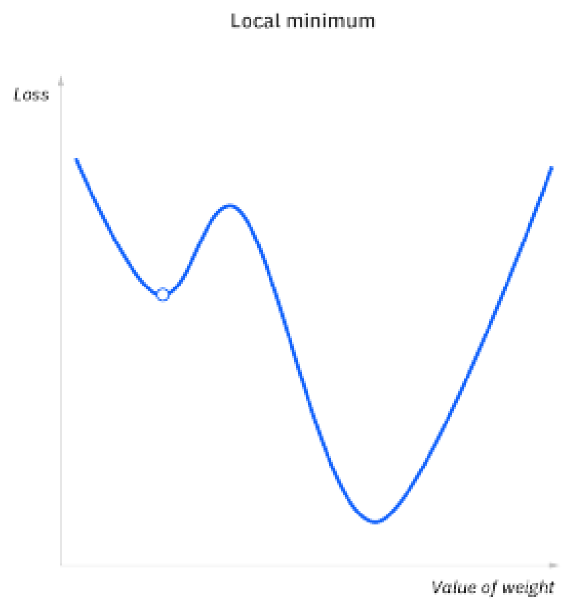
Issues with gd

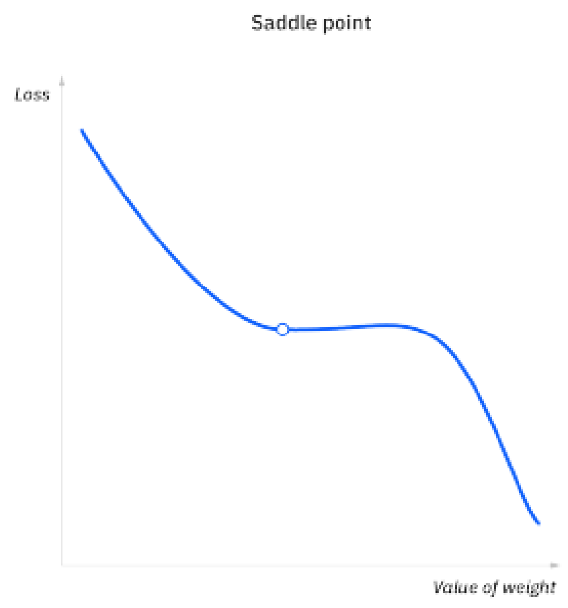
Local minima and saddle points:

For convex problems, gradient descent can find the global minimum with ease, but as nonconvex problems emerge, gradient descent can struggle to find the global minimum, where the model achieves the best results.

Vanishing and Exploding Gradients

- **Vanishing gradients:** This occurs when the gradient is too small. As we move backwards during backpropagation, the gradient continues to become smaller, causing the earlier layers in the network to learn more slowly than later layers. When this happens, the weight parameters update until they become insignificant—i.e. 0—resulting in an algorithm that is no longer learning.
- **Exploding gradients:** This happens when the gradient is too large, creating an unstable model. In this case, the model weights will grow too large, and they will eventually be represented as NaN. One solution to this issue is to leverage a dimensionality reduction technique, which can help to minimize complexity within the model





Reffrences

https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21
https://www.ibm.com/topics/gradient-descent
https://medium.com/geekculture/mathematics-behind-gradient-descent-f2a49a0b714f
https://www.infinitycodex.in/data-science-ss-106gradient-descent-and
https://rumn.medium.com/convex-vs-non-convex-functions-why-it-matters-in-optimization-for-machine-learning-39cd9427dfcc
https://www.analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/