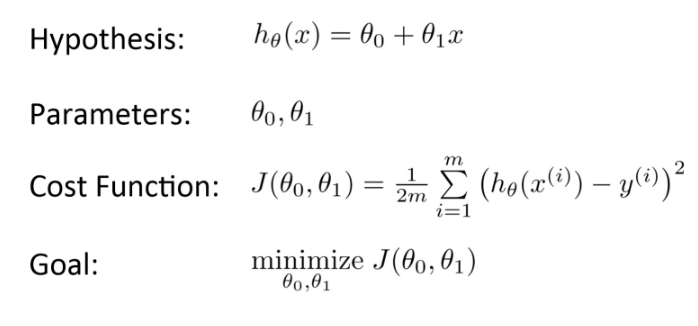
Gradient Descent

# Introduction

Imagine you’re lost in a dense forest with no map or compass. What do you do? You follow the path of steepest descent, taking steps in the direction that decrease the slope and brings you closer to your destination. Similarly, gradient descent is the go-to algorithm for navigating by iteratively adjusting them in the opposite direction of the gradient.

# What is a Cost Function?

Cost function is a function that measures the performance of a model for any given data. Cost Function quantifies the error between the predicted values and expected values and presents it in the formal of a single real number.

After making a hypothesis with initial parameters, we calculate the Cost Function. And with a goal to reduce the cost function, we modify the parameters by using the Gradient Descent Algorithm over the given data. Here’s the mathematical representation for it:

# What is Gradient Descent?

Gradient Descent is an optimization algorithm used in machine learning to minimize the cost function by iteratively adjusting parameters in the direction of the negative gradient, aiming to find the optimal set of parameters.

The cost function represents the discrepancy between the predicted output of the model and the actual output. The goal of gradient descent is to find the set of parameters that minimize this discrepancy and improves the model’s performance.

The algorithm operates by calculating the gradient of the cost function, which indicates the direction and magnitude of steepest ascent. However, since the objective is to minimize the cost function, gradient descent moves in the opposite direction of the gradient, known as the negative gradient direction.

By iteratively updating the model’s parameters in the negative gradient direction, gradient descent gradually converges towards the optimal set of parameters that yields the lowest cost. The learning rate, a hyperparameter, determines the step size taken in each iteration, influencing the speed and stability of converge.

Gradient descent can be applied to various machine learning algorithms, including linear regression, neural networks and support vectors machines. It provides a general framework for optimizing models by iteratively refining their parameters based on the cost function.

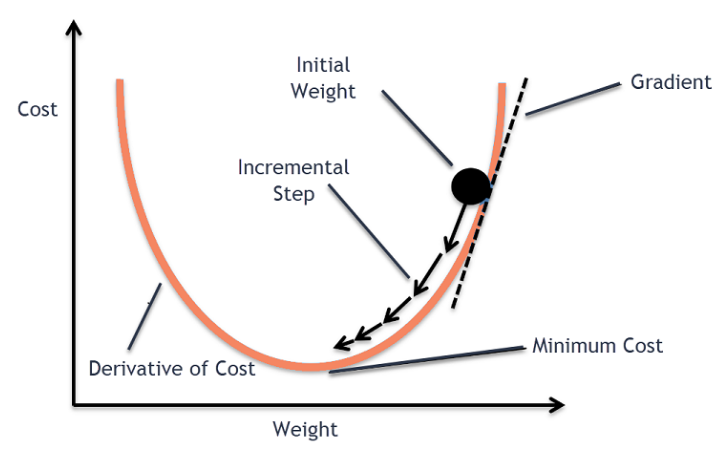
# Example of Gradient Descent

Let’s say you are playing a game where the players are at the top of the mountains, and they are asked to reach the lowest point of the mountain. Additionally, they are blindfolded. So, what approach to you think would have made you reach the lake?

The best way is to observe the ground and find where the land descents. From that position, take a step in the descending direction and iterate this process until we reach the lowest point.

Gradient descent is an iterative optimization algorithm for finding the local minimum of a function.

To find the local minimum of a function using gradient descent, we must take steps proportional to the negative of the gradient (move away from the gradient) of function at the current point. If we take steps proportional to the positive of the gradient (moving towards the gradient), we will approach a local maximum of the function, and the procedure is called Gradient Ascent.

Gradient descent was originally proposed by CAUCHY in 1847. It is also known as steepest descent.

The goal of gradient descent algorithm is to minimize the given function (say Cost Function). To achieve this goal, it performs 2 steps iteratively:

1. Compute the gradient (slope), the first order derivation of the function at that point.
2. A math equation with black text

   Description automatically generatedMake a step (move) in the direction opposite to the gradient, opposite direction of slope increases from the current point by alpha times the gradient at that point.

Alpha is called Learning rate --- a tuning parameter in the optimization process. It decides the length of the steps.

# How does gradient descent work?

1. Gradient descent is an optimization algorithm used to minimize the cost function of a model.
2. The cost function measure how well the model fits the training data and is defined based on the difference between the predicted and actual values.
3. The gradient of the cost function is derivative with respect to the model’s parameters and points in the direction of the steepest ascent.
4. The algorithm starts with an initial set of parameters and updates them in small steps to minimize the cost function.
5. In each iteration of the algorithm, the gradient of the cost function with respect to each parameter is computed.
6. The gradient tells us the direction of the steepest ascent, and by moving in the opposite direction, we can find the direction of the steepest descent.
7. The size of the step is controlled by the learning rate, which determines how quickly the algorithm moves towards the minimum.
8. The process is repeated until the cost function converges to a minimum, indicating that the model has reached the optimal set of parameters.
9. There are different variations of gradient descent, including batch gradient descent, stochastic gradient descent, and mini-batch gradient descent, each with its own advantages and limitations.
10. Efficient implementation of gradient descent is essential for achieving good performance in machine learning tasks. The choice of the learning rate and the number of iterations can significantly impact the performance of the algorithm.

# Types of Gradient Descent?

The choice of gradient descent algorithm depends on the problem at hand and the size of the dataset. Batch gradient descent is suitable for small datasets, while stochastic gradient descent is more suitable for large datasets. Mini-batch gradient descent is a good compromise between the two and is often used in practice.

**Batch Gradient Descent**

Batch Gradient Descent updates the model’s parameters using the gradient of the entire training set. It calculates the average gradient of the cost function for all the training examples and update the parameters in the opposite direction. Batch gradient descent guarantees convergence to the global minimum but can be computationally expensive and slow for large datasets.

**Stochastic Gradient Descent**

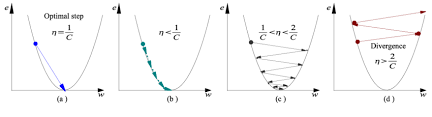
Stochastic Gradient Descent updates the model’s parameters using the gradient of one training example at a time. It randomly selects a training example, computes the gradient of the cost function for that example, and updates the parameters in the opposite direction. Stochastic gradient descent is computationally efficient and can converge faster than batch gradient descent. However, it can be noisy and may not converge to the global minimum.

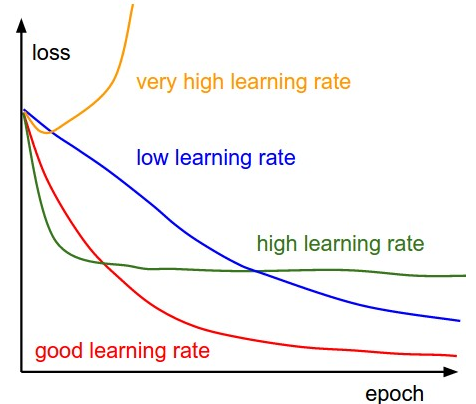
**Mini-Batch Gradient Descent**

Mini-Batch Gradient Descent updates the model’s parameters using the gradient of a small subset of the training set, known as a mini-batch. It calculates the average gradient of the cost function for the mini-batch and updates the parameters in the opposite direction. Mini-Batch Gradient Descent combines the advantages of both batch and stochastic gradient descent and is the most used method in practice. It is computationally efficient and less noisy than stochastic gradient descent, while still being able to converge to a good solution.

# Alpha --- The Learning Rate

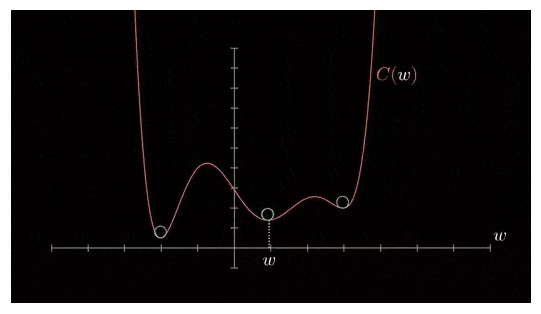
We have the direction we want to move in, now we must decide the size of the step we must take.

* If the learning rate is too high, we might **OVERSHOOT** the minima and keep bouncing, without reaching the minima.
* If the learning rate is too small, the training might turn out to be too long.

1. Learning rate is optimal, model converges to the minimum.
2. Learning rate is too small, it takes more time but converges to the minimum.
3. Learning rate is too high than the optimal values, it overshoots but converges.
4. Learning rate is verge large, it overshoots and diverges, move away from the minima, performance decrease on learning.

Note: As the gradient decreased while moving towards the local minima, the size of the step decreases. So, the learning rate (alpha) can be constant over the optimization and need not to be varied iteratively.

# Local Minima

The cost function may consist of many minimum points. The gradient may settle on any once of the minima., which depends on the initial point (i.e., initial parameter(theta)) and the learning rate. Therefore, optimization may converge to different points with different starting points and learning rate.

# Gradient Descent Implement in Python

# Challenges of Gradient Descent

While gradient descent is a powerful optimization algorithm, it can also present some challenges that can affect its performance. Some of these challenges include:

1. Local Optima --- Gradient descent can converge to local optima instead of global optimum, especially if the cost functions have multiple peaks and valleys
2. Learning Rate Selection --- The choice of learning rate can significantly impact the performance of gradient descent. If the learning rate is too high, the algorithm may overshoot the minimum, and if it is too low, the algorithm may take too long to converge.
3. Overfitting --- Gradient descent can overfit the training data if the model is too complex or the learning rate is too high. This can lead to poor generalization performance on new data.
4. Convergence Rate --- The converge rate of gradient descent can be slow for large datasets or high-dimensional spaces, which can make the algorithm computationally expensive.
5. Saddle Points --- In high-dimensional spaces, the gradient of the cost function can have saddle points, which can cause gradient descent to get struck in a plateau instead of converging to a minimum.

To overcome these challenges, several variations of gradient descent have been deployed, such as adaptive learning rate methods, momentum-based methods, and second-order methods. Additionally, choosing the right regularization method, model, or architecture, and hyperparameters can also help improve the performance of gradient descent.