Gradient Descent Algorithm: How does it Work in Machine Learning?

BEGINNER MACHINE LEARNING MATHS PYTHON

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 decreases the slope and brings you closer to your destination. Similarly, gradient descent is the go-to algorithm for navigating the complex landscape of machine learning. It helps models find the optimal set of parameters by iteratively adjusting them in the opposite direction of the gradient. In this article, we'll take a deep dive into the world of gradient descent, exploring its different flavors, applications, and challenges. Get ready to sharpen your optimization skills and join the ranks of the machine learning elite!

This article was published as a part of the Data Science Blogathon.

Table of contents

What is a Cost Function?

It is a **function** that measures the performance of a model for any given data. **Cost Function** quantifies the error between predicted values and expected values and presents it in the form 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:

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} ($

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Source: Coursera

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 minimizes 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 convergence.

Gradient descent can be applied to various machine learning algorithms, including linear regression, logistic regression, neural networks, and support vector 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 a mountain, and they are asked to reach the lowest point of the mountain. Additionally, they are blindfolded. So, what approach do you think would make you reach the lake?

Take a moment to think about this before you read on.

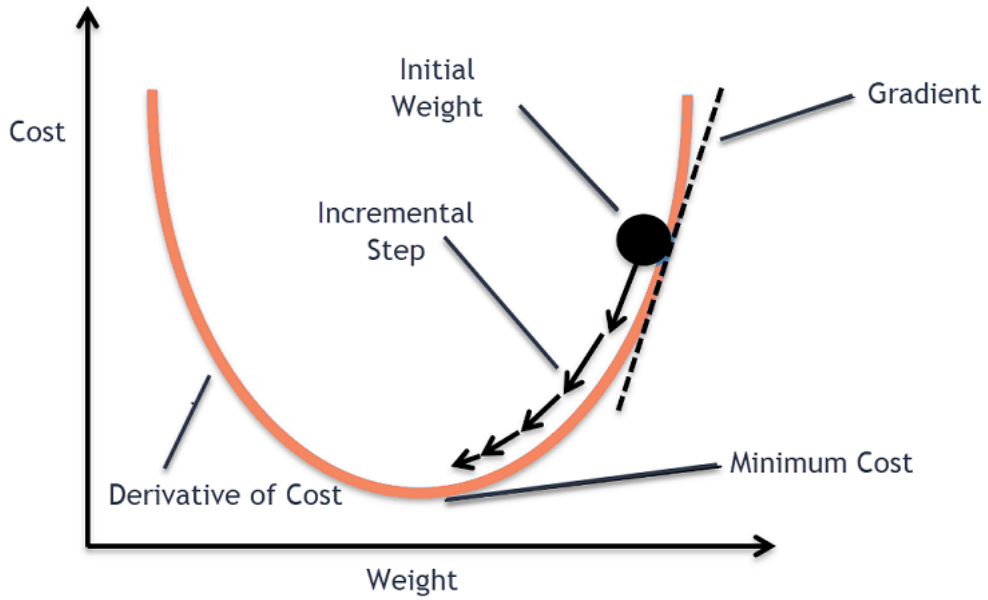
The best way is to observe the ground and find where the land descends. From that position, take a step in the descending direction and iterate this process until we reach the lowest point.

Finding the lowest point in a hilly landscape. (Source: Fisseha Berhane)

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 the 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.**

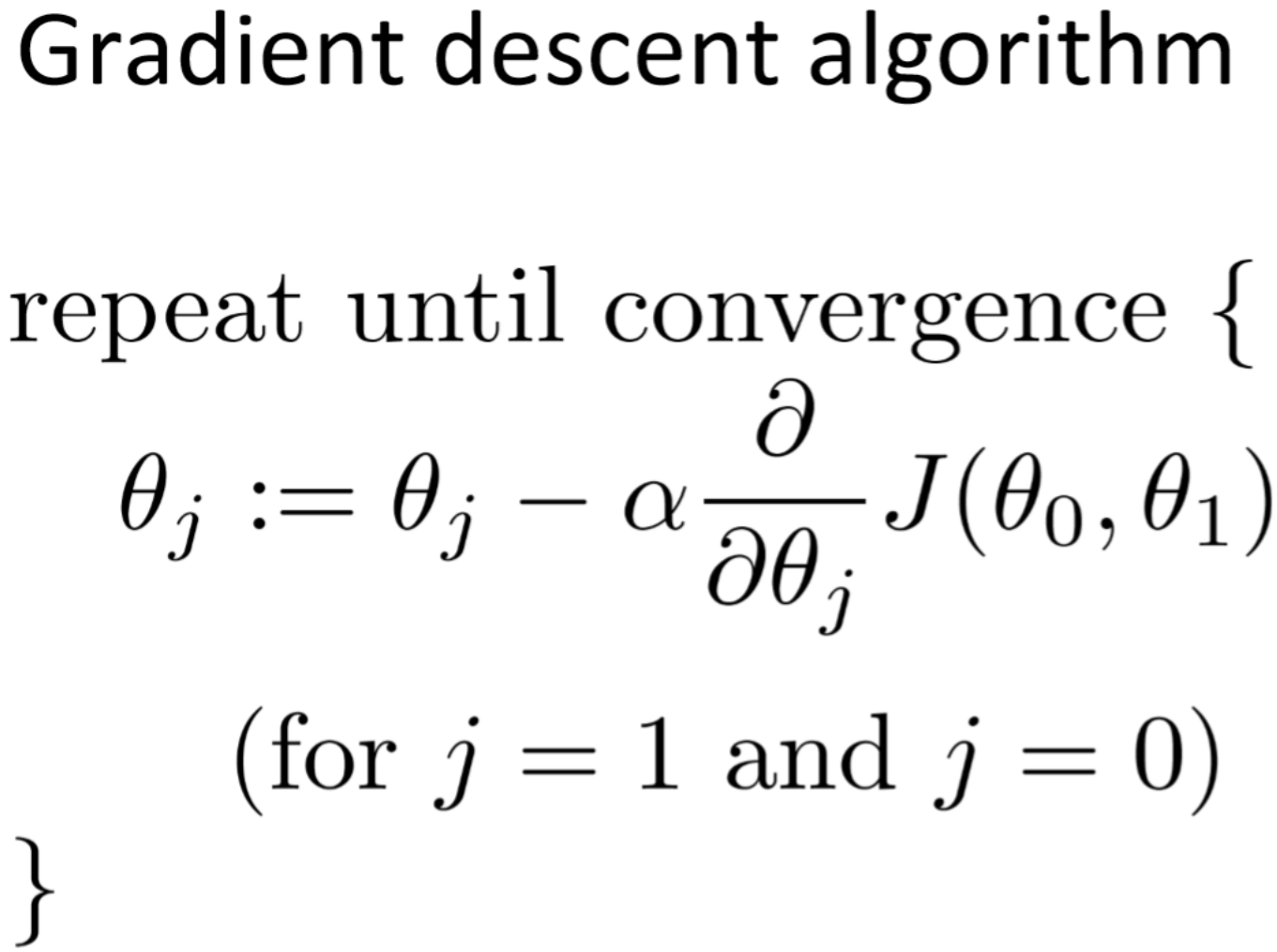
Gradient descent was originally proposed by **CAUCHY** in 1847. It is also known as steepest descent.



Source: Clairvoyant

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

1. **Compute the gradient** (slope), the first order derivative of the function at that point
2. **Make a step (move) in the direction opposite to the gradient**, opposite direction of slope increase from the current point by alpha times the gradient at that point



Source: Coursera

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. It is is an optimization algorithm used to minimize the cost function of a model.
2. The cost function measures 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 the 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 algorithm is more suitable for large datasets. Mini-batch 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 updates 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.

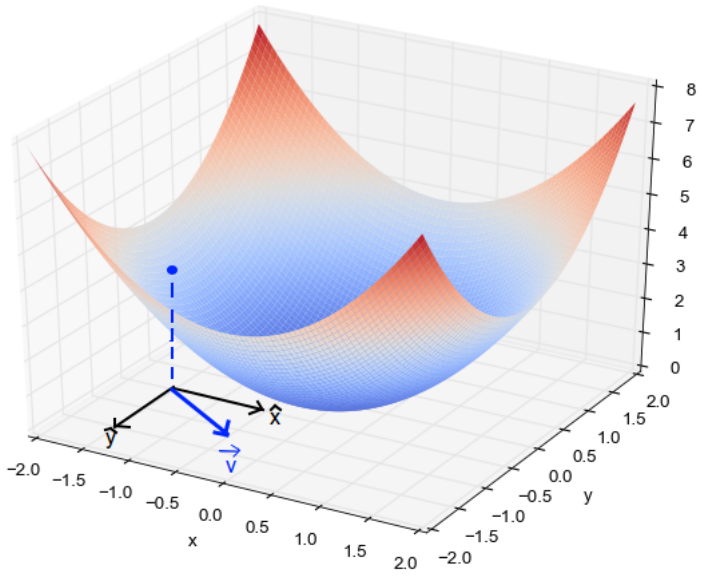
Mini-Batch Gradient Descent

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 algorithm combines the advantages of both batch and stochastic gradient descent, and is the most commonly 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.

> Gradient Descent and its Types

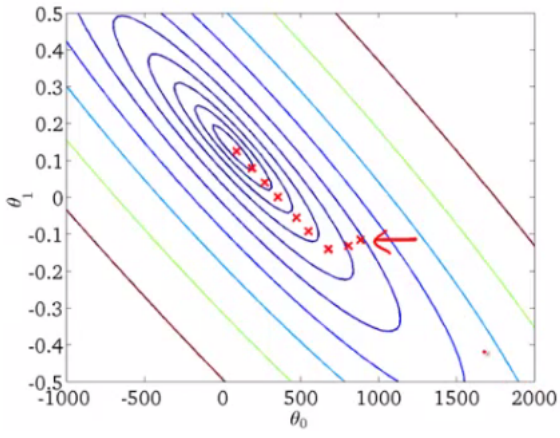
Plotting the Gradient Descent Algorithm

When we have a single parameter (theta), we can plot the dependent variable cost on the y-axis and theta on the x-axis. If there are two parameters, we can go with a 3-D plot, with cost on one axis and the two parameters (thetas) along the other two axes.



cost along z-axis and parameters(thetas) along x-axis and y-axis (source: Research gate)

It can also be visualized by using **Contours.** This shows a 3-D plot in two dimensions with parameters along both axes and the response as a contour. The value of the response increases away from the center and has the same value along with the rings. The response is directly proportional to the distance of a point from the center (along a direction).



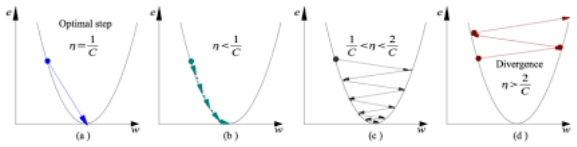
Gradient descent using Contour Plot. (source: Coursera)

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.

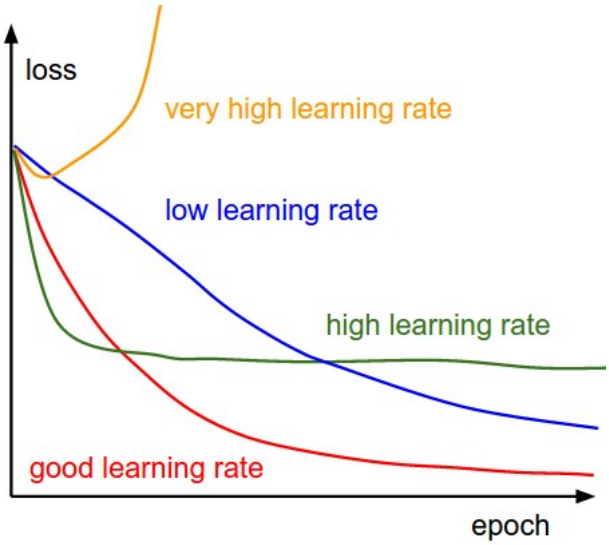
It must be chosen carefully to end up with local minima.

- 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



Source: Coursera

- Learning rate is optimal, model converges to the minimum
- Learning rate is too small, it takes more time but converges to the minimum
- Learning rate is higher than the optimal value, it overshoots but converges (1/C < η <2/C)
- Learning rate is very large, it overshoots and diverges, moves away from the minima, performance decreases on learning

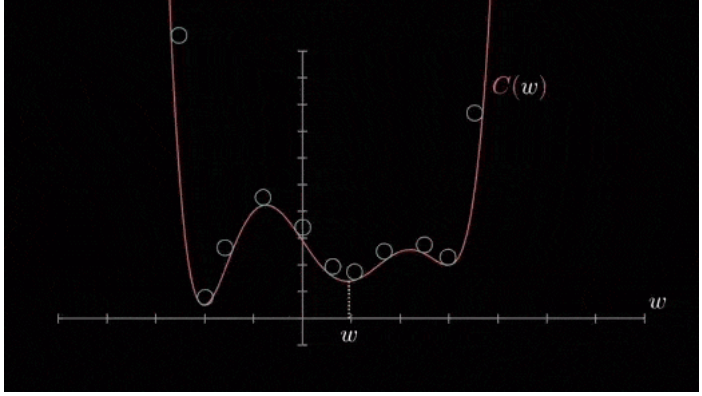


Source: researchgate

Note: *As the gradient decreases 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 be varied iteratively.*

Local Minima

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



Convergence of cost function with different starting points (Source: Gfycat)

Code Implementation of Gradient Descent in Python

```
def train(X, y, W, B, alpha, max_iters):

    '''
    Performs GD on all training examples,
    X: Training data set,
    y: Labels for training data,
    W: Weights vector,
    B: Bias variable,
    alpha: The learning rate,
    max_iters : Maximum GD iterations.'''

    dW = 0                                          # Weights gradient accumulator
    dB = 0                                          # Bias gradient accumulator
    m = X.shape[0]                                  # No. of training examples
    for i in range(max_iters):
        dW = 0                                      # Resetting the accumulators
        dB = 0
        for j in range(m):
                # 1. Iterate over all examples,
                # 2. Compute gradients of the weights and biases in w_grad and b_grad,
                # 3. Update dW by adding w_grad and dB by adding b_grad
        W = W - alpha * (dW / m)                    # Update the weights
        B = B - alpha * (dB / m)                    # Update the bias

    return W, B                                     # Return the updated weights and bias
```

Gradient Descent Algorithm

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 the global optimum, especially if the cost function has 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 convergence 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 stuck in a plateau instead of converging to a minimum.

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

> Gradient Descent in Linear Regression

Conclusion

Gradient descent is a powerful optimization algorithm used to minimize the cost function of a model by iteratively adjusting its parameters in the opposite direction of the gradient. While it has several variations and advantages, there are also some challenges associated with gradient descent that need to be addressed.

If you want to enhance your skills in gradient descent and other advanced topics in machine learning, check out the Analytics Vidhya Blackbelt program. This program provides comprehensive training and hands-on experience with the latest tools and techniques used in data science, including gradient descent, deep learning, natural language processing, and more. By enrolling in this program, you can gain the knowledge and skills needed to advance your career in data science and become a highly sought-after professional in this fast-growing field. Take the first step towards your data science career today!

Frequently Asked Questions

Q1. What are the three types of gradient descent?

A. The three types of gradient descent are batch gradient descent, stochastic gradient descent, and mini-batch gradient descent. These methods differ in how they update the model's parameters and the size of the data batches used in each iteration.

Q2. What is gradient descent in linear regression?

A. Gradient descent is an optimization algorithm used to minimize the cost function in linear regression. It iteratively updates the model's parameters by computing the partial derivatives of the cost function with respect to each parameter and adjusting them in the opposite direction of the gradient.

Q3. Which ML algorithms use gradient descent?

A. Several machine learning algorithms use gradient descent, including linear regression, logistic regression, neural networks, and support vector machines. These algorithms use gradient descent to optimize their respective cost functions and improve their performance on the training data.

Q4. What is gradient descent and backpropagation?

A. Gradient descent and backpropagation are two algorithms commonly used in training neural networks. Gradient descent updates the weights of the network by minimizing the cost function, while backpropagation calculates the gradient of the cost function with respect to each weight and propagates it backwards through the network.

Q5. What is gradient descent in simple terms?

A. Gradient descent is an optimization algorithm used to find the minimum of a function by iteratively adjusting the parameters in the opposite direction of the gradient. It is commonly used in machine learning to optimize the parameters of models and improve their performance on a given task.

Q6. What are the two types of gradient descent?

A. The two types of gradient descent are batch gradient descent and stochastic gradient descent. Batch gradient descent updates the model's parameters using the entire training set in each iteration, while stochastic gradient descent updates the parameters using only one training sample at a time.

Article Url - https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/

Crypto1