AdaGrad

Adaptive Gradient

Main Funda of this Optimizer is Learning Rate fix nahi rehta isme , We keep on changing or we keep on adapting based on situation

AdaGrad Performs Better compare to other Optimizer when:

1. ***Input Scale is very different***
   E.g CGPA & Salary , CGPA will be 0 to 10 & Salary can be anything,
   In This Kindoff Data Adagrad will be used, generally Ese dataset ko Normalize kardete hai

2. ***If Features are Sparse,***
   Sparse Means Most of the Value is 0,
   If Data will be sparse,

AdaGrad, *which stands for Adaptive Gradient Algorithm, is an optimization algorithm commonly used in the field of machine learning and deep learning. It was designed to adaptively adjust the learning rates of individual parameters, allowing for larger updates for infrequently occurring parameters and smaller updates for frequently occurring parameters. The key idea behind AdaGrad is to make larger updates for parameters with smaller historical gradients and smaller updates for parameters with larger historical gradients.*

Here's a brief overview of how AdaGrad works:

1. Learning Rates Adjustment:

- *AdaGrad adapts the learning rates for each parameter individually based on their historical gradients.*
- *Parameters that have large gradients in the past will have their learning rates reduced, and parameters with small gradients will have their learning rates increased.*

2. Accumulating Squared Gradients:

- *AdaGrad maintains an accumulated squared gradient for each parameter. This is done by summing the squares of the past gradients for each parameter.*
- *The update for each parameter is then divided by the square root of the accumulated squared gradient.*

3. Parameter Update:

- *The update for each parameter is calculated by multiplying the learning rate by the gradient for that*

parameter divided by the square root of the accumulated squared gradient.

4. Benefits:

- *AdaGrad is particularly useful when dealing with sparse data, where some features may have more frequent updates than others.*
- *It helps to overcome the challenge of manually tuning learning rates for each parameter.*

However, there are some issues with AdaGrad, such as the fact that the accumulated squared gradients keep growing during training, potentially causing the learning rates to become very small, and it may not perform well in non-convex optimization problems.

In practice, variations like RMSprop and Adam have been developed to address some of the limitations of AdaGrad while retaining its adaptive learning rate features.

AdaGrad mai Learning Rate kesath Khelte

E.g Gradient bada hai tou LR small krdo & if Gradient chota hai then Increase LR

Har parameter keliye alag LR set krte hai

Disadvantages of AdaGrad:

There is Big disadvantage of AdaGrad , because of which Neural Network mai use nahi hota , maybe Linear Regression mai use karskte hai

AdaGrad Gets closer to Solution but never converge , We can't Reach Global Minima

While AdaGrad has its advantages, it also comes with certain disadvantages that have led to the development of newer optimization algorithms. Here are some drawbacks of AdaGrad in the context of deep learning:

Learning Rate Decay:

AdaGrad adapts the learning rates based on the historical squared gradients. However, in deep learning scenarios, the accumulated squared gradients keep growing over time, leading to a continuous decrease in the effective learning rates.

This can cause the learning rates to become very small as training progresses, which may result in slow convergence or premature convergence to suboptimal solutions.

Memory Requirement:

AdaGrad maintains an accumulated squared gradient for each parameter, requiring additional memory to store these values.

For deep neural networks with a large number of parameters, the memory requirement can become a significant drawback. This may limit the size of the models that can be effectively trained using AdaGrad.

Non-Convex Optimization Challenges:

AdaGrad was originally designed with convex optimization in mind. In non-convex optimization scenarios, such as training deep neural networks, the accumulated squared gradients may lead to over-adaptation to noisy or irrelevant features.

This can result in poor generalization performance and make the algorithm sensitive to the choice of hyperparameters.

Lack of Momentum:

AdaGrad does not incorporate momentum, which is a technique that helps accelerate convergence and navigate through flat regions of the loss landscape.

Momentum-based algorithms, such as SGD with momentum, RMSprop, and Adam, have been developed to address this limitation and often outperform AdaGrad in deep learning tasks.

Difficulty with Sparse Data:

AdaGrad is not well-suited for sparse data because it accumulates squared gradients over all time steps, which can lead to very small learning rates for infrequently occurring parameters.

This issue can be problematic in scenarios where some features are more sparse or have less impact on the overall loss function.

Due to these limitations, researchers have proposed and developed alternative optimization algorithms, such as RMSprop and Adam, which aim to address the shortcomings of AdaGrad and provide better performance in the context of deep learning. These newer algorithms often include adaptive learning rate mechanisms along with momentum-like terms to improve convergence speed and generalization