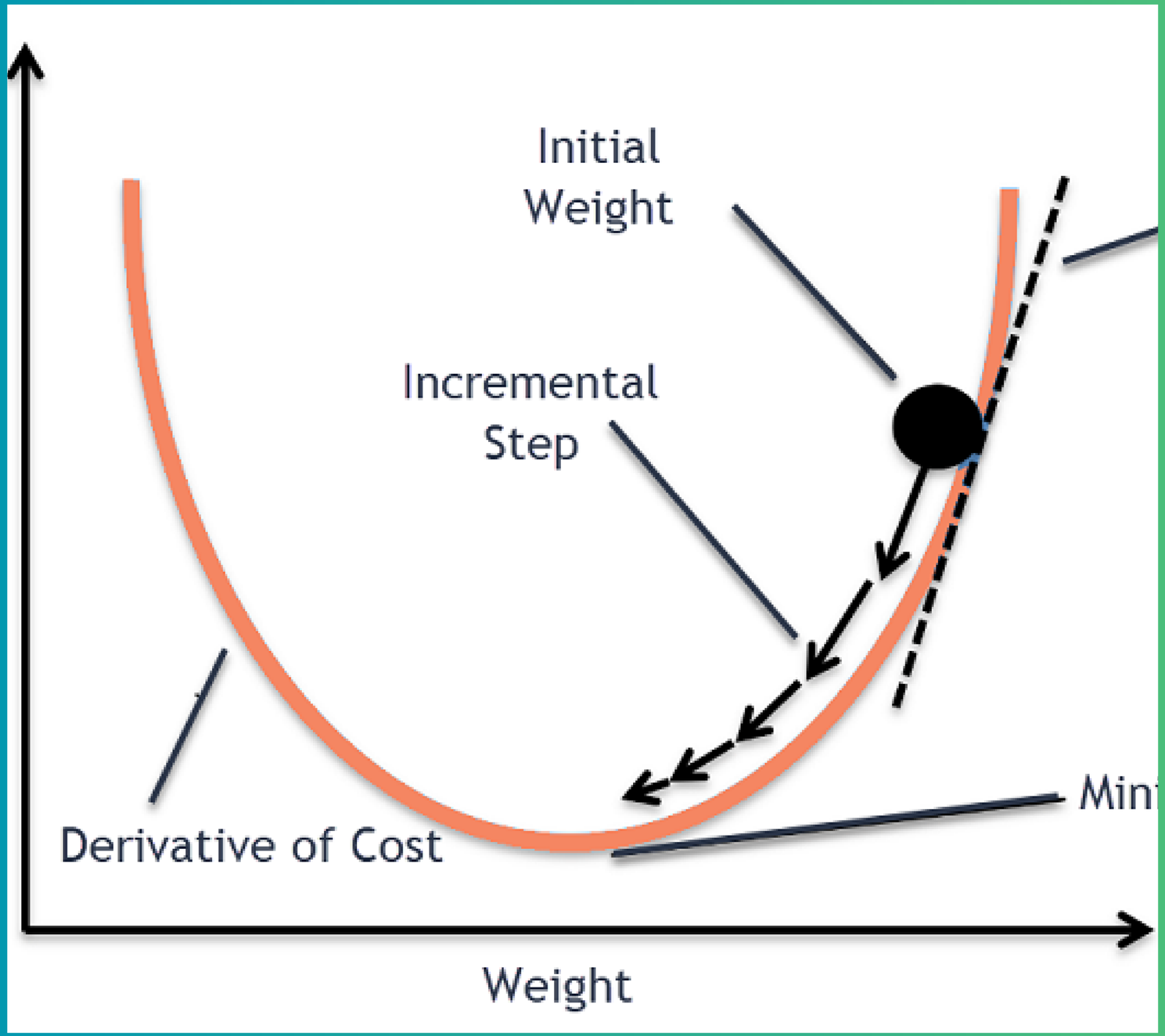
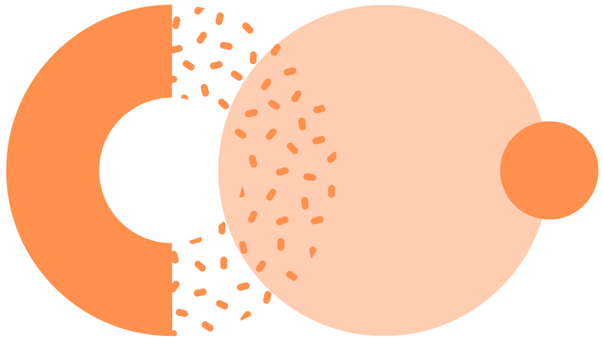


All About Optimizers



What is mean by gradients?

In the context of machine learning and neural networks, gradients represent the slope or rate of change of a function with respect to its parameters.

The loss function measures how well the model's predictions match the desired outputs, and the gradients of the loss function indicate how the loss changes with respect to each parameter.

During training, the gradients are computed by applying the chain rule of calculus to propagate the errors from the output layer to the earlier layers of the neural network. This process, known as backpropagation

What is an Optmizer?

- Optimizer is an algorithm or method used to adjust the model parameters of a model in order to reduce the loss function or maximize the objective function.
- Optimize iterativly optimizers the model parameters to improve the model performance.
- They determine the direction and magnitude of the parameters updates based on the gradient of the loss or objective function.
- **Types of optimizers**
  - Gradient descent(batch Gradient descent
  - Stochastic GD
  - Mini Batch GD
  - Stochastic GD with momentum
  - Adagrad
  - Adadelta and RMSprop
  - Adam optimizer

Gradient Descent (Batch Gradient Descent):

- **Gradient Descent (GD), also known as Batch Gradient Descent, calculates the gradient of the cost function using the entire training dataset.**

Advantages:

- Converges to the global minimum
- Can guarantee convergence for convex optimization problems.
- Provides a smooth and stable convergence path.

Disadvantages:

- Computationally expensive for large datasets as it requires processing all training examples in each iteration.
- Memory-intensive for large models due to the need to store gradients for all parameters

Stochastic Gradient Descent (SGD)

- **Stochastic Gradient Descent (SGD) randomly selects a single training example for computing the gradient.**
- **It is computationally efficient but introduces more noise into the optimization process.**

Advantages:

- Computationally efficient, especially for large datasets.
- Can escape local minima or plateaus more easily due to the stochastic nature.
- Suitable for online learning scenarios where new examples continuously arrive.

Disadvantages:

- Noisy updates can cause convergence to oscillate and slow down.
- Requires careful tuning of the learning rate to balance convergence speed and stability.

Mini-Batch Gradient Descent

- **Mini-Batch Gradient Descent computes the gradient using a small subset or mini-batch of training examples.**
- **It combines advantages of GD and SGD.**

Advantages:

- Faster convergence compared to SGD due to more stable updates.
- Efficient computation using parallel processing capabilities of modern hardware.

Disadvantages:

- Not suitable for very small datasets where the mini-batch size becomes too small.

Stochastic Gradient Descent with Momentum

- **Stochastic Gradient Descent with Momentum incorporates momentum, which accumulates historical gradients, to speed up convergence.**

Advantages:

- Accelerates convergence by using accumulated gradients to maintain a consistent direction.
- Efficient for large datasets due to the stochastic nature.

Disadvantages:

- Can converge to suboptimal solutions if the momentum coefficient is set too high.

Adagrad

- **Adagrad uses a adaptive learning rate methods that adjust the learning rate based on the history of gradients encountered during optimization.**

Advantages:

- Automatically adapts the learning rate for each parameter based on its historical gradients.
- Effective for handling sparse features or parameters with sparse updates.
- Suitable for non-convex problems and noisy or sparse datasets.

Disadvantages:

- Accumulation of squared gradients can lead to a very small learning rate(Diminishing of learning rates), making it difficult to escape shallow local minima or plateaus.
- May require tuning additional hyperparameters for optimal performance.

AdaDelta and RMSprop

- **Adadelta and RMSprop are both optimization algorithms that address the issue of diminishing learning rates in the Adagrad algorithm.**

Advantages:

- Addresses the diminishing learning rate issue of Adagrad by using a moving average of squared gradients.
- Provides better convergence behavior, especially for non-stationary gradients.
- Allows different parameters to have different learning rates based on their gradients.

Disadvantages:

- Does not consider the direction of gradients, which can result in oscillations or slow convergence in certain scenarios.
- May require additional hyperparameter tuning.

Adam Optimizer

- **Adam (Adaptive Moment Estimation) is an extension of RMSprop that combines the advantages of both AdaGrad and RMSprop. It includes momentum and adapts learning rates.**

Advantages:

- Efficient and effective on a wide range of deep learning tasks.
- Provides adaptive learning rates for each parameter, enabling faster convergence.
- Addresses the diminishing learning rate issue and handles sparse gradients.

Disadvantages:

- Requires tuning of hyperparameters such as learning rate, momentum, and decay rates.
- May suffer from noise in early training stages for some problems.

Meda Shabarish

Thank you!

Linkedin