It's easy!

LET'S LEARN ABOUT:

Optmization & Optmizers

Let's answer three simple Questions

What, How & Types

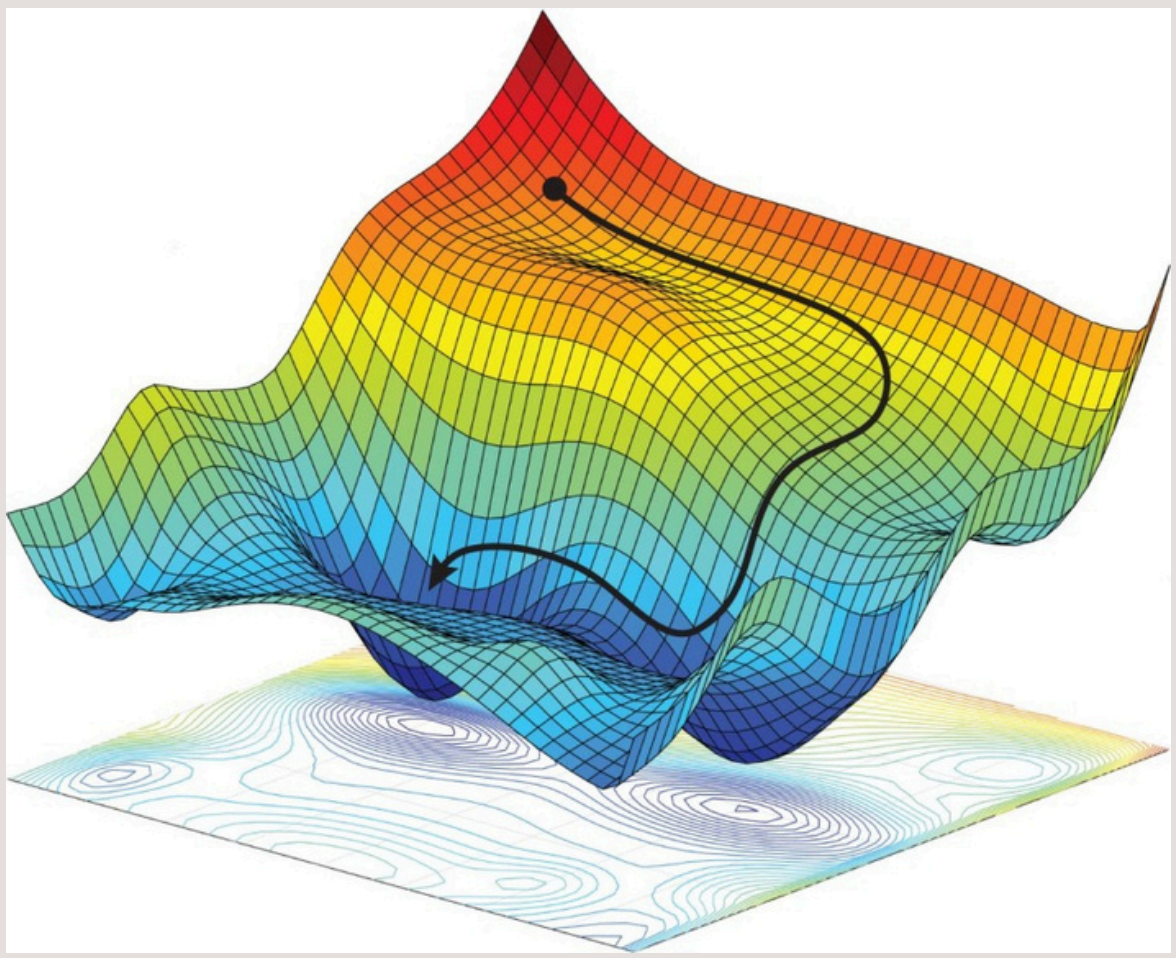
A series of posts purely based on ML, DL and LLM's

Presented by:
Sumanth Pagadala

What are Optmizers?

Optimizers in deep learning are algorithms designed to fine-tune the model's internal parameters, like weights and biases, to improve its performance.

When a model makes predictions, it often starts with random guesses, which means its predictions will have a lot of error initially. The goal is to adjust these parameters gradually so that the model becomes more accurate over time.



How Optmizers work?

OPTIMIZERS WORK BY ITERATIVELY ADJUSTING THE MODEL'S PARAMETERS TO MINIMIZE A LOSS FUNCTION, WHICH QUANTIFIES HOW FAR OFF THE MODEL'S PREDICTIONS ARE FROM THE ACTUAL VALUES. THIS PROCESS, CALLED "OPTIMIZATION," IS LIKE FINDING THE LOWEST POINT OF A LANDSCAPE (THE MINIMUM OF THE LOSS FUNCTION) BY FOLLOWING THE SLOPES DOWNWARD.

1. **INITIALIZE PARAMETERS:** THE MODEL STARTS WITH RANDOM WEIGHTS AND BIASES, WHICH ARE ITS INTERNAL PARAMETERS.

2. **COMPUTE LOSS:** THE MODEL MAKES PREDICTIONS WITH THESE INITIAL PARAMETERS, AND THE DIFFERENCE BETWEEN PREDICTIONS AND ACTUAL RESULTS IS CALCULATED USING A LOSS FUNCTION.

3. **CALCULATE GRADIENTS:** THE OPTIMIZER THEN CALCULATES GRADIENTS—ESSENTIALLY, THE DIRECTION AND RATE OF CHANGE OF THE LOSS FUNCTION WITH RESPECT TO EACH PARAMETER.

4. **UPDATE PARAMETERS:** USING THESE GRADIENTS, THE OPTIMIZER ADJUSTS THE PARAMETERS IN A DIRECTION THAT REDUCES THE LOSS. THE SIZE OF THESE ADJUSTMENTS IS DETERMINED BY A "LEARNING RATE,".

5. **REPEAT:** THIS PROCESS OF COMPUTING LOSS, CALCULATING GRADIENTS, AND UPDATING PARAMETERS IS REPEATED OVER MULTIPLE STEPS (OR "ITERATIONS") UNTIL THE MODEL’S PERFORMANCE PLATEAUS AT A MINIMUM LOSS.

Types of Optmizers

Stochastic Gradient Descent (SGD)

What: SGD is a basic form of gradient descent that updates model parameters for each individual training example, rather than averaging over a batch of samples.

How:

$$\theta = \theta - \eta \nabla J(\theta; x^{(i)}; y^{(i)})$$

Where: Suitable for large datasets or online learning, where computation of the full gradient is infeasible. It's often used in cases where faster, noisier updates are acceptable.

Types of Optmizers

Mini-Batch Gradient Descent

What: An extension of SGD that updates parameters using a small batch of samples instead of individual examples.

How:

$$\theta = \theta - \eta \frac{1}{m} \sum_{i=1}^{m} \nabla J(\theta; x^{(i)}; y^{(i)})$$

Where: Commonly used in practice since it balances the efficiency of computation and the stability of updates, reducing noise compared to pure SGD.

Types of Optmizers

Adagrad

What: Adagrad adapts the learning rate for each parameter, scaling it inversely with the square root of the sum of all past squared gradients.

How:

$$\theta = \theta - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla J(\theta)$$

Where: Useful for sparse data, such as text or image data, where certain parameters require larger updates than others.

Types of Optmizers

Adadelta

What: Adadelta is an extension of Adagrad that seeks to overcome its diminishing learning rate problem by using a decaying average of past squared gradients instead of the full sum.

How:

$$\theta = \theta - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla J(\theta)$$

Where: Often applied in models, especially RNNs, where maintaining a stable learning rate over many iterations is crucial for effective training.

Types of Optmizers

RMSprop

What: RMSprop also adapts learning rates like Adagrad but with a moving average of squared gradients to avoid the diminishing rate issue.

How:

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1 - \rho) g_t^2$$

Where: Often used for RNNs and in settings with non-stationary objectives, as it is good at handling such dynamic data.

Types of Optmizers

Adam (Adaptive Moment Estimation)

What: Adam is a powerful optimizer that combines the benefits of two techniques: RMSprop and momentum. It keeps track of both the average gradient (first moment) and the average squared gradient (second moment), allowing it to adapt the learning rate dynamically for each parameter.

How:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\theta = \theta - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

Where: Adam is widely used in deep learning across various types of neural networks, including CNNs and RNNs. It is particularly effective for models with noisy data or dynamic objectives, as it adjusts learning rates based on the data. Its adaptability makes it popular in both research and practical applications, from image recognition to NLP tasks.

Types of Optmizers

Adamx

What: Adamax is a variant of Adam that replaces the second moment (mean squared gradient) with the infinity norm (maximum absolute gradient). This can offer more stable behavior when dealing with sparse gradients or very high-dimensional data.

How:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$u_t = \max(\beta_2 u_{t-1}, |g_t|)$$

$$\theta = \theta - \frac{\eta}{u_t} m_t$$

Where: Adam is widely used in deep learning across various types of neural networks, including CNNs and RNNs. It is particularly effective for models with noisy data or dynamic objectives, as it adjusts learning rates based on the data. Its adaptability makes it popular in both research and practical applications, from image recognition to NLP tasks.

It's easy!

Congratulations!

You learnt a new topic today!

Save the post and follow me for more content on ML,DL and LLM's

SUMANTH PAGADALA