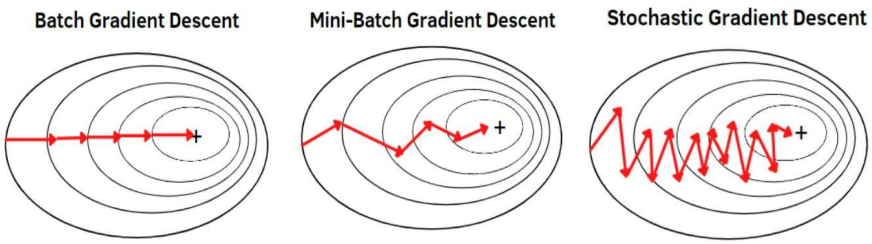
Fundamentals of Deep Learning :

Types of Gradient Descent



1) Standard Gradient Descent

⟶ First order optimization algorithm

⟶ Also called Full batch gradient descent

⟶ Minimizes a function by iteratively moving in the direction opposite to the gradient of the function at the current point.

⟶ Uses the entire dataset to compute the gradient of the loss function.

⟶ For every iteration, the algorithm **computes the gradient based on all training samples**.

Follows the rule :

$$W_{new} = W_{new} - \eta * \frac{\partial}{\partial w} J$$

Advantages :

+ Every update step utilizes gradient of the entire dataset (full-batch).
+ Converges in a smooth manner.

Disadvantages :

– Training becomes very slow.

– Computationally expensive, requires more memory.

2) Stochastic Gradient Descent

⟶ Optimization algorithm for neural networks

⟶ Stochastic gradient descent **computes the gradient using only a single randomly chosen data point** (or a small batch of data points) at each iteration.

⟶ The parameters are updated by moving them in the direction opposite to the gradient of the loss function with respect to those parameters.

⟶ Since a single data point is considered, the update steps are less informed, but happen more often.

Considering a single data point (xi,yi) :

$$W_{new} = W_{new} - \eta * \frac{\partial J(x_i,y_i)}{\partial w}$$

Advantages :

+ Convergence is faster
+ Every update step considers a single random data point, which is simpler.
+ Suitable for large datasets

Disadvantages :

– Since updates are based on single data points, they can be noisy.

– The step size should be small to avoid overshooting, but not reduce training time (the impact of learning rate parameter is critical)

3) Mini Batch Gradient Descent

⟶ Optimization algorithm using subset of data (mini-batch)

⟶ Is a compromise between stochastic gradient descent and full batch gradient descent

⟶ Gets **derivative for a subset of data points**, and uses it to update the weights

Considering a mini-batch with m samples ((x,y),(x1,y1)....(xm,ym)),
it follows the rule :

$$W_{new} = W_{new} - \eta * \left(\frac{1}{m} \sum_{i=1}^{m} \frac{\partial J(W(x_i, y_i))}{\partial x} \right)$$

Note :

1. The size of mini-batch, typically 32-64 rows, can be adjusted.
2. The mini-batches can be shuffled.
3. The learning rate (η) plays a crucial role in performance, and a learning rate schedule is used to adjust the value dynamically.

Data Shuffling

⟶ Splitting the dataset to balanced mini-batches, and shuffling them at each step.

$$\{FULL - BATCH\} = \left\{ \begin{matrix} \underline{BATCH - 1} \\ \underline{BATCH - 2} \\ \underline{BATCH - 3} \\ \underline{BATCH - 4} \\ \underline{BATCH - 5} \end{matrix} \right\}$$

⟶ Use one mini-batch to update the weights for one epoch, then use a random mini-batch for the next epoch.

⟶ Avoids cyclical movement, repeating the same path and aids convergence.