Learning Rate Optimizers

1. How Adagrad Optimizer Works
2. α_t^1 Value Calculation
3. α_t Values Calculation
4. Important Graph Explanation
5. Disadvantage

1. How Adagrad Optimizer:

- Up to now we knew some optimizers like GD, SGD, MBSGD. These all optimizers are used for reduce the loss function.
- The weight updating formula which is used in GD, SGD, MBSGD is:

$$\omega_{new} = \omega_{old} - \alpha \frac{\partial L}{\partial \omega_{old}} \dots\dots\dots\dots \quad (1)$$

- In Adagrad Optimizer the formula has changed as:

$$\omega_t = \omega_{t-1} - \alpha_t^1 \frac{\partial L}{\partial \omega_{old}} \dots\dots\dots\dots \quad (2)$$

- The main change in this optimization is, we are taking different learning rates for each neuron in each layer at each iteration. That's why the α is replaced with α_t^1.
  Here α_t = Present weight;
  t = Iteration number;
  α_{t-1} = Previous weight
- Adagrad Optimizer taking different alpha values to avoid DENSE, SPARSE, BOW problems.

1.1. α_t^1 Value Calculation:

- Now the task is finding α_t^1 value. the correspond formula is:

$$\alpha_t^1 = \frac{\alpha}{\sqrt{\alpha_t + \varepsilon}}$$

- Here ε is a small +ve value Which is added to α_t to avoid the zero values problem. let's consider a formula without ε.

$$\alpha_t^1 = \frac{\alpha}{\sqrt{\alpha_t}}$$

- If we consider the above formula. If $\alpha_t = 0$ then $\alpha_t^1 = \infty$ finally the $w_t = \infty$. to avoid this problem, we add a small positive value ε.

1.2. α_t Values Calculation:

- α_t is the sum of derivative of loss function from starting iteration to current iteration.

$$\alpha_t = \sum_{i=1}^{t} \left[\frac{\partial L}{\partial w_i}\right]^2$$

- Let's consider we calculate α_t for 3rd iteration then the formula become:

$$\alpha_3 = \sum_{i=1}^{3} \left[\frac{\partial L}{\partial w_i}\right]^2$$

- α_t value is always higher value, because it is the sum-up component of derivative functions. which means we are adding all derivative terms to calculate α_t.
- Whenever α_t value is high, then α_t^1 value will be decreased because α_t^1 is inversely proportional to $\sqrt{\alpha_t}$.

$$\alpha_t^1 \propto \frac{1}{\sqrt{\alpha_t}}$$

- α_t^1 decreases for every iteration then the weight also will decrease slowly if weights are decreased then cost function will coverage to minimum point.

1.3. Important Graph Explanation:

- In Adagrad Optimization the weights are decreasing slowly and finally converge to minima as shown in the graph.
- Initially the weights are very high and continuously those will decrease step by step.

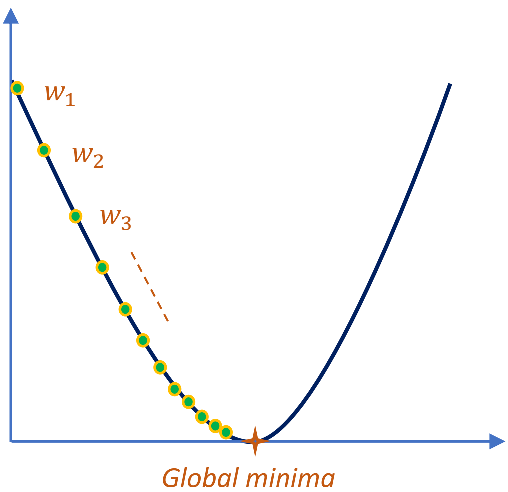

Figure 1 - Convex function

1.4. Disadvantage:

This Adagrad Optimizer is also having one disadvantage due to α_t. Actually α_t is some of squares of derivative terms.

- Consider if number of iterations are increasing α_t value will also increase. if α_t increasing drastically α_t^1 value will be very low. This situation can create approximate zero values of α_t^1.
- So, this optimization technique fails at a higher number of iterations of training.

2. Adadelta Optimizer:

- Adadelta is an advanced version of Adagrad Optimizer. It Addresses the issues which are in Adagrad.
- Adagrad has *Diminishing Learning Rates* problem that can resolve by Adadelta.
- The α_t is the main reason of *Diminishing Learning Rates* problem, the Adadelta Optimizer formula created by w_{avg} instead α_t.

$$\alpha_t^1 = \frac{\alpha}{\sqrt{w_{avg} + \varepsilon}}$$

$$w_{avg(t)} = \gamma \, w_{avg(t-1)} + (1-\gamma) \left[\frac{\partial L}{\partial w_t}\right]^2$$

- The main difference between. w_{avg} and α_t is high value. α_t value is always high when we compare α_t with w_{avg}.
- The main reason is w_{avg} don't have any summation terms and it haven't all iterations loss function terms. It has only one loss function term. so, it doesn't have any problem about high number of iterations.
- Here γ is a small fractional multiplier and it is 0.95 in most cases.