RMSProp Overview

RMSProp (Root Mean Square Propagation) is an optimization algorithm that adapts the learning rate for each parameter by maintaining a moving average of the squared gradients. It prevents oscillations and speeds up convergence, particularly useful for non-convex optimization.

Mathematical Equations

1. **Compute the Moving Average of Squared Gradients:** RMSProp uses an **Exponential Weighted Moving Average (EWMA)** to track the squared gradients:

$$S_{t+1} = \beta S_t + (1 - \beta)(\nabla L(\theta_t))^2$$

   - S_t: Accumulated moving average of squared gradients at time step t.
   - β: Decay rate (commonly $\beta = 0.9$).
   - $\nabla L(\theta_t)$: Gradient of the loss function with respect to parameters at t.

2. **Update Parameters:**

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{S_{t+1} + \epsilon}} \nabla L(\theta_t)$$

- η: Learning rate.
- ϵ: Small constant (e.g., 10^{-8}) added for numerical stability.

Step-by-Step Calculation Example

Setup

1. **Loss Function:** Let $L(\theta) = \frac{1}{2}\theta^2$, so $\nabla L(\theta) = \theta$.
2. **Initial Parameters:**
   - $\theta_0 = 2$.
   - $S_0 = 0$.
3. **Hyperparameters:**
   - $\eta = 0.1$ (learning rate),
   - $\beta = 0.9$,
   - $\epsilon = 10^{-8}$.
4. **Iterations:** Perform 5 updates.

Calculations

1. **Iteration 0:**
   - $\theta_0 = 2$,
   - $\nabla L(\theta_0) = 2$,
   - $S_1 = \beta S_0 + (1-\beta)(\nabla L(\theta_0))^2$,

   $$S_1 = 0.9 \cdot 0 + 0.1 \cdot 2^2 = 0.4$$

   - Update:

   $$\theta_1 = \theta_0 - \frac{\eta}{\sqrt{S_1 + \epsilon}} \nabla L(\theta_0)$$

   $$\theta_1 = 2 - \frac{0.1}{\sqrt{0.4 + 10^{-8}}} \cdot 2 \approx 1.683$$

2. **Iteration 1:**
   - $\theta_1 = 1.683$,
   - $\nabla L(\theta_1) = 1.683$,
   - $S_2 = \beta S_1 + (1-\beta)(\nabla L(\theta_1))^2$,

   $$S_2 = 0.9 \cdot 0.4 + 0.1 \cdot 1.683^2 \approx 0.5266$$

   - Update:

   $$\theta_2 = \theta_1 - \frac{\eta}{\sqrt{S_2 + \epsilon}} \nabla L(\theta_1)$$

$$\theta_2 = 1.683 - \frac{0.1}{\sqrt{0.5266 + 10^{-8}}} \cdot 1.683 \approx 1.449$$

3. **Iteration 2:**

   - $\theta_2 = 1.449$,
   - $\nabla L(\theta_2) = 1.449$,
   - $S_3 = \beta S_2 + (1 - \beta)(\nabla L(\theta_2))^2$,

   $$S_3 = 0.9 \cdot 0.5266 + 0.1 \cdot 1.449^2 \approx 0.6119$$

   - Update:

   $$\theta_3 = \theta_2 - \frac{\eta}{\sqrt{S_3 + \epsilon}} \nabla L(\theta_2)$$

   $$\theta_3 = 1.449 - \frac{0.1}{\sqrt{0.6119 + 10^{-8}}} \cdot 1.449 \approx 1.292$$

Similarly, continue for further steps.

Pros of RMSProp

1. **Adapts Learning Rate:**

- Automatically adjusts learning rates for each parameter, making it robust for optimization problems with varying gradient magnitudes.

2. **Handles Oscillations:**
   - Reduces oscillations in the optimization path by scaling gradients.
3. **Suitable for Non-Convex Problems:**
   - Works well for problems with non-convex loss landscapes.
4. **Efficient Memory Usage:**
   - Only requires a running average of squared gradients, making it computationally efficient.

Cons of RMSProp

1. **Hyperparameter Sensitivity:**
   - The choice of β and η can significantly affect convergence.
2. **Plateau Issues:**
   - May struggle with flat regions where gradients are very small.
3. **Bias in Early Iterations:**
   - The moving average S_t has an initial bias toward smaller values.

Key Takeaway:

RMSProp balances gradient scaling using EWMA of squared gradients, making it more robust than basic Gradient Descent, especially for non-convex problems or those with gradients of varying scales. However, it requires careful tuning of hyperparameters to achieve optimal performance.

explain RMS prop, maths equation using EWMA of gradients and use of this to update the parameters? show calculation using math equation . and generate table ? pros and cons?

RMSProp Explanation

RMSProp (Root Mean Square Propagation) is an adaptive learning rate optimization algorithm that uses an **Exponential Weighted Moving Average (EWMA)** of squared gradients to adjust the learning rate for each parameter dynamically. This helps address issues like oscillations in the optimization path and works well for non-convex problems.

Mathematical Equations

1. **EWMA of Squared Gradients:**

$$S_{t+1} = \beta S_t + (1 - \beta)(\nabla L(\theta_t))^2$$

- S_t: Moving average of squared gradients at time t.
- β: Decay rate (e.g., $\beta = 0.9$).
- $\nabla L(\theta_t)$: Gradient of the loss with respect to the parameter θ at time t.

2. **Parameter Update Rule:**

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{S_{t+1} + \epsilon}} \nabla L(\theta_t)$$

- η: Learning rate.
- ϵ: Small constant for numerical stability (e.g., 10^{-8}).

Example Calculation with Table

Setup

- Loss Function: $L(\theta) = \frac{1}{2}\theta^2$, so $\nabla L(\theta) = \theta$.
- Initial Parameter: $\theta_0 = 2$.
- Hyperparameters: $\beta = 0.9$, $\eta = 0.1$, $\epsilon = 10^{-8}$.
- Number of Iterations: 5.

Step-by-Step Calculations

Iteration (t)	θ_t	$\nabla L(\theta_t)$	S_t	θ_{t+1}
0	2.000	2.000	0.000	1.683
1	1.683	1.683	0.400	1.449
2	1.449	1.449	0.526	1.292
3	1.292	1.292	0.612	1.179
4	1.179	1.179	0.670	1.095

Detailed Calculation for Each Iteration

1. **Iteration 0:**

- $\nabla L(\theta_0) = 2$,
- $S_1 = \beta S_0 + (1-\beta)(\nabla L(\theta_0))^2 = 0.9 \cdot 0 + 0.1 \cdot 2^2 = 0.4$,
- $\theta_1 = \theta_0 - \frac{\eta}{\sqrt{S_1+\epsilon}} \nabla L(\theta_0)$,

$$\theta_1 = 2 - \frac{0.1}{\sqrt{0.4 + 10^{-8}}} \cdot 2 \approx 1.683.$$

2. **Iteration 1:**
   - $\nabla L(\theta_1) = 1.683$,
   - $S_2 = \beta S_1 + (1-\beta)(\nabla L(\theta_1))^2 = 0.9 \cdot 0.4 + 0.1 \cdot 1.683^2 \approx 0.5266$,
   - $\theta_2 = \theta_1 - \frac{\eta}{\sqrt{S_2+\epsilon}} \nabla L(\theta_1)$,

$$\theta_2 = 1.683 - \frac{0.1}{\sqrt{0.5266 + 10^{-8}}} \cdot 1.683 \approx 1.449.$$

3. **Iteration 2:**
   - $\nabla L(\theta_2) = 1.449$,
   - $S_3 = \beta S_2 + (1-\beta)(\nabla L(\theta_2))^2 = 0.9 \cdot 0.5266 + 0.1 \cdot 1.449^2 \approx 0.6119$,
   - $\theta_3 = \theta_2 - \frac{\eta}{\sqrt{0.6119+\epsilon}} \nabla L(\theta_2)$,

$$\theta_3 = 1.449 - \frac{0.1}{\sqrt{0.6119 + 10^{-8}}} \cdot 1.449 \approx 1.292.$$

Pros and Cons of RMSProp

Pros:

1. **Adapts Learning Rate:**
    - Automatically scales the learning rate for each parameter based on gradient history.
2. **Reduces Oscillations:**
    - Helps avoid oscillations in steep or narrow regions of the loss landscape.
3. **Effective for Non-Convex Problems:**
    - Performs well for deep learning problems with complex error surfaces.
4. **Efficient Memory Usage:**
    - Only requires maintaining the EWMA, making it computationally lightweight.

Cons:

1. **Hyperparameter Sensitivity:**
    - The decay rate (β) and learning rate (η) require careful tuning.
2. **Bias in Early Iterations:**
    - The moving average (S_t) has an initial bias toward zero.

3. **Not Globally Optimal:**
    - May settle in suboptimal minima due to its local nature.

Conclusion

RMSProp's use of EWMA helps adapt learning rates effectively, especially in challenging optimization problems. By balancing gradient scaling, it reduces oscillations and converges efficiently, making it one of the most popular optimizers in modern machine learning.