Understanding Workings of Neural Networks

Video 7: Common Optimization Techniques - Part 2

Analytics Vidhya

AdaGrad

AdaGrad

Adaptive Gradient Descent uses different learning rates for each iteration

AdaGrad

Adaptive Gradient Descent uses different learning rates for each iteration

- Parameters with **infrequent updates → Bigger updates**
- Parameters with **frequent updates → Smaller updates**

AdaGrad

1. Updated weights:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla L(w_t)$$

2. Modified learning rate:

$$\eta' = w_t - \frac{\eta}{\sqrt{G_t + \epsilon}}$$

ϵ = Small positive constant to ensure numerical stability

G_t = Sum of squares of the gradients upto time step t

Analytics Vidhya

AdaGrad

1. Updated weights

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla L(w_t)$$

2. Modified the learning rate

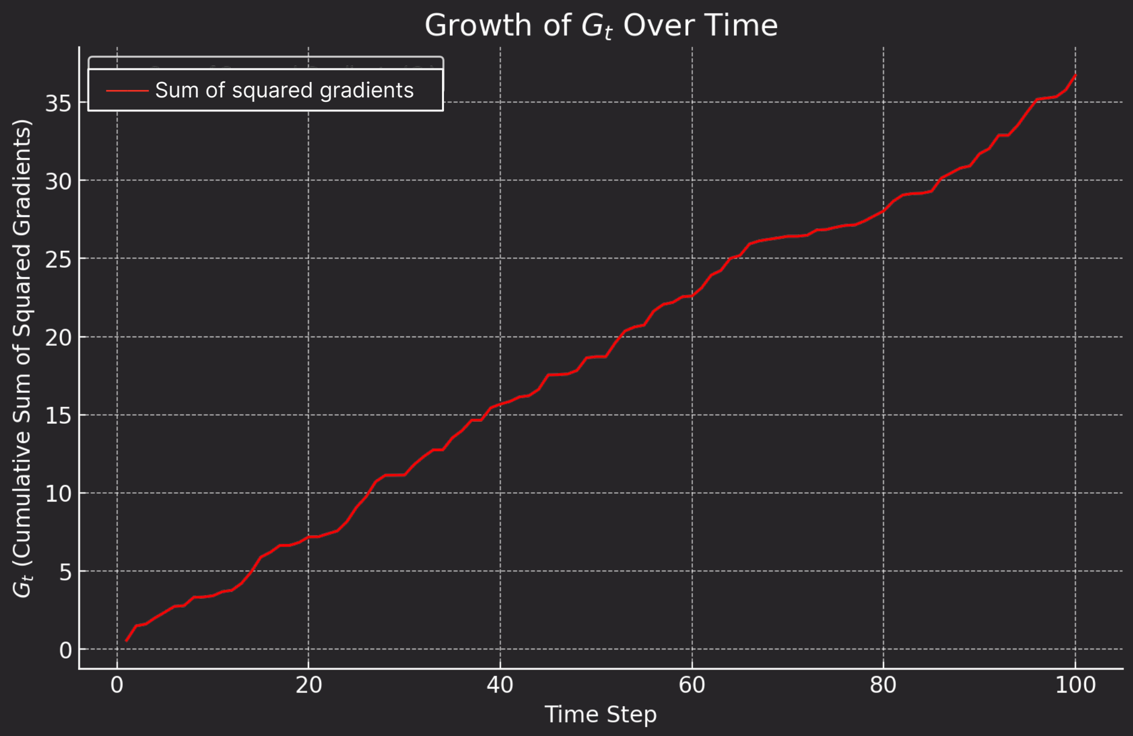
$$\eta' = w_t - \frac{\eta}{\sqrt{G_t + \epsilon}}$$

- If G_t is large → η' **will be small**
- If G_t is small → η' **will be large**

G_t = Sum of squares of the gradients upto time step t

AdaGrad - Drawback

Problem: AdaGrad may <u>reduce the learning rate aggressively</u> .



Learning Rate Decay

Learning Rate Decay

Learning rate decay is used to reduce the learning rate over time. It can be fixed or scheduled or dynamically adjusted.

$$\eta' = \frac{\eta_0}{1 + \text{Decay\_Rate} \times \text{Epoch\_Number}}$$

- **Decay_Rate** is the rate at which the learning rate decays,
- **Epoch_Number** is the current epoch number in the training process.

Learning Rate Decay

Learning rate decay is used to reduce the learning rate over time. It can be fixed or scheduled or dynamically adjusted.

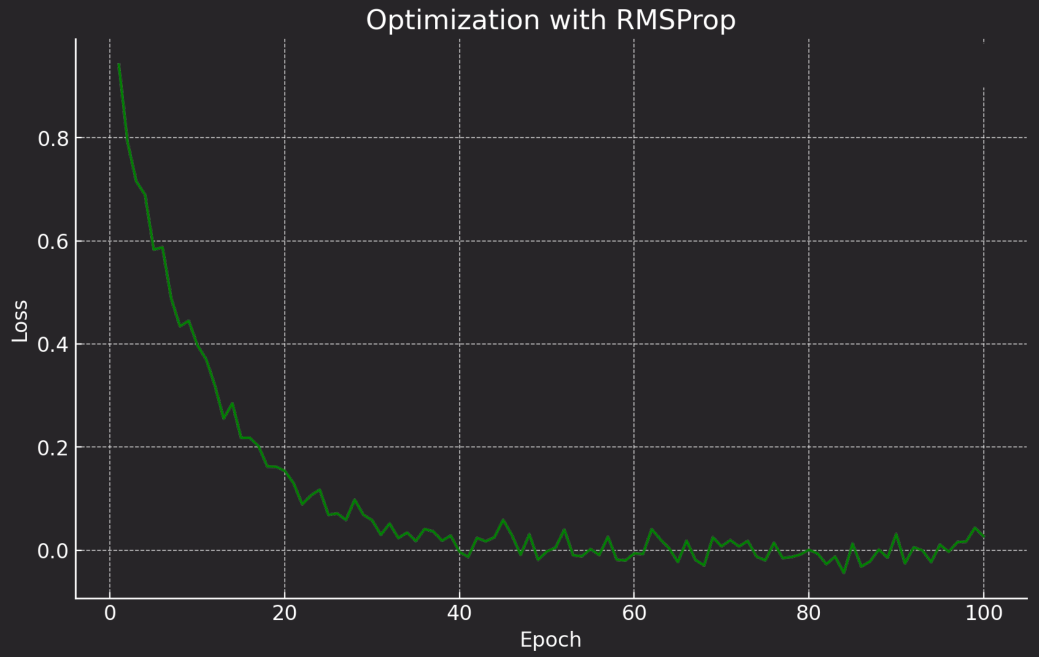
$$\eta' = \frac{\eta_0}{1 + \text{Decay\_Rate} \times \text{Epoch\_Number}}$$

- **Default value = 0.1** ; in PyTorch
- **Higher decay rate → Lower new learning rate**

RMSProp

RMSProp

Root Mean Squared Propagation accelerates the optimization process by reducing the number of updates needed to reach the minima.



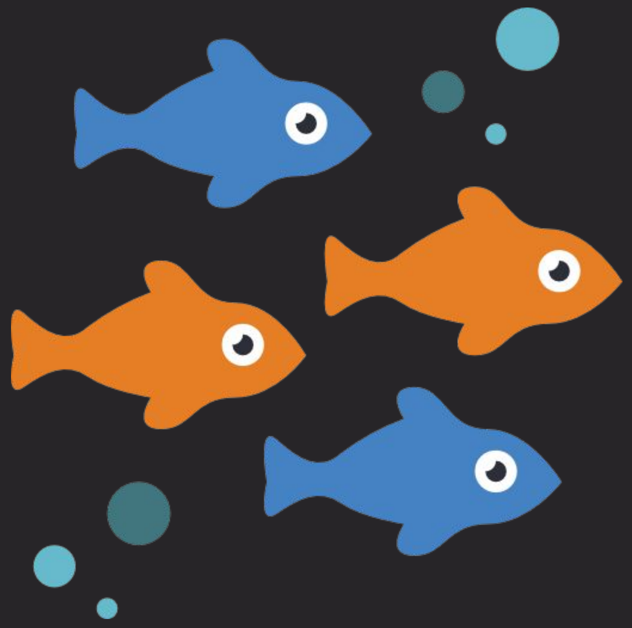
RMSProp

Root Mean Squared Propagation accelerates the optimization process by reducing the number of updates needed to reach the minima.

$$v(w_t) = \beta v(w_{t-1}) + (1 - \beta) \nabla L(w_t)^2$$

$v(w_t)$ = is the moving average of the squared gradients up to time step t

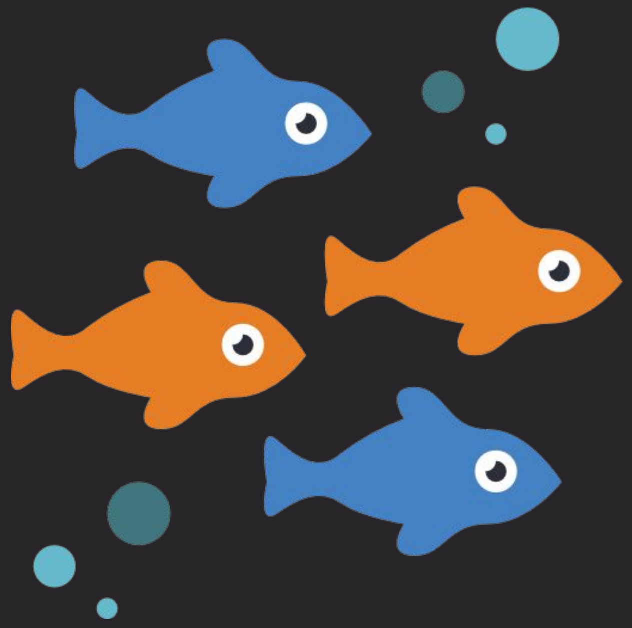
RMSProp



Model to classify variety of fishes

Primary Factor: Color

RMSProp



Model to classify variety of fishes

Primary Factor: Color

- Penalizes parameter "Color"
- Rely on other features

RMSProp

1. Moving average of squared gradients

$$v(w_t) = \beta v(w_{t-1}) + (1 - \beta)\, \nabla L(w_t)^2$$

β = Decay Factor

2. Final weight updation

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v(w_t) + \epsilon}} \cdot \nabla L(w_t)$$

RMSProp

1. Moving average of squared gradients

$$v(w_t) = \beta v(w_{t-1}) + (1 - \beta) \nabla L(w_t)^2$$

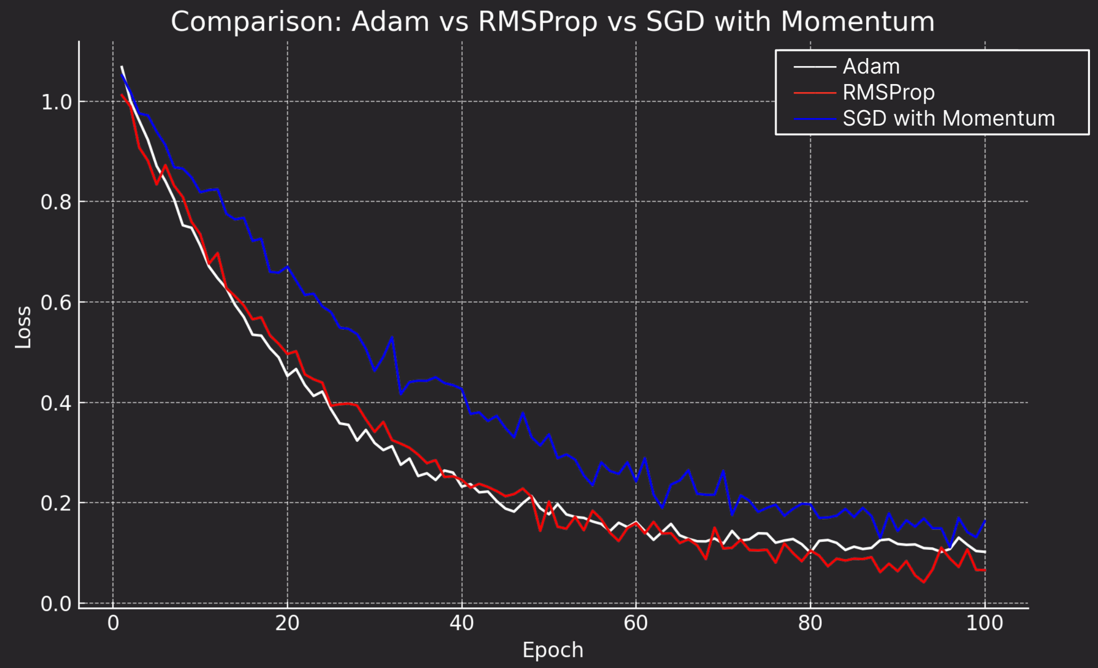
2. Final weight updation

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v(w_t) + \epsilon}} . \nabla L(w_t)$$

Adam Optimizer

Adam Optimizer

Adaptive Moment Estimation or Adam is a combination of RMSProp and Momentum.



Adam Optimizer

1. First Moment (Mean) Estimate

$$m_t = \beta_1 m_{t-1} + (1 - \beta) \nabla L(w_t)$$

m_t = First moment vector (moving average of the gradients) at time step t

β_1 = Exponential decay for the first moment estimate

2. Second Moment (Uncentered Variance) Estimate

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

v_t = Second moment vector (moving average of the squared gradients) at time step t

β_2 = Exponential decay for the second moment estimate

Analytics Vidhya

Adam Optimizer

1. First Moment (Mean) Estimate

$$m_t = \beta_1 m_{t-1} + (1 - \beta) \nabla L(w_t)$$

2. Second Moment (Uncentered Variance) Estimate

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

3. Update rule for Adam Optimizer

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t\hat{}} + \epsilon} \cdot m_t\hat{}$$

Hands-on: Applying these optimizers to our model