6. Adaptive Gradient (AdaGrad)

- Adaptive Gradient as the name suggests adopts the learning rate of parameters by updating it at each iteration depending on the position it is present, i.e- by adapting slower learning rates when features are occurring frequently and adapting higher learning rate when features are infrequent.
- The motivation behind Adagrad is to have different learning rates for each neuron of each hidden layer for each iteration.

But why do we need different learning rates?

Data sets have two types of features:

- Dense features, e.g. House Price Data set (Large number of non-zero valued features), where we should perform smaller updates on such features; and
- Sparse Features, e.g. Bag of words (Large number of zero valued features), where we should perform larger updates on such features.

It has been found that Adagrad greatly improved the robustness of SGD, and is used for training large-scale neural nets at Google.

$$\text{For Gradient Descent: } w_t = w_{t-1} - \eta \nabla J(w)$$

$$\text{For ADA Grad : } w_t = w_{t-1} - \eta'_t \nabla J(w)$$

$$\eta'_t = \frac{\eta}{\sqrt{(\alpha_t + \varepsilon)}} \qquad \alpha_t = \sum_{i=1}^{t-1} (\nabla J(w))^2$$

η : initial Learning rate

ϵ : smoothing term that avoids division by zero

w: Weight of parameters

- In SGD learning Rate same for all weight but in Adagrad this is different for all.

Advantage:

- No need to update the learning rate manually as it changes adaptively with iterations.
- If we have some Sparse and Dense feature it automatically takes out what learning rate is suitable.

Disadvantage:

- As the number of iteration becomes very large learning rate decreases to a very small number which leads to slow convergence.
- Computationally expensive as a need to calculate the second order derivative.

Adadelta, RMSProp, and adam tries to resolve Adagrad's radically diminishing learning rates.

7. AdaDelta

- It is simply an extension of AdaGrad that seeks to reduce its monotonically decreasing learning rate.
- Instead of summing all the past gradients, AdaDelta restricts the no. of summation values to a limit (w).
- In AdaDelta, the sum of past gradients (w) is defined as "Decaying Average of all past squared gradients". The current average at the iteration then depends only on the previous average and current gradient.

$$\text{For Gradient Descent: } w_t = w_{t-1} - \eta \nabla J(w)$$

$$\text{For ADA Grad : } w_t = w_{t-1} - \eta'_t \nabla J(w)$$

$$v_t = \gamma v_{t-1} + (1-\gamma)\nabla J(w_{t-1})^2$$

$$\eta'_t = \frac{\eta_{t-1}}{\sqrt{(V_t + \varepsilon)}}$$

- Instead of inefficiently storing all previous squared gradients, we recursively define a decaying average of all past squared gradients. The running average at each time step then depends (as a fraction γ , similarly to the Momentum term) only on the previous average and the current gradient.

Advantages:

Now the learning rate does not decay and the training does not stop.

Disadvantages:

Computationally expensive.

8. RMSProp

- RMSProp is Root Mean Square Propagation. It was devised by Geoffrey Hinton.
- RMSProp tries to resolve Adagrad's radically diminishing learning rates by using a moving average of the squared gradient. It utilizes the magnitude of the recent gradient descents to normalize the gradient.
- In RMSProp learning rate gets adjusted automatically and it chooses a different learning rate for each parameter.
- RMSProp divides the learning rate by the average of the exponential decay of squared gradients
- Its cost function same as Adadelta

9. Adam — Adaptive Moment Estimation

- It is a combination of RMSProp and Momentum.
- This method computes adaptive learning rate for each parameter.
- In addition to storing the previous decaying average of squared gradients, it also holds the average of past gradient similar to Momentum. Thus, Adam behaves like a heavy ball with friction which prefers flat minima in error surface.
- Another method that calculates the individual adaptive learning rate for each parameter from estimates of first (Momentum) and second (RMSProp) moments of the gradients.

Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla J(w_t)$$

$$w_t = w_{t-1} - v_t$$

RMS Prop

$$v_t = \gamma v_{t-1} + (1-\gamma) \nabla J(w_{t-1})^2$$

$$\eta'_t = \frac{\eta_{t-1}}{\sqrt{(V_t + \varepsilon)}}$$

ADAM

Momentum Component ⟶ $m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla J(w)$

RMS Prop Component ⟶ $v_t = \beta_2 v_{t-1} + (1-\beta_2) \nabla J(w)^2$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \varepsilon}} * m_t$$

Advantages:

- The method is too fast and converges rapidly.
- Rectifies vanishing learning rate, high variance.

Disadvantages:

- Computationally costly.