Advanced Algorithms Project

Root Mean Square Propagation Implementation on Neural Networks

JEEVANA R HEGDE – PES1201700633

SAAHIL B JAIN – PES1201700241

Abstract

RMSprop is an unpublished optimization algorithm designed for neural networks. RMSprop lies in the realm of adaptive learning rate methods.

Rprop tries to resolve the problem that gradients may vary widely in magnitudes. Some gradients may be tiny, and others may be huge, which result in very difficult problem — trying to find a single global learning rate for the algorithm. Rprop doesn't really work when we have very large datasets and need to perform mini-batch weights updates. The reason it doesn't work is that it violates the central idea behind stochastic gradient descent, which is when we have small enough learning rate, it averages the gradients over successive mini-batches.

On the other hand, the central idea of *RMSprop* is keep the moving average of the squared gradients for each weight. And then we divide the gradient by square root the mean square. Hence the name **RMSprop**(root mean square).

RMSprop is using the same concept of the exponentially weighted average of the gradients like gradient descent with momentum but the difference is the update of parameters.
During backward propagation, we use dW and db to update our parameters W and b as follows:

$$W = W - \text{learning rate} * dW$$

$$b = b - \text{learning rate} * db$$

In RMSprop, instead of using dW and db independently for each epoch, we take the exponentially weighted averages of the square of dW and db.

$$SdW = ß * SdW + (1 - ß) * dW2$$
$$Sdb = ß * Sdb + (1 - ß) * db2$$

Where beta 'ß' is another hyperparameter and takes values from 0 to 1. It sets the weight between the average of previous values and the square of current value to calculate the new weighted average.

After calculating exponentially weighted averages, we will update our parameters.

$$W = W - \text{learning rate} * dW / \text{sqrt}(SdW)$$
$$b = b - \text{learning rate} * db / \text{sqrt}(Sdb)$$

SdW is relatively small so that here we're dividing dW by relatively small number whereas Sdb is relatively large so that here we're dividing db with a relatively larger number to slow down the updates on a vertical dimension.