Gradient desent:-

Optimizers:-
Gradient desent , Standard Gradient Desent, RMSprop:-

Please read this file:-"https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b"

1. Introduction
There are a myriad of hyperparameters that you could tune to improve the performance of your neural network. But, not all of them significantly affect the performance of the network. One parameter that could make the difference between your algorithm converging or exploding is the optimizer you choose. There are a considerable number of optimizers you could choose from, let’s take a look at the two of the most widely used ones.
---
2. Gradient Descent Optimizer:-
Gradient descent is probably the most popular and widely used out of all optimizers. It is a simple and effective method to find the optimum values for the neural network. The objective of all optimizers is to reach the global minima where the cost function attains the least possible value. If you try to visualize the cost function in three-dimension it would something like the figure shown below.

Convex cost function:-
Stochastic Gradient Descent - Algorithm
For each example in the data
    - find the value predicted by the neural network
    - calculate the loss from the loss function
    - find partial derivatives of the loss function, these partial derivatives produce gradients
    - use the gradients to update the values of weights and biases
Each time we find the gradient and update the values of weights and biases, we move closer to the optimum value. Before we start training our neural network, our cost would be high which is represented by the point A shown in the image above. Through each iteration of training the neural network(finding gradients and updating the weights and biases), the cost reduces and moves closer to the global minimum value which is represented by the point B in the image above. The simulation below would provide a better intuitive understanding of how we reach the global minima as we iteratively train our model.

Not always our cost function is as smooth as depicted in the images above. A lot of times, these cost functions would be non-convex. The problem with non-convex function is that there is a chance that you could get stuck in a local minima and your loss might never converge to the global minimum value. Take a look the image below.

Non-Convex cost function:-

As you can see from the above image, there are two minimas in the graph and only one out of the two is the global minimum value. There is every chance that our neural network could miss the global minima and converge to the local minima. There are methods to restrict the network from converging, ex: we could change the learning rate or use momentum etc.

2.1 Learning Rate:-

Example Gradient Descent Optimizer:-

Learning rate is probably the most important aspect of gradient descent and also other optimizers as well. Let me draw upon an analogy to better explain learning rate. Imagine the cost function as a pit, you will be starting from the top and your objective is to get to the bottom of the pit. You can think of learning rate as the step that you are going to take to reach the bottom(global minima) of the pit. If you choose a large value as learning rate, you would be making drastic changes to the weights and bias values, i.e you would be taking huge jumps to reach the bottom. There is also a huge probability that you will overshoot the global minima(bottom) and end up on the other side of the pit instead of the bottom. With a large learning rate, you will never be able to converge to the global minima and will always wander around the global minima. If you choose a small value as learning rate, you lose the risk of overshooting the minima but your algorithm will longer time to converge, i.e you take shorter steps but you have to take more number of steps. Hence, you would have to train for a longer period of time. Also, if the cost function is non-convex, your algorithm might be easily trapped in a local minima and it will be unable to get out and converge to the global minima. There is no generic right value for learning rate. It comes down to experimentation and intuition.

2.2 Gradient Descent with Momentum:-

Almost always, gradient descent with momentum converges faster than the standard gradient descent algorithm. In the standard gradient descent algorithm, you would be taking larger steps in one direction and smaller steps in another direction which slows down the algorithm. In the image shown below, you can see that standard gradient descent takes larger steps in the y- direction and smaller steps in the x-direction. If our algorithm is able to reduce the steps taken in the y-direction and concentrate the direction of the step in the x-direction, our algorithm would converge faster. This is what momentum does, it restricts the oscillation in one direction so that our algorithm can converge faster. Also, since the number of steps taken in the y-direction is restricted, we can set a higher learning rate.

Standard gradient descent:-

3. RMSprop Optimizer:-

The RMSprop optimizer is similar to the gradient descent algorithm with momentum. The RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take

larger steps in the horizontal direction converging faster. The difference between RMSprop and gradient descent is on how the gradients are calculated. The following equations show how the gradients are calculated for the RMSprop and gradient descent with momentum. The value of momentum is denoted by beta and is usually set to 0.9. If you are not interested in the math behind the optimizer, you can just skip the following equations.

See this "https://www.guru99.com/rnn-tutorial.html"