***COPS ASSIGNMENT REPORT***

***IIT BHU***

***OPTIMIZERS***

NAME: UTKARSH KUMAR SAHU

ROLL NO: 20035064

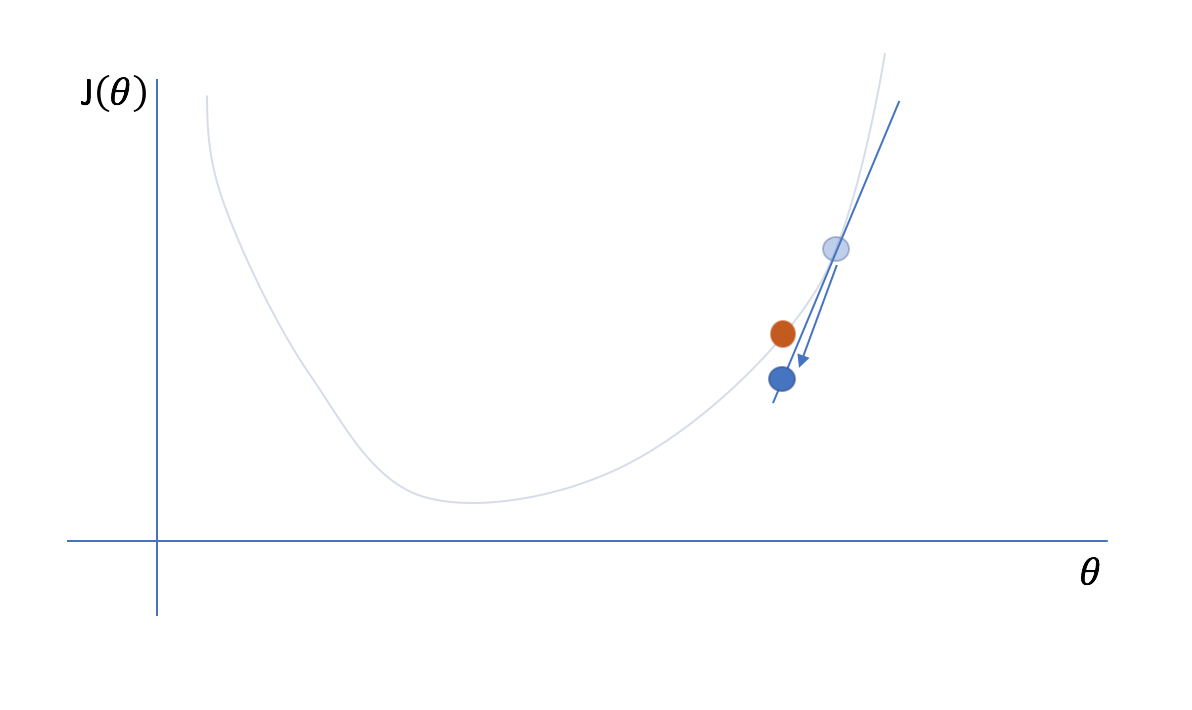
***What are Optimizers*** ?

***Optimizers****are algorithms or methods used to change the attributes of the neural network such as weights and learning rates in order to reduce the losses. They are used to solve****optimization****problems by minimizing the function.*

*In simple terms we seek to minimize the error. As such, the****objective function****is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply “loss.”*

# ***Gradient Descent***

*Gradient descent is an optimization technique commonly used in training machine learning algorithms. Often when we're building a machine learning model, we'll develop a cost function which is capable of measuring how well our model is doing. This function will penalize any error our model makes (by assigning a cost) with respect to the current parameter values. Thus, by minimizing the cost function we can find the optimal parameters that yield the best model performance.*



*Generally, we can define the process of gradient descent as*

*Θi := Θi + ΔΘi*

*where,*

*ΔΘi = −α ∂J(Θ) / ∂ Θi*

*ΔΘi is the "step" we take walking along the gradient, and we set a learning rate, αα, to control the size of our steps.*

***DISADVANTAGES:***

*As this method calculates the gradient for the entire data set in one update, the calculation is very slow, it will be very tricky to encounter a large number of data sets, and you cannot invest in new data to update the model in real-time.* ***Gradient descent can converge to a global minimum for convex functions and to a local minimum for non-convex functions.***

***Stochastic gradient descent (SGD)***

*To over the above problem we have SGD. Compared with BGD’s calculation of gradients with all data at one time, SGD updates the gradient of each sample with each update.*

***x +=-lr\* dx [lr=learning rate]***

*where x is a parameter, dx is the gradient and learning rate is constant*

*For large data sets, there may be similar samples, so BGD calculates the gradient. There will be redundancy, and SGD is updated only once, there is no redundancy, it is faster, and new samples can be added.*

***Disadvantages:***

*However, because SGD is updated more frequently, the cost function will have severe oscillations.  
BGD can converge to a local minimum, of course, the oscillation of SGD may jump to a better local minimum.* ***When we decrease the learning rate slightly, the convergence of SGD and BGD is the same.***

***Momentum***

*One approach to this problem is to use a****moving average gradient****instead of the immediate gradient at each time step. As you approach the optima your immediate gradient will get smaller, but because you're averaging over n timesteps.*

*This said, it's still possible to get stuck at a saddle point if we only have momentum in the direction of the minimum. In order to implement momentum, we'll keep track (incrementally) of the moving average gradient, where β is a hyperparameter of the optimization scheme. β essentially controls how far back we'll look when calculating our moving average.*

*VdW = βVdW + (1−β)dW*

*Vdb = βVdw + (1−β)db*

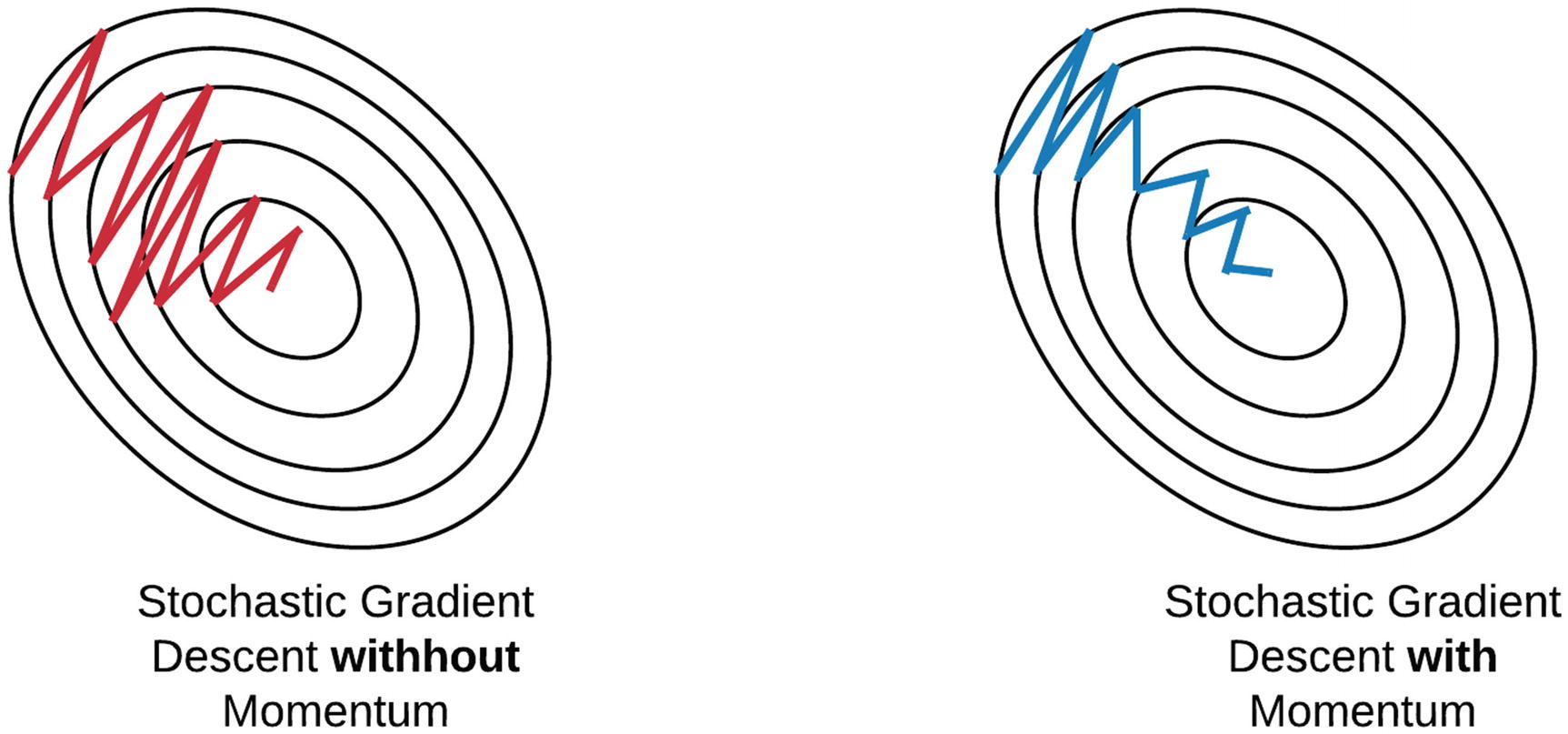
Note: Some formulas for moving average will have a bias correction, but this implementation typically disregards it.

*This is sometimes referred to as the velocity term. Next, when updating the gradients, we'll use the velocity term in place of the immediate gradient value.*

*W = W −αVdW*

*b=b−αVdb*

*Momentum offers improvements navigating ravines, reducing back and forth oscillations (because in a moving average those would cancel out) and focusing on the steepest descent path.*



***Mini-batch gradient descent (MBGD)***

***We use Mini-batch gradient as it*** *uses a small batch of samples, that is, n samples to calculate each time. In this way, it can reduce the variance when the parameters are updated, and the convergence is more stable.  
It can make full use of the highly optimized matrix operations in the deep learning library for more efficient gradient calculations. The difference from SGD is that each cycle does not act on each sample, but a batch with n samples.* ***Setting the value of hyper-parameters: n Generally value is a power of 2.***

***Disadvantages:***

*Mini-batch gradient descent does not guarantee good convergence, if the learning rate is too small, the convergence rate will be slow. If it is too large, the loss function will oscillate or even deviate at the minimum value.  
One measure is to set a larger learning rate. When the change between two iterations is lower than a certain threshold, the learning rate is reduced.*

***Adagrad***

*Adagrad is an algorithm for gradient-based optimization which adapts the learning rate to the parameters, using low learning rates for parameters associated with frequently occurring features, and using high learning rates for parameters associated with infrequent features. It is well-suited for dealing with sparse data.*

***Disadvantage:***

***The learning rate is monotonically decreasing.  
The learning rate in the late training period is very small. It requires manually setting a global initial learning rate.***

*For example, some parameters may have reached the stage where only fine-tuning is needed, but some parameters need to be adjusted a lot due to the small number of corresponding samples. Adagrad proposed this problem, an algorithm that adaptively assigns different learning rates to various parameters among them. The implication is that for each parameter, as its total distance updated increases, its learning rate also slows.Though there are some disadvantages but with the help of adagrad, we* ***eliminate the need to manually tune the learning rate.***

***RMSProp***

*The full name of the RMSProp algorithm is called****Root Mean Square Prop****, which is an adaptive learning rate optimization algorithm proposed by Geoff 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. The difference is that RMSProp calculates the****differential squared weighted average of the gradient****. This method is beneficial to eliminate the direction of large swing amplitude and is used to correct the swing amplitude so that the swing amplitude in each dimension is smaller. On the other hand, it also makes the network function converge faster. 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.*

*We keep track of the exponentially weighted average of the gradient in each dimension using the following equation, where ββ again represents a hyperparameter that controls how far back we look when computing the exponential moving average.*

*sdW = βsdW + (1−β)dW2*

*sdb = βsdb + (1−β)db2*

*We can then update our paramters using the standard form of gradient descent, but scale the learning rate αα by our exponentially weighted average.*

*W = W − α dW/√sdW*

*B = b − α db/√sdb*

*This has the effect of dampening out steep gradients as to reduce oscillations within ravines during our journey to the global optimum.*

***Adam***

***Adaptive Moment Estimation (Adam)****is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop. Adam also keeps an exponentially decaying average of past gradients, similar to momentum. Adam can be viewed as a combination of Adagrad and RMSprop. Adam implements the****exponential moving average of the gradients****to scale the learning rate instead of a simple average as in Adagrad. It keeps an exponentially decaying average of past gradients. Adam is computationally efficient and has very little memory requirements.* ***Adam optimizer is one of the most popular and famous gradient descent optimization algorithms.***

*We'll calculate the first and second moments, v and s, using two separate β values.*

*vdW = β1vdW +(1−β1)dW*

*sdW = β2sdW +(1−β2)dW2*

*vdb = β1vdb + (1−β1)db*

*sdb = β2sdb + (1−β2)db2*

*For this technique, we do apply bias correction to the moving averages.*

*vcorrecteddW = vdW / 1−βt1*

*vcorrected db = vdb / 1−βt1*

*scorrecteddW = sdW / 1−βt 2*

*scorrecteddb = sdb / 1−βt2*

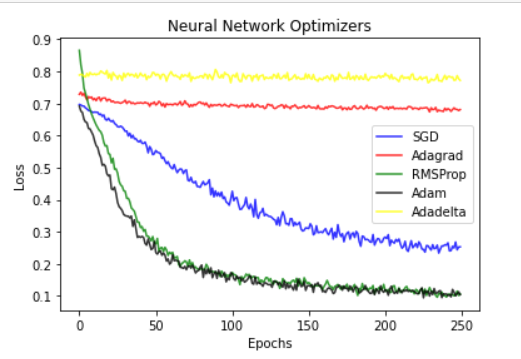
*We can then update our parameter values as follows.*

*W = W − α vcorrecteddW  / √ scorrecteddW + ε  
b=b−α vcorrecteddb / √ scorrecteddb +ε*

***Conclusion***

*We trained our neural network on a Binary classification problem, using different optimizers*

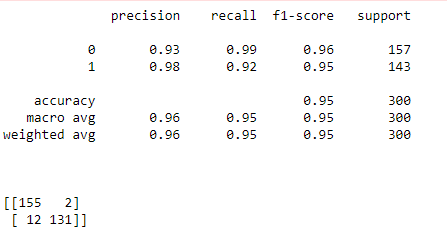
*This is our graph of loss function :*



*We can clearly see that Adam and RMSProp have a better graph and have minimum loss.*

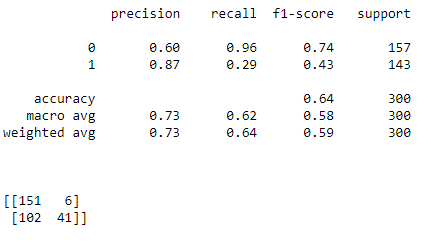
*We will also inspect there Classification and Confusion report and check for best optimizer.*

# ***SGD OPTIMIZER***[***¶***](http://localhost:8888/notebooks/Desktop/COPS_Assignment2_Optimizers.ipynb#1)-SGD-OPTIMIZER)



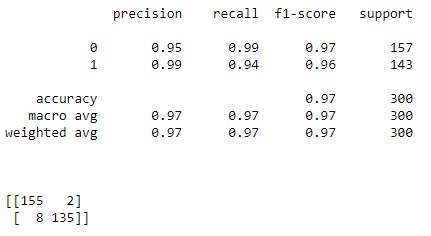
*We can see that SGD Optimizer has a 95% accuracy and a good f1-score too , so we can consider it as a good model .*

# ***Adagrad OPTIMIZER***



*We can see that Adagrad Optimizer has a 64% accuracy and a decent f1-score too , so we will consider it as a bad model and won’t be using for practice.*

# ***RMSProp OPTIMIZER***



*We can see that RMSProp Optimizer has a 97% accuracy and a great f1-score too , so we will consider it as one of our best model so far and will be using for practice.*

# ***4) Adam OPTIMIZER***

# 

*We can see that Adam Optimizer has a 96% accuracy and a great f1-score too , so we will consider it as a great model and can be using for practice.*

***Conclude :***

*So we can clearly see that RMSProp had best results though Adam and SGD were not bad, we can also conclude that the new and advance optimizers have a greater chances of predicting a good results, we could have talked about the mathematical reasoning behind each and every optimizer but the maths is way too tedious , tricky and out of scope. As these mathematical proofs are derived by PHD holders and research scholars.*

*In future there are high chances that we make a much better optimizer and improve our model prediction*.

***THANK YOU !!!***

x