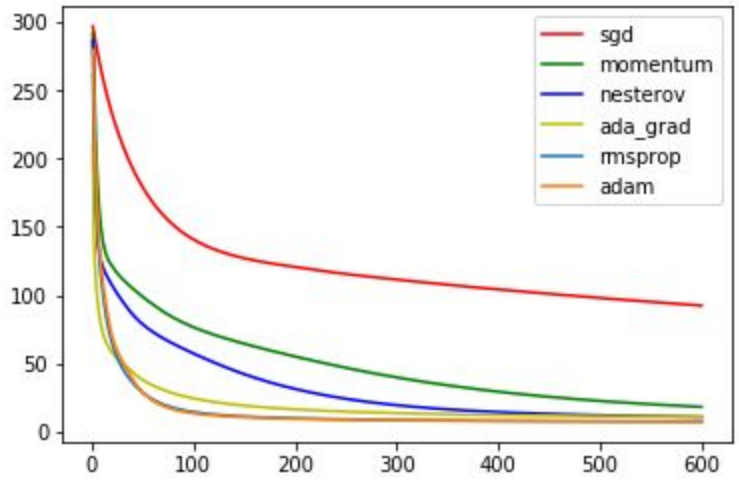
Note: MLP(all).ipynb has the same initialisation of weights for all the optimisation methods ie it has been used for comparison.
Data set **IRIS DATA SET**

ALL THE GRAPHS AND VALUES ARE PRESENT IN THE IPYNB FILE



Observations:

Stochastic Gradient Descent:

Observations:

1) The training loss slowly and gradually reduces ; and takes higher number of epochs to converge than other methods.
2) The accuracy on test data in MLP(all) is 0.8 at the end of 600 epochs and needed 1000 epochs to get an accuracy of 1. Meaning it is slowly converging.

Momentum Descent:

Observations:

1) The training loss steeply reduces and then with sharper change in the curvature gradually reduces to a smaller value of loss than SGD by the end of same number of epochs for both.

   Conclusion: SGD is slow to converge, The method of momentum is designed to accelerate learning, especially in the face of high curvature, small but consistent gradients, or noisy gradients. The momentum algorithm accumulates an exponentially decaying moving average of past gradients and continues to move in their direction.

2) The accuracy reached 1 at the end of same number of epochs as SGD whereas SGD did not achieve accuracy 1

Nesterov Momentum Descent:

Observations:

1) If we observe the training loss curve the abrupt change of slope in the case of Momentum has been reduced here keeping the rate of reduce same as momentum meaning it has smoothened out the abrupt change in the curvature. The rate of convergence is not appreciably improved though.

2) The accuracy is also the same as momentum approach for same number of epochs as Momentum approach.

   Conclusion: The difference between Nesterov momentum and standard momentum is where the gradient is evaluated. With Nesterov momentum, the gradient is evaluated after the current velocity is applied. Thus one can interpret Nesterov momentum as attempting to add a correction factor to the standard method of momentum. It enjoys stronger theoretical converge guarantees for convex functions and in practice it also consistently works slightly better than standard momentum.

Comparison among Adaptive algorithms

Ada grad:

Observations:

1) The training loss curve is gradually reducing ie the curve is smooth and there are no abrupt sharp curves as Momentum descent. This is because the learning rate is being adapted to the existing gradient values, and trying to modify them so that the convergence rate increases. Though the accumulation of squared gradients from the beginning of training can result in a premature and excessive decrease in the effective learning rate. AdaGrad performs well for some but not all deep learning models.
2) Therefore it needed more number of epochs than RMSProp and ADAM to get accuracy of 1.

Conclusion: When applied to a nonconvex function to train a neural network, the learning trajectory may pass through many different structures and eventually arrive at a region that is a locally convex bowl. AdaGrad shrinks the learning rate according to the entire history of the squared gradient and may have made the learning rate too small before arriving at such a convex structure.

RMSProp:

Observations:

1) The training loss curve is gradually reducing ie the curve is smooth and there are no abrupt sharp curves as Momentum descent.
2) It achieves 1 accuracy within very few epochs thus one of the best performer in this **setting**.

Conclusion
The RMSProp modifies AdaGrad to perform better in the nonconvex setting by changing the gradient accumulation into an exponentially weighted moving average. RMSProp uses an exponentially decaying average to discard history from the extreme past so that it can converge rapidly after finding a convex bowl, as if it were an instance of the AdaGrad algorithm initialized within that bowl

ADAM:

Observation:

1)Looking from the training loss curve it is fairly clear that ADAM is the best for this case of iris data.

It incorporates both the first moment and 2nd moment for calculating the gradients.