# Artificial Neural Network

## Lab 1: Report

**Prepared by:** ***Subhranil Bagchi***

## Part 1

For the purpose of the lab, the part 1 included implementation of various loss functions, and 5 different optimizers, namely, Mini-Batch Gradient Descent, Nesterov's Accelerated Gradient Descent, Adagrad, RMSProp and Adam.

The first loss function was given, which was a quadratic loss function, given as-

$$f(w_1,w_2) = a_1(w_1-b_1)^2 + a_2(w_2-b_2)^2 + c,$$

which is a convex loss function, for which a single global minima would exist at (0,0). The performance of the optimizers were compared.

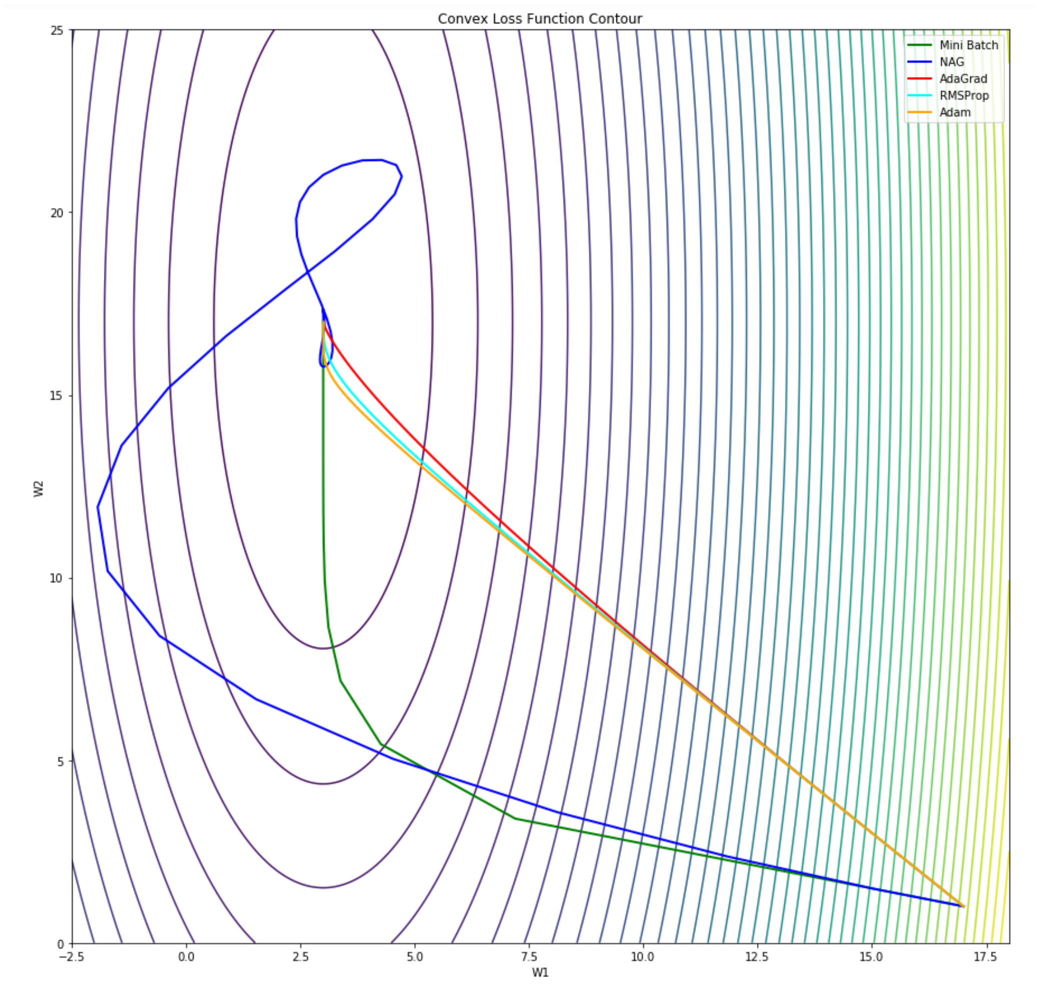


Fig. Contour Surface and Optimizers' Performances for Quadratic Loss

As we can see from the plot that mini bactch gradient descent performs descently, and still after taking a long route, reaches the gloabl minima. However, the Nesterov's Accelerated Gradient Optimizer makes quite aggressive jumps and makes drastic changes to reach the same minina. And Adagrad, RMSProp and Adam, behaves almost similar to each other, taking near optimal path. However, still in comparison Adam perform better then the two, followed by RMSProp, which is followed by Adagrad.

The second loss function, which I have considered for my experiment is a long valley function, which is given by the equation-

$$\mathbf{f(w_1,w_2) = w_1^2 - w_2^2,}$$

which the purpose that it consists of a saddle point as (0,0). The presence of such saddle point have been theoretically proven to diminish the learning process.

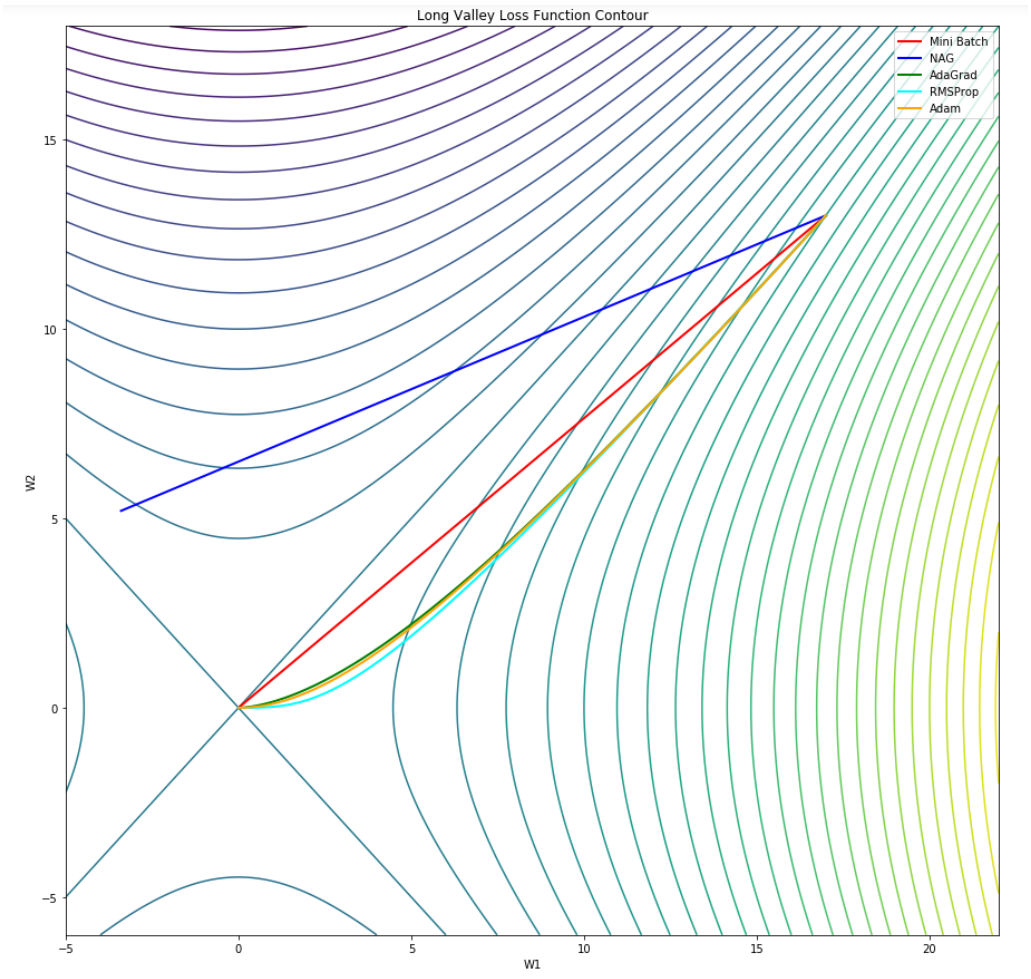


Fig. Contour Surface and Optimizers' Performances for Long Valley Function

As it can be seen from the, contour plot from my experiment, all the 5 optimizers reaches towards the point (0,0), and get stuck there. Keeping Nesterov's Accelerated Gradient Aside, all the 4 optimizers stops the learning at the mentioned point. And NAG takes a slightly different path, and halts at point somewhat nearer to (0,0).

For such a loss function, it can be said that, none of the optimizers are performing in a good way in my experiments, and presence of such saddle points affects the learning considerably.

The third, loss function considered is Beale's Function, given by-

$$\mathbf{f(w_1,w_2)=[1.5-w_1(1-w_2)]^2+[2.25-w_1(1-w_2^2)]^2+[2.625-w_1(1-w_2^3)]^2,}$$

for which, Adam, RMSProp and Adagrad performs almost similar, followed by Mini Batch Gradient Descent. Lastly, it can be seen that the NAG, as the other losses above, gains too much of momentum, ad thus makes to much of direction change to reach the same point.

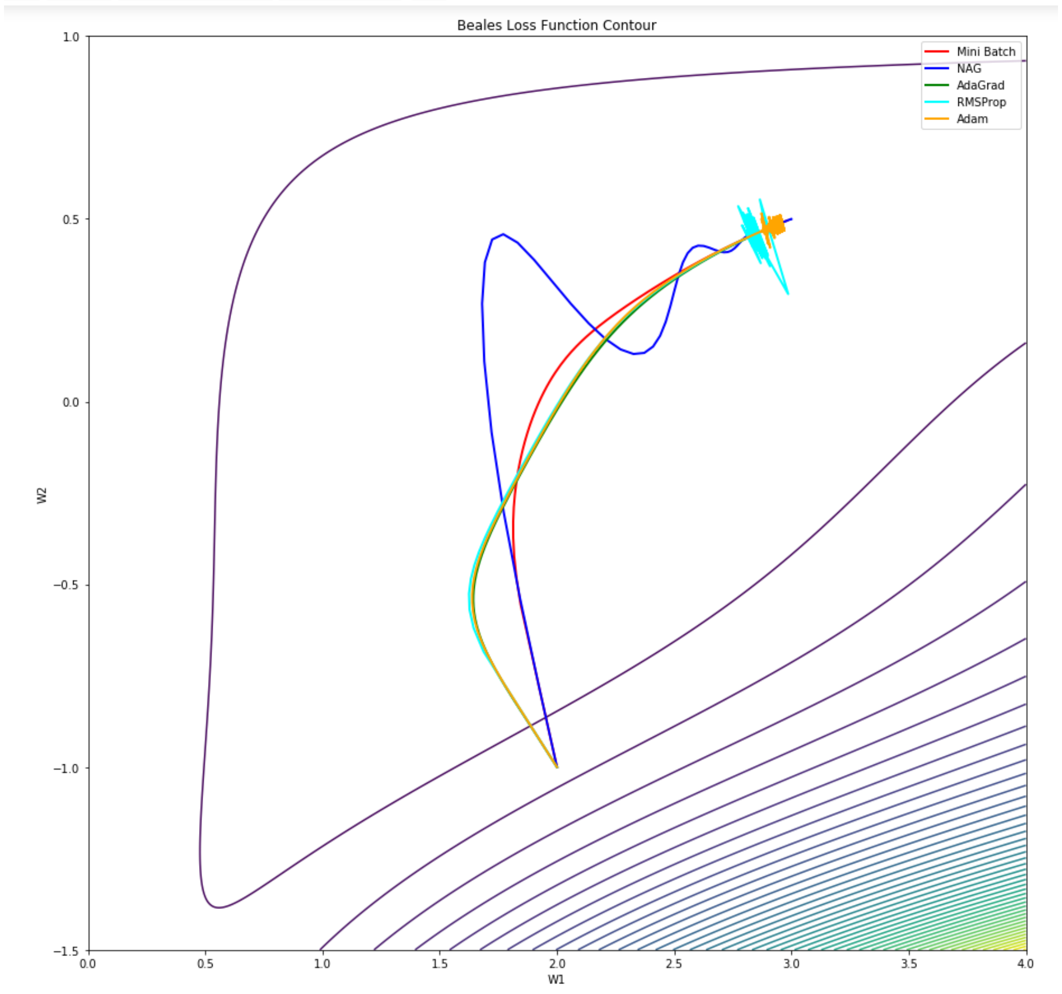


Fig. Contour Surface and Optimizers' Performances for Beale Function

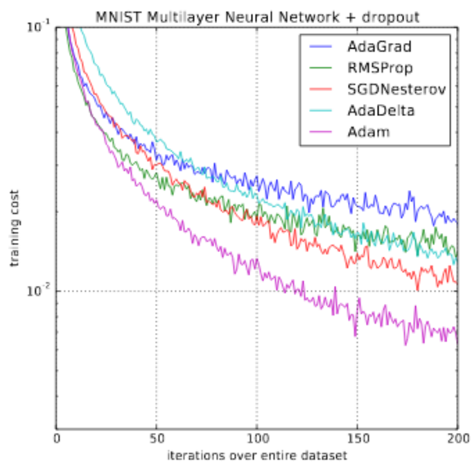
## Part 2

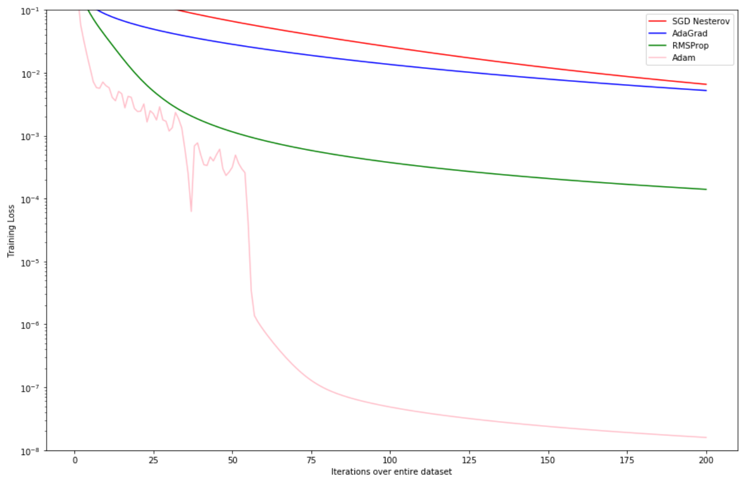
The part 2 of the lab is divided into two submodules. The first one is on MNIST dataset, where as the second one is on Cifer10 dataset.

### *MNIST*

For this portion of the lab, the four among the five optimizers, which were implemented in Part 1 of the lab were considered, namely SGDNesterov's, Adagrad, RMSProp, Adam.

Since, the neural networks training depends on multiple random parameters, 5 trials for each optimizers were taken, and losses per epoch were averaged out.





(a) (b)

Fig. (a) Training loss in the paper, (b) Training loss as implemented

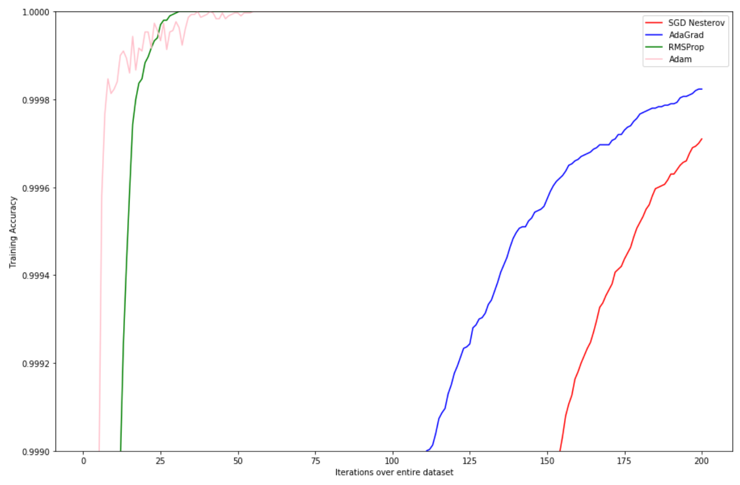


Fig. Training Accuracy

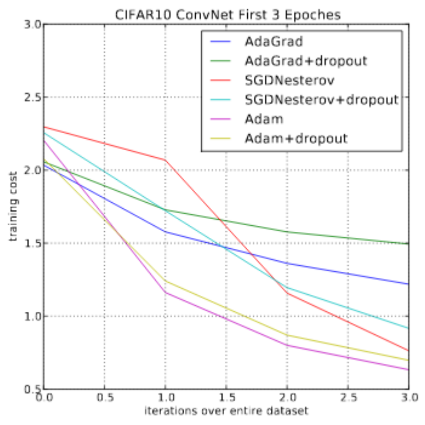
As it can be seen from our experiments, Adam performs significantly better, when compared with all other optimizers. However, in our can, the performace is much better when compared with the paper's version, which could probably due to the learning rate taken and dropout's probability, both of which were unknown in the paper.

One key thing to observe there is the smoothness of my results, which actually fluctuates in the paper. I believe this could be the number of trials I have taken into consideration. One key difference from the paper is that, in the plot it's given that SGDNesterov's performs better than both Adagrad, and RMSProp, however, in my experiments, RMSProp is performing significantly better than both SGDNesterov's and Adagrad.

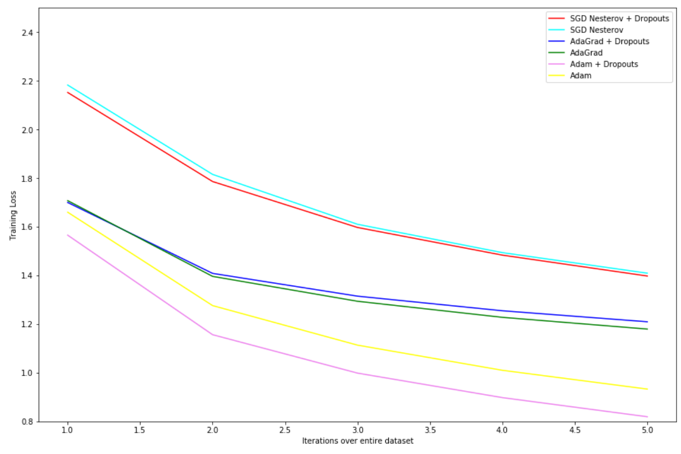
Also, observing the training accuracy, it can be seen that both Adam and RMSProp achieve high accuracy, in te begining of the training only, when compared to Adagrad, and SGDNesterov's.

## ***CIFER10***

For the Cifer10 the two models that were mentioned in the paper were implemented, and the three Optimizers Adagrad, SGDNewsterov's and Adam were taken into consideration. Even though in the paper, the training ran for 45 epoch, I could only train for 5 epochs due to time constraints at last minute.



(a)



(b)

Fig. (a) Training loss in the paper, (b) Training loss as implemented

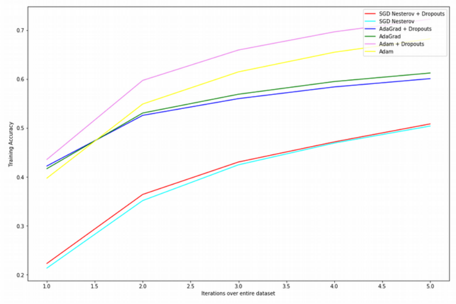


Fig. Training Accuracy

From the plots, like the above experiment it can be seen that for the early stages for the training Adam performs significantly, better compared to all other models, which is implicated by such decrease in loss. However, in the paper it is shown that for the early stages, SGDNesterov performs better than Adagrad, however, in my experiments, Adagrad is achieving superior results than SGDNesterov's.