Deep Learning

High Performance Computing

Exercises

Olga Valls Murcia

Introduction

The aim of this Laboratory is to made some modifications to the code provided in order to obtain different performance plots for the different optimizers.

Exercise 1: Optimization Problem with Gradient Descent

For Gradient Descent, a learning rate of 0.001 has been tested, as well as 0.01 from the original code. Adam Optimizer and RMSProp Optimizer, with learning rate 0.01 has been also tested.

The results are shown in Figure 1 where Gradient Descent with learning rate 0.01 starts with lower loss value and gets values closer to zero faster than the other optimizers. From the 200th iteration, the loss values for Gradient Descent Optimizer with learning rate 0.01 are the same as the ones for RMSProp Optimizer.

Gradient Descent Optimizer with learning rate 0.001 starts better than Adam and RMSProp, although the final result is the worse, as Adam, being the worse at the beginning, descents slower but at iteration 700 gets the same loss values than Gradient Descent Optimizer with learning rate 0.01 and RMSProp Optimizer.

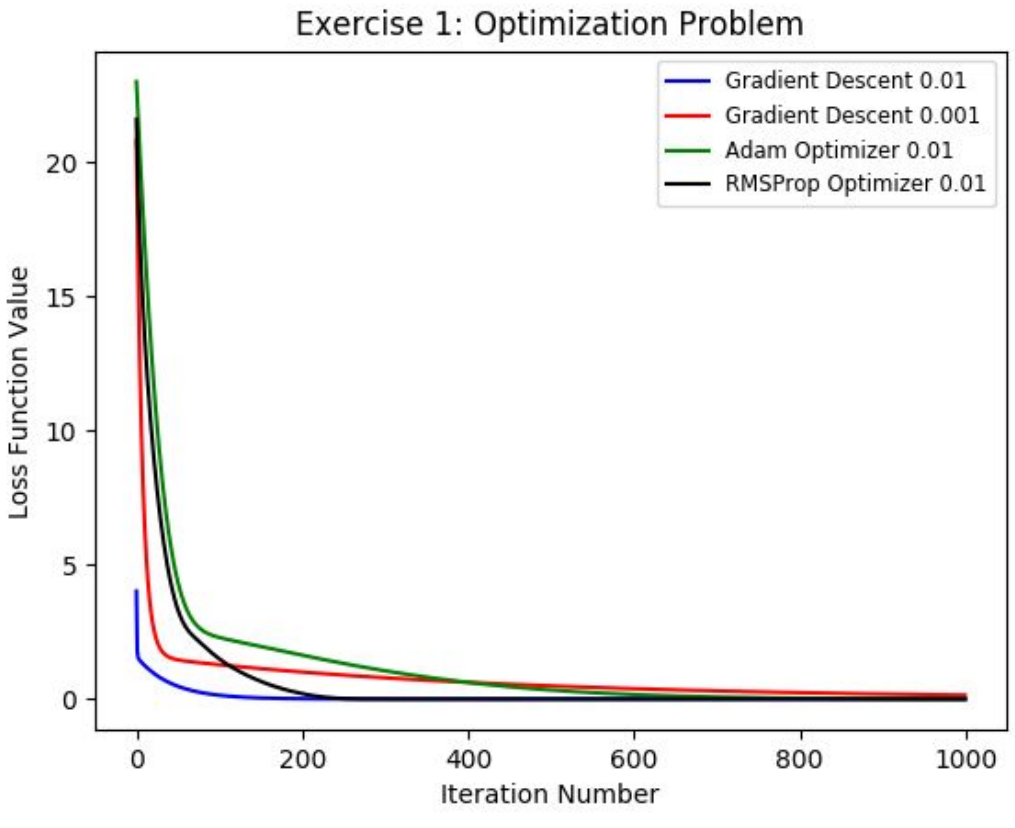

Figure 1: Plot of the loss value for different Optimizers

Exercise 2: MNIST Single Layer Network

The basis for this exercise was Gradient Descent with a learning rate of 0.5. In Figure 2 the different Optimizers that have been tested are shown. As in Exercise 1 (Figure 1), RMSProp with learning rate 0.01 and Gradient Descent with learning rate 0.5 provide the best results from iteration 100, although RMSProp starts converging later than the first, who converges very similar as Adam Optimizer with learning rate 0.01.

The worst results are for Gradient Descent with learning rate 0.1. A test has been made with an even smaller value of the learning rate for Gradient Descent to show that, opposite to the previous exercise, a very small value of the learning rate doesn't provide good results, as shown in the figure in olive color.

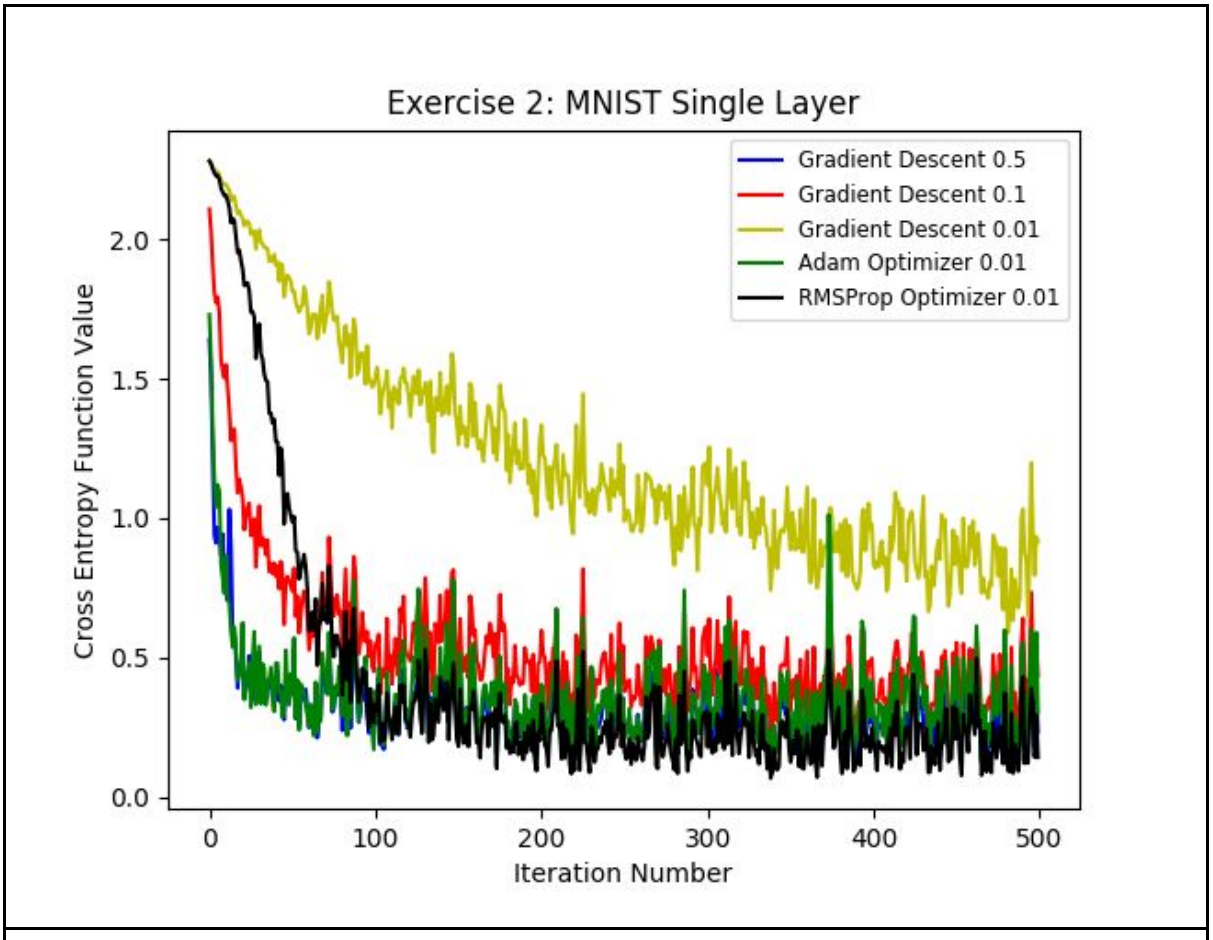

Figure 2: Plot of the cross entropy value for different Optimizers

Exercise 3: MNIST Multiple Layer Network

For this exercise, different Batch sizes have been tested to see the performance of the Network, which is shown in Figure 3. As the plots show, for the training images, a Batch size of 25, 40 or 50 images provides the best accuracy. The higher the Batch size, the smoother is the plot, so Batch 50 would give the best results:

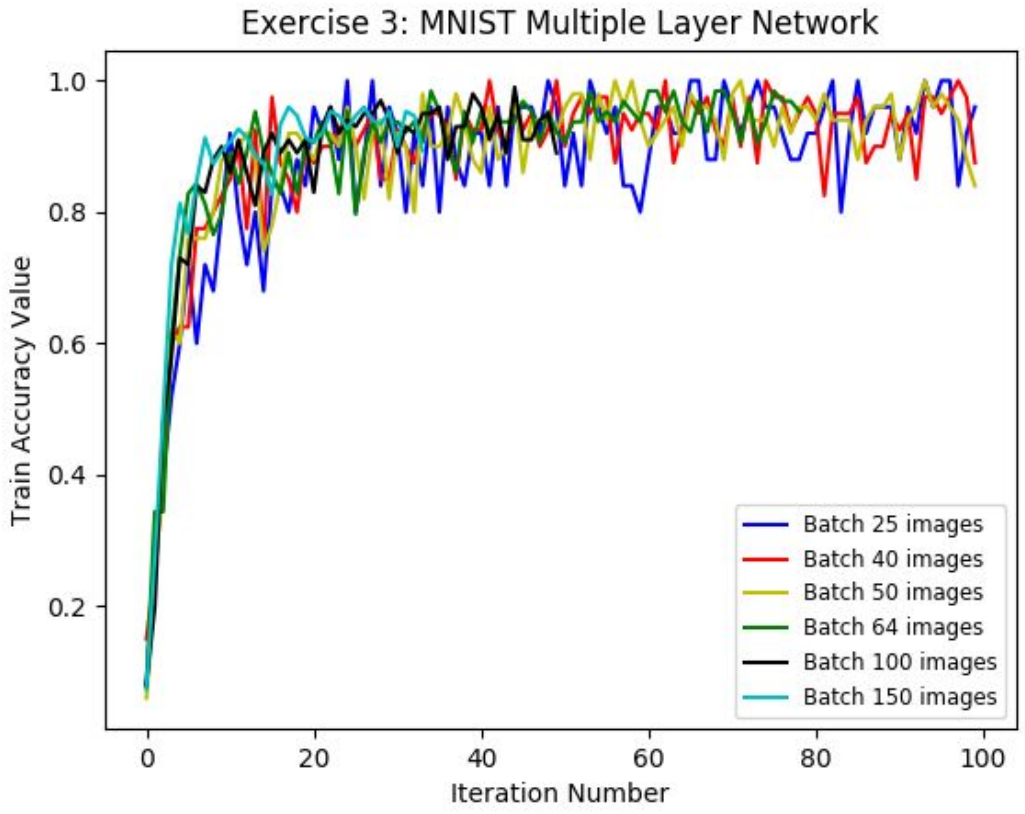

Figure 3: Plot of the Training Accuracy value for different Batch sizes

Testing the Network for those different Batch Sizes, as seen in Figures 3 and 4, Batch size of 50 gives the best result, an Accuracy of 96.11%.

Batch Size	25	40	50	64	100	150
Test Accuracy	0.9571	0.9596	**0.9611**	0.9398	0.9479	0.9413

Figure 4: Results for theTest Accuracy value for different Batch sizes

Another approach is to pick a **random** number between 1 and the total number of images - batch size, and use the images from that index forward.

Figure 5 shows the result for this approach for each of the different batch sizes, where, for the training images, all the results are very similar.

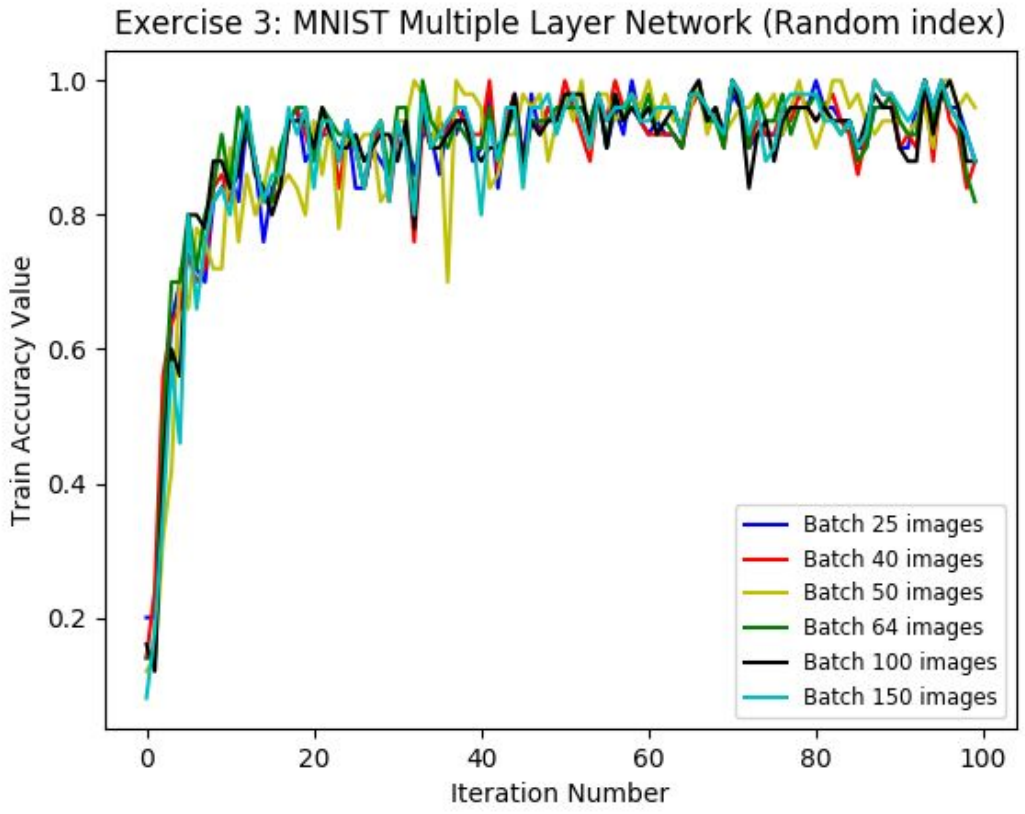

Figure 5: Plot of the Training Accuracy value for different Batch sizes (random init index)

Testing the Network for those different Batch Sizes, as seen in Figures 5 and 6, for all the different Batch sizes, the results are above 96%, higher than the best result obtained with the first approach (except for the Batch size of 40).

Batch Size	25	40	50	64	100	150
Test Accuracy	0.9617	0.9602	0.9628	0.9628	0.9631	**0.9636**

Figure 6: Results for theTest Accuracy value for different Batch sizes (random init index)

Git Repository

In https://github.com/ovalls/mai_dl you can find the code for this exercises along with the results for each one:

- `gradient_descent.py`: Exercise 1
- `singlelayer.py`: Exercise 2
- `multilayer.py`: Exercise 3
- `multilayer_random.py`: Exercise 3 with random index for the first image in each iteration.
- `plotsMLP.py`: script to plot MLP results
- `plotsMLP_random.py`: script to plot MLP random results
- `Exercise1.png`: Plots for the Exercise 1
- `Exercise2.png`: Plots for the Exercise 2
- `Exercise3_1.png`: Plots for Exercise 3
- `Exercise3_2.png`: Plots for Exercise 3 with random init index