Introduction

This lab is about using the same type of full connected network from lab 1 but with a second layer containing 50 nodes. Here the middle layer has an activation function called ReLu. The input data is normalized, and a momentum will be added to speed up the training.

The assignment

I checked the gradients by comparing with the given numerical gradient computation called ComputeGradsNumSlow. Used the same initiation parameters for gradient calculation and compared the vectors of the bias and the weight matrices by taking the Euclidean distance which can be seen in table 1.

Euclidean distance of $\nabla W1$	Euclidean distance of $\nabla W2$	Euclidean distance of $\nabla b1$	Euclidean distance of $\nabla b2$
4.4393e-04	5.0610e-09	2.9881e-06	1.2695e-10

Table 1. Euclidean distances between the gradients from ComputeGradsNumSlow and my own.

One can clearly see that the matrices and vectors are not far from each other, by calculating the distance from each of the corresponding entries and taking the root mean square of the differences.

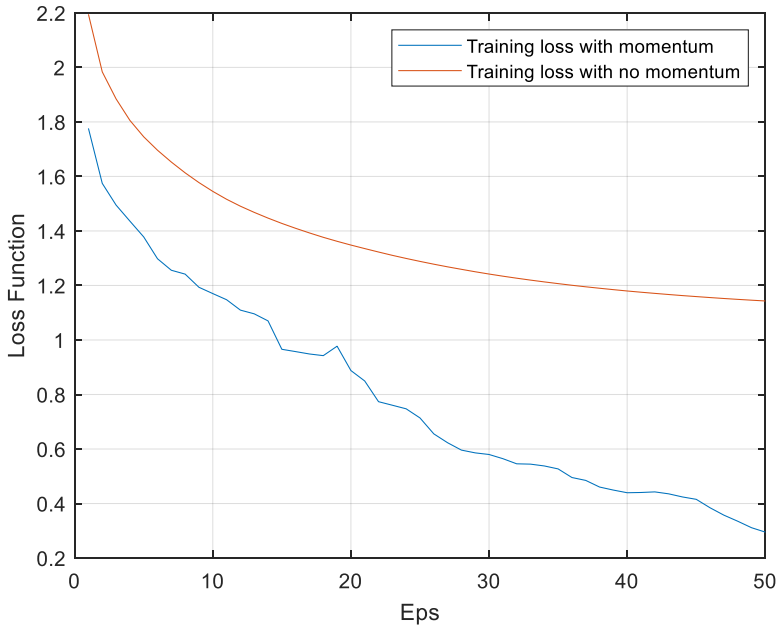

Fig 1. Loss values from training with and without momentum.

Figure 1 shows that the training with momentum goes faster than without, by looking at the loss values going down faster. Also, the time needed for training is faster with momentum.

Elapsed time with momentum: 140.471957 seconds.

Elapsed time without momentum: 149.373201 seconds.

Next for getting the hyper-parameters I started with the coarse search where I first stated the range of eta by looking at the loss from training.

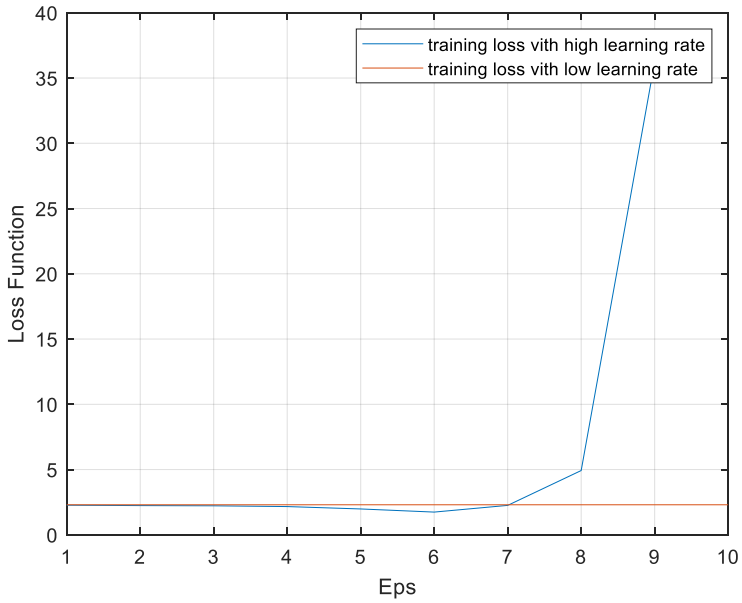

Fig 2. Loss values from training with high and low eta.

With lambda equal to 0.000001. Eta (blue loss) is 2, the loss diverged, and the final loss value was NaN, indicating the learning rate was too high. The (red loss) was with a too small learning rate 0.0001, the loss barely changed.

By taking the information from fig 2. I started with the range of eta from 1 and 0.0001. And for lambda 0 to 0.000001. Iterated over 50 types of combinations of the parameters and extracted the best parameters which got the best accuracy on the validation set.

Eta	Lambda	Validation accuracy
0.0309	2.2495e-07	0.4244
0.0375	3.7455e-07	0.4206
0.0254	1.7415e-06	0.4193

Table 3. The 3 best parameters settings from coarse search.

For the fine search I narrowed the range of eta and lambda by taking the best and the second-best values from Table 3. Eta range was set from 0.038 to 0.03 and the lambda range from 3.8e-07 to 2.2e-07.

Eta	Lambda	Validation accuracy
0.0342	2.7653e-07	0.4274
0.0319	3.1747e-07	0.4271
0.0309	2.2552e-07	0.4263

Table 4. The 3 best parameters settings from fine search.

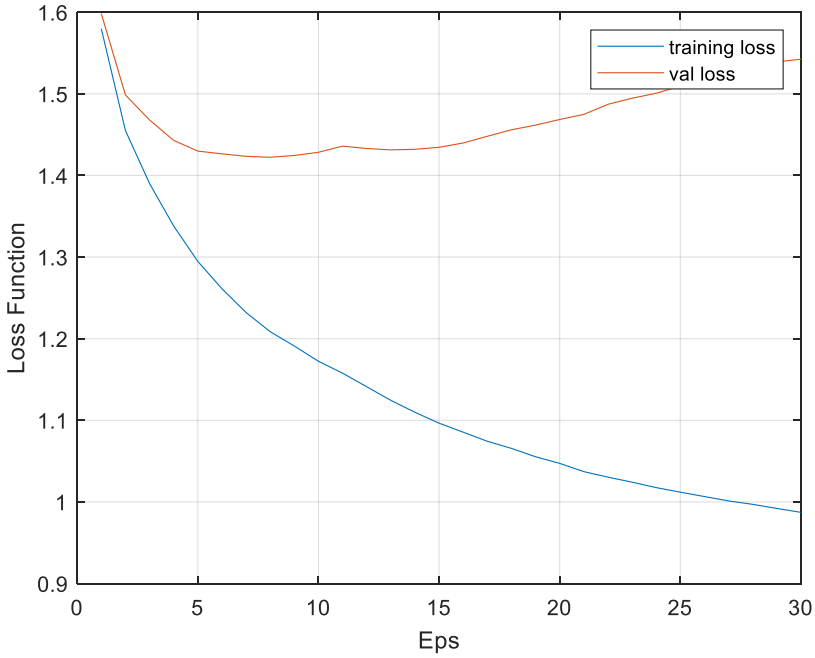

Fig 3. Loss values from training and validation set with the hyper-parameters.

The plot above shows the loss from the training and validation set with the best parameter setting from table 4. Notice that approximately 6 epochs would have been sufficient for training as the validation loss started to go up after 6 epochs. And table 5 shows the final accuracy of the network.

eta, lambda	Training accuracy	Validation accuracy	Test accuracy
0.0342, 2.7653e-07	0.6504	0.4930	0.4919

Table 5. The 3 best parameters settings from fine search.