Salih Hasan Siddiqi
Matriculation Number
Deep Learning Lab course
Exercise-1 Report

Implementation Details

The exercise sheet helped me a lot getting my concepts clear about MultiLayer-Perceptrons and Neural Networks. I started off the exercise by implementing simple helper functions to compute different activation functions and their derivatives. Then I moved to the part where I implemented a fully connected hidden layer. The fprop and brop functions were of utmost importance because there we had to implement the logic for doing forward and backward propagation. Using forward propagation, I computed network given the input handwritten digit images and values of weight and bias. Once I had my network prediction, I compared it against the given label values and calculated error. I, then backpropagated the error using partial derivatives and chain rule from Calculus, and updated values for weight and bias. I then moved on to implement Linear and Softmax output layer. There I implemented input_grad and loss functions to help compute loss on output layer.

Finally, in the Neural network class, after making changes to fprop and bprop function, I implemented stochastic gradient descent which is an optimization algorithm to compute optimal values for weight and bias. I also implemented gradient descent. The difference between GD and SGD was when the values of **w** and **b** were being updated. In SGD, I made batches and values of **w** and **b** were updated after the execution of each batch. Whereas, there were no such batches in GD. At the end, I implemented gradient_check method to validate the validity of gradients we were calculating in bprop using finite difference method. Finally, I made the network class and train and tested the network on mnist and everything seemed to be working fine.

Evaluation

The predefined network in the exercise sheet gave suboptimal results. I tried to tune the network and came to following conclusions

- Optimal value for epochs ≈ 30. The validation error didn't go down significantly by setting epoch value greater than 30. Though, training error kept decreasing, that indicated overfitting, so I stopped increasing epochs.
- Optimal learning rate ≈ 0.2. 0.1 converged slowly and 0.3 didn't help much for reducing validation error. So, I settled for 0.2 which gave me validation error of ≈ 2%
- Optimal Activation function for first hidden layer = sigmoid. Helped me reduce validation error ≈ 2%

- Optimal Activation function for second hidden layer = tanh. Helped me reduce validation error ≈ 2%
- Optimal batch size ≈ 30. I found out that frequent updates to **w** and **b** could help reduce validation error.
- SGD performed better than GD due to frequent updates to **w** and **b.**
- Random weight initialization and initializing bias to 0 also helped reduce the error.