Assignment 2

av

Lars-Erik Bakkland Moi and Oskar Markussen

i

IKT450
Deep Neural Networks

Universitetet i Agder

Grimstad, August 2023

Contents

1 Task

In this task you are suppose to implement 2 types of mulilayer Perceptrons: 1.

- Download the Ecoli dataset: https://archive.ics.uci.edu/ml/datasets/ Ecoli
- Predict the two classes: cp and im (remove the rest of the dataset).
- Make the necessary adjustments to the data.
- Implement and test a Multilayer Perceptron from scratch using only Python.
- Implement, train and test a Multilayer Perceptron using a high level library (e.g., Torch, Keras, TensorFlow).

2 Multilayer Perception

3 From Scatch

As a starting point we used the provided code from the lecture, and modified it to suit the assignment goal which was to predict cp and im. We implemented it by first loading the dateset, then following up by preprocessing the provided dataset and filtering out the rows other than cp, and im. We split the data in to training and testing and converted certain labels and so on, used a 20-80% split. We defined a set with weights and biases, created a sigmoid activation function, further on we implemented the first and second layer using the weights and provided data rows to calculate a result of each layer in which we put through the activation function. The first layer consisted of two neurons, and the second of one neuron. Second to last we created the updating weights function in which we train the first and the second layer of weights, giving us improved weights after each epoch. Lastly we created the prediction function in which we tested the accuracy of the trained weights, we know it said not to train them, but to test if our weights was trained correctly we added it anyways.

4 Not From Scatch

we leveraged the 'torch' library for our mulitlayer perception. The initizalation of the layers we used the 'nn' sub module and the linear function to create the neural network. The prepossess of the Ecoli dataset we removed every entry that not had the 'cp' or 'im' classes, and then we turned them into 0's and 1 corresponding to their classes, we also split the dataset in a training set and a validation set 80/20 split with a random generator to randomize the split. The initial layer consists of 7 neurons, which aligns with the 7 features present in the Ecoli dataset. This choice ensures that each feature is consided. We introduced a hidden layer with 2 neurons. The sigmoid activation function was applied at this layer. The choice of the sigmoid function ensures that the outputs are mapped between 0 and 1. Our network culminates in an output layer with a single neuron, also utilizing the sigmoid activation function. We first tried the Loss function Mean-Squred-error(MSE), then we switched to BinaryCrossEntropy and that improved our result. For our Optimizer first we used Stochastic gradient descent(SGT) for our optimizer but after trail and error we switched to Adam and it resulted in better results. For our learning rate we tried 0.01, 0.001 and 0.1, and we achieved the best result with 0.1, where 0.001 and 0.01 learned too slow for our small dataset, where the predictions would coinflip some of our predictions resulting in inconsistent result after each try, for example we would get a validation accuracy of 63% two times in a row then get 80% and it would fluctuate around 60% up to 95%. We used 100 epochs here i don't think adding any more would add anything here, from the accurcy chart we can observe that it converges on around 20 epochs.

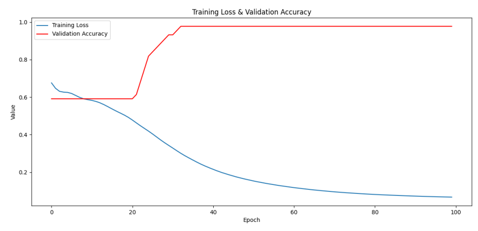

Figure 1: Training Loss & Validation Accuracy

5 Conclusion

We concluded with that the easier way of implementing this assignment is through third party libraries, this was evident from our accuracy results as well as time used to implement the project. We also discovered that with the limited amount of data we had it was very easy to over-fit the data when using the pre-made libraries.