ELL409: Assignment 1 Report

Arundhati Dixit
2016ME10824

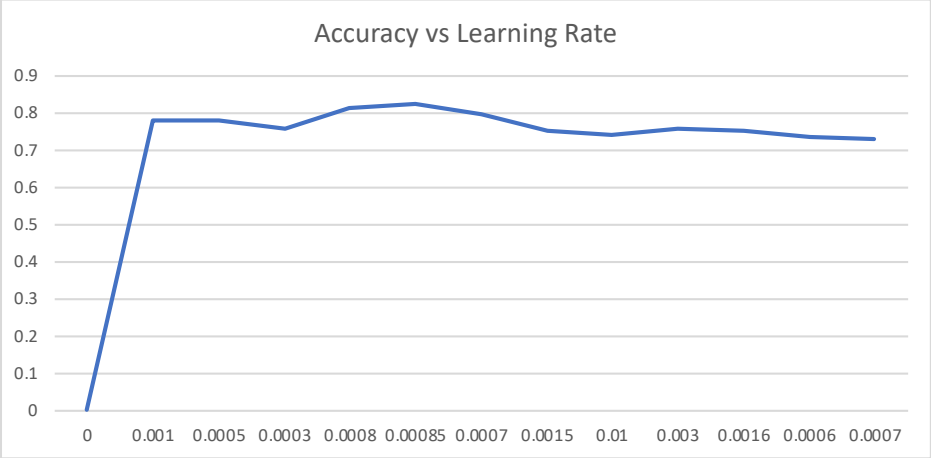
Back propagation - 9 February 2020

The code is written in the following five parts with fair amount of commenting making it easily understandable, and separate modules for each function allowing flexibility:

1. Network initialisation
2. Forward propagation
3. Backward propagation
4. Training the network
5. Testing the unlabelled dataset

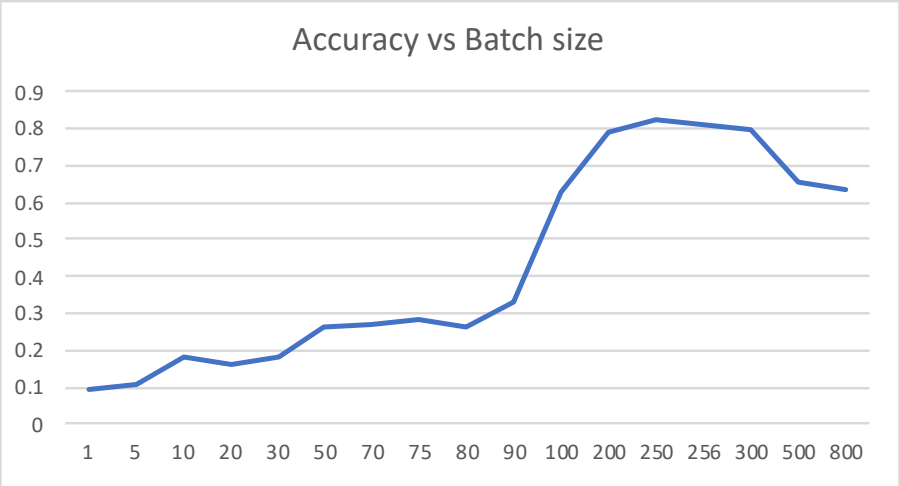
The hyper parameters that I varied are learning rate, activation function, batch size and epoch, and the model parameters I varied are number of layers and number of neurons in each layer. I discuss each, point wise below. The entire procedure felt more like tuning and less like optimisation.

- **learning rate**: decent accuracy was attained only at learning rates in the order of 0.001, which is why I evaluated the accuracy at some near about points and plotted the graph presented below. An optimum point was obtained at 0.0085. But this was for a specific batch size and layer size and number. As soon as I varied that, I had to tweak the blue of learning rate again to reach the optimum.

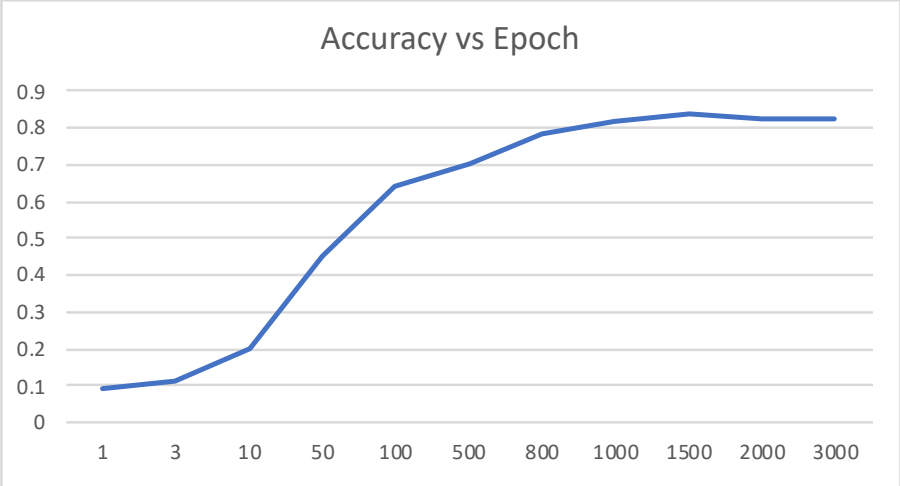


- **activation function**: I tried three activation functions- relu, tanh and sigmoid. Tanh and sigmoid gave similar results, but relu sometimes gave good results, sometimes bad. Also, the code for it was different because again in the last layer, sigmoid had to be used. So I ended up using sigmoid for consistency and ease of understanding as we had used that in lecture classes too. Also, the data was easier to normalise for sigmoid, where the value varies from 0 to 1 rather than -1 to 1, as in case of tanh. Overall, I was not able to see considerable difference in accuracy, and it was only the ease of coding that prompted me to use sigmoid.

- **batch size**: this was a significant parameter affecting accuracy, and I had a lot of trouble with this, since this was not something that is discussed in lectures or books. Initially, I had taken batch size of the order of 20 to 50, and got accuracy of no more than 18%. Then I increased it to 100, 150 and further 250, and this quadrupled my accuracy. Batch size, coupled with epoch number was highly significant in pinning down my choice of model.

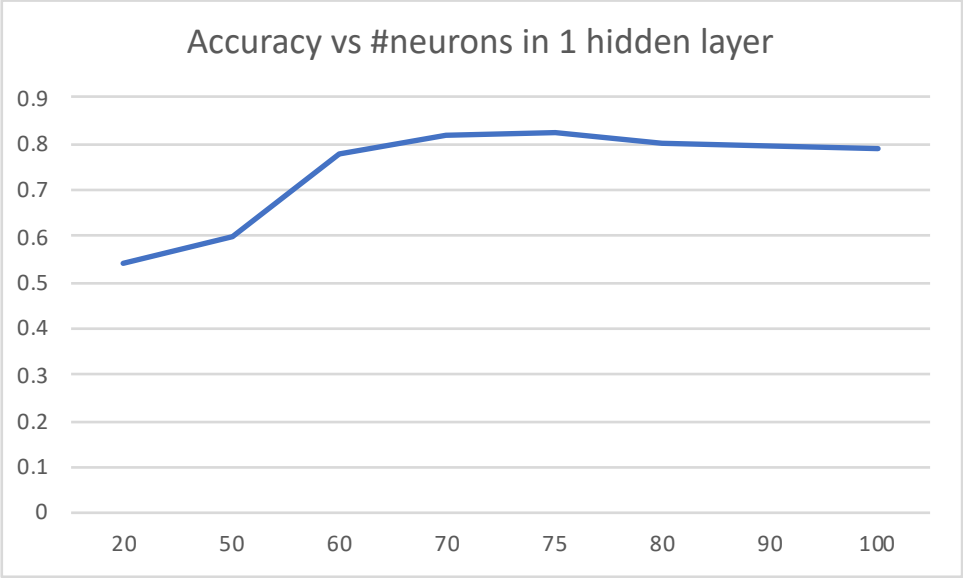


- **epoch**: Initially, I had set it to 100. But increasing it improved the accuracy steadily. Accuracy saturated at a certain value for large enough epoch number for all choices of other parameters I tried.



- **number of layers**: I tweaked this, but I could not observe any set pattern to the trend of accuracy vis a vis number of layers. Since the maximum accuracy I could observe was comparable for one and two layers, varying number of neutrons in each, I decided to keep the code simple and stuck to one hidden layer. My code is flexible for testing of any number.

- **number of neurons in each layer**: This was again a tuning parameter, but there was definitely an optimum, keeping the rest of the parameters fixed. Further, this optimum changed and was different for each layer, depending on number of hidden layers being added.



In addition to this, I generated the continuous plot varying by number of iterations by increasing batch size, and I could observe multiple peaks in that case. I could not find some explanation for this kind of behaviour.