Archit Bansal 4363946

Assignment 1 - Report

- Basic implementation of forward propagation was simple and straight forward. The implementation of the various activation functions and their derivatives was cross checked using manual calculations as well as code based tests, including numerical differentiation. The tests returned positive results.
- Implementation of backward propagation was surprisingly difficult and consumed the greatest portion of the time spent on the assignment. There were two primary challenges in the implementation of backpropagation.
- The first challenge was to map the backpropagation formulae of the various derivates from scalar values to matrices. This was my first attempt at actually implementing such a scenario and it took a lot of time and effort to understand the nuances of both linear algebra in programming as well as of numpy.
- The second challenge was to achieve the required accuracy in calculation of backpropagation gradients. This part of the implementation turned out to require more time and effort than the rest of the assignment put together, due to a buggy implementation of traversing a list in reverse in Python which was required for backpropagation.
- In order to calculate loss and gradient of loss w.r.t. the input of the softmax output layer, literature was consulted and it was determined that the most simplified version of the equation is:

$$inp\_grad = Y\_pred - Y$$

- Implementation of the Neural Network itself was fairly straightforward. I was able to successfully train the network to close to 1% training error and almost 4% validation error. I am sure that some more experimentation with the hyper parameters or a better strategy than grid search/random search will help improve the results further.
- An interesting observation during experimentation with the Neural Network hyper-parameters was ‘ReLU Death’. This happened when the Neural Network architecture consisted of three layers of ReLU units. After the first few epochs, all the ReLU units constantly generated activations of 0 and backpropagation repeatedly generated gradients of value 0 for the weights and biases of all the layers. It was initially suspected to be a bug in the code and was clarified after consulting literature. Basically, due to certain randomly generated data, all the ReLU units were given a strong negative bias, thus resulting in this situation.
- In conclusion, the basic Neural Network was successfully implemented in Python using Numpy and the hyper-parameters optimized to some extent using random search.