Report for Neural Network Assignment

Aniket Bhosale, Brinston Gonsalves

October 28, 2017

In the Neural Network Assignment, given a template of required functions, we were asked to fill in the appropriate code. This being a stochastic system, we could not imply the correctness of the system by the results of a single run and thus, we ran our code multiple times to verify the outcomes of our code.

We were provided with the rules for forward and backward propagation which we utilized to build the system. For calculation of the error, we utilized the mean square error metric. This error function provided with a numeric value proportional to the difference between the calculated value and the correct value thus giving us an appropriate idea of how much should we be tweaking to reach closer to the intended output.

The sigmoid function was responsible for conversion of the provided step function into a sigmoid. Since the sigmoid lies on both sides of the y axis, we set it to the value of y corresponding to x=0 if the value provided was too small. Thus, whenever we calculate a new step, we utilize this sigmoid function to calculate the equivalent position of the step on the sigmoid.

In each of the forward propagate and back propagate functions, we calculate the matrices (v & w) which were utilized to represent the built neural network.

The net build function was responsible for the construction of these matrices which it performed by continuously trying to reduce the error by utilizing the metric provided by the net-error function.

The simple generalization function took a dataset, and utilized a part of it to train and remaining to test it. And then, returned the mean error. This function gave us a good idea of how well system works as a measure of accuracy.

The k-fold-validation was a kind of extension to the simple generalization function where in each iteration it created a test set and train set out of a singe provided dataset and used it to find the error in calculated and correct results.

While tuning the parameters, we found that, as the value of alpha approached 1, the weights changed faster in each step. Therefore, for an example like hill climbing, we now understood that, the system will reach the peak faster but it might jump over the highest point and thus it will take time to converge or it may never be able to converge above a certain extent. In general, for the hidden number of layers, I found that as the data becomes linearly in-separable, we need more hidden layers.

In general, also as the number of iterations to build the network increased, the system got better at classification. But as the value of number of iterations went on increasing, the change of values after each iteration decreased. So, changing the number of iteration to a very large value did not have much benefit relative to the increase in time to create the network.