Artificial Neural Networks for MNIST

Roberto Fernandez

1 Introduction

In this project I attempted to implement an artificial neural networks algorithm using a constant learning rate and with the options of either the sigmoid function or softmax function.

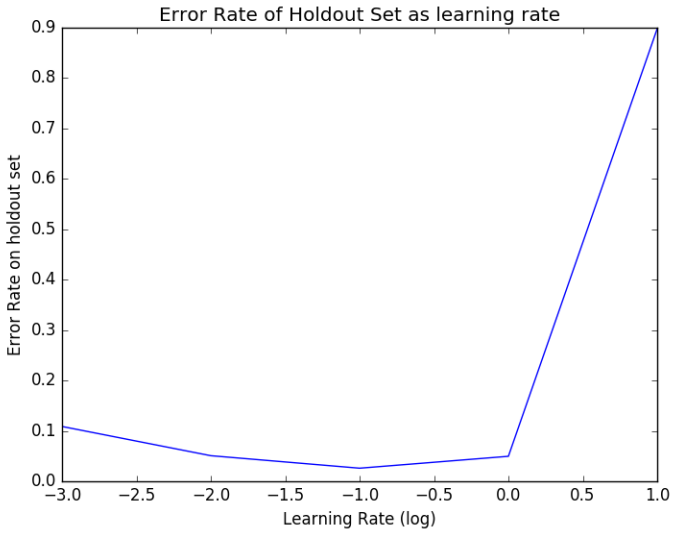
2 Implementation

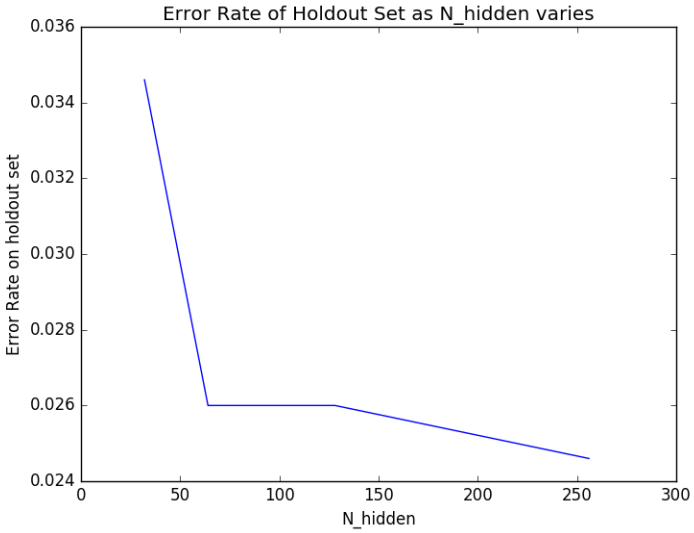
2.1 Efficiency

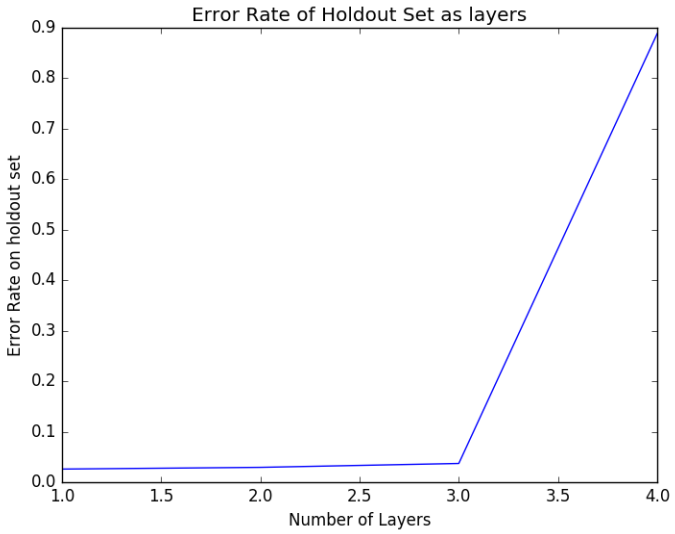
After an initial implementation that utilized many for loops I opted for the representation of neural networks learning as matrix operations in order to make the process run much faster (and it did! learns a three layer network in a couple minutes.) I also included batch processing of the data points.

3 Results

The results for the test data sets are provided in separate files but below are some plots that compare how our error rate evolves as a function of $N_{hidden}, \eta,$and the number of layers that we have. all plots are line plots where the x-axis represents whichever variable we are changing and the $y-$axis represents the error rate of the model (measured on the hold out set) given the x-axis parameter.







we thus see some interesting patterns since the error rate seems to monotonically decrease as a function of N_{hidden} while such a pattern is not present as a function of the learning rate or layers, but that is perhaps due to how our implementation works and steps could be taken to make deep layered networks effective. All of these thus seem like tradeoffs between training speed, data requirements, and accuracy.

4 Future Improvements

Since we are training on images it would be beneficial to use a ConvNet but time constraints made that difficult. It would also be interesting to experiment with different numbers of parameters (and especially multilayered networks where not every layers has the same number of neurons.)