Draft for Machine learning assignment coms 3007

# Purpose

The purpose of my assignment is to run several machine learning algorithms in order to correctly classify hand written digits as their corresponding ASCII equivalents, or at least predict within an acceptable order of magnitude. These sets of experiments will hopefully show that machine learning can be applied to the field of optical character recognition.

# Tools Used

I made used of the nod**e.**js framework for my coding and scripting of the various experiments. I used the Synaptic neural network library for node.js. This library was capable of handling constructing the given dimensions of a network, changing relevant values, randomising the data and calculating the error during training.

# Dataset Used

I made use of the MNIST database. This is a large database of hand written digits which has been compiled by taking a combination of two databases of hand written digits compiled in the United States. The website for this database can be found here: <http://yann.lecun.com/exdb/mnist/>

This database has a very large training set of 60 000 examples and a set of 10 000 test examples. All the images are normalised, and centred in the same fixed size of 28x28 pixels.

I used the mnist library for node.js to help me load the database into Synaptic. It can be found here: <https://github.com/cazala/mnist>

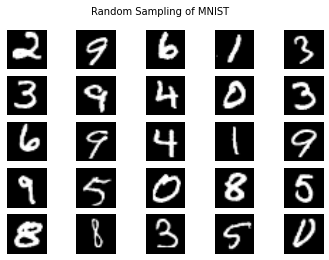
An example of the data used (several sets of samples compiled together) 

<https://camo.githubusercontent.com/d440ac2eee1cb3ea33340a2c5f6f15a0878e9275/687474703a2f2f692e7974696d672e636f6d2f76692f3051493378675875422d512f687164656661756c742e6a7067>

Using this library I was able to specify the number of training samples and testing samples I wanted. No two sets are the same using the functionality of this library. I specified 700 training examples and 20 testing examples.

My neural networks all had 784 input nodes (which correspond per pixel to the input image samples) and 10 output nodes (which correspond to the digits 0-9) as my attributes of classification. My networks setup using Synaptic JS accept a 784 length array of values (floats) and outputs a 10 length array of values (floats). I run the test data on the network after training and round the output values to the nearest integer after the network.

Here is an example of a random sampling of data:



<http://ampcamp.berkeley.edu/6/exercises/img/mnist-example-ipy.png>

# General Restrictions and Normalisation Applied

# Algorithms Used:

Note: See results for the error returned on the test set.

Synaptic runs multiple iterations of each network when training. Each iteration randmosies the data so the error has variation, each iteration. On all experiments I ran 10 iterations (detailed in Results).

Experiment 1: Neural network with one hidden layer with 100 nodes and learning rate of 0.3

Experiment 2: Neural network with two hidden layers with 50 nodes on each layer and a learning rate of 0.3

Experiment 3: Neural network with three hidden layers with 25 nodes on each layer and a learning rate of 0.3

Experiment 4: Neural network with three hidden layers with 50 nodes on each layer and a learning rate of 0.3

Experiment 5: Neural network with two hidden layers with 50 nodes on each layer and a learning rate of 0.2

Experiment 6: Neural network with three hidden layers with 20 nodes on each layer and a learning rate of 0.2

Experiment 7: Neural network with three hidden layers with 25 nodes on layer 1, 50 modes on layer, 25 nodes on layer 3 and a learning rate of 0.3

Experiment 8: RBF network with 25 RBF nodes and random centres and a learning rate of 0.3

# Results

# Conclusion

# Discussion