Introduction to Learning and Intelligent Systems - Spring 2015

jmohan@student.ethz.ch
nleow@student.ethz.ch
wongs@student.ethz.ch

April 26, 2015

Project 3 : Image Classification

As this is a classification problem with 10 labels and 2048 features, we evaluated the following methods

- Random Forest
- Extra Random Trees
- Decision Tree Classifier
- Linear Discriminant Analysis
- Gradient Boosting
- Naive Bayes
- Nearest Neighbours
- Deep Belief Networks

Most of the methods were tried using the relevant scikit-learn[1] implementation, except for deep belief networks where we utilized nolearn[2]

In general, we deduced the following

- Methods based on constructing random Decision Trees and Gradient Boosting resulted in long training times
- Normalization of data will not improve the prediction metric based on Decision Trees, Naive Bayes or Linear Discriminant Analysis
- Decision Trees and Naive Bayes on various heuristics could not break the easy benchmark
- Linear Discriminant Analysis with Single value decomposition met the easy benchmark

[1] `http://scikit-learn.org`
[2] `http://pythonhosted.org/nolearn`

- Deep Belief Networks gave the best prediction metrics

The CUDAMat library[3] was used with Deep Belief Networks to perform matrix calculations on the GPU, which allowed shorter training times by about 50 times. This made the training of multiple deeper networks and over greater number of epochs feasible, leading to better predictions.

We do not yet understand how the number of hidden layers or number of nodes in each layer in the network correlates with the performance of the network. Hence, we had to design the deep belief network by trial and error, using the rule of thumb that the number of nodes in each layer should be between the number of features and the number of classifications.

Using trial and error techniques, we found that a network with 6 hidden layers, with 1024, 512, 256, 128, 64, and 32 nodes in each respectively, trained over 50 epochs at a learning rate of 0.01 gave a good performance. Moreover, we used a modal technique, training 20 neural networks and outputting the modal prediction to counteract errors due to overfit. This technique allowed an improvement from an average error of about 0.233 using a single network to an overall error of 0.1748 on the validation data, beating the hard baseline.

[3] `https://github.com/cudamat/cudamat`