Classifying 2D and 5D Datasets with both Non-Neural Network Algorithms and Neural Networks

Andrew Pale

Department of Mechanical Engineering, Temple University
Tuc31002@temple.edu

Introduction: Two datasets were chosen to classify using both non-neural network algorithms and neural networks. The first dataset was 2D while the second was 5D. Both datasets were separated into three groups: training, development, and evaluation. All groups were labeled except the evaluation data for the 5D dataset due to that being the benchmark for how well the chosen algorithms performed. Multiple algorithms will be tested to determine how well each perform on the labeled data, which will be assumed to be an estimate of how well they will perform on the unlabeled data. The method from each category that performs best on the labeled data will be used to classify the unlabeled data and submitted to be score to determine overall performance. The researcher with the lowest error rate in will be awarded with a grade of an A for the class.

Non-Neural Network Algorithms: The first choice of a non-neural network to classify the data was the k-nearest neighbors algorithm. Initially, only the training set was used to train the data over a range of k values. The results were promising with both the 2D and 5D sets averaging at or below the given benchmarks. However, oddly, the error rate on the training data increased as the value of k increased while the error on the development data and evaluation data decreased with the value of k. One would assume this is related to the classifier being overtrained at lower values of k and generalizing more at greater values of k. Following those results, both the training set and development set were used to train the data over the same range of k values. As expected, the error rates of the development set decreased, however, the error rates of the evaluation set did not significantly change. Although, in both the case were only the training set was used for training and the combined training and development set were used for training, the performance appeared to peak around a k-value of 25.

The next choice of a non-neural network to classify the data was a decision tree to classifier. As before, initially only the training set was used to train the data over a range of values for the number of trees. The results were not quite as promising as the k-nearest neighbors algorithm however, with both the 2D and 5D sets averaging above the benchmark. For this model, the odd behavior was the training error rate was quickly driven to zero as the number of trees increased. However, the error rate of the development and evaluation data slightly decreased as well, so the model was not overtraining. Afterward, both the training set and the development set were used to train the data over the same range of values for the number of trees. The odd behavior continued for this data as well with now the error rates for both the training and development data approaching zero as the number of trees increase. However, as with the k-nearest neighbors algorithm, the error rates of the evaluation set when only training set was used and when the both the training and development sets were used did not significantly change.

The last choice of a non-neural network to classify the data was a state vector machine. As with the last two algorithms, initially only the training set was used to train the data. The results of the 5D data set averages were promising, as they were on par with the benchmark results. However, the 2D data set averages were much worse than the results of the other two algorithms. Following, both the training set and the development set were used to train the data as well. The addition of the development set to the training set did not noticeably improve the results. It is possible that using a combination of different kernel functions and optimization routines could produce greater results, however, none of the combinations attempted outperformed the k-nearest neighbor algorithm.

Neural Networks: The first neural network used to classify the data was patternnet, a feed-forward multi-layer perceptron neural network offered by Matlab. The function call to the network accepts three arguments: the data, the classes, and the size of the hidden layer. The network was trained using 10, 25, 50, 100, 175, 250, 500, and 1000 hidden layers with only the training set used to train the data. The results were promising with both the 2D and 5D sets averaging at or below the given benchmarks with the best results appearing around a hidden layer value of 100. Then, as expected, the procedure was repeated using both the training and development data sets to train the network. Curiously, the error rates significantly increased for the evaluation classification in the 2D dataset when both the training and development data were used for training. This could not be explained and was assumed to be some quirk of feed-forward neural networks. Lastly, an attempt was made to use transfer learning with a one of Matlab's pretrained neural networks. This was found to be very difficult since all of Matlab's pretrained networks were developed for image recognition. It took a large amount of time a research to learn how to manipulate the neural network to train on the data correctly. The pretrained convolutional neural network alexnet was used as the base from which the new neural network would be built. It is unknown exactly how useful it was using transfer learning from alexnet over building a neural network due to having to remove the convolution layer and maxpooling layer due to there being not spatial or temporal correlation with the data. The layers that were kept were obviously the input and classification layer as well as the fully connected layer, the softmax layer, and the ReLU layer. However, the ReLU layer was modifier to accept negative values due to some appearing in the data. The results using transfer learning were not quite as promising with none of the error rates performing better than the benchmark.

Results: Overall, for the non-neural network algorithm, the k-nearest neighbors algorithm yielded the lowest error rates. The results of training on only the training data and training on both the training and development data were very similar. When deciding which yielded the best results, the mean of error rates on the evaluation data was calculated for both and it was found that the former contained a slight advantage over the latter. Therefore the final results utilizing a non-neural network algorithm were chosen to be generated via the k-nearest neighbor algorithm using a value of k of 27, viewed below in Table 1. The classification of the 5D evaluation data was done using the results of the k-nearest neighbor algorithm using both methods of training on the training data only and training on both the training and development data as well as the results of the decision tree classifier training on both the training and development data. The final classification results were decided using a majority rules method of classification of all the results.

When comparing the results of neural networks, the feed-forward neural network yielded the lowest error rates. The results of training on only the training data and training on both the training and development data were not very similar and the training only on the former clearly outperformed the latter. Therefore the final results utilizing a neural network were chosen to be generated via the feed-forward multi-layer perceptron neural network using a number of layers of 10, viewed below in Table 1. The classification of the 5D evaluation data was done by using the results of the feed-forward neural network using both methods of training on the training data only and training on both the training and development data as well as the results of the transfer learning on both the training and development data. The final classification results were decided using a majority rules method of classification of all the results.

	2D			5D		
Algorithm	**Train**	**Dev**	**Eval**	**Train**	**Dev**	**Eval**
MATLAB: K Nearest Neighbors (KNN)	7.93%	7.90%	8.05%	34.42%	38.24%	38.31%
MATLAB: Multilayer Perceptron (MLP)	8.35%	8.00%	7.75%	36.96%	37.06%	37.81%

Table 1. Error rates on the data found using Matlab's k-nearest neighbor algorithm and Matlab's multilayer perceptron, patternnet