Behavior Analysis of Simple MNIST Neural Net

Lillian Soucy
Department of Physics and Astronomy, Northwestern University, Evanston, IL, USA

Northwestern
Department of Physics and Astronomy

Overview: The new prevalence of AI and neural nets motivated an exploration of these networks so called "black boxes". While seemingly simple in their training, thousands of neurons and layers with nonlinearities allow for an impressive classification system to arise. A simple neural net model was trained on the MNIST handwritten digit data set to help better understand what types of transformations neural nets use for classification, and its performance was compared to human techniques for data classification. A hypothesis for the "black box" structure is also explored.

Setup of the Neural Net

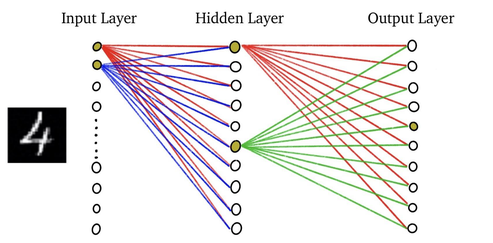

Figure 1. The structure of the simple neural net, with 784 inputs, 10 hidden neurons, and 10 outputs

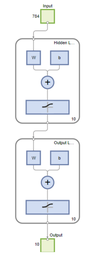

Figure 2. MATLAB's Neural Net Display

The MNIST data set consists of 60000 training and 10000 test images of handwritten digits for training a neural net. Neural nets containing thousands of nodes and many hidden layers can classify these digits with greater than 99% accuracy, but we have no idea how. To try and understand what might be happening an incredibly simple neural net was built, trained, and analyzed. Each digit is 28 by 28 pixels but was unwrapped into a 784 vector with a label corresponding to the number pictured. A custom MATLAB neural net consisting of an input layer, one hidden layer, and an output layer was trained to classify these images. The input layer contained 784 inputs (one for each pixel), the hidden layer contained 10 neurons, and the output layer contained 10 outputs (one for each digit 0-9). The 10 neurons were trained using MATLAB'S traincgf function which sets weight and bias values according to conjugate gradient backpropagation with Fletcher-Reeves updates. A logarithmic sigmoid transfer function was used for neuron activation and mean squared error was used for the cost function. After completing training on the 60000 training images (~95 epochs), the 10000 test images were used to determine the accuracy. **The trained neural net performed with 90.92% accuracy on the test images**.

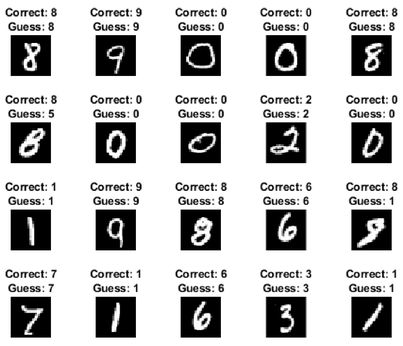

Figure 3. MNIST digit images displayed with the guesses from the neural net

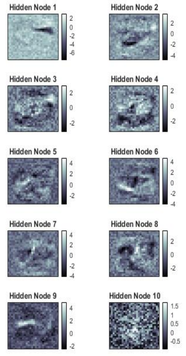

Figure 4. The 10 hidden neurons after training

A logical, **non-neural net, approach** of averaging each of the digit categories for the 60000 test images, **yielded 82.08% accuracy** when predicting the digits in the 10000 test images (when using a normalized dot product to assess similarity). This means the neural net is capturing more subtle variances in each digit for classification than just looking for average shapes.

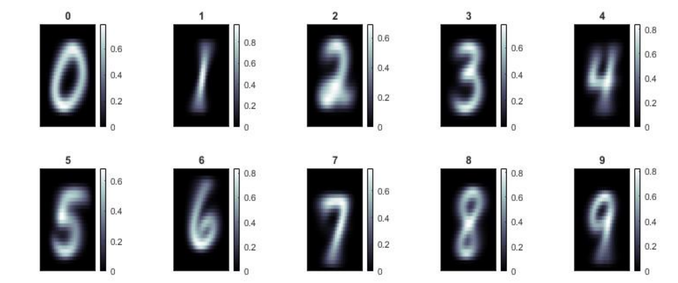

Figure 5. Averages of each digit category for the 60000 training images

Natural Clustering of the Data

As a human it seems logical to look for inherent clustering in the data to see if it is possible to classify digits based off some geometry in their 784-dimensional vector pixel space. To look for such clustering, pairwise distances between image vectors were calculated, and then a linkage matrix computed between points using the unweighted average distance. This was plotted in a dendrogram, and a cutoff point was set to have 10 clusters. The distribution of digits in each of these clusters was plotted, and the average image of each cluster was determined. While this clustering technique is able to pull out certain digits, it does not separate the data into 10 distinct digit categories.

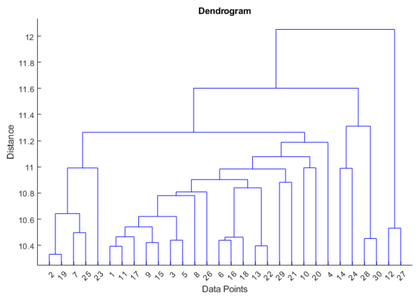

Figure 6. Dendrogram from linkage matrix

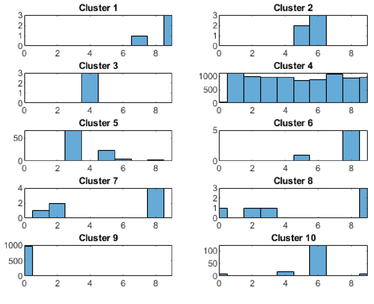

Figure 7. Digit distribution in 10 clusters from dendrogram

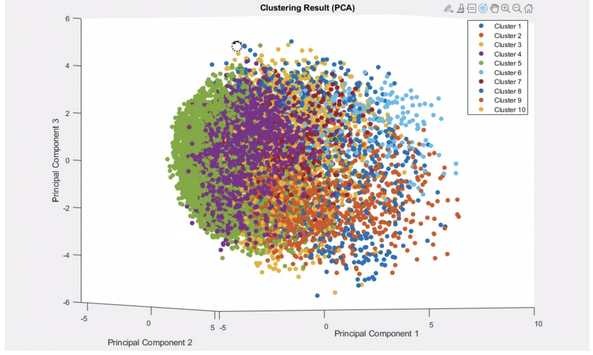

Figure 8. Clustering from dendrogram visualized in 3D after PCA to reduce dimensionality

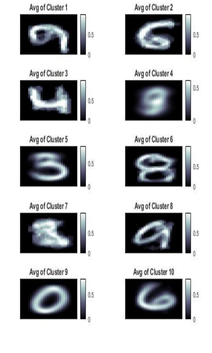

Figure 9. Average images from clusters

An alternative clustering approach is through the use of k-means. Each training image is assigned to one of 10 clusters defined by centroids from the average digit images. For each image, image-to-cluster-centroid distances are computed, and each image is assigned to the cluster with the closest centroid. The centroids are then updated via averaging, and clustering continues. **With the average centroids used for initialization 65% of the numbers can be accurately classified.** Without seeding the initial centroids with the average digit images this drops to 59%, but the clustering does not recognize the shape of a 5.

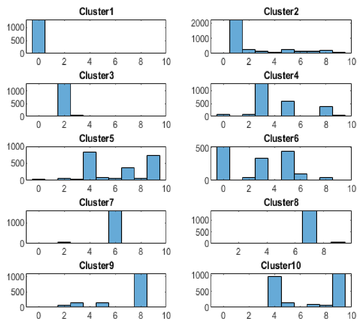

Figure 10. Digit clustering from k-means with avg initialization

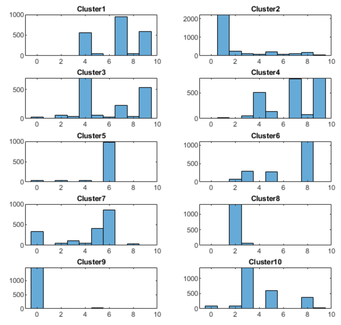

Figure 11. Digit clustering from k-means without initialization

Predictions of Neuron Behavior

The initial prediction for the hidden layer neuron behavior was some form of average Fourier transform. Looking at the hidden neuron images, this was clearly not what was happening. To learn more about what components of the images were essential to digit classification principal component analysis (PCA) was performed on the 10000 test images and then the images were filtered based on components. The PCA eigenvectors were ranked based on their eigenvalues, and then iterative PCA filtering was applied to the test images. First filtering was performed with only the largest component and then the filtered images were sorted by the neural net. Next the largest 2 components were used for filtering, then largest 3, and so forth until the prediction accuracy of the neural net was around 90% (its maximum). It was found that approximately 40 principal components can be used to capture enough of variance in the 784-dimensional space. While the performance of the neural net seemed to improve as the higher frequency components were added(Figure 12. Blue), running the filtering process in reverse(Figure 12. Purple), starting with smaller eigenvalues and working towards larger, showed that the neural net is heavily reliant on the largest principal components for digit recognition.

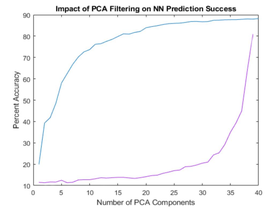

Figure 12. NN prediction success using low to high (blue) and high to low (purple) frequency component filtering

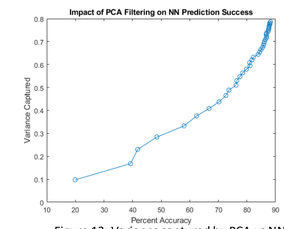

Figure 13. Variance captured by PCA vs NN prediction accuracy

The next approach was to determine if certain principal components corresponded more to certain digits than others. This was done by decomposing the average digit images into the 40 principal components used above and seeing via dot product the amount they contained of each of the 40. The digit that contained the most of a certain principal component was deemed the "winner" of that component and each digit had its winning principal components averaged to form a new type of image. This approach led to structures very similar to those that appear in the hidden neurons. Remarkably, the distribution of principle components in hidden neuron 10 (Figure 15. Orange) seem to correspond to the distribution of components in the average 9 image (Figure 15. Blue), but this correlation was not found in other hidden neurons. Since every time the neural net is retrained with random weights it settles into a new local minima, the hidden neurons shapes change each time the net is retrained from scratch. The PCA of hidden neuron 9 may be significant, or it may be a byproduct of the local minima found for the other 9 neurons. My new hypothesis is that the neural net is finding the best weighting of principle components to capture the variance in each handwritten digit. Future work will contain further exploration via linear discriminant analysis (LDA).

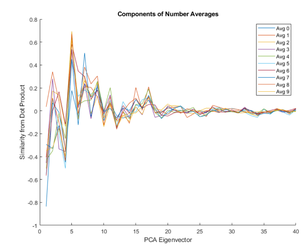

Figure 14. Amount of PCA components in average digit images

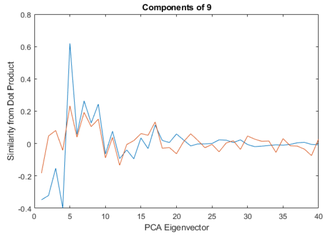

Figure 15. PCA of avg digit 9 (blue) vs PCA of hidden neuron 9 (orange)

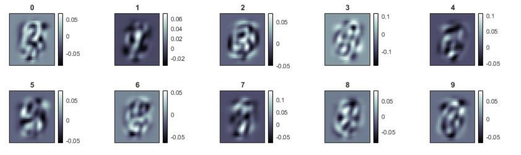

Figure 16. Average of "winning" PCA components for each digit yield structures similar to the hidden neurons

References & Acknowledgements

"Mnist." *GTDLBench*, git-disl.github.io/GTDLBench/datasets/mnist_datasets/.
Nielsen, Jesper. *Exercise 1 - Recognition of Handwritten Digits*, 13 Nov. 2018, homes.civil.aau.dk/jen/MachineLearning/Exercise1/html/HandwrittenDigits.html.
Sanderson, Grant. *3Blue1Brown*, www.3blue1brown.com/topics/neural-networks.
I would also like to acknowledge MATLAB 's extensive documentation and Open AI's Chat GPT for helping when code wouldn't run properly.