COMS W4737 Final Project Report

Varun Ravishankar, vr2263 Jervis Muindi, jj2190

12/11/11

1 Digit Recognition

Digit recognition aims to classify the digits zero through nine through automated methods and lends itself to useful applications, including automated sorting by post offices, scanning in checks at the bank, and recognizing license plates and other forms of numerical identification. It is a subset of a larger problem, handwriting recognition, and faces many of the same problems that the larger problem faces, such as recognizing that the same character that can be written in thousand of ways, and attempting to collect a dataset that can be representative of the myriad of ways that people can write that character. Image processing is a key part of digit recognition, as images of any representative image such as a zipcode must be separated into individual digits, preprocessed and cleaned up, and then output into an image that can be classified. The goal of this project is to create a classifier that is robust against the innate differences in the way people write numbers and to then apply it to a dataset that must be first be processed, and then classified.

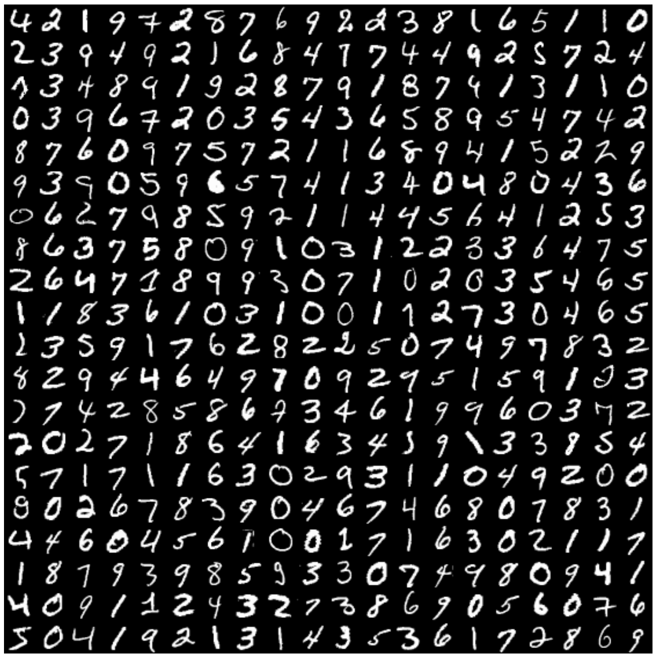
2 Dataset

Our main dataset consists of the MNIST database[1] of handwritten digits, put together by Yann LeCun at the Courant Institute, NYU, and Corinna Cortes from Google [1]. This database contains a training set of 60,000 examples and a test set of 10,000 examples and is a subset of a larger database from NIST. The digits are all a uniform size and are centered in the image.

Each image is centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so that the center of mass is at the center of the image. Each image is a binary image of a handwritten digit and consists of 30,000 patterns from two databases, one from Census Bureau employees and the other from high-school students, from about 250 different writers in total. The test set consists of 5,000 patterns from each database. The datasets are labeled to enable easy verification.

[1]http://yann.lecun.com/exdb/mnist/

Figure 1: The MNIST dataset.



We also collected an additional dataset from 20 people, consisting of the digits zero through nine written with a pen. These images were then scanned, cropped to 28x28 pixels, and converted to binary images.

3 Approach and Results

Once the image gathering and preprocessing was completed, we built our classifier[2]. The image was normalized, so we built a nearest neighbor classifier that compared the raw pixel values' Euclidean distance, and iterated through the 60,000 images to find the best match. When we looked at the entire training dataset, our classifier was able to process all 10,000 test images at a rate of about 120 images per minute.

We then modified the classifier to use k-nearest neighbors and to then take the majority vote of those k neighbors. After we performed this modification, we ran a cross-validation test on 2,000 images and got the following:

k	Accuracy
1	96.00%
2	94.75%
3	94.65%

[2]I worked on the MNIST dataset, and Jervis worked on the collected dataset

When we ran the classifier on the full 10,000 image sample, we received the following results:

k	Accuracy	Rate
1	96.91%	120 images/min
2	96.42%	69 images/min
3	96.17%	66 images/min

It is interesting to note that increasing values of k caused the accuracy to drop. This may be because when digits are similar to each other, such as 8s and 0s, finding additional nearest neighbors may cause the classifier to include other similar-looking but different digits and thus cause performance to drop. It thus seems that the additional computation necessary to use a k-nearest neighbor classifier is not worth the trouble with this problem, at least without additional preprocessing.

We also attempted principle component analysis (PCA) on the independently collected data set. However, this performed terribly, and we found that the classifier had 9% accuracy.

Finally, based upon Jervis' feature selection and image processing on the collected dataset, we ran the nearest neighbor classifier on the collected dataset and found 92% accuracy by comparing each test image to all the 60,000 training images. Due to time constraints, we were unable to run PCA on the MNIST dataset or run LDA on either set.

4 Conclusion

After running our experiments and cross-validation tests, we found that the dataset performed reliably even with the simplest classifier, a nearest neighbor classifier. We can attribute this to the large size of the training data set and note that in order to attain this level of accuracy, we had to sacrifice much time, spending nearly an hour and a half to test each classifier on the full test dataset. We also note that the images of the digits had significant preprocessing done to them in advance, as the images were already normalized and in binary form, and it was possible to classify them without removing noise or otherwise modifying the image significantly.

When it came time to testing the collected dataset, we found that a simple nearest neighbor classifier performed about as well, and would likely perform even better if the images were either traced before being scanned in (to thicken the lines and make the image more robust to scaling down), and more sophisticated preprocessing was done.

In the future, we would like to improve efficiency of the k-nearest neighbor classifier. We looked into better ways of doing k-nearest neighbor classification based upon better data structures, such as a Voronoi diagram or a VP tree. Both depend on spatial separation of points to choose groups of points more likely to be close to the mean and thus be a nearest neighbor, and are able to achieve better speedups than simply iterating through all the training samples. However, due to time constraints, we were unable to pursue this further. We also looked into perhaps limiting our training set to a smaller size, but other than to choose images at random, we were unable to come up with any precise method to throw away useless samples. Given more time, we would like to pursue coming up with extremely close samples for a specific class and discard some to get the smallest but most diverse range of all the ways a digit can be written.

5 References

1. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.