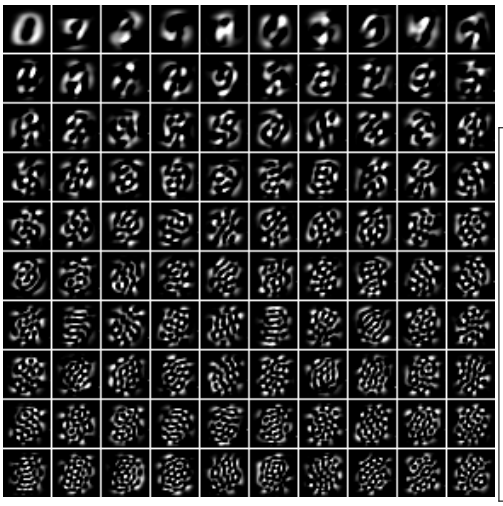
Machine Learning: Homework 2

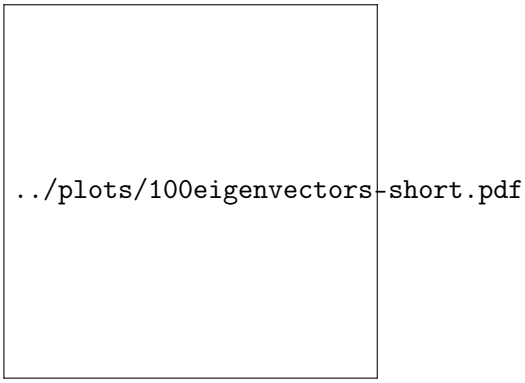

- **Email**: chrisbrown@utexas.edu
- **EID**: chb595


# 1 Algorithm

The PCA (Principal Component Analysis) algorithm is pretty straightforward, and the MATLAB implementation closely follows the general instructions. The `hw2FindEigendigits` function is a short 12 lines of code, and could be much shorter by embedding many of the functions. One complication (due to the original version of the assignment) was that sometimes the width of the matrix did not exceed the number of original dimensions of the data (784 in our case), and since the function was required to return a matrix the same size as the input image data, I found that padding it out with zeros was a suitable solution. In those cases where the number of training images was less than 748, truncating the eigenvector matrix to match that number was easy enough. Also, the assignment noted that the eigenvector matrix would need to be sorted by eigenvalues, but MATLAB's `eig` does that automatically, and only a reverse is needed. However, in line with the assignment, I still sort the eigenvalues and return the corresponding eigenvectors.



(a) Using all training data




(b) Using 100 training images

Figure 1: The first 100 eigenvectors (ordered in rows, not columns)

# 2 Evaluation

I tried out four basic evaluation metrics:

- kNN (k = 1)
- kNN (k = 10) – majority vote
- kNN (k = 10) – weighted vote
- Cosine similarity

Most of the computation time is spent evaluating each point relative to the training data. It's easy to project all the training/labeled data and the test data into the space of reduced dimensionality quickly, so I only had to do that once per choice of evaluation metric, number of training images, and number of top eigenvectors. But to evaluate a large group of test data, the algorithm would evaluate each one in term, analyzing the entirety of the training data to find the closest points. If speed was an issue, we could probably do some preprocessing on the training data to assign different regions of the PCA space certain labels, and then the test points could be queried against these regions, instead of all 60,000 training points.

There are two variables we can plot accuracy on: the top N eigenvectors, and the number K of training examples.

Also, k in kNN, while taking the most common out of 20 K, for example.

What happens if you trim the vector down and only use the top n eigenvectors? How accurately can you classify using nearest neighbor using cosine similarity? Plot figures showing the accuracy as the number of training points increases, and as the number of eigenvectors increases.

Must the images be centered exactly? what if they aren't?