Homework 1: Eigendigits

Jeremy Hintz

February 2016

1 Introduction

For the first assignment, principal component analysis (PCA) was used to classify handwritten numerical digits from the MNIST database, which can be found at http://yann.lecun.com/exdb/mnist/.

The images in our dataset are represented as 28×28 matrices, with each cell of the matrix being a value from 0 to 255 representing the gray-scale pixel value (where 0 is black and 255 is white). When flattened, the 784 element vector brings forth challenges known as *the curse of dimensionality*. One of the challenges we face if we wish to do any meaningful computation with these images is to concoct ways of reducing the dimensions with which we will be working.

When we look at matrices that have lots of information available in far less space than their original dimensions, we should immediately consider using the covariance matrix of the data and its eigenvectors, sorted by largest eigenvalues. From there, we have some options as to what algorithms we can employ in order to classify the images. In this paper, we explore a k-nearest neighbors approach and a max-likelihood approach.

The implementation of these algorithms is done in MATLAB. The main focus in my approach was looking at the algorithms themselves. Many who have tried to classify this dataset previously have used pre-processing techniques such as edge blurring, pixel shifting, noise removal, deskewing/deslanting, and Haar features. While some of these methods have proven valuable by past attempts, no pre-processing was performed in advance of the algorithms described in this report.

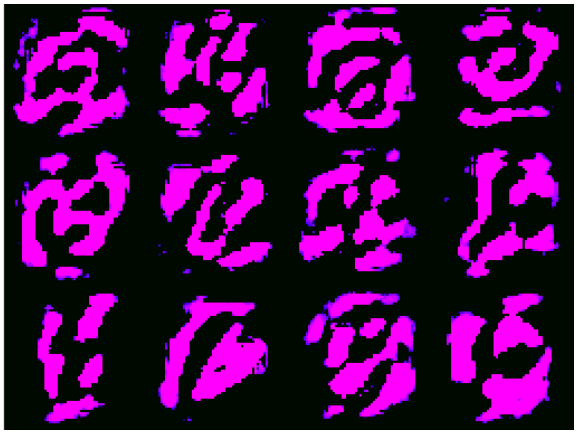

Figure 1: Eigenvectors shown using imshow function of MATLAB

2 Mathematical Overview

Consider an $n \times n$ image, I. Using I, we construct a new matrix A, whose columns are the flattened columns of I. For a data set containing S samples, the resulting dimension of A will be $n^2 \times K$. Next we wish to find the covariance of A. We do so as follows:

$$Cov(A) = \mathrm{E}((A - \mathrm{E}(A))(A - \mathrm{E}(A))^T)$$

(Cite Bishop Text) We want to find some set of eigenvectors associated with A. To do this, we construct a new matrix $B = A - \mathrm{E}(A)$. Hearkening back to our earlier discussion of issues arising from high dimensionality, we do not want to deal with B directly, but instead want to reduce it to an easier to work with form. Concretely, we find eigenvectors for $B^T A$, and multiply them with B to get eigenvectors of AA^T.

Our procedure for finding the eigenvectors of AA^T yields a new matrix, which we will denote by V. V is sorted descending by corresponding eigenvalue. We sort in this manner to put more emphasis on the eigenvectors that contain the most meaningful information about the images.

Now let $\mathcal{B}$ denote the basis we are using and $\mathcal{S}$ denote the eigenspace. Then our result V from above is nothing more than a linear transformation from $\mathcal{S}$ to $\mathcal{B}$, denoted henceforth by $T_{\mathcal{BS}}$. Let $T_{\mathcal{BS}}^{-1}$ be denoted by $T_{\mathcal{SB}}$. But $T_{\mathcal{SB}} = T_{\mathcal{BS}}^T$ since we are working with symmetric covariant matrices.

2.1 K-Nearest Neighbors

K-Nearest-Neighbors is a well-known classification algorithm in the field of Machine Learning that uses the *distances* between a given sample in order to place that sample in a given class. Again, before running this algorithm it is necessary to deal with issues of dimensionality. For this reason, we choose to deal with the vectors in S. We next use the sum of squares to calculate distances between the vectors and then sort them by lowest distance. Finally, we assign a classification to the sample by taking the mode of the top k closest "neighboring" samples.

K-Nearest-Neighbors will serve as the benchmark for other methods we try.

2.2 Maximum Likelihood Estimation

Maximum Likelihood is a probabilistic approach to classification that uses a group of samples. It assumes that the features in the model follow a Gaussian distribution, with some undetermined mean and variance. Because of this assumption on the way the data is distributed, we can extrapolate the mean and variance given just a few samples. From this, we can compute the posterior probability of a sample for every class. Clearly, the sample is placed in the class that corresponds to the largest probability.

3 Model Evaluation

The above two techniques were used to classify the MNIST images. The data set has 60,000 training samples and 10,000 test samples. We want to evaluate our models on how much input data they need to achieve their threshold of accuracy, as well as what that threshold is. Clearly as with most machine learning algorithms, the more training data our model consumes, the better the predictions.

Figure 2 shows our results for the algorithms, including multiple values for k that were used for the number of nearest neighbors to consider. We see that that the max likelihood beats the k-nearest neighbors, if only by a small margin. This is likely because of the fact that KNN is a clustering algorithm, and therefore has some trouble with the boundary cases between classes. With hand written digits, it is not hard for a 4 to look like a 9 or a

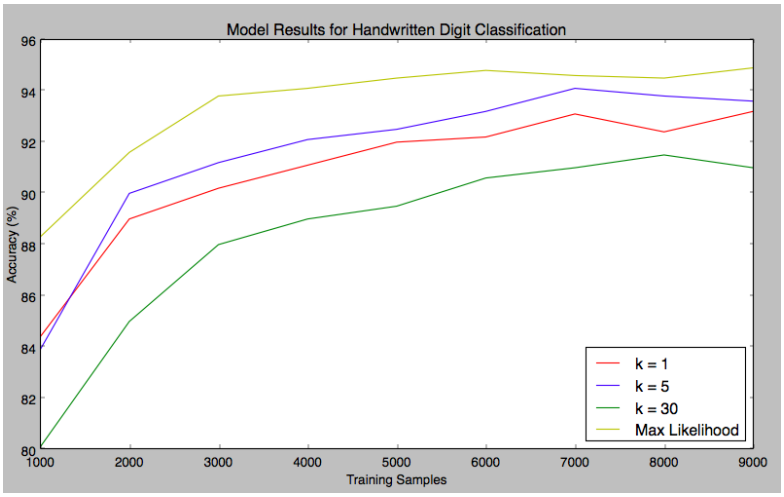

Figure 2: Model Results for K-Nearest Neighbors and Max Likelihood

5 to look like an 8. Because max likelihood uses the mean and variance of the two classes, it is better at handling these somewhat ambiguous cases.

4 Conclusion

PCA often requires reducing the dimensionality of data while simultaneously minimizing the amount of useful information that we trim away. For this reason, we used eigenvectors and eigenvalues of the covariance matrix.

There are many ways of classifying the MNIST data set. The technique of max likelihood works very well because of its assumption that the data follows a Gaussian distribution, which more or less panned out for this case.