Supervised and Unsupervised Machine Learning Methods For Handwritten Digits Recognition

Anna Mengjie Yu[1,2]*

1. Department of Integrative Biology, The University of Texas at Austin, Austin, USA
2. Department of Statistics and Data Sciences, The University of Texas at Austin, Austin, USA

* annayu2010@gmail.com

Abstract

Digit recognition is an active topic in optical character recognition and pattern recognition research. In this project, both unsupervised and supervised machine learning methods were tested for hand written digit recognition. Principal component analysis was used to reduce the dimensions of hand written digit data from MNIST database. Data exploration was performed using unsupervised learning method K-Means clustering. Supervised learning methods K Nearest Neighbour (KNN) and Support Vector Machine (SVM) were tested for classification accuracy. The results show that prediction accuracy increases as the number of training images increases, however, at the tradeoff of computation time. The increasing number of top eignevectors increases the prediction accuracy, however, as the eigenvector number goes above 80, it adds more noise instead of signal. The highest prediction accuracy 98.13% was achieved using SVM with Gaussian kernel, outperforms the result of KNN of 96.89% accuracy.

Introduction

Digit recognition is an active topic in Optical Character Recognition (OCR) applications and pattern recognition research. It is dealt in many fields such as postal mailing sorting, bank check processing, *etc*. Various approaches have been proposed to improve the accuracy of digit recognition accuracy, including multilayer neural networks, support vector machines, nearest neighbor methods.

Principal Component Analysis (PCA) is a popular method for dimension reduction and information extraction. It uses orthogonal transformation to convert observations to linearly uncorrelated variables, *i.e.* principal components, then calculates the eigenvalues and corresponding eigenvectors of the covariance matrix. The proportion of the variance of each eigenvector is proportional to the corresponding eigenvalues.

K-Means clustering is an unsupervised machine learning method that partions observarations into k clusters aming at minimizing the within-cluster sum of squares (sum of distance of each point in the cluster to centroid).

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

where μ_i is the mean of points in S_i.

Supervised learning analyzes labeled training data and produces an inferred function, which is used for testing new examples. K-nearest neighbor classifier (KNN) classifies unknown by relating the unknown to the known using distance function. KNN is a brute-force computation of all pairs of points in the dataset. If k = 1, it simply assigns the unknown to the class of the nearest neighbor, also called the nearest neighbor algorithm.

Support vector machine (SVM), a binary classifier, searches the hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of each class from the hyperplane. Assuming we have n training samples and each sample is represented by xi with class labels yi, where $yi \in \{-1, +1\}$. SVM can be formulated as the following optimization problem:

$$\max \sum_{i=1}^{n} ai - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} ai\ aj\ yi\ yj\ [\phi(\mathrm{xi}), \phi(\mathrm{xj})]$$
$$s.t.\ \ 0 \le ai \le C, i = 1, \dots, n$$

$$\textstyle\sum_{i=1}^{n} ai\ yi = 0$$

where $\phi(x)$ is the feature vectors of input x, and C is the penalty. $K(xi, xj) = \phi(\mathrm{xi}) \cdot \phi(\mathrm{yi})$ is the kernel function. Additional kernel functions enable SVM to operate in a high-dimensional space by simply computing the inner products between all pairs of data in the feature space. Three most common kernel functions are:

Linear Kernel:	$K(x, x') = x \cdot x'$
Polynomial Kernel:	$K(x, x') = (1 + x \cdot x')^p$
Gaussian Kernel:	$K(x, x') = \exp(-\| x - x' \|^2)$

Algorithms like KNN, have zero training time but are expensive during runtime. Support vector machines have the opposite problems requiring huge amounts of training data and time to learn good models.

In this project, PCA is used to reduce image dimensions and extract features. K-Means clustering is applied on testing images to further explore data. KNN and SVM with different kernel methods are evaluated on their performance of accuracy prediction. The effects of training image number, top eigenvector number on prediction accuracy are also evaluated.

Method

The data used in this project can be downloaded from MNIST database (http://yann.lecun.com/exdb/mnist/). The data consist of 60,000 training images, and 10,000 testing images. Each sample represents a digit ranging from 0 to 9, and is centered in a gray-scale image with size 28 pixel x 28 pixel.

Principal Component Analysis (PCA) was used to extract digit information and project digits to low dimension spaces. It first normalizes the matrices by subtracting the column vector mean, then calculates the covariance matrices. Eigenvector matrices are returned with the corresponding eigenvalues sorted in descending order, and normalized to be unit vectors. The testing images were reconstructed using the equations: $I' = V * V^T * (I - m)$, where V is the eigenvector matrix and m is the average column vector of the training data.

```
% mean column vector of A with dimension 2
vectorM = mean(matrixA,2);

% subtract the column mean from matrixA; mean normalization
matrixA = double(matrixA) - repmat(vectorM,1,k);

 [matrixV, matrixD] = eig(matrixA * matrixA');

 %sort the eigenvectors in descending order, get index
 [matrixD_sorted,Index] = sort(diag(matrixD),'descend');
 matrixV = matrixV(:,Index);

 %normalize the vector
 matrixV = normc(matrixV);
```

K-Means clustering first initialize random centroids for 10 digit classes, 10 clusters are created by associating every data point with the nearest centroids. The centroids are then updated by the new mean, the processes are repeated till convergence. Top 2 principal components were selected for visualizating K-Means clustering result, a series of maximum iteration number [1, 10, 100] were tested for the convergence for 400 testing images.

```
 %select top 2 PCs
 X = matrixV(:,1:2);

 % Initialize first centroid. Specify k = 10 clusters.
rng(1); % For reproducibility
[idx,C] = kmeans(X,10);

% Use kmeans to compute the distance from each centroid to points on a grid.
% To do this, pass the centroids (C) and points on a grid to kmeans,

x1 = min(X(:,1)):0.0001:max(X(:,1));
x2 = min(X(:,2)):0.0001:max(X(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)]; % Defines a fine grid on the plot

% Assigns each node in the grid to the closest centroid
[idx2Region,C,sumd,D] = kmeans(XGrid,10,'MaxIter',100,'Start',C);
```

For K Nearest Neighbour classification, a certain number of training images [500, 1000, 2000, 4000, 8000, 20000, 40000] were randomly selected from the total training set, and the prediction accuracy were tested with a combination of Ks [1, 5, 10, 20, 40, 100]. All the recognition accuracies were calculated at the selection of top 100 eigenvectors. Besides, a combination of different top eigenvectors [10, 50, 100, 200, 300, 400, 500, 600, 700, 782] were selected at the fixed training image number 4000, and calculated the corresponding accuracy.

```
% run KNN to get to predict the labels.
% evaluate the test samples using N training samples
[D, I] = pdist2(trainImages_p(:,1:ImageNum)',testImages_p','euclidean','Smallest',k);

if k == 1 %do not need to vote
    predict = double(trainLabels(I));
else
    % vote for the most frequent predicted digit
    predict = mode(double(trainLabels(I)));
end

% estimate the accuracy
k_vectorNum_result(j,i) = sum(predict - double(testLabels) ==0) / 10000
```

Support Vector Machine (SVM) classifier with three different kernel functions were tested to classify handwritten digits. Original digit data and PCA processed digit data were tested with the same kernel method to compare the accuracy.

Matlab built in SVM functions were used. templateSVM( ) returns a support vector machine learner template for multiclass models, and kernel functions can be chosen from three available options (Gaussian, linear or polynomial), or self defined. Gaussian kernel was specified with kernel scale set to auto. Polynomial kernel was specified with polynomial order set to 2.

Fitcecoc( ) reads predictor matrix and predictor labels, and returns a full, trained error-correcting output codes multiclass model. It implements a one-versus-one coding design, creates K(K-1)/2 binary SVM models, where K is the number of unique class labels. Predict( ), predicts labels for multiclass classifiers.

```
% Build SVM template
t = templateSVM('KernelFunction','gaussian','KernelScale','auto');

% Build multi-class SVM classifier with corresponding kernal function
Mdl_PCA_kernel = fitcecoc(trainImages_p',trainLabels', 'Learners',t);

% predict test images
testImages_p = double(testImages_p);
predicted_labels = predict(Mdl_PCA_kernel, testImages_p');

trueLabels = testLabels';

%calculate accuracy
accuracy = sum(double(predicted_labels) - double(trueLabels) ==0)/10000;
svm_PCADigit_Ns_Vs(i,j) = accuracy
```

Result

1. PCA reduced data

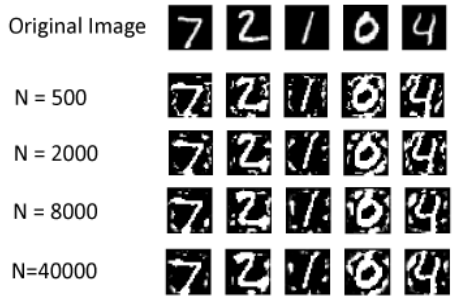

Figure 1. Reconstructed Image On Principal Component Plane

The original pixel data were projected onto the principal component axis plane. From Figure 1 we can see the majority of the digit information was retained on the image. The reconstructed image become more clear with increasing number of training images.

2. K-Means Clustering

The first 2 principal components were used to test for K-Means clustering. In Figure 2, the top image shows the initiation of the clusters of 10 different digits, each indicated by different colors. When the number of iteration increases to 10, the shape of each clusters changes, as indicated by the middle image in Figure 2. When iteration number increases to 100, the shape of each clusters stays very similar to the cluster shape with 10 iterations.

From Figure 3, we can see the within cluster sum of distance from each data point to its centroid decreases as the number of iteration goes up. When iteration number equals 1, the initiation stage, the within cluster sum of distance is large for several clusters (indicated by the blue color). When iteration number increases to 10, the within cluster sum of distance is more similar for each cluster. When iteration number further increases to 100, the within cluster sum of distance does not change much.

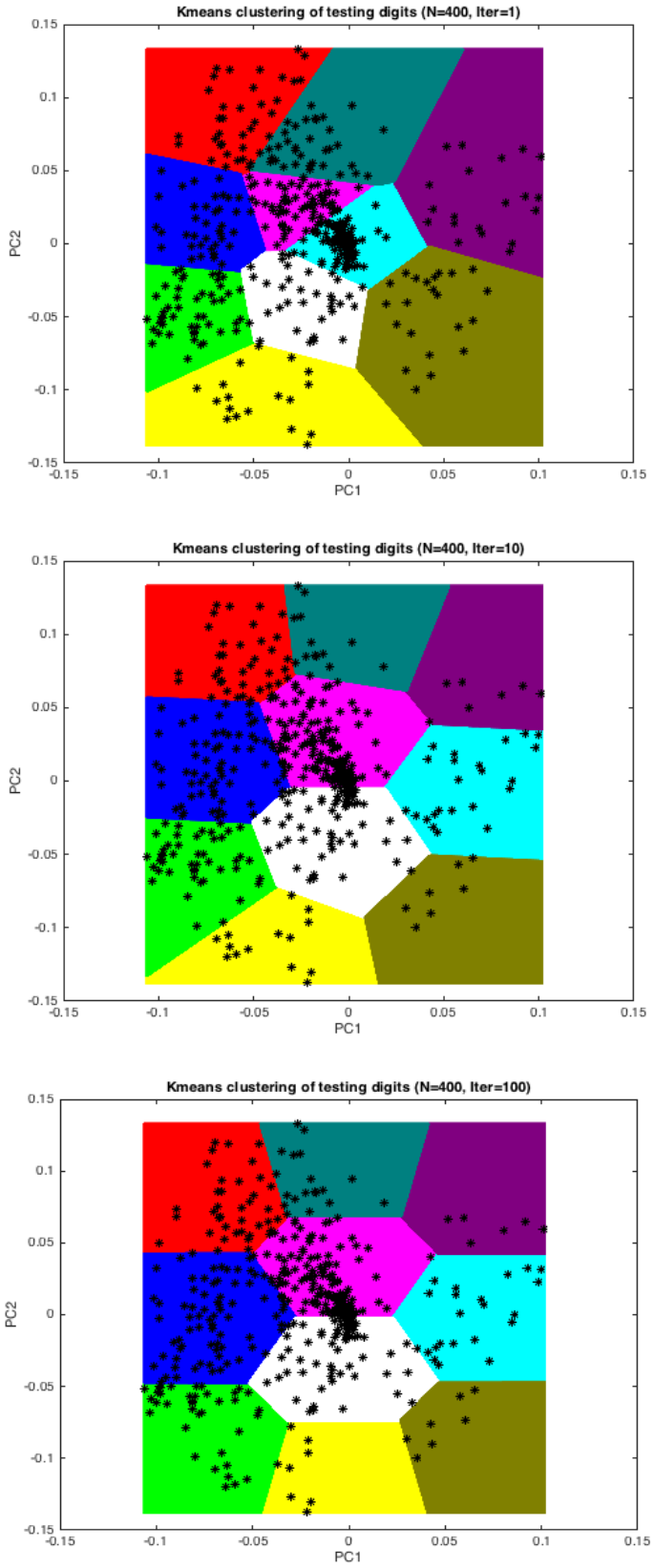

Figure 2. K-Means Clustering on Top 2 Principal Component Axis with Different Numbers of Iterations

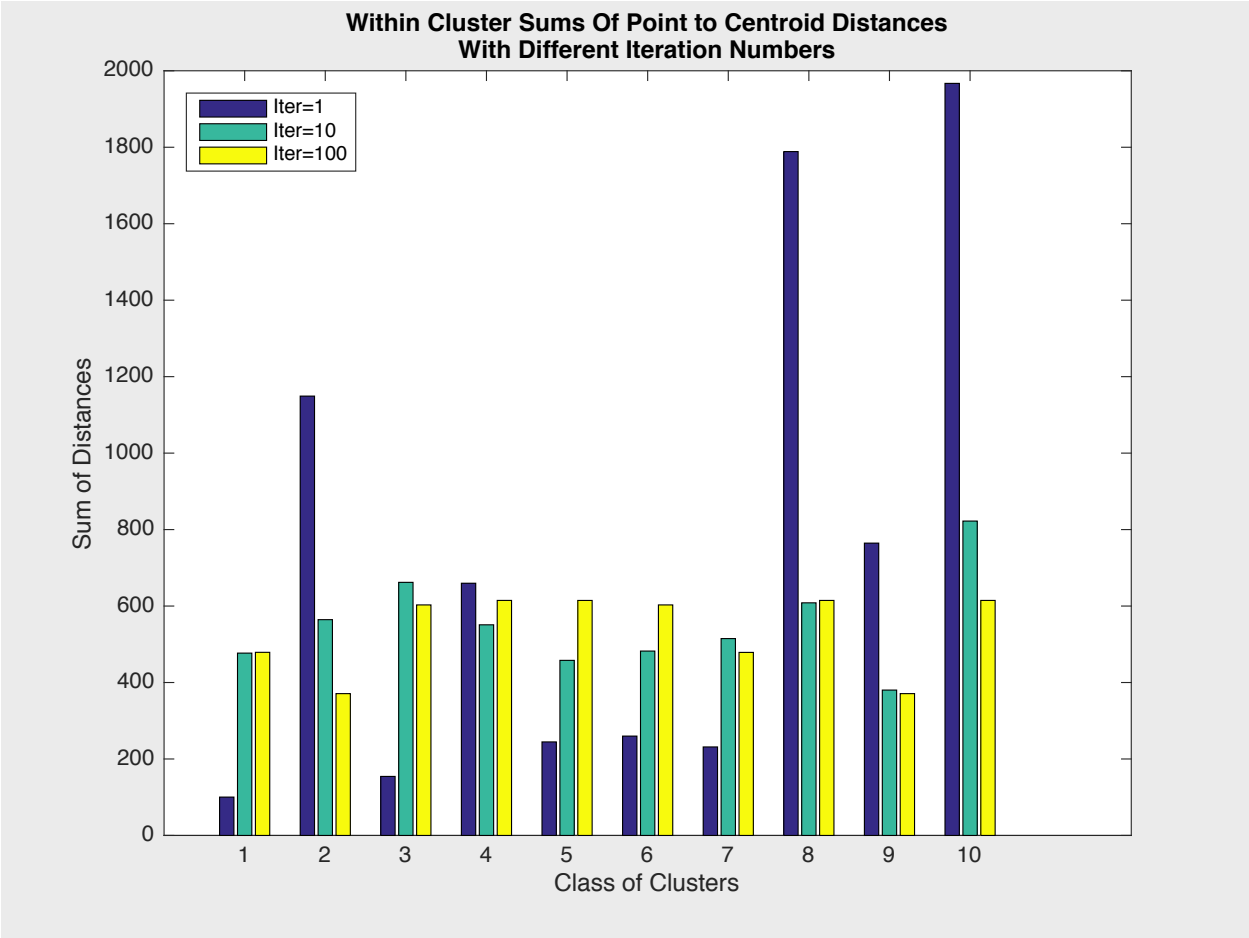

Figure 3. With Cluster Sum of Point to Centroid Distances with Different Iteration Numbers

3. K Nearest Neighbour Classification

From Figure 4 and Supplementary Table 1, we can see that the prediction accuracy increases as the number of training image increases. The prediction accuracy increases most rapidly when the number of training images increases from 500 to 4000, and accuracy increasing slows down as more training images were added. From Supplementary Table 1, we can see that the highest accuracy is achieved with the largest number of training images.

From Figure 5 and Supplementary Table 2, we can see that when the top N eigenvector number increase from 10 to 80, the accuracy increases, however, as more eigenvectors included, the accuracy stays stable or drops slightly, indicating the less important eigenvectors might add noise, rather than signal, to the data.

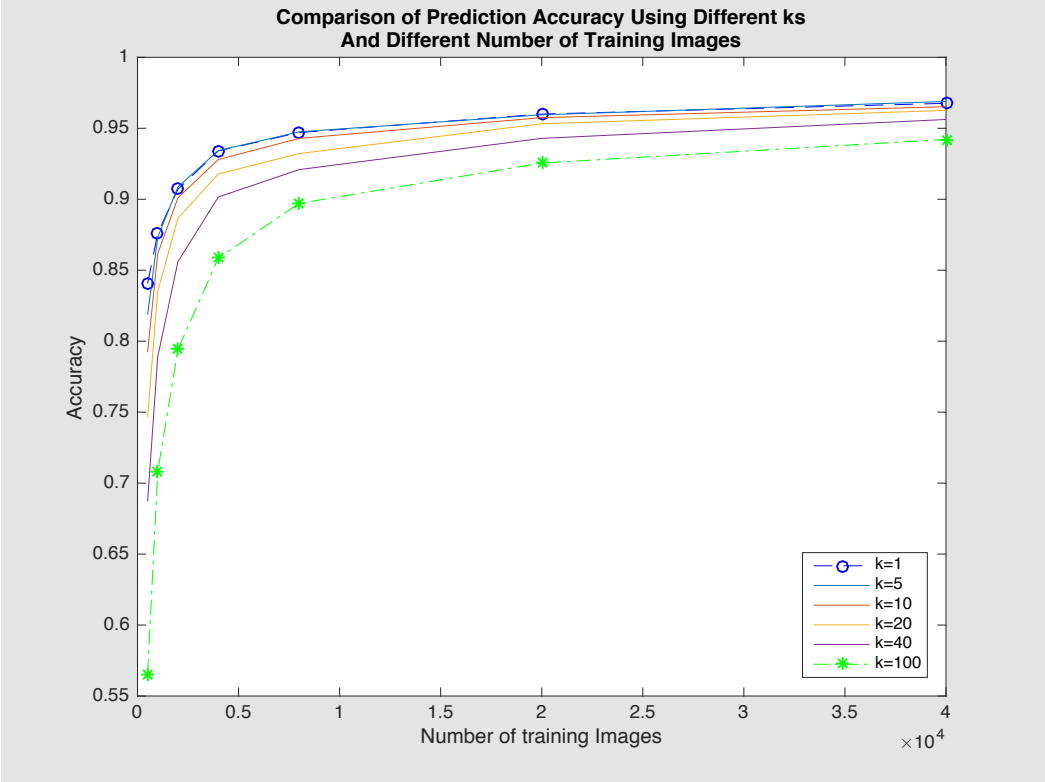

Figure 4. Prediction Accuracy with Different Number of Training Image and Ks

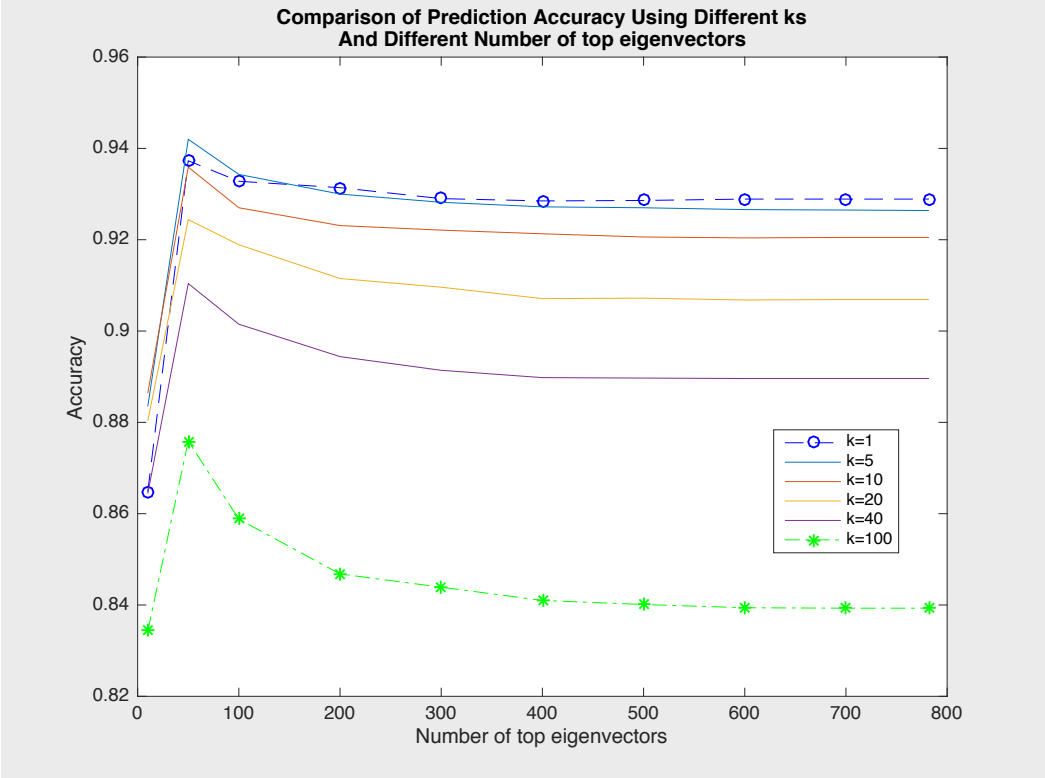

Figure 5. Prediction Accuracy With Different Number of Top Eigenvector and Ks

4. Support Vector Machine Classification

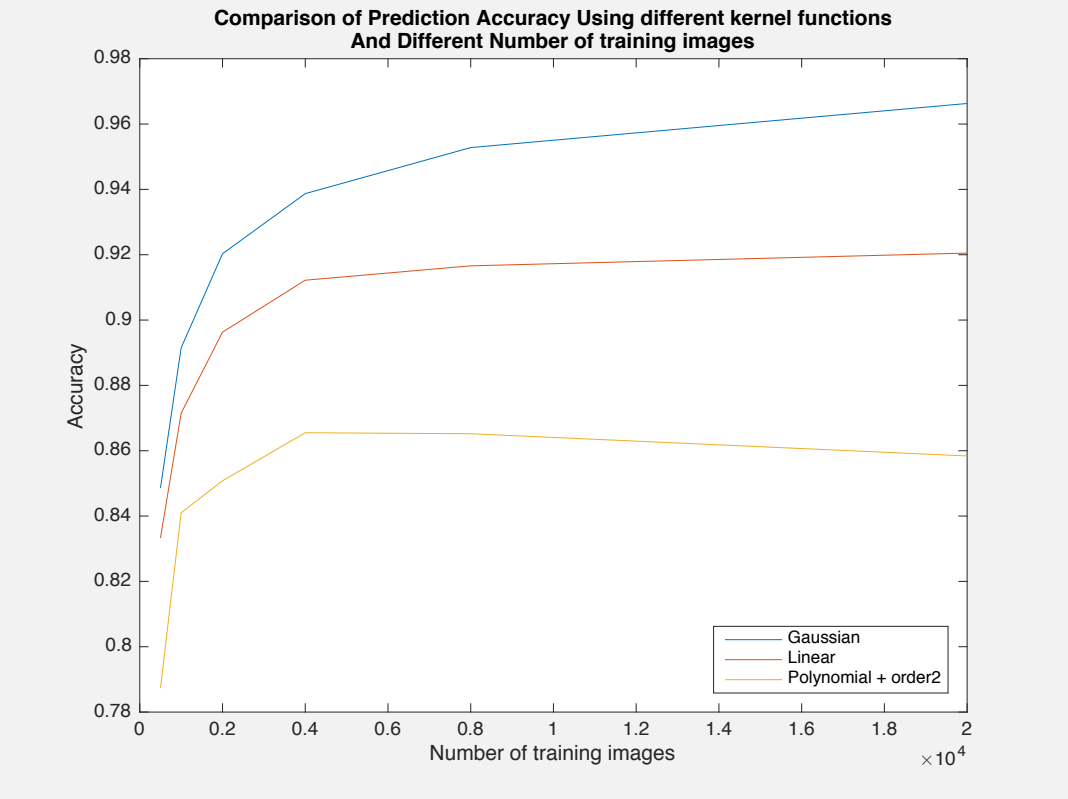

Figure 6. Comparison of accuracy using different kernel functions on original data

Comparing the performance of Gaussian, linear and polynomial kernels, we can see Gaussian kernel achieved the highest accuracy from 84.86 % (training Image number = 500) to 96.63% (training Image number = 20,000) (Figure 6, Supplementary Table 3). Linear kernel achieved 92.05% accuracy (training Image number =20,000), and polynomial kernel only achieved 85.54% accuracy (Figure 6, Supplementary Table 3).

From Figure 6, we can see that the accuracy increases sharply as training image number increases from 500 to 1000 (Supplementary Table 3). The accuracy increase slows down when training image number goes to 4000. In Gaussian kernel, we can see the accuracy still keeps going up as training image number further increases (Figure 6). In general, we see a trend of increasing accuracy with increasing number of training images (Figure 6, Supplementary Table 3). From Table 1, we can see the digit 1 has the highest classification accuracy among all the 10 digits. And digit 7 and digit 9 are mostly difficult to distinguish, with 19 misidentification cases between them (Table 1).

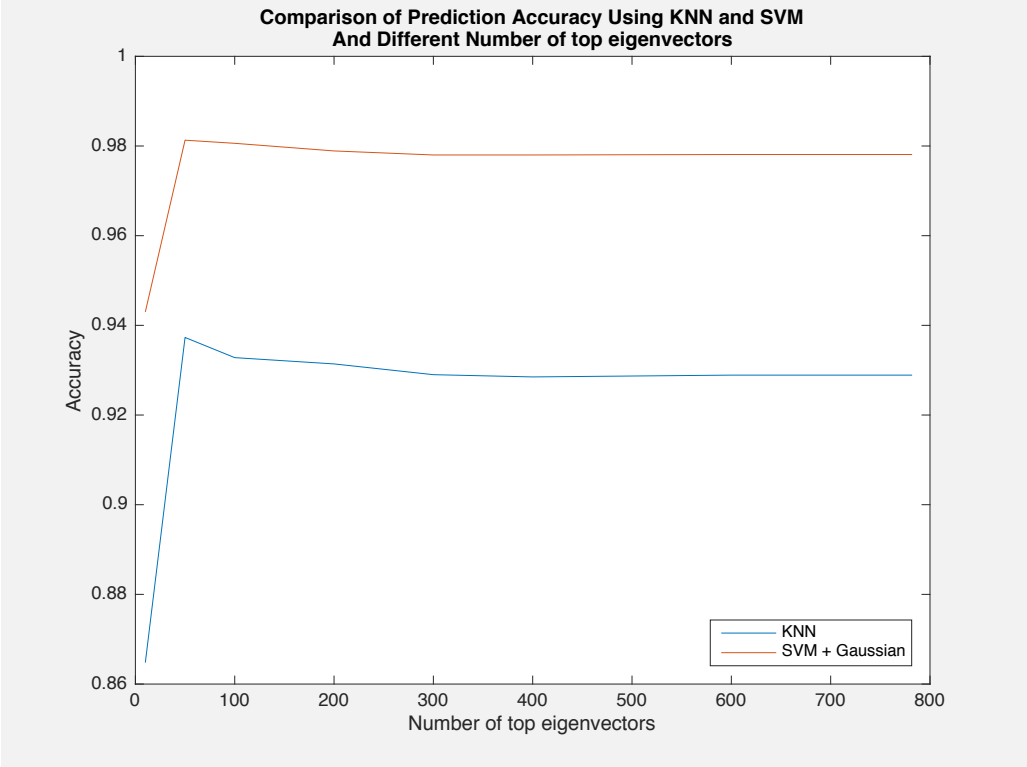

Figure 7. Comparison of accuracy using different top eigenvector numbers on PCA reduced data (training image number = 4000)

		Predicted Class									
		0	1	2	3	4	5	6	7	8	9
True Class	0	968	0	2	0	0	4	3	1	2	0
	1	0	1120	3	3	0	2	2	0	4	1
	2	5	0	992	4	4	2	3	9	12	1
	3	0	0	4	979	0	8	0	9	8	2
	4	1	0	3	0	952	0	5	1	2	18
	5	4	1	1	18	2	847	9	1	6	3
	6	6	3	3	1	6	8	926	0	5	0
	7	1	6	18	5	4	0	0	972	3	19
	8	3	0	4	10	4	5	0	4	942	2
	9	4	7	3	7	13	2	0	5	8	960

Table 1. Confusion matrix of SVM + Gaussian Kernel with 20,000 training images on original data

The best performed SVM kernel function was compared with KNN on the effect of different top eigenvector numbers (Figure 7), with SVM outperforming KNN with 98.13% accuracy at top 50 eigenvectors and 4000 training images (Supplementary Table 4). Both KNN (data from hw1) and SVM showed a similar trend that as top eigenvector increases to around 80, the accuracy increases. However, as top eigenvector number further increases, the accuracy drops (Figure 6, Supplementary Table 4).

When training image number is 4000, comparing Supplementary Table 3 and Supplementary Table 4, we can see the accuracy for original data with Gaussian kernel is 93.87. However with PCA processed data, the Gaussian kernel accuracy ranges from 94.30 to 98.13% with different top eigenvector numbers, higher than original data.

Conclusion and Discussion

In this project, the digit prediction accuracy of different machine learning methods, KNN and SVM were tested with various of parameters explored. The effect of training image numbers, data processing, and top eigenvector numbers on prediction accuracy were also tested.

In PCA orthogonal linear transformation, the great variance by the projection of the data lies on the first principal component, the second greatest variance on the second coordinate, and so on. From our results, for both KNN and SVM, the prediction accuracy increases as the top number of eigenvectors increase from 50 to 100, then the accuracy stays stable or drops slightly. This shows the majority of useful information is captured in first 100 top eigenvectors, and more eigenvectors of less importance might introduce noise, rather than signal. Both KNN and SVM show increasing accuracy with increasing number of training images.

K-nearest neighbor classifier (KNN), a brute-force computation of all pairs of points, classifies unknown by relating the unknown to the known using distance function. KNN has zero training time but are expensive during runtime. In KNN, the increase of K will reduce the effect of noise on the classification, however, large values of k makes the boundaries between classes less distinct. That is why we see a trend of decreasing accuracy when k gets large.

SVM requires large amount of time to learn the classifier. Our result indicates that support vector machine with Gaussian kernel outperforms linear and polynomial in classifying digits. We also see data preprocessing (PCA) helps increase classification accuracy.

Comparing KNN with SVM, SVM with Gaussian kernel outperforms at training image number 4000 achieving accuracy of 98.13%, higher than that of KNN (96.89%). Other classifier like neural network has also been tested in reported researches. In neural classifier, the parameters of neural networks are optimized in discriminative supervised learning to separate patterns of different classes. However, neural network takes very long time to train the model, and the weights are hard to interpret. In future work, it might be interesting to compare neural network performance with SVM. Distributed computing tools such as OpenMP, MPI, Hadoop might also be helpful in reducing the cost of computation time.

Supplementary

Supplementary Table 1: Prediction Accuracy using different ks and training image numbers (topN eigenvector = 100)

Accuracy	imageNum	500	1000	2000	4000	8000	20000	40000
K = 1		0.8406	0.8760	0.9071	0.9341	0.9470	0.9600	0.9676
K = 5		0.8187	0.8731	0.9087	0.9343	0.9475	0.9595	0.9689
K = 10		0.7923	0.8615	0.9010	0.9280	0.9429	0.9575	0.9652
K = 20		0.7464	0.8351	0.8866	0.9178	0.9322	0.9532	0.9626
K = 40		0.6871	0.7889	0.8560	0.9016	0.9209	0.9429	0.9562
K = 100		0.5654	0.7080	0.7941	0.8586	0.8971	0.9255	0.9422

Supplementary Table 2: Prediction Accuracy using different ks and top eigenvector numbers (training image number = 4000)

Accuracy	topN	10	50	100	200	300	400	500	600	700	782
K = 1		0.8648	0.9373	0.9328	0.9314	0.9290	0.9285	0.9286	0.9289	0.9289	0.9289
K = 5		0.8835	0.9420	0.9343	0.9300	0.9282	0.9272	0.9270	0.9266	0.9265	0.9264
K = 10		0.8864	0.9359	0.9270	0.9231	0.9221	0.9213	0.9206	0.9204	0.9205	0.9205
K = 20		0.8803	0.9244	0.9189	0.9115	0.9096	0.9071	0.9072	0.9068	0.9069	0.9069
K = 40		0.8644	0.9104	0.9015	0.8944	0.8914	0.8898	0.8897	0.8896	0.8896	0.8896
K = 100		0.8343	0.8759	0.8588	0.8468	0.8439	0.8410	0.8401	0.8394	0.8393	0.8393

Supplementary Table 3: Prediction Accuracy using different kernels and training image numbers (raw data)

Kernel	imageNum	500	1000	2000	4000	8000	20000
Linear		0.8333	0.8715	0.8963	0.9122	0.9166	0.9205
Gaussian		0.8486	0.8915	0.9203	0.9387	0.9528	0.9663
Polynomial		0.7873	0.8410	0.8505	0.8655	0.8652	0.8554

Supplementary Table 4: Prediction Accuracy with different top eigenvector numbers on PCA reduced data (training image number = 4000)

Accuracy	topN	10	50	100	200	300	400	600	782
KNN		0.8648	0.9373	0.9328	0.9314	0.9290	0.9285	0.9289	0.9289
SVM+Gaussian		0.9430	0.9813	0.9806	0.9789	0.9780	0.9780	0.9781	0.9781