Final Project

Comparative Analysis on Representation Methods for Classification

I. Introduction

This paper presents an analysis of various types of data representation methods used for classification problems. This comparative analysis has been done on the USPS handwritten digits dataset that is available here. The dataset consists of 1100 examples of handwritten digits, 0 to 9. Each example is shaped as a column vector of size 256x1. To obtain the measures of analysis, a series of experiments were executed on the dataset using combinations of dimensionality reduction techniques (such as PCA, kPCA, and LE) and non-linear classification algorithms (kNN and kSVM).

Pre-processing the data using dimensionality reduction techniques can make the task of analyzing and visualizing high dimensional datasets much easier and faster without losing much information while not affecting model's performance. Three of such techniques take the center stage in this comparative analysis of the USPS dataset – PCA, Kernel-PCA, and Laplacian Eigenmaps. To compare these techniques, the dataset was classified using k-nearest-neighbors and kernel-SVM algorithm, and for each algorithm, individual success rates for each digit were used as one of the measures of comparison between the techniques. The other measure of comparison considered in this paper is the global accuracy score of the dataset obtained from the kNN classification.

II. Raw Data

To obtain the raw performance of the model with no dimensionality reduction done on the data beforehand, an extra layer of analysis was added. In this section we look at the performance of the model built from raw data.

The dimensions of the raw data are 256x1100x10. The dataset was reshaped into 11000x256, such that each row is an observation, and each column is a dimension. This dataset was then split into training and testing set, where the training set consists of first 1050 examples of each digit and the testing set consists of the rest 50 examples of each digit. This structure of split has been proven to be an optimal train-test split in the previous two projects. Two classification algorithms were then implemented using the testing and training set: kNN and kSVM. The measures of success obtained from both classification techniques are discussed below:

a. kNN

Before performing kNN, validation was done on the training set. Validation helps to find an optimal number of nearest neighbors (k) that need to be found in order to

classify a query point in the testing set. For this reason, the training set of size 10500x256, was further split into two sets: 1) a subset of training set, consisting of first 900 examples of each digit and 2) a validation set consisting of the rest 150 examples of each digit. Using kNN classification, accuracies scores of k ranging from 1 to 20 were found for validation set using the subset of training set. Following fig-1 contains the outcomes of validation.

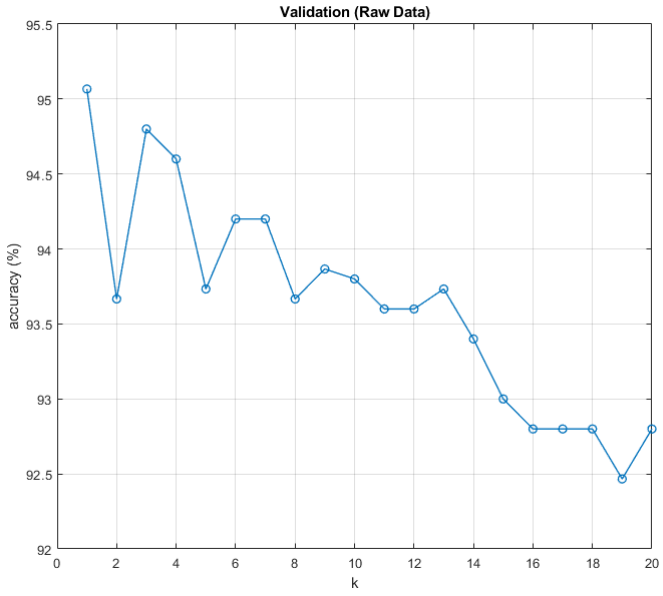

fig-1

In order to find an optimal k from the scores above, I decided to look for the 'k' for which the accuracy is closest to the mean. The reason for choosing such a 'k' even though k=1 would have the best accuracy is that considering just the 1st nearest neighbor out of 10500 possible nearest neighbors would mean ignoring rest of 10499 nearest neighbors which not the right thing to do because then the chances of a digit being classified correctly would be the same regardless of it being written legibly. If we consider k>1 nearest neighbors, the chances of digit being classified correctly would now be dependent on how legibly the digit has been written since more than one nearest neighbor are now considered. For my analysis, it is more important to understand the nature of the digit that gets misclassified instead of getting a better overall accuracy score. Using the idea above, **k=11** was found to be the optimal k.

Using this k=11, kNN using Euclidean distance metric was performed. Below are the results obtained:

- Global Accuracy Score: **93.2%** - not much different than the accuracy score obtained during validation. Hence, the validation set proved to be a good representation of our actual testing set.

- Individual success rates:

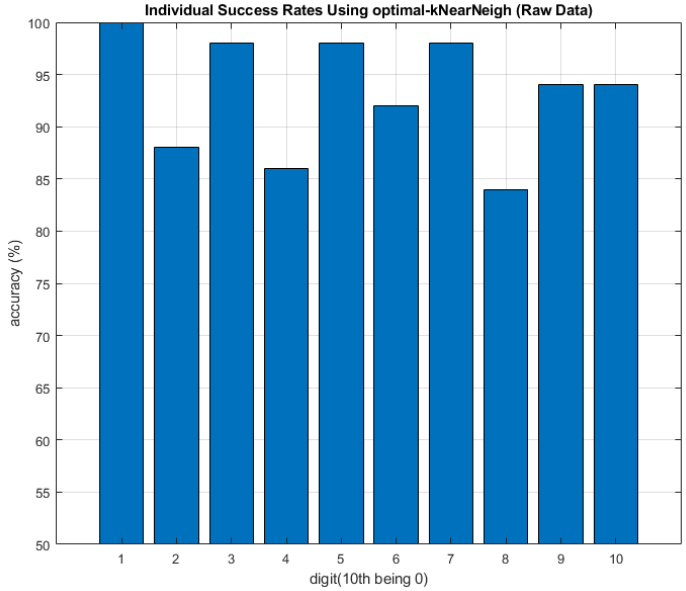

fig-2

- Digit-1 performs the best while digit-8 performs the worst.

b. <u>kSVM</u>

Kernel support vector machine was used as a second method of classification. Using the 'fitcsvm' function in MATLAB along with 'rbf' (gaussian) as the kernel function parameter, two-class classification was performed for each digit in the dataset using the test-train split specified in the last section. Hence, 50 examples of each digit were classified using a model that has been trained using the first 1050 examples of each digit. Following fig-3 depicts the outcomes:

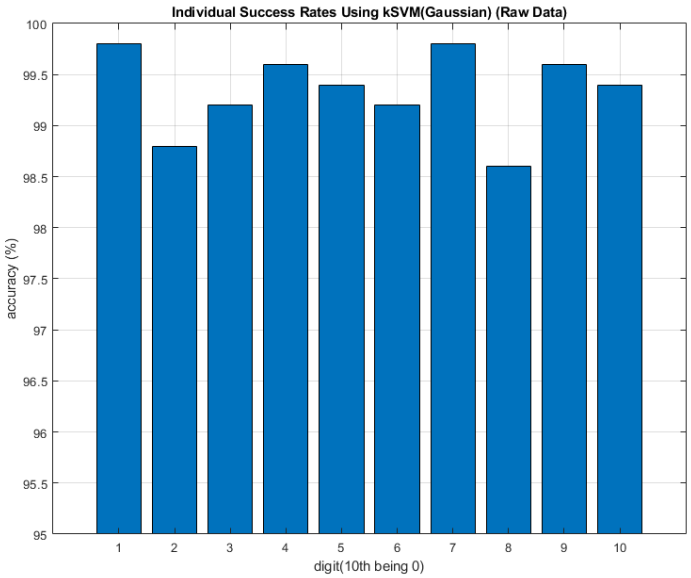

fig-3

- Once again, digit-1 performs the best while digit-8 performs the worst.
- Although, the overall accuracies seem to have increased if compared to fig-2.

III. Method 1: Principal Component Analysis (PCA)

After analyzing the performance of just the raw data, now we look at how the model performs after applying PCA on the whole dataset. To perform PCA, the dimensionality reduction toolbox called DR Toolbox that is available here was used. The dataset was reduced to 225 dimensions. Then, the dataset was split into training and testing set in a similar manner as last section (1050/50). Further, kNN and kSVM were performed to classify the handwritten digits. Following are the outcomes from our experiments:

a. kNN

Just like the last section, validation was performed to find an optimal 'k'. Following in fig-4, we have our results from validation.

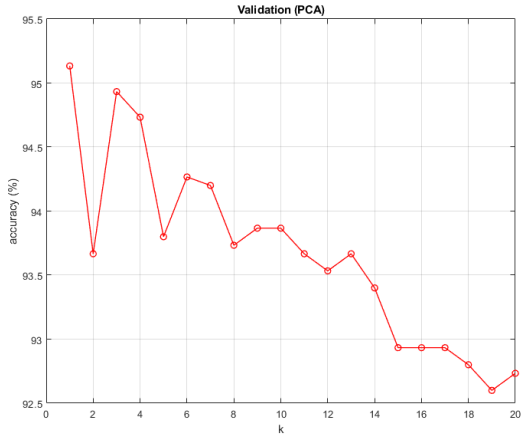

fig-4

The optimal 'k' for which the validation score was closest to the mean was found to be k=2. Using k=2, kNN was performed to classify the query points in the testing set and here are results:

- Global Accuracy Score: **92.4%**
- Individual success rates:

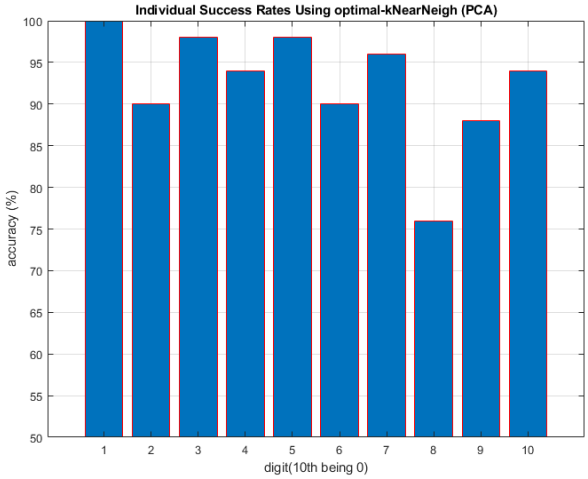

fig-5

- Digit-1 consistently performs the best and on the contrary, digit-8 performs the worst. Comparing the individual scores obtained from kNN after PCA to the scores obtained in the last section through kNN without PCA, we see a decline in overall individual scores of each digit except for digit-1.

b. kSVM

Just like section-I, using the 'fitcsvm' function in MATLAB along with 'rbf' (gaussian) as the kernel function parameter, two-class classification was performed for each digit in the dataset using the 1050/50 train-test split. Below are the results from this experiment:

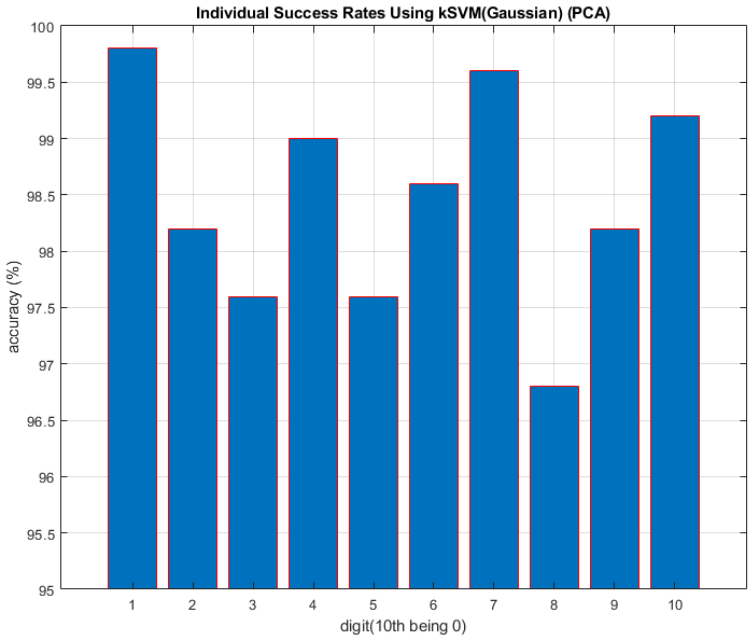

fig-6

- A common pattern can be seen so far, digit-1 topping the charts while digit-8 lying deep. The scores obtained above are significantly more than the scores obtained from kNN in part-a, almost 30% increase in scores for digits that usually perform poorly. Still, these scores are less than the scores we obtained from kSVM on raw data in section-I. Although, since the dimensions were reduced the classification computations were slightly quicker compared to previous section.

IV. Method 2: Kernel Principal Component Analysis (kPCA)

After performing PCA, we will now look at how the model built after kPCA performs. For this experiment, kPCA was performed on the whole dataset using kernel PCA function by M. Kitayama available here on mathworks, was used. Similar to section-II, the dataset was reduced to 225 dimensions using the linear-kernel PCA, and then split 1050/50 into train and test set. Further, kNN and kSVM were performed to get measures of success:

a. kNN

Just like previous sections, validation was performed to find the optimal-k for the kNN algorithm. Here are the results from validation:

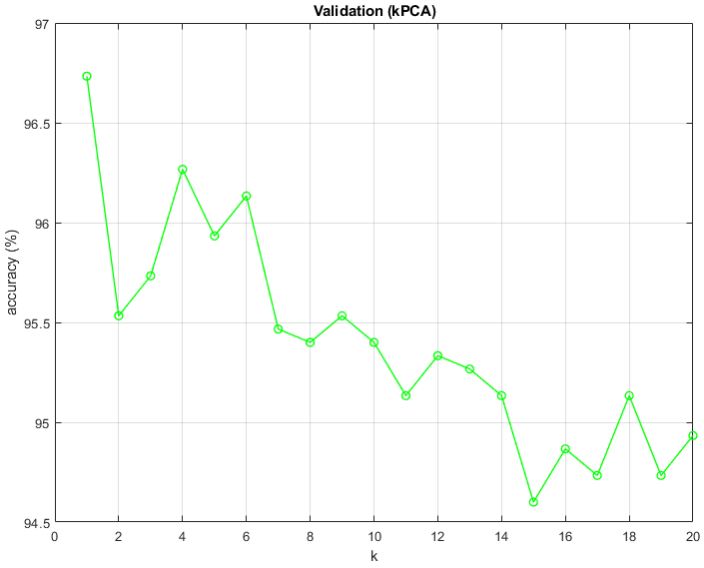

fig-7

Just by looking at the validation scores above, we can tell that overall accuracies for 'k' (1 to 20) obtained using kPCA is better than the ones obtained from raw data and after PCA. The optimal 'k' for kPCA processed data found using the same algorithm as previous sections II and III is k=8. At k=8, the accuracy in validation is closest to the mean of all validation accuracies. Hence, k=8 was used to perform kNN to classify the test set. Using the optimal k=8, following are the results:

- Global Accuracy Score: **94.0%** (best so far)
- Individual success rates:

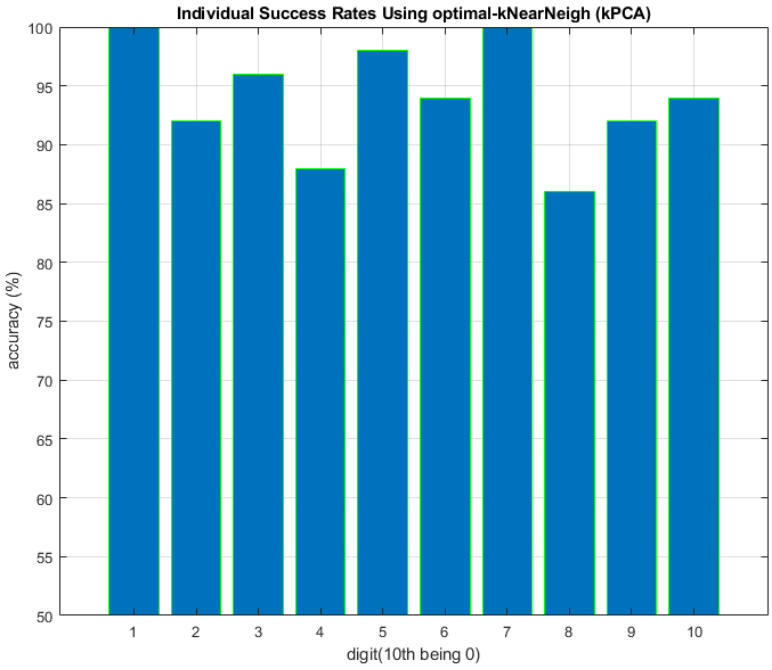

fig-8

- Individual success rates have risen quite significantly compared to the ones we obtained after just PCA or raw data, making the numbers above in fig-8, **best** so far.

b. kSVM

Just like section II and III, using the 'fitcsvm' function in MATLAB along with 'rbf' (gaussian) as the kernel function parameter, two-class classification was performed for each digit in the dataset using the 1050/50 train-test split. Below are the results from this experiment:

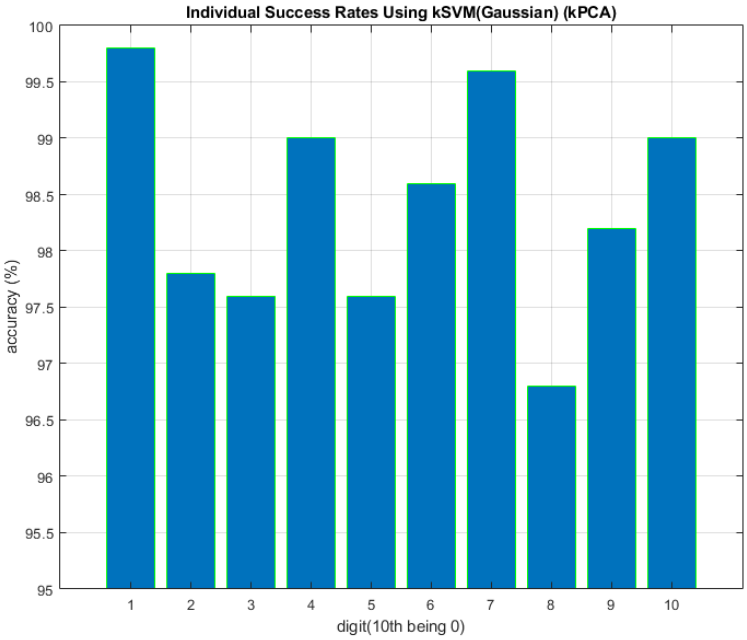

fig-9

- While the digit-1 and digit-8 maintain their positions in accuracy rankings, the accuracy scores for kSVM have risen as well compared to ones from raw data and non-kernel PCA. The numbers above are the best so far.

V. Method 3: Laplacian Eigenmaps (LE)

Finally, we will look at how the model built on dataset, for which the dimensions have been reduced using LE, performs. Just like section-I (PCA), DR Toolbox was used to apply Laplacian Eigenmaps dimensionality reduction technique to the whole dataset. The dataset was reduced to 225 dimensions. Then, the dataset was split into training and testing set in a similar manner as previous sections (1050/50). Further, kNN and kSVM were performed to classify the handwritten digits. Following are the outcomes from our experiments:

a. kNN

Again, validation was performed on the training set to find an optimal 'k' for kNN algorithm. Following fig-10 has the plot that depicts the results from validation:

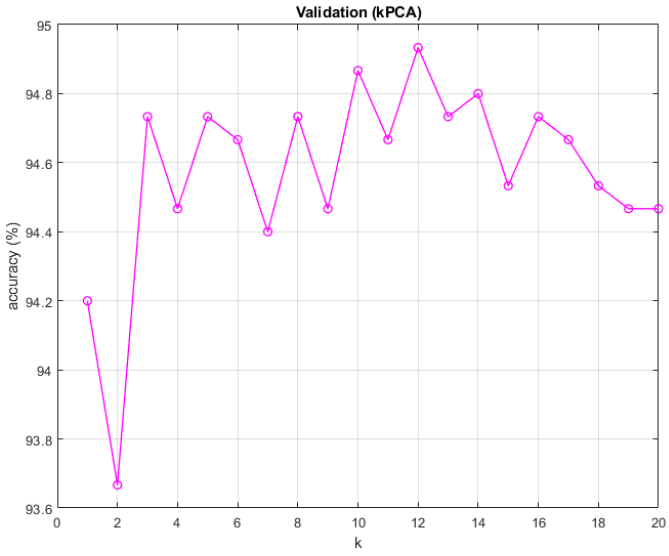

fig-10

From the results of validation, k=15 was chosen to be the optimal value of k. One thing to notice in this experiment is that the outcomes of validation after performing LE on the data differ significantly from the validation outcomes we found in the earlier sections (fig-1,4,7). In precious sections, we usually saw k=1 have the best accuracy and then the accuracy gradually drops. Whereas here the accuracy for k=1 is one of the worsts, and then accuracy drops for k=2 and then increases and continues to fluctuate between 94.5% and 95% (becomes almost constant). This is one of the reasons why blindly considering k=1 as the optimal k would yield inaccurate results.

Using k=15, kNN using Euclidean distance metric was performed to classify the test set. Following are the results:

- Global Accuracy Score: **93.4%**
- Individual success rates:

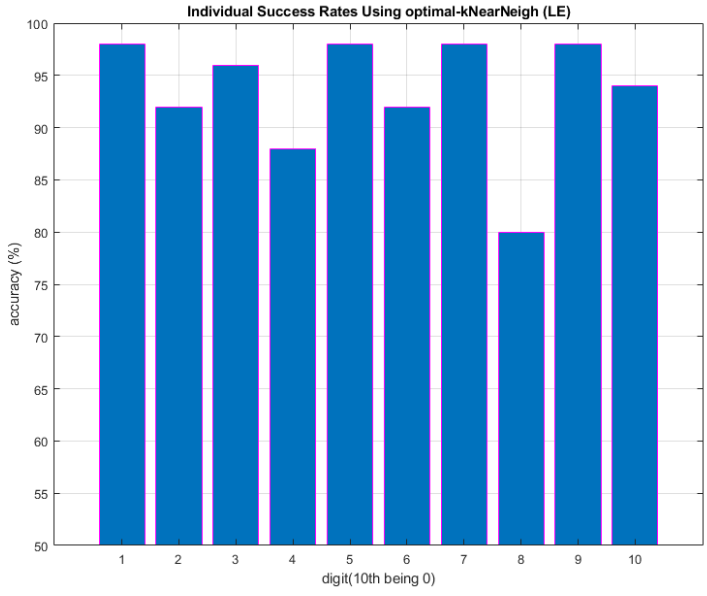

fig-11

- The individual success rates from kNN in this case could not reach as high as the ones from linear-kPCA or raw data, yet the numbers were still better than the ones obtained in PCA.

b. kSVM

Just like previous sections, using the 'fitcsvm' function in MATLAB along with 'rbf' (gaussian) as the kernel function parameter, two-class classification was performed for each digit in the dataset using the 1050/50 train-test split. Below are the results from this experiment:

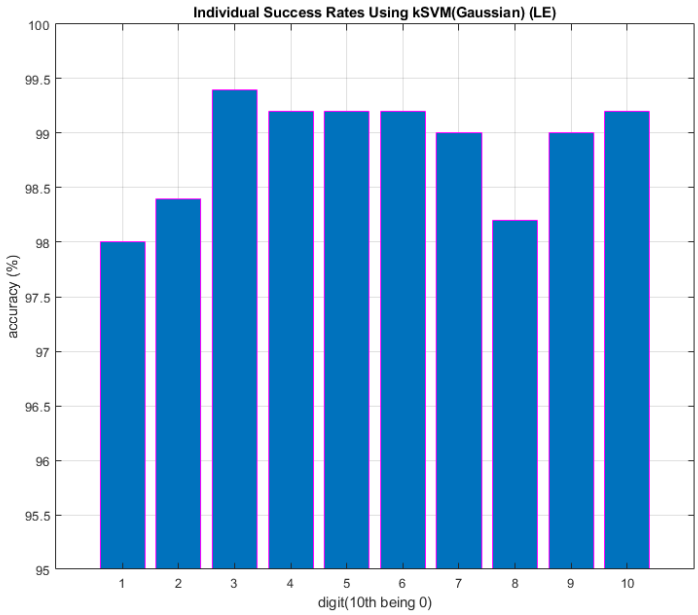

fig-12

- The above individual scores are much better than the individual scores from kNN in part-a. The individual scores here, compared to the ones obtained from raw-data, PCA and kPCA, are still one of the best. Although, this is only case where digit-1 managed to perform the worst, while digit-8 classification performing second worst.

VI. Conclusions

Following are the takeaways from the comparative analysis done on the USPS handwritten digits dataset:

- The best data representation technique amongst raw data, PCA, kPCA and LE is linear-kPCA. Linear kernel PCA resulted in the best global as well as individual-digit accuracy scores while managing to reduce the dimensions to 225, and hence performing faster computations than raw data. Raw data comes second in the pecking order because of the performance and more importantly, the computation time. Next, I would put LE. kNN model after performing LE did not produce ideal results but the results were still better than PCA. While the results from kSVM after performing LE were one of the

best. Hence, LE comes in third. Lastly, by all means, PCA performed not too bad, but still the worst of all four representation methods, considering mostly the accuracy scores.

- Next thing to note is that the accuracy scores from kSVM classification always gave better results than classification through kNN for all four data representation techniques.
- The above two points would mean that our data is evidently linearly separable.
- Lastly, digit-1 performed the best of all digits while digit-8 being the worst by some distance. Laplacian Eigenmaps being the only exception to this, where digit-1 performed very poorly compared to other digits.

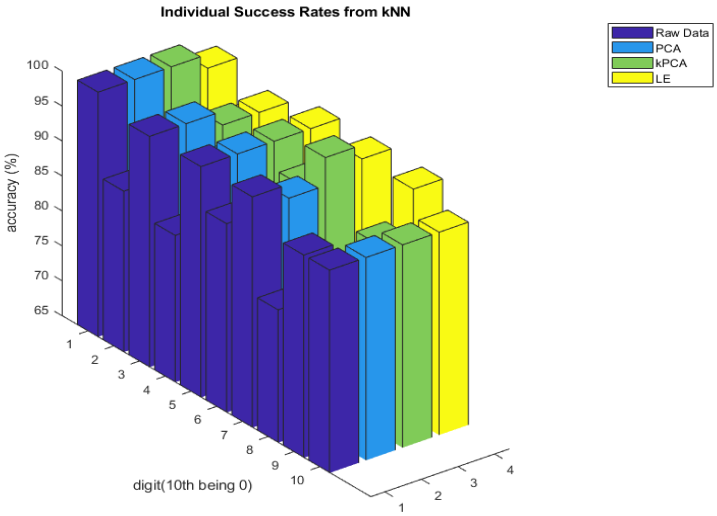

fig-13

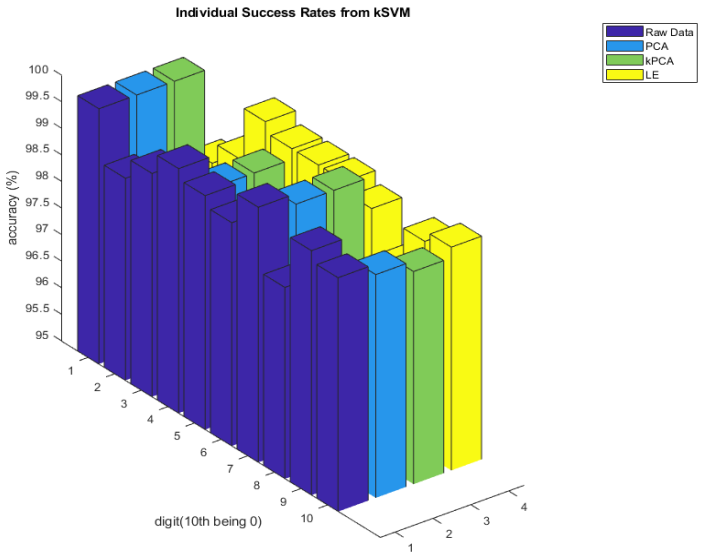

fig-14