Combining PCA and K-Nearest Neighbor methods in Handwritten Digits Classification

Billy Okal
Jacobs University Bremen
D-28759, Bremen, Germany
`b.okal@jacobs-university.de`

Abstract

The classification of handwritten digits has been investigated thoroughly in the past and different algorithms for classification have been used with various success rates. Extensive effort has also been directed at reducing the dimensionality of the data. The complexity of the algorithms used and the amount of work involved has also been shown to relate the success rates. Great success has been achieved using convolution neural networks and support vector machines among other methods. In this paper, I tackle the problem by first reducing the data dimensionality by extracting features using Principal Component Analysis (PCA) and then use K-Nearest Neighbors algorithm to classify the digits. Experimental results show high classification rates on a sample dataset.

1 Introduction

Classification of handwritten digits has been and continues to be an active topic of research in pattern recognition and Optical Character Recognition (OCR) applications. The vast array of applications such as sorting of postal mail, bank cheques, etc research in this field has attracted large attention as manifested by the volumes of literature tackling various aspects of the task. Different methods have been proposed to tackle the problem from various perspectives. The overall goal however remains to be improving the performance of the proposed techniques when deployed to 'real' world scenarios as these represent the ultimate benchmarks.

A common setup of the problem can be phrased as; given a dataset of handwritten digits with their corresponding 'group truth' values, design an algorithm to automatically classify new handwritten digits into the appropriate class. Such as dataset usually contains a large set of digit images, in this case 2000. These images are usually represented in various resolutions and sizes depending on the source, and in our case the digit images were of size 16×16 pixels. Samples of some of the digit images used in this project are shown in Figure 1. It is immediately clear that the size of such datasets are usually large and the constituents being of high dimensions[1], and one therefore naturally encounters the question of how to deal with the high dimensionality of the dataset of digits. As one would expect, extensive research has been carried out in this respect and a number of techniques have been proposed to try reduce the dimensionality by eliminating unnecessary information while retaining as much meaningful information as possible. A large crop of these techniques are based on the idea of extracting 'features', usually of much lower dimensionality from the original dataset and then only woking with such 'features'.

After tackling the problem of high dimensionality, one then has dataset of digits images with a considerably low dimensionality, and one then has to come up with algorithms for learning the

[1]Taken to be the number of pixels in this case (16×16)

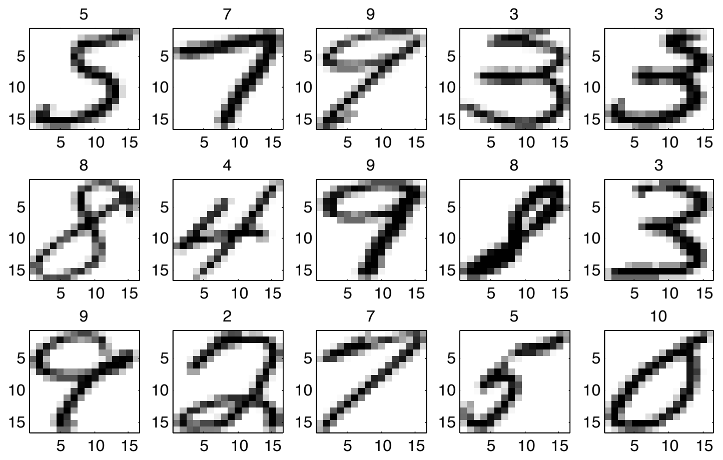

Figure 1: Sample of handwritten digits with corresponding labels

structure of the digits such that it is able to classify a new digits correctly[2]. Again a number of techniques have been proposed both of the flavor of supervised and unsupervised learning. The performance of such techniques varies across application disciplines and benchmarks and generally require integration carefully designed feature extraction techniques into the classification techniques. Most of these techniques involve learning models that capture the underlying structure of the data at hand and as such probabilistic and statistical techniques have proven indispensable in the search for well performing techniques. Most techniques are able to register considerably high classification rates on the datasets on which they are trained. However, the ultimate performance measure of these classification techniques however depends heavily on how well they perform on new digit sets, i.e. how we they generalize the process by which they capture the underlying structure of the data. Again a number of techniques have also been proposed to tackle this generalization problem and among them cross-validation is the most commonly used one. Others include regularization as well as manipulation of the original datasets in ways so as to capture this generality already in the dataset itself. For the case of images, this could include adding images to the dataset by distorting the existing ones in some specific fashion, such as by convolving with spatial operators to extract structural information.

2 Approach

The following sections outline the approach used to tackle the main problems involved in developing the classifier introduced in section 1 with emphasis on feature extraction, the actual classification and generalization of the learned models in subsection 2.1,subsection 2.2 and subsection 2.3 respectively.

2.1 Feature Extraction

Given the dataset of known digits represented as 16×16 pixels images, which one can unroll (concatenate the rows) into vectors $\mathbf{x}_i$ such that $\mathbf{x}_i \in \mathbb{R}^n, \quad n = 16 \times 16 = 256$. One thus requires a means to produce from the $\mathbf{x}_i$ new vectors $\tilde{\mathbf{x}}_i \in \mathbb{R}^p$ such that $p \ll n$.

I use Principal Component Analysis (PCA) to compress the data into a smaller dimension by retaining only a few the principal components as features, which also have the added benefit that they are de-correlated. The full theory of PCA is detailed in [Shl09], but I nevertheless give a overview of the concepts relevant to this task. More generally, the goal of PCA is to re-express the given original dataset in a new basis with the hope that this new basis filters out some of the noise that are inherently present in the data and also reveals part of the structure underlying the

[2]Understood usually as, low misclassifications on average over a large enough sample

data. There are number of assumptions made while applying PCA to a given dataset that are important to note, these are;

- *Linearity* - which allows the problem to be expressed as a change of basis, even though various efforts have been made to relax this constrains for instance by using Kernels as in Kernel PCA.
- *Large variance in the dataset have interesting structure* - which in turns implies that principal components with large variances have interesting structure and those with low variances are noise. The consequence of this assumption is that the dataset need to have a high signal to noise ratio.
- *Principal components are orthogonal* - which is mainly used to allow application of linear algebra techniques to find the solution.

The steps involved in the computation of the principal components and hence the basis are detailed as follows; I first write the vectors $\mathbf{x}_i \in \mathbb{R}^n$ into a matrix $\mathbf{X}$ of size $M \times n$, meaning I have M samples. In our case I have $n = 256,\ M = 2000$. I further use $\tilde{\mathbf{X}}$ to denote the matrix of $\tilde{\mathbf{x}}_i$ unrolled as the result that I want to achieve by PCA.

1. *Centering the dataset so that the resultant has zero mean.* Let $\mathbf{X}(i, j)$ be the pixel value at row i and column j. I then compute the column means using Equation 1.

$$\mu_i = \frac{1}{M}\sum_{j=1}^{M} \mathbf{X}(j, i) \qquad , \forall i \in [1, n] \tag{1}$$

   Then using $\mathbf{X}_0$ to denote the zero mean digits dataset, and $\vec{\mu}$ to denote the column vector of computed means. I center the original dataset using these means using Equation 2. The original dataset matrix has to be transposed to have the digits as column vectors.

$$\mathbf{X}_0 = \mathbf{X}^T - \mathbf{I}\vec{\mu} \tag{2}$$

2. *Covariance Matrix*. I then form the covariance matrix of the zero mean dataset by left-multiplying $\mathbf{X}_0$ by its transpose. The formal treatment of why this is done is explained in [Shl09].

$$\hat{\mathbf{\Sigma}} = \mathbf{X}_0^T \mathbf{X}_0 \tag{3}$$

   It is worthy to note that the matrix $\hat{\mathbf{\Sigma}}$ is square and symmetric, and the diagonal elements represent the variance between the measurement types (digit classes in this case) while the off-diagonal terms represent the covariance again between measurement types.

3. *Eigenvector decomposition of the Covariance matrix*. I then compute the eigenvalues and eigenvectors of $\hat{\mathbf{\Sigma}}$ as $\mathbf{\Lambda}$ and $\mathbf{U}$ respectively and order the eigenvalues and eigenvector pairwise in decreasing order of the eigenvalues. The columns of $\mathbf{U}$ then already form the new basis[3] required to represent the dataset.

4. *Representing the original dataset in the new basis*. I then select a small subset $\tilde{\mathbf{U}} \subseteq \mathbf{U}$ of the eigenvectors corresponding to dimension p and project the data from $\mathbf{X}_0$ onto $\tilde{\mathbf{X}}$ using Equation 4. This concludes the PCA procedure and now I have the dataset represented in a basis with much lower dimension p.

$$\tilde{\mathbf{X}} = \mathbf{X}_0 \tilde{\mathbf{U}} \tag{4}$$

2.2 Classification

Now given, a new digit image $d \in \mathbb{R}^{256}$, compute a class label $y \in [0, 9]$[4] to 'identify' the digit. This class labels were computed using K-Nearest Neighbors (K-NN) approach[TH09], which

[3]stated without proof as this is extensively shown in referred literature, esp [Shl09]

[4]Implementation note: I use $y \in [1, 10]$ as Matlab indices start from 1, and use 10 for 'zero'

involves using certain predetermined metric to compute distance of equivalently similarity between d and the digits in the original dataset, hereby referred to as neighbors. The implication is that d belongs the digits class for which the metric returns the smallest distance (or equivalently largest similarity). This technique's challenge then reduces to deciding on how many neighbors to consider when deciding the class membership and which metric to use for comparison. A rigorous treatment of some of the distance metrics used with K-NN is given in [WS09]. I however highlight some basic requirements of distance metrics.

Generally, a mapping $f : \mathcal{X} \times \mathcal{X} \longrightarrow \Re_0^+$ over a vector space $\mathcal{X}$ is a *metric* if $\forall \vec{x}_i, \vec{x}_j, \vec{x}_k \in \mathcal{X}$, the following hold (metrics satisfying only the first three conditions are known as *pseudo metrics*[5]);

1. Triangular inequality: $f(\vec{x}_i, \vec{x}_j) + f(\vec{x}_j, \vec{x}_k) \geq f(\vec{x}_i, \vec{x}_k)$
2. Non-negativity: $f(\vec{x}_i, \vec{x}_j) \geq 0$
3. Symmetry: $f(\vec{x}_i, \vec{x}_j) = f(\vec{x}_j, \vec{x}_i)$
4. Distiguishability/Uniqueness: $f(\vec{x}_i, \vec{x}_j) = 0 \iff \vec{x}_i = \vec{x}_j$

The choice of number of neighbors is made by experimenting with the data and observing the results. For selecting the metric, selected the following distance metrics to use in the classification task based on the properties outlined above. Assuming an n dimensional vector space. For each case the new digit d is first centered and projected onto the new basis so that the resulting digit is the desired one $\tilde{d} \in \mathbb{R}^p$ using Equation 5.

$$\tilde{d} = (d - \vec{\mu})\tilde{\mathbf{U}} \tag{5}$$

1. *2-Norm (Euclidean Distance)*. Which is the most common metric used with K-NN and is defined as given below;

$$f(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{\frac{1}{2}} \tag{6}$$

2. *Correlation Distance*. Which measures the statistical dependence between any two data vectors, and hence zero when the vectors are independent. As such, when used in K-NN it is defined as;

$$f(\vec{x}, \vec{y}) = 1 - \frac{(\vec{x} - \overline{x})(\vec{y} - \overline{y})^T}{\sqrt{(\vec{x} - \overline{x})(\vec{x} - \overline{x})^T}\sqrt{(\vec{y} - \overline{y})(\vec{y} - \overline{y})^T}}, \qquad \overline{x} = \frac{1}{n}\sum_i x_i, \;\; \overline{y} = \frac{1}{n}\sum_i y_i \tag{7}$$

3. *Cosine Distance*. Which defines the measure of similarity between any two given vectors by the angle between them.

$$f(\vec{x}, \vec{y}) = 1 - \frac{\vec{x}\vec{y}^T}{\sqrt{(\vec{x}\vec{x}^T)(\vec{y}\vec{y}^T)}} \tag{8}$$

4. *1-Norm (Manhattan Distance)*, defined as;

$$f(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{9}$$

With the distances computed for some k neighbors, I then decide on the class label by first plotting histogram of the class labels as they occur in the original dataset and select the label that occurs most frequently. This then forms a simple voting scheme among the k neighbors. The K-NN algorithms then reduces to simple steps summarized in algorithm 1 whose syntax is heavily influenced my MATLAB's syntax.

The procedure HISTOGRAM_LABELS(k), constructs a histogram of how frequent the given k labels appear on the dataset and returns the most popular. The procedure SORT(distance) simply orders the distances computed in increasing order and DIST_METRIC($\vec{x}$,$\mathbf{X}(i, :)$) computes the distance between the test digit and and a given sample of the known digits in the training set.

[5] http://en.wikipedia.org/wiki/Pseudometric_space

Algorithm 1: K-NN Search: Find k nearest neighbors to a given set

```
Input: $k, \vec{x}, \mathbf{X}$ // test sample and dataset
Output: label $y$
begin
    i = 1;
    distance = zeros(size($\mathbf{X}$,1)) ;
    while $i \leq size(\mathbf{X}, 1)$ // loop over rows of dataset
    do
        // Compute distances using a given metric
        distance(i) = DIST_METRIC($\vec{x}$,$\mathbf{X}(i, :)$) ;
    end
    // sort the distances in ascending order
    SORT(distance);
    // select k smallest distances and vote among them
    y = max(HISTOGRAM_LABELS(k)) ;
    return y;
end
```

2.3 Model Generalization

The generalization of the models when using K-NN algorithm reduces to proper selection of the number of neighbors k and the distance metric used. These are evaluated by partitioning the training dataset into disjoint sets for training and for testing and evaluating the different values of k and distance metric performance. This procedure is known as cross-validation. This can be done using various partitioning methods that divide the dataset into various proportions.

3 Experiments

The approach outlines in section 2 was implemented using the programming language MATLAB[MAT10]. Some of the routines and techniques used in the implementation of the PCA algorithm were based on [Nab02]. The dataset use was represented using `.txt` files, one for training data and another for corresponding labels.

To demonstrate the effect of PCA on the dataset, I plot the original dataset represented in a new basis formed by the first three principal components as shown in Figure 2. As one can notice from the plot, the projected dataset is highly 'glued' together such that separation into classes becomes difficult. One therefore has to use more principal components. For visualization, I cannot plot such a representation as we are limited to only three dimensions. I however, show how the resultant digits look after being represented in the new basis in Figure 3 showing the first 15 principal components of the digit '2'. Such digits are sometimes referred to as *eigendigits*, a term coined from one of the most popular application case of PCA in face recognition with eigenfaces[6].

The relation between the variance captured and the number of principal components used is depicted in Figure 4. From the plot, it can be seen that one can use as few as 65 principal components to capture already as much as 90% of the variance of the original dataset. In my experiments, I used 64 principal components as this gave best overall results with trading off between run times and performance.

With the number of principal components decided on, one then needs to decide on the number of neighbors to consider during the classification. To make this choice, I ran experiments with various values for the number of neighbors and compared the overall performance on the classification task. These results are shown in section 4 that follows.

[6] http://www.scholarpedia.org/article/Eigenfaces

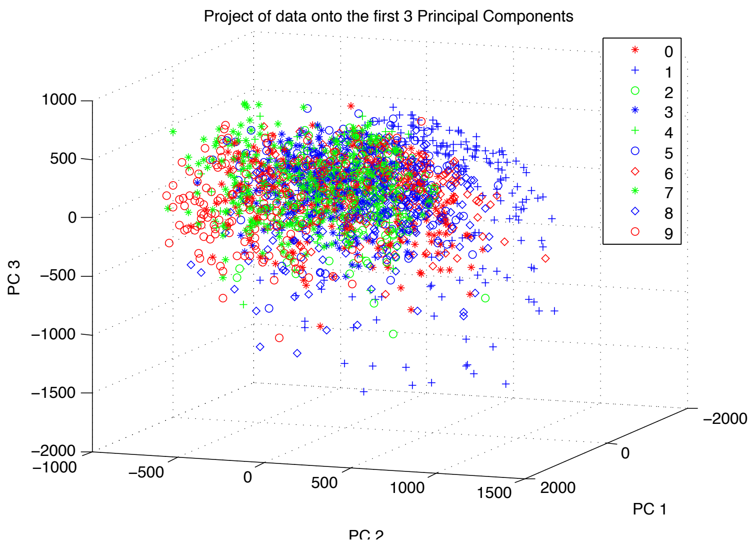

Figure 2: Training set project onto first two principal components

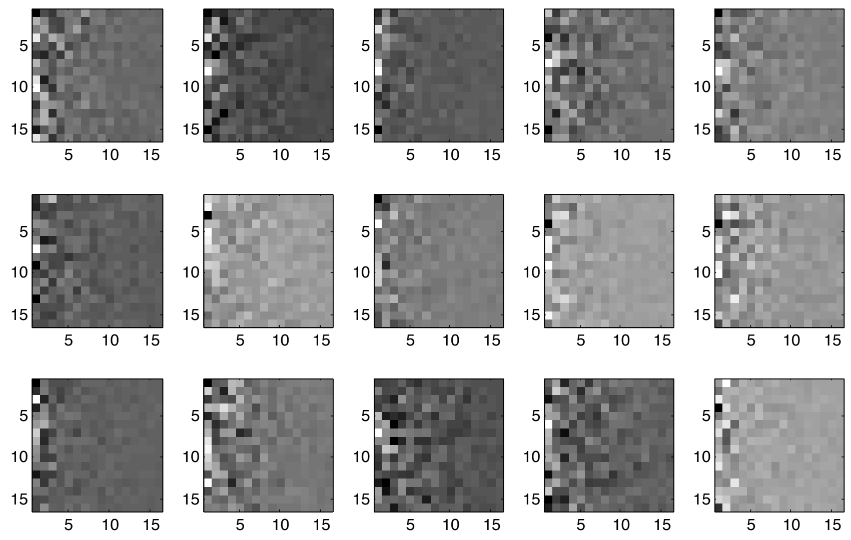

Figure 3: First 15 'eigendigits' of '2' based on 256 principal components

4 Results

Some of the results of the feature extraction (and in this case equivalently, dimensionality reduction) have already been shown in Figure 4, Figure 3 and Figure 2. The results showing the relation between the distance metric used by K-NN algorithm and classification rates are shown below for various distance metrics. Table 1 also gives the run times on the training set. For all the cases the number of nearest neighbors considered was kept constant at $k = 3$.

The results showing the relation between the number of nearest neighbors used and the classification rates on the training data are depicted in Figure 5. The results show on the the cases in which the distance metric used was either 2-norm or 1-norm. The cases for Cosine and Cor-

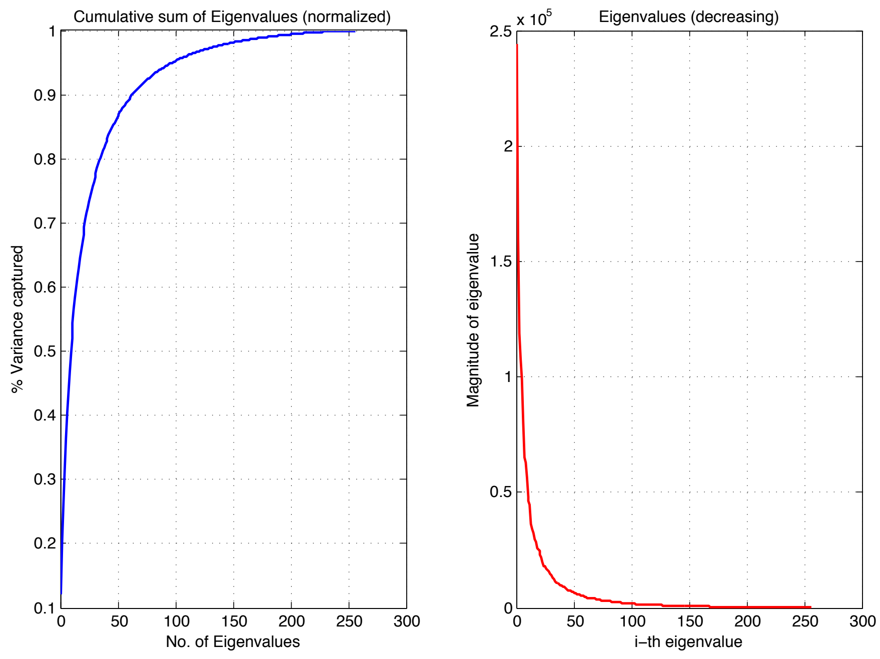

Figure 4: Relation between number of eigenvector(PCs) and variance of $\mathbf{X}$

Distance Metric	Classification Rate($\%$)	Time taken (s)
2-Norm	96.65	9.17
1- Norm	96.15	9.17
Correlation	96.65	1609
Cosine	96.55	1596

Table 1: Classification Rates compared to distance metrics on the Training data

relation distance metrics are not shown due to to prohibitively long run times associated with generating such plots.

To check how appropriate the choice of distance metric was, I partitioned the dataset into two blocks, training block consisting of 60% of the original dataset and a testing block containing the remaining data. I then ran the algorithm on each dataset block and compared the results. This choice of very simplistic cross-validation is justified by the fact that the algorithm used has very few parameters to be tuned and the results from the classification of the training data already show minimal variation between the different distance metrics and choice of number of nearest neighbors to consider. Table 2 gives the comparison of the classification rates for each of the distance metrics on both the training data and test data for a fixed number of nearest neighbors.

Distance Metric	Training Data($\%$)	Testing Data ($\%$)
2-Norm	95.33	89.00
1- Norm	93.58	88.00
Correlation	95.67	90.38
Cosine	95.00	90.25

Table 2: Classification rates on training and test data for $k = 3$

Another interesting result observed from the experiment was that all the distance metric had the highest misclassifications of the digit '8' on average over the whole training set when evaluated

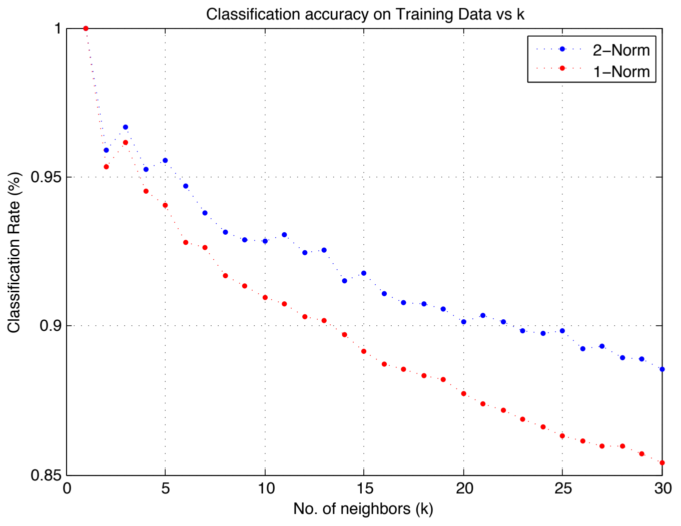

Figure 5: Relation between number of neighbors, k and classification rates

using three nearest neighbors. This result is depicted in Figure 6 below. It is also interesting to note that the digit '1' was correctly classified by all the distance metrics with digits '5', '7' and '0' a;so registering similarly low misclassification rates and the remaining digits averaging to about 9 misclassifications each.

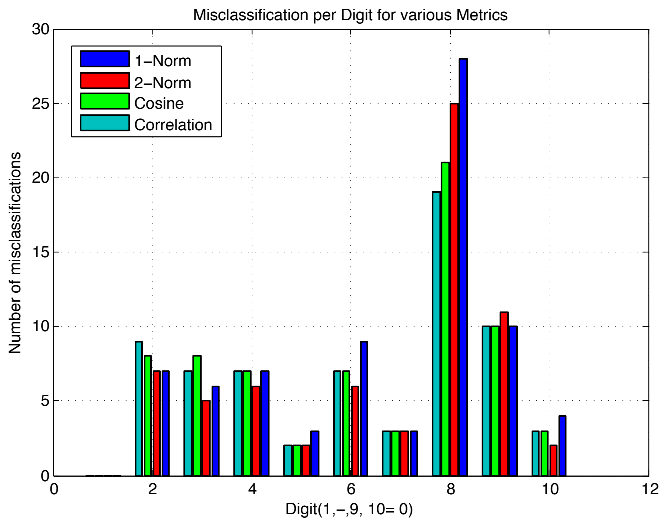

Figure 6: Misclassifications per Digit for the distance metrics

5 Conclusion

The approach detailed in this paper was able to tackle the problem of classifying handwritten digits by decomposing the problem into two main steps, the first being compressing the data (or equivalently extracting features of lower dimensions) by applying Principal Component Analysis which is based on three main assumptions and a rigorous treatment of linear algebra. With the extracted features, I was able to then classify new digits by applying the K-Nearest Neighbor algorithm detailed herein.

A number of key factors were identified to influence the performance of the classification namely, the choice of the number of principal components to retain while making a new basis for re-representing the original dataset, the number of neighbors to consider when applying the K-NN algorithm and lastly the choice of distance metric to use when searching for the nearest neighbors of the test digit.

References

[MAT10] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[Nab02] I. T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer Verlag, London, 2002.

[Shl09] Jonathon Shlens. A tutorial on principal component analysis, 2009.

[TH09] Jerome H. Friedman Trevor Hastie, Robert Tibshirani. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[WS09] K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009.