CSIC 5011: Mini-Project 1

A Short Exploration of Dimension Reduction: Handwritten Digit Recognition and PCA Family

LO Yi-Su (SID: 203999880) * **Cao Guiyu (SID: 20392411) ****

Department of Mathematics
The Hong Kong University of Science and Technology

* yloab@connect.ust.hk ** gcaoaa@connect.ust.hk

Abstract

In this mini-project, we try to complete a well-known classification task regarding the handwritten ZIP Code digits. Firstly, we carry out principal component analysis (PCA), sparse PCA, or kernel PCA as approaches of dimension reduction. Secondly, we apply support vector machine (SVM) to train classifiers for recognizing the projections of handwritten digits in the feature space. Finally, the performance of the classifiers which obtained with distinct dimension reduction techniques and SVM are compared.

1 Motivation

Automatic handwriting recognition is of great academic and commercial interest. Post offices use them to sort letters; bank use them to read personal checks. Currently, many machine learning algorithms are available to recognize and classify handwritten digits, such as the support vector machine, random forest, etc. Many of them require or could be benefited by an effective dimension reduction method. The principal component analysis (PCA) is a widely-used method for this purpose, as well as the sparse PCA and kernel PCA. We are interested in discovering their natures and the way they affect the performance of a classification algorithm.

2 Problem statement

The trial data set and problem of interest in this project is a well-know task to classify the handwritten digits. The data is a set of images for handwritten ZIP Code digits $0, 1, \ldots, 9$, scanned from envelopes. One can download them at

Training set:	`http://www-stat.stanford.edu/~hastie/ElemStatLearn/datasets/zip.digits/`
Test set:	`http://statweb.stanford.edu/~hastie/ElemStatLearn/datasets/zip.test.gz`

Table **??** and Figure **??** below show some information and ingredient of the captioned data set. Note that it contains a training set, for classifier training, and a test set, for evaluating the performance of the classifier. Every sample of the data set consists of 16×16 pixels which take value as the grey scale from 0 (black) to 1 (white). In fact, we consider every sample a data point in the high-dimensional space $[\,0, 1\,]^{256}$.

Digit	0	1	2	3	4	5	6	7	8	9	Total
Training set	1194	1005	731	658	652	556	664	645	542	644	7291
Test set	359	264	198	166	200	160	170	147	166	177	2007

Table 1: Number of samples for every digit in training and test sets.

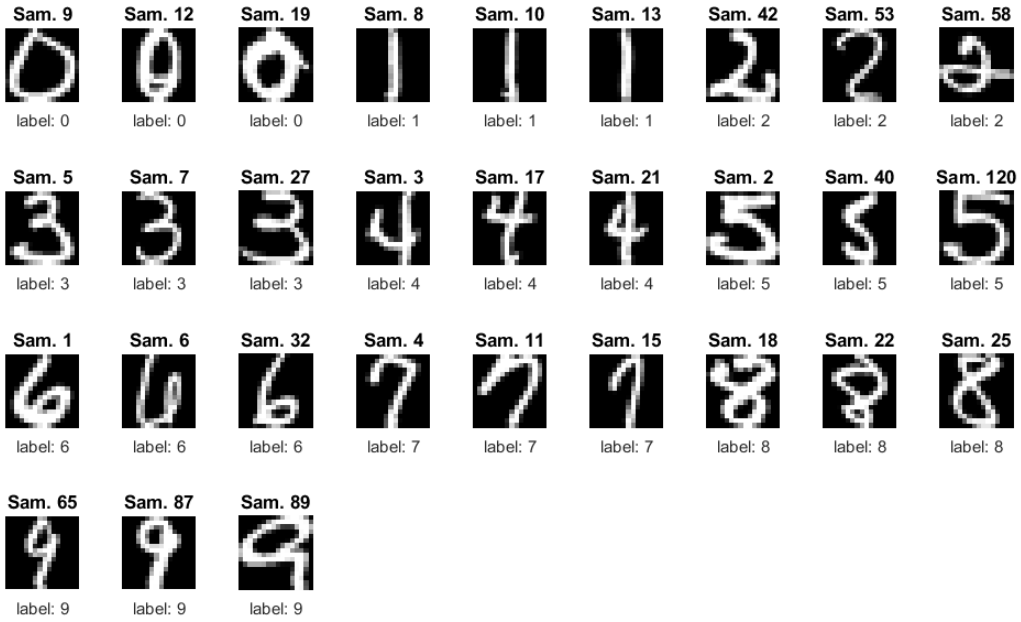


Figure 1: Some example samples.

3 Methods and software

The process of classifier training and classification could be described as

1. Carry out a dimension reduction algorithm which selects a certain number of basis vectors which form a feature space;
2. Train a classifier with the projections of training data in the feature space;
3. Project the test data into the feature space and classify them with the trained classifier.

We have a quick review of the three dimension reduction methods, PCA, sparse PCA and kernel PCA, and the classifier training method, support vector machine, in this section. The latest update of this report (if any) and the `Matlab` source code for experiments in this project are accessible at

`https://goo.gl/nDyy5x`

3.1 Principal component analysis (PCA)

PCA was introduced by Pearson (1901) and Hotelling (1933). It could be helpful for dimension reduction and feature extraction.

For a given data set $\{x_i : i = 1, 2, \ldots, n\}$, it solves

$$\min_{\mu,\beta_i,U} \sum_i \parallel x_i - (\mu + U\beta_i) \parallel,$$

where μ is the sample mean i.e. $\mu = \frac{1}{n}\sum_i x_i$, U consists of the *principle components* and $\sum_i \beta_i = 0$. In the case the number of principal components is assigned as k, it is equivalent to

$$\min_L \parallel X - L \parallel \qquad \text{s.t.} \qquad \text{rank}(L) \leq k,$$

where X is the data matrix collecting all data points x_i.

The PCA in this project are realized with `Matlab` built-in functions.

3.2 Sparse principal component analysis (SPCA)

SPCA is firstly proposed by H. Zhou et al. It tries to locate sparse principle components, which extends the classic PCA by adding sparsity constraint on the input variables.

Classical PCA can be written in the semidefinite programing (SDP) form

$$\begin{aligned} \max\ & \text{trace}(\Sigma Y) \\ s.t.\ & \text{trace}(Y) = 1 \\ & Y \succeq 0 \end{aligned}$$

Now we are looking for sparse principle components, i.e. $\#\{Y_{ij} \neq 0\}$ are small. Using 1-norm convexification, we have the following SDP form for SPCA

$$\begin{aligned} \max\ & \text{trace}(\Sigma Y) - \lambda ||Y||_1 \\ s.t.\ & \text{trace}(Y) = 1 \\ & Y \succeq 0 \end{aligned}$$

The solution would be a rank-1 Y along the first principal component. The same step could be repeated on the remaining part of the data after extraction (i.e. $\Sigma - Y$), and so on.

In this project, we have SPCA realized with Thomas Bühler and Matthias Hein's `Matlab` code available at

`https://github.com/tbuehler/sparsePCA`

3.3 Kernel principal component analysis (KPCA)

KPCA is an extension of PCA using techniques of kernel methods. The use of a kernel, the originally linear operation of PCA, is performed in a reproducing kernel Hilbert space.

First, consider the Gaussian kernel

$$k(x, y) = e^{-\frac{||x-y||^2}{2\sigma^2}}$$

Secondly, construct the normalized kernel matrix of the data

$$\widetilde{K} = K - 2\,1_{1/n}K + 1_{1/n}K1_{1/n}$$

where $1_{1/n}$ is a matrix with all elements $1/n$ Then, solve an eigenvalue problem

$$\widetilde{K}\alpha_i = \lambda_i \alpha_i$$

Finally, any data can represent it as

$$y_j = \sum_{i=1}^{n} \alpha_{ji} K(x, x_i),\ j\ =\ 1, \cdots, d.$$

In this project, we employ the `Matlab` code for KPCA which is written by Quan Wang. The MathWorks page for it is at

`https://www.mathworks.com/matlabcentral/fileexchange/`
`39715-kernel-pca-and-pre-image-reconstruction`

3.4 Support vector machine (SVM)

In a data set $\{(x_i, y_i) : y_i = \pm 1, i = 1, 2, \ldots, n\}$, the *hard-margin* SVM could be regarded as solving the optimization problem

$$\min_{\beta, b}\ \frac{1}{2}\beta^T\beta \qquad \text{subject to} \qquad y_i\, f(x_i) \geq 1,\ \forall\, i,$$

where $f(x_i) = \beta^T \phi(x_i) + b$. Once the optimizer $(\hat{\beta}, \hat{b})$ is determined, the value of $\hat{f}(x) = \hat{\beta}^T \phi(x) + \hat{b}$ presents the distance from any data point x to the hyperplane/decision boundary, as well as the *classification score* of x.

Usually, the data points might not be perfectly separated by any hyperplane. In this case we may adopt the *soft-margin* SVM which solves

$$\min_{\beta, b, \xi_i} \ \frac{1}{2}\beta^T\beta + C\sum_i \xi_i^2$$
$$\text{subject to } \begin{cases} y_i\, f(x_i) \geq 1 - \xi_i, \\ \xi_i \geq 0, \end{cases} \quad \forall\, i,$$

where ξ_i's serve as *slack variables*. C is often called the *box constraint* in which higher value of C allows a higher tolerance to the classification errors. By considering the Lagrange function of the problem

$$\mathcal{L}(\beta, b, \xi_i, \alpha_i, \mu_i) = \frac{1}{2}\beta^T\beta + C\sum_i \xi_i^2 - \sum_i \alpha_i(y_i f(x_i) - (1 - \xi_i)) - \sum_i \mu_i \xi_i$$

and the KKT condition $\nabla\mathcal{L} = 0$, we derive the dual problem

$$\max_{\alpha_i} \ \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j K(x_i, x_j)$$
$$\text{subject to } \begin{cases} y_i\alpha_i = 0, \\ 0 \leq \alpha_i \leq C, \end{cases} \quad \forall\, i,$$

where $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is the *kernel function* and the inequality of α_i explains the reason we call C a box constraint. For K, a popular choice is the Gaussian kernel function, or say the *radial basis function* (RBF) kernel

$$K(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2},$$

where $\gamma = \frac{1}{2\sigma^2}$ is the scaling parameter. Note that γ and C are usually determined through *cross validation* and *grid search* approach.

Since a hyperplane divides the space into two regions, SVM is in fact an approach for binary classification. However, we have totally 10 classes (from 0 to 9) in the data set of interest. In this case, we apply the *one versus the rest* strategy in which we find a hyperplane for every digit and the rest. In the end, 10 hyperplanes are combined to divide the space into 10 regions. For the un-classified regions and multi-classified regions, the *distance rule* which classifies a point by the classification score with respect to every hyperplane is adopted.

The SVM tasks in this project are completed through the `Matlab` built-in functions and several user-defined subroutines.

4 Results and discussion

We have three dimension reduction methods to examine. However, for SPCA, we tried three levels of sparsity requirement: 64, 32, and 16. The nonzero elements of every principal component couldn't exceed these upper bounds. In the following subsections, we present the results of five experiments on:

(a) PCA + SVM,
(b) SPCA with sparsity requirement 64 + SVM,
(c) SPCA with sparsity requirement 32 + SVM,
(d) SPCA with sparsity requirement 16 + SVM,
(e) KPCA + SVM.

4.1 Glimpse into the feature space

We draw the projections of training data points in the feature space with respect to the top two principal components in Figure **??**.

Moreover, it is also interesting to have a closer look at those principal components extracted by PCA and SPCA. We expect the later ones will be more sparse, according to the sparsity requirement. Please see Figure **??**. Note that we skip KPCA since the principal components from it are obtained in the kernel space, rather than the original one.

4.2 Performance

In the experiment, the performance of the trained SVM classifiers were measured by the error rate

$$\text{error rate} = \frac{\text{number of mis-classified samples}}{\text{number of total samples}} \times 100\%.$$

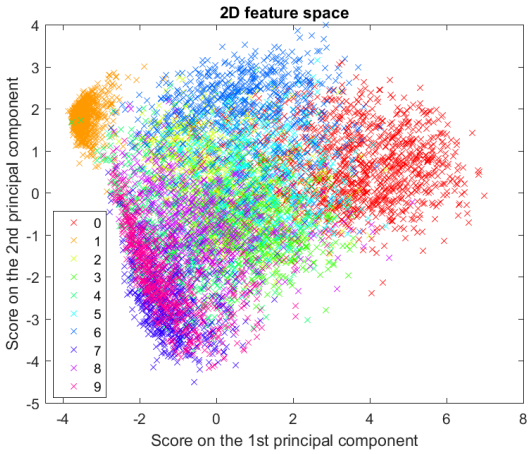
The error rate of the classification of the test set was the main performance index. Besides, we also measured the error rate in training and the elapsed time during the training. The results are listed in Table **??**.

Scheme \ # Prin. comp.	sparsity	8	16	32
PCA	-	0.55 / 3.69% / 11.31%	0.37 / 0.64% / 5.63%	0.50 / 0.40% / 4.88%
SPCA	64	3.08 / 3.51% / 11.96%	7.87 / 0.89% / 6.38%	23.56 / 0.47% / 4.58%
	32	6.27 / 3.33% / 11.16%	10.53 / 1.03% / 6.28%	20.28 / 0.53% / 5.08%
	16	2.19 / 4.51% / 14.95%	6.38 / 0.80% / 6.43%	14.67 / 0.47% / 4.98%
KPCA	-	447.76 / 3.52% / 11.61%	446.80 / 0.85% / 5.93%	447.40 / 0.49% / 4.93%

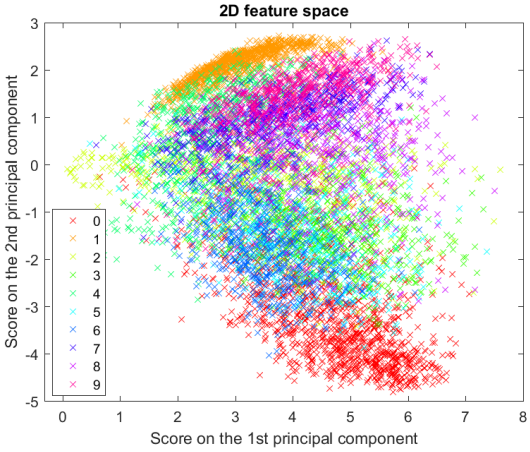
Table 2: Comparison on elapsed time (the first number which measured in sec), training error rate (the percentage in blue color) and test error rate (the percentage in red color).

We draw several conclusions from Table **??**:

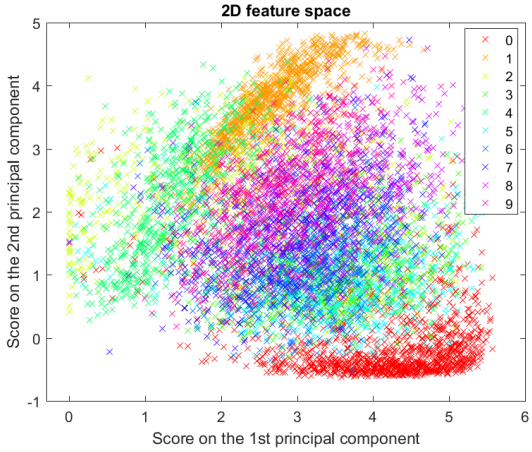
- The time expense of PCA and KPCA tends to be stable, while the elapsed time of SPCA grows as the number of principal components increases. It is not surprising since SPCA is a recursive process.
- PCA spends less time than SPCA and KPCA, but it is probably because PCA is implemented by the `Matlab` built-in function whose efficiency is optimized.
- The classification is more accurate as we use more principal components. However, the effect is decreasing. It indeed indicates that a certain number of dimensions is sufficient for classification.
- The highest accuracy is achieved with SPCA. But overall, all three methods are effective dimension reduction approaches.



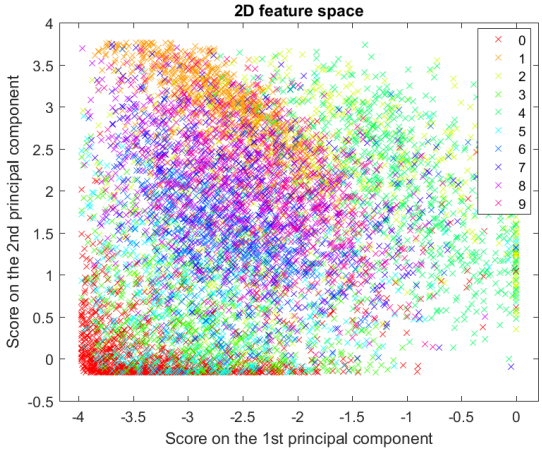
(a) PCA



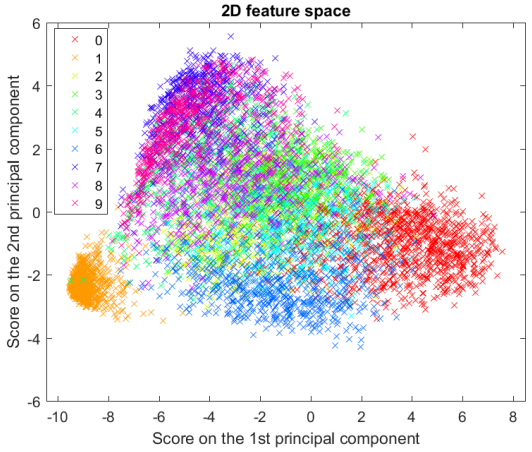
(b) SPCA with sparsity requirement 64



(c) SPCA with sparsity requirement 32

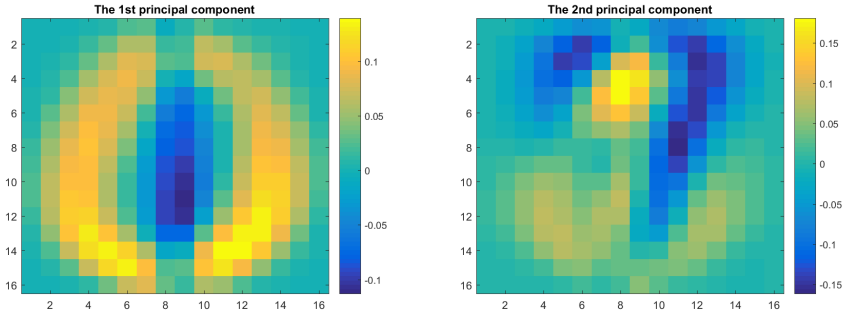


(d) SPCA with sparsity requirement 16

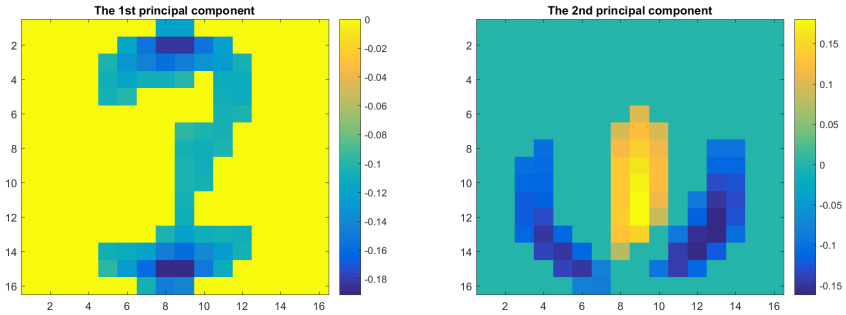


(e) KPCA

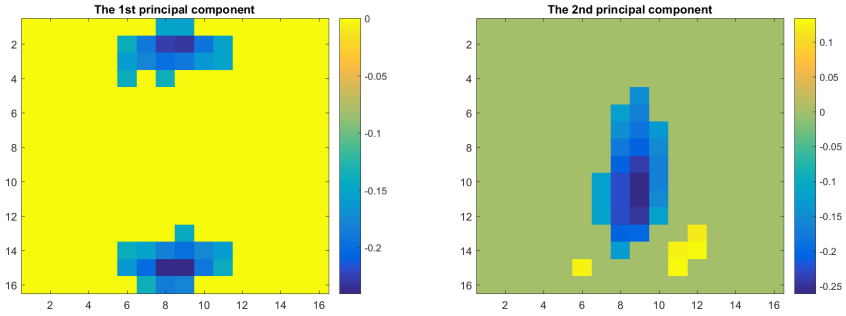
Figure 2: The two-dimensional feature space extracted by PCA, SPCA and KPCA.



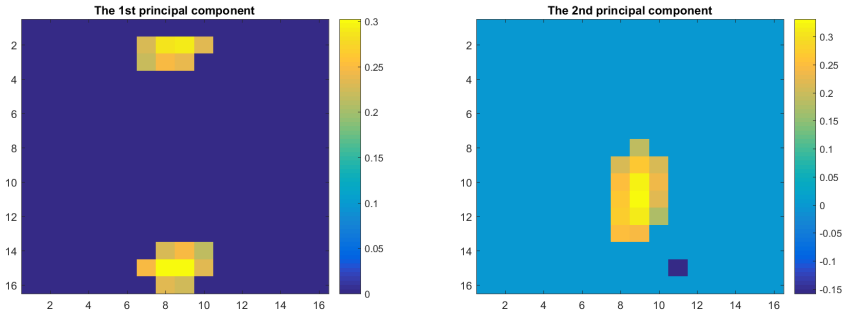
(a) PCA



(b) SPCA with sparsity requirement 64



(c) SPCA with sparsity requirement 32



(d) SPCA with sparsity requirement 16

Figure 3: Top two principal components extracted by PCA and SPCA.

- SPCA and KPCA don't outperform traditional PCA in Table **??**. But we guess they could be more powerful in the other data sets, especially the data sets who have highly nonlinear structure.

4.3 Error samples

We randomly choose some test samples which were wrongly classified under the use of 32 principal components. Perhaps they can help us get an idea why the classification could be challenging. See Figure **??**.

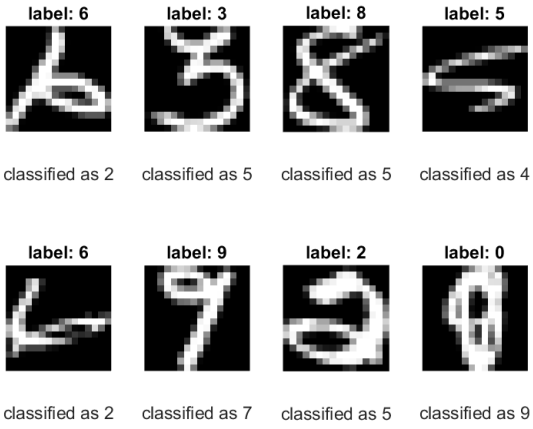


Figure 4: Test samples which were wrongly classified.

5 Conclusion

In this project, we examine three dimension reduction methods: PCA, SPCA and KPCA. We visually inspect the feature spaces and principal components extracted by these unsupervised methods. We also train classifiers and test it. The experiment results show the three methods are effective dimension reduction approaches which help SVM classifies the data in a low-dimensional feature space. However, their efficiency is different due to the algorithmic reason. Finally, we obtain a SVM classifier whose accuracy is over 95%.