3.5 Dimension Reduction

Many machine learning problems involve high dimensional data. In the case of classifying handwritten digits from MNIST (Mixed National Institute of Standards and Technology) database, for example, each digit is an image of size 28 by 28 (Fig. 3.13 shows 54 digits from the database), therefore the data are in 784-dimensional space. On the other hand, images of handwritten digits are known to live in low-dimensional manifolds, therefore low-dimensional representation of high-dimensional data of interest is possible in this case. This section introduces several basic tools that are found useful in materializing such low-dimensional representations. The general machinery of these tools is referred to as *dimension reduction* or *dimensionality reduction*.

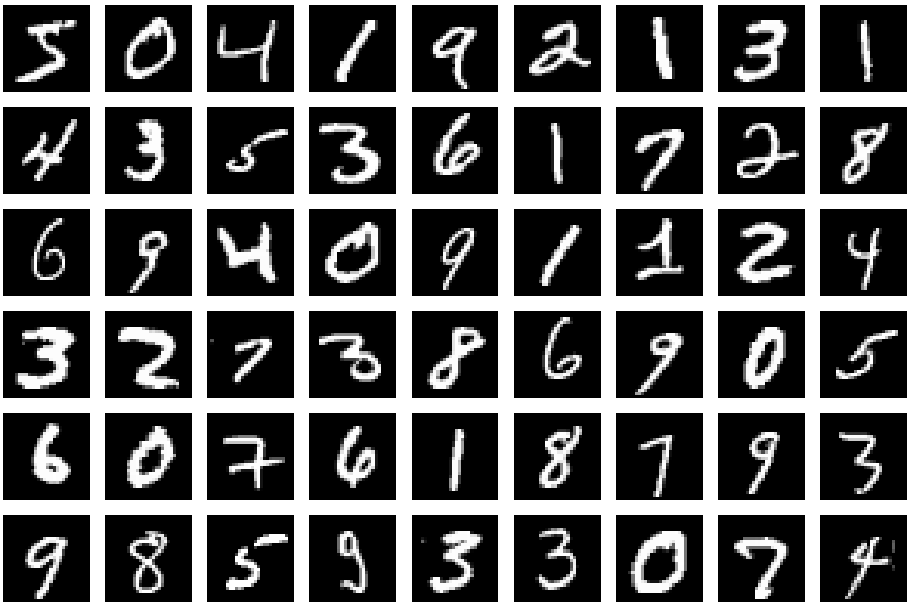

Fig. 3.13 Handwritten digits from MNIST database.

A. Singular Value Decomposition

Let $\boldsymbol{X}$ be a real-valued matrix of size *m* by *n* whose rank is *r*, there exist orthogonal matrices $\boldsymbol{U} \in R^{m\times m}$ and $\boldsymbol{V} \in R^{n\times n}$ such that

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{3.33}$$

where

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{S} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$$

and $\boldsymbol{S} = \text{diag}\{\sigma_1, \sigma_2, \cdots, \sigma_r\}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. The σ_i's are called *singular values* of $\boldsymbol{X}$, and the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are called left and right singular vectors of $\boldsymbol{X}$, respectively, and (3.33) is said to be the *singular value decomposition* (SVD) of $\boldsymbol{X}$. The concepts of singular values and singular vectors are closely related to more popular eigenvalues and eigenvectors. The connection can be seen immediately through the products

$$\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{U}\begin{bmatrix} \boldsymbol{S}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\boldsymbol{U}^T, \quad \boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V}\begin{bmatrix} \boldsymbol{S}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\boldsymbol{V}^T$$

In words, the nonzero singular values of $\boldsymbol{X}$ are the square roots of the nonzero eigenvalues of $\boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{X}^T\boldsymbol{X}$; the left singular vectors are the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$; and the right singular vectors are the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$. The MATLAB function svd implements (3.33) as [U,S,V] = svd(X).

Following (3.33), the SVD of $\boldsymbol{X}$ can be written as

$$\boldsymbol{X} = \boldsymbol{U}_r\boldsymbol{S}\boldsymbol{V}_r^T \tag{3.34}$$

where $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$ are the matrices composed of the first *r* columns of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. Among other things, SVD is useful in finding optimal low-rank approximations of $\boldsymbol{X}$. In effect, given a positive integer *r*' < *r* where *r* is the rank of $\boldsymbol{X}$, an optimal rank- *r*' approximation of $\boldsymbol{X}$, denoted by $\boldsymbol{X}'$, can be constructed via (3.34) as

$$\boldsymbol{X}_{r'} = \boldsymbol{U}_{r'}\boldsymbol{S}_{r'}\boldsymbol{V}_{r'}^T \tag{3.35}$$

where $\boldsymbol{U}_{r'}$ and $\boldsymbol{V}_{r'}$ are the matrices composed of the first *r*' columns of $\boldsymbol{U}_r$ and $\boldsymbol{V}_r$, respectively, and $\boldsymbol{S}_{r'} = \text{diag}\{\sigma_1, \sigma_2, \cdots, \sigma_{r'}\}$. The approximation error introduced by $\boldsymbol{X}_{r'}$ in Frobenius norm is controlled by the 2-norm of the remaining singular values (the residue energy), namely

$$\| \boldsymbol{X} - \boldsymbol{U}_{r'}\boldsymbol{S}\boldsymbol{V}_{r'}^T \|_F = \left(\sum_{i=r'+1}^{r} \sigma_i^2 \right)^{1/2} \tag{3.36}$$

B. <u>Dimension Reduction of Data Matrix X Using SVD</u>

One may think of matrix $\boldsymbol{X}$ in (3.33) as a data array that collects a set of *n* data points $\{\boldsymbol{x}_i, i = 1, 2, \ldots, n\}$ (as column vectors) in *m*-dimensional Euclidean space, that is, $\boldsymbol{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2 \cdots \boldsymbol{x}_n]$ whose rank is *r*. Using (3.34), we define $\boldsymbol{Z}_r = \boldsymbol{S}\boldsymbol{V}_r^T$ which is a matrix of size *r* by *n*, and write it as $\boldsymbol{Z}_r = [\boldsymbol{z}_1\ \boldsymbol{z}_2 \cdots \boldsymbol{z}_n]$ with $\boldsymbol{z}_i \in R^{r\times 1}$. It can be readily verified that the set $\{\boldsymbol{z}_i, i = 1, 2, \ldots, n\}$ preserves the topological structure of the original data set $\{\boldsymbol{x}_i, i = 1, 2, \ldots, n\}$ while reducing the dimension from *m* to *r* in the sense that the distance and "angle" between each pair of the data points as well as the length (energy) of each individual vector (signal) are preserved, namely,

$$\| \boldsymbol{z}_i \| = \| \boldsymbol{x}_i \|, \ \| \boldsymbol{z}_i - \boldsymbol{z}_j \| = \| \boldsymbol{x}_i - \boldsymbol{x}_j \| \text{ and } \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \text{ for } 1 \le i, j \le n \tag{3.37}$$

A simple proof of (3.37) would be based on the fact that the two data sets differ only by $\boldsymbol{U}_r$ as seen from (3.34), whose columns are orthonormal. This SVD-based dimension reduction is particularly attractive when *r* << *m*.

What if *r* << *m* does not hold? In this scenario one examines the *r* nonzero singular values to see if the number of the dominating singular values, denoted by *r'*, is small. If so, based on (3.35) and (3.36) one can claim that

$$\boldsymbol{X} \approx \boldsymbol{U}_{r'}\boldsymbol{S}_{r'}\boldsymbol{V}_{r'}^T \tag{3.38}$$

Consequently, if we define $\boldsymbol{Z}_{r'} = \boldsymbol{S}\boldsymbol{V}_{r'}^T$ and write it as $\boldsymbol{Z}_{r'} = [\boldsymbol{z}_1\ \boldsymbol{z}_2 \cdots \boldsymbol{z}_n]$ with $\boldsymbol{z}_i \in R^{r'\times 1}$, then we

have

$$\| \boldsymbol{z}_i \| \approx \| \boldsymbol{x}_i \|, \| \boldsymbol{z}_i - \boldsymbol{z}_j \| \approx \| \boldsymbol{x}_i - \boldsymbol{x}_j \| \text{ and } \langle \boldsymbol{z}_i, \boldsymbol{z}_j \rangle \approx \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \text{ for } 1 \le i, j \le n \qquad (3.39)$$

demonstrating the validity of using $\boldsymbol{Z}_{r'} = \boldsymbol{S}\boldsymbol{V}_{r'}^T$ as a low-dimensional approximation of the original data $\boldsymbol{X}$.

Perhaps the biggest problem with the SVD-based dimension reduction is that of computing the SVD of $\boldsymbol{X}$, especially for large size $\boldsymbol{X}$. A well-known solution to this problem is the Johnson-Lindenstrauss lemma [16]. Let $\boldsymbol{P} \in R^{p \times m}$ be a projection matrix with $p << m$ that is generated by sampling p row vectors independently and uniformly on the unit sphere in the m-dimensional space which are in turn orthonormalized with each row scaled to have its 2-norm equal to $\sqrt{m/p}$. The dimension reduction of data matrix $\boldsymbol{X}$ is accomplished in two steps: First, we compute $\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{X}$ that projects the m-dimensional vectors $\boldsymbol{x}_i$'s to p-dimensional vectors $\boldsymbol{y}_i$'s where $\boldsymbol{y}_i$ is the ith column of $\boldsymbol{Y}$. Next, the SVD-based dimension reduction is applied to data set $\boldsymbol{Y}$:

$$\boldsymbol{Y} = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^T = \underbrace{\hat{\boldsymbol{U}}_r}_{\substack{\text{first } r \\ \text{columns} \\ \text{of } \hat{\boldsymbol{U}}}} \cdot \hat{\boldsymbol{S}}_r \cdot \underbrace{\hat{\boldsymbol{V}}_r^T}_{\substack{\text{first } r \\ \text{rows} \\ \text{of } \hat{\boldsymbol{V}}^T}}$$

from which we obtain $\hat{\boldsymbol{Z}}_r = \hat{\boldsymbol{S}}_r \cdot \hat{\boldsymbol{V}}_r^T$. It can be shown (Gilbert-Park-Wakin 2012) that if

$$p \ge \frac{r \log(42/\varepsilon) + \log(2/\delta)}{f\left(\varepsilon/\sqrt{2}\right)}$$

where $\varepsilon \in (0, 1), \delta \in (0, 1)$ and $f(\cdot)$ is quadratic, then with probability $1 - \delta$

$$(1-\varepsilon)^{1/2} \le \frac{\hat{\sigma}_j}{\sigma_j} \le (1+\varepsilon)^{1/2}$$

for $j = 1, 2, \ldots, r$, and

$$\left\| \boldsymbol{v}_j - \hat{\boldsymbol{v}}_j \right\|_2 \le \min\left\{ \sqrt{2}, \frac{\varepsilon\sqrt{1+\varepsilon}}{\sqrt{1-\varepsilon}} \max_{i \ne j} \frac{\sqrt{2}\sigma_i\sigma_j}{\min\limits_{c \in [-1, 1]} \{ | \sigma_i^2 - \sigma_j^2(1+c\varepsilon) | \}} \right\}$$

C. Principal Component Analysis (PCA)

Again we consider a real-valued data matrix $\boldsymbol{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \cdots \boldsymbol{x}_n]$ of size m by n. Define the *average vector* and *covariance matrix* of the n component vectors $\boldsymbol{x}_i$'s as

$$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i, \quad \boldsymbol{C} = \frac{1}{n-1}\sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \qquad (3.40)$$

respectively. Note that $\boldsymbol{C}$ is at least positive semidefinite if not positive definite, hence its SVD is identical to its eigen-decomposition

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{U}^T \tag{3.41}$$

where $\boldsymbol{U} = [\boldsymbol{u}_1\ \boldsymbol{u}_2 \cdots \boldsymbol{u}_m]$ is an orthogonal matrix and $\boldsymbol{S} = \text{diag}\{\sigma_1, \sigma_2, \cdots, \sigma_m\}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$, and $\boldsymbol{u}_i$ is $\boldsymbol{C}$'s *i*th eigenvector associated with *i*th eigenvalue σ_i. By taking into account only the *K* largest eigenvalues and the associated eigenvectors, we obtain a useful approximation of variance matrix $\boldsymbol{C}$ as

$$\boldsymbol{C} \approx \boldsymbol{U}_K \boldsymbol{S}_K \boldsymbol{U}_K^T \ \text{ with } \ \boldsymbol{U}_K = [\boldsymbol{u}_1\ \boldsymbol{u}_2 \cdots \boldsymbol{u}_K] \text{ and } \boldsymbol{S}_K = \text{diag}\{\sigma_1, \sigma_2, \cdots, \sigma_K\} \tag{3.42}$$

The usefulness of (3.42) may be understood from two different but related perspectives as follows.

(1) Since $\boldsymbol{C}$ is the variance for $\{\boldsymbol{x}_i - \bar{\boldsymbol{x}}, i = 1, 2, ..., n\}$, the *K* eigenvectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_K\}$ capture the most significant variations between the individual data points and their average, each point $\boldsymbol{x}_i - \bar{\boldsymbol{x}}$ is well represented by its projection onto the *K*-dimensional subspace spanned by $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_K\}$, i.e.,

$$\boldsymbol{z}_i = \boldsymbol{U}_K^T (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \tag{3.43a}$$

In effect, point $\boldsymbol{x}_i$ is approximated by a point in that subspace (plus the average $\bar{\boldsymbol{x}}$) as

$$\boldsymbol{x}_i \approx \boldsymbol{U}_K \boldsymbol{z}_i + \bar{\boldsymbol{x}} \tag{3.43b}$$

In this way, we see how the original data set $\boldsymbol{X}$ in R^m is reduced to an approximately equivalent data set $\boldsymbol{Z} = [\boldsymbol{z}_1\ \boldsymbol{z}_2 \cdots \boldsymbol{z}_n]$ in space R^K (plus a single average vector $\bar{\boldsymbol{x}}$). This PCA-based dimensionality reduction becomes significant when $K << m$.

(2) Now let us consider a case in supervised multi-class classification with a training set that consists of *L* data classes of the form $\{(\boldsymbol{x}_i^{(j)}, y_j), i = 1, 2, ..., n_j\}$ for *j* = 1, 2, …, *L*. The problem is to classify a point $\boldsymbol{x}$ outside the training data to one of the *L* classes. To this end, PCA is applied to each data class $\{(\boldsymbol{x}_i^{(j)}, y_j), i = 1, 2, ..., n_j\}$ by first computing the average vector and covariance matrix as

$$\bar{\boldsymbol{x}}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} \boldsymbol{x}_i^{(j)}, \quad \boldsymbol{C}_j = \frac{1}{n_j - 1} \sum_{i=1}^{n_j} (\boldsymbol{x}_i^{(j)} - \bar{\boldsymbol{x}}_j)(\boldsymbol{x}_i^{(j)} - \bar{\boldsymbol{x}}_j)^T$$

then performing eigen-decomposition of $\boldsymbol{C}_j$

$$\boldsymbol{C}_j = \boldsymbol{U}_j \boldsymbol{S}_j \boldsymbol{U}_j^T$$

so as to approximate it as

$$\boldsymbol{C}_j \approx \boldsymbol{U}_K^{(j)} \boldsymbol{S}_K^{(j)} \boldsymbol{U}_K^{(j)T}$$

where $\boldsymbol{U}_K^{(j)} = [\boldsymbol{u}_1^{(j)}\ \boldsymbol{u}_2^{(j)} \cdots \boldsymbol{u}_K^{(j)}]$ consists of K eigenvectors of $\boldsymbol{C}_j$ that are associated K largest eigenvalues and $\boldsymbol{S}_K^{(j)}$ is a diagonal matrix that collects the K largest eigenvalues in descent order. The classification of a testing point $\boldsymbol{x}$ is carried out as follows: (i) Using each data class, compute projection $\boldsymbol{z}_j = \boldsymbol{U}_K^{(j)T}(\boldsymbol{x} - \overline{\boldsymbol{x}}_j)$ and approximate point $\boldsymbol{x}$ in the jth class as $\hat{\boldsymbol{x}}_j = \boldsymbol{U}_K^{(j)}\boldsymbol{z}_j + \overline{\boldsymbol{x}}_j$; (ii) Evaluate the difference between $\boldsymbol{x}$ and its approximation $\hat{\boldsymbol{x}}_j$ in 2-norm, i.e. , $e_j = \| \boldsymbol{x} - \hat{\boldsymbol{x}}_j \|_2$ for j = 1, 2, …, L. Point $\boldsymbol{x}$ is classified to class j^* if e_{j^*} reaches the minimum among $\{e_j, j = 1, 2, \ldots, L\}$.

D. *<u>An Example: Classification of Handwritten Digits from MNIST Data Base Using PCA</u>*

The MNIST database provides 60,000 handwritten digits for training purposes and a separate 10,000 digits testing, where each digit is a black-and-white image of size 28 by 28. In this example the training images were partitioned into ten classes, with the images in the jth class representing digit j, for j = 0, 1, …, 9. We select from each class 500 images at random, then convert each image into a column vector of dimension 784 to construct a data a data matrix $\boldsymbol{X}_j$ of size 784 by 500. For convenience, these ten data matrices are combined to a signal data matrix X500 of size 784 by 5000 as a .mat file which you can download from the course website. The PCA-based classification method as described above was applied to classify the 10K testing digits. With K = 24 (see part C for the meaning of K), an error rate of 4.4% was achieved.
The MATLAB file that implements the simulation is enclosed below.

```
% To implement the example in Sec. 3.5.D for classifying handwritten digits
% from MNIST database.
% Input:
% X: Ten classes of input data, each is of size 784 by nt (here we use nt = 500).
% n: the actual size of each data matrix is 784 by n. Hence n is limited to n <= nt.
% K: the number of "eigen-digits" to be used in PCA. Typically K << m = 784.
% Te: testing data (here we use Te28 of size 784 by 10000).
% Lte: Labels for the testing data.
% Output:
% Js: the numerical values of testing digits obtained by PCA-based classification.
% er: classification error.
% Written by W.-S. Lu, University of Victoria. Last modified: March 9, 2015.
% Example: load X500; load Te28; load Lte28;
% [Js,er] = class_pca_digits(X500,500,24,Te28,Lte28);
function [Js,er] = class_pca_digits(X,n,K,Te,Lte)
U = [];
Xb = [];
t = size(X,2);
nt = t/10;
for i = 1:10,
   Tw = X(:,((i-1)*nt+1):(i*nt));
   Ti = Tw(:,1:n);
   xbi = mean(Ti')';
   Xh = Ti - xbi*ones(1,n);
   [u,s,v] = svd(Xh*Xh');
```

```
   U = [U u(:,1:K)];
   Xb = [Xb xbi];
end
Lte = Lte(:);
M = length(Lte);
Js = zeros(M,1);
for m = 1:M,
   xw = Te(:,m);
   e = zeros(1,10);
   for j = 1:10;
      Uj = U(:,((j-1)*K+1):(j*K));
      xbj = Xb(:,j);
      zj = Uj'*(xw-xbj);
      xjh = Uj*zj + xbj;
      e(j) = norm(xw-xjh);
   end
   [emin,ind] = min(e);
   Js(m) = ind - 1;
end
E = (Lte ~= Js);
er = sum(E)/M
```

Problems

3.14 Prove (3.37). (Hint: use Eq. (3.34))

3.15 The Frobenius norm of a real-valued matrix $\boldsymbol{A} = \{ a_{i,j} \}$ is equal to the square root of the sum of all $a_{i,j}^2$.

(i) Prove that $\| \boldsymbol{A} \|_F^2 = \text{trace}(\boldsymbol{A}\boldsymbol{A}^T)$.

(ii) Use the result from part (i) to show Eq. (3.36).

3.16 The computational complexity of performing the eigen-decomposition of C in (3.41) becomes intractably high when *m* is very large. In the case of n << m, the computational burden may be greatly reduced and this exercise problem introduces such a technique.

(i) Define $\hat{\boldsymbol{X}} = \frac{1}{\sqrt{n-1}} \left[\boldsymbol{x}_1 - \overline{\boldsymbol{x}} \quad \boldsymbol{x}_2 - \overline{\boldsymbol{x}} \quad \cdots \quad \boldsymbol{x}_n - \overline{\boldsymbol{x}} \right]$. Verify that $\boldsymbol{C} = \hat{\boldsymbol{X}}\, \hat{\boldsymbol{X}}^T$.

(ii) Show that the nonzero eigenvalues of $\boldsymbol{C}$ are the same as the nonzero eigenvalues of $\Phi = \hat{\boldsymbol{X}}^T \hat{\boldsymbol{X}}$.

(iii) Let $\boldsymbol{v}$ be an eigenvector of Φ, show that $\hat{\boldsymbol{X}}\, \boldsymbol{v}$ is an eigenvector of $\boldsymbol{C}$.

(iv) Based on the results from parts (i) to (iii), develop an efficient method to compute $\boldsymbol{U}$ and $\boldsymbol{S}$ in (3.41) without computing the eigenvalues and eigenvectors of $\boldsymbol{C}$.