Preface

Nowadays, huge amount of multimedia data are being constantly generated in various forms from various places around the world. With ever increasing complexity and variability of multimedia data, traditional rule-based approaches where humans have to discover the domain knowledge and encode it into a set of programming rules are too costly and incompetent for analyzing the contents, and gaining the intelligence of this glut of multimedia data.

The challenges in data complexity and variability have led to revolutions in machine learning techniques. In the past decade, we have seen many new developments in machine learning theories and algorithms, such as boosting, regressions, Support Vector Machines, graphical models, etc. These developments have achieved great successes in a variety of applications in terms of the improvement of data classification accuracies, and the modeling of complex, structured data sets. Such notable successes in a wide range of areas have aroused people's enthusiasms in machine learning, and have led to a spate of new machine learning text books. Noteworthily, among the ever growing list of machine learning books, many of them attempt to encompass most parts of the entire spectrum of machine learning techniques, resulting in a shallow, incomplete coverage of many important topics, whereas many others choose to dig deeply into a specific branch of machine learning in all aspects, resulting in excessive theoretical analysis and mathematical rigor at the expense of loosing the overall picture and the usability of the books. Furthermore, despite a large number of machine learning books, there is yet a text book dedicated to the audience of the multimedia community to address unique problems and interesting applications of machine learning techniques in this area.

The objectives we set for this book are two-fold: (1) bring together those important machine learning techniques that are particularly powerful and effective for modeling multimedia data; and (2) showcase their applications to common tasks of multimedia content analysis. Multimedia data, such as digital images, audio streams, motion video programs, etc, exhibit much richer structures than simple, isolated data items. For example, a digital image is composed of a number of pixels that collectively convey certain visual content to viewers. A TV video program consists of both audio and image streams that complementally unfold the underlying story and information. To recognize the visual content of a digital image, or to understand the underlying story of a video program, we may need to label sets of pixels or groups of image and audio frames jointly because the label of each element is strongly correlated with the labels of the neighboring elements. In machine learning field, there are certain techniques that are able to explicitly exploit the spatial, temporal structures, and to model the correlations among different elements of the target problems. In this book, we strive to provide a systematic coverage on this class of machine learning techniques in an intuitive fashion, and demonstrate their applications through various case studies.

There are different ways to categorize machine learning techniques. Chapter 1 presents an overview of machine learning methods through four different categorizations: (1) Unsupervised versus supervised; (2) Generative versus discriminative; (3) Models for i.i.d. data versus models for structured data; and (4) Model-based versus modeless. Each of the above four categorizations represents a specific branch of machine learning methodologies that stem from different assumptions/philosophies and aim at different problems. These categorizations are not mutually exclusive, and many machine learning techniques can be labeled with multiple categories simultaneously. In describing these categorizations, we strive to incorporate some of the latest developments in machine learning philosophies and paradigms.

The main body of this book is composed of three parts: I. unsupervised learning, II. Generative models, and III. Discriminative models. In Part I, we present two important branches of unsupervised learning techniques: dimension reduction and data clustering, which are generic enabling tools for many multimedia content analysis tasks. Dimension reduction techniques are commonly used for exploratory data analysis, visualization, pattern recognition, etc. Such techniques are particularly useful for multimedia content analysis because multimedia data are usually represented by feature vectors of extremely

high dimensions. The curse of dimensionality usually results in deteriorated performances for content analysis and classification tasks. Dimension reduction techniques are able to transform the high dimensional raw feature space into a new space with much lower dimensions where noise and irrelevant information are diminished. In Chapter 2, we describe three representative techniques: Singular Value Decomposition (SVD), Independent Component Analysis (ICA), and Dimension Reduction by Locally Linear Embedding (LLE). We also apply the three techniques to a subset of handwritten digits, and reveal their characteristics by comparing the subspaces generated by these techniques.

Data clustering can be considered as unsupervised data classification that is able to partition a given data set into a predefined number of clusters based on the intrinsic distribution of the data set. There exist a variety of data clustering techniques in the literature. In Chapter 3, instead of providing a comprehensive coverage on all kinds of data clustering methods, we focus on two state-of-the-art methodologies in this field: spectral clustering, and clustering based on non-negative matrix factorization (NMF). Spectral clustering evolves from the spectral graph partitioning theory that aims to find the best cuts of the graph that optimize certain predefined objective functions. The solution is usually obtained by computing the eigenvectors of a graph affinity matrix defined on the given problem, which possess many interesting and preferable algebraic properties. On the other hand, NMF-based data clustering strives to generate semantically meaningful data partitions by exploring the desirable properties of the non-negative matrix factorization. Theoretically speaking, because the non-negative matrix factorization does not require the derived factor-space to be orthogonal, it is more likely to generate the set of factor vectors that capture the main distributions of the given data set.

In the first half of Chapter 3, we provide a systematic coverage on four representative spectral clustering techniques from the aspects of problem formulation, objective functions, and solution computations. We also reveal the characteristics of these spectral clustering techniques through analytical examinations of their objective functions. In the second half of Chapter 3, we describe two NMF-based data clustering techniques, which stem from our original works in recent years. At the end of this chapter, we provide a case study where the spectral and NMF clustering techniques are applied to the text clustering task, and their performance comparisons are conducted through experimental evaluations.

In Part II and III, we focus on various graphical models that are aimed to explicitly model the spatial, temporal structures of the given data set, and therefore are particularly effective for modeling multimedia data. Graphical models can be further categorized as either generative or discriminative. In Part II, we provide a comprehensive coverage on generative graphical models. We start by introducing basic concepts, frameworks, and terminologies of graphical models in Chapter 4, followed by in-depth coverages of the most basic graphical models: Markov Chains and Markov Random Fields in Chapter 5 and 6, respectively. In these two chapters, we also describe two important applications of Markov Chains and Markov Random Fields, namely Markov Chain Monte Carlo Simulation (MCMC) and Gibbs Sampling. MCMC and Gibbs Sampling are the two powerful data sampling techniques that enable us to conduct inferences for complex problems for which one can not obtain closed-form descriptions of their probability distributions. In Chapter 7, we present the Hidden Markov Model (HMM), one of the most commonly used graphical models in speech and video content analysis, with detailed descriptions of the forward-backward and the Viterbi algorithms for training and finding solutions of the HMM. In Chapter 8, we introduce more general graphical models and the popular algorithms such as sum-production, max-product, etc. that can effectively carry out inference and training on graphical models.

In recent years, there have been research works that strive to overcome the drawbacks of generative graphical models by extending the models into discriminative ones. In Part III, we begin with the introduction of the Conditional Random Field (CRF) in Chapter 9, a pioneer work in this field. In the last chapter of this book, we present an innovative work, Max-Margin Markov Networks (M^3-nets), which strives to combine the advantages of both the graphical models and the Support Vector Machines (SVMs). SVMs are known for their abilities to use high-dimensional feature spaces, and for their strong theoretical generalization guarantees, while graphical models have the advantages of effectively exploiting problem structures and modeling correlations among inter-dependent variables. By implanting the kernels, and introducing a margin-based objective function, which are the core ingredients of SVMs, M^3-nets successfully inherit the advantages of the two frameworks. In Chapter 10, we first describe the concepts and algorithms of SVMs and Kernel methods, and then provide an in-depth coverage of the M^3-nets. At the end of the chapter, we also provide our insights into why discriminative

graphical models generally outperform generative models, and M^3-nets are generally better than discriminative models.

This book is devoted to students and researchers who want to apply machine learning techniques to multimedia content analysis. We assume that the reader has basic knowledge in statistics, linear algebra, and calculus. We do not attempt to write a comprehensive catalog covering the entire spectrum of machine learning techniques, but rather to focus on the learning methods that are powerful and effective for modeling multimedia data. We strive to write this book in an intuitive fashion, emphasizing concepts and algorithms rather than mathematical completeness. We also provide comments and discussions on characteristics of various methods described in this book to help the reader to get insights and essences of the methods. To further increase the usability of this book, we include case studies in many chapters to demonstrate example applications of respective techniques to real multimedia problems, and to illustrate factors to be considered in real implementations.

California, U.S.A. *Yihong Gong*
May 2007 *Wei Xu*

2

Dimension Reduction

Dimension reduction is an important research topic in the area of unsupervised learning. Dimension reduction techniques aim to find a low-dimensional subspace that best represents a given set of data points. These techniques have a broad range of applications including data compression, visualization, exploratory data analysis, pattern recognition, etc.

In this chapter, we present three representative dimension reduction techniques: *Singular Value Decomposition* (SVD), *Independent Component Analysis* (ICA), and *Local Linear Embedding* (LLE). Dimension reduction based on singular value decomposition is also referred to as principal component analysis (PCA) by many papers in the literature. We start the chapter by discussing the goals and objectives of dimension reduction techniques, followed by detailed descriptions of SVD, ICA, and LLE. In the last section of the chapter, we provide a case study where the three techniques are applied to the same data set and the subspaces generated by these techniques are compared to reveal their characteristics.

2.1 Objectives

The ultimate goal of statistical machine learning is to create a model that is able to explain a given phenomenon, or to model the behavior of a given system. An observation $\mathbf{x} \in \mathbb{R}^p$ obtained from the phenomenon/system can be considered as a set of indirect measurements of an underlying source $\mathbf{s} \in \mathbb{R}^q$. Since we generally have no ideas on what measurements will be useful for modeling the given phenomemon/system, we usually attempt to measure all we can get from the target, resulting in a q that is often larger than p.

Since an observation $\mathbf{x}$ is a set of indirect measurements of a latent source $\mathbf{s}$, its elements may be distorted by noises, and may contain strong correlations or redundancies. Using $\mathbf{x}$ in analysis will not only result in poor performance accuracies, but also incur excessive modeling costs for estimating an excessive number of model parameters, some of which are redundant.

The primary goal of dimension reduction is to find a low-dimensional subspace $\mathbb{R}^{p'} \in \mathbb{R}^p$ that is optimal for representing the given data set with respect to a certain criterion function. The use of different criterion functions leads to different types of dimension reduction techniques.

Besides the above primary goal, one is often interested in inferencing the latent source $\mathbf{s}$ itself from the set of observations $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$. Consider a meeting room with two microphones and two simultaneous talking people. The two microphones pick up two different mixtures x_1, x_2 of the two independent sources s_1, s_2. It will be very useful if we can estimate the two original speech signals s_1 and s_2 using the recorded (observed) signals x_1 and x_2. This is an example of the classical *cocktail party problem*, and independent component analysis is intended to provide solutions to blind source separations.

2.2 Singular Value Decomposition

Assume that $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ are a set of centered data points[1], and that we want to find a k-dimensional subspace to represent these data points with the least loss of information. Standard PCA strives to find a $p \times k$ linear projection matrix $\mathbf{V}_k$ so that the sum of squared distances from the data points $\mathbf{x}_i$ to their projections is minimized:

$$L(\mathbf{V}_k) = \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{V}_k \mathbf{V}_k^T \mathbf{x}_i||^2 . \tag{2.1}$$

In (2.1), $\mathbf{V}_k^T \mathbf{x}_i$ is the projection of $\mathbf{x}_i$ onto the k-dimensional subspace spanned by the column vectors of $\mathbf{V}_k$, and $\mathbf{V}_k \mathbf{V}_k^T \mathbf{x}_i$ is the representation of the projected vector $\mathbf{V}_k^T \mathbf{x}_i$ in the original p-dimensional space. It can be easily verified that (2.1) can be rewritten as (see Problem 2.2 at the end of the chapter):

$$\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{V}_k \mathbf{V}_k^T \mathbf{x}_i||^2 = \sum_{i=1}^{n} ||\mathbf{x}_i||^2 - \sum_{i=1}^{n} ||\mathbf{V}_k \mathbf{V}_k^T \mathbf{x}_i||^2 . \tag{2.2}$$

[1] A centered vector $\mathbf{x}_c$ is generated by subtracting the mean vector $\mathbf{m}$ from the original vector $\mathbf{x}$: $\mathbf{x}_c = \mathbf{x} - \mathbf{m}$, so that $\mathbf{x}_c$ is a zero-mean vector.

This means that minimizing $L(\mathbf{V}_k)$ is equivalent to maximizing the term $\sum_{i=1}^{n} ||\mathbf{V}_k \mathbf{V}_k^T \mathbf{x}_i||^2$, which is the empirical variance of these projections. Therefore, the projection matrix $\mathbf{V}_k$ that minimizes $L(\mathbf{V}_k)$ is the one that maximizes the variance in the projected space.

The solution $\mathbf{V}_k$ can be computed by Singular Value Decomposition (SVD). Denote by $\mathbf{X}$ the $n \times p$ matrix where the i'th row corresponds to the observation $\mathbf{x}_i$. The singular value decomposition of the matrix $\mathbf{X}$ is defined as:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T , \tag{2.3}$$

where $\mathbf{U}$ is an $n \times p$ orthogonal matrix ($\mathbf{U}^T\mathbf{U} = \mathbf{I}$) whose column vectors $\mathbf{u}_i$ are called the *left singular vectors*, $\mathbf{V}$ is a $p \times p$ orthogonal matrix ($\mathbf{V}^T\mathbf{V} = \mathbf{I}$) whose column vectors $\mathbf{v}_j$ are called the *right singular vectors*, and $\mathbf{D}$ is a $p \times p$ diagonal matrix with the *singular values* $d_1 \geq d_2 \cdots d_p \geq 0$ as its diagonal elements.

For a given number k, the matrix $\mathbf{V}_k$ that is composed of the first k columns of $\mathbf{V}$ constitutes the *rank* k solution to (2.1). This result stems from the following famous theorem [11].

Theorem 2.1. *Let the SVD of matrix* $\mathbf{X}$ *be given by (2.3),* $\mathbf{U} = [\mathbf{u}_1\, \mathbf{u}_2 \cdots \mathbf{u}_p]$, $\mathbf{D} = \mathrm{diag}(d_1, d_2, \ldots, d_p)$, $\mathbf{V} = [\mathbf{v}_1\, \mathbf{v}_2 \cdots \mathbf{v}_p\,]$, *and* $\mathrm{rank}(\mathbf{X}) = r$. *Matrix* $\mathbf{X}_\tau$ *defined below is the closest rank-*τ *matrix to* $\mathbf{X}$ *in terms of the Euclidean and Frobenius norms.*

$$\mathbf{X}_\tau = \sum_{i=1}^{\tau} \mathbf{u}_i d_i \mathbf{v}_i^T . \tag{2.4}$$

The use of τ-largest singular values to approximate the original matrix with (2.4) has more implications than just dimension reduction. Discarding small singular values is equivalent to discarding linearly semi-dependent or practically nonessential axes of the original feature space. Axes with small singular values usually represent either non-essential features or noise within the data set. The truncated SVD, in one sense, captures the most salient underlying structure, yet at the same time removes the noise or trivial variations in the data set. Minor differences between data points will be ignored, and data points with similar features will be mapped near to each other in the τ-dimensional partial singular vector space. Similarity comparison between data points in this partial singular vector space will certainly yield better results than in the raw feature space.

The singular value decomposition in (2.3) has the following interpretations:

- Column j of the matrix $\mathbf{UD}$ (n-dimensional) corresponds to the projected values of the n data points $\mathbf{x}_i$ onto the j'th right singular vector $\mathbf{v}_j$. This is because $\mathbf{XV} = \mathbf{UD}$, $\mathbf{Xv}_j$ is the projection of $\mathbf{X}$ onto $\mathbf{v}_j$, which equals the j'th column of $\mathbf{UD}$.
- Similarly, row j of the matrix $\mathbf{DV}^T$ (p-dimensional) corresponds to the projected values of the p column vectors of $\mathbf{X}$ onto the j'th left singular vector $\mathbf{u}_j$. This is because $\mathbf{U}^T\mathbf{X} = \mathbf{DV}^T$, $\mathbf{u}_j^T\mathbf{X}$ is the projection of $\mathbf{X}$ onto $\mathbf{u}_j$, which equals the j'th row of $\mathbf{DV}^T$.
- The left singular vectors $\mathbf{u}_j$ and the diagonal elements of the matrix $\mathbf{D}^2$ are the eigenvectors and eigenvalues of the kernel matrix $\mathbf{XX}^T$ [2]. This is because

$$\mathbf{XX}^T = \mathbf{UDV}^T\mathbf{VDU}^T = \mathbf{UD}^2\mathbf{U}^T \quad \Rightarrow \quad \mathbf{XX}^T\mathbf{U} = \mathbf{UD}^2 \,.$$

- Similarly, the right singular vectors $\mathbf{v}_j$ and the diagonal elements of the matrix $\mathbf{D}^2$ are the eigenvectors and eigenvalues of the covariance matrix $\mathbf{X}^T\mathbf{X}$ of the n data points. This is because

$$\mathbf{X}^T\mathbf{X} = \mathbf{VDU}^T\mathbf{UDV}^T = \mathbf{VD}^2\mathbf{V}^T \quad \Rightarrow \quad \mathbf{X}^T\mathbf{XV} = \mathbf{VD}^2 \,.$$

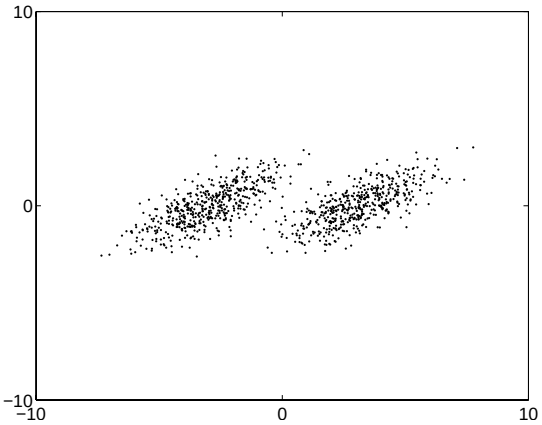
It can be verified that for each column $\mathbf{v}_i$ of $\mathbf{V}$, the following equality holds (see Problem 2.3 at the end of the chapter):

$$\mathrm{var}(\mathbf{Xv}_i) = d_i^2 \,, \tag{2.5}$$

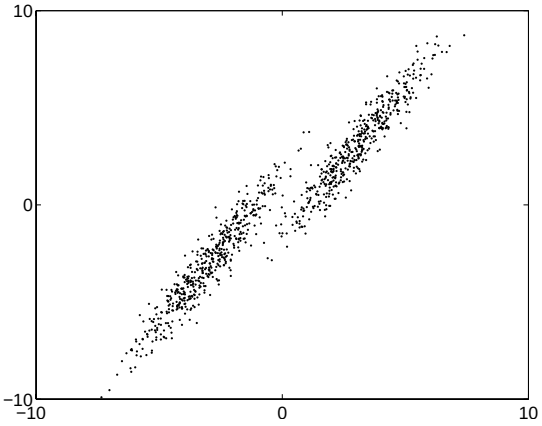
where d_i is the i'th eigenvalue. This means that the columns $\mathbf{v}_1$, $\mathbf{v}_2$, $\cdots$ of $\mathbf{V}$ correspond to the directions with the largest, second largest, $\cdots$ sample variances, which confirms that the matrix $\mathbf{V}_k$ that is composed of the first k columns of $\mathbf{V}$ does constitute the *rank* k solution to (2.1).

We use a synthetic data set to demonstrate the effect of singular value decomposition. Figure 2.1 shows two parallel Gaussian distributions in a 3-D space. These two Gaussian distributions have similar shapes, with the mass stretching mainly along one direction. Figure 2.2 shows the subspace spanned by the first two principal components found by the singular value decomposition. The horizontal and the vertical axes correspond to the first and second principal components, respectively, which are the axes with the largest, and second largest variances.

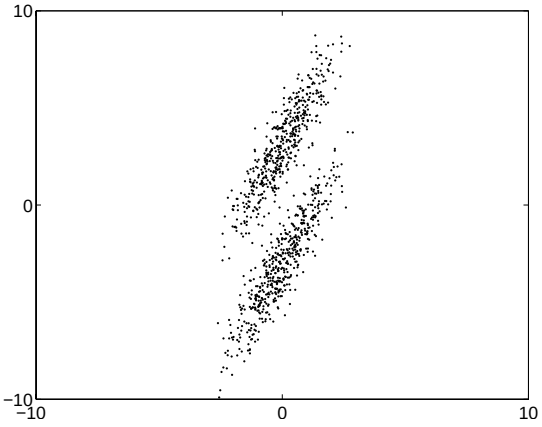
[2] We call $\mathbf{XX}^T$ a kernel matrix because its (i, j)'th element is dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ of the data points $\mathbf{x}_i$ and $\mathbf{x}_j$.



(a) The x-y subspace



(b) The x-z subspace



(c) The y-z subspace

Fig. 2.1. A synthetic data set in a 3-D space

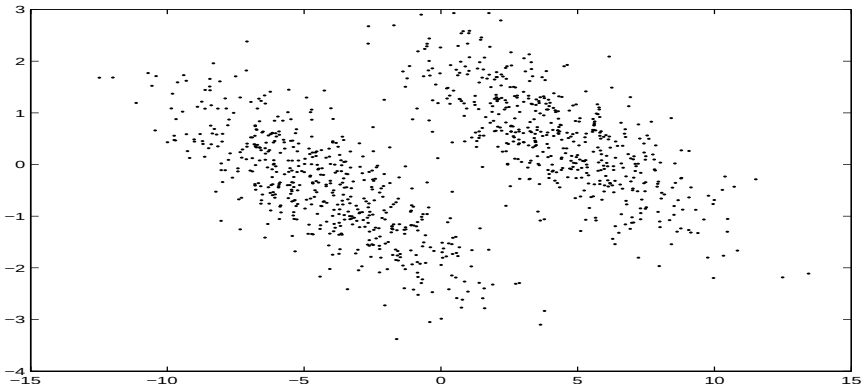


Fig. 2.2. The subspace spanned by the first two principal components

2.3 Independent Component Analysis

Independent component analysis aims to estimate the latent source from a set of observations [12]. Assume that we observe n linear mixtures of n independent components $s_1, s_2, \ldots, s_n$,

$$\begin{aligned} x_1 &= a_{11}s_1 + a_{12}s_2 + \cdots + a_{1n}s_n \\ x_2 &= a_{21}s_1 + a_{22}s_2 + \cdots + a_{2n}s_n \\ &\vdots \qquad\qquad \vdots \\ x_n &= a_{n1}s_1 + a_{n2}s_2 + \cdots + a_{nn}s_n \,. \end{aligned} \tag{2.6}$$

Without loss of generality, we assume that both the mixture variables and the independent components have zero mean. If this is not true, we can always center the mixture variables x_i by subtracting the sample means, which makes the independent components s_i zero mean as well.

Let $\mathbf{x}$ be the vector of the observed (mixture) variables $x_1, x_2, \ldots, x_n$, $\mathbf{s}$ the vector of the latent variables (independent components) $s_1, s_2, \ldots, s_n$, and $\mathbf{A}$ the matrix of the mixture coefficients a_{ij}. Using the vector-matrix notation, (2.6) can be written as

$$\mathbf{x} = \mathbf{A}\mathbf{s} \,. \tag{2.7}$$

The ICA model is a generative model because it describes how the observed data are generated by a process of mixing the latent components s_i. In (2.7), both the mixing matrix $\mathbf{A}$ and the latent vector $\mathbf{s}$ are unknown, and we must estimate both $\mathbf{A}$ and $\mathbf{s}$ using the observed vector $\mathbf{x}$.

It is clear from (2.7) that the ICA model is ambiguous because given any diagonal $n \times n$ matrix $\mathbf{R}$, we have

$$\begin{aligned} \mathbf{x} &= \mathbf{A}\mathbf{s} \\ &= \mathbf{A}\mathbf{R}^{-1}\mathbf{R}\mathbf{s} \\ &= \mathbf{A}^*\mathbf{s}^* . \end{aligned} \tag{2.8}$$

To make the solution unique, we add the constraint that requires each latent variable s_i to have the unit variance: $E[s_i^2] = 1, \forall i$. Note that this constraint still leaves the ambiguity of sign: we can multiply the latent variables by -1 without affecting the model. Fortunately, this ambiguity is not a serious problem in many applications.

The key assumption for ICA is that the latent variables s_i are statistically independent, and must have non-Gaussian distributions (see Sect. 2.3.2 for explanations). The standard ICA model also assumes that the mixing matrix $\mathbf{A}$ is square, but this assumption can be sometimes relaxed, as explained in [12]. With these assumptions, the ICA problem can be formulated as: Find a matrix $\mathbf{A}$ such that the latent variables obtained by

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{x} \tag{2.9}$$

are as independent and non-Gaussian as possible.

There are several metrics that can be used to measure the degrees of independence and non-Gaussianity. Here we provide three metrics that have been widely utilized in ICA implementations [12].

Kurtosis

Kurtosis is a classical measure of non-Gaussianity. The kurtosis of a random variable y is defined by

$$\text{kurt}(y) = E[y^4] - 3(E[y^2])^2 . \tag{2.10}$$

For a variable y with unit variance, $\text{kurt}(y) = E[y^4] - 3$, which is simply a normalized version of the fourth moment $E[y^4]$.

Kurtosis is zero for Gaussian variables, and non-zero for most (but not all) non-Gaussian random variables. Negative kurtosis values typically correspond to spiky probabilistic distributions that have a sharp peak and a long, low-altitude tail, while positive kurtosis values typically correspond to flat

probabilistic distributions that have a rather flat peak, and taper off gradually.

Kurtosis has some drawback in practice. It is very sensitive to outliers, meaning that a few data points in the tails of a distribution may largely affect its value. Therefore, kurtosis is not a robust measure of non-Gaussianity.

Negentropy

The differential entropy $H(\mathbf{y})$ of a random vector $\mathbf{y}$ is given by

$$H(\mathbf{y}) = -\int P(\mathbf{y}) \log P(\mathbf{y}) d\mathbf{y}\,, \tag{2.11}$$

where $P(\mathbf{y})$ is the probabilistic density distribution of $\mathbf{y}$. Entropy is a measurement of the degree of information on a random variable. The more random (i.e. unpredictable and unstructured) the variable is, the larger its entropy. A well-known result in the information theory says that among all random variables with an equal variance, Gaussian variables have the maximum entropy. This means that entropy can be used as a measure of non-Gaussianity. Inspired by this observation, Hyvarinen and Oja introduced the negentropy $J(\mathbf{y})$ defined by [13]

$$J(\mathbf{y}) = H(\mathbf{y}_g) - H(\mathbf{y})\,, \tag{2.12}$$

where $\mathbf{y}_g$ is a Gaussian random variable with the same covariance matrix as $\mathbf{y}$. Negentropy is always non-negative, and becomes zero if and only if $\mathbf{y}$ is a Gaussian variable.

Although negentropy is well justified, and has certain preferable statistical properties, its estimation, however, is problematic because it requires an estimation of the probabilistic density distribution $P(\mathbf{y})$, which is difficult to obtain for all but very simple problems.

In [13], Hyvarinen proposed a simple approximation to negentropy that can be estimated on empirical data. For a random variable $\mathbf{y}$ with zero mean and unit variance, the approximation is given by

$$J(\mathbf{y}) \approx \left(E[G(\mathbf{y})] - E[G(\mathbf{y}_g)]\right)^2\,, \tag{2.13}$$

where $\mathbf{y}_g$ is a Gaussian variable with zero mean and unit variable, and $G(\mathbf{y}) = \frac{1}{a}\log\cosh(a\mathbf{y})$ for $1 \le a \le 2$.

Mutual Information

The mutual information I between the components of a random vector $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$ is defined as

$$I(y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} H(y_i) - H(\mathbf{y}) \,. \tag{2.14}$$

The quantity $I(y_1, y_2, \ldots, y_n)$ is equivalent to the famous Kullback-Leibler divergence between the joint density $p(\mathbf{y})$ and the product of its marginal densities $\prod_{i=1}^{n} p(y_i)$, which is an independent version of $p(\mathbf{y})$. It is always non-negative, and becomes zero if and only if the variables are statistically independent.

Mutual information can be interpreted as a metric of the code length reduction from the information theory's point of view. The terms $H(y_i)$ give the code lengths for the components y_i when they are coded separately, and $H(\mathbf{y})$ gives the code length when all the components are coded together. Mutual information shows what code length reduction is obtained by coding the whole vector instead of the separate components. If the components y_i are mutually independent, meaning that they give no information on each other, then $\sum_{i=1}^{n} H(y_i) = H(\mathbf{y})$, and there will be no code length reduction no matter whether the components y_i are coded separately or jointly.

An important property of mutual information is that, for an invertible linear transformation $\mathbf{y} = \mathbf{W}\mathbf{x}$ we have

$$I(y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} H(y_i) - H(\mathbf{x}) - \log |\det \mathbf{W}| \,. \tag{2.15}$$

If both $\mathbf{x}$ and $\mathbf{y}$ have the identity covariance matrix $\mathbf{I}$, then $\mathbf{W}$ is a orthogonal matrix (see the derivation of (2.17)), and $I(y_1, y_2, \ldots, y_n)$ becomes

$$I(y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} H(y_i) - H(\mathbf{x}) \,. \tag{2.16}$$

This property implies that computation cost can be reduced if we conduct the whitening pre-processing before estimating the latent variables using (2.9) (see Sect. 2.3.1 for more descriptions).

2.3.1 Preprocessing

The most basic and necessary preprocessing is to center the observed variables $\mathbf{x}$, which means that we subtract $\mathbf{x}$ with its mean vector $\mathbf{m} = E[\mathbf{x}]$ to make $\mathbf{x}$ a zero-mean vector.

Another useful preprocessing is to first whiten the observed variables $\mathbf{x}$ before estimating $\mathbf{A}$ in (2.9). This means that we transform the observed variables $\mathbf{x}$ linearly into new variables $\tilde{\mathbf{x}} = \mathbf{B}\mathbf{x}$ such that $E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = \mathbf{I}$. The whitening preprocessing transforms the mixing matrix $\mathbf{A}$ in (2.9) into an orthogonal matrix. This can be seen from

$$\begin{aligned}\mathbf{I} = E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] &= E[\mathbf{B}\mathbf{x}(\mathbf{B}\mathbf{x})^T] \\ &= E[\mathbf{B}\mathbf{A}\mathbf{s}(\mathbf{B}\mathbf{A}\mathbf{s})^T] \\ &= E[\tilde{\mathbf{A}}\mathbf{s}(\tilde{\mathbf{A}}\mathbf{s})^T] \\ &= \tilde{\mathbf{A}}E[\mathbf{s}\mathbf{s}^T]\tilde{\mathbf{A}}^T \\ &= \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T, \end{aligned} \tag{2.17}$$

where $\tilde{\mathbf{A}} = \mathbf{B}\mathbf{A}$, and the last equality is derived from the assumption that the latent variables $\mathbf{s}$ are independent, have zero mean and unit variance.

Transforming the mixing matrix $\mathbf{A}$ into an orthogonal one reduces the number of parameters to be estimated. An $n \times n$ orthogonal matrix contains $n(n-1)/2$ degrees of freedom, while an arbitrary matrix of the same size contains n^2 elements (parameters). For matrixes with large dimensions, the whitening preprocessing roughly reduces the number of parameters to be estimated to half, which dramatically decreases the complexity of the problem.

The whitening preprocessing can be always accomplished using the eigenvalue decomposition of the covariance matrix $E[\mathbf{x}\mathbf{x}^T] = \mathbf{E}\mathbf{D}\mathbf{E}^T$, where $\mathbf{E}$ is the orthogonal matrix of the eigenvectors of $E[\mathbf{x}\mathbf{x}^T]$, and $\mathbf{D} = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ is the diagonal matrix of its eigenvalues. It is easy to verify that the vector $\tilde{\mathbf{x}}$ given by

$$\tilde{\mathbf{x}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{x} \tag{2.18}$$

satisfies $E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = \mathbf{I}$, and therefore, it is the whitened version of $\mathbf{x}$.

2.3.2 Why Gaussian is Forbidden

As demonstrated by (2.8), there exist certain ambiguities with the ICA formulation. The assumption of statistical independence of the latent variables $\mathbf{s}$ serves to remove these ambiguities. Intuitively, the assumption of non-correlation determines the covariances (the second-degree cross-moments) of a multivariate distribution, while the assumption of statistical independence determines all of the cross-moments. These extra moment conditions allow us to remove the ambiguities, and to uniquely identify elements of the mixing matrix

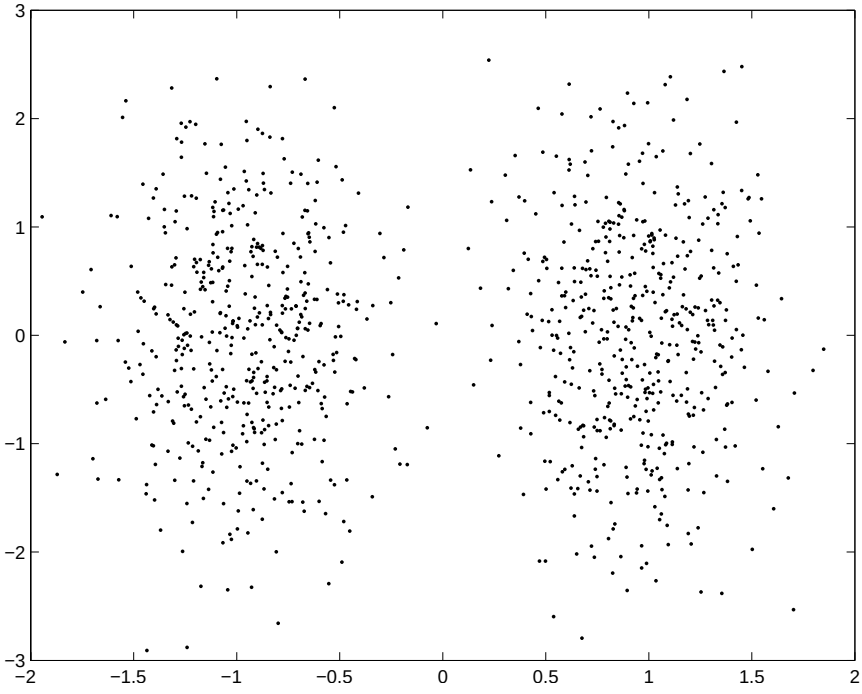


Fig. 2.3. The subspace spanned by the two independent components

A. The additional moment conditions, however, do not help Gaussian distributions because they are determined by the second-degree moments alone, and do not involve higher degree cross-moments. As a result, any Gaussian independent components can be only determined up to a rotation.

In summary, ICA aims to find a linear projection $\mathbf{A}$ of the observed data $\mathbf{x}$ such that the projected data $\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}$ look as far from Gaussian, and as independent as possible. This amounts to maximizing one of the non-Gaussian, independence metrics introduced in this section. Maximizing these metrics can be achieved using the standard gradient decent algorithm and its variations. An algorithm that efficiently computes the latent variables $\mathbf{s}$ by maximizing the approximation of negentropy given by (2.13) can be found in [12].

Figure 2.3 shows the subspace obtained by applying the ICA algorithm to the synthetic data set shown in Fig. 2.1. The data distribution in the figure confirms that the two axes of this subspace correspond to the two directions that provide the maximum statistical independence.

2.4 Dimension Reduction by Locally Linear Embedding

Many complex data represented by high-dimensional spaces typically have a much more compact description. Coherent structures in the world lead to strong correlations between components of objects (such as neighboring pixels in images), generating observations that lie on or close to a smooth low-dimensional manifold. Finding such a low-dimensional manifold for the given data set can not only provide a better insight into the internal structure of the data set, but also dramatically reduce the number of parameters to be estimated for constructing reasoning models.

In this section, we present one of the latest techniques for manifold computations: dimension reduction by locally linear embedding (LLE) [14]. The LLE method strives to compute a low-dimensional embedding of the high-dimensional inputs which preserves the neighborhood structure of the original space. The method also does not have the local minimum problem, and guarantees to generate the globally optimal solution.

The LLE algorithm is based on simple geometric intuitions. Consider a manifold in a high dimensional feature space, such as the one shown in Fig. 2.4. Such a manifold can be decomposed into many small patches. If each patch is small enough, it can be approximated as a linear patch. Assume that a data set sampled from the manifold consists of N real-valued, D-dimensional vectors $\mathbf{x}_i$. If we have sufficient data points such that the manifold is well-sampled, we expect each data point and its neighbors to lie on or close to a locally linear patch of the manifold. Therefore, each data point $\mathbf{x}_i$ can be reconstructed as a linear combination of its neighbors $\mathbf{x}_j$

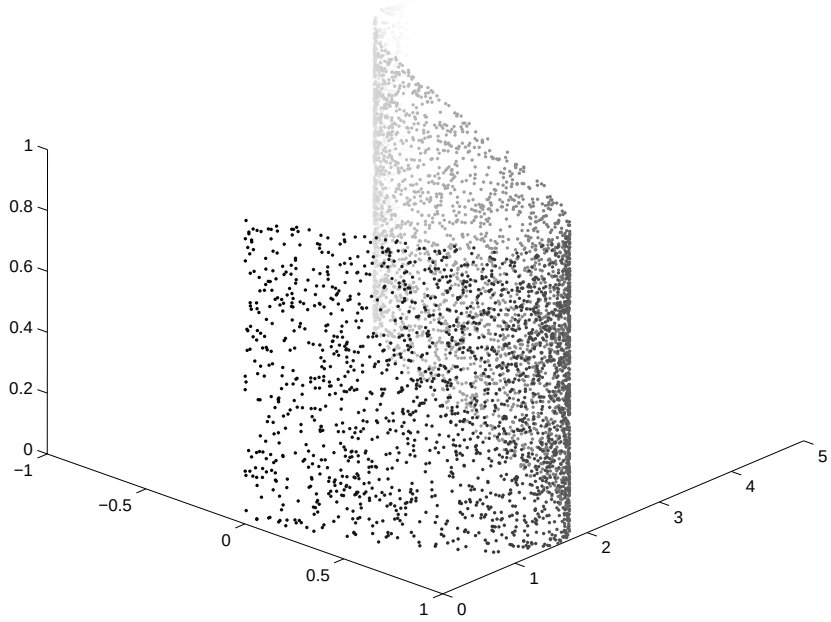
$$\mathbf{x}_i \approx \sum_j w_{ij}\mathbf{x}_j, \tag{2.19}$$

and the local geometry of each patch can be characterized by the linear coefficients w_{ij}. The LLE algorithm strives to find the matrix $\mathbf{W}$ of the linear coefficients w_{ij} for all the data points $\mathbf{x}_i$ by minimizing the following reconstruction error

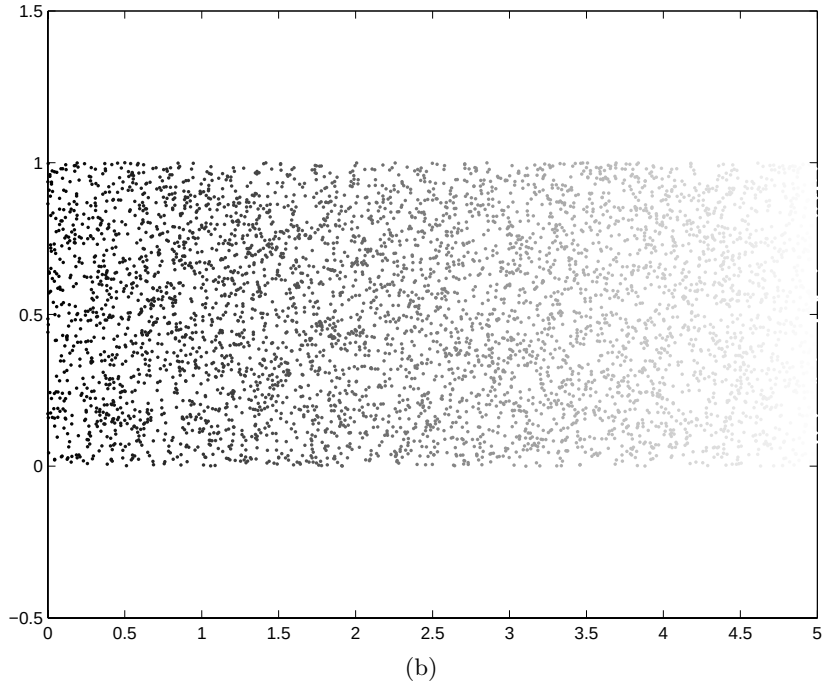
$$E(\mathbf{W}) = \sum_i ||\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j||^2 . \tag{2.20}$$

The minimization of the reconstruction error $E(\mathbf{W})$ is conducted subject two the following two constraints:

1. Each data point $\mathbf{x}_i$ is reconstructed only from its neighbors, enforcing $w_{ij} = 0$ if $\mathbf{x}_j$ does not belong to the set of neighbors of $\mathbf{x}_i$.



(a)



(b)

Fig. 2.4. An example of manifold. (a) shows a manifold in a 3-D space. (b) shows the projected manifold in the 2-D subspace generated by the LLE algorithm

2. The rows of the weight matrix $\mathbf{W}$ sum to one: $\sum_j w_{ij} = 1$.

The set of neighbors for each data point can be obtained either by choosing the K nearest neighbors in Euclidean distance, or by selecting data points within a fixed radius, or by using certain prior knowledge. The LLE algorithm described in [14] reconstructs each data point using its K nearest neighbors.

The optimal weights w_{ij} subject to the above two constraints can be obtained by solving a least-squares problem, and the result is given by

$$w_{ij} = \sum_k \mathbf{C}_{jk}^{-1}(\mathbf{x}_i \cdot \mathbf{x}_k + \lambda)\,, \tag{2.21}$$

where $\mathbf{C}^{-1}$ is the inverse of the neighborhood correlation matrix $\mathbf{C} = \{c_{jk}\}$, $c_{jk} = \mathbf{x}_j \cdot \mathbf{x}_k$, $\mathbf{C}_{jk}^{-1}$ is the (j,k)'th element of the inverse matrix $\mathbf{C}^{-1}$, and $\lambda = \frac{1-\sum_{jk}\mathbf{C}_{jk}^{-1}(\mathbf{x}_i\cdot\mathbf{x}_k)}{\sum_{jk}\mathbf{C}_{jk}^{-1}}$.

The constrained weights that minimize the reconstruction error $E(\mathbf{W})$ have the important property that for any data points, they are invariant to rotations, rescalings, and translations of the data points and their neighbors. Note that the invariance to translations is specifically enforced by the sum-to-one constraint on the rows of the weight matrix $\mathbf{W}$.

After obtaining the weight matrix $\mathbf{W}$, the next step is to find a linear mapping that maps the high-dimensional coordinates of each neighborhood to global internal coordinates on the manifold of lower dimensionality $d << D$. The linear mapping may consist of a translation, rotation, rescaling, etc. By design, the reconstruction weights w_{ij} reflect intrinsic geometric properties of the data that are invariant to exactly these transformations. Therefore, we expect their characterization of local geometry in the original data space to be equally valid for local patches on the manifold. In particular, the same weights w_{ij} that reconstruct the data point $\mathbf{x}_i$ in the original D-dimensional space should also reconstruct its embedded manifold coordinates in the lower d-dimensional space.

Based on the above idea, LLE constructs a neighborhood-preserving mapping matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$ that minimizes the following embedded cost function

$$\Theta(\mathbf{Y}) = \sum_i ||\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j||^2\,, \tag{2.22}$$

where $\mathbf{y}_i$ is the global internal coordinates of the data point $\mathbf{x}_i$ on the manifold. This cost function, like (2.20), is based on locally linear reconstruction errors, but here we fix the weights w_{ij} while optimizing the mapping matrix $\mathbf{Y}$. To

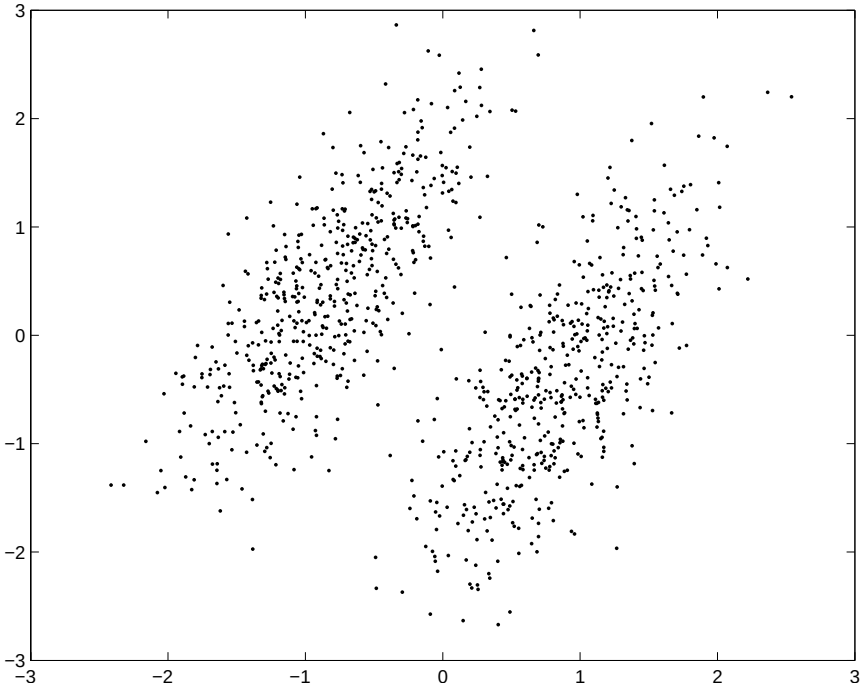


Fig. 2.5. The 2-D manifold obtained by the LLE algorithm

make the problem well-posed, the optimization is performed subject to the following two constraints:

1. The coordinates $\mathbf{y}_i$ are centered to the origin: $\sum_i \mathbf{y}_i = 0$. This is to remove the freedom that $\mathbf{y}_i$ can be translated by a constant displacement without affecting the cost $\Theta(\mathbf{Y})$.
2. The mapping matrix $\mathbf{Y}$ has an unit covariance matrix: $\mathbf{Y}\mathbf{Y}^T = \mathbf{I}$.

With the above two constraints, the optimal embedding, up to a global rotation of the embedding space, is obtained by computing the bottom $d+1$ eigenvectors of the matrix $\mathbf{M} = [m_{ij}]$, where

$$m_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_k w_{ki} w_{kj} \,, \tag{2.23}$$

and δ_{ij} is 1 if $i = j$ and 0 otherwise. The detailed mathematical derivations can be found in [14].

In summary, given the user's input on the number of dimensions d of the manifold and the number of neighbors K for each data point, the LLE algorithm consists of the following three major steps:

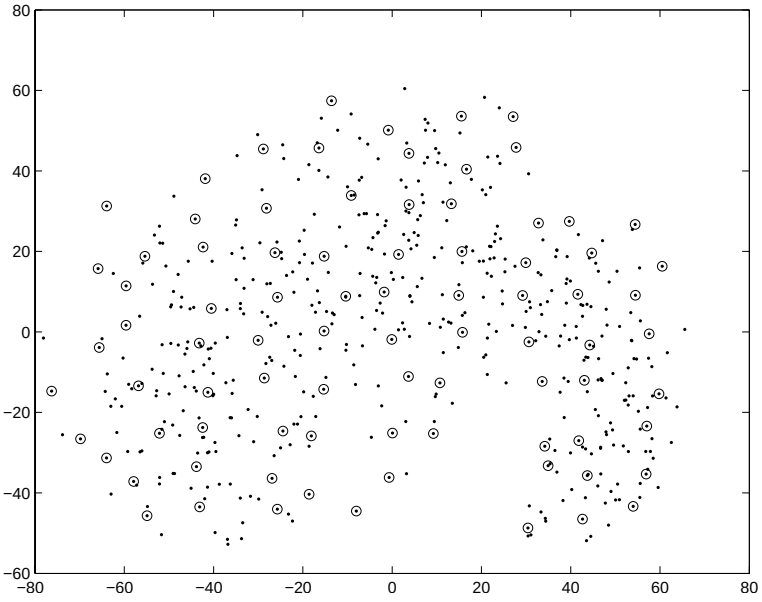
1. For each data point $\mathbf{x}_i$, choose the K nearest neighbors as its neighborhood set.
2. Use (2.21) to compute the optimal weights w_{ij}.
3. Use (2.23) to compute the matrix $\mathbf{M}$, and the embedding vectors $\mathbf{y}_i$.
4. Repeat Step $1 \sim 3$ until all the data points are processed.

Figure 2.5 shows the 2-D manifold obtained by applying the LLE algorithm to the synthetic data set shown in Fig. 2.1. The data distribution in the figure is almost identical to the one shown in Fig. 2.2 if we flip the space vertically. This indicates that the 2-D manifold is formed by preserving the first two principal components, and discarding the least important one of the original space.

2.5 Case Study

In this section, we provide a case study where the three dimension reduction techniques described in this chapter, namely SVD, ICA, and LLE, are applied to a subset of handwritten digits from the MNIST database [15]. The MNIST database has a total of 60,000 handwritten digits, each of which is normalized to a 28×28 gray-scale image with each pixel ranging in intensity from 0 to 255. Preprocessing is conducted to center each handwritten digit within the 28×28 image. Among the 60,000 handwritten digits, there are 5421 fives in the MNIST database, from which we have randomly selected 539 images to form our experimental test set.

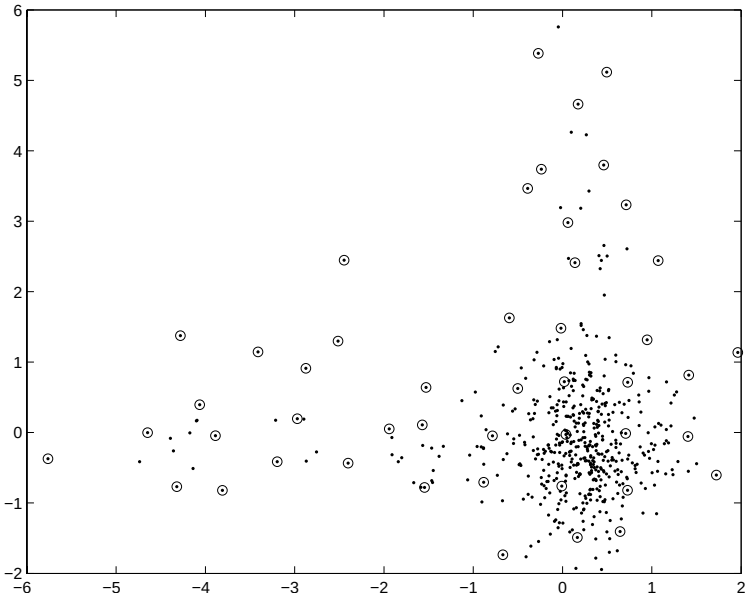
Figure 2.6 shows the subspace generated by the singular value decomposition. In this figure, (a) shows the subspace spanned by the first two principal components, where the circled points are the projected images closest to the vertices of a square grid, and (b) displays the images corresponding to the circled points in (a). Plot (b) allows us to visualize the natures of the first two principal components. We see that the horizontal axis mainly accounts for the length of the upper and lower tails of digit five, while the vertical axis accounts for character thickness. Although there are a total of 784 possible principal components, the first 50 components account for approximately 90% of the variation in handwritten fives.



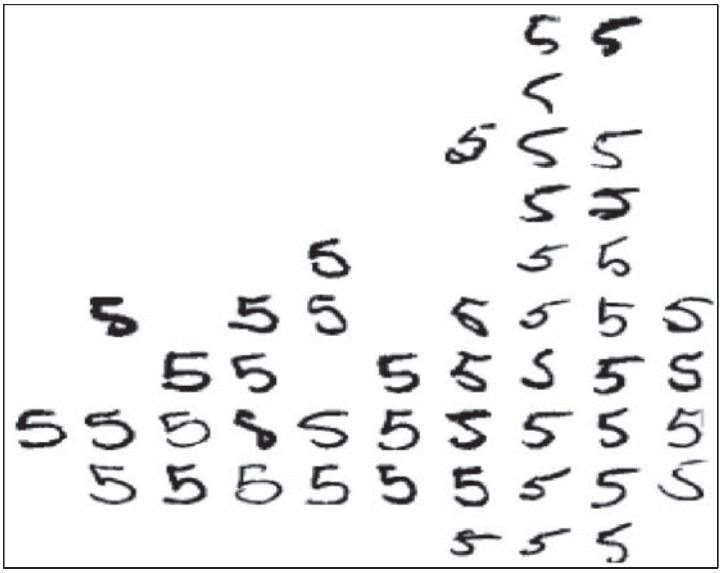
(a)

(b)

Fig. 2.6. The subspace generated by the singular value decomposition. (a) shows the subspace spanned by the first two principal components. The circled points are the projected images closest to the vertices of a square grid. (b) displays the images corresponding to the circled points in (a)

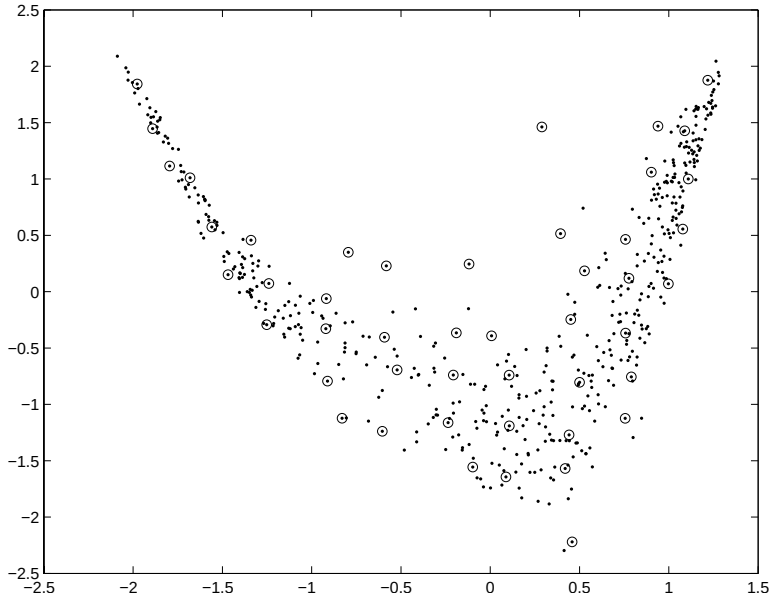


(a)

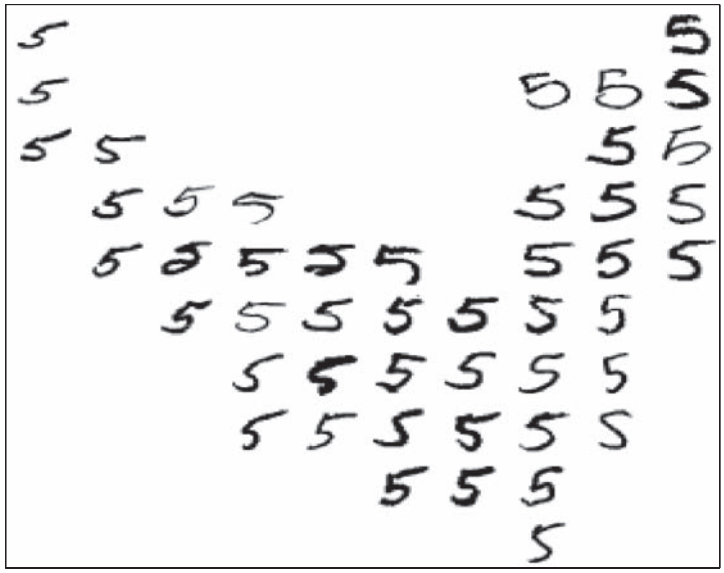


(b)

Fig. 2.7. The subspace generated by the independent component analysis. (a) shows the subspace spanned by the two independent components. The circled points are the projected images closest to the vertices of a square grid. (b) displays the images corresponding to the circled points in (a)



(a)



(b)

Fig. 2.8. The subspace generated by the locally linear embedding method. (a) shows the two-dimensional linear embedded space. The circled points are the projected images closest to the vertices of a square grid. (b) displays the images corresponding to the circled points in (a)

Figure 2.7 shows the subspace spanned by the two independent components. Same as Fig. 2.6, we superimpose a square grid on the space, and display the projected images that are closest to the vertices of the grid. It is not surprising that the subspace shown in (a) has a long-tailed distribution, because ICA specifically looks for non-Gaussian distributions. The sample images displayed in (b) do not show salient trends along the horizontal and the vertical axes, and we are unable to tell the physical implications of the two axes.

Figure 2.8 shows the two-dimensional subspace generated by the locally linear embedding method. From (b) we see that the horizontal axis mainly accounts for the lengths of the upper and lower tails, and the vertical axis accounts for the width of the handwritten fives.

Problems

2.1. Let $\mathbf{X} \in R^2$ follows uniform distribution in region $|X_1 + 2X_2| \leq 1$.
a) What is the principle components of $\mathbf{X}$?
b) What is the independent components of $\mathbf{X}$?

2.2. For an orthogonal linear projection matrix $\mathbf{V}^T\mathbf{V} = I$, prove

$$\|\mathbf{x} - \mathbf{V}\mathbf{V}^T\mathbf{x}\|^2 = \|\mathbf{x}\|^2 - \|\mathbf{V}\mathbf{V}^T\mathbf{x}\|^2 .$$

2.3. The singular value decomposition of a matrix $\mathbf{X}$ is defined as

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T .$$

Prove that for each column $\mathbf{v}_i$ of $\mathbf{V}$, the following equality holds:

$$\text{var}(\mathbf{X}\mathbf{v}_i) = d_i^2 ,$$

where d_i is the i'th eigenvalue.

2.4. Let X and Y be two Gaussian random variable. Show that the mutual information between X and Y is:

$$I(X, Y) = \frac{1}{2} \log \frac{1}{1 - \rho^2}$$

where $\rho = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}}$ is the correlation coefficient between X and Y.

2.5. Let $\mathbf{A}$ be an $M \times N$ matrix, $\mathbf{w} = [w_1, w_2, \ldots, w_N]^T$ an N dimensional vector, $\mathbf{b}$ an M dimensional vector. Find the solution for the following constrained optimization problem

$$\min_{\mathbf{w}} \|\mathbf{A}\mathbf{w} - \mathbf{b}\|^2 ,$$

subject to constraint $\sum_i w_i = 1$.

2.6. Let $P(X, Y)$ be the joint distribution function of random variables X and Y. $f(X, Y) = \log P(X, Y)$. Assume f is twice differentiable. Prove that $\frac{\partial^2 f}{\partial x \partial y} = 0$ if and only if X and Y are independent.

2.7. Let X_1 and X_2 be two independent random variables with the distribution functions $P(X)$ and $Q(X)$. Assume $f(X) = \log P(X)$ and $g(X) = \log Q(X)$ are twice differentiable. Prove that (a) and (b) are equivalent

(a) X_1 and X_2 are two Gaussian random variables with the same variance.

(b) For any $A \in R^{2\times 2}$ such that $AA^T = I$, $[Y_1 Y_2]^T = A[X_1 X_2]^T$ transforms X_1 and X_2 into two independent random variables Y_1 and Y_2. (Hint: Use Problem 2.6).

2.8. Show that the LLE algorithm is rotational and translational invariant, i.e., LLE will find the same result if the original data is subject to some rotation and/or translation.

2.9. Show that kernel trick can be applied to principle component analysis, i.e., the principle components can be obtained from inner products between the data vectors without the need of referring to the original vectors.