An interesting property of δ-pCluster is that if $I \times J$ is a δ-pCluster, then every $x \times y$ $(x, y \geq 2)$ submatrix of $I \times J$ is also a δ-pCluster. This monotonicity enables us to obtain a succinct representation of nonredundant δ-pClusters. A δ-pCluster is maximal if no more rows or columns can be added into the cluster while maintaining the δ-pCluster property. To avoid redundancy, instead of finding all δ-pClusters, we only need to compute all maximal δ-pClusters.

MaPle is an algorithm that enumerates all maximal δ-pClusters. It systematically enumerates every combination of conditions using a set enumeration tree and a depth-first search. This enumeration framework is the same as the pattern-growth methods for frequent pattern mining (Chapter 4). Consider gene expression data. For each condition combination, J, MaPle finds the maximal subsets of genes, I, such that $I \times J$ is a δ-pCluster. If $I \times J$ is not a submatrix of another δ-pCluster, then $I \times J$ is a maximal δ-pCluster.

There may be a huge number of condition combinations. MaPle prunes many unfruitful combinations using the monotonicity of δ-pClusters. For a condition combination, J, if there does not exist a set of genes, I, such that $I \times J$ is a δ-pCluster, then we do not need to consider any superset of J. Moreover, we should consider $I \times J$ as a candidate of a δ-pCluster only if for every $(|J| - 1)$-subset J' of J, $I \times J'$ is a δ-pCluster. MaPle also employs several pruning techniques to speed up the search while retaining the completeness of returning all maximal δ-pClusters. For example, when examining a current δ-pCluster, $I \times J$, MaPle collects all the genes and conditions that may be added to expand the cluster. If these candidate genes and conditions together with I and J form a submatrix of a δ-pCluster that has already been found, then the search of $I \times J$ and any superset of J can be pruned. Interested readers may refer to the bibliographic notes for additional information on the MaPle algorithm.

An interesting observation here is that the search for maximal δ-pClusters in MaPle is somewhat similar to mining frequent closed itemsets. Consequently, MaPle borrows the depth-first search framework and ideas from the pruning techniques of pattern-growth methods for frequent pattern mining. This is an example where frequent pattern mining and cluster analysis may share similar techniques and ideas.

An advantage of MaPle and the other algorithms that enumerate all biclusters is that they guarantee the completeness of the results and do not miss any overlapping biclusters. However, a challenge for such enumeration algorithms is that they may become very time consuming if a matrix becomes very large, such as a customer-purchase matrix of hundreds of thousands of customers and millions of products.

9.4 Dimensionality reduction for clustering

Subspace clustering methods try to find clusters in subspaces of the original data space. In some situations, it is more effective to construct a new space instead of using subspaces of the original data. This is the motivation behind dimensionality reduction methods for clustering high-dimensional data. This section explores dimensionality reduction for clustering.

We start from the traditional linear dimensionality reduction methods and use principal component analysis (PCA) as an example. Then, we look at the general dimensionality and matrix decomposition methods. We use nonnegative matrix factorization (NMF) methods as an example and the general framework, and explain the relation between NMF and the traditional k-means clustering. Last, we change the angle and discuss spectral clustering, which rebuilds a new feature space based on the similarity graph and conduct clustering.

9.4.1 Linear dimensionality reduction methods for clustering

In many applications, the challenges in clustering analysis on high dimensional data come from two obstacles in data sets.

First, when a high-dimensional data set is collected, attributes may be correlated. For example, if we use three cameras to detect the spatial positions of objects, that is, the locations of objects are projected into three 2-D spaces. If each camera captures the 2-D coordinates of an object, then each object is recorded in a 6-D space. As the objects are in a 3-D space, the underlying location information should be described sufficiently using three intrinsic dimensions. Some correlation and thus redundancy may exist among the data collected using the three cameras. In other words, although a data set may have many dimensions, the underlying structures and relations may have a substantially lower dimensionality, which are often deeply hidden.

Second, observations on different attributes may be recorded using different scales and are not normalized properly. For example, different cameras may record the coordinates using different units, some may measure in metric and the others may measure in imperial. Moreover, those three cameras may not be set up orthogonally, two may be close to each other and one may be far away. In other words, the data recorded may be stretched.

Dimensionality reduction, or dimension reduction, transforms a high-dimensional data set into a low-dimensional space so that the low-dimensional representation retains meaningful properties of the original data, ideally approaching the intrinsic dimensions of the underlying structures. To understand the intuition, let us consider the following example.

Example 9.12. Clustering in a derived space. Consider the two clusters of points in Fig. 9.11. It is not possible to cluster these points in any subspace of the original space, $X \times Y$, because both clusters would end up being projected onto overlapping areas in the x and y axes.

What if, instead, we construct a new dimension, $-\frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y$ (shown as a dashed line in the figure)? By projecting the points onto this new dimension, the two clusters become apparent. □

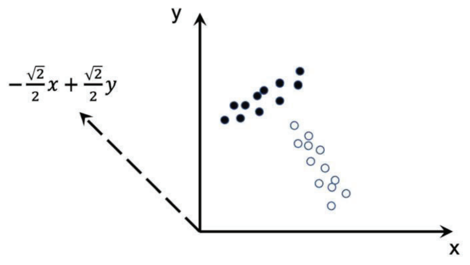

FIGURE 9.11

Clustering in a derived space may be more effective.

Although Example 9.12 involves only two dimensions, the idea of constructing a new space (so that any clustering structure hidden in the data becomes well manifested) can be extended to high-dimensional data. Preferably, the newly constructed space should have low dimensionality.

There are many dimensionality reduction methods. **Principal component analysis** (PCA) is frequently used to identify the most meaningful basis to re-express a data set. Before conducting clustering analysis, one may first apply PCA to reduce the dimensionality of a data set. Applying PCA can help to remove or reduce the correlation among attributes and normalize the attributes.

Let us take a look at the intuition behind PCA. Suppose we have n data objects, each of m dimensions. To represent the original data set, we can write an $n \times m$ matrix $\mathbf{X}$, where each row represents an object, and each column represents a dimension. In general, a **linear transformation** tries to find an $m \times m$ matrix $\mathbf{P}$ such $\mathbf{XP} = \mathbf{Y}$, where $\mathbf{Y}$ is another $m \times n$ matrix. Intuitively, matrix $\mathbf{P}$ is a rotation and a stretch that transforms $\mathbf{X}$ to $\mathbf{Y}$, and the columns of $\mathbf{P}$ are the set of new basis vectors re-expressing the rows of $\mathbf{X}$. To further illustrate the intuition, let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be the rows of $\mathbf{X}$, that is, the row vectors of the objects, and $\mathbf{p}_1, \ldots, \mathbf{p}_m$ be the columns of $\mathbf{P}$. Then, we have

$$\mathbf{XP} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_m \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1\mathbf{p}_1 & \cdots & \mathbf{x}_1\mathbf{p}_m \\ \vdots & \ddots & \vdots \\ \mathbf{x}_n\mathbf{p}_1 & \cdots & \mathbf{x}_n\mathbf{p}_m \end{bmatrix} = \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix},$$

where $\mathbf{y}_1, \ldots, \mathbf{y}_n$ are the rows of $\mathbf{Y}$. Then, we have

$$\mathbf{y}_i = \begin{bmatrix} \mathbf{x}_i\mathbf{p}_1 & \cdots & \mathbf{x}_i\mathbf{p}_m \end{bmatrix}.$$

As can be seen, each data object $\mathbf{x}_i$ is transformed to $\mathbf{y}_i$ by a dot-product with the corresponding column of P. In other words, $\mathbf{y}_i$ is the projection of $\mathbf{x}_i$ on to the basis of $\mathbf{p}_1, \ldots, \mathbf{p}_m$. Through this transformation, the data set $\mathbf{X}$ is re-expressed as $\mathbf{Y}$ using the new basis of $\mathbf{p}_1, \ldots, \mathbf{p}_m$.

In a nutshell, PCA is a method to choose a good base $\mathbf{P}$ that best re-express $\mathbf{X}$ into $\mathbf{Y}$ so that signals are maximized and noise is minimized. Let us measure the noise and redundancy in the original representation $\mathbf{X}$. Recall that the correlation between two attributes A and B can be measured by the covariance $\sigma_{AB}^2 = \frac{1}{n}\sum_{i=1}^{n} a_i b_i$, the larger the more correlation. To measure the correlation between every two dimensions in $\mathbf{X}$, PCA computes the covariance between very pair of dimensions by constructing the covariance matrix $\mathbf{C_X} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$. The covariance matrix $\mathbf{C_X}$ reflects the noise and redundancy in the original representation $\mathbf{X}$.

By transforming $\mathbf{X}$ to $\mathbf{Y}$, PCA tries to remove or reduce redundancy caused by correlation among dimensions in the original data space. Thus the covariance matrix $\mathbf{C_Y}$ of $\mathbf{Y}$ should have two properties. First, all off-diagonal terms in $\mathbf{C_Y}$ should be zero, that is, the dimensions in the new basis are not correlated with each other. Second, the dimensions in $\mathbf{Y}$ should be ordered by variance, the larger the variance, the more signals the dimension carries, and thus the more important the dimension.

We can build the connection between the covariance matrices $\mathbf{C_X}$ and $\mathbf{C_Y}$ as follows.

$$\begin{aligned} \mathbf{C_Y} &= \frac{1}{n}\mathbf{Y}^T\mathbf{Y} = \frac{1}{n}(\mathbf{XP})^T(\mathbf{XP}) = \frac{1}{n}\mathbf{P}^T\mathbf{X}^T\mathbf{XP} = \mathbf{P}^T(\frac{1}{n}\mathbf{X}^T\mathbf{X})\mathbf{P} \\ &= \mathbf{P}^T\mathbf{C_X}\mathbf{P}. \end{aligned} \tag{9.26}$$

From linear algebra, we know that every symmetric matrix is diagonalized by an orthogonal matrix of its eigenvectors. The covariance matrix $\mathbf{C_X}$ is obviously symmetric. The matrix $\mathbf{P}$ where each column

$\mathbf{p}_i$ is an eigenvector of $\mathbf{C_X} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$ is exactly the transformation matrix serving our purpose. The eigenvectors are ranked in the eigenvalue descending order.

To summarize the above rationale, computing PCA in practice of a data set **X** can be conducted in three steps.

1. **Normalize X**. For each dimension in **X**, we calculate the mean of each column, subtract off the mean of every observation on the dimension and normalize by the standard deviation. That is, entry x_{ij} is normalized to $\frac{x_{ij}-\mu_j}{\sigma_j}$, where μ_j and σ_j are the mean and the standard deviation of the jth column, respectively. For the sake of simplicity, let us still denote by **X** the normalized matrix.
2. **Compute the eigenvectors of the covariance matrix $\mathbf{C_X}$**. The eigenvectors form the new basis.
3. **Choose the top-k eigenvectors and transform the original data in the new space of reduced dimensionality**. The eigenvalues reflect the variances on the corresponding eigenvectors. We can choose the top-k eigenvectors as the new basis and reduce the dimensionality so that the cumulated eigenvalue dominates the sum of eigenvalues.

Example 9.13. Consider a 3-D data set that has four objects:

$$\mathbf{X} = \begin{bmatrix} 5 & 9 & 3 \\ 4 & 10 & 6 \\ 3 & 8 & 11 \\ 6 & 3 & 7 \end{bmatrix}.$$

After normalization, the matrix is

$$\mathbf{X} = \begin{bmatrix} 0.387 & 0.482 & -1.135 \\ -0.387 & 0.804 & -0.227 \\ -1.162 & 0.161 & 1.286 \\ 1.162 & -1.447 & 0.076 \end{bmatrix}.$$

The covariance matrix is

$$\mathbf{C_X} = \begin{bmatrix} 0.750 & -0.411 & -0.439 \\ -0.411 & 0.549 & -0.152 \\ -0.439 & -0.152 & 0.750 \end{bmatrix}.$$

The three eigenvectors are $[-1.339, 0.567, 1]^T$ with eigenvalue 1.252, $[0.282, -1.098, 1]^T$ with eigenvalue 0.793, and $[1.270, 1.237, 1]^T$ with eigenvalue 0.004. In other words, the transformation matrix is

$$\mathbf{P} = \begin{bmatrix} -1.339 & 0.282 & 1.270 \\ -0.567 & -1.098 & 1.237 \\ 1 & 1 & 1 \end{bmatrix}.$$

Correspondingly, **X** is transformed to

$$\mathbf{Y} = \mathbf{XP} = \begin{bmatrix} -8.798 & -5.472 & 20.483 \\ -5.026 & -3.852 & 23.45 \\ 2.447 & 3.062 & 24.706 \\ -2.735 & 5.398 & 18.331 \end{bmatrix}.$$

Since the last eigenvalue 0.004 is very small comparing to the first two eigenvalues, we can reduce the dimensionality from 3 to 2 by dropping the last dimension in **Y**. That is, the data set can be represented in the 2-D space using the first two eigenvectors as the basis. The representation is

$$\begin{bmatrix} -8.798 & -5.472 \\ -5.026 & -3.852 \\ 2.447 & 3.062 \\ -2.735 & 5.398 \end{bmatrix}.$$

□

To recap, the core idea of PCA is to assume that the variance along a small number of principal components can provide a good characterization of a high-dimensional data set. PCA is parameter-free. One can apply PCA on any numeric data sets. On the one hand, this is an advantage since it is easy to use and thus is employed in many applications. On the other hand, the agnostic nature of PCA to data sources also comes as a weakness, since it cannot take advantage of any background knowledge to conduct dimensionality reduction.

9.4.2 Nonnegative matrix factorization (NMF)

What is the intuitive idea behind the popularly used nonnegative matrix factorization methods? We know that a data set **X** of n objects in an m-dimensional space can be represented using an $n \times m$ matrix, where each row represents an object and each column represents a dimension. Intuitively, the problem of clustering the objects in **X** into k clusters can be modeled as factorizing **X** into two matrices **W** and **H** such that $\mathbf{X} \approx \mathbf{HW}$, where **H** is an $n \times k$ matrix representing how the n objects are assigned to the k clusters, and **W** is a $k \times m$ matrix and each row in **W** represents the "center" of a cluster. An object may be similar to more than one cluster. Therefore the entries in **H** are nonnegative. Here, we do not want to have a negative entry in **H**, since a negative weight between an object and a cluster is hard to explain.

For example, consider a database of 2429 facial images, each consisting of 19×19 pixels. A 2429×361 matrix is constructed. Fig. 9.12 shows the 7×7 montages. The example face shown at top right

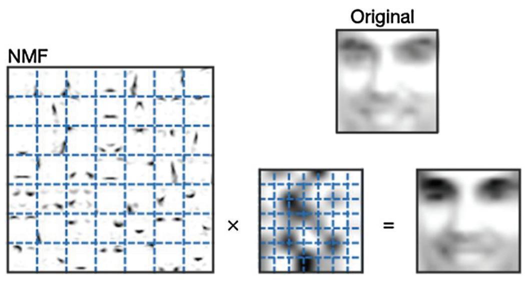

FIGURE 9.12

An example of using NMF in clustering images. Extracted from D. D. Lee and H. S. Seung, *Nature* Vol. 401, October 21, 1999.