Chapter 6
Data Reduction

Abstract The most common tasks for data reduction carried out in Data Mining consist of removing or grouping the data through the two main dimensions, examples and attributes; and simplifying the domain of the data. A global overview to this respect is given in Sect. 6.1. One of the well-known problems in Data Mining is the "curse of dimensionality", related with the usual high amount of attributes in data. Section 6.2 deals with this problem. Data sampling and data simplification are introduced in Sects. 6.3 and 6.4, respectively, providing the basic notions on these topics for further analysis and explanation in subsequent chapters of the book.

6.1 Overview

Currently, it is not difficult to imagine the disposal of a data warehouse for an analysis which contains millions of samples, thousands of attributes and complex domains. Data sets will likely be huge, thus the data analysis and mining would take a long time to give a respond, making such analysis infeasible and even impossible.

Data reduction techniques can be applied to achieve a reduced representation of the data set,it is much smaller in volume and tries to keep most of the integrity of the original data [11]. The goal is to provide the mining process with a mechanism to produce the same (or almost the same) outcome when it is applied over reduced data instead of the original data, at the same time as when mining becomes efficient. In this section, we first present an overview of data reduction procedures. A closer look at each individual technique will be provided throughout this chapter.

Basic data reduction techniques are usually categorized into three main families: *DR*, *sample numerosity reduction* and *cardinality reduction*.

DR ensures the reduction of the number of attributes or random variables in the data set. DR methods include *FS* and *feature extraction/construction* (Sect. 6.2 and Chap. 7 of this book), in which irrelevant dimensions are detected, removed or combined. The transformation or projection of the original data onto a smaller space can be done by *PCA* (Sect. 6.2.1), *factor analysis* (Sect. 6.2.2), *MDS* (Sect. 6.2.3) and *LLE* (Sect. 6.2.4), being the most relevant techniques proposed in this field.

S. García et al., *Data Preprocessing in Data Mining*,
Intelligent Systems Reference Library 72, DOI 10.1007/978-3-319-10247-4_6

Sample numerosity reduction methods replace the original data by an alternative smaller data representation. They can be either parametric or non-parametric methods. The former requires a model estimation that fits the original data, using parameters to represent the data instead of the actual data. They are closely-related DM techniques (regression and log-linear models are common parametric data reduction techniques) and we consider their explanation to be out of the scope of this book. However, non-parametric methods work directly with data itself and return other data representations with similar structures. They include *data sampling* (Sect. 6.3), different forms of data grouping, such as *data condensation*, *data squashing* and *data clustering* (Sects. 6.3.1, 6.3.2 and 6.3.3, respectively) and IS as a more intelligent form of sample reduction (Chap. 8 of this book).

Cardinality reduction comprises the transformations applied to obtain a reduced representation of the original data. As we have mention at the beginning of this book, there may be a high level of overlapping between data reduction techniques and data preparation techniques, this category being a representative example with respect to data transformations. As data reduction, we include the *binning* process (Sect. 6.4) and the more general discretization approaches (Chap. 9 of this book).

In the next sections, we will define the main aspects of each one of the aforementioned strategies.

6.2 The Curse of Dimensionality

A major problem in DM in large data sets with many potential predictor variables is the *the curse of dimensionality*. Dimensionality becomes a serious obstacle for the efficiency of most of the DM algorithms, because of their computational complexity. This statement was coined by Richard Bellman [4] to describe a problem that increases as more variables are added to a model.

High dimensionality of the input increases the size of the search space in an exponential manner and also increases the chance to obtain invalid models. It is well known that there is a linear relationship between the required number of training samples with the dimensionality for obtaining high quality models in DM [8]. But when considering non-parametric learners, such as those instance-based or decision trees, the situation is even more severe. It has been estimated that as the number of dimensions increase, the sample size needs to increase exponentially in order to have an effective estimate of multivariate densities [13].

It is evident that the curse of dimensionality affects data differently depending on the following DM task or algorithm. For example, techniques like decision trees could fail to provide meaningful and understandable results when the number of dimensions increase, although the speed in the learning stage is barely affected. On the contrary, instance-based learning has high dependence on dimensionality affecting its order of efficiency.

In order to alleviate this problem, a number of dimension reducers have been developed over the years. As linear methods, we can refer to factor analysis [18] and

PCA [7]. Nonlinear models are LLE [25], ISOMAP [26] and derivatives. They are concerned with the transformation of the original variables into a smaller number of projections. The underlying assumptions are that the variables are numeric and that the dimensions can be expressed as combinations of the actual variables, and vice versa. Further analysis on this type of techniques will be given in this chapter, especially for the two most popular techniques: PCA and LLE.

A set of methods are aimed at eliminating irrelevant and redundant features, reducing the number of variables in the model. They belong to the FS family of methods. They have the following immediate positive effects on the analysis and mining:

- Speed up the processing of the DM algorithm.
- Improve data quality.
- Increase the performance of the DM algorithm.
- Make the results easier to understand.

Formally, the problem of FS can be defined as follows [14]: Let A be the original set of features, with cardinality m. Let f represent the desired number of features in the selected subset B, $B \subset A$. Let the FS criterion function for the set B be represented by $J(B)$. Without any loss of generality, a lower value of J is considered to be a better feature subset, thus, J could represent the generalization error. The problem of FS is to find an optimal subset B that solves the following optimization problem:

$$\begin{aligned} &\min J(Z) \\ &s.t. \\ &Z \subset A \\ &|Z| = d \end{aligned}$$

A brute force search would require examining all $\frac{m!}{d!\cdot(m-d)!}$ possible combinations of the feature set A. A vast number of FS approaches, trends and applications have been proposed over the years, and therefore FS deserves a complete chapter of this book: Chap. 7.

Other forms of widely used DR also deserve to be described in this section. They are slightly more complicated than that previously seen, but also very widely used in conjunction with advanced DM approaches and real applications.

6.2.1 Principal Components Analysis

In this subsection, we introduction the Principal Components Analysis (PCA) as a DR method [17]. A detailed theoretical explanation is out of the scope of this book, hence we intend to give details on the basic idea, the method of operation and the objectives this technique pursues. PCA is one of the oldest and most used methods for reduction of multidimensional data.

The basic idea is to find a set of linear transformations of the original variables which could describe most of the variance using a relatively fewer number of variables. Hence, it searches for k n-dimensional orthogonal vectors that can best represent the data, where $k \leq n$. The new set of attributes are derived in a decreasing order of contribution, letting the first obtained variable, the one called *principal component* contain the largest proportion of the variance of the original data set. Unlike FS, PCA allows the combination of the essence of original attributes to form a new smaller subset of attributes.

The usual procedure is to keep only the first few principal components that may contain 95 % or more of the variance of the original data set. PCA is particularly useful when there are too many independent variables and they show high correlation.

The basic procedure is as follows:

- To normalize the input data, equalizing the ranges among attributes.
- To compute k orthonormal vectors to provide a basis for the normalized input data. These vectors point to a direction that is perpendicular to the others and are called *principal components*. The original data is in linear combination of the principal components. In order to calculate them, the eigenvalue-eigenvectors of the covariance matrix from the sample data are needed.
- To sort the principal components according to their strength, given by their associated eigenvalues. The principal components serve as a new set of axes for the data, adjusted according the variance of the original data. In Fig. 6.1, we show an illustrative example of the first two principal components for a given data set.

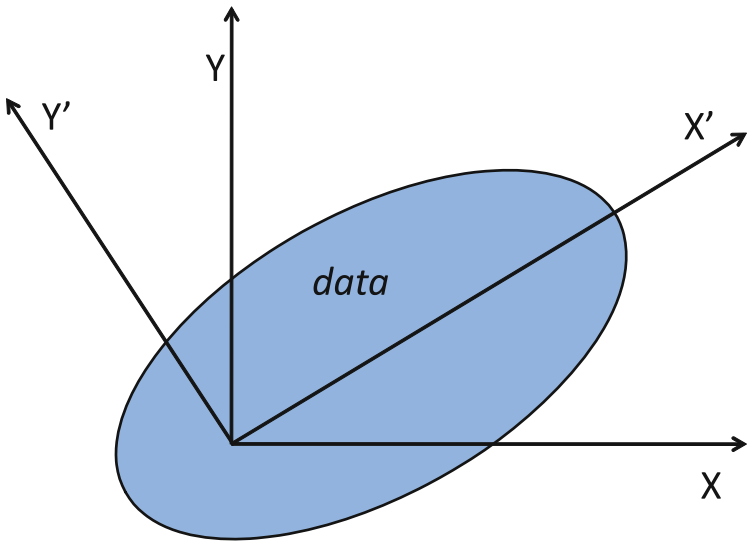

Fig. 6.1 PCA. X' and Y' are the first two principal components obtained

- To reduce the data by removing weaker components, with low variance. A reliable reconstruction of the data could be possible by using only the strongest principal components.

The final output of PCA is a new set of attributes representing the original data set. The user would use only the first few of these new variables because they contain most of the information represented in the original data. PCA can be applied to any type of data. It is also used as a data visualization tool by reducing any multidimensional data into two- or three-dimensional data.

6.2.2 Factor Analysis

Factor analysis is similar to PCA in the sense that it leads to the deduction of a new, smaller set of variables that practically describe the behaviour given in the original data. Nevertheless, factor analysis is different because it does not seek to find transformations for the given attributes. Instead, its goal is to discover hidden factors in the current variables [17]. Although factor analysis has an important role as a process of data exploration, we limit its description to a data reduction method.

In factor analysis, it is assumed that there are a set of unobservable *latent factors* z_j, $j = 1, \ldots, k$; which when acting together generate the original data. Here, the objective is to characterize the dependency among the variables by means of a smaller number of factors.

The basic idea behind factor analysis is to attempt to find a set of hidden factors so that the current attributes can be recovered by performing a set of linear transformations over these factors. Given the set of attributes $a_1, a_2, \ldots, a_m$, factor analysis attempts to find the set of factors $f_1, f_2, \ldots, f_k$, so that

$$\begin{aligned} a_1 - \mu_1 &= l_{11} f_1 + l_{12} f_2 + \cdots + l_{1k} f_k + \varepsilon_1 \\ a_2 - \mu_2 &= l_{21} f_1 + l_{22} f_2 + \cdots + l_{2k} f_k + \varepsilon_2 \\ &\vdots \\ a_m - \mu_m &= l_{m1} f_1 + l_{m2} f_2 + \cdots + l_{mk} f_k + \varepsilon_m \end{aligned}$$

where $\mu_1, \mu_2, \ldots, \mu_m$ are the means of the attributes $a_1, a_2, \ldots, a_m$, and the terms $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m$ represent the unobservable part of the attributes, also called *specific factors*. The terms l_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, k$ are known as the loadings. The factors $f_1, f_2, \ldots, f_k$ are known as the *common factors*.

The previous equation can be written in matrix form as:

$$\mathbf{A} - \mu = \mathbf{LF} + \varepsilon$$

Thus, the factor analysis problem can be stated as given the attributes $\mathbf{A}$, along with the mean μ, we endeavor to find the set of factors $\mathbf{F}$ and the associated loadings $\mathbf{L}$, and therefore the above equation is accurate.

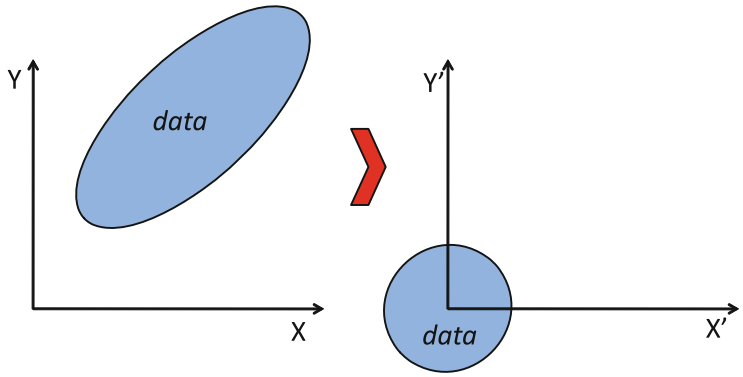

Fig. 6.2 Factors are independent unit normals that are scaled, rotated and translated to compose the inputs

To find **F** and **L**, three common restrictions on their statistical properties are adopted: (1) all the factors are independent, with zero mean and variance of unity, (2) all the error terms are also independent, with zero mean and constant variance, (3) the errors are independent of the factors.

There are two methods for solving the factor model equations for the matrix **K** and the factors **F**: (1) the maximum likelihood method and (2) the principal component method. The first assumes that original data is normally distributed and is computationally expensive. The latter is very fast, easy to interpret and guarantees to find a solution for all data sets.

1. Unlike PCA, factor analysis assumes and underlying structure that relates the factors to the observed data.
2. PCA tries to rotate the axis of the original variables, using a set of linear transformations. Factor analysis, instead, creates a new set of variables to explain the covariances and correlations between the observed variables.
3. In factor analysis, a two-factor model is completely different from a three-factor model, whereas in PCA, when we decide to use a third component, the two first principal components remain the same.
4. PC is fast and straightforward. However, in factor analyses, there are various alternatives to performing the calculations and some of them are complicated and time consuming.

Figure 6.2 exemplifies the process of factor analysis. The differences between PCA and factor analysis can be enumerated.

6.2.3 Multidimensional Scaling

Let us assume N points, and that we know the distances between the pairs of points, d_{ij}, for all $i, j = 1, \ldots, N$. Moreover, we do not know the precise coordinates of the

points, their dimensionality or the way the distances between them were computed. *Multidimensional scaling* (MDS) is the method for situating these points in a low space such that a classical distance measure (like Euclidean) between them is as close as possible to each d_{ij}. There must be a projection from some unknown dimensional space to another space whose number of dimensions is known.

One of the most typical examples of MDS is to draw an approximation of the map that represents the travel distances between cities, knowing only the distance matrix. Obviously, the outcome is distorted due to the differences between the distances measured taking into account the geographical obstacles and the actual distance in a straight line between the cities. It common for the map to be stretched out to accommodate longer distances and that the map also is centered on the origin. However, the solution is not unique, we can get any rotating view of it.

MDS is within the DR techniques because we can compute the distances in a d-dimensional space of the actual data points and then to give as input this distance matrix to MDS, which then projects it in to a lower-dimensional space so as to preserve these distances.

Formally, let us say we have a sample $X = \{x^t\}_{t=1}^N$ as usual, where $x^t \in \mathbb{R}^d$. For the two points r and s, the squared Euclidean distance between them is

$$\begin{aligned} d_{rs}^2 &= ||x^r - x^s||^2 = \sum_{j=1}^{d} (x_j^r - x_j^s)^2 = \sum_{j=1}^{d} (x_j^r)^2 - 2\sum_{j=1}^{d} x_j^r x_j^s + \sum_{j=1}^{d} (x_j^s)^2 \\ &= b_{rr} + b_{ss} - 2b_{rd} \end{aligned}$$

where b_{rs} is defined as

$$b_{rs} = \sum_{j=1}^{d} x_j^r x_j^s$$

To constrain the solution, we center the data at the origin and assume

$$\sum_{t=1}^{N} x_j^t = 0, \quad \forall j = 1, \ldots, d$$

Then, summing up the previous equation on r, s, and defining

$$T = \sum_{t=1}^{n} b_{tt} = \sum_t \sum_j (x_j^t)^2$$

we get

$$\sum_r d_{rs}^2 = T + N b_{ss}$$

$$\sum_s d_{rs}^2 = Nb_{rr} + T$$

$$\sum_r \sum_s d_{rs}^2 = 2NT$$

When we define

$$d_{\cdot s}^2 = \frac{1}{N}\sum_r d_{rs}^2, d_{r\cdot}^2 = \frac{1}{N}\sum_s d_{rs}^2, d_{\cdot\cdot}^2 = \frac{1}{N^2}\sum_r \sum_s d_{rs}^2$$

and using the first equation, we get

$$b_{rs} = \frac{1}{2}(d_{r\cdot}^2 + d_{\cdot s}^2 - d_{\cdot\cdot}^2 - d_{rs}^2)$$

Having now calculated b_{rs} and knowing that $\mathbf{B} = \mathbf{X}\mathbf{X}^T$, we look for an approximation. We know from the spectral decomposition that $\mathbf{X} = \mathbf{C}\mathbf{D}^{1/2}$ can be used as an approximation for $\mathbf{X}$, where $\mathbf{C}$ is the matrix whose columns are the eigenvectors of $\mathbf{B}$ and $\mathbf{D}^{1/2}$ is a diagonal matrix with square roots of the eigenvalues on the diagonals. Looking at the eigenvalues of $\mathbf{B}$ we decide on a dimensionality k lower than that of d. Let us say $\mathbf{c}_j$ are the eigenvectors with λ_j as the corresponding eigenvalues. Note that $\mathbf{c}_j$ is N-dimensional. Then we get the new dimension as

$$z_j^t = \sqrt{\lambda_j c_j^t}, \quad j = 1, \ldots, k, \ t = 1, \ldots, N$$

That is, the new coordinates of instance t are given by the tth elements of the eigenvectors, $\mathbf{c}_j$, $j = 1, \ldots, k$, after normalization.

In [5], it has been shown that the eigenvalues of $\mathbf{X}\mathbf{X}^T$ $(N \times N)$ are the same as those of $\mathbf{X}^T\mathbf{X}$ $(d \times d)$ and the eigenvectors are related by a simple linear transformation. This shows that PCA does the same work with MDS and does it more easily.

In the general case, we want to find a mapping $\mathbf{z} = g(\mathbf{x}|\theta)$, where $\mathbf{z} \in \mathbb{R}^k$, $\mathbf{x} \in \mathbb{R}^d$, and $g(\mathbf{x}|\theta)$ is the mapping function from d to k dimensions defined up to a set of parameters θ. Classical MDS we discussed previously corresponds to a linear transformation

$$\mathbf{z} = g(\mathbf{x}|\mathbf{W}) = \mathbf{W}^T\mathbf{x}$$

but in a general case, nonlinear mapping can also be used: this is called *Sammon mapping*. the normalized error in mapping is called the *Sammon stress* and is defined as

$$\begin{aligned} E(\theta|X) &= \sum_{r,s} \frac{(||\mathbf{z}^r - \mathbf{z}^s|| - ||\mathbf{x}^r - \mathbf{x}^s||)^2}{||\mathbf{x}^r - \mathbf{x}^s||^2} \\ &= \sum_{r,s} \frac{(||g(\mathbf{x}^r|\theta) - g(\mathbf{x}^s|\theta)|| - ||\mathbf{x}^r - \mathbf{x}^s||^2)}{||\mathbf{x}^r - \mathbf{x}^s||^2} \end{aligned}$$

In the case of classification, the class information can be included in the distance as

$$d'_{rs} = (1 - \alpha)d_{rs} + \alpha c_{rs}$$

where c_{rs} is the "distance" between the classes $\mathbf{x}^r$ and $\mathbf{x}^s$ belong to. This interclass distance should be supplied subjectively and α could be optimized using CV.

6.2.4 Locally Linear Embedding

Locally Linear Embedding (LLE) recovers global nonlinear structure from locally linear fits [25]. Its main idea is that each local patch of the manifold can be approximated linearly and given enough data, each point can be written as a linear, weighted sum of its neighbors.

The LLE algorithm is based on simple geometric intuitions. Suppose the data consists of N real-valued vectors $\mathbf{X_i}$, each of dimensionality D, sampled from some smooth underlying manifold. It is expected that each data point and its neighbors to lie on or close to a locally linear patch of the manifold. The local geometry of these patches can be characterized by linear coefficients that reconstruct each data point from its neighbors. In the simplest formulation of LLE, the KNN are estimated per data point, as measured by Euclidean distance. Reconstruction errors are then measured by the cost function:

$$\varepsilon(W) = \sum_i \left| \mathbf{X_i} - \sum_j W_{ij} \mathbf{X_j} \right|^2$$

which adds up the squared distances between all the data points and their reconstructions. The weights W_{ij} summarize the contribution of the jth data point to the $i$sth reconstruction. To compute the weights W_{ij}, it is necessary to minimize the cost function subject to two constraints: first, that each data point $\mathbf{X_i}$ is reconstructed only from its neighbors, enforcing $W_{ij} = 0$ if $\mathbf{X_j}$ does not belong to this set; second, that the rows of the weight matrix sum to one: $\sum_j W_{ij} = 1$ s. The optimal weights W_{ij} subject to these constraints are found by solving a least squares problem.

The constrained weights that minimize these reconstruction errors are invariant to rotations, scaling, and translations of that data point and its neighbors. Suppose the data lie on or near a smooth nonlinear manifold of dimensionality $d \ll D$. To achieve a good approximation, then, there exists a linear mapping that maps the high dimensional coordinates of each neighborhood to global internal coordinates on the manifold. By design, the reconstruction weights W_{ij} reflect intrinsic geometric properties of the data that are invariant to exactly such transformations. We therefore 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 ith data point in D dimensions should also reconstruct its embedded manifold coordinates in d dimensions.

LLE constructs a neighborhood preserving mapping based on the above idea. In the final step of the algorithm, each high dimensional observation $\mathbf{X_i}$ is mapped to a low dimensional vector $\mathbf{Y_i}$ representing global internal coordinates on the manifold. This is done by choosing d-dimensional coordinates $\mathbf{Y_i}$ to minimize the embedding cost function:

$$\Phi(Y) = \sum_i \left| \mathbf{Y_i} - \sum_j W_{ij} \mathbf{Y_j} \right|^2$$

This cost function, like the previous one, is based on locally linear reconstruction errors, but here, the weights W_{ij} are fixed while optimizing the coordinates $\mathbf{Y_i}$. Now, the embedding cost can be minimized by solving a sparse $N \times N$ eigenvector problem, whose bottom d non-zero eigenvectors provide an ordered set of orthogonal coordinates centered on the origin.

It is noteworthy that while the reconstruction weights for each data point are computed from its local neighborhood, the embedding coordinates are computed by an $N \times N$ eigensolver, a global operation that couples all data points in connected components of the graph defined by the weight matrix. The different dimensions in the embedding space can be computed successively; this is done simply by computing the bottom eigenvectors from previous equation one at a time. But the computation is always coupled across data points. This is how the algorithm leverages overlapping local information to discover global structure. Implementation of the algorithm is fairly straightforward, as the algorithm has only one free parameter: the number of neighbors per data point, K.

6.3 Data Sampling

Sampling is used to ease the analysis and modeling of large data sets. In DM, data sampling serves four purposes:

- *To reduce the number of instances submitted to the DM algorithm*. In many cases, predictive learning can operate with 10–20 % of cases without a significant deterioration of the performance. After that, the addition of more cases should have expected outcomes. However, in descriptive analysis, it is better to have as many cases as possible.
- *To support the selection of only those cases in which the response is relatively homogeneous*. When you have data sets where different trends are clearly observable or the examples can be easily separated, you can partition the data for different types of modelling. For instance, imagine the learning of the approving decision of bank loans depending on some economic characteristics of a set of customers.

If data includes consumer loans and mortgages, it seems logical to partition both types of loans because the parameters and quantities involved in each one are completely different. Thus, it is a good idea to build separate models on each partition.

- *To assist regarding the balance of data and occurrence of rare events*. Predictive DM algorithms like ANNs or decision trees are very sensitive to imbalanced data sets. An imbalanced data set is one in which one category of the target variable is less represented compared to the other ones and, usually, this category has is more important from the point of view of the learning task. Balancing the data involves sampling the imbalanced categories more than average (over-sampling) or sampling the common less often (under-sampling) [3].
- *To divide a data set into three data sets to carry out the subsequent analysis of DM algorithms*. As we have described in Chap. 2, the original data set can be divided into the training set and testing set. A third kind of division can be performed within the training set, to aid the DM algorithm to avoid model over-fitting, which is a very common strategy in ANNs and decision trees. This partition is usually known as validation set, although, in various sources, it may be denoted as the testing set interchangeably [22]. Whatever the nomenclature used, some learners require an internal testing process and, in order to evaluate and compare a set of algorithms, there must be an external testing set independent of training and containing unseen cases.

Various forms of data sampling are known in data reduction. Suppose that a large data set, T, contains N examples. The most common ways that we could sample T for data reduction are [11, 24]:

- **Simple random sample without replacement (SRSWOR) of size** s: This is created by drawing s of the N tuples from T ($s < N$), where the probability of drawing any tuple in T is $1/N$, that is, all examples have equal chance to be sampled.
- **Simple random sample with replacement (SRSWR) of size** s: This is similar to SRSWOR, except that each time a tuple is drawn from T, it is recorded and replaced. In other words, after an example is drawn, it is placed back in T and it may be drawn again.
- **Balanced sample**: The sample is designed according to a target variable and is forced to have a certain composition according to a predefined criterion. For example, 90 % of customers who are older tah or who are 21 years old, and 10 % of customers who are younger than 21 years old. One of the most successful application of this type of sampling has been shown in imbalanced learning, as we have mentioned before.
- **Cluster sample**: If the tuples in T are grouped into G mutually disjointed groups or clusters, then an SRS of s clusters can be obtained, where $s < G$. For example, in spatial data sets, we may choose to define clusters geographically based on how closely different areas are located.
- **Stratified sample**: If T is divided into mutually disjointed parts called *strata*, a stratified sample of T is generated by obtaining an SRS at each stratum. This

assists in ensuring a representative sample. It is frequently used in classification tasks where the class imbalance is present. It is very closely related with balanced sample, but the predefined composition of the final results depends on the natural distribution of the target variable.

An important preference of sampling for data reduction is that the cost of obtaining a sample is proportionate to the size of the sample s, instead of being proportionate to N. So, the sampling complexity is sub-linear to the size of data and there is no need to conduct a complete pass of T to make decisions in order to or not to include a certain example into the sampled subset. Nevertheless, the inclusion of examples are made by unfounded decisions, allowing redundant, irrelevant, noisy or harmful examples to be included. A smart way to make decisions for sampling is known as IS, a topic that we will extend in Chap. 8.

Advanced schemes of data sampling deserve to be described in this section. As before, they are more difficult and allow better adjustments of data according to the necessities and applications.

6.3.1 Data Condensation

The selection of a small representative subset from a very large data set is known as data condensation. In some sources of DM, such as [22], this form of data reduction is differentiated from others. In this book, data condensation is integrated as one of the families of IS methods (see Chap. 8).

Data condensation emerges from the fact that naive sampling methods, such as random sampling or stratified sampling, are not suitable for real-world problems with noisy data since the performance of the algorithms may change unpredictably and significantly. The data sampling approach practically ignores all the information present in the samples which are not chosen in the reduced subset.

Most of the data condensation approaches are studied on classification-based tasks, and in particular, for the KNN algorithm. These methods attempt to obtain a minimal consistent set, i.e., a minimal set which correctly classifies all the original examples. The very first method of this kind was the condensed nearest neighbor rule (CNN) [12]. For a survey on data condensation methods for classification, we again invite the reader to check the Chap. 8 of this book.

Regarding the data condensation methods which are not affiliated with classification tasks, termed generic data condensation, condensation is performed by vector quantization, such as the well-known self-organizing map [19] and different forms of data clustering. Another group of generic data condensation methods are situated on the density-based techniques which consider the density function of the data for the aspiration of condensation instead of minimizing the quantization error. These approaches do not concern any learning process and, hence, are deterministic, (i.e., for a concrete input data set, the output condensed set is established). Clear examples of this kind of approaches are presented in [10, 21].

6.3.2 Data Squashing

A data squashing method seeks to compress, or "squash", the data in such a way that a statistical analysis carried out on the compressed data obtains the same outcome that the one obtained with the original data set; that is, the statistical information is preserved.

The first approach of data squashing was proposed in [6] and termed DS, as a solution of constructing a reduced data set. DS approach to squashing is model-free and relies on moment-matching. The squashed data set consists of a set of artificial data points chosen to replicate the moments of the original data within subsets of the actual data. DS studies various approaches to partitioning and ordering the moments and also provides a theoretical justification of their method by considering a Taylor series expansion of an arbitrary likelihood function. Since this relies upon the moments of the data, it should work well for any application in which the likelihood is well-approximated by the first few terms of a Taylor series. In practice, it is only proven with logistic regression.

In [20], the authors proposed the "likelihood-based data squashing" (LDS). LDS is similar to DS because it first partitions the data set and then chooses artificial data points corresponding to each subset of the partition. Nevertheless, the algorithms differ in how they build the partition and how they build the artificial data points. The DS algorithm partitions the data along certain marginal quartiles, and then matches moments. The LDS algorithm partitions the data using a likelihood-based clustering and then selects artificial data points so as to mimic the target sampling or posterior distribution. Both algorithms yield artificial data points with associated weights. The usage of squashed data requires algorithms that can use these weights conveniently. LDS is slightly more general than DS because it is also prepared for ANN-based learning.

A subsequent approach described in [23] presents a form of data squashing based on empirical likelihood. This method re-weights a random sample of data to match certain expected values to the population. The benefits of this method are the reduction of optimization cost in terms of computational complexity and the interest in enhancing the performance of boosted random trees.

6.3.3 Data Clustering

Clustering algorithms partition the data examples into groups, or *clusters*, so that data samples within a cluster are "similar" to one another and different to data examples that belong to other clusters. The similarity is usually defined by means of how near the examples are in space, according to a distance function. The quality of a cluster could be measured as a function of the length of its diameter, which is the maximum distance between any two samples belonging to the cluster. The average distance of each object within the cluster to the centroid is an alternative measure of cluster

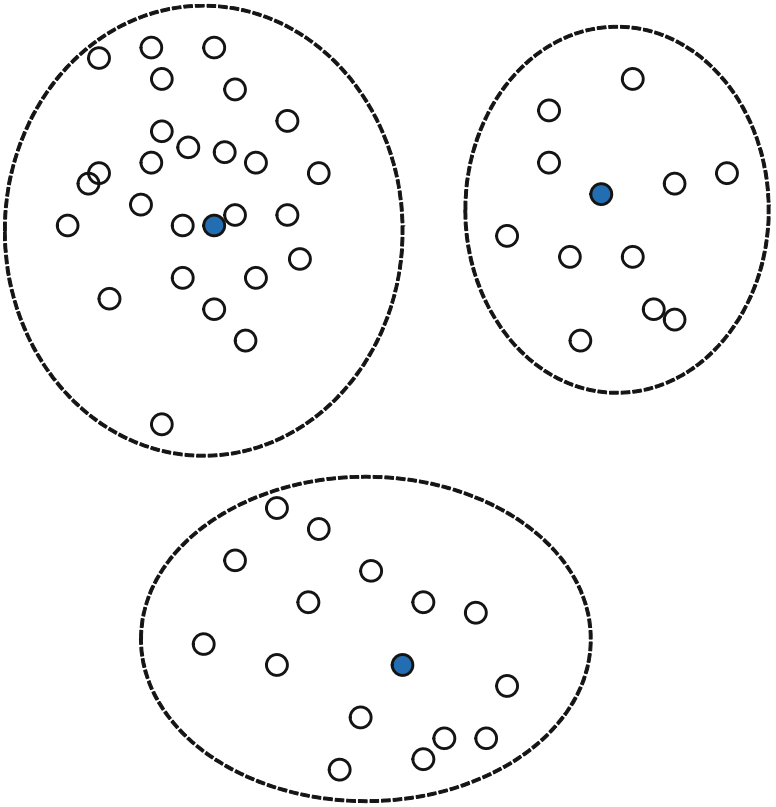

Fig. 6.3 Three clusters derived from a set of two-dimensional data

quality. An illustration of a three cluster derivation from a set of 2-D data points is depicted in Fig. 6.3.

In terms of data reduction, the cluster representations of the data are used instead of the actual data. In many applications, such as those in which data can be organized into distinct groups, this technique is higly effective.

There is a vast number of clustering techniques for defining clusters and for measuring their quality. In fact, clustering is surely the most popular and common form of unsupervised learning in DM, as we have mentioned in Chap. 1 of this book. For this reason, we have included it here due to the clear overlapping that clustering has with data reduction. Unfortunately, this book is not specifically devoted to learning and a deep study on clustering is beyond the scope of this book. However, the reader may consult the following references to an in-depth study: [1, 2, 9, 11, 15, 16, 27].

6.4 Binning and Reduction of Cardinality

Binning is the process of converting a continuous variable into a set of ranges. Then, each range can be treated as categories, with the choice of imposing order on them. This last choice is optional and depends on the further analysis to be made on the data. For example, we can bin the variable representing the annual income of a customer into ranges of 5,000 dollars (0–5,000; 5,001–10,000; 10,001–15,000, . . . , etc.). Such a binning could allow the analysis in a business problem may reveal that customers in the first range have less possibility to get a loan than customers in the last range, grouping them within an interval that bounds a numerical variable. Therefore, it demonstrates that keeping the strict order of bins is not always necessary.

Cardinality reduction of nominal and ordinal variables is the process of combining two or more categories into one new category. It is well known that nominal variables with a high number of categories are very problematic to handle. If we perform a transformation of these large cardinality variables onto indicator variables, that is, binary variables that indicate whether or not a category is set for each example; we will produce a large number of new variables, almost all equal to zero. On the other hand, if we do not perform this conversion and use them just as they are in with the algorithm that can tolerate them, such as decision trees, we run into the problem of over-fitting the model. It is realistic to consider reducing the number of categories in such variables.

Both processes are two common transformations used to achieve two objectives:

- Reduce the complexity of independent and possible dependent variables.
- Improve the predictive power of the variable, by carefully binning or grouping the categories in such a way that we model the dependencies regarding the target variable in both estimation and classification problems.

Binning and cardinality reduction are very similar procedures, differing only in the type of variable that we want to process. In fact, both processes are distinctively grouped within the term **discretization**, which constitutes the most popular notation in the literature. It is also very common to distinguish between binning and discretization depending on the ease of the process performed. Binning is usually associated with a quick and easy discretization of a variable. In [11], the authors distinguish among three types of discretization: binning, histogram analysis-based and advanced discretization. The first corresponds to a splitting technique based on the specification of the number of bins. The second family is related with unsupervised discretization and finally, a brief inspection of the rest of the methods is drawn.

Regardless of the above, and under the *discretization* nomenclature, we will discuss all related issues and techniques in Chap. 9 of this book.

References

1. Aggarwal, C., Reddy, C.: Data clustering: recent advances and applications. Chapman and Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis Group, Boca Raton (2013)
2. Aggarwal, C.C., Reddy, C.K. (eds.): Data Clustering: Algorithms and Applications. CRC Press, New York (2014)
3. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. Newsl. **6**(1), 20–29 (2004)
4. Bellman, R.E.: Adaptive control processes—a guided tour. Princeton University Press, Princeton (1961)
5. Chatfield, C., Collins, A.J.: Introduction to Multivariate Analysis. Chapman and Hall, London (1980)
6. DuMouchel, W., Volinsky, C., Johnson, T., Cortes, C., Pregibon, D.: Squashing flat files flatter. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '99, pp. 6–15 (1999)
7. Dunteman, G.: Principal Components Analysis. SAGE Publications, Newbury Park (1989)
8. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. Academic Press Professional, Inc., San Diego (1990)
9. Gan, G., Ma, C., Wu, J.: Data Clustering—Theory, Algorithms, and Applications. SIAM, Philadelphia (2007)
10. Girolami, M., He, C.: Probability density estimation from optimally condensed data samples. IEEE Trans. Pattern Anal. Mach. Intell. **25**(10), 1253–1264 (2003)
11. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco (2011)
12. Hart, P.E.: The condensed nearest neighbor rule. IEEE Trans. Inf. Theory **14**, 515–516 (1968)
13. Hwang, J., Lay, S., Lippman, A.: Nonparametric multivariate density estimation: a comparative study. IEEE Trans. Signal Process. **42**, 2795–2810 (1994)
14. Jain, A., Zongker, D.: Feature selection: evaluation, application, and small sample performance. IEEE Trans. Pattern Anal. Mach. Intell. **19**(2), 153–158 (1997)
15. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Comput. Surv. **31**(3), 264–323 (1999)
16. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recogn. Lett. **31**(8), 651–666 (2010)
17. Johnson, R.A., Wichern, D.W.: Applied Multivariate Statistical Analysis. Prentice-Hall, Englewood Cliffs (2001)
18. Kim, J.O., Mueller, C.W.: Factor Analysis: Statistical Methods and Practical Issues (Quantitative Applications in the Social Sciences). Sage Publications, Inc, Beverly Hills (1978)
19. Kohonen, T.: The self organizing map. Proc. IEEE **78**(9), 1464–1480 (1990)
20. Madigan, D., Raghavan, N., DuMouchel, W., Nason, M., Posse, C., Ridgeway, G.: Likelihood-based data squashing: a modeling approach to instance construction. Data Min. Knowl. Disc. **6**(2), 173–190 (2002)
21. Mitra, P., Murthy, C.A., Pal, S.K.: Density-based multiscale data condensation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(6), 734–747 (2002)
22. Nisbet, R., Elder, J., Miner, G.: Handbook of Statistical Analysis and Data Mining Applications. Academic Press, Boston (2009)
23. Owen, A.: Data squashing by empirical likelihood. Data Min. Knowl. Disc. **7**, 101–113 (2003)
24. Refaat, M.: Data Preparation for Data Mining Using SAS. Morgan Kaufmann Publishers Inc., San Francisco (2007)
25. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
26. Tenenbaum, J.B., Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
27. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. Neural Networks **16**(3), 645–678 (2005)