3

DATA REDUCTION

Chapter Objectives

- Identify the differences in dimensionality reduction based on features, cases, and reduction of value techniques.
- Explain the advantages of data reduction in the preprocessing phase of a data-mining process.
- Understand the basic principles of feature-selection and feature-composition tasks using corresponding statistical methods.
- Apply and compare entropy-based technique and principal component analysis (PCA) for feature ranking.
- Understand the basic principles and implement ChiMerge and bin-based techniques for reduction of discrete values.
- Distinguish approaches in cases where reduction is based on incremental and average samples.

For small or moderate data sets, the preprocessing steps mentioned in the previous chapter in preparation for data mining are usually enough. For really large data sets,

Data Mining: Concepts, Models, Methods, and Algorithms, Second Edition. Mehmed Kantardzic.

there is an increased likelihood that an intermediate, additional step—data reduction—should be performed prior to applying the data-mining techniques. While large data sets have the potential for better mining results, there is no guarantee that they will yield better knowledge than small data sets. Given multidimensional data, a central question is whether it can be determined, prior to searching for all data-mining solutions in all dimensions, that the method has exhausted its potential for mining and discovery in a reduced data set. More commonly, a general solution may be deduced from a subset of available features or cases, and it will remain the same even when the search space is enlarged.

The main theme for simplifying the data in this step is dimension reduction, and the main question is whether some of these prepared and preprocessed data can be discarded without sacrificing the quality of results. There is one additional question about techniques for data reduction: Can the prepared data be reviewed and a subset found in a reasonable amount of time and space? If the complexity of algorithms for data reduction increases exponentially, then there is little to gain in reducing dimensions in big data. In this chapter, we will present basic and relatively efficient techniques for dimension reduction applicable to different data-mining problems.

3.1 DIMENSIONS OF LARGE DATA SETS

The choice of data representation and selection, reduction, or transformation of features is probably the most important issue that determines the quality of a data-mining solution. Besides influencing the nature of a data-mining algorithm, it can determine whether the problem is solvable at all, or how powerful the resulting model of data mining is. A large number of features can make available samples of data relatively insufficient for mining. In practice, the number of features can be as many as several hundred. If we have only a few hundred samples for analysis, dimensionality reduction is required in order for any reliable model to be mined or to be of any practical use. On the other hand, data overload, because of high dimensionality, can make some data-mining algorithms non-applicable, and the only solution is again a reduction of data dimensions. For example, a typical classification task is to separate healthy patients from cancer patients, based on their gene expression "profile." Usually fewer than 100 samples (patients' records) are available altogether for training and testing. But the number of features in the raw data ranges from 6000 to 60,000. Some initial filtering usually brings the number of features to a few thousand; still it is a huge number and additional reduction is necessary. The three main dimensions of preprocessed data sets, usually represented in the form of flat files, are *columns* (features), *rows* (cases or samples), and *values* of the features.

Therefore, the three basic operations in a data-reduction process are delete a column, delete a row, and reduce the number of values in a column (smooth a feature). These operations attempt to preserve the character of the original data by deleting data that are nonessential. There are other operations that reduce dimensions, but the new data are unrecognizable when compared with the original data set, and these operations are mentioned here just briefly because they are highly application-dependent. One

approach is the replacement of a set of initial features with a new composite feature. For example, if samples in a data set have two features, person height and person weight, it is possible for some applications in the medical domain to replace these two features with only one, body mass index, which is proportional to the quotient of the initial two features. Final reduction of data does not reduce the quality of results; in some applications, the results of data mining are even improved.

In performing standard data-reduction operations (deleting rows, columns, or values) as a preparation for data mining, we need to know what we gain and/or lose with these activities. The overall comparison involves the following parameters for analysis:

1. *Computing Time.* Simpler data, a result of the data-reduction process, can hopefully lead to a reduction in the time taken for data mining. In most cases, we cannot afford to spend too much time on the data-preprocessing phases, including a reduction of data dimensions, although the more time we spend in preparation the better the outcome.
2. *Predictive/Descriptive Accuracy.* This is the dominant measure for most data-mining models since it measures how well the data are summarized and generalized into the model. We generally expect that by using only relevant features, a data-mining algorithm can learn not only faster but also with higher accuracy. Irrelevant data may mislead a learning process and a final model, while redundant data may complicate the task of learning and cause unexpected data-mining results.
3. *Representation of the Data-Mining Model.* The simplicity of representation, obtained usually with data reduction, often implies that a model can be better understood. The simplicity of the induced model and other results depends on its representation. Therefore, if the simplicity of representation improves, a relatively small decrease in accuracy may be tolerable. The need for a balanced view between accuracy and simplicity is necessary, and dimensionality reduction is one of the mechanisms for obtaining this balance.

It would be ideal if we could achieve reduced time, improved accuracy, and simplified representation at the same time, using dimensionality reduction. More often than not, however, we gain in some and lose in others, and balance between them according to the application at hand. It is well known that no single data-reduction method can be best suited for all applications. A decision about method selection is based on available knowledge about an application (relevant data, noise data, meta-data, correlated features), and required time constraints for the final solution.

Algorithms that perform all basic operations for data reduction are not simple, especially when they are applied to large data sets. Therefore, it is useful to enumerate the desired properties of these algorithms before giving their detailed descriptions. Recommended characteristics of data-reduction algorithms that may be guidelines for designers of these techniques are as follows:

1. *Measurable Quality.* The quality of approximated results using a reduced data set can be determined precisely.

2. *Recognizable Quality.* The quality of approximated results can be easily determined at run time of the data-reduction algorithm, before application of any data-mining procedure.
3. *Monotonicity.* The algorithms are usually iterative, and the quality of results is a nondecreasing function of time and input data quality.
4. *Consistency.* The quality of results is correlated with computation time and input data quality.
5. *Diminishing Returns.* The improvement in the solution is large in the early stages (iterations) of the computation, and it diminishes over time.
6. *Interruptability.* The algorithm can be stopped at any time and provide some answers.
7. *Preemptability.* The algorithm can be suspended and resumed with minimal overhead.

3.2 FEATURE REDUCTION

Most of the real-world data mining applications are characterized by high-dimensional data, where not all of the features are important. For example, high-dimensional data (i.e., data sets with hundreds or even thousands of features) can contain a lot of irrelevant, noisy information that may greatly degrade the performance of a data-mining process. Even state-of-the-art data-mining algorithms cannot overcome the presence of a large number of weakly relevant and redundant features. This is usually attributed to the "curse of dimensionality," or to the fact that irrelevant features decrease the signal-to-noise ratio. In addition, many algorithms become computationally intractable when the dimensionality is high.

Data such as images, text, and multimedia are high-dimensional in nature, and this dimensionality of data poses a challenge to data-mining tasks. Researchers have found that reducing the dimensionality of data results in a faster computation while maintaining reasonable accuracy. In the presence of many irrelevant features, mining algorithms tend to overfit the model. Therefore, many features can be removed without performance deterioration in the mining process.

When we are talking about data quality and improved performances of reduced data sets, we can see that this issue is not only about noisy or contaminated data (problems mainly solved in the preprocessing phase), but also about irrelevant, correlated, and redundant data. Recall that data with corresponding features are not usually collected solely for data-mining purposes. Therefore, dealing with relevant features alone can be far more effective and efficient. Basically, we want to choose features that are relevant to our data-mining application in order to achieve maximum performance with minimum measurement and processing effort. A feature-reduction process should result in

1. less data so that the data-mining algorithm can learn faster;
2. higher accuracy of a data-mining process so that the model can generalize better from the data;

3. simple results of the data-mining process so that they are easier to understand and use; and
4. fewer features so that in the next round of data collection, savings can be made by removing redundant or irrelevant features.

Let us start our detailed analysis of possible column-reduction techniques, where features are eliminated from the data set based on a given criterion. To address the curse of dimensionality, dimensionality-reduction techniques are proposed as a data-preprocessing step. This process identifies a suitable low-dimensional representation of original data. Reducing the dimensionality improves the computational efficiency and accuracy of the data analysis. Also, it improves comprehensibility of a data-mining model. Proposed techniques are classified as supervised and unsupervised techniques based on the type of learning process. Supervised algorithms need a training set with the output class label information to learn the lower dimensional representation according to some criteria. Unsupervised approaches project the original data to a new lower dimensional space without utilizing the label (class) information. Dimensionality-reduction techniques function either by transforming the existing features to a new reduced set of features or by selecting a subset of the existing features. Therefore, two standard tasks are associated with producing a reduced set of features, and they are classified as:

1. *Feature Selection.* Based on the knowledge of the application domain and the goals of the mining effort, the human analyst may select a subset of the features found in the initial data set. The process of feature selection can be manual or supported by some automated procedures.

   Roughly speaking, feature-selection methods are applied in one of three conceptual frameworks: the *filter model*, the *wrapper model*, and *embedded methods*. These three basic families differ in how the learning algorithm is incorporated in evaluating and selecting features. In the filter model the selection of features is done as a preprocessing activity, without trying to optimize the performance of any specific data-mining technique directly. This is usually achieved through an (ad hoc) evaluation function using a search method in order to select a subset of features that maximizes this function. Performing an exhaustive search is usually intractable due to the large number of initial features. Therefore, different methods may apply a variety of search heuristics. Wrapper methods select features by "wrapping" the search around the selected learning algorithm and evaluate feature subsets based on the learning performance of the data-mining technique for each candidate feature subset. The main drawback of this approach is its computational complexity. Main characteristics of both approaches are given in Figure 3.1. Finally, embedded methods incorporate feature search and the learning algorithm into a single optimization problem formulation. When the number of samples and dimensions becomes very large, the filter approach is usually a choice due to its computational efficiency and neutral bias toward any learning methodology.

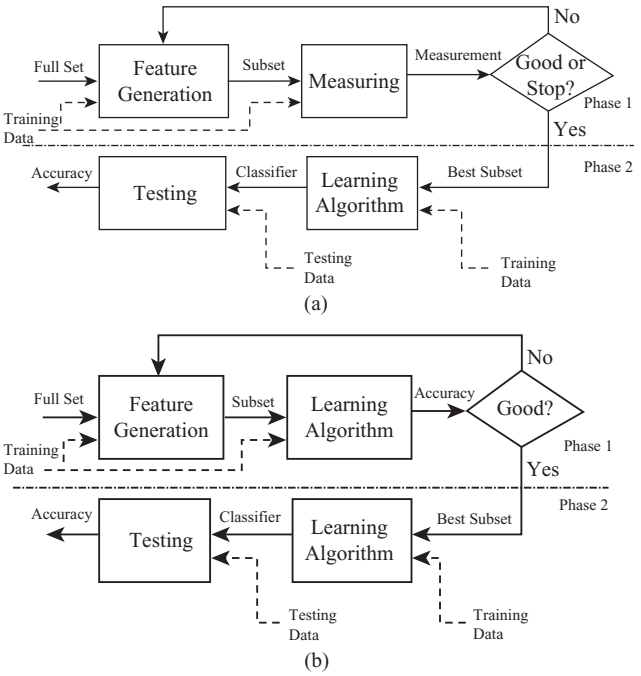

Figure 3.1. Feature-selection approaches. (a) Filter model; (b) wrapper model.

2. *Feature Extraction/Transformation.* There are transformations of data that can have a surprisingly strong impact on the results of data-mining methods. In this sense, the composition and/or transformation of features is a greater determining factor in the quality of data-mining results. In most instances, feature composition is dependent on knowledge of the application, and an interdisciplinary approach to feature composition tasks produces significant improvements in the preparation of data. Still, some general purpose techniques, such as PCA, are often used and with great success.

   Feature selection is typically preferred over extraction/transformation when one wishes to keep the original meaning of the features and wishes to determine which of those features are important. Moreover, once features are selected, only those features need to be calculated or collected, whereas in transformation-based methods all input features are still needed to obtain the reduced dimension.

3.2.1 Feature Selection

In data mining, feature selection, also known as variable selection, feature reduction, attribute selection, or variable subset selection, is a set of techniques that select a subset of relevant features for building robust learning models by removing most irrelevant

and redundant features from the data. The objective of feature selection is threefold: improving the performance of a data-mining model, providing a faster and more cost-effective learning process, and providing a better understanding of the underlying process that generates the data. Feature-selection algorithms typically fall into two categories: *feature ranking* and *subset selection*. Feature ranking ranks all features by a specified metric and eliminates all features that do not achieve an adequate score. Subset selection, on the other hand, searches the set of all features for the optimal subset where features in the selected subset are not ranked. We have to be aware that different feature-selection methods may give different reduced feature sets.

In the feature-ranking algorithm, one can expect a ranked list of features that are ordered according to a specific evaluation measure. A measure can be based on accuracy of available data, consistency, information content, distances between samples, and finally, statistical dependencies between features. These algorithms do not tell you what the minimum set of features for further analysis is; they indicate only the relevance of a feature compared with others. Minimum-subset algorithms, on the other hand, return a minimum feature subset and no differences are made among features in the subset—all have the same ranking. The features in the subset are relevant for the mining process; the others are irrelevant. In both types of algorithms, it is important to establish a feature-evaluation scheme: the way in which the features are evaluated and then ranked, or added to the selected subset.

Feature selection in general can be viewed as a search problem, with each state in the search space specifying a subset of the possible features. If, for example, a data set has three features, $\{A_1, A_2, A_3\}$, and in the process of selecting features, the presence of a feature is coded with 1 and its absence with 0, then there should be a total of 2^3 reduced-feature subsets coded with {0, 0, 0}, {1, 0, 0}, {0, 1, 0}, {0, 0, 1}, {1, 1, 0}, {1, 0, 1}, {0, 1, 1}, and {1, 1, 1}. The problem of feature selection is relatively trivial if the search space is small, since we can analyze all subsets in any order and a search will get completed in a short time. However, the search space is usually not small. It is 2^N where the number of dimensions N in typical data-mining applications is large ($N > 20$). This makes the starting point and the search strategy very important. An exhaustive search of all subsets of features very often is replaced with some heuristic search procedures. With knowledge of the problem, these procedures find near-optimal subsets of features that further improve the quality of the data-mining process.

The objective of feature selection is to find a subset of features with data-mining performances comparable to the full set of features. Given a set of features m, the number of subsets to be evaluated as candidates for column reduction is finite, but still very large for iterative analysis through all cases. For practical reasons, an optimal search is not feasible, and simplifications are made to produce reasonable, acceptable, and timely results. If the reduction task is to create a subset, one possibility—the so-called bottom-up approach—starts with an empty set, and fills it in by choosing the most relevant features from the initial set of features. This process is based on some heuristic criteria for a feature evaluation. The top-down approach, on the other hand, begins with a full set of original features and then removes one-by-one those that are shown as irrelevant based on the selected heuristic evaluation measure. Additional approximations to the optimal approach are

1. to examine only promising subsets of features where promising subsets are usually obtained heuristically—this provides enough room for exploration of competing alternatives;
2. to substitute computationally simple distance measures for the error measures—this approximation will reduce the computation time yet give satisfactory results for comparison of subset alternatives;
3. to select features based only on subsets of large amounts of data, but the subsequent steps of data mining will be applied on the whole set.

The application of feature selection and reduction of data dimensionality may be used in all phases of the data-mining process for successful knowledge discovery. It has to be started in the preprocessing phase, but, on many occasions, feature selection and reduction is a part of the data-mining algorithm, even if it is applied in postprocessing for better evaluation and consolidation of obtained results.

Let us return to the promising subsets of features. One possible technique for feature selection is based on comparison of *means* and *variances*. To summarize the key characteristics of the distribution of values for a given feature, it is necessary to compute the mean value and the corresponding variance. The main weakness in this approach is that the distribution for the feature is not known. If it is assumed to be a normal curve, the statistics can work out very well, but this may in fact be a poor assumption. Without knowing the shape of the distribution curve, the means and variances are viewed as heuristics for feature selection, not exact, mathematical modeling tools.

In general, if one feature describes different classes of entities, samples of two different classes can be examined. The means of feature values are normalized by their variances and then compared. If the means are far apart, interest in a feature increases; it has potential, in terms of its use in distinguishing between two classes. If the means are indistinguishable, interest wanes in that feature. It is a heuristic, nonoptimal approach to feature selection, but it is consistent with practical experience in many data-mining applications in the triage of features. Next, equations formalize the test, where A and B are sets of feature values measured for two different classes, and n_1 and n_2 are the corresponding number of samples:

$$\mathrm{SE(A\text{-}B)} = \sqrt{(\mathrm{var(A)}/n_1 + \mathrm{var(B)}/n_2)}$$

$$\mathrm{TEST}: |\,\mathrm{mean\ (A)} - \mathrm{mean\ (B)}\,| / \mathrm{SE(A\text{-}B)} > \mathrm{threshold\ value}$$

The mean of a feature is compared in both classes without taking into consideration relationship to other features. In this approach to feature selection, we assumed a priori that the given feature is independent of the others. A comparison of means is a natural fit to classification problems. For the purposes of feature selection, a regression problem can be considered as a pseudo-classification problem. For k classes, k pairwise comparisons can be made, comparing each class with its complement. A feature is retained if it is significant for any of the pairwise comparisons.

We can analyze this approach in feature selection through one example. A simple data set is given in Table 3.1 with two input features X and Y, and an additional output

TABLE 3.1. Dataset with Three Features

X	Y	C
0.3	0.7	A
0.2	0.9	B
0.6	0.6	A
0.5	0.5	A
0.7	0.7	B
0.4	0.9	B

feature C that classifies samples into two classes, A and B. It is necessary to decide whether the features X and Y are candidates for reduction or not. Suppose that the threshold value of the applied test is 0.5.

First, we need to compute a mean value and a variance for both classes and both features X and Y. The analyzed subsets of the feature's values are

$$X_A = \{0.3, 0.6, 0.5\}, X_B = \{0.2, 0.7, 0.4\}, Y_A = \{0.7, 0.6, 0.5\}, \text{and } Y_B = \{0.9, 0.7, 0.9\}$$

and the results of applied tests are

$$SE(X_A\text{-}X_B) = \sqrt{(var(X_A)/n_1 + var(X_B)/n_2)} = \sqrt{(0.0233/3 + 0.06333/3)} = 0.1699$$

$$SE(Y_A\text{-}Y_B) = \sqrt{(var(Y_A)/n_1 + var(Y_B)/n_2)} = \sqrt{(0.01/3 + 0.0133/3)} = 0.0875$$

$$|mean(X_A) - mean(X_B)|/SE(X_A\text{-}X_B) = |0.4667 - 0.4333|/0.1699 = 0.1961 < 0.5$$

$$|mean(Y_A) - mean(Y_B)|/SE(Y_A\text{-}Y_B) = |0.6 - 0.8333|/0.0875 = 2.6667 > 0.5$$

This analysis shows that X is a candidate for reduction because its mean values are close and, therefore, the final test is below the threshold value. On the other hand, the test for feature Y is significantly above the threshold value; this feature is not a candidate for reduction because it has the potential to be a distinguishing feature between two classes.

A similar idea for feature ranking is shown in the algorithm that is based on correlation criteria. Let us consider first the prediction of a continuous outcome y. The Pearson correlation coefficient is defined as:

$$R(i) = \frac{cvo(X_i, Y)}{\sqrt{var(X_i)var(Y)}}$$

where *cov* designates the covariance and *var* the variance. The estimate of $R(i)$ for the given data set with samples' inputs $x_{k,j}$ and outputs y_k is defined by:

$$R(i) = \frac{\sum_{k=1}^{m}(x_{k,i} - \overline{x}_i)(y_k - \overline{y})}{\sqrt{\sum_{k=1}^{m}(x_{k,i} - \overline{x}_i)^2 \sum_{k=1}^{m}(y_k - \overline{y})^2}}$$

where the bar notation stands for an average over the index k (set of all samples). Using $R(i)^2$ as a variable-ranking criterion enforces a ranking according to goodness of linear fit of individual variables. Correlation criteria such as $R(i)^2$ can only detect linear dependencies between input features and target or output feature (variable). One common criticism of variable ranking is that it leads to the selection of a redundant subset. The same performance could possibly be achieved with a smaller subset of complementary variables. Still, one may wonder whether deleting presumably redundant variables can result in a performance gain.

Practical experiments show that noise reduction and consequently better model estimation may be obtained by adding features that are presumably redundant. Therefore, we have to be very careful in the preprocessing analysis. Yes, perfectly correlated variables are truly redundant in the sense that no additional information is gained by adding them. But, even variables with relatively high correlation (or anti-correlation) do not guarantee absence of variables' complementarity. We can find cases where a feature looks completely useless by itself, and it is ranked very low, but it can provide significant information to the model and performance improvement when taken with others. These features by themselves may have little correlation with the output, target concept, but when combined with some other features, they can be strongly correlated with the target feature. Unintentional removal of these features can result in poor mining performance.

The previous simple methods test features separately. Several features may be useful when considered separately, but they may be redundant in their predictive ability. If the features are examined collectively, instead of independently, additional information can be obtained about their characteristics and mutual relations. Assuming normal distributions of values, it is possible to describe an efficient technique for selecting subsets of features. Two descriptors characterize a multivariate normal distribution:

1. M, a vector of the m feature means, and
2. C, an m × m covariance matrix of the means, where $C_{i,i}$ are simply the variance of feature i, and $C_{i,j}$ terms are correlations between each pair of features

$$C_{i,j} = 1/n \sum_{k=1}^{n} (((v(k,i) - m(i)) * (v(k,j) - m(j)))$$

where

v(k,i) and v(k,j) are the values of features indexed with i and j,
m(i) and m(j) are feature means, and
n is the number of dimensions.

These two descriptors, M and C, provide a basis for detecting redundancies in a set of features. If two classes exist in a data set, then the heuristic measure, DM, for filtering features that separate the two classes is defined as

$$DM = (M_1 - M_2)(C_1 + C_2)^{-1}(M_1 - M_2)^T$$

where M_1 and C_1 are descriptors of samples for the first class, and M_2 and C_2 for the second class. Given the target of k best features, all subsets of k from m features must be evaluated to find the subset with the largest DM. With large data sets that have large numbers of features, this can be a huge search space, and alternative heuristic methods should be considered. One of these methods selects and ranks features based on an entropy measure. Detailed explanations are given in Section 3.4. The other heuristic approach, explained in the following, is based on a combined correlation and covariance analyses and ranking of all features.

Existing efficient feature-selection algorithms usually rank features under the assumption of feature independence. Under this framework, features are ranked as relevant mainly based on their individual high correlations with the output feature. Because of the irreducible nature of feature interactions, these algorithms cannot select interacting features. In principle, it can be shown that a feature is relevant due to two reasons: (1) It is strongly correlated with the target feature, or (2) it forms a feature subset and the subset is strongly correlated with the target. A heuristic approach is developed to analyze features of type (2) in the selection process.

In the first part of a selection process, the features are ranked in descending order based on their correlation values with output using a previously defined technique. We may assume that a set of features S can be divided into subset S1 including relevant features, and subset S2 containing irrelevant ones. Heuristically, critical for removal are features in S2 first, while features in S1 are more likely to remain in the final set of selected features.

In the second part, features are evaluated one by one starting from the end of the S2 ranked feature list. The monotonic property suggests that the backward elimination search strategy fits best in feature selection. That is, one can start with the full feature set and successively eliminate features one at a time from the bottom of the ranked list if their interaction does not contribute to better correlation with output. The criterion could be, for example, based on a covariance matrix. If a feature, together with previously selected features, shows influence on the output with less than threshold value (it could be expressed through some covariance matrix factor), the feature is removed; otherwise it is selected. Backward elimination allows every feature to be evaluated with the features it may interact with. The algorithm repeats until all features in the S2 list are checked.

3.2.2 Feature Extraction

The art of data mining starts with the design of appropriate data representations. Better performance is often achieved using features derived from the original input. Building a feature representation is an opportunity to incorporate domain knowledge into the data and can be very application-specific. Transforming the input set into a new, reduced set of features is called *feature extraction*. If the features extracted are carefully chosen, it is expected that the new feature set will extract the relevant information from the input data in order to perform the desired data-mining task using this reduced representation.

Feature-transformation techniques aim to reduce the dimensionality of data to a small number of dimensions that are linear or nonlinear combinations of the original dimensions. Therefore, we distinguish two major types of dimension-reduction methods: linear and nonlinear. Linear techniques result in *k* new derived features instead of initial p (k << p). Components of the new feature are a linear combination of the original features:

$$s_i = w_{i,1}x_1 + \ldots + w_{i,p}x_p \qquad \text{for } i = 1, \ldots, k$$

or in a matrix form

$$s = W\,x$$

where $W_{k \times p}$ is the linear transformation weight matrix. Such linear techniques are simpler and easier to implement than more recent methods considering nonlinear transforms.

In general, the process reduces feature dimensions by combining features instead of by deleting them. It results in a new set of fewer features with totally new values. One well-known approach is merging features by *principal components*. The features are examined collectively, merged, and transformed into a new set of features that, it is hoped, will retain the original information content in a reduced form. The basic transformation is linear. Given p features, they can be transformed into a single new feature F′ by the simple application of weights:

$$F' = \sum_{j=1}^{p} w(j) \cdot f(j)$$

Most likely, a single set of weights, w(j), will not be an adequate transformation for a complex multidimensional data set, and up to p transformations are generated. Each vector of p features combined by weights w is called a principal component, and it defines a new transformed feature. The first vector of m weights is expected to be the strongest, and the remaining vectors are ranked according to their expected usefulness in reconstructing the original data. Eliminating the bottom-ranked transformation will reduce dimensions. The complexity of computation increases significantly with the number of features. The main weakness of the method is that it makes an advance assumption to a linear model that maximizes the variance of features. Formalization of PCA and the basic steps of the corresponding algorithm for selecting features are given in Section 3.4.

Examples of additional methodologies in feature extraction include factor analysis (FA), independent component analysis (ICA), and multidimensional scaling (MDS). Probably the last one is the most popular and it represents the basis for some new, recently developed techniques. Given n samples in a p-dimensional space and an $n \times n$ matrix of distances measures among the samples, MDS produces a k-dimensional (k << p) representation of the items such that the distances among the points in the new space reflect the distances in the original data. A variety of distance measures may be

used in the technique and the main characteristics for all these measures is: The more similar two samples are, the smaller their distance is. Popular distance measures include the Euclidean distance (L2 norm), the Manhattan distance (L1, absolute norm), and the maximum norm; more details about these measures and their applications are given in Chapter 9. MDS has been typically used to transform the data into two or three dimensions; visualizing the result to uncover hidden structure in the data. A rule of thumb to determine the maximum number of k is to ensure that there are at least twice as many pairs of samples than the number of parameters to be estimated, resulting in $p \geq k + 1$. Results of the MDS technique are indeterminate with respect to translation, rotation, and reflection of data.

PCA and metric MDS are both simple methods for linear dimensionality reduction, where an alternative to MDS is FastMap, a computationally efficient algorithm. The other variant, *Isomap*, has recently emerged as a powerful technique for nonlinear dimensionality reduction and is primarily a graph-based method.

Isomap is based on computing the low-dimensional representation of a high-dimensional data set that most faithfully preserves the pairwise distances between input samples as measured along geodesic distance (details about geodesic are given in Chapter 12, the section about graph mining). The algorithm can be understood as a variant of MDS in which estimates of geodesic distances are substituted for standard Euclidean distances.

The algorithm has three steps. The first step is to compute the k-nearest neighbors of each input sample, and to construct a graph whose vertices represent input samples and whose (undirected) edges connect k-nearest neighbors. The edges are then assigned weights based on the Euclidean distance between nearest neighbors. The second step is to compute the pairwise distances between all nodes (i, j) along shortest paths through the graph. This can be done using the well-known Djikstra's algorithm with complexity $O(n^2 \log n + n^2 k)$. Finally, in the third step, the pairwise distances are fed as input to MDS to determine a new reduced set of features.

With the amount of data growing larger and larger, all feature-selection (and reduction) methods also face a problem of oversized data because of computers' limited resources. But do we really need so much data for selecting features as an initial process in data mining? Or can we settle for less data? We know that some portion of a huge data set can represent it reasonably well. The point is which portion and how large should it be. Instead of looking for the right portion, we can randomly select a part, P, of a data set, use that portion to find the subset of features that satisfy the evaluation criteria, and test this subset on a different part of the data. The results of this test will show whether the task has been successfully accomplished. If an inconsistency is found, we shall have to repeat the process with a slightly enlarged portion of the initial data set. What should be the initial size of the data subset P? Intuitively, we know that its size should not be too small or too large. A simple way to get out of this dilemma is to choose a percentage of data, say 10%. The right percentage can be determined experimentally.

What are the results of a feature-reduction process, and why do we need this process for every specific application? The purposes vary, depending upon the problem on hand, but, generally, we want

1. to *improve performances* of the model-generation process and the resulting model itself (typical criteria are speed of learning, predictive accuracy, and simplicity of the model);
2. to *reduce dimensionality* of the model without reduction of its quality through
   (a) elimination of irrelevant features,
   (b) detection and elimination of redundant data and features,
   (c) identification of highly correlated features, and
   (d) extraction of independent features that determine the model; and
3. to help the user *visualize* alternative results, which have fewer dimensions, to improve decision making.

3.3 RELIEF ALGORITHM

Reliefis a feature weight-based algorithm for feature selection inspired by the so-called instance-based learning. It relies on relevance evaluation of each feature given in a training data set, where samples are labeled (classification problems). The main idea of Relief is to compute a ranking score for every feature indicating how well this feature separates neighboring samples. The authors of the Relief algorithm, Kira and Rendell, proved that the ranking score is large for relevant features and small for irrelevant ones.

The core of the Relief algorithm is to estimate the quality of features according to how well their values distinguish between samples close to each other. Given training data S, the algorithm randomly selects subset of samples size m, where m is a user-defined parameter. Relief analyzes each feature based on a selected subset of samples. For each randomly selected sample X from a training data set, Relief searches for its two nearest neighbors: one from the same class, called nearest hit H, and the other one

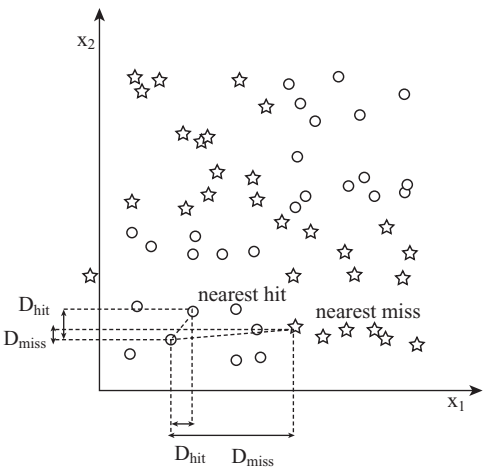

Figure 3.2. Determining nearest hit H and nearest miss M samples.

from a different class, called nearest miss M. An example for two-dimensional data is given in Figure 3.2.

The Relief algorithm updates the quality score $W(A_i)$ for all feature A_i depending on the differences on their values for samples X, M, and H:

$$W_{new}(A_i) = W_{old}(A_i) - (\text{diff}(X[A_i], H[A_i])^2 + \text{diff}(X[A_i], M[A_i])^2)/m$$

The process is repeated m times for randomly selected samples from the training data set and the scores $W(A_i)$ are accumulated for each sample. Finally, using threshold of relevancy τ, the algorithm detects those features that are statistically relevant to the target classification, and these are the features with $W(A_i) \geq \tau$. We assume the scale of every feature is either nominal (including Boolean) or numerical (integer or real). The main steps of the Relief algorithm may be formalized as follows:

Initialize: $W(A_j) = 0$; i = 1, . . . , p (p is the number of features)

For i = 1 to m

Randomly select sample X from training data set S.
Find nearest hit H and nearest miss M samples.

For j = 1 to p

$$W(A_j) = W(A_j) - (\text{diff}(X[A_j], H[A_j])^2 + \text{diff}(X[A_j], M[A_j])^2)/m$$

End.

End.

Output: Subset of feature where $W(A_j) \geq \tau$

For example, if the available training set is given in Table 3.2 with three features (the last one of them is classification decision) and four samples, the scoring values W for the features F_1 and F_2 may be calculated using Relief:

$$W(F_1) = (0 + [-1 + 4] + [-1 + 9] + [-1 + 9] + [-1 + 4])/4 = 5.5$$
$$W(F_2) = (0 + [-1 + 4] + [-1 + 1] + [-1 + 4] + [-1 + 1])/4 = 1.5$$

TABLE 3.2. Training Data Set for Applying Relief Algorithm

Sample	F_1	F_2	Class
1	3	4	C1
2	2	5	C1
3	6	7	C2
4	5	6	C2

In this example the number of samples is low and therefore we use all samples (m = n) to estimate the feature scores. Based on the previous results feature F_1 is much more relevant in classification than feature F_2. If we assign the threshold value of $\tau = 5$, it is possible to eliminate feature F_2 and build the classification model only based on feature F_1.

Relief is a simple algorithm that relies entirely on a statistical methodology. The algorithm employs few heuristics, and therefore it is efficient—its computational complexity is O(mpn). With m, number of randomly selected training samples, being a user-defined constant, the time complexity becomes O(pn), which makes it very scalable to data sets with both a huge number of samples *n* and a very high dimensionality p. When n is very large, it is assumed that m << n. It is one of the few algorithms that can evaluate features for real-world problems with large feature space and large number of samples. Relief is also noise-tolerant and is unaffected by feature interaction, and this is especially important for hard data-mining applications. However, Relief does not help with removing redundant features. As long as features are deemed relevant to the class concept, they will be selected even though many of them are highly correlated to each other.

One of the Relief problems is to pick a proper value of τ. Theoretically, using the so-called Cebysev's inequality τ may be estimated:

$$\tau \ll 1/\sqrt{(\alpha\, m)}$$

While the above formula determines τ in terms of α (data-mining model accuracy) and m (the training data set size), experiments show that the score levels display clear contrast between relevant and irrelevant features so τ can easily be determined by inspection.

Relief was extended to deal with multi-class problems, noise, redundant, and missing data. Recently, additional feature-selection methods based on feature weighting are proposed including ReliefF, Simba, and I-Relief, and they are improvements of the basic Relief algorithm.

3.4 ENTROPY MEASURE FOR RANKING FEATURES

A method for unsupervised feature selection or ranking based on entropy measure is a relatively simple technique, but with a large number of features its complexity increases significantly. The basic assumption is that all samples are given as vectors of a feature's values without any classification of output samples. The approach is based on the observation that removing an irrelevant feature, a redundant feature, or both from a set may not change the basic characteristics of the data set. The idea is to remove as many features as possible and yet maintain the level of distinction between the samples in the data set as if no features had been removed. The algorithm is based on a similarity measure S that is in inverse proportion to the distance D between two n-dimensional samples. The distance measure D is small for close samples (close to 0) and large for distinct pairs (close to one). When the features are numeric, the similarity measure S of two samples can be defined as

$$S_{ij} = e^{-\alpha D_{ij}}$$

where D_{ij} is the distance between samples x_i and x_j and α is a parameter mathematically expressed as

$$\alpha = -(\ln 0.5)/D$$

D is the average distance among samples in the data set. Hence, α is determined by the data. But, in a successfully implemented practical application, it was used a constant value of $\alpha = 0.5$. Normalized Euclidean distance measure is used to calculate the distance D_{ij} between two samples x_i and x_j:

$$D_{ij} = \left[\sum_{k=1}^{n} ((x_{ik} - x_{jk})/(\max_k - \min_k))^2 \right]^{1/2}$$

where n is the number of dimensions and $\max_k$ and $\min_k$ are maximum and minimum values used for normalization of the *k*th dimension.

All features are not numeric. The similarity for nominal variables is measured directly using Hamming distance:

$$S_{ij} = \left(\sum_{k=1}^{n} |x_{ik} = x_{jk}| \right) / n$$

where $|x_{ik} = x_{jk}|$ is 1 if $x_{ik} = x_{jk}$, and 0 otherwise. The total number of variables is equal to n. For mixed data, we can discretize numeric values and transform numeric features into nominal features before we apply this similarity measure. Figure 3.3a is an example of a simple data set with three categorical features; corresponding similarities are given in Figure 3.3b.

The distribution of all similarities (distances) for a given data set is a characteristic of the organization and order of data in an n-dimensional space. This organization may be more or less ordered. Changes in the level of order in a data set are the main criteria for inclusion or exclusion of a feature from the feature set; these changes may be measured by entropy.

Sample	F_1	F_2	F_3
R_1	A	X	1
R_2	B	Y	2
R_3	C	Y	2
R_4	B	X	1
R_5	C	Z	3

(a)

	R_1	R_2	R_3	R_4	R_5
R_1		0/3	0/3	2/3	0/3
R_2			2/3	1/3	0/3
R_3				0/3	1/3
R_4					0/3

(b)

Figure 3.3. A tabular representation of similarity measures S. (a) Data set with three features; (b) a table of similarity measures categorical feature S_{ij} between samples.

From information theory, we know that entropy is a global measure, and that it is less for ordered configurations and higher for disordered configurations. The proposed technique compares the entropy measure for a given data set before and after removal of a feature. If the two measures are close, then the reduced set of features will satisfactorily approximate the original set. For a data set of N samples, the entropy measure is

$$E = -\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(S_{ij}\times\log S_{ij} + (1-S_{ij})\times\log(1-S_{ij}))$$

where S_{ij} is the similarity between samples x_i and x_j. This measure is computed in each of the iterations as a basis for deciding the ranking of features. We rank features by gradually removing the least important feature in maintaining the order in the configurations of data. The steps of the algorithm are based on sequential backward ranking, and they have been successfully tested on several real-world applications.

1. Start with the initial full set of feature F.
2. For each feature $f \in F$, remove one feature f from F and obtain a subset F_f. Find the difference between entropy for F and entropy for all F_f. In the example in Figure 3.2, we have to compare the differences $(E_F - E_{F\text{-}F1})$, $(E_F - E_{F\text{-}F2})$, and $(E_F - E_{F\text{-}F3})$.
3. Let f_k be a feature such that the difference between entropy for F and entropy for F_{fk} is minimum.
4. Update the set of feature $F = F - \{f_k\}$, where "–" is a difference operation on sets. In our example, if the difference $(E_F - E_{F\text{-}F1})$ is minimum, then the reduced set of features is $\{F_2, F_3\}$. F_1 becomes the bottom of the ranked list.
5. Repeat steps 2–4 until there is only one feature in F.

A ranking process may be stopped in any iteration, and may be transformed into a process of selecting features, using the additional criterion mentioned in step 4. This criterion is that the difference between entropy for F and entropy for F_f should be less than the approved threshold value to reduce feature f_k from set F. A computational complexity is the basic disadvantage of this algorithm, and its parallel implementation could overcome the problems of working with large data sets and large number of features sequentially.

3.5 PCA

The most popular statistical method for dimensionality reduction of a large data set is the Karhunen-Loeve (K-L) method, also called *PCA*. In various fields, it is also known as the singular value decomposition (SVD), the Hotelling transform, and the empirical orthogonal function (EOF) method. PCA is a method of transforming the initial data

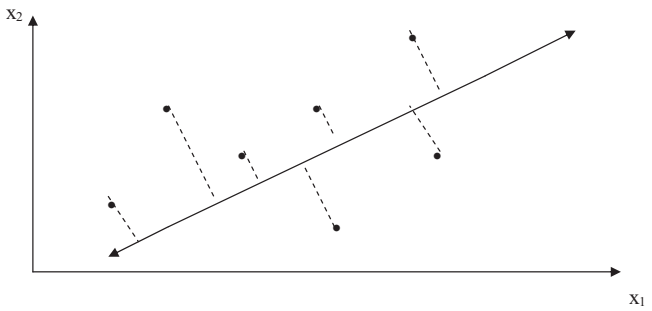

Figure 3.4. The first principal component is an axis in the direction of maximum variance.

set represented by vector samples into a new set of vector samples with derived dimensions. The goal of this transformation is to concentrate the information about the differences between samples into a small number of dimensions. Practical applications confirmed that PCA is the best linear dimension-reduction technique in the mean-square error sense. Being based on the covariance matrix of the features, it is a second-order method. In essence, PCA seeks to reduce the dimension of the data by finding a few orthogonal linear combinations of the original features with the largest variance. Since the variance depends on the scale of the variables, it is customary to first standardize each variable to have mean 0 and standard deviation 1. After the standardization, the original variables with possibly different units of measurement are all in comparable units.

More formally, the basic idea can be described as follows: A set of n-dimensional vector samples $X = \{x_1, x_2, x_3, \ldots, x_m\}$ should be transformed into another set $Y = \{y_1, y_2, \ldots, y_m\}$ of the same dimensionality, but y-s have the property that most of their information content is stored in the first few dimensions. This will allow us to reduce the data set to a smaller number of dimensions with low information loss.

The transformation is based on the assumption that high information corresponds to high variance. So, if we want to reduce a set of input dimensions X to a single dimension Y, we should transform X into Y as a matrix computation

$$Y = A \cdot X$$

choosing A such that Y has the largest variance possible for a given data set. The single dimension Y obtained in this transformation is called *the first principal component*. The first principal component is an axis in the direction of maximum variance. It minimizes the distance of the sum of squares between data points and their projections on the component axis, as shown in Figure 3.4 where a two-dimensional space is transformed into a one-dimensional space in which the data set has the highest variance.

In practice, it is not possible to determine matrix A directly, and therefore we compute the covariance matrix S as a first step in feature transformation. Matrix S is defined as

$$S_{n\times n} = 1/(n-1)\left[\sum_{j=1}^{n}(x_j - x')^T(x_j - x')\right]$$

$$\text{where } x' = (1/n)\sum_{j=1}^{n} x_j.$$

The eigenvalues of the covariance matrix S for the given data should be calculated in the next step. Finally, the m eigenvectors corresponding to the m largest eigenvalues of S define a linear transformation from the n-dimensional space to an m-dimensional space in which the features are uncorrelated. To specify the principal components we need the following additional explanations about the notation in matrix S:

1. The eigenvalues of $S_{n\times n}$ are $\lambda_1, \lambda_2, \ldots, \lambda_n$, where

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0.$$

2. The eigenvectors $e_1, e_2, \ldots, e_n$ correspond to eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, and they are called the principal axes.

Principal axes are new, transformed axes of an n-dimensional space, where the new variables are uncorrelated, and variance for the *i*th component is equal to the *i*th eigenvalue. Because λ_i's are sorted, most of the information about the data set is concentrated in a few first-principal components. The fundamental question is how many of the principal components are needed to get a good representation of the data? In other words, what is the effective dimensionality of the data set? The easiest way to answer the question is to analyze the proportion of variance. Dividing the sum of the first m eigenvalues by the sum of all the variances (all eigenvalues), we will get the measure for the quality of representation based on the first m principal components. The result is expressed as a percentage, and if, for example, the projection accounts for over 90% of the total variance, it is considered to be good. More formally, we can express this ratio in the following way. The criterion for feature selection is based on the ratio of the sum of the *m* largest eigenvalues of S to the trace of S. That is a fraction of the variance retained in the m-dimensional space. If the eigenvalues are labeled so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, then the ratio can be written as

$$R = \left(\sum_{i=1}^{m}\lambda_i\right) \Big/ \left(\sum_{i=1}^{n}\lambda_i\right)$$

When the ratio R is sufficiently large (greater than the threshold value), all analyses of the subset of m features represent a good initial estimate of the n-dimensionality space. This method is computationally inexpensive, but it requires characterizing data with the covariance matrix S.

We will use one example from the literature to show the advantages of PCA. The initial data set is the well-known set of Iris data, available on the Internet for data-

TABLE 3.3. The Correlation Matrix for Iris Data

	Feature 1	Feature 2	Feature 3	Feature 4
Feature 1	1.0000	–0.1094	0.8718	0.8180
Feature 2	–0.1094	1.0000	–0.4205	–0.3565
Feature 3	0.8718	–0.4205	1.0000	0.9628
Feature 4	0.8180	–0.3565	0.9628	1.0000

TABLE 3.4. The Eigenvalues for Iris Data

Feature	Eigenvalue
Feature 1	2.91082
Feature 2	0.92122
Feature 3	0.14735
Feature 4	0.02061

mining experimentation. This data set has four features, so every sample is a four-dimensional vector. The correlation matrix, calculated from the Iris data after normalization of all values, is given in Table 3.3.

Based on the correlation matrix, it is a straightforward calculation of eigenvalues (in practice, usually, one of the standard statistical packages is used), and these final results for the Iris data are given in Table 3.4.

By setting a threshold value for R* = 0.95, we choose the first two features as the subset of features for further data-mining analysis, because

$$R = (2.91082 + 0.92122)/(2.91082 + 0.92122 + 0.14735 + 0.02061) = 0.958 > 0.95$$

For the Iris data, the first two principal components should be adequate description of the characteristics of the data set. The third and fourth components have small eigenvalues and therefore, they contain very little variation; their influence on the information content of the data set is thus minimal. Additional analysis shows that based on the reduced set of features in the Iris data, the model has the same quality using different data-mining techniques (sometimes the results were even better than with the original features).

The interpretation of the principal components can be difficult at times. Although they are uncorrelated features constructed as linear combinations of the original features, and they have some desirable properties, they do not necessarily correspond to meaningful physical quantities. In some cases, such loss of interpretability is not satisfactory to the domain scientists, and they prefer others, usually feature-selection techniques.

3.6 VALUE REDUCTION

A reduction in the number of discrete values for a given feature is based on the second set of techniques in the data-reduction phase; these are the *feature-discretization*

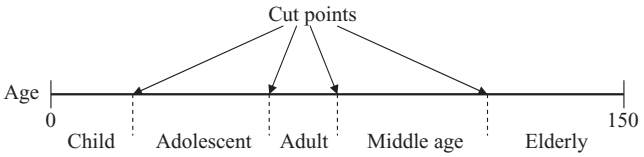

Figure 3.5. Discretization of the *age* feature.

techniques. The task of feature-discretization techniques is to discretize the values of continuous features into a small number of intervals, where each interval is mapped to a discrete symbol. The benefits of these techniques are simplified data description and easy-to-understand data and final data-mining results. Also, more data-mining techniques are applicable with discrete feature values. An "old-fashioned" discretization is made manually, based on our a priori knowledge about the feature. For example, using common sense or consensus, a person's age, given at the beginning of a data-mining process as a continuous value (between 0 and 150 years), may be classified into categorical segments: child, adolescent, adult, middle age, and elderly. Cutoff points are subjectively defined (Fig. 3.5). Two main questions exist about this reduction process:

1. What are the cutoff points?
2. How does one select representatives of intervals?

Without any knowledge about a feature, a discretization is much more difficult and, in many cases, arbitrary. A reduction in feature values usually is not harmful for real-world data-mining applications, and it leads to a major decrease in computational complexity. Therefore, we will introduce, in the next two sections, several automated discretization techniques.

Within a column of a data set (set of feature values), the number of distinct values can be counted. If this number can be reduced, many data-mining methods, especially the logic-based methods explained in Chapter 6, will increase the quality of a data analysis. Reducing the number of values by smoothing feature values does not require a complex algorithm because each feature is smoothed independently of other features and the process is performed only once, without iterations.

Suppose that a feature has a range of numeric values, and that these values can be ordered from the smallest to the largest using standard greater-than and less-than operators. This leads naturally to the concept of *placing the values in bins*—partitioning into groups with close values. Typically, these bins have a close number of elements. All values in a bin will be merged into a single concept represented by a single value—usually either the mean or median of the bin's values. The mean or the mode is effective for a moderate or large number of bins. When the number of bins is small, the closest boundaries of each bin can be candidates for representatives in a given bin.

For example, if a set of values for a given feature f is {3, 2, 1, 5, 4, 3, 1, 7, 5, 3}, then, after sorting, these values will be organized into an ordered set:

{1, 1, 2, 3, 3, 3, 4, 5, 5, 7}

Now, it is possible to split the total set of values into three bins with a close number of elements in each bin:

$$\{\underline{1, 1, 2}; \underline{3, 3, 3}; \underline{4, 5, 5, 7}\}$$

$$\text{BIN}_1; \text{BIN}_2; \text{BIN}_3$$

In the next step, different representatives can be selected for each bin. If the smoothing is performed based on bin modes, the new set of values for each bin will be

$$\{\underline{1, 1, 1}; \underline{3, 3, 3}; \underline{5, 5, 5, 5}\}$$

$$\text{BIN}_1; \text{BIN}_2; \text{BIN}_3$$

If the smoothing is performed based on mean values, then the new distribution for reduced set of values will be

$$\{\underline{1.33, 1.33, 1.33}; \underline{3, 3, 3}; \underline{5.25, 5.25, 5.25, 5.25}\}$$

$$\text{BIN}_1; \text{BIN}_2; \text{BIN}_3$$

and finally, if all the values in a bin are replaced by the closest of the boundary values, the new set will be

$$\{\underline{1, 1, 2}; \underline{3, 3, 3}; \underline{4, 4, 4, 7}\}$$

$$\text{BIN}_1; \text{BIN}_2; \text{BIN}_3$$

One of the main problems of this method is to find the best cutoffs for bins. In theory, a decision about cutoffs cannot be made independently of other features. Still, heuristic decisions for every feature independently give good results in many data-mining applications. The value-reduction problem can be stated as an optimization problem in the selection of k bins. Given the number of k bins, distribute the values in the bins to minimize the average distance of a value from its bin mean or median. The distance is usually measured as the squared distance for a bin mean and as the absolute distance for a bin median. This algorithm can be computationally very complex, and a modified heuristic procedure is used to produce a near-optimum solution. The procedure consists of the following steps:

1. Sort all values for a given feature.
2. Assign approximately equal numbers of sorted adjacent values (v_i) to each bin, where the number of bins is given in advance.
3. Move a border element v_i from one bin to the next (or previous) when that reduces the global distance error (ER; the sum of all distances from each v_i to the mean or mode of its assigned bin).

A simple example of the dynamic bin procedure for feature discretization is given next. The set of values for a feature f is {5, 1, 8, 2, 2, 9, 2, 1, 8, 6}. Split them into three bins (k = 3), where the bins will be represented by their modes. The computations in the first iteration of the algorithm are

1. Sorted set of values for feature f is {1, 1, 2, 2, 2, 5, 6, 8, 8, 9}
2. Initial bins (k = 3) are

$$\{\underline{1, 1, 2}; \underline{2, 2, 5}; \underline{6, 8, 8, 9}\}$$

$$\text{BIN}_1; \text{BIN}_2; \text{BIN}_3$$

3. (i) Modes for the three selected bins are {1, 2, 8}. After initial distribution, the total error, using absolute distance for modes, is

$$\text{ER} = 0+0+1+0+0+3+2+0+0+1 = 7.$$

4. (iv) After moving two elements from BIN_2 into BIN_1 and one element from BIN_3 to BIN_2 in the next three iterations and obtaining smaller and smaller ER, the new and final distribution of elements will be

$$\text{f} = \{\underline{1, 1, 2, 2, 2}; \underline{5, 6}; \underline{8, 8, 9}\}$$

$$\text{Final bins} \Rightarrow \text{BIN}_1; \text{BIN}_2; \text{BIN}_3$$

The corresponding modes are {2, 5, 8}, and the total minimized error ER is 4. Any additional move of elements between bins will cause an increase in the ER value. The final distribution, with medians as representatives, will be the solution for a given value-reduction problem.

Another very simple method of smoothing feature values is *number approximation by rounding*. Rounding is a natural operation for humans; it is also natural for a computer, and it involves very few operations. First, numbers with decimals are converted to integers prior to rounding. After rounding, the number is divided by the same constant. Formally, these steps are described with the following computations applied to the feature value X:

1. Integer division: $Y = \text{int}\ (X/10^k)$
2. Rounding: *If* $(\text{mod}\ [X,10^k] \geq [10^k/2])$ *then* $Y = Y + 1$
3. Integer multiplication: $X = Y * 10^k$

where k is the number of rightmost decimal places to round. For example, the number 1453 is rounded to 1450 if k = 1, rounded to 1500 if k = 2, and rounded to 1000 if k = 3.

Given a number of values for a feature as an input parameter of the procedure, this simple rounding algorithm can be applied in iterations to reduce these values in large

data sets. First, a feature's values are sorted so that the number of distinct values after rounding can be counted. Starting at k = 1, rounding is performed for all values, and the number of distinct values counted. Parameter k is increased in the next iteration until the number of values in the sorted list is reduced to less than the allowable maximum, typically in real-world applications between 50 and 100.

3.7 FEATURE DISCRETIZATION: CHIMERGE TECHNIQUE

ChiMerge is one automated discretization algorithm that analyzes the quality of multiple intervals for a given feature by using χ^2 statistics. The algorithm determines similarities between distributions of data in two adjacent intervals based on output classification of samples. If the conclusion of the χ^2 test is that the output class is independent of the feature's intervals, then the intervals should be merged; otherwise, it indicates that the difference between intervals is statistically significant, and no merger will be performed.

ChiMerge algorithm consists of three basic steps for discretization:

1. Sort the data for the given feature in ascending order.
2. Define initial intervals so that every value of the feature is in a separate interval.
3. Repeat until no χ^2 of any two adjacent intervals is less then threshold value.

After each merger, χ^2 tests for the remaining intervals are calculated, and two adjacent features with the χ^2 values are found. If the calculated χ^2 is less than the lowest threshold, merge these intervals. If no merge is possible, and the number of intervals is greater than the user-defined maximum, increase the threshold value.

The χ^2 test or the contingency table test is used in the methodology for determining the independence of two adjacent intervals. When the data are summarized in a contingency table (its form is represented in Table 3.5), the χ^2 test is given by the formula:

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} (A_{ij} - E_{ij})^2 / E_{ij}$$

where

k = the number of classes,
A_{ij} = the number of instances in the *i*th interval, *j*th class,
E_{ij} = the expected frequency of A_{ij}, which is computed as $(R_i C_j)/N$,
R_i = the number of instances in the *i*th interval = $\sum A_{ij}$, j = 1,…, k,
C_j = the number of instances in the *j*th class = $\sum A_{ij}$, i = 1,2,
N = the total number of instances = $\sum R_i$, i = 1,2.

If either R_i or C_j in the contingency table is 0, E_{ij} is set to a small value, for example, $E_{ij} = 0.1$. The reason for this modification is to avoid very small values in the

denominator of the test. The degree of freedom parameter of the χ^2 test is for one less than the number of classes. When more than one feature has to be discretized, a threshold for the maximum number of intervals and a confidence interval for the χ^2 test should be defined separately for each feature. If the number of intervals exceeds the maximum, the algorithm ChiMerge may continue with a new, reduced value for confidence.

For a classification problem with two classes (k = 2), where the merging of two intervals is analyzed, the contingency table for 2×2 has the form given in Table 3.5. A_{11} represents the number of samples in the first interval belonging to the first class; A_{12} is the number of samples in the first interval belonging to the second class; A_{21} is the number of samples in the second interval belonging to the first class; and finally A_{22} is the number of samples in the second interval belonging to the second class.

We will analyze the ChiMerge algorithm using one relatively simple example, where the database consists of 12 two-dimensional samples with only one continuous feature (F) and an output classification feature (K). The two values, 1 and 2, for the feature K represent the two classes to which the samples belong. The initial data set, already sorted with respect to the continuous numeric feature F, is given in Table 3.6.

We can start the algorithm of a discretization by selecting the smallest χ^2 value for intervals on our sorted scale for F. We define a middle value in the given data as a

TABLE 3.5. A Contingency Table for 2×2 Categorical Data

	Class1	Class2	$\sum$
Interval-1	A_{11}	A_{12}	R_1
Interval-2	A_{21}	A_{22}	R_2
$\sum$	C_1	C_2	N

TABLE 3.6. Data on the Sorted Continuous Feature F with Corresponding Classes K

Sample	F	K
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

TABLE 3.7. Contingency Table for Intervals [7.5, 8.5] and [8.5, 10]

	K = 1	K = 2	$\sum$
Interval [7.5, 8.5]	$A_{11} = 1$	$A_{12} = 0$	$R_1 = 1$
Interval [8.5, 9.5]	$A_{21} = 1$	$A_{22} = 0$	$R_2 = 1$
$\sum$	$C_1 = 2$	$C_2 = 0$	N = 2

TABLE 3.8. Contingency Table for Intervals [0, 7.5] and [7.5, 10]

	K = 1	K = 2	$\sum$
Interval [0, 7.5]	$A_{11} = 2$	$A_{12} = 1$	$R_1 = 3$
Interval [7.5, 10]	$A_{21} = 2$	$A_{22} = 0$	$R_2 = 2$
$\sum$	$C_1 = 4$	$C_2 = 1$	N = 5

splitting interval point. For our example, interval points for feature F are 0, 2, 5, 7.5, 8.5, and 10. Based on this distribution of intervals, we analyze all adjacent intervals trying to find a minimum for the χ^2 test. In our example, χ^2 was the minimum for intervals [7.5, 8.5] and [8.5, 10]. Both intervals contain only one sample, and they belong to class K = 1. The initial contingency table is given in Table 3.7.

Based on the values given in the table, we can calculate the expected values

$$E_{11} = 2/2 = 1$$

$$E_{12} = 0/2 \approx 0.1$$

$$E_{21} = 2/2 = 1, \text{and}$$

$$E_{22} = 0/2 \approx 0.1$$

and the corresponding χ^2 test

$$\chi^2 = (1-1)^2/1 + (0-0.1)^2/0.1 + (1-1)^2/1 + (0-0.1)^2/0.1 = 0.2$$

For the degree of freedom d = 1, and $\chi^2 = 0.2 < 2.706$ (the threshold value from the tables for chi-squared distribution for $\alpha = 0.1$), we can conclude that there are no significant differences in relative class frequencies and that the selected intervals can be merged. The merging process will be applied in one iteration only for two adjacent intervals with a minimum χ^2 and, at the same time, with $\chi^2 <$ threshold value. The iterative process will continue with the next two adjacent intervals that have the minimum χ^2. We will just show one additional step, somewhere in the middle of the merging process, where the intervals [0, 7.5] and [7.5, 10] are analyzed. The contingency table is given in Table 3.8, and expected values are

$$E_{11} = 12/5 = 2.4$$

TABLE 3.9. Contingency Table for Intervals [0, 10] and [10, 42]

	K = 1	K = 2	∑
Interval [0, 10.0]	$A_{11} = 4$	$A_{12} = 1$	$R_1 = 5$
Interval [10.0, 42.0]	$A_{21} = 1$	$A_{22} = 3$	$R_2 = 4$
∑	$C_1 = 5$	$C_2 = 4$	N = 9

$$E_{12} = 3/5 = 0.6$$

$$E_{21} = 8/5 = 1.6$$

$$E_{22} = 2/5 = 0.4$$

while the χ^2 test is

$$\chi^2 = (2-2.4)^2/2.4 + (1-0.6)^2/0.6 + (2-1.6)^2/1.6 + (0-0.4)^2/0.4 = 0.834$$

Selected intervals should be merged into one because, for the degree of freedom d = 1, $\chi^2 = 0.834 < 2.706$ (for $\alpha = 0.1$). In our example, with the given threshold value for χ^2, the algorithm will define a final discretization with three intervals: [0, 10], [10, 42], and [42, 60], where 60 is supposed to be the maximum value for the feature F. We can assign to these intervals coded values 1, 2, and 3 or descriptive linguistic values low, medium, and high.

Additional merging is not possible because the χ^2 test will show significant differences between intervals. For example, if we attempt to merge the intervals [0, 10] and [10, 42]—contingency table is given in Table 3.9—and the test results are $E_{11} = 2.78$, $E_{12} = 2.22$, $E_{21} = 2.22$, $E_{22} = 1.78$, and $\chi^2 = 2.72 > 2.706$, the conclusion is that significant differences between two intervals exist, and merging is not recommended.

3.8 CASE REDUCTION

Data mining can be characterized as a secondary data analysis in the sense that data miners are not involved directly in the data-collection process. That fact may sometimes explain the poor quality of raw data. Seeking the unexpected or the unforeseen, the data-mining process is not concerned with optimal ways to collect data and to select the initial set of samples; they are already given, usually in large numbers, with a high or low quality, and with or without prior knowledge of the problem at hand.

The largest and the most critical dimension in the initial data set is the number of cases or samples or, in other words, the number of rows in the tabular representation of data. Case reduction is the most complex task in data reduction. Already, in the preprocessing phase, we have elements of case reduction through the elimination of outliers and, sometimes, samples with missing values. But the main reduction process is still ahead. If the number of samples in the prepared data set can be managed by the selected data-mining techniques, then there is no technical or theoretical reason for case

reduction. In real-world data-mining applications, however, with millions of samples available, that is not the case.

Let us specify two ways in which the sampling process arises in data analysis. First, sometimes the data set itself is merely a sample from a larger, unknown population, and sampling is a part of the data-collection process. Data mining is not interested in this type of sampling. Second (another characteristic of data mining), the initial data set represents an extremely large population and the analysis of the data is based only on a subset of samples. After the subset of data is obtained, it is used to provide some information about the entire data set. It is often called estimator and its quality depends on the elements in the selected subset. A sampling process always causes a sampling error. Sampling error is inherent and unavoidable for every approach and every strategy. This error, in general, will decrease when the size of subset increases, and it will theoretically become nonexistent in the case of a complete data set. Compared with data mining of an entire data set, practical sampling possesses one or more of the following advantages: reduced cost, greater speed, greater scope, and sometimes even higher accuracy. As yet there is no known method of sampling that ensures that the estimates of the subset will be equal to the characteristics of the entire data set. Relying on sampling nearly always involves the risk of reaching incorrect conclusions. Sampling theory and the correct selection of a sampling technique can assist in reducing that risk, but not in eliminating it.

There are various strategies for drawing a representative subset of samples from a data set. The size of a suitable subset is determined by taking into account the cost of computation, memory requirements, accuracy of the estimator, and other characteristics of the algorithm and data. Generally, a subset size can be determined so that the estimates for the entire data set do not differ by more than a stated margin error in more than δ of the samples. By setting up a probability inequality $P(|e - e_0| \geq \varepsilon) \leq \delta$, we solve it for the subset of sample size n, and for a given value ε (confidence limit) and δ (where $1 - \delta$ is the confidence level). The parameter e stands for an estimate from the subset and it is generally a function of the subset size n, while e_0 stands for the true value obtained from entire data set. However, e_0 is usually unknown too. In this case, a practical way to determine the required size of the data subset can be done as follows: In the first step we select a small preliminary subset of samples of size m. Observations made based on this subset of data will be used to estimate e_0. After replacing e_0 in the inequality, it is solved for n. If $n \geq m$, additional $n - m$ samples are selected in the final subset for analysis. If $n \leq m$ no more samples are selected, and the preliminary subset of data is used as the final.

One possible classification of sampling methods in data mining is based on the scope of application of these methods, and the main classes are

1. general-purpose sampling methods
2. sampling methods for specific domains.

In this text we will introduce only some of the techniques that belong to the first class because they do not require specific knowledge about the application domain and may be used for a variety of data-mining applications.

Systematic sampling is the simplest sampling technique. For example, if we want to select 50% of a data set, we could take every other sample in a database. This approach is adequate for many applications and it is a part of many data-mining tools. However, it can also lead to unpredicted problems when there are some regularities in the database. Therefore, the data miner has to be very careful in applying this sampling technique.

Random sampling is a method by which every sample from an initial data set has the same chance of being selected in the subset. The method has two variants: *random sampling without replacement* and *random sampling with replacement*. Random sampling without replacement is a popular technique in which n distinct samples are selected from N initial samples in the data set without repetition (a sample may not occur twice). The advantages of the approach are simplicity of the algorithm and nonexistence of any bias in a selection. In random sampling with replacement, the samples are selected from a data set such that all samples are given an equal chance of being selected, no matter how often they already have been drawn, that is, any of the samples may be selected more than once. Random sampling is not a one-time activity in a data-mining process. It is an iterative process, resulting in several randomly selected subsets of samples. The two basic forms of a random sampling process are as follows.

1. *Incremental Sampling.* Mining incrementally larger random subsets of samples that have many real-world applications helps spot trends in error and complexity. Experience has shown that the performance of the solution may level off rapidly after some percentage of the available samples has been examined. A principal approach to case reduction is to perform a data-mining process on increasingly larger random subsets, to observe the trends in performances, and to stop when no progress is made. The subsets should take big increments in data sets, so that the expectation of improving performance with more data is reasonable. A typical pattern of incremental subsets might be 10, 20, 33, 50, 67, and 100%. These percentages are reasonable, but can be adjusted based on knowledge of the application and the number of samples in the data set. The smallest subset should be substantial, typically, no fewer than 1000 samples.
2. *Average Sampling.* When the solutions found from many random subsets of samples of cases are *averaged* or *voted*, the combined solution can do as well or even better than the single solution found on the full collection of data. The price of this approach is the repetitive process of data mining on smaller sets of samples and, additionally, a heuristic definition of criteria to compare the several solutions of different subsets of data. Typically, the process of voting between solutions is applied for classification problems (if three solutions are class1 and one solution is class2, then the final voted solution is class1) and averaging for regression problems (if one solution is 6, the second is 6.5, and the third 6.7, then the final averaged solution is 6.4). When the new sample is to be presented and analyzed by this methodology, an answer should be given by each solution, and a final result will be obtained by comparing and integrating these solutions with the proposed heuristics.

Two additional techniques, *stratified sampling* and *inverse sampling*, may be convenient for some data-mining applications. Stratified sampling is a technique in which the entire data set is split into nonoverlapping subsets or strata, and sampling is performed for each different stratum independently of another. The combination of all the small subsets from different strata forms the final, total subset of data samples for analysis. This technique is used when the strata are relatively homogeneous and the variance of the overall estimate is smaller than that arising from a simple random sample. Inverse sampling is used when a feature in a data set occurs with rare frequency, and even a large subset of samples may not give enough information to estimate a feature value. In that case, sampling is dynamic; it starts with the small subset and it continues until some conditions about the required number of feature values are satisfied.

For some specialized types of problems, alternative techniques can be helpful in reducing the number of cases. For example, for time-dependent data the number of samples is determined by the frequency of sampling. The sampling period is specified based on knowledge of the application. If the sampling period is too short, most samples are repetitive and few changes occur from case to case. For some applications, increasing the sampling period causes no harm and can even be beneficial in obtaining a good data-mining solution. Therefore, for time-series data the windows for sampling and measuring features should be optimized, and that requires additional preparation and experimentation with available data.

3.9 REVIEW QUESTIONS AND PROBLEMS

1. Explain what we gain and what we lose with dimensionality reduction in large data sets in the preprocessing phase of data mining.

2. Use one typical application of data mining in a retail industry to explain monotonicity and interruptability of data-reduction algorithms.

3. Given the data set X with three input features and one output feature representing the classification of samples, X:

I_1	I_2	I_3	O
2.5	1.6	5.9	0
7.2	4.3	2.1	1
3.4	5.8	1.6	1
5.6	3.6	6.8	0
4.8	7.2	3.1	1
8.1	4.9	8.3	0
6.3	4.8	2.4	1

(a) Rank the features using a comparison of means and variances.
(b) Rank the features using Relief algorithm. Use all samples for the algorithm (m = 7).

4. Given four-dimensional samples where the first two dimensions are numeric and the last two are categorical

X_1	X_2	X_3	X_4
2.7	3.4	1	A
3.1	6.2	2	A
4.5	2.8	1	B
5.3	5.8	2	B
6.6	3.1	1	A
5.0	4.1	2	B

(a) Apply a method for unsupervised feature selection based on entropy measure to reduce one dimension from the given data set.

(b) Apply Relief algorithm under the assumption that X4 is output (classification) feature.

5. (a) Perform bin-based value reduction with the best cutoffs for the following:

(i) the feature I_3 in problem 3 using mean values as representatives for two bins.

(ii) the feature X_2 in problem 4 using the closest boundaries for two bin representatives.

(b) Discuss the possibility of applying approximation by rounding to reduce the values of numeric attributes in problems 3 and 4.

6. Apply the ChiMerge technique to reduce the number of values for numeric attributes in problem 3.

Reduce the number of numeric values for feature I_1 and find the final, reduced number of intervals.

Reduce the number of numeric values for feature I_2 and find the final, reduced number of intervals.

Reduce the number of numeric values for feature I_3 and find the final, reduced number of intervals.

Discuss the results and benefits of dimensionality reduction obtained in (a), (b), and (c).

7. Explain the differences between averaged and voted combined solutions when random samples are used to reduce dimensionality of a large data set.

8. How can the incremental-sample approach and the average-sample approach be combined to reduce cases in large data sets.

9. Develop a software tool for feature ranking based on means and variances. Input data set is represented in the form of flat file with several features.

10. Develop a software tool for ranking features using entropy measure. The input data set is represented in the form of a flat file with several features.

11. Implement the ChiMerge algorithm for automated discretization of selected features in a flat input file.

12. Given the data set F = {4, 2, 1, 6, 4, 3, 1, 7, 2, 2}, apply two iterations of *bin method for values reduction* with best cutoffs. Initial number of bins is 3. What are the final medians of bins, and what is the total minimized error?

13. Assume you have 100 values that are all different, and use equal width discretization with 10 bins.

(a) What is the largest number of records that could appear in one bin?
(b) What is the smallest number of records that could appear in one bin?
(c) If you use equal height discretization with 10 bins, what is largest number of records that can appear in one bin?
(d) If you use equal height discretization with 10 bins, what is smallest number of records that can appear in one bin?
(e) Now assume that the maximum value frequency is 20. What is the largest number of records that could appear in one bin with equal width discretization (10 bins)?
(f) What about with equal height discretization (10 bins)?

3.10 REFERENCES FOR FURTHER STUDY

Fodor, I. K., *A Survey of Dimension Reduction Techniques*, *LLNL Technical Report*, June 2002.

The author reviews PCA and FA, respectively, the two most widely used linear dimension-reduction methods based on second-order statistics. However, many data sets of interest are not realizations from Gaussian distributions. For those cases, higher order dimension-reduction methods, using information not contained in the covariance matrix, are more appropriate. It includes ICA and method of random projections.

Liu, H., H. Motoda, eds., *Instance Selection and Construction for Data Mining*, Kluwer Academic Publishers, Boston, MA, 2001.

Many different approaches have been used to address the data-explosion issue, such as algorithm scale-up and data reduction. Instance, sample, or tuple selection pertains to methods that select or search for a representative portion of data that can fulfill a data-mining task as if the whole data were used. This book brings researchers and practitioners together to report new developments and applications in instance-selection techniques, to share hard-learned experiences in order to avoid similar pitfalls, and to shed light on future development.

Liu, H., H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, (Second Printing), Kluwer Academic Publishers, Boston, MA, 2000.

The book offers an overview of feature-selection methods and provides a general framework in order to examine these methods and categorize them. The book uses simple examples to show the essence of methods and suggests guidelines for using different methods under various circumstances.

Liu, H., H. Motoda, *Computational Methods of Feature Selection*, CRC Press, Boston, MA, 2007.

The book represents an excellent surveys, practical guidance, and comprehensive tutorials from leading experts. It paints a picture of the state-of-the-art techniques that can boost the capabilities of many existing data-mining tools and gives the novel developments of feature selection that have emerged in recent years, including causal feature selection and Relief. The book contains real-world case studies from a variety of areas, including text classification, web mining, and bioinformatics.

Saul, L. K., et al., Spectral Methods for Dimensionality Reduction, in *Semisupervised Learning*, B. Schööelkopf, O. Chapelle and A. Zien eds., MIT Press, Cambridge, MA, 2005.

Spectral methods have recently emerged as a powerful tool for nonlinear dimensionality reduction and manifold learning. These methods are able to reveal low-dimensional structure in high-dimensional data from the top or bottom eigenvectors of specially constructed matrices. To analyze data that lie on a low-dimensional sub-manifold, the matrices are constructed from sparse weighted graphs whose vertices represent input patterns and whose edges indicate neighborhood relations.

Van der Maaten, L. J. P., E. O. Postma, H. J. Van den Herik, Dimensionality Reduction: A Comparative Review, *Citeseer*, Vol. 10, February 2007, pp. 1–41.

http://www.cse.wustl.edu/~mgeotg/readPapers/manifold/maaten2007_survey.pdf

In recent years, a variety of nonlinear dimensionality-reduction techniques have been proposed that aim to address the limitations of traditional techniques such as PCA. The paper presents a review and systematic comparison of these techniques. The performances of the nonlinear techniques are investigated on artificial and natural tasks. The results of the experiments reveal that nonlinear techniques perform well on selected artificial tasks, but do not outperform the traditional PCA on real-world tasks. The paper explains these results by identifying weaknesses of current nonlinear techniques, and suggests how the performance of nonlinear dimensionality-reduction techniques may be improved.

Weiss, S. M., N. Indurkhya, *Predictive Data Mining: A Practical Guide*, Morgan Kaufman, San Francisco, CA, 1998.

This book focuses on the data-preprocessing phase in successful data-mining applications. Preparation and organization of data and development of an overall strategy for data mining are not only time-consuming processes but fundamental requirements in real-world data mining. A simple presentation of topics with a large number of examples is an additional strength of the book.