Chapter 4
Data Mining for Information System Data

Typically, the term "data mining" refers to the exploration and the analysis of large quantities of data, in order to discover meaningful patterns and rules.

In this book, we will make use of a particular kind of data mining technique known as *clustering*. We will present this technique up to the concepts and definitions needed later. Some other common data mining techniques will be presented as well but, for more information about these techniques and data mining in general, we refer to [126]. Typical data mining tasks, as reported in [13], are: *classification*, examining the features of a "new" object in order to assign it to one of the predefined set of classes (discrete outcomes); *estimation*, similar to classification, but deals with continuously valued outcomes (e.g. regression models or Neural Networks); *affinity grouping* (or *association rules*), determining how things can be grouped together (for example in a shopping cart at the supermarket); *clustering*, the task of segmenting a heterogeneous population into a certain number of homogeneous clusters (no predefined classes); *profiling*, simplification of the description of complicated databases in order to explain some behaviours (e.g. decision trees).

In the next sections one technique per task will be briefly presented.

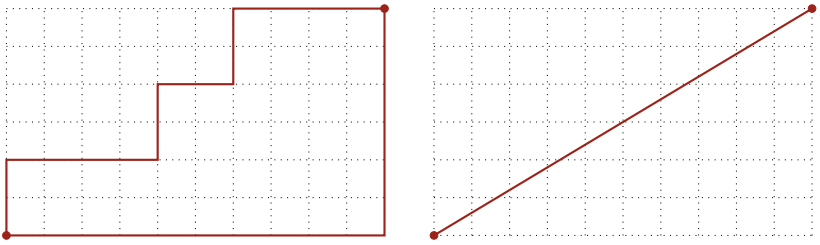
4.1 Classification with Nearest Neighbor

The idea underpinning the nearest neighbor algorithm is that the properties of a certain instance are likely to be similar to the ones in its "neighborhood". To apply this idea, a distance function is necessary, in order to calculate the distance between any two objects. Typical distance functions are the "Manhattan distance" (d_M) and the "Euclidean distance" (d_E):

$$d_M(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} |\mathbf{b}_i - \mathbf{a}_i| \qquad d_E(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{n} (\mathbf{b}_i - \mathbf{a}_i)^2}$$

A. Burattin: *Process Mining Techniques in Business Environments*, LNBIP 207,
DOI 10.1007/978-3-319-17482-2_4

where **a** and **b** are two vectors in the n-dimensional space. Graphical representations of these two distances are reported in Fig. 4.1. This space is "populated" with all the pre-classified elements (examples): each object has a label that defines its class.



(a) Two equivalent representations of d_M. **(b)** Representation of d_E.

Fig. 4.1 Graphical representations of "Manhattan distance" (d_M) and "Euclidean distance" (d_E) in a two dimensional space.

The idea is that the classification is obtained selecting the neighborhood of the new instance (typically, a parameter k indicates to select the first k nearest objects to the current one). Then the class of the new instance is selected as the most common class with respect to the current neighborhood. For example, if $k = 1$ then the class of the new instance is the same class of the nearest one already classified.

4.2 Neural Networks Applied to Estimation

Artificial Neural Networks are mathematical models that typically are represented as directed graphs where each vertex is a "neuron" that can be directly connected to other neurons. The function of a connection is to propagate the activation of one unit to the others. Each connection has a weight that determines the strength of the connection itself. There are three types of neurons: *input* (whose signals collect the external input), *output* (that will produce the result) and *hidden* (the ones that are between the input and the output neurons), as presented in the example of Fig. 4.2.

Each unit has an activation function that combines and transforms all its input values into signal for its output. A typical activation function is the one where the combination of all its input has to reach a certain threshold in order to increment the output. When the combination of input is above the threshold, the output is very high.

The training of the network aims at defining the weights of the connections between units (e.g. $w_{1,1}$ and $w_{n,m}$ in Fig. 4.2) so that, when a new instance is presented to the model, the output values can be calculated and returned as output. The main drawback of this approach is that the trained model is described only in terms of a set of weights that are not able to explain the training data.

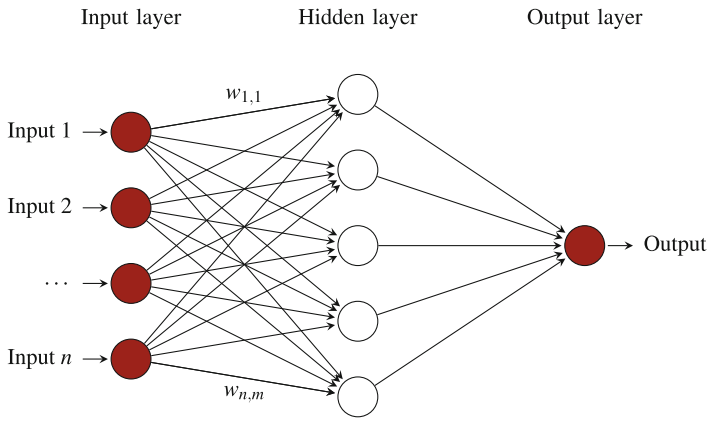


Fig. 4.2 Example of Neural Network with an input, a hidden and an output layer. This network receives input from n neurons and produces output in one neuron.

4.3 Association Rules Extraction

An example of an association rule is *"if a customer purchases onions and potatoes than, the same customer, probably will purchase also a burger*".

One of the most common algorithm to extract association rules is Apriori [4] that, given a set of transactions (called itemset), tries to find subsets of item that are common in many other itemsets (the basic idea is that a subset of a frequent itemset must also be a frequent itemset). These "frequent subsets" are incrementally extended until a termination condition is reached (i.e. there are no more possible extensions).

Starting from the frequent itemset B, the generation of the rules is done considering all the combination of subsets L and $H = B - L$. A rule $L \Rightarrow H$ is added if its confidence (i.e. how much H is observed, given the presence of L) is above a threshold.

4.4 Clustering

The term *clustering* refers to the problem of grouping together objects. In particular, elements of each group, called *cluster*, are supposed to be similar to each other, with respect to some distance measure (low intra-cluster distance). Moreover, elements belonging to different clusters are required to have low similarity (high inter-cluster distance).

4.4.1 Clustering with Self-organizing Maps

Self-organizing maps (SOM) can be considered as a variant of Neural Network. It has an input and an output layer, that consists of many units: each output unit is

connected to all the units in the input layer. Since the output layer is organized as a "grid" it can be graphically visualized. The most important difference with respect to Neural Networks is that Self-Organizing Maps use neighborhood function in order to preserve the topological properties of the input space. This is the main characteristic of SOM: elements that are somehow "close" in the input space should enable neurons that are topologically close in the output layer. To achieve this result, when a training element is learned, not only the weights of the winning output neuron are adjusted, but also the weights for units in its immediate neighborhood are adjusted to strengthen their response to the input.

The training of a SOM makes possible to group together elements that are close but, as in the case of Neural Networks, there is no way to know the reasons that lead to such clustering.

4.4.2 Clustering with Hierarchical Clustering

Another technique to perform clustering is Hierarchical Clustering. In this case, the basic idea is to create an hierarchy of clusters. Clearly, a hierarchy of clusters is much more informative with respect to unrelated sets of clusters[1]. There are two possible approaches to implement Hierarchical Clustering: Hierarchical Agglomerative Clustering (HAC) and Hierarchical Divisive Clustering (HDC).

In HAC, at the beginning, each element constitutes a singleton cluster; and each iteration of the approach merges the closest clusters. The procedure ends when all the elements are agglomerated into a single cluster. HDC adopts the opposite approach: initially all elements belong to the same cluster so that each iteration splits the clusters until all elements constitute a singleton cluster.

The typical way to represent the result of Hierarchical Clustering is using a dendrogram, as shown in Fig. 4.3. Every time two elements are merged, the corresponding lines, on the dendrogram, are merged too. The numbers close to each merge represent the values of the *distance measure* for the two merged clusters.

To extract unrelated sets of clusters out of a dendrogram (as in flat clustering), it is necessary to set a *cut* (or *threshold*). This cut represents the maximum distance allowed to merge clusters. Therefore, only clusters with a distance lower than the threshold are considered as grouped. In the example of Fig. 4.3, two possible cuts are reported. Considering the cut at 0.5, these are the clusters extracted:

$$\{1\} \quad \{2, 3\} \quad \{4\} \quad \{5, 6\} \quad \{7\} \quad \{8\} \quad \{9\} \quad \{10\}.$$

Instead, the cut at 0.8 generates only two clusters:

$$\{1, 2, 3, 4, 5, 6\} \quad \{7, 8, 9, 10\}.$$

[1] In literature, sometimes, techniques generating a hierarchy of clusters and techniques generating unrelated sets of clusters are identified, respectively, as *hierarchical clustering* and *flat clustering*.

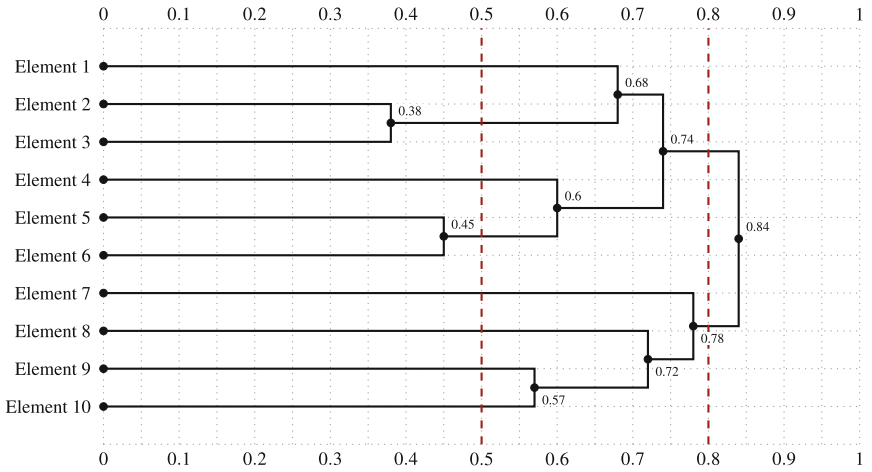


Fig. 4.3 Dendrogram example, with 10 elements. Two possible cuts are reported with red dotted lines (corresponding to values 0.5 and 0.8) (Color figure online).

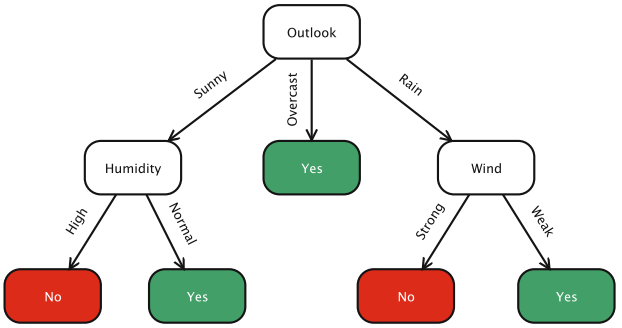


Fig. 4.4 A decision tree that can detect if the weather conditions allow to play tennis.

4.5 Profiling Using Decision Trees

Decision trees are data structure (trees, indeed) that are used to split collections of records in smaller subsets, by applying a sequence of simple decision rules.

The construction of a decision tree is performed top-down, choosing an attribute (the next best) and splitting the current set according to the values of the selected attribute. With each iterative division of the set, the elements of the resulting subsets become more and more similar one another. In order to select the "best" attribute a typical approach is to choose the one that splits the set into homogeneous subsets, however, there are different formulations of such definition.

An interesting characteristic of decision trees is that each path from the root to a leaf node can be considered as a conjunctions of tests on the attributes values; more paths towards the same leaf value encode disjunctions of conjunctions, so a logic representation of the tree can be obtained.

Figure 4.4 represents a decision tree that can detect if the weather conditions allow to play tennis. The logic representation of the tree is:

$$\begin{gathered}(\text{outlook} = \text{'sunny'} \ \wedge \ \text{humidity} = \text{'normal'}) \vee \\ (\text{outlook} = \text{'overcast'}) \ \vee \ (\text{outlook} = \text{'rain'} \ \wedge \ \text{wind} = \text{'weak'})\end{gathered}$$

This peculiarity of decision trees improves their understandability by human beings and, at the same time, results fundamental in order to be processed by third-party systems (e.g. algorithms that need this information).