

It may be summed up in one short sentence:
'Advise your son not to purchase your wine in
Cambridge'

C. Babbage, *Passages from the life of a
philosopher*, 1864, p. 25 [1]

In this chapter we will review some analytical tools. We will need them to analyse the effect of data protection methods on the data. We do not distinguish here whether the tools described belong to the statistics community or to the machine learning community.

Having said that, there is some discussion in the literature on the similarities and differences between machine learning and statistics (see e.g. [2–7]). As a personal comment, and following [4,5], I consider that there is a large overlapping that is increasing year by year due to the research directions in artificial intelligence and machine learning. But of course, some particular topics do not belong to the overlapping region. This is the case of e.g. inductive logic programming and official statistics. The first topic falls within the area of machine learning and the second in the area of statistics.

Some argue that machine learning focuses on prediction without focusing too much on the underlying distribution of the data. For example, [7] illustrates this with the case of data with more variables than samples. This type of problem has been studied in machine learning for e.g. recommender systems, and it is difficult to build parametric models for this type of data. Observe however the study of predictive inference [8] within statistics.

Others [5] mention as a main difference the size of the data, being machine learning devoted to large data sets. From my point of view, this is not an issue of machine

learning, but of data mining, where issues such as dimensionality reduction and scalability are of extreme importance. Data mining has its origin on commercial data where files and databases are typically of large dimensions (although not as large as today's big data). In addition, current research on statistics (see e.g. [9–11]) also consider high-dimensional data and large data sets.

In any case, in real applications, techniques originating from both, machine learning and the statistics communities are needed. As this chapter is mainly instrumental, we describe the tools without distinguishing their origin or their development. In order to present the different tools, we will mainly follow the terminology of Hastie, Tibshirani, and Friedman in [12].

2.1 Classification of Techniques

Methods and techniques in machine learning are typically classified into three large classes according to the type of information available. They are supervised learning, unsupervised learning, and reinforcement learning. We discuss them below. All of them presume a set of labeled examples X used in the learning process. We will also presume that each example x_i in X is described in terms of a set of attributes A_1, \dots, A_M . We will use $A_j(x_i)$ to denote the value of the attribute A_j for example x_i . That is, x_i is a vector in an M -dimensional space.

- **Supervised learning.** In this case, it is presumed that for each example in X there is a distinguished attribute A_y . The goal of supervised learning algorithms is to build a model of this attribute with respect to the other attributes. For example, if the attributes $A_1 \dots A_M$ are numerical and the distinguished attribute A_y is also numerical, the goal of the supervised learning algorithm might be to express A_y as a linear regression of A_1, \dots, A_k for $k \leq M$. In general, A_y is expressed as a function of A_1, \dots, A_k . When A_y is categorical, we call this attribute the class. Figure 2.1 summarizes the notation for supervised learning.

For the sake of simplicity, it is usual to use x_i to denote $A(x_i)$, X to denote the full matrix (or just the full matrix without the attribute A_y), and Y to denote the column A_y . In some applications we need some training sets C that are subsets of X . That is, $C \subseteq X$. Then, we denote by M_C that we have a model learnt from C .

Formally, let us consider a training set C defined in terms of the examples X where for each example x_i in X , we have its known label (outcome or class label) y . Then, we presume that we can express y in terms of a function f of x_i and some

Fig. 2.1 Notation for supervised machine learning

	A_1	\dots	A_M	A_y
x_1	$A_1(x_1)$	\dots	$A_M(x_1)$	$y_1 = A_y(x_1)$
\vdots	\vdots		\vdots	\vdots
x_N	$A_1(x_N)$	\dots	$A_M(x_N)$	$y_N = A_y(x_N)$

error. That is, $y = f(x_i) + \varepsilon$. With this notation we can say that the goal is to build a model M_C that depends on the training set C such that $M_C(x_i)$ approximates $f(x_i)$ for all x_i in the training set. This model is then used to classify unseen instances.

Within supervised learning algorithms we investigate regression problems and classification problems. They are described further below.

- **Regression problems.** They correspond to the case in which A_y is numerical. Models include linear regression (i.e., models of the form $A_y = \sum_{i=1}^k a_i A_i + a_0$ for real numbers a_i), non-linear regression, and neural networks.
 - **Classification problems.** They correspond to the case in which A_y is categorical. Models include logistic regression, decision trees, and different types of rule based systems.
 - **Search problems.** They correspond to the problems in artificial intelligence to speed up search algorithms. However, this type of problems are of no interest in this book.
-
- **Unsupervised learning.** In this case, all the attributes are equal with respect to the learning process, and there is no such a distinguishable attribute. In this case, algorithms try to discover patterns or relationships in the data that can be of interest. Unsupervised learning includes clustering and association rules mining. The former discovers partitions in the data and the latter discovers rules between the attributes.
 - **Reinforcement learning.** In this case we presume that there is already a system (or model) that approximates the data. When the model is used, the system receives a reward if the outcome of the model is successful and a penalty if the outcome is incorrect. These rewards are used to update the model and increase its performance.

2.2 Supervised Learning

A large number of algorithms for supervised machine learning have been developed. We give a brief overview of a few of them. For details and further algorithms the reader is referred to [12, 13].

2.2.1 Classification

We explain superficially decision trees and the nearest neighbor.

2.2.1.1 Decision Trees

A decision tree classifies an element x by means of a chain of (usually binary) questions. These questions are organized as a tree with the first question in the top (the root) and classes in the leaves.

Machine learning algorithms build decision trees from data. The goal is to classify new elements correctly, and minimize the height of the tree (i.e., minimize the number of questions to be asked when a new element arrives).

2.2.1.2 Nearest Neighbor

Classification of a new example x is based on finding the nearest record from a set of stored records (the training set C). The class of this record is returned. Formally,

$$class(x) = class(\arg \min_{x' \in C} d(x', x))$$

where $class(x) = A_y(x)$ using the notation given above.

An alternative is to consider the k nearest records and then return the class of the majority of these k records. This approach corresponds to the k -nearest neighbor.

2.2.2 Regression

There are different approaches to build models for regression. In this section we only review the expressions for linear regression models. There are, however, alternatives. For example, we have non-linear regression models and we can use k -nearest neighbor for regression. The k -nearest neighbor for regression follows the approach of the k -nearest neighbor explained above but instead of returning the class of the majority, the mean of the output of the majority is used.

Let us now focus on linear regression. We will give the expressions in matrix form. For details on regression see e.g. [14].

We denote the data (the training set) by the pair X, Y where Y corresponds to the variable to be modeled (the dependent variable) and X corresponds to the variables of the model (the independent or explanatory variables). In linear regression the model has the following form

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_M x_{iM} + \varepsilon_i.$$

In matrix form, using

$$Y^T = (y_1 y_2 \dots y_N)$$

$$\beta^T = (\beta_0 \beta_1 \beta_2 \dots \beta_M)$$

and

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1M} \\ 1 & x_{21} & x_{22} & \dots & x_{2M} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{N1} & x_{N2} & \dots & x_{NM} \end{pmatrix} \quad (2.1)$$

we have that the model has this form:

$$Y = X\beta + \varepsilon.$$

Then, the ordinary least squares (OLS) method estimates the parameters β of the model computing

$$\beta = [X^T X]^{-1} X^T Y.$$

Statistical properties of this method (as e.g. the Gauss-Markov theorem) can be found in [14].

2.2.3 Validation of Results: k-Fold Cross-Validation

This is one of the most used approaches to evaluate the performance of a model. The approach is based on having a data set Z and then building several pairs of (*training*, *testing*) sets from this single data set. For each pair we can compute a model using the training set and evaluate its performance with the test set.

For a given parameter k , we divide the set Z into k subsets of equal size. Let

$$Z = (Z_1, Z_2, \dots, Z_k)$$

be these sets. Then, we define for $i = 1, \dots, k$ the pair of training and test sets (C_i^{Tr}, C_i^{Ts}) as follows:

$$C_i^{Tr} = \cup_{j \neq i} Z_j$$

$$C_i^{Ts} = Z_i.$$

Given these sets, we can compute the accuracy of any machine learning algorithm that when applied to the training set C returns a classification model M_C using the following expression:

$$accuracy = \frac{\sum_{i=1}^k |\{x | M_C(x) = A_y(x), x \in C_i^{Ts}\}|}{\sum_{i=1}^k |C_i^{Ts}|}.$$

Note that accuracy is not the only way to evaluate the performance of a classifier. Nevertheless, we will not discuss alternatives here. Cross-validation can also be used to evaluate regression.

2.3 Unsupervised Learning

The area of unsupervised learning has developed several families of methods to extract information from unclassified raw data. In this section we will focus on methods for clustering, for association rule mining, and on the expectation-maximization algorithm. They are the ones that will be used later in this book.

2.3.1 Clustering

The objective is not to choose a ‘best’ clustering technique or program. Such a task would be fruitless and contrary to the very nature of clustering.

Dubes and Jain, 1976 [15], p. 247

The goal of clustering, also known as cluster analysis, is to detect the similarities between the data in a set of examples. Different cluster methods differ on the type of data considered and on the way used to express the similarities.

For example, most clustering methods are applicable to numerical data. Nevertheless, other methods can be used on categorical data, time series, search logs, or even on nodes in social networks. With respect to the way used to express the similarities between the data, some clustering methods build partitions of the data objects, others build fuzzy partitions of these data, fuzzy relationships between the objects, and hierarchical structures (dendrograms).

In all cases, the goal of a clustering method is to put similar objects together in the same group or cluster, and put dissimilar ones in different clusters. For achieving this, a crucial point is how to measure the similarity between objects. Different definitions of similarity and distance lead to different clusters.

Methods and algorithms for clustering can be classified according to several dimensions. As expressed above, one is the type of data being clustered, another is the type of structure built around the data. Reference [12] (page 507) considers another dimension that refers to our assumptions on data. The following classes of clustering algorithms are considered: combinatorial algorithms, algorithms for mixture modeling, and algorithms that are mode seekers. See the outline in Fig. 2.2. We briefly describe these classes below.

- **Combinatorial algorithms.** They do not presume any underlying probability distribution. They directly work on the data.
- **Mixture modeling algorithms.** They presume an underlying probability density function. Assuming a parametric approach, clustering consists of finding the parameters of the model (a mixture of density functions). E.g., two Gaussian distributions are fitted to a set of points.

Clustering methods

- Combinatorial methods.
 - Partitive methods (top-down): c -means, fuzzy c -means.
 - Agglomerative methods (bottom-up): single linkage.
- Mixture modeling methods.
- Mode seeker methods.

Fig. 2.2 A classification of some clustering methods

- **Mode seeker algorithms.** They also presume an underlying probability density function but in this case the perspective is nonparametric. So, there is no such a prior assumption that data follows a particular model.

In the rest of this section we review some methods for clustering. We focus on methods for numerical data that lead to crisp and fuzzy partitions. These algorithms belong to the family of combinatorial algorithms. Both type of methods are *partitive*, this means that we have initially a single set of data (a single cluster) and then we partition this cluster into a set of other clusters. In contrast, we find in the literature agglomerative methods that start with as many clusters as data, and then merge some of these clusters to build new ones. Agglomerative methods can be seen as bottom-up methods, and partitive methods as top-down.

Following Dubes and Jain [15, 16], we can distinguish between clustering methods (or techniques) and clustering algorithms (or programs). A clustering method is to specify the general strategy for defining the clusters. In contrast, a clustering algorithm implements the strategy and might use some heuristics. This difference will be further stressed below when describing the k -means.

2.3.1.1 Crisp Clustering

Given a data set X a crisp clustering algorithm builds a partition of the objects in X . Formally, $\Pi = \{\pi_1, \dots, \pi_c\}$ is a partition of X if $\cup \pi_i = X$ and for all $i \neq j$ we have $\pi_i \cap \pi_j = \emptyset$.

For any set of n objects, given c , the number of possible partitions of c clusters is the Stirling number of the second kind (see [16] p. 91, and [12] Sect. 14.30):

$$S(n, c) = \frac{1}{c!} \sum_{k=1}^c (-1)^{c-k} \binom{c}{k} k^n.$$

When c is not known and any number of clusters of $c = 1, \dots, n$ is possible, the number of possible partitions of a set with n elements is the Bell number.

$$B_n = \sum_{k=1}^n S(n, k).$$

It is known (see [17]) that for $n \in \mathbb{N}^n$

$$\left(\frac{n}{e \ln n}\right)^n < B_n < \left(\frac{0.792n}{\ln(n+1)}\right)^n.$$

Different methods exist for selecting or constructing one of these partitions. In optimal clustering, the partition is selected as the one that minimizes an objective function. That is, given an objective function OF , and a space of solutions S , select Π as the solution s that minimizes OF . Formally,

$$\Pi = \arg \min_{s \in S} OF(s).$$

One of the most used methods for clustering is k -means, also known as crisp c -means in the community of fuzzy clustering. This algorithm uses as inputs the data set X and also the number of clusters c . This method is defined as an optimal clustering with the following objective function.

$$OF(\Pi) = \sum_{k=1}^c \sum_{x \in \pi_k} \|A(x) - p_k\|^2 \quad (2.2)$$

Here, π_k , which is a part of partition Π , corresponds to a cluster and p_k is the centroid or prototype of this cluster. $\|u\|$ is the norm of the vector u . That is, $\|u\| = \sqrt{u_1^2 + \dots + u_M^2}$.

Expression 2.2 can be rewritten in terms of characteristic functions χ_k of sets π_k . That is, for each set π_k we have a characteristic function $\chi_k : X \rightarrow \{0, 1\}$ such that $\chi_k(x) = 1$ if and only if $x \in \pi_k$. Using this notation, the goal of the clustering algorithm is to determine the set of characteristic functions $\chi = \{\chi_1, \dots, \chi_c\}$ as well as the cluster centroids $P = \{p_1, \dots, p_c\}$.

The characteristic functions define a partition. Because of that we require χ to satisfy

- $\chi_k(x) \in \{0, 1\}$ for all $k = 1, \dots, c$ and $x \in X$, and that
- for all $x \in X$ there is exactly one k_0 such that $\chi_{k_0}(x) = 1$.

The last condition can be equivalently expressed as $\sum_{k=1}^c \chi_k(x) = 1$ for all $x \in X$.

Taking all this into account, we formalize the c -means problem as follows:

Minimize

$$OF(\chi, P) = \sum_{k=1}^c \sum_{x \in X} \chi_k(x) \|A(x) - p_k\|^2 \quad (2.3)$$

subject to

$$\chi \in M_c = \left\{ \chi_k(x) \mid \chi_k(x) \in \{0, 1\}, \sum_{k=1}^c \chi_k(x) = 1 \text{ for all } x \in X \right\}$$

This optimization problem is usually solved by means of an iterative algorithm that interleaves two steps. In the first step, we presume that P is known and determines the partition χ that minimizes the objective function $OF(\chi, P)$ given P . In the second step, we presume that the partition χ is known and we determine the cluster centers P that minimize the objective function $OF(\chi, P)$ given χ . This process is repeated until convergence.

This algorithm does not ensure a global minimum, but ensures convergence to a local minimum. We discuss this in more detail later.

Let us now formalize the steps above and give expressions for their calculation. The steps are as follows.

- Step 1. Define an initial partition and compute its set of centroids P .
- Step 2. Solve $\min_{\chi \in M_c} OF(\chi, P)$.
- Step 3. Solve $\min_P OF(\chi, P)$.
- Step 4. Repeat steps 2 and 3 till convergence.

The solution of Step 2 consists of assigning each object in X to the nearest cluster. Formally, for all $x \in X$ use the following assignments.

- $k_0 := \arg \min_i \|A(x) - p_i\|^2$
- $\chi_{k_0}(x) := 1$
- $\chi_j(x) := 0$ for all $j \neq k_0$

Note that in this definition k_0 depends on $x \in X$.

To prove that this is the optimal solution of the problem stated in Step 2, let us consider the objective function

$$OF(\chi, P) = \sum_{k=1}^c \sum_{x \in X} \chi_k(x) \|A(x) - p_k\|^2.$$

Naturally, we have that for a given x and p_1, \dots, p_c , it holds

$$\|A(x) - p_k\| \geq \|A(x) - p_{k_0}\|$$

for all $k \in \{1, \dots, c\}$, when k_0 is the index $k_0 = \arg \min_i \|A(x) - p_i\|^2$. Therefore, the assignment $\chi_{k_0}(x) = 1$ and $\chi_j(x) = 0$ for all $j \neq k_0$ minimizes

$$\sum_{x \in X} \chi_k(x) \|A(x) - p_k\|^2$$

for all $k \in \{1, \dots, c\}$, and thus the objective function.

The solution of Step 3 consists in computing for all $k = 1, \dots, c$.

$$p_k = \frac{\sum_{x \in X} \chi_k(x) A(x)}{\sum_{x \in X} \chi_k(x_i)} \quad (2.4)$$

To prove that this is the optimal centroid we consider again the objective function $OF(\chi, P)$ and derive it with respect to p_k . Taking into account that $\frac{\partial OF}{\partial p_k} = 0$, we obtain an expression for p_k . Note that

$$0 = \frac{\partial OF}{\partial p_k} = 2 \sum_{x \in X} \chi_k(x) (A(x) - p_k) (-1),$$

and, therefore, we get the equation

$$-2 \sum_{x \in X} \chi_k(x) A(x) + \sum_{x \in X} \chi_k(x) p_k = 0,$$

that leads to Eq. 2.4.

If we put all the items together, we get Algorithm 1.

As stated above, there is no guarantee that this algorithm leads to the global optimal solution. However, it can be proven that it converges to a local optimal one. Note that at each step the objective function is reduced. In Step 2, with fixed centroids P , the objective function is reduced changing χ . Then, in Step 3, with fixed χ , the objective function is reduced changing P . As the objective function is always positive, convergence is ensured.

Different executions of this algorithm using the same initialization lead to the same results. Nevertheless, due to the fact that the algorithm does not ensure a global

Algorithm 1: Clustering: c -means.

Step 1. Define an initial partition and compute its centroid P .

Step 2. Solve $\min_{\chi \in M_c} OF(\chi, P)$ as follows:

- For all $x \in X$,
 - $k_0 := \arg \min_i \|A(x) - p_i\|^2$
 - $\chi_{k_0}(x) := 1$
 - $\chi_j(x) := 0$ for all $j \in \{1, \dots, c\}$ s.t. $j \neq k_0$

Step 3. Solve $\min_P OF(\chi, P)$ as follows:

- for all $k \in \{1, \dots, c\}$,
 - $p_k := \frac{\sum_{x \in X} \chi_k(x) A(x)}{\sum_{x \in X} \chi_k(x_i)}$

Step 4. Repeat steps 2 and 3 till convergence

minimum but a local one, we have the situation where different initializations can lead to different local minima. This fact is very important when we need to compare clusters obtained from the application of this algorithm.

To partially solve this problem, we can use some of the existing methods for selecting a good initialization for a dataset X . For some initialization methods, see e.g. [18]. Another option is to apply the same algorithm several times to the same data set X , but with different initializations. Then, each application will lead to a partition with its corresponding value for the objective function. Let $r = 1, \dots, R$ denote the r th application, Π_r the partition obtained and OF_r its corresponding objective function. All partitions Π_r are local optima of the same objective function OF . Then, we select the partition Π_r with minimum OF_r . That is, we select the partition

$$r_0 = \arg \min OF_r.$$

This approach does not ensure finding the global optimum, it can still lead to a local optimum. Nevertheless it gives us more chances of finding it. We have used this approach in [19], where 20 different executions were used, and in [20] where 30 were used.

The outcome of c -means permits us to define classification rules for any element d in the same domain D of the elements in X . That is, not only the elements x can be classified but any $d \in D$ can be classified to one of the clusters π_1, \dots, π_c . The classification rule is:

$$cluster(d) = \arg \min_{k=1}^c ||d - p_k||^2.$$

The application of this classification rule in a domain D results into a Voronoi diagram described by the centers P and the Euclidean distance. Recall that the Voronoi diagram of a domain D divides D into a set of regions. Here, the regions are $(R_k)_{k \in \{1, \dots, c\}}$, where

$$R_k = \{d \in D \mid ||d - p_k|| \leq ||d - p_j|| \text{ for all } j \neq k\}.$$

2.3.1.2 Fuzzy Clustering

Fuzzy clustering algorithms return a fuzzy partition instead of a crisp partition. In fuzzy partitions, clusters typically overlap. This causes elements $x \in X$ to have partial membership to different clusters. Partial membership is represented by a value in the $[0, 1]$ interval.

In this section we review some of the algorithms that lead to fuzzy partitions. We begin by reviewing the notion of membership function used to define fuzzy sets [21], and then the notion of fuzzy partition. For a discussion on the difference between fuzzy and probabilistic uncertainty (from a *fuzzy* point of view) see [22].

Definition 2.1 [21]

Let X be a reference set. Then $\mu : X \rightarrow [0, 1]$ is a membership function.

Definition 2.2 [23] Let X be a reference set. Then, a set of membership functions $\mathcal{M} = \{\mu_1, \dots, \mu_c\}$ is a fuzzy partition of X if for all $x \in X$ we have

$$\sum_{i=1}^c \mu_i(x) = 1$$

Fuzzy c -means (FCM) [24] is one of the most used algorithms for fuzzy clustering. It can be seen as a generalization of crisp c -means that has a similar objective function. The solution of the problem is a fuzzy partition. That is, given a value c , the algorithm returns c membership functions μ_1, \dots, μ_c that define a fuzzy partition of the elements of the domain X . Figure 2.3 discusses a naive fuzzification of c -means.

The notation follows the one of c -means. X is the set of records, $P = \{p_1, \dots, p_c\}$ representing the cluster centers or centroids, μ_i is the membership function of the i th cluster and, then, $\mu_i(x_k)$ is the membership of the k th record to the i th cluster. μ_{ik} is also used as an expression equivalent to $\mu_i(x_k)$.

Fuzzy c -means has two parameters. One is the number of clusters c , as in the c -means. Another is a value m that measures the degree of fuzziness of the solution. The value m should be larger than or equal to one. When $m = 1$, the problem to optimize corresponds to the c -means and the algorithm returns a crisp partition. Then, the larger the m , the fuzzier is the solution. In particular, for large values of m , we have that the solutions are completely fuzzy and memberships in all clusters are $\mu_i(x_k) = 1/c$ for all i and $x_k \in X$.

The optimization problem follows.

$$\begin{aligned} &\text{Minimize} \\ &OF_{FCM}(\mu, P) = \{\sum_{i=1}^c \sum_{x \in X} (\mu_i(x))^m ||x - p_i||^2\} \\ &\text{subject to} \\ &\mu_i(x) \in [0, 1] \text{ for all } i \in \{1, \dots, c\} \text{ and } x \in X \\ &\sum_{i=1}^c \mu_i(x) = 1 \text{ for all } x \in X. \end{aligned} \tag{2.5}$$

This problem is usually solved using Algorithm 2. The algorithm is an iterative process similar to the one of c -means. It iterates two steps. One step estimates the membership functions of elements to clusters (taking centroids as fixed). The other step estimates the centroids for each cluster (taking membership functions as fixed). The algorithm converges but as in the case of c -means the solution can be a local

Naive fuzzy c -means. A naive fuzzification of the c -means algorithm is to replace the constraint of χ in $\{0, 1\}$ in Equation 2.3 by another requiring χ to be a value in $[0, 1]$. Nevertheless, this fuzzification has no practical effect. It does not lead to fuzzy solutions. In other words, all solutions of this alternative problem are crisp partitions. That is, although χ is permitted to take values different to 0 and 1, all solutions have values of χ in the extremes of the interval $[0, 1]$.

Fig. 2.3 Remark on a naive fuzzy c -means

Algorithm 2: Clustering: fuzzy c -means.

- Step 1. Generate initial P
 Step 2. Solve $\min_{\mu \in M} OF_{FCM}(\mu, P)$ by computing for all $i \in \{1, \dots, c\}$ and $x \in X$:

$$\mu_i(x) := \left(\sum_{j=1}^c \left(\frac{\|x - p_i\|^2}{\|x - p_j\|^2} \right)^{\frac{1}{m-1}} \right)^{-1}$$

- Step 3. Solve $\min_P OF_{FCM}(\mu, P)$ by computing for all $i \in \{1, \dots, c\}$:

$$p_i := \frac{\sum_{x \in X} (\mu_i(x))^m x}{\sum_{x \in X} (\mu_i(x))^m}$$

- Step 4. If the solution does not converge, go to Step 2; otherwise, stop.

optimum. The algorithm does not discuss the case of denominators equal to zero. This is solved with adhoc definitions for μ (see e.g. [25–27]).

Expressions for $\mu_i(x)$ and p_i in Steps 2 and 3 are determined using Lagrange multipliers (see e.g. [24]). The expression to minimize includes the objective function OF_{FCM} as well as the constraints $\sum_{i=1}^c \mu_i(x) = 1$ for all $x \in X$. Each constraint is multiplied by the corresponding Lagrange multiplier λ_k (for $k = 1, \dots, N$).

$$\begin{aligned} L &= OF_{FCM}(\mu, P) + \sum_{k=1}^N \lambda_k \left(\sum_{i=1}^c \mu_i(x_k) - 1 \right) \\ &= \sum_{i=1}^c \sum_{k=1}^N (\mu_i(x_k))^m \|x - p_i\|^2 + \sum_{k=1}^N \lambda_k \left(\sum_{i=1}^c \mu_i(x_k) - 1 \right) \end{aligned} \quad (2.6)$$

Now, in order to find the expression for $\mu_i(x_k)$, we consider the partial derivatives of L with respect to $\mu_i(x_k)$, that need to be zero. These partial derivatives are

$$\frac{\partial L}{\partial \mu_i(x_k)} = m(\mu_i(x_k))^{m-1} \|A(x_k) - p_i\|^2 + \lambda_k = 0$$

Therefore, we have the following expression for $\mu_i(x_k)$

$$\mu_i(x_k) = \left(\frac{-\lambda_k}{m \|A(x_k) - p_i\|^2} \right)^{\frac{1}{m-1}}$$

Now, taking advantage of the fact that $\sum_{i=1}^c \mu_i(x) = 1$ for all $x \in X$, we get rid of λ_k and obtain the expression for $\mu_i(x_k)$ in Step 2.

Similarly, in order to find the expression for p_i , we proceed with the partial derivative of L with respect to p_i . That is,

$$\frac{\partial L}{\partial p_i} = \sum_{k=1}^N m (\mu_i(x_k))^{m-1} 2 (A(x_k) - p_i) (-1) = 0.$$

From this expression we get the expression for p_i in Step 3.

2.3.1.3 Variations for Fuzzy Clustering

There are several variations of fuzzy c -means. One of them is entropy based fuzzy c -means (EFCM). This method, which was proposed in [28], introduces fuzziness into the solution by adding to the objective function a term based on entropy. In a way similar to fuzzy c -means the algorithm uses a parameter λ ($\lambda \geq 0$) to control the degree of fuzziness. The larger the parameter, the more crisp is the solution. When $\lambda \rightarrow \infty$, the added term becomes negligible and the algorithm corresponds to standard c -means. In contrast, the near the parameter λ is to zero, the fuzzier is the solution. For λ near to zero, solutions have memberships $\mu_i = 1/c$ for all $i = 1, \dots, c$.

The optimization problem for EFCM is defined as follows.

Minimize

$$OF_{EFCM}(\mu, P) = \sum_{x \in X} \sum_{i=1}^c \{\mu_i(x) \|x - p_i\|^2 + \lambda^{-1} \mu_i(x) \log \mu_i(x)\}$$

s.t.

$$\begin{aligned} \mu_i(x) &\in [0, 1] \\ \sum_{i=1}^c \mu_i(x) &= 1 \text{ for all } x \in X. \end{aligned} \quad (2.7)$$

As in the previous algorithms, this problem is solved by an iterative process that repeats two steps. One finds membership values ($\mu_i(x)$) that minimize the objective function given centers, and the other that find centers (p_i) given membership values. The expressions follows.

$$p_i = \frac{\sum_{x \in X} \mu_i(x) x}{\sum_{x \in X} \mu_i(x)} \quad (2.8)$$

$$\mu_i(x) = \frac{e^{-\lambda \|x - p_i\|^2}}{\sum_{j=1}^c e^{-\lambda \|x - p_j\|^2}} \quad (2.9)$$

The last expression for $\mu_i(x)$ can be rewritten as follows.

$$\mu_i(x) = \frac{1}{1 + \frac{\sum_{j \neq i}^c e^{-\lambda \|x - p_j\|^2}}{e^{-\lambda \|x - p_i\|^2}}} \quad (2.10)$$

There are several variations of the algorithms described above. One of them was introduced in [29]. Fuzzy c -means has an implicit assumption that all clusters have equal size. Because of that, given two clusters, with cluster centers p_1 and p_2 the mid-point between the two centers has equal membership to both clusters. That is, $\mu_1((p_1 + p_2)/2) = \mu_2((p_1 + p_2)/2) = 0.5$. If one of the clusters is larger than the other, we might have some elements incorrectly classified. This is illustrated in Fig. 2.4. There is one cluster with 1000 points centered in $(-2, 0)$ and another cluster with 10 points centered in $(2, 0)$. Classification according to fuzzy c -means assigns all points (x, y) with $x \leq 0$ to the cluster centered in $(-2, 0)$ and all points with $x > 0$ to the cluster centered in $(2, 0)$. This will incorrectly assign some of the

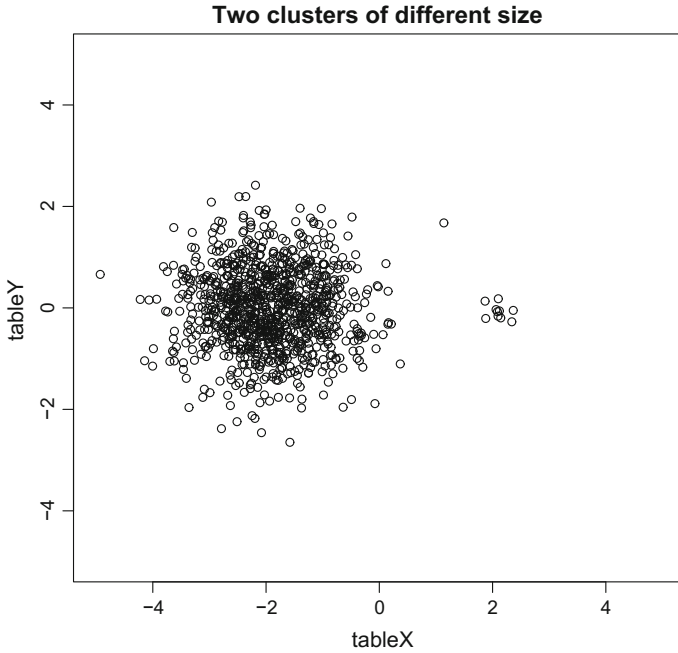


Fig. 2.4 Two clusters of different size. Fuzzy c -means and entropy based fuzzy c -means assigns some of the points to the incorrect cluster

points of the cluster in the left to the one in the right. The data represented in the figure was generated according to independent Normal distributions $N(0, 0.8)$ on each variable. One cluster with mean $(-2, 0)$ and the other with mean $(2, 0)$.

The algorithm presented in [29] solves this problem introducing a variable for each cluster that corresponds to its size. Then, the algorithm determines the variables corresponding to the sizes as well as the membership functions and the cluster centers. A similar approach was also introduced in [30].

Let us consider the same notation above, and let α_i denote the size of the i th cluster for $i \in \{1, \dots, c\}$. Then, the variable size fuzzy c -means problem corresponds to the following optimization problem.

Minimize

$$OF_{VSFCM}(\alpha, \mu, P) = \sum_{i=1}^c \alpha_i \sum_{x \in X} (\alpha_i^{-1} \mu_i(x))^m \|x - p_i\|^2$$

subject to

$$\begin{aligned} \mu_i(x) &\in [0, 1] \text{ for all } i \in \{1, \dots, c\} \text{ and } x \in X \\ \sum_{i=1}^c \mu_i(x) &= 1 \text{ for all } x \in X \\ \sum_{i=1}^c \alpha_i &= 1 \\ \alpha_i &\geq 0 \text{ for all } i = 1, \dots, c. \end{aligned}$$

(2.11)

Note that now the size of the i th cluster weights the contribution of each cluster in the objective function. The constraints are the same we had before, adding two constraints for the new variables α .

Algorithm 3: Clustering: variable size fuzzy c -means.

Step 1. Generate initial α and P

Step 2. Solve $\min_{\mu \in M} OF_{VSFCM}(\alpha, \mu, P)$ by computing for all $i = 1, \dots, c$ and $x \in X$:

$$\mu_i(x) := \left(\sum_{j=1}^c \left(\frac{\alpha_j}{\alpha_i} \right) \left(\frac{\|x - p_i\|^2}{\|x - p_j\|^2} \right)^{\frac{1}{m-1}} \right)^{-1}$$

Step 3. Solve $\min_P OF_{VSFCM}(\alpha, \mu, P)$ by computing for all $i = 1, \dots, c$:

$$p_i := \frac{\sum_{x \in X} (\mu_i(x))^m x}{\sum_{x \in X} (\mu_i(x))^m}$$

Step 4. Solve $\min_{\alpha} OF_{VSFCM}(\alpha, \mu, P)$ by computing for all $i = 1, \dots, c$:

$$\alpha_i := \left(\sum_{j=1}^c \left(\frac{\sum_{x \in X} (\mu_j(x))^m \|x - p_j\|^2}{\sum_{x \in X} (\mu_i(x))^m \|x - p_i\|^2} \right)^m \right)^{-1}$$

Step 5. If the solution does not converge, go to Step 2; otherwise, stop.

This optimization problem is also solved using an iterative procedure that, in this case, iterates three steps. We have two that, as the ones we had in fuzzy c -means, compute membership values and cluster centers, and we have an additional step about the computation of the values α_i . The expressions for μ , p and α are given in Algorithm 3. Note that the expressions for p does not change, and that now the membership values depend on the values α .

The problem discussed above with the size of the clusters also appears in the case of entropy based fuzzy c -means. Because of that, a variable size entropy based fuzzy c -means was also defined in [28]. The optimization problem considers the following objective function.

$$OF_{EFCM}(\alpha, \mu, P) = \sum_{x \in X} \sum_{i=1}^c \mu_i(x) \|x - p_i\|^2 + \lambda^{-1} \sum_{i=1}^c \mu_i(x) \log(\alpha_i^{-1} \mu_i(x))$$

with the same constraints we had in variable size fuzzy c -means. In this case the iterative solution leads to the following expressions. Note that the expression for the centroids has not changed.

$$\mu_i(x) = \frac{\alpha_i e^{-\lambda \|x - p_i\|^2}}{\sum_{j=1}^c \alpha_j e^{-\lambda \|x - p_j\|^2}}$$

$$\alpha_i = \frac{\sum_{x \in X} \mu_i(x)}{|X|}$$

$$p_i = \frac{\sum_{x \in X} \mu_i(x)x}{\sum_{x \in X} \mu_i(x)}$$

2.3.1.4 Fuzzy Clustering and Noisy Data

In this section we mention a few more fuzzy clustering algorithms that have as their main characteristics that attempt to deal with the problem of noise.

Dave introduced in 1991, see [31], the Noise clustering method to reduce the effects of noisy data in the clusters obtained by FCM. The algorithm defines a noise cluster in order to assign noisy data to it. The distance of any element to this cluster is constant. This constant is a parameter of the algorithm (parameter δ). The other parameters are the ones of the fuzzy c -means. That is, the number of clusters c and the fuzziness m .

Later, Krishnapuram and Keller [32] introduced the Possibilistic c -means (PCM). While in fuzzy c -means, entropy fuzzy c -means and in all their variations the memberships of one element to clusters add up to one, this is not a requirement in the possibilistic c -means. In this case, values are only required to be in the $[0, 1]$ interval. This fact implies that the distribution of memberships defines a possibility distribution, and, thus, a possibilistic interpretation is possible.

As [32] points out, a possibilistic solution cannot be achieved restating the optimization problem of the fuzzy c -means (Eq. 2.5) by just removing the constraint $\sum_{i=1}^c \mu_i(x) = 1$. Note that in this case the minimum is obtained with $\mu_i(x) = 0$ for all x and i . To avoid this problem, the authors remove the constraint and add an extra term to $OF_{FCM}(\mu, P)$ that depends on a parameter for each class: v_i for $i = 1, \dots, c$. This parameter, using authors' words, "determines the distance at which the membership value of a point in a cluster becomes 0.5". So, in some sense this is related to the size of the cluster and similar to the variables α_i in variable size cluster. Note that while α_i are determined by the algorithm, v_i is a parameter of the algorithm. Nevertheless, the authors suggest an expression to compute v_i from data that has some resemblances with the expression for computing α_i in Algorithm 3.

The fact that v_i determines the distance where a membership of 0.5 is achieved implies that this value can be considered as a threshold value and that data with a distance to the centroid larger than v_i is considered as noise for that cluster.

In applications, this algorithm tends to locate the centroids in dense regions, and this causes that clusters have a large overlapping. This problem has been reported in [33, 34]. The fuzzy possibilistic c -means (FPCM) is a variation of this algorithm proposed in [35] to avoid coincident clusters and to make the final clusters less sensitive to initializations. This algorithm was further improved in [36] where the authors of FPCM and PCM introduced possibilistic-fuzzy c -means (PFCM).

We review below the possibilistic-fuzzy c -means (PFCM). In the fuzzy c -means we have that the solutions are such that memberships to clusters add up to one; in the possibilistic fuzzy c -means this is not the case but memberships are only required to be positive. According to [32, 36] this second type of memberships can be understood as the typicality of the element to the cluster. Then, possibilistic-fuzzy c -means includes both values membership and typicality. We will use μ to represent

the usual membership and T to represent the set of typicalities, and $t_i(x)$ to represent the typicality of x to cluster i . The solution (the degree of possibility) for an element x will be $a(\mu_i(x))^m + b(t_i(x))^\eta$ with μ and t representing the degree of fuzziness of membership and typicality, respectively.

Values a and b are given and correspond, respectively, to the importance of membership and typicality. Naturally, when $a = 1$ and $b = 0$, we would have only membership as in the case of FCM.

The definition is not restricted to the Euclidean distance as previously. Now, the definition is in terms of an inner product norm, and thus other distances defined on inner product norms are valid. We will use $\|X\|_A = \sqrt{X^T A X}$ to denote any inner product norm based on matrix A .

Taking all this into account, the possibilistic-fuzzy c -means is defined by the following optimization problem.

$$\begin{aligned}
 &\text{Minimize} \\
 &OF_{PFM}(\mu, T, P) = \sum_{x \in X} \sum_{i=1}^c (a(\mu_i(x))^m + b(t_i(x))^\eta) \times \|x - p_i\|_A^2 \\
 &\quad + \sum_{i=1}^c \gamma_i \sum_{x \in X} (1 - t_i(x))^\eta \\
 &\text{subject to} \\
 &\quad \sum \mu_i(x) = 1 \text{ for all } x \in X \\
 &\quad 0 \leq \mu_i(x) \text{ for all } x \in X \text{ and all } i = \{1, \dots, c\} \\
 &\quad t_i(x) \leq 1 \text{ for all } x \in X \text{ and all } i = \{1, \dots, c\}
 \end{aligned} \tag{2.12}$$

The parameters of the clustering method are a, b, m, η and γ_i for $i = \{1, \dots, c\}$. In this algorithm, a and b define, as described above, the relative importance of fuzzy membership and typicality values in the objective function. We require $a > 0$ and $b > 0$. Then, m is the degree of fuzziness in the solution corresponding to the fuzzy c -means (we require $m > 1$), and η is the fuzzifier corresponding to the PCM (we also require $\eta > 1$). Then, the parameters $\gamma_i > 0$ for $i = 1, \dots, c$ correspond to the parameters v_i of PCM (i.e., “the distance at which the membership value of a point in a cluster becomes 0.5” [32]).

The original paper gives some hints about the definition of the parameters γ_i . One of them is to bootstrap the algorithm using the fuzzy c -means and then define γ_i as follows for a $K > 0$ (with $K = 1$ the most common choice).

$$\gamma_i = K \frac{\sum_{x \in X} (\mu_i(x))^m D_{ikA}^2}{\sum_{x \in X} (\mu_i(x))^m}$$

where $D_{ikA} = \|x_k - v_i\|_A$.

It is clear that when $b = 0$, and also $\gamma_i = 0$ for all i , the problem reduces to a FCM. Reference [36] also proves that also when $b = 0$, the problem implicitly becomes equivalent to FCM. With $a = 0$ it corresponds to PCM.

This optimization problem is solved using an iterative algorithm that interleaves the following three equations.

Let $D_{ikA} = ||x_k - v_i||_A$, then

$$u_{ik} = \left(\sum_{j=1}^c \left(\frac{D_{ikA}}{D_{jkA}} \right)^{\frac{2}{(m-1)}} \right)^{-1}$$

for all $1 \leq i \leq c$ and $1 \leq k \leq n$,

$$t_{ik} = \left(\sum_{j=1}^n \left(\frac{D_{ikA}}{D_{ijA}} \right)^{\frac{2}{(\eta-1)}} \right)^{-1}$$

for all $1 \leq i \leq c$ and $1 \leq k \leq n$, and

$$v_i = \frac{\sum_{k=1}^n (u_{ik}^m + t_{ik}^\eta) x_k}{\sum_{k=1}^n (u_{ik}^m + t_{ik}^\eta)}$$

for all $i = 1, \dots, c$.

The algorithm can be bootstrapped using a set of initial cluster centers. The algorithm requires first to fix the parameters a, b, m, η, γ_i .

2.3.1.5 Comparison of Cluster Results

The need to compare different sets of clusters on the same data set appears naturally when we want to analyse the result of clustering algorithms. We can either need to compare the results of the clustering algorithm with respect to a known set of clusters (a reference partition), or to compare different executions of the same clustering algorithm.

Rand [37] (1971, Section 3), Hubert and Arabie [38], and Anderson et al. [39] mention the following applications related to clustering where functions to compare clusters are needed.

- **Comparison with a reference partition or golden standard.** This golden standard are the “natural” clusters using Rand’s terminology. The results of a clustering algorithm are compared with the reference partition.
- **Comparison with noisy data.** The results of clustering the original data and the perturbed data permits us to measure the sensitivity of an algorithm to noisy data.
- **Comparison with suppressed data (to missing individuals).** Comparison of an original data set X and the same data set after suppression measures the sensitivity of an algorithm to missing data.
- **Comparison of two algorithms.** The results of two different algorithms applied to the same data are compared.
- **Comparison of two successive partitions given by the same algorithm.** This is useful for defining stopping criteria in iterative algorithms.

Table 2.1 Definitions for comparing two partitions Π and Π' .

	$I_{\Pi}(x_1) = I_{\Pi}(x_2)$	$I_{\Pi}(x_1) \neq I_{\Pi}(x_2)$	Total
$I_{\Pi'}(x_1) = I_{\Pi'}(x_2)$	r	t	$r + t = np(\Pi')$
$I_{\Pi'}(x_1) \neq I_{\Pi'}(x_2)$	s	u	$s + u$
	$r + s = np(\Pi)$	$t + u$	$\binom{ X }{2}$

- **Comparison for prediction.** This is about using one of the partitions to predict the other.

In the remaining part of this section we discuss some of the distances and similarity measures for clusters that can be found in the literature. In our definitions we use Π and Π' to denote two partitions of a data set X . We presume that both partitions have the same number of parts, and that this number is n . Then, let $\Pi = \{\pi_1, \dots, \pi_n\}$ and $\Pi' = \{\pi'_1, \dots, \pi'_n\}$, that is, π_i and π'_i denotes a part of Π and, therefore, $\pi_i \subseteq X$ and $\pi'_i \subseteq X$ for all $i = 1, \dots, n$.

Let $I_{\Pi}(x)$ denote the cluster of x in the partition Π . Then, let us define r, s, t , and u as follows:

- r is the number of pairs (x_1, x_2) , with x_1 and x_2 elements of X , and both in the same cluster in Π and also both in the same cluster in Π' . That is, r is the cardinality of the set

$$\{(x_1, x_2) \text{ with } x_1 \in X, x_2 \in X, \text{ and } x_1 \neq x_2 | I_{\Pi}(x_1) = I_{\Pi}(x_2) \text{ and } I_{\Pi'}(x_1) = I_{\Pi'}(x_2)\};$$

- s is the number of pairs (x_1, x_2) where x_1 and x_2 are in the same cluster in Π but not in Π' . That is,

$$\{(x_1, x_2) \text{ with } x_1 \in X, x_2 \in X, \text{ and } x_1 \neq x_2 | I_{\Pi}(x_1) = I_{\Pi}(x_2) \text{ and } I_{\Pi'}(x_1) \neq I_{\Pi'}(x_2)\};$$

- t is the number of pairs where x_1 and x_2 are in the same cluster in Π' but not in Π . That is,

$$\{(x_1, x_2) \text{ with } x_1 \in X, x_2 \in X, \text{ and } x_1 \neq x_2 | I_{\Pi}(x_1) \neq I_{\Pi}(x_2) \text{ and } I_{\Pi'}(x_1) = I_{\Pi'}(x_2)\};$$

- u is the number of pairs where x_1 and x_2 are in different clusters in both partitions.

In addition, we denote $np(\Pi)$ as the number of pairs within clusters in the partition Π . That is

$$np(\Pi) = |\{(x_1, x_2) | x_1 \in X, x_2 \in X, x_1 \neq x_2, I_{\Pi}(x_1) = I_{\Pi}(x_2)\}|$$

where $|\cdot|$ is the cardinality of the set.

Note that using the notation above, $np(\Pi) = r + s$ and $np(\Pi') = r + t$. Table 2.1 presents a summary of these definitions.

The literature presents a large number of indices and distances to compare (crisp) partitions. The most well known are the Rand, the Jaccard and the Adjusted Rand index. We present these three and a few other ones. See e.g. [39,40] for other indices.

- **Rand Index.** Defined by Rand in [37], this index is defined as follows:

$$RI(\Pi, \Pi') = (r + u)/(r + s + t + u)$$

For any Π and Π' , we have $RI(\Pi, \Pi') \in [0, 1]$, with $RI(\Pi, \Pi) = 1$.

- **Jaccard Index.** It is defined as follows:

$$JI(\Pi, \Pi') = r/(r + s + t)$$

For any Π and Π' , we have $JI(\Pi, \Pi') \in [0, 1]$, with $RI(\Pi, \Pi) = 1$.

- **Adjusted Rand Index.** This is a correction of the Rand index so that the expectation of the index for partitions with equal number of objects is 0. This adjustment was done assuming generalized hypergeometric distribution as the model of randomness. That is, if we consider a random generation of two partitions so that they have both n sets, the adjusted Rand index is zero. The definition of the index is as follows.

$$ARI(\Pi, \Pi') = \frac{r - exp}{max - exp}$$

where

$$exp = (np(\Pi)np(\Pi'))/(n(n-1)/2)$$

and where

$$max = 0.5(np(\Pi) + np(\Pi')).$$

First discussion of the adjusted Rand index is due to Morey and Agresti [41] and current expression is due to Hubert and Arabie [38].

- **Wallace Index.** This index is defined in terms of the following expression.

$$WI(\Pi, \Pi') = r/\sqrt{np(\Pi)np(\Pi')}$$

- **Mántaras Distance.** This distance proposed in [42] is defined for two partitions by

$$MD(\Pi, \Pi') = \frac{I(\Pi/\Pi') + I(\Pi'/\Pi)}{I(\Pi' \cap \Pi)}$$

where

$$I(\Pi/\Pi') = - \sum_{i=1}^n P(\pi'_i) \sum_{j=1}^n P(\pi_j/\pi'_i) \log P(\pi_j/\pi'_i)$$

$$I(\Pi' \cap \Pi) = - \sum_{i=1}^n \sum_{j=1}^n P(\pi'_i \cap \pi_j) \log P(\pi'_i \cap \pi_j)$$

and where $P(A)$ is the probability of A estimated as $P(A) = |A|/|X|$.

Transaction number	Itemsets purchased	Items (only first letter)
x_1	{apple, biscuits, chocolate, doughnut, ensaïmada, flour}	{a, b, c, d, e, f}
x_2	{apple, biscuits, chocolate}	{a, b, c}
x_3	{chocolate, doughnut, ensaïmada}	{c, d, e}
x_4	{biscuits}	{b}
x_5	{chocolate, doughnut, ensaïmada, flour}	{c, d, e, f}
x_6	{biscuits, chocolate, doughnut}	{b, c, d}
x_7	{ensaïmada}	{e}
x_8	{chocolate, flour}	{c, f}

Fig. 2.5 Database \mathcal{D} with 8 transactions for the itemset $I = \{\text{apple, biscuits, chocolate, doughnut, ensaïmada, flour, grapes}\}$

In the case of fuzzy clusters, a comparison can be done through α -cuts. That is, all those elements with a membership larger than α have their membership assigned to the value 1, others assigned to zero. Nevertheless, this process does not generate in general partitions and the above expressions cannot be used. Instead, we can compute the absolute distance between memberships. For binary memberships, this distance corresponds to the Hamming distance.

Alternatively, there are a few definitions to generalize some of the existing distances for crisp partitions to the case of fuzzy partitions. First attempts can be found in [43,44] and more consolidated work can be found in [39,45].

2.3.2 Association Rules

Association rules establish relationships between attributes or items in a database. Association rule learning algorithms try to find relevant rules in a given database. A typical application of these algorithms is market basket analysis. A market basket is the set of items a customer buys in a single purchase. Then, a rule relates a set of some items that can be purchased with some other items that consumers usually buy at the same time.

Formally, a database $matchscr D = \{x_1, \dots, x_N\}$ has N transactions (or records) consisting each one in a subset of a predefined set of items. Let $I = \{I_1, \dots, I_m\}$ be the set of items. Then, $x_i \subset I$ for all $i \in 1, \dots, N$. We call itemset any subset of I . Thus, x_i are itemsets.

In order to simplify the algorithms, it is usual to presume that the items in I are ordered in a particular preestablished order, and that the itemsets are not empty (i.e., $|x_i| \geq 1$).

An association rule is an implication of the form

$$X \Rightarrow Y$$

where X and Y are nonempty itemsets with no common items. Formally, $X, Y \subseteq I$ such that $|X| \geq 1$, $|Y| \geq 1$, and $X \cap Y = \emptyset$. X is called the antecedent and Y the consequent of the rule.

We review now some definitions that are needed later. We give examples for each definition based on the database \mathcal{D} in Fig. 2.5.

- **Matching.** An itemset S *matches* a transaction T if $S \subseteq T$. For example, $S_1 = \{\text{chocolate}, \text{doughnut}\}$ matches transactions x_1, x_3, x_5, x_6 , $S_2 = \{\text{flour}\}$ matches transactions x_1, x_5 , and x_8 , and there is no transaction matching $S_3 = \{\text{grapes}\}$.
- **Support count.** The support count of an itemset S , expressed by $\text{Count}(S)$, is the number of transactions (or records) that match S in the database \mathcal{D} . For example, $\text{Count}_{\mathcal{D}}(S_1) = 4$, $\text{Count}_{\mathcal{D}}(S_2) = 3$, and $\text{Count}_{\mathcal{D}}(S_3) = 0$.
- **Support.** The support of an itemset S , expressed by $\text{Support}(S)$ is the proportion of transactions that contain all items in S in the database \mathcal{D} . That is, $\text{Support}_{\mathcal{D}}(S) = \text{Count}_{\mathcal{D}}(S)/|\mathcal{D}|$. For example,

$$\text{Support}_{\mathcal{D}}(S_1) = 4/8, \text{Support}_{\mathcal{D}}(S_2) = 3/8, \text{and } \text{Support}_{\mathcal{D}}(S_3) = 0/8.$$

When the context makes clear the database used, we will simply use $\text{Count}(s)$ and $\text{Support}(s)$ instead of $\text{Count}_{\mathcal{D}}(s)$ and $\text{Support}_{\mathcal{D}}(s)$.

Lemma 2.1 *Let I_1 and I_2 be two itemsets; then if $I_1 \subseteq I_2$ then*

$$\text{Support}(I_1) \geq \text{Support}(I_2).$$

Note that this holds because I_1 matches more itemsets in the database (because has less requirements) than I_2 .

Given a rule of the form $R = (X \Rightarrow Y)$, its support will be the proportion of transactions in which both X and Y hold. This is computed defining the support of the rule as the support of the union of the two itemsets that define the rule.

- **Support of a rule.** The support of the rule $R = (X \Rightarrow Y)$ is the support of $X \cup Y$. I.e.,

$$\text{Support}(R) = \text{Support}(X \cup Y).$$

For example, the support of the rule

$$R_0 = (X \Rightarrow Y)$$

with $X = \{\text{chocolate}, \text{doughnut}\}$ and $Y = \{\text{ensaïmada}\}$ is

$$\text{Support}(X \Rightarrow Y) = \text{Support}(X \cup Y) = \text{Support}(\{\text{chocolate}, \text{doughnut}, \text{ensaïmada}\}) = 3$$

because $\{\text{chocolate}, \text{doughnut}, \text{ensaïmada}\}$ matches x_1, x_3 , and x_5 .

Note that the rule R_0 in the last example does not hold for all the transactions in the database. Although the support of $X = \{\text{chocolate, doughnut}\}$ includes x_1, x_3, x_5, x_6 , we have that x_6 does not include *ensaïmada*. Therefore, the rule does not hold for this transaction. In general, rules do not hold 100% of the time.

In order for a rule to be interesting, it should

- apply to a large proportion of records in the database, and
- have a large prediction capability.

For measuring the first aspect we can use the support. Note that support precisely measures the proportion of itemsets where the rule is applicable and holds. For the example above R_0 applies to

$$\text{Support}(R_0) = 3/8$$

of the transactions in \mathcal{D} .

For measuring the second aspect, we need to estimate the predictive accuracy of a rule. This is measured by the *confidence* of a rule, which is defined in terms of the support of the rule with respect to the support of the antecedent of the rule. In other words, the confidence states how many times the rule leads to a correct conclusion.

- **Confidence of a rule.** The confidence of the rule $R = (X \Rightarrow Y)$ is the support of the rule divided by the support of X . That is,

$$\text{Confidence}(R) = \text{Support}(X \cup Y) / \text{Support}(X).$$

Or, equivalently, using *Count*:

$$\text{Confidence}(R) = \text{Count}(X \cup Y) / \text{Count}(X).$$

As stated above, it is usual that rules are not exact. That is, $\text{Confidence}(R) < 1$ because for Lemma 2.1, $\text{Support}(X \cup Y) \leq \text{Support}(X)$.

As an example, the confidence of the rule $R_0 = (X \Rightarrow Y)$ is

$$\begin{aligned} \text{Confidence}(X \Rightarrow Y) &= \text{Support}(X \cup Y) / \text{Support}(X) \\ &= \text{Support}(\{c, d, e\}) / \text{Support}(\{c, d\}) = 3/4. \end{aligned} \quad (2.13)$$

In order to filter the rules that are not interesting, we will reject all the rules that have a support below a certain threshold. That is, we reject all the rules that only apply to a small set of transactions. For example, the rules with a support less than 0.01. Given a threshold for the support $\text{thr} - s$, we say that an itemset I_0 is supported when $\text{Support}(I_0) \geq \text{thr} - s$.

Algorithm 4: Association Rule Generation: Simple algorithm.

```

Step 1.   $R := \emptyset$ 
Step 2.   $L$  be the set of supported itemsets with cardinality larger than 2 ( $thr - s$ )
Step 3.  for all  $l \in L$ 
Step 4.    for all  $X \subset l$  with  $X \neq \emptyset$  (generate all possible rules from  $l$ )
Step 5.    if ( $Confidence(X \Rightarrow (l \setminus X)) \geq thr - c$ ) then
Step 6.       $R := R \cup (X \Rightarrow (l \setminus X))$ 
Step 7.    end if
Step 8.  end for
Step 9.  end for
Step 10. return  $R$ 

```

For supported itemsets, the following holds.

Lemma 2.2 *Let I_0 be a supported itemset. Then, any non empty subset I'_0 of I_0 (i.e., $I'_0 \subseteq I_0$ such that $|I'_0| \geq 1$) is also supported* \square

Proof From Lemma 2.1 and the fact that I_0 is supported, it follows

$$Support(I'_0) \geq Support(I_0) \geq thr - s.$$

So, I'_0 is also supported. \square

In addition we will also reject the rules below a certain confidence level. That is, the rules that fail too often. For example, rules that apply correctly less than 75% of the times are not valid. We will denote this threshold by $thr - c$.

So as a summary, we are interested in finding rules R such that

$$Support(R) \geq thr - s \tag{2.14}$$

and

$$Confidence(R) \geq thr - c. \tag{2.15}$$

Algorithms to find such rules are known by rule mining algorithms. Algorithm 4 is a simple algorithm for rule mining. The algorithm first considers all supported itemsets. That is, it selects all itemsets with a support larger than $thr - s$. Then, for each of these itemsets, it generates all possible rules, and all those rules with enough confidence are returned.

The cost of this algorithm is very large. If the number of items in the itemset I is m , there are 2^m subsets of I . Of these, $2^m - m - 1$ are the number of subsets of I with cardinality larger than or equal to 2. With small m , the cost becomes unfeasible. For example for the cardinalities (number of different products in the supermarket) $m = 10, 20$, and 100 we have $2^{10} = 1024$, $2^{20} \approx 10^6$, and $2^{100} \approx 10^{30}$.

Because of this, more efficient and heuristic methods have been defined to find relevant and interesting rules.

Support and confidence can be understood as probabilities. In particular, the support of an itemset X can be understood as the probability that X occurs. Therefore, we can estimate $P(X)$ as follows:

$$P(X) = \text{Support}(X) = \frac{\text{transactions satisfying } X}{\text{number of transactions}}.$$

Then, the confidence of rule $R = (X \Rightarrow Y)$ can be understood as the conditional probability of $X \cup Y$ given X . So, we can estimate the probability for this rule $P(Y|X)$ as

$$P(Y|X) = \frac{P(X \cup Y)}{P(X)}.$$

2.3.2.1 Apriori Algorithm

The Apriori algorithm [46] is a well known algorithm for association rule learning. The algorithm incrementally defines candidate itemsets of length k from itemsets of length $k - 1$. This process is based on the following lemma, which is known as the *downward closure lemma*.

Lemma 2.3 [47] *Let L_k be all itemsets with cardinality k . That is,*

$$L_k = \{S \mid |S| \geq k, \text{Support}(K) \geq \text{thr} - s\}.$$

Then, if L_k is empty, $L_{k'}$ is empty for all $k' > k$.

Proof Let us presume that L_k is empty but $L_{k'}$ is not for $k' = k + 1 > k$. This means that there exists a supported itemset I_0 of cardinality $k + 1$. Let i_0 be an item in I_0 . Then, $I_0 \setminus \{i_0\}$ is also supported by Lemma 2.2. As I_0 has cardinality k , we have a contradiction and the proposition is proved. \square

This result permits us to define an algorithm that considers supported itemsets of increasing cardinality. For each itemset L_{k-1} of cardinality $k - 1$ we will define a candidate set for the itemsets of cardinality k and then prune all those that are not sufficiently supported. The supported ones will define L_k . Algorithm 5 describes this process.

The process of constructing the new candidate set is given in Algorithm 6 (see [46, 48]). For each pair of itemsets J_1 and J_2 that share all items except one we compute the union that will have exactly k items. The union will be in C_k . Formally,

$$C_k = \{J_1 \cup J_2 \mid J_1, J_2 \in L_{k-1} \text{ and } |J_1 \cap J_2| = k - 2\}.$$

Algorithm 5: Association Rule Generation: Apriori algorithm.

```

Step 1.   $L_1$  be the set of supported itemsets of cardinality one( $thr - s$ ).
Step 2.  Set  $k := 2$ 
Step 3.  while ( $L_{k-1} \neq \emptyset$ )
Step 4.     $C_k :=$  new candidate set ( $L_{k-1}$ )
Step 5.     $L_k :=$  remove non supported itemsets in  $C_k(thr - s)$ 
Step 6.     $k := k + 1$ 
Step 7.  end while
Step 8.  return  $L_1 \cup L_2 \cup \dots \cup L_k$ 

```

Algorithm 6: Association Rule Generation: Apriori algorithm. Computation of the new candidate set from L_{k-1} . If the elements in the itemsets J_1 and J_2 are ordered, this order can be exploited to speed up the procedure.

```

Step 1.   $C_k := \emptyset$ 
Step 2.  for each pair  $J_1, J_2$  in  $L_{k-1}$ 
Step 3.    if ( $J_1$  and  $J_2$  share  $k - 2$  items) then
Step 4.       $C_k := C_k \cup \{J_1 \cup J_2\}$ 
Step 5.    end if
Step 6.  end for
Step 7.  return  $C_k$ 

```

For example, let us consider $k = 5$ and $L_{k-1} = L_4$ including, among others, $J_1 = \{a, b, c, d\}$ and $J_2 = \{a, b, c, e\}$. Then, as J_1 and J_2 share $k - 2 = 5 - 2 = 3$ items, we will include in C_5 the itemset $J_1 \cup J_2 = \{a, b, c, d, e\}$.

In order to know if an itemset $c \in C_k$ should be in L_k or not, we first check whether its subsets of cardinality $k - 1$ are in L_{k-1} . If one fails to be in L_{k-1} , then c should not be in L_k . However, this is not enough to ensure that the support of c is larger than the threshold. This has to be checked also. Algorithm 7 describes this process.

In the remaining part of this section we consider the application of the Apriori algorithm (Algorithm 5) to the database in Table 2.2. This example is from [46]. We will use a threshold of $thr - s = 2/4$. For the sake of simplicity we will work with the count instead of the support, so, we will use a threshold of 2.

The algorithm starts with the definition of supported itemsets of cardinality one. These itemsets define L_1 . Using the database in Table 2.2, we compute the following count values:

- $\text{Count}(\{1\}) = 2$
- $\text{Count}(\{2\}) = 3$
- $\text{Count}(\{3\}) = 3$
- $\text{Count}(\{4\}) = 1$
- $\text{Count}(\{5\}) = 3$

Algorithm 7: Association Rule Generation: Apriori algorithm. Removal of non supported itemsets in $C_k(thr - s)$ to compute L_k .

```

Step 1.  for all  $c$  in  $C_k$ 
Step 2.    for all subsets  $c'$  of  $c$  with  $k - 1$  elements
Step 3.      remove  $c$  from  $C_k$  if  $c'$  is not in  $L_{k-1}$ 
Step 4.    end for
Step 5.  end for
Step 6.  for all  $c$  in  $C_k$ 
Step 7.    if  $(Support(c) < thr - s)$  then remove  $c$  from  $C_k$ 
Step 8.  end for
Step 9.  return  $C_k$ 

```

Table 2.2 Database for the example of the Apriori algorithm, from [46]

Transaction number	Itemsets
x_1	{1, 3, 4}
x_2	{2, 3, 5}
x_3	{1, 2, 3, 5}
x_4	{2, 5}

Therefore, L_1 consists of all items except 4. That is,

$$L_1 = \{\{1\}, \{2\}, \{3\}, \{5\}\}.$$

The next step (Step 4) is to compute C_2 , the candidate set of itemsets with cardinality 2. This is computed from L_1 using Algorithm 6. This consists in combining itemsets from L_1 such that they have all elements except one in common. In the case of C_2 this corresponds to the pairs J_1 and J_2 from L_1 . Therefore, we get

$$C_2 = \{\{1, 2\}, \{1, 3\}, \{1, 5\}, \{2, 3\}, \{2, 5\}, \{3, 5\}\}.$$

Let us now apply Step 5, which consists in removing all non supported itemsets from C_2 to define L_2 . To do so we apply Algorithm 7. So, first we have to remove all elements in C_2 with subsets not in L_1 . In our case, there is no such set as all subsets of itemsets in C_2 are in L_1 . Then, we have to check in the database if itemsets in C_2 are supported. The following counts are found:

- $Count(\{1, 2\}) = 1$
- $Count(\{1, 3\}) = 2$
- $Count(\{1, 5\}) = 1$
- $Count(\{2, 3\}) = 2$
- $Count(\{2, 5\}) = 3$
- $Count(\{3, 5\}) = 2$

As $thr - s = 2$, we have

$$L_2 = \{\{1, 3\}, \{2, 3\}, \{2, 5\}, \{3, 5\}\}.$$

From this set, we compute C_3 obtaining

$$C_3 = \{\{1, 2, 3\}, \{1, 2, 5\}, \{1, 3, 5\}, \{2, 3, 5\}\}.$$

Then, L_3 will contain only the itemsets from C_3 that are supported using Algorithm 7. Step 3 in this algorithm removes $\{1, 2, 3\}$ and $\{1, 2, 5\}$ as $\{1, 2\}$ is not supported, $\{1, 3, 5\}$ as $\{1, 5\}$ is not supported and only $\{2, 3, 5\}$ remains. Then, in Step 7 we check in the database if $\{2, 3, 5\}$ is supported and as this is so ($Count(\{2, 3, 5\}) = 2$) we get

$$L_3 = \{2, 3, 5\}.$$

Then, in the next step we compute C_4 , but this is empty, so the algorithm finishes and we have obtained the following sets:

- $L_1 = \{\{1\}, \{2\}, \{3\}, \{5\}\},$
- $L_2 = \{\{1, 3\}, \{2, 3\}, \{2, 5\}, \{3, 5\}\},$
- $L_3 = \{\{2, 3, 5\}\}.$

Therefore, the algorithm returns these sets from which rules will be generated.

2.3.3 Expectation-Maximization Algorithm

In this section we describe the EM algorithm, where EM stands for expectation-maximization. The algorithm looks for maximum likelihood estimates. We define in this section first likelihood estimate and then the EM algorithm.

2.3.3.1 Likelihood Function and Maximum Likelihood

The *maximum likelihood* is a method for estimating the parameters of a given probability density. Let us consider a probability density $f(z|\theta)$. That is, f is a parametric model of the random variable z with parameter θ (or, parameters, because θ can be a vector). Let $\mathbf{z} = \{z_1, \dots, z_e\}$ be a sample of the variable z . Then, the *likelihood* of \mathbf{z} under a particular model $f(z|\theta)$ is expressed by:

$$f(\mathbf{z} = (z_1, \dots, z_e)|\theta) = \prod_{i=1}^e f(z_i|\theta)$$

That is, $f(\mathbf{z}|\theta)$ is the probability of the sample \mathbf{z} under the particular model $f(z_i|\theta)$ with a particular parameter θ . The likelihood function is the function above when the sample is taken as constant and θ is the variable. This is denoted by $L(\theta|\mathbf{z})$. Thus,

$$L(\theta|\mathbf{z}) = \prod_{i=1}^e f(z_i|\theta)$$

Often, the *log-likelihood* function is used instead of the likelihood function. The former is the logarithm of the latter and is denoted by $l(\theta|\mathbf{z})$ or, sometimes, by $l(\theta)$. Therefore,

$$l(\theta|\mathbf{z}) = \log L(\theta|\mathbf{z}) = \log \prod_{i=1}^e f(z_i|\theta) = \sum_{i=1}^e \log f(z_i|\theta)$$

Given a sample \mathbf{z} and a model $f(\mathbf{z}|\theta)$, the maximum likelihood estimate of the parameter θ is the $\hat{\theta}$ that maximizes $l(\theta|\mathbf{z})$. Equivalently, the estimate is $\hat{\theta}$ such that

$$l(\theta|\mathbf{z}) \leq l(\hat{\theta}|\mathbf{z})$$

for all θ .

2.3.3.2 EM Algorithm

The EM algorithm [49] (where EM stands for *Expectation-Maximization*) is an iterative process for the computation of maximum likelihood estimates. The method starts with an initial estimation of the parameters and then in a sequence of two step iterations builds more accurate estimations. The two steps considered are the so-called Expectation step and Maximization step.

The algorithm is based on the consideration of two sample spaces \mathcal{Y} and \mathcal{X} and a many-to-one mapping from \mathcal{X} to \mathcal{Y} . We use y to denote this mapping, and $X(y)$ to denote the set $\{x|y = y(x)\}$. Only data y in \mathcal{Y} are observed, and data x in \mathcal{X} are only observed indirectly through y . Due to this, x are referred to as complete data and y as the observed data.

Let $f(x|\theta)$ be a family of sampling densities for x with parameter θ , it is clear that the corresponding family of sampling densities $g(y|\theta)$ can be computed from $f(x|\theta)$ as follows:

$$g(y|\theta) = \int_{X(y)} f(x|\theta) dx$$

Now, roughly speaking, the expectation step consists on estimating the complete data x and the maximization step consists on finding a new estimation of the

parameters θ by maximum likelihood. In this way, the EM algorithm tries to find the value θ that maximizes $g(y|\theta)$ given an observation y . However, the method also uses $f(x|\theta)$.

References

1. Babbage, C.: Passages from the life of a philosopher, Longman, Green, Longman, Roberts & Green (1864)
2. Breiman, L.: Statistical modeling: the two cultures. *Stat. Sci.* **16**(3), 199–231 (2001)
3. Friedman, J.H.: Data mining and statistics: what's the connection? (1997). <http://www-stat.stanford.edu/~jhf/ftp/dm-stat.pdf>
4. <http://brenocon.com/blog/2008/12/statistics-vs-machine-learning-fight/>. Accessed Jan 2017
5. <http://normaldeviate.wordpress.com/2012/06/12/statistics-versus-machine-learning-5-2/>. Accessed Jan 2017
6. <http://stats.stackexchange.com/questions/1521/data-mining-and-statistical-analysis>. Accessed Jan 2017
7. <http://stats.stackexchange.com/questions/6/the-two-cultures-statistics-vs-machine-learning>. Accessed Jan 2017
8. Geisser, S.: Predictive Inference: An Introduction. CRC Press (1993)
9. Srivastava, M.S.: Minimum distance classification rules for high dimensional data. *J. Multivar. Anal.* **97**(9), 2057–2070 (2006)
10. Yata, K., Aoshima, M.: Effective PCA for high-dimension, low-sample-size data with noise reduction via geometric representations. *J. Multivar. Anal.* **105**(1), 193–215 (2012)
11. Yata, K., Aoshima, M.: Correlation tests for high-dimensional data using extended cross-data-matrix methodology. *J. Multivar. Anal.* **117**, 313–331 (2013)
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2009)
13. Witten, I.H., Frank, E., Hall, M.A.: Data Mining. Elsevier (2011)
14. Ryan, T.P.: Modern Regression Methods. Wiley (1997)
15. Dubes, R., Jain, A.K.: Clustering techniques: the user's dilemma. *Pattern Recogn.* **8**, 247–260 (1976)
16. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs (1988)
17. Berend, D., Tassa, T.: Improved bounds on bell numbers and on moments of sumes of random variables. *Probab. Math. Stat.* **30**(2), 185–205 (2010)
18. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley (1990)
19. Torra, V., Endo, Y., Miyamoto, S.: On the comparison of some fuzzy clustering methods for privacy preserving data mining: towards the development of specific information loss measure. *Kybernetika* **45**(3), 548–560 (2009)
20. Torra, V., Endo, Y., Miyamoto, S.: Computationally intensive parameter selection for clustering algorithms: the case of fuzzy c-means with tolerance. *Int. J. Intel. Syst.* **26**(4), 313–322 (2011)
21. Zadeh, L.A.: Fuzzy sets. *Inf. Control* **8**, 338–353 (1965)
22. Bezdek, J.C.: The parable of Zoltan. In: Seising, R., Trillas, E., Moraga, C., Termini, S. (eds.) *On Fuzziness: Volume 1 (STUDFUZZ 298)*, pp. 39–46. Springer (2013)
23. Ruspini, E.H.: A new approach to clustering. *Inf. Control* **15**, 22–32 (1969)
24. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
25. Miyamoto, S.: Introduction to Fuzzy Clustering (in Japanese), ed. Morikita, Japan (1999)

26. Miyamoto, S., Ichihashi, H., Honda, K.: Algorithms for Fuzzy Clustering. Springer (2008)
27. Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. Wiley (1999)
28. Miyamoto, S., Mukaidono, M.: Fuzzy c -means as a regularization and maximum entropy approach. In: Proceedings of the 7th International Fuzzy Systems Association World Congress, IFSA 1997, vol. II, pp. 86–92 (1997)
29. Miyamoto, S., Umayahara, K.: Fuzzy c -means with variables for cluster sizes (in Japanese). In: 16th Fuzzy System Symposium, pp. 537–538 (2000)
30. Ichihashi, H., Honda, K., Tani, N.: Gaussian mixture PDF approximation and fuzzy c -means clustering with entropy regularization. In: Proceedings of the 4th Asian Fuzzy System Symposium, 31 May–3 June, Tsukuba, Japan, pp. 217–221 (2000)
31. Davé, R.N.: Characterization and detection of noise in clustering. Pattern Recogn. Lett. **12**, 657–664 (1991)
32. Krishnapuram, R., Keller, J.M.: A possibilistic approach to clustering. IEEE Trans. Fuzzy Syst. **1**, 98–110 (1993)
33. Barni, M., Cappellini, V., Mecocci, A.: Comments on “a possibilistic approach to clustering”. IEEE Trans. Fuzzy Syst. **4**(3), 393–396 (1996)
34. Ladra, S., Torra, V.: On the comparison of generic information loss measures and cluster-specific ones. Int. J. Uncertainty Fuzziness Knowl. Based Syst. **16**(1), 107–120 (2008)
35. Pal, N.R., Pal, K., Bezdek, J.C.: A mixed c -means clustering model. In: Proceedings of the 6th IEEE International Conference on Fuzzy Systems, pp. 11–21 (1997)
36. Pal, N.R., Pal, K., Keller, J.M., Bezdek, J.C.: A possibilistic fuzzy c -means clustering algorithm. IEEE Trans. Fuzzy Syst. **13**(4), 517–530 (2005)
37. Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**(336), 846–850 (1971)
38. Hubert, L.J., Arabie, P.: Comparing partition. J. Classif. **2**, 193–218 (1985)
39. Anderson, D.T., Bezdek, J.C., Popescu, M., Keller, J.M.: Comparing fuzzy, probabilistic, and possibilistic partitions. IEEE Trans. Fuzzy Syst. **18**(5), 906–918 (2010)
40. Albatineh, A.N., Niewiadomska-Bugaj, M., Mihalko, D.: On similarity indices and correction for chance agreement. J. Classif. **23**, 301–313 (2006)
41. Morey, L., Agresti, A.: The measurement of classification agreement: an adjustment to the rand statistic for chance agreement. Educ. Psychol. Meas. **44**, 33–37 (1984)
42. López de Mántaras, R.: A distance-based attribute selection measure for decision tree induction. Mach. Learn. **6**, 81–92 (1991)
43. Hüllermeier, E., Rifqi, M.: A fuzzy variant of the Rand index for comparing clustering structures. In: Proceedings, IFSA/EUSFLAT (2009)
44. Brouwer, R.K.: Extending the Rand, adjusted Rand and jaccard indices to fuzzy partitions. J. Intell. Inf. Syst. **32**, 213–235 (2009)
45. Hüllermeier, E., Rifqi, M., Henzgen, S., Senge, R.: Comparing fuzzy partitions: a generalization of the rand index and related measures. IEEE Trans. Fuzzy Syst. **20**(3), 546–556 (2012)
46. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on VLDB, pp. 478–499. Also as research report RJ 9839, IBM Almaden Research Center, San Jose, California, June 1994
47. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216 (1993)
48. Mannila, H., Toivonen, H., Verkamo, A.I.: Efficient algorithms association for discovering rules. AAAI technical report WS-94-03. <http://www.aaai.org/Papers/Workshops/1994/WS-94-03/WS94-03-016.pdf> (1994)
49. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc. **39**, 1–38 (1977)

Data Privacy: Foundations, New Developments and the
Big Data Challenge

Torra, V.

2017, XIV, 269 p. 22 illus., Hardcover

ISBN: 978-3-319-57356-4