UNIVERSITÀ DEGLI STUDI DI CATANIA
FACOLTÀ DI SCIENZE MM. FF. NN.
DIPARTIMENTO DI MATEMATICA E INFORMATICA

DOTTORATO DI RICERCA IN INFORMATICA

Tesi di Dottorato

GEOPHYSICAL TIME SERIES DATA MINING

Dottorando:
Carmelo Cassisi

Tutor:
Alfredo Pulvirenti
Placido Montalto

ANNO ACCADEMICO 2011-2012

A Maria Luisa,
e alla mia famiglia

Acknowledgements

It is a pleasure to thank the many people who made this thesis possible.

I would like to express my deep and sincere gratitude to my supervisors, Dr. Alfredo Pulvirenti, Department of Computer Science, University of Catania, and Dr. Placido Montalto, INGV Section of Catania – Osservatorio Etneo. Their wide knowledge and their logical way of thinking have been of great value for me. Their understanding, encouraging and personal guidance have provided a good basis for the present thesis.

I warmly thank Dr. Alfredo Pulvirenti for teaching me the discipline of data mining and for introducing me to the world of research and work by accepting me as a Ph.D. student. I am grateful to other people of the Department of Computer Science, University of Catania, who supported me in research: Prof. Alfredo Ferro and Dr. Rosalba Giugno, for their encouragement and insightful comments.

My sincere thanks go to Dr. Placido Montalto for the continuous support of my Ph.D. study and research, for his patience, motivation and enthusiasm. His guidance helped me in all the time of research and writing of this thesis. Besides him, I warmly thank two special INGV colleagues, who helped me in my professional growth: Dr. Andrea Cannata and Dr. Marco Aliotta. For their kind assistance, for providing a stimulating and fun environment in which to learn and grow.

During this work I collaborated with many colleagues for whom I have great regard, and I wish to extend my warmest thanks to all those who have helped me with my work: Dr. Michele Prestifilippo, Dr. Deborah Lo Castro, Dr. Daniele Andronico and Dr. Letizia Spampinato.

The financial support of the Istituto Nazionale di Geofisica e Vulcanologia (INGV) Section of Catania – Osservatorio Etneo is gratefully acknowledged. A special thanks goes to the director Dr. Domenico Patanè for giving me the opportunity to work with an organization of such a prestige.

Lastly, and most importantly, I want to thank my girlfriend Maria Luisa for always believing in me and for all time spent together, advising me in every important decision, and my family, especially my parents Rosario and Donatella, which always supported me in any area of my life. I wish to thank Giuseppe and Fulvia, my friends and all relatives for all the emotional support, camaraderie, entertainment, and caring they provided. I thank also all other saints for supporting me spiritually throughout my life. In a special way my warm thanks go to Gianni and Gioacchino Paolo. All these people raised me, supported me, taught me, and loved me. To them I dedicate this thesis.

But above all, I praise God, the almighty for providing me all I need and granting me the capability to proceed successfully.

Preface

The process of automatic extraction, recognition, description and classification of patterns from huge amount of data plays an important role in modern volcano monitoring techniques. In particular, the ability of certain systems to recognize different volcano status can help the researchers to better understand the complex dynamics underlying the geophysical system. The geophysical data are automatically measured and recorded by geophysical instruments. Their interpretation is very important for the investigation of earth's behavior.

The fundamental task of volcano monitoring is to follow volcanic activity and promptly recognize any changes. To achieve such goals, different geophysical techniques (i.e. seismology, ground deformation, remote sensing, magnetic and electromagnetic studies, gravimetric) are used to obtain precise measurements of the variations induced by an evolving magmatic system. To proper exploit the wealth of such heterogeneous data, algorithms and techniques of data mining are fundamental tools. This thesis can be considered a detailed report about the application of the data mining discipline in the geophysical area. After introducing the basic concepts and the most important techniques constituting the state-of-art in the data mining field, we will apply several methods able to reach important results about the extraction of unknown recurrent patterns in seismic and infrasonic signals, and we will show the implementation of systems representing efficient tools for the monitoring purpose.

The thesis is organized as follows. Chapter 1 briefly introduces to the data mining discipline; Chapter 2 discusses the similarity matching problem, explaining the importance of using efficient data structures to perform search, and the choice of adequate distance measures. It also lists the most common similarity/distance measures used for data mining, devoting a deepening part for time series similarity. Chapter 3 reviews the state-of-art dimensionality reduction techniques for the summarization of time series in data mining. Chapter 4 provides some basic principles on supervised classification, while Chapter 5 analyzes main clustering methods, for the unsupervised classification task, together with some recent developments.

A broad range of data mining applications will be devoted in Chapter 6, where the classification and prediction tasks are applied on geophysical data.

Contents

Chapter 1

Introduction

Data mining has a very recent origin and has no single definition. Various definitions have been proposed:

"Data mining is the search for relationships and global patterns that exist in large databases, but are 'hidden' among the vast amounts of data, such as a relationship between patient data and their medical diagnosis. These relationships represent valuable knowledge about the database and objects in the database and, if the database is a faithful mirror, of the real world registered by the database" (Holsheimer and Siebes, 1994).

"Data mining is the exploration and analysis, by means of automatic and semi-automatic methods, of large amounts of data in order to discover meaningful patterns and rules" (Berry et al., 1997).

Data mining can be considered as the 'art' of knowledge extraction from huge amount of data. The term is often used as a synonym for *Knowledge Discovery in Databases* (*KDD*):

"Knowledge discovery is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" (Frawley et al., 1992).

It would be more accurate to speak of knowledge discovery when referring to the process of knowledge extraction, and data mining as a particular phase of this process, consisting in the application of specific algorithms for the identification of "patterns" (Fayyad et al., 1996).
For example, in an industrial or operative domain, useful knowledge is hidden but relevant information. Today the main problem of analysts is to be capable to properly extract the wealth of information which is intrinsically present in the data. Starting from the Fayyad et al. (1996)

considerations, the extraction process consists of several phases, each of which brings its rate of information (Figure 1.1): selection from raw data; pre-processing; transformation; application of algorithms for patterns search (in this context a "pattern" means a structure, a model, or, in general, a synthetic representation of the data), followed by their interpretation and evaluation. The identified patterns can be considered, in turn, the starting point to speculate and to verify new relationships among phenomena. They also can be useful to make predictions on new data sets.

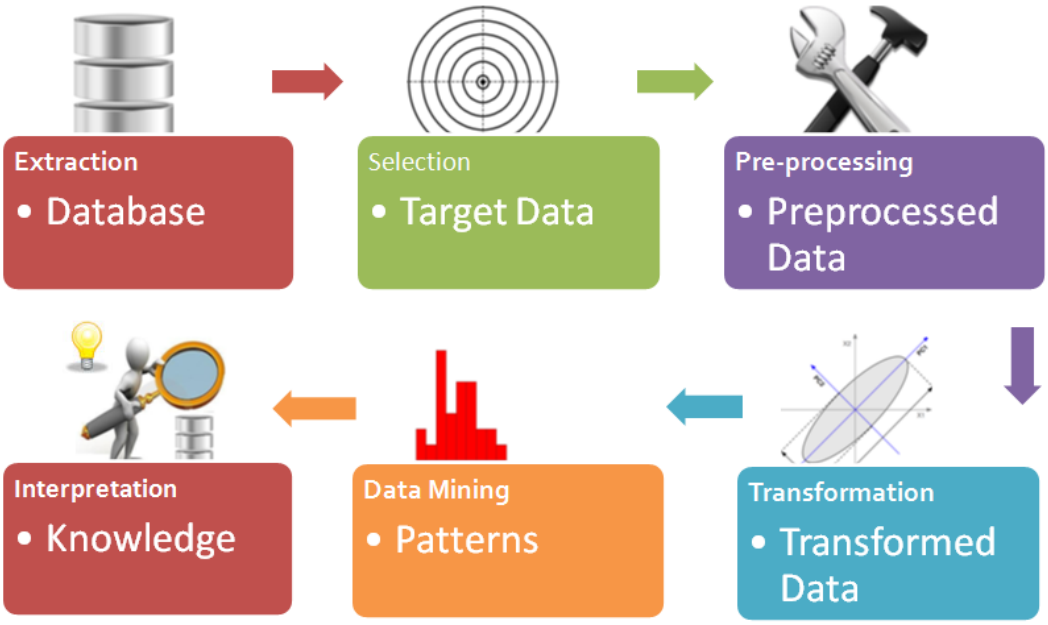


Fig. 1.1. General schema of Knowledge Discovery in Database (KDD) process (redrawn from Fayyad et al., 1996).

Data mining is a multidisciplinary field which borrows algorithms and techniques from many research areas such as: machine learning, statistics, neural networks, artificial intelligence, high performance computing technology, database technology, data visualization techniques. The main factors contributing to data mining progress are: the increasing of electronic data, the cheap data storage, and new techniques for analysis (machine learning, pattern recognition).

Algorithms are pillars of data mining techniques, and have to guarantee the effectiveness and efficiency of analysis. Scalability is a fundamental property, because data mining deals with huge amounts of data, and

algorithms' implementation must provide high speed computation, faster than manual data analysis. The execution time of a data mining algorithm must be at least predictable and acceptable according to the size of the analyzed database. Data mining algorithms have to adapt to the hardware advances, trying to properly exploit its potential, such as the computing on multiple processors (parallel or distributed computing), and then afford the problems inherent to the database size.

Data mining is an interdisciplinary branch of the science which takes its origins in statistics. The reason behind the wide usage of data mining techniques relies on its simplicity (w.r.t. classical statistical methods), its scalability and its wide range application domains. In principle any kind of data can be analyzed with proper mining technique. However, data mining has similar limits to all statistical approaches, such as the *GIGO* (*Garbage In, Garbage Out*): if the input is "garbage", then the output will be "garbage". Thus, the optimal strategy is to use statistics and data mining as two complementary approaches.

The data mining techniques can be divided into two broad categories: supervised and unsupervised learning techniques.

The supervised learning aims to realize a computer system able to automatically solving a specific trained problem. In order to produce a generalized model for the problem, a supervised learning algorithm makes use of some examples, in particular: (i) it defines a set of input data I; (ii) it defines a range of output values O; (iii) and defines a function h that maps each input data to correct output value. Providing a large number of examples, the algorithm of supervised learning will be able to identify a new function h_1 that will approximate the function h. Of course, the goodness of the algorithm depends on the dataset used in the training phase. In fact, it has to avoid the "overtraining" on input: a model is considered to be good if it is able to predict the output of never learned data, only with the knowledge provided by the input. It may happen in fact that the model specializes only on the recognition of the sample of the input data, producing the "overfitting" problem. Through the supervised learning, data mining is able to face problems concerning: (i) classification operations, where observations are "labeled" (or associated to a class) on the basis of well-known characteristics of the class; (ii) model estimation, such as to see whether the distribution of an observation follows a

statistical known model, such as the Gaussian distribution; (iii) attempt to identify future trends (prediction) of an observed variable or characteristic.
The unsupervised learning builds models without apriori knowledge. Different from supervised learning, in the learning phase, the labels or the model of the examples are not provided. The algorithms are based on comparisons between data and the search for similarities and differences. In this contest can be realized: (i) operations of grouping (or clustering), by finding homogeneous groups that present characteristical regularities within them and differences among groups (clusters); (ii) association rules, which identify any associations between data. These last are widely applied in the transactional database to find relationships between products purchased together, as in the case of the market basket analysis, to implement marketing strategies, such as promotional offers, or the positioning of the products on the shelves.

1.1 Typical data mining tasks

Given a collection of objects, a database *D*, most of data mining research is related to the similarity matching problem, including the following tasks:

- **Indexing:** given a query object *Q*, and a similarity/dissimilarity measure *dist*(*o*,*p*) defined for $\forall o, p \in D$, it consists on building a data structure, allowing speed-up search of the nearest neighbor of *Q* in *D*. There are two ways to post a similarity query [3]:
  - **k-nearest neighbors**: dealing with the search of the set of first *k* objects *D* more similar to *Q*.
  - **range query**: finds the set $R \subseteq D$ of objects that are within distance *r* from *Q*.

- **Clustering:** consists of division of data into groups (*clusters*) of similar objects under some similarity/dissimilarity measure. The search for clusters is unsupervised. It is often complementary to the anomaly/outlier detection problem, which seeks for objects showing different attributes respect to the whole dataset.

- **Classification:** assigns unlabeled objects to predefined classes after a supervised learning process, based on classes properties.

1.2 Time series data mining

In the last years, there has been an increasing interest in methods dealing with time series data. It depended on the rapid growth of generated daily information from several areas, e.g., finance, computational biology, sensor networks, location-based services, etc. A time series is *"a sequence X* = (x_1, x_2, …, x_m) *of observed data over time*", where *m* is the number of observations. Tracking the behavior of a specific phenomenon/data in time can produce important information (Fig. 1.2). A large variety of real world applications, such as meteorology, geophysics and astrophysics, collect observations that can be represented as time series.

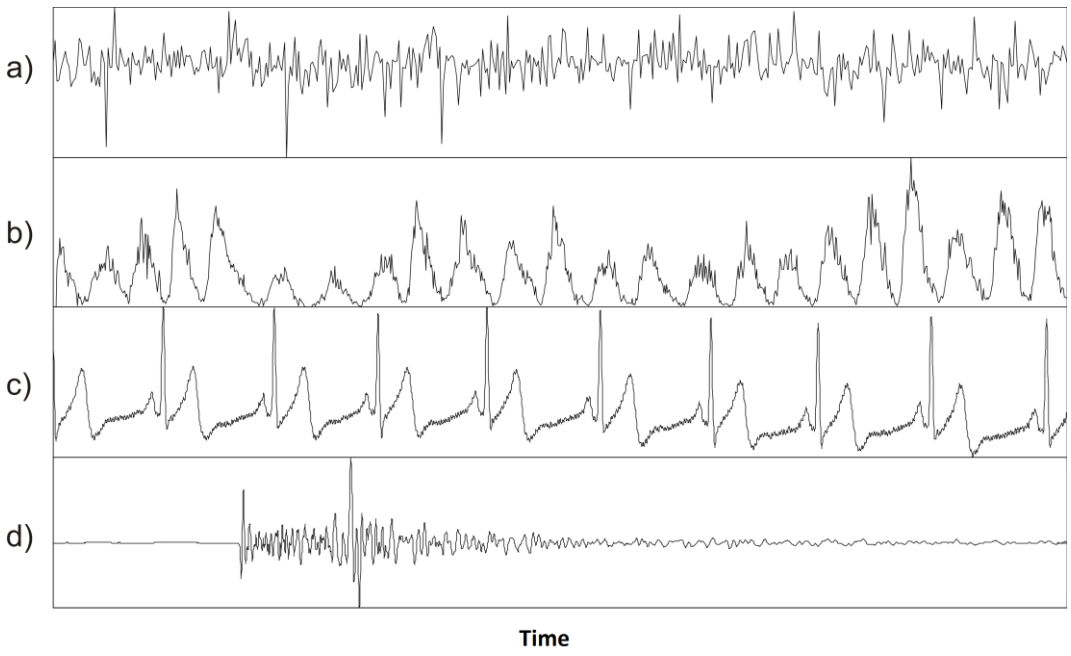


Fig. 1.2. Examples of time series data relative to a) monsoon, b) sunspots, c) ECG (ElectroCardioGram), d) seismic signal.

Often in time series data mining these other two following tasks are relevant:

- **Summarization:** given a raw time series Q of length m, makes a new time series representation Q' lighter than Q (with length $m' < m$) in term of space, that approximate Q fitness. This task is very important to achieve the best performances on previous tasks.
- **Prediction**: finds a model for the sequence of observations, to check dependencies among them, and then predict a future value. While in the classification task an observation is assigned to a specific class, providing an output of categorical, the prediction task provides a specific observation value. It can be also used to replace missing values on data.

1.3 Data mining on geophysics

The geophysical data are automatically measured and recorded by geophysical instruments. Their interpretation is very important for the investigation of earth's behavior. Generally, the amount of data is very large and relatively standard. It is suitable to be processed and be analyzed by data mining techniques. The analysis can be conducted on real time data, or on historical data.
The former task is the most interesting type of analysis, because permits to monitor alert situations and to prevent most of human risks: this thesis focuses on volcano monitoring. The latter task often consists of extracting previously unknown recurrent patterns from available data, and constitutes a crucial step in geophysical time series analysis, because allow to increment the suitable amount of information for the monitoring task.

Chapter 2

Fundamentals on similarity matching

In data mining, several terms are used to refer to a single data into database: *object*, *point*, *record, observation*, *item* or *tuple*. In this chapter, we will use the term *object* to denote a single element of a dataset. In multi-dimensional spaces, an object is described by a number of *components* or *features*, which we will refer as *attributes*.

More formally, a dataset with N objects, each of which is described by m attributes, is denoted by $D = \{x_1, x_2, \ldots, x_N\}$, where $x_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(m)})$ is a vector denoting the ith object and $x_i^{(j)}$ is a value denoting the jth attribute of x_i. The number of attributes m indicates the *dimensionality* of the data set. The general representation of such data is a matrix $N \times m$ used by most of the algorithms described below.

$$D = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(j)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(j)} & \cdots & x_2^{(m)} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_i^{(1)} & x_i^{(2)} & \cdots & x_i^{(j)} & \cdots & x_i^{(m)} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_N^{(1)} & x_N^{(2)} & \cdots & x_N^{(j)} & \cdots & x_N^{(m)} \end{pmatrix} \tag{2.1}$$

2.1 Types of data

Data mining algorithms strongly depend on attributes of managed data types. A basic classification distinguishes two main categories of attributes: *quantitative* and *qualitative*. Quantitative attributes come from numeric measurements, and can represent *continuous* values (e.g. height), and *discrete* values (e.g. number of children). For quantitative attribute, there is another distinction between *interval* attributes, where measurements are disposed on a linear scale, and *ratio* attributes, which are disposed on a nonlinear scale.

Qualitative attributes come from previously established categories, and can be *categorical* (or *nominal*; e.g. eye's color), *binary* (e.g. sex), and *ordinal* (e.g. military rank). Binary attribute is a special case of categorical attribute taking exactly two categories: it is possible to give same weight for both categories (*symmetric*) or not (*asymmetric*). It is possible to realize a map from interval to ordinal and nominal attributes, or from ordinal to nominal, and vice-versa (Gan et al., 2007).
In the real world, however, there exist various other data types, such as image data, graph, or time series containing, in turn, quantitative and/or qualitative data.

2.2 Indexing

In many cases, datasets are supported by special data structures, especially when dataset get larger, that are referred as *indexing* structures. Indexing consists of building a data structure I that enables efficient searching within database (Ng and Cai, 2004). Usually, it is designed to face two principal similarity queries: the (i) *k-nearest neighbors* (*knn*), and the (ii) *range query* problem. Given a query object Q in D, and a similarity/dissimilarity measure $d(x,y)$ defined for each pair x, y in D, the former query deals with the search of the set of first k objects in D more similar to Q. The latter query finds the set R of objects that are within distance r from Q. When dealing with a collection of time series, a *TSDB* (*Time Series DataBase*), given an indexing structure I, there are two ways to post a similarity query (Ng and Cai, 2004):

- *whole matching*: given a *TSDB* of time series, each of length m, whole matching relates to computation of similarity matching among time series along their whole length.

- *subsequence matching*: given a *TSDB* of N time series S_1, S_2, …, S_N, each of length m_i, and a short query time series Q of length $m_q < m_i$, with $0 < i < N$, subsequence matching relates to finding matches of Q into subsequences of every S_i, starting at every position.

Indexing is crucial for reaching efficiency on data mining tasks, such as *clustering* or *classification*, especially for huge database such as *TSDB*s. Clustering is related to the unsupervised division of data into groups (clusters) of similar objects under some similarity or dissimilarity (*distances*) measures. Sometimes, on time series domain, a similar problem to clustering is the *motif discovery* problem (Mueen et al., 2009), consisting of searching main cluster (or *motif*) into a *TSDB*. The search for clusters is unsupervised. Classification assigns unlabeled objects to predefined classes after a supervised learning process. Both tasks make massive use of distance computations.

Distance measures play an important role in similarity matching problem. Concerning a distance measure, it is important to understand if it can be considered *metric* function. A metric function on a set D is a function $f : D \times D \rightarrow \mathrm{R}$ (where R is the set of real numbers). For all x, y, z in D, this function obeys to four fundamental properties:

1. $f(x, y) \geq 0$ (*non-negativity*) (2.2)
2. $f(x, y) = 0$ if and only if $x = y$ (*identity*) (2.3)
3. $f(x, y) = f(y, x)$ (*symmetry*) (2.4)
4. $f(x, z) \leq f(x, y) + f(y, z)$ (*triangle inequality*) (2.5)

If any of these is not obeyed, the distance is considered non-metric. Using a metric function is desired, because the triangle inequality property (Eq. 2.5) can be used to perform the indexing of the space for speed-up search in large datasets. By means of adequate indexing data structures (general tree structures) such as *kd*-tree (Bentley, 1975), *R** tree (Beckmann et al., 1990), or *Antipole* tree (Cantone et al., 2005), it is possible to perform pruning during search on the space. Best efforts have been devoted to similarity searching, with emphasis on metric space searching. In this sense *SISAP*, a conference devoted to similarity searching (http://sisap.org/Home.html) provides the *Metric Space Library* (http://sisap.org/Metric_Space_Library.html) allowing to use a wide range of indexing techniques for general spaces (metric and non-metric). Another well known framework for indexing, overall for multimedia and time series data, is *GEMINI* (*GEneric Multimedia INdexIng*; Faloutsos et al.,

1994), that designs fast search algorithms for locating objects series that match, in an exact or approximate way, a query time series Q.
Algorithms dealing with relative small datasets mostly use a simple data structure, an $N \times N$ matrix, called also *distance matrix*, *proximity matrix*, or *affinity matrix*, storing distances between each pair of dataset objects:

$$distMatrix = \begin{pmatrix} 0 & & & & & \\ d(2,1) & 0 & & & & \\ d(3,1) & d(3,2) & \ddots & & & \\ \vdots & \vdots & \cdots & 0 & & \\ \vdots & \vdots & \cdots & \vdots & 0 & \\ d(N,1) & d(N,2) & \cdots & \cdots & d(N,N\ \ 1) & 0 \end{pmatrix} \tag{2.6}$$

where $d(i,j)$ indicates distance between ith and jth object (for $0 < i, j < N$). For metric distance functions, the matrix is symmetric, because of the metric symmetric property (Eq. 2.4), and all diagonal elements have zero values, since $d(i,i) = 0$ for the identity property (Eq. 2.3). If we store similarity measures instead of distances the resulting matrix will be called *similarity* matrix.

2.3 Similarity and Distance Measures

A common data mining task is the estimation of similarity among objects. A similarity measure is a relation between a pair of objects and a scalar number. In this subsection some of the common distance measures, used for numerical data and not, are formally described. Let be two objects X and Y of m attributes, and x_i and y_i the ith attributes of X and Y, respectively. Let us list the following measures.

2.3.1 Numerical Similarity Measures

Common intervals used to mapping the similarity are [-1, 1] or [0, 1], where 1 indicates the maximum of similarity.
Considering the similarity between two attributes x_i and y_i as :

$$numSim(x_i, y_i) = 1 - \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{2.7}$$

Mean Similarity

$$tsim(X,Y) = \frac{1}{m}\sum_{i=1}^{m} numSim(x_i, y_i) \tag{2.8}$$

Root Mean Square Similarity:

$$rtsim(X,Y) = \sqrt{\frac{1}{m}\sum_{i=1}^{m} numSim(x_i, y_i)^2} \tag{2.9}$$

Peak similarity (Fink and Pratt, 2004):

$$psim(X,Y) = \frac{1}{m}\sum_{i=1}^{m}\left[1 - \frac{|x_i - y_i|}{2\max(|x_i|,|y_i|)}\right] \tag{2.10}$$

Cosine similarity. In some applications, such as information retrieval, text document clustering, and biological taxonomy, it is possible that the measures mentioned above are not used. Often, when dealing with vector objects, the most used distance function is the *cosine similarity*.
The *cosine similarity* computes the cosine of the angle θ between two objects X and Y, and is defines as:

$$\cos(\theta) = \frac{\sum_{i=1}^{m} x_i y_i}{\sqrt{\sum_{i=1}^{m}(x_i)^2}\sqrt{\sum_{i=1}^{m}(y_i)^2}} \tag{2.11}$$

This measure provides values in range [-1, 1]. The lower boundary indicates that the X and Y vectors are exactly opposite; the upper boundary indicates that the vectors are exactly the same; finally the 0 value indicates the independence.

Cross-correlation. Another common similarity function used to perform complete or partial matching between time series is the *cross-correlation* function (or *Pearson's correlation* function) (Von Storch and Zwiers, 2001).

The cross correlation between two time series X and Y of length m, allowing a shifted comparison of l positions, is defined as:

$$r_{XY} = \frac{\sum_{i=1}^{m}\left(x_i - \overline{X}\right)\left(y_{i-l} - \overline{Y}\right)}{\sqrt{\sum_{i=1}^{m}\left(x_i - \overline{X}\right)^2}\sqrt{\sum_{i=1}^{m}\left(y_{i-l} - \overline{Y}\right)^2}} \tag{2.12}$$

where $\overline{X}$ and $\overline{Y}$ are the means of X and Y. The correlation r_{XY} provides the degree of linear dependence between the two vectors X and Y from perfect linear relationship ($r_{XY} = 1$), to perfect negative linear relation ($r_{XY} = -1$).

2.3.2 Numerical Distance Measures

Euclidean Distance. The most used distance function in many applications. It is defined as:

$$d(X,Y) = \left[\sum_{i=1}^{m}(x_i - y_i)^2\right]^{\frac{1}{2}} \tag{2.13}$$

Manhattan Distance. Also called "city block distance". It is defined as:

$$d(X,Y) = \sum_{i=1}^{m}|x_i - y_i| \tag{2.14}$$

Maximum Distance. It is defined to be the maximum value of the distances of the attributes:

$$d(X,Y) = \max_{0<i<m}|x_i - y_i| \tag{2.15}$$

Minkowski Distance. The *Euclidean* distance (Eq. 2.13), *Manhattan* distance (Eq. 2.14), and *Maximum* distance (Eq. 2.15), are particular instances of the *Minkowski* distance, called also L_p*-norm*. It is defined as:

$$d(X,Y) = \left[\sum_{i=1}^{m}(x_i - y_i)^p\right]^{\frac{1}{p}} \tag{2.16}$$

where p is called the order of *Minkowski* distance. In fact, for Manhattan distance $p = 1$, for the Euclidean distance $p = 2$, while for the Maximum distance $p = \infty$.

Mahalanobis Distance. The *Mahalanobis* distance is defined as:

$$d(X,Y) = \sqrt{(x-y)\Sigma^{-1}(x-y)^T} \tag{2.17}$$

where Σ is the *covariance* matrix (see Appendix A, Eq. A.3; Duda et al., 2001).

2.3.3 Binary and Categorical Data Measures

Binary data can have only two values: 0 and 1 (or *true* and *false*, *positive* and *negative*). To compute distance between two data objects X, Y, containing m binary attributes, it is usual to fill a 2 × 2 matrix T, called *contingence table*, which contains all possible test results:

$$T = X\overbrace{\begin{bmatrix} & 1 & 0 & sum \\ 1 & q & r & q+r \\ 0 & s & t & s+t \\ sum & q+s & r+t & m \end{bmatrix}}^{Y} \tag{2.18}$$

where q is the number of attributes that equal 1 for both objects X and Y, r is the number of attributes that equal 1 for object X but that are 0 for object Y, s is the number of attributes that equal 0 for object X but equal 1 for object Y, and t is the number of attributes that equal 0 for both objects X and Y.

For symmetric binary data, where both values have the same weight, it is used a very common distance function, defined as:

$$d(X,Y) = \frac{r+s}{q+r+s+t} = \frac{r+s}{m} \tag{2.19}$$

For asymmetric binary data, by convention, 1 is associated to the weight having more importance. This criterion has most application in medical tests for diseases: positive test, the rarest, have greater significance than negative test. So, it is usual to assign 1 to positive test and 0 to negative test. *t*, in this case, is considered unimportant, and thus is ignored in the computation of the following distance function:

$$d(X,Y) = \frac{r+s}{q+r+s} \tag{2.20}$$

This distance function is often known as *Jaccard* distance (Han and Kamber, 2000).
Categorical data are a generalization of binary data. Let *r* and *s* be two categories of categorical data. A matching between these two categories can be defined in this simple way:

$$\delta(r,s) = \begin{cases} 0 & r \neq s \\ 1 & r = s \end{cases} \tag{2.21}$$

The *Simple Matching* distance for two categorical data *X* and *Y*, described by *m* attributes, can be defined as:

$$d(X,Y) = \sum_{i=1}^{m} \delta(x_i, y_i) \tag{2.22}$$

where x_i and y_i corresponding to the *i*th attributes of *X* and *Y*, respectively.

2.3.4 Measures for Time Series Data

A time series is a sequence of real numbers representing measurements over time. When treating time series, the similarity between two sequences of the same length can be calculated by summing the ordered *point-to-point* distance between them (Fig. 2.1), where "point" stays for a single measurement into time series.

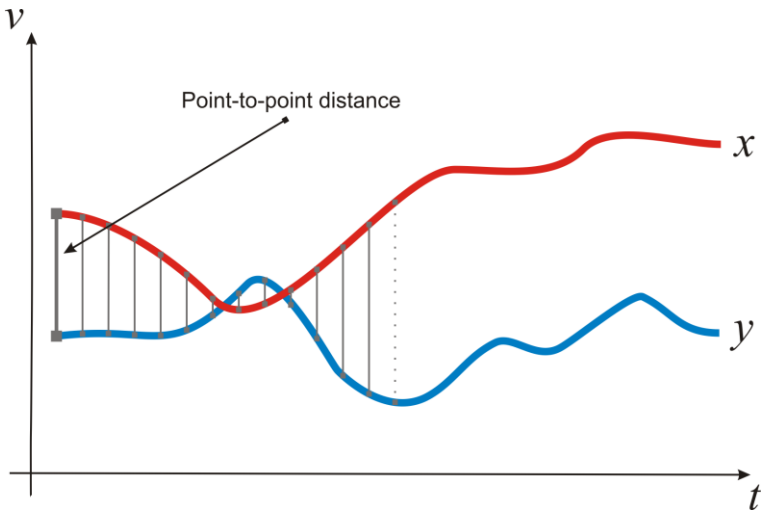


Fig. 2.1. *x* and *y* are two time series of a particular variable *v*, along the time axis *t*. The Euclidean distance results the sum of the *point-to-point* distances (gray lines), along all the time series.

In this sense, the most used distance function is the *Euclidean* distance (Faloutsos et al., 1994), corresponding to the second degree of general L_p-*norm*. This distance measure is cataloged as a metric distance function, since it obeys to the metric properties: *non-negativity*, *identity*, *symmetry* and *triangle inequality* (Eq. 2.3~2.5). *Euclidean* distance is surprisingly competitive with other more complex approaches, especially when dataset size gets larger (Shieh and Keogh, 2008). In every way, Euclidean distance and its variants present several drawbacks, which make inappropriate their use in certain applications:

- It compares only time series of the same length.
- It cannot handle outliers or noise.
- It is very sensitive respect to six signal transformations: shifting, uniform amplitude scaling, uniform time scaling, uniform bi-scaling, time warping and non-uniform amplitude scaling (Perng et al., 2000).

For these reasons, other distance measure techniques were proposed to give more robustness to the similarity computation. In this sense it is required to cite also the well known *Dynamic Time Warping* (*DTW*; Keogh and Ratanamahatana, 2002) taking advantage of dynamic programming to allow comparison of one-to-many points; and the *Longest Common SubSequence* (*LCSS*) similarity measure (Vlachos et al., 2002), a similar

dynamic programming solution as DTW, but more resilient to noise. In the literature there exist other distance measures that overcome signal transformation problems, such as the *Landmarks similarity*, which does not follow traditional similarity models that rely on point-wise Euclidean distance (Perng et al., 2000) but, in correspondence of human intuition and episodic memory, relies on similarity of those points (times, events) of "greatest importance" (for example local maxima, local minima, inflection points). Unfortunately, none of them is metric, so they cannot take advantage of any indexing structure.

Dynamic Time Warping. *Dynamic Time Warping* (Berndt and Clifford, 1994) gives more robustness to the similarity computation. By this method, also time series of different length can be compared, because it replaces the one-to-one point comparison, used in Euclidean distance, with a many-to-one (and viceversa) comparison. The main feature of this distance measure is that it allows recognizing similar shapes, even if they present signal transformations, such as shifting and/or scaling (Fig. 2.2).
Given two time series $T = \{t_1, t_2, \ldots, t_n\}$ and $S = \{s_1, s_2, \ldots, s_m\}$ of length n and m, respectively, an alignment by *DTW* method exploits information contained in an $n \times m$ distance matrix:

$$distMatrix = \begin{pmatrix} d(T_1,S_1) & d(T_1,S_2) & \cdots & d(T_1,S_m) \\ d(T_2,S_1) & d(T_2,S_2) & & \\ \vdots & & \ddots & \\ d(T_n,S_1) & & & d(T_n,S_m) \end{pmatrix} \tag{2.23}$$

where $distMatrix(i, j)$ corresponds to the distance of ith point of T and jth point of S $d(T_i, S_j)$, with $1 \le i \le n$ and $1 \le j \le m$.
The *DTW* objective is to find the *warping path* $W = \{w_1, w_2, \ldots, w_k, \ldots, w_K\}$ of contiguous elements on *distMatrix* (with $\max(n, m) < K < m + n$ -1, and $w_k = distMatrix(i, j)$), such that it minimizes the following function:

$$DTW(T,S) = \min\left(\sqrt{\sum_{k=1}^{K} w_k}\right) \tag{2.24}$$

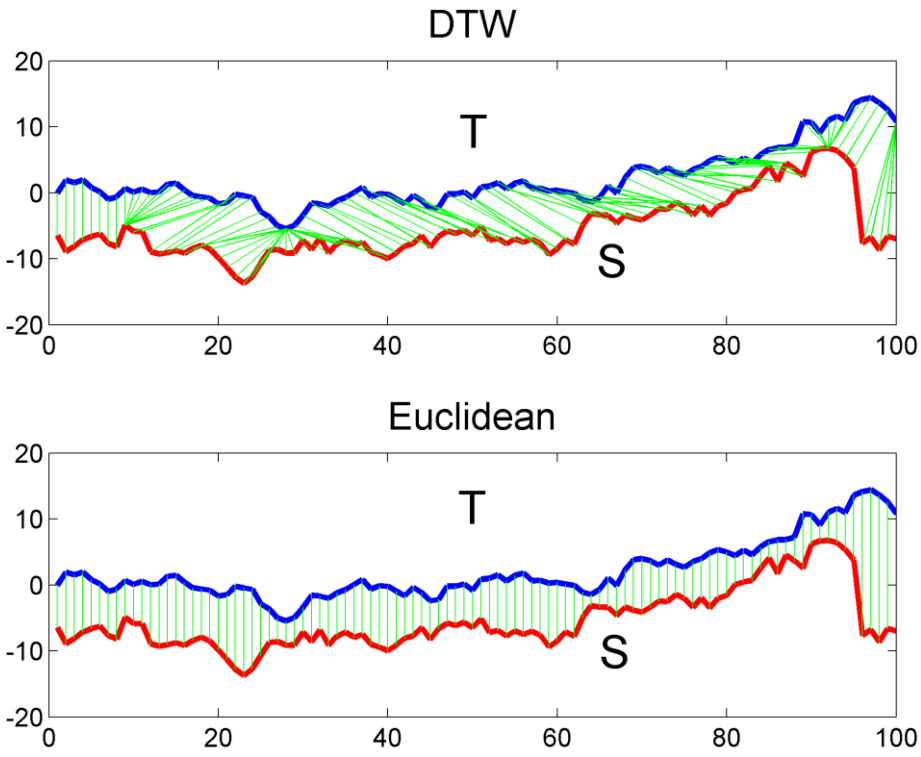


Fig. 2.2. Difference between *DTW* distance and Euclidean distance (green lines represent mapping between points of time series *T* and *S*). The former allows many-to-one point comparisons, while Euclidean point-to-point distance (or one-to-one).

The warping path is subject to several constraints (Keogh and Ratanamahatana, 2002). Given $w_k = (i, j)$ and $w_{k-1} = (i', j')$ with $i, i' \leq n$ and $j, j' \leq m$:

1. ***Boundary conditions***. $w_1 = (1,1)$ and $w_K = (n, m)$.
2. ***Continuity***. $i - i' \leq 1$ and $j - j' \leq 1$.
3. ***Monotonicity***. $i - i' \geq 0$ and $j - j' \geq 0$.

The warping path can be efficiently computed using dynamic programming (Cormen et al. 1990). By this method, a cumulative distance matrix γ of the same dimension as the *distMatrix*, is created to store in the cell (i, j) the following value (Fig. 2.3):

$$\gamma(i,j) = d(T_i, S_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \qquad (2.25)$$

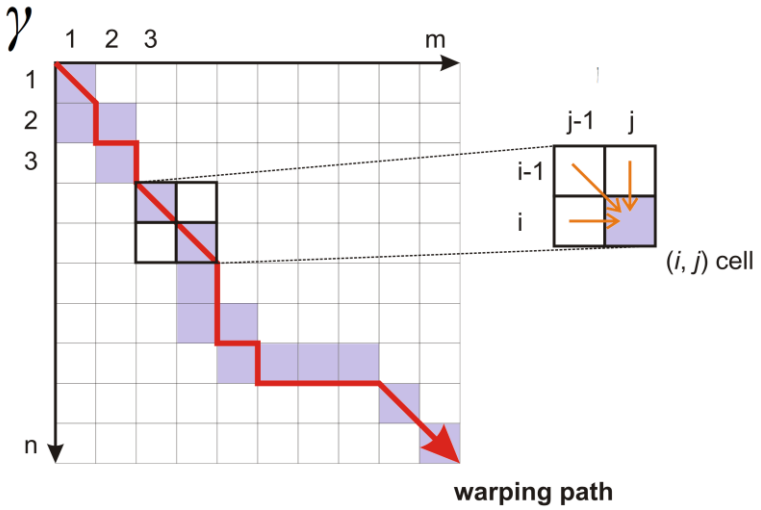


Fig. 2.3. Warping path computation using dynamic programming. The lavender cells correspond to the warping path. The red arrow indicates its direction. The warping distance at the (*i*, *j*) cell will consider, besides the distance between T_i and S_j, the minimum value among adjacent cells at positions: (*i*-1, *j*-1), (*i*-1, *j*) and (*i*, *j*-1). The Euclidean distance between two time series can be seen as a special case of *DTW*, where path's elements belong to the γ matrix diagonal.

The overall complexity of the method is relative to the computation of all distances in *distMatrix* that is O (nm). The last element of the warping path, w_K corresponds to the distance calculated with the *DTW* method.
In many cases, this method can bring to undesired effects. An example is when a large number of points of a time series T are mapped to a single point of another time series S (Fig. 2.4a, 2.5a). A common way to overcome this problem is to restrict the warping path in such a way it has to follow a direction along the diagonal (Fig. 2.4b, 2.5b). To do this, we can restrict the path enforcing the recursion to stop at a certain depth, represented by a threshold δ. Then, the cumulative distance matrix γ will be calculated as follows:

$$\gamma(i,j) = \begin{cases} d(T_i,S_j) + \min\{\gamma(i-1,j-1),\gamma(i-1,j),\gamma(i,j-1)\} & |i-j| < \delta \\ \infty & otherwise \end{cases} \quad (2.26)$$

Figure 2.5a shows the computation of a restricted warping path, using a threshold δ = 10. This constraint, besides limiting extreme or degenerate mappings, allows to speed-up *DTW* distance calculation, because we need to store only distances which are at most δ positions away (in horizontal

and vertical direction) from the *distMatrix* diagonal. This reduces the computational complexity to $O((n + m)\delta)$. The above proposed constraint is known also as *Sakoe-Chiba band* (Fig. 2.6a; Sakoe and Chiba, 1978) , and it is classified as global constraint. Another most common global constraint is the *Itakura parallelogram* (Fig. 2.6b; Itakura, 1975).

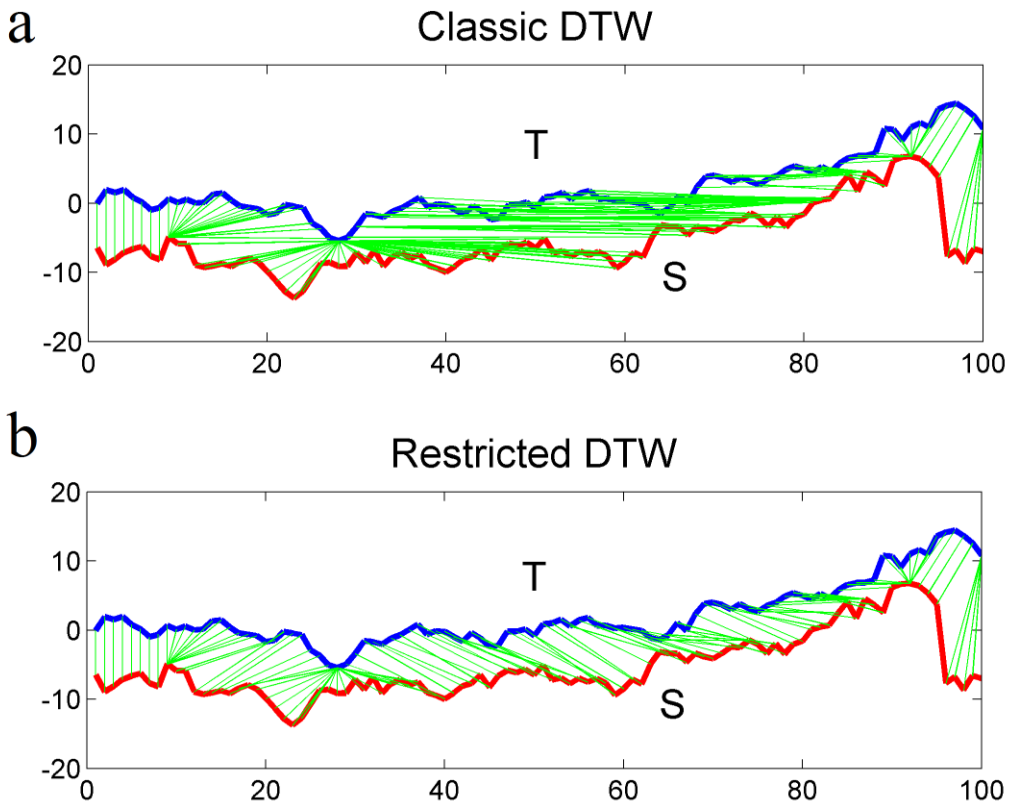


Fig. 2.4. Different mappings obtained with the classic implementation of *DTW* (a), and with the restricted path version using a threshold δ = 10 (b). Green lines represent mapping between points of time series *T* and *S*.

Local constraints are subject of research and are different from global constraints (Keogh and Ratanamahatana, 2002), because they provide local restrictions on the set of the alternative depth steps of the recurrence function (Eq. 2.25). For example we can replace Eq. 2.25 with:

$$\gamma(i,j) = d(T_i, S_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j-2), \gamma(i-2, j-1)\} \qquad (2.27)$$

to define a new local constraint.

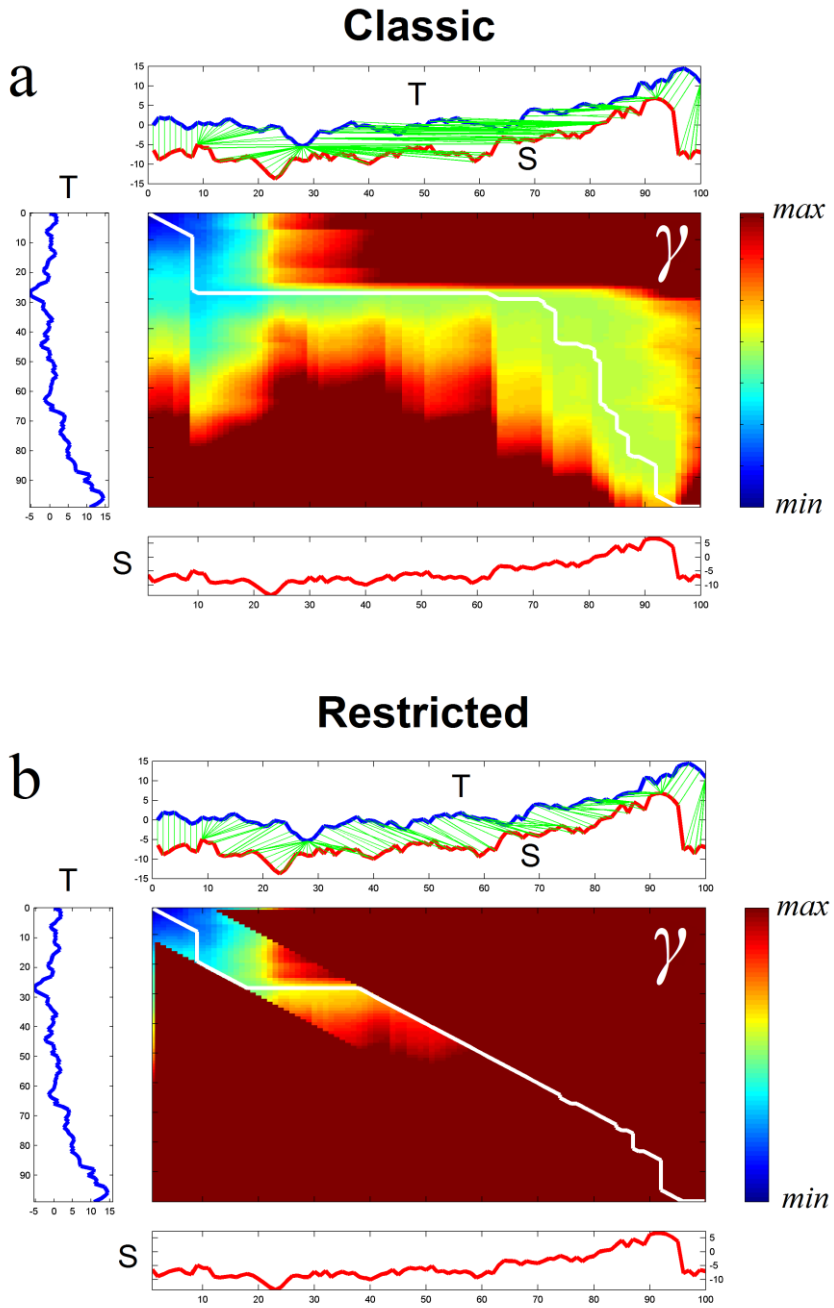


Fig. 2.5. (a) Classic implementation of DTW. (b) Restricted path, using a threshold δ = 10. For each plot (a) and (b): on the center, the warping path calculated on matrix γ. On the top, the alignment of time series T and S, represented by the green lines. On the left, the time series T. On the bottom, the time series S. On the right, the color bar relative to the distance values into matrix γ.

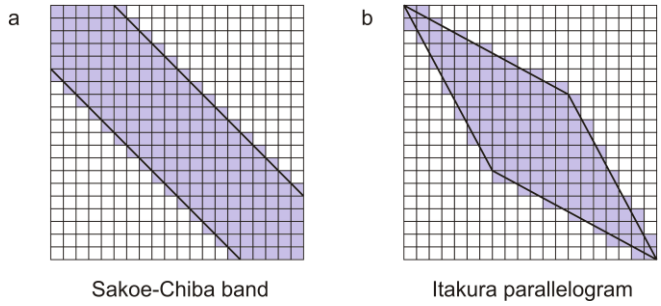


Fig. 2.6. Examples of global constraints: (a) Sakoe-Chiba band; (b) Itakura parallelogram.

Longest Common SubSequence. Another well known method that takes advantage of dynamic programming to allow comparison of one-to-many points is the *Longest Common SubSequence* (*LCSS*) similarity measure (Vlachos et al., 2002). An interesting feature of this method is that it is more resilient to noise than *DTW*, because allows some elements of time series to be unmatched (Fig. 2.7). This solution builds a matrix *LCSS* similar to γ, but considering similarity instead of distances.
Given the time series T and S of length n and m, respectively, the recurrence function is expressed as follows:

$$LCSS(i,j)=\begin{cases}0 & i=0,\\ 0 & j=0,\\ 1+LCSS[i-1,j-1] & if \quad T_i=S_j,\\ \max(LCSS[i-1,j],LCSS[i,j-1]) & otherwise\end{cases} \tag{2.28}$$

with $1 \le i \le n$ and $1 \le j \le m$. Since exact matching between T_i and S_j can be strict for numerical values (Eq. 3.22 is best indicated for string distance computation, such as the edit distance), a common way to relax this definition is to apply the following recurrence function:

$$LCSS(i,j)=\begin{cases}0 & i=0,\\ 0 & j=0,\\ 1+LCSS[i-1,j-1] & if \left|T_i-S_j\right|<\varepsilon,\\ \max(LCSS[i-1,j],LCSS[i,j-1]) & otherwise\end{cases} \tag{2.29}$$

The cell $LCSS(n, m)$ contains the similarity between T and S, because it corresponds to length l of the longest common subsequence of elements

between time series T and S. To define a distance measure, we can compute (Ratanamahatana et al., 2010):

$$LCSSdist(T,S) = \frac{n+m+2l}{m+n} \tag{2.30}$$

Also for LCSS the time complexity is $O(nm)$, but it can be improved to $O((n + m)\delta)$ if a restriction is used (i.e. when $|i - j| < \delta$).

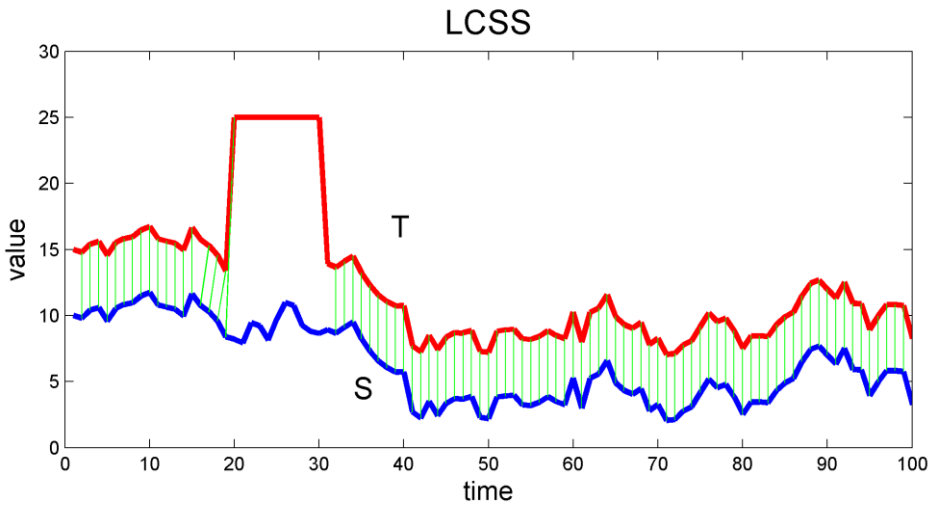


Fig. 2.7. Alignment using *LCSS*. Time series *T* (red line) is obtained from *S* (blue line), by adding a fixed value = 5, and further "noise" at positions starting from 20 to 30. In these positions there is no mapping (green lines).

Chapter 3

Dimensionality reduction techniques

Time series are often high dimensional data objects, thus dealing directly with the raw representation can be expensive in terms of space and time. In this sense, an important aspect to achieve efficiency, by means of space compression, is the use of dimensionality reduction techniques; while effective querying on time series data can be reached by using adequate similarity measures and space indexing. The goal of this chapter is to provide an overview of main dimensionality reduction algorithms (Cassisi et al., 2012c). Mining high-dimensional involves addressing a range of challenges, among them: i) the curse of dimensionality (Agrawal et al., 1993), and ii) the meaningfulness of the similarity measure in the high-dimensional space. A key aspect to achieve efficiency, when mining time series data, is to work with a data representation that is lighter than the raw data. This can be done by reducing the dimensionality of data, still maintaining its main properties. An important feature to be considered, when choosing a representation, is the *lower bounding* property.
Given two raw representations of the time series T and S, by this property, after establishing a true distance measure d_{true} for the raw data (such as the Euclidean distance), the distance $d_{feature}$ between two time series, in the reduced space, $R(T)$ and $R(S)$, have to be always less or equal than d_{true}:

$$d_{feature}(R(T), R(S)) \leq d_{true}(T, S) \tag{3.1}$$

If dimensionality reduction techniques ensure that the reduced representation of a time series satisfies such a property, we can assume that the similarity matching in the reduced space maintains its meaning. Moreover, we can take advantage of indexing structure such as *GEMINI* (Section 2.2) to perform speed-up search even avoiding false negative results. *GEMINI* was introduced to accommodate any dimensionality reduction method for time series, and then allows indexing on new

representation (Ng and Cai, 2004). *GEMINI* guarantees no false negatives on index search if two conditions are satisfied: (i) for the raw time series, a metric distance measure must be established; (ii) to work with the reduced representation, a specific requirement is that it guarantees the *lower bounding* property. In the following subsections, we will review the main dimensionality reduction techniques that preserve the *lower bounding* property. By this property, after establishing a true distance measure for the raw data (in this case the Euclidean distance), the distance between two time series, in the reduced space, results always less or equal than the true distance. Such a property ensures exact indexing of data (i.e. with no false negatives). The following representations describe the state-of-art in this field: spectral decomposition through *Discrete Fourier Transform* (*DFT*) (Agrawal et al., 1993); *Singular Value Decomposition* (*SVD*) (Korn et al., 1997); *Discrete Wavelet Transform* (*DWT*) (Chan and Fu, 1999); *Piecewise Aggregate Approximation* (*PAA*) (Keogh et al., 2000); *Piecewise Linear Approximation* (*PLA*) (Keogh et al., 2001); *Adaptive Piecewise Constant Approximation* (*APCA*) (Chakrabarti et al., 2002); and *Chebyshev Polynomials* (*CHEB*) (Ng and Cai, 2004). Many researchers have also included symbolic representations of time series, that transform time series measurements into a collection of discretized symbols; among them we cite the *Symbolic Aggregate approXimation* (*SAX*) (Lin et al., 2007), based on *PAA*, and the evolved multi-resolution representation *iSAX 2.0* (Shieh and Keogh, 2008). Symbolic representation can take advantage of efforts conducted by the text-processing and bioinformatics communities, who made available several data structures and algorithms for efficient pattern discovery on symbolic encodings (Lawrence et al., 1993; Bailey and Elkan, 1995; Tompa and Buhler, 2001).

3.1 DFT

The dimensionality reduction, based on the *Discrete Fourier Transform* (*DFT*) (Agrawal et al., 1993), was the first to be proposed for time series. The DFT decomposes a signal into a sum of sine and cosine waves, called *Fourier Series*. Each wave is represented by a complex number known as *Fourier coefficient* (Fig. 3.1) (Ng and Cai, 2004; Ratanamahatana et al., 2010). The most important feature of this method is the data compression,

because the original signal can be reconstructed by means of information carried by the waves with higher Fourier coefficient. The rest can be discarded with no significant loss.

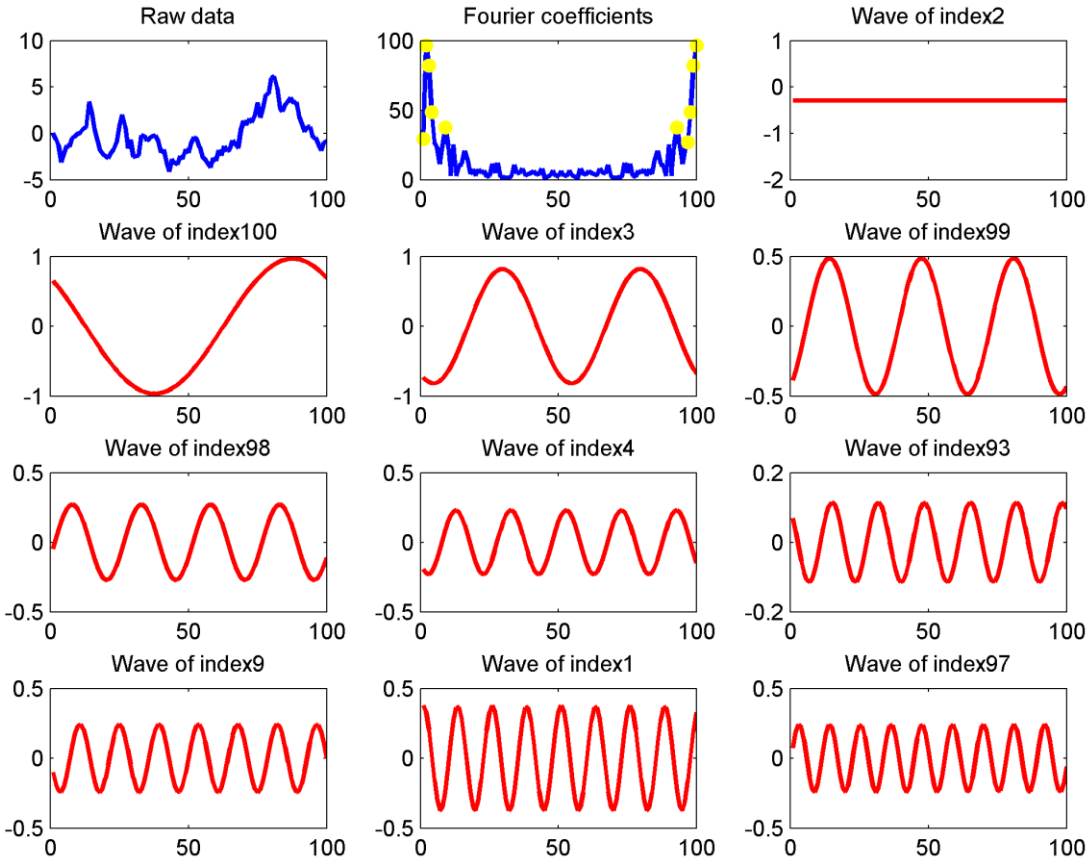


Fig. 3.1. The raw data is in the top-left plot. In the first row, the central plot ("Fourier coefficients" plot) shows the magnitude for each wave (Fourier coefficient). Yellow points are drawn for the top ten highest values. The remaining plots (in order from first row to last, and from left to right) represent the waves corresponding to the top ten highest coefficients in decreasing order, respectively of index {2, 100, 3, 99, 98, 4, 93, 9, 1, 97}, in the "Fourier coefficients" plot.

More formally, given a signal $x = \{x_1, x_2, \ldots, x_n\}$, the n-point *Discrete Fourier Transform* of x is a sequence $X = \{X_1, X_2, \ldots, X_n\}$ of complex numbers. X is the representation of x in the frequency domain. Each wave/frequency X_F is calculated as:

$$X_F = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_i e^{-\frac{2\pi i j F}{n}} \quad (j = \sqrt{-1}) \quad F = 1, \ldots, n \tag{3.2}$$

The original representation of x, in the time domain, can be recovered by the inverse function:

$$x_i = \frac{1}{\sqrt{n}} \sum_{F=1}^{n} X_F e^{-\frac{2\pi ijF}{n}} \quad (j = \sqrt{-1}) \quad i = 1, \ldots, n \tag{3.3}$$

The energy $E(x)$ of a signal x is given by:

$$E(x) = \|x\|^2 = \sum_{i=1}^{n} |x_i|^2 \tag{3.4}$$

A fundamental property of *DFT* is guaranteed by the *Parseval's Theorem*, which asserts that the energy calculated on time series domain for signal x is preserved on frequency domain, and then:

$$E(x) = \sum_{i=1}^{n} |x_i|^2 = \sum_{F=1}^{n} |X_F|^2 = E(X) \tag{3.5}$$

If we use the Euclidean distance (Eq. 2.13), by this property, the distance $d(x,y)$ between two signals x and y on time domain is the same as calculated in the frequency domain $d(X,Y)$, where X and Y are the respective transforms of x and y. The reduced representation $X' = \{X_1, X_2, \ldots, X_k\}$ is built by only keeping first k coefficients of X to reconstruct the signal x (Fig. 3.2).

For the Parseval's Theorem we can be sure that the distance calculated on the reduced space is always less than the distance calculated on the original space, because $k \leq n$ and then the distance measured using Eq. 2.13 will produce:

$$d(X', Y') \leq d(X, Y) = d(x, y) \tag{3.6}$$

that satisfies the lower bounding property defined in Eq. 3.1.

The computational complexity of *DFT* is $O(n^2)$, but it can be reduced by means of the *FFT* algorithm (Cooley and Tukey, 1965), which computes the *DFT* in $O(n \log n)$ time. The main drawback of *DFT* reduction

technique is the choice of the best number of coefficients to keep for a faithfully reconstruction of the original signal.

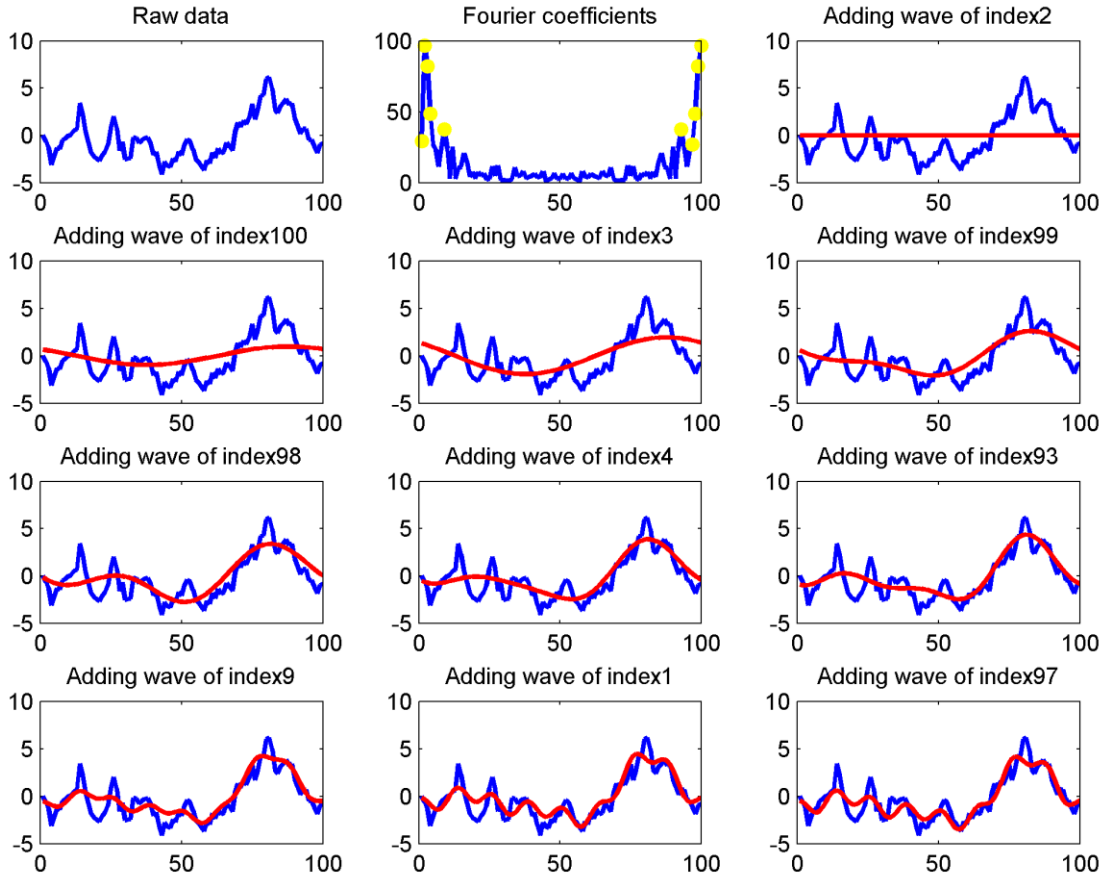


Fig. 3.2. The raw data is in the top-left plot. In the first row, the central plot ("Fourier coefficients" plot) shows the magnitude (Fourier coefficient) for each wave. Yellow points are drawn for the top ten highest values. The remaining plots (in order from first row to last, and from left to right) represent the reconstruction of the raw data using the wave with highest values (of index 2) firstly, then by adding the wave relative to second highest coefficient (of index 100), and so on.

3.2 DWT

Another technique for decomposing signals is the *Wavelet Transform* (*WT*). The basic idea of *WT* is data representation in terms of sum and difference of prototype functions, called *wavelets*. The discrete version of *WT* is the *Discrete Wavelet Transform* (*DWT*). Similarly to *DFT*, wavelet coefficients give local contributions to the reconstruction of the signal, while Fourier

coefficients always represent global contributions to the signal over all the time (Ratanamahatana et al., 2010).

The *Haar* wavelet is the simplest possible wavelet. Its formal definition is given by Chan and Fu (1999). An example of *DWT* based on Haar wavelet is shown in Table 3.1. The general *Haar* transform $H_L(T)$ of a time series T of length n can be formalized as follows:

$$A_{L'+1}(i) = \frac{A_{L'}(2i) + A_{L'}(2i+1)}{2} \tag{3.7}$$

$$D_{L'+1}(i) = \frac{D_{L'}(2i) - D_{L'}(2i+1)}{2} \tag{3.8}$$

$$H_L(T) = (A_L, D_L, D_{L-1}, \ldots, D_o) \tag{3.9}$$

where $0 < L' \leq L$, and $1 \leq i \leq n$.

Level (L)	*Averages coefficients (A)*	*Wavelet coefficients (D)*
1	10, 4, 6, 6	
2	8, 6	3, 0
3	7	1

Table 3.1. The *Haar* transform of T = {10, 4, 8, 6} depends on the chosen level, and corresponds to merging *Averages coefficients* (column 2) at the chosen level and all *Wavelet coefficients* (column 3) in decreasing order from the chosen level. At level 1 the representation is the same of time series: $H_1(T)$ = {10, 4, 6, 6} + {} = {10, 4, 6, 6} = T. At level 2 is $H_2(T)$ = {8, 6} + {3, 0} + {} = {8, 6, 3, 0}. At level 3 is $H_3(T)$ = {7} + {1} + {3, 0} = {7, 1, 3 0}.

The main drawback of this method is that it is well defined for time series which length n is a power of 2 ($n = 2^m$). The computational complexity of *DWT* using *Haar* Wavelet is $O(n)$.

Chan and Fu (1999) demonstrated that the Euclidean distance between two time series T and S, $d(T,S)$, can be calculated in terms of their *Haar* transform $d(H(T), H(S))$, by preserving the lower bounding property in Eq. 3.1, because:

$$d(H(T), H(S)) = \sqrt{2}d(T,S) < d(T,S) \tag{3.10}$$

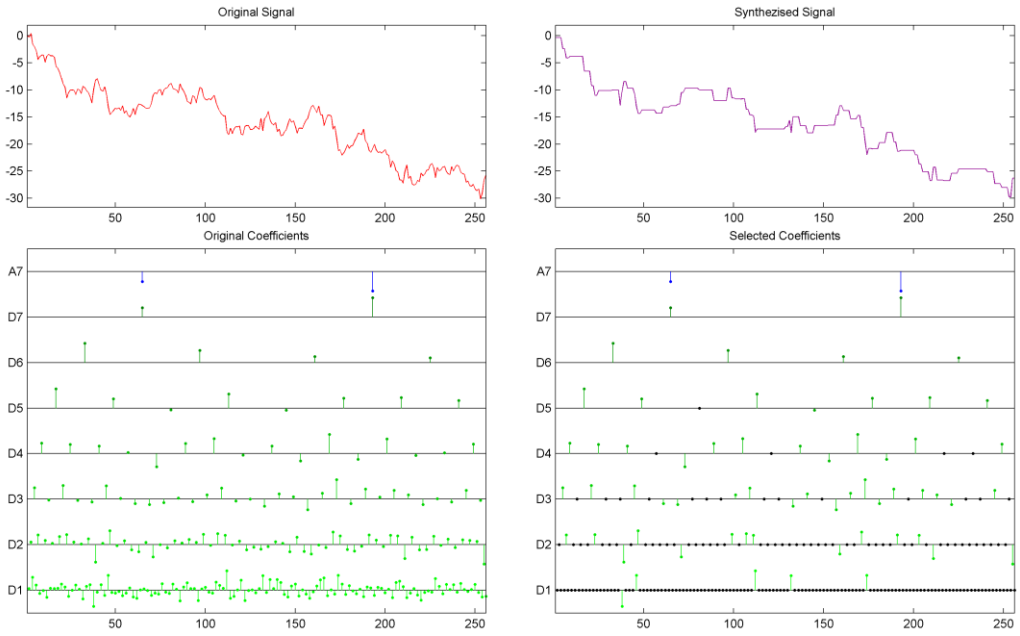


Fig. 3.3. *DWT* using *Haar* Wavelet with *MATLAB Wavelet Toolbox™ GUI tools*. *T* is a time series of length *n* = 256 and it is shown on the top-left plot (*Original Signal*). On the bottom-left plot (*Original Coefficients*) there are the entire A_L, represented by blue stems, and $D_{L'}$ coefficients ($L' < L = 7$), represented by green stems (stems' length is proportional to coefficients value). On the top-right plot, the *Synthesized Signal* by selecting only the 64 biggest coefficients, as reported on the bottom-right plot (*Selected Coefficients*): black points represent unselected coefficients.

3.3 SVD

As we have just seen in Chapter 2, a dataset with m time series (*TSDB*), each of length n, can be represented by an $m \times n$ matrix D (Eq. 2.1). An important result from linear algebra is that D can always be written in the form (Golub and Van Loan, 1996):

$$D = UWV^T \tag{3.11}$$

where U is an $m \times n$ matrix, W and V are $n \times n$ matrices. This is called the *Singular Value Decomposition* (*SVD*) of the matrix D, and the elements of the $n \times n$ diagonal matrix W are the *singular values* w_i:

$$W = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{pmatrix} \tag{3.12}$$

V is orthonormal, because $VV^T = V^TV = I_n$, where I_n is the identity matrix of size n. So, we can multiply both sides of Eq. 3.11 by V to get:

$$AV = UWV^TV \quad \rightarrow \quad AV = UW \tag{3.13}$$

UW represents a set of n-dimensional vectors $AV = \{X_1, X_2, \ldots, X_m\}$ which are rotated from the original vectors $A = \{x_1, x_2, \ldots, x_m\}$ (Ng and Cai, 2004):

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_m \end{pmatrix} \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{pmatrix} \tag{3.14}$$

Similarly to sine/cosine waves for *DFT* (Section 3.1) and to wavelet for *DWT* (Section 3.2), U vectors represent basis for AV, and their linear combination with W (that represents their coefficients) can reconstruct AV.
We can perform dimensionality reduction by selecting the first ordered k biggest singular values and their relative entries in A, V and U, to obtain a new k-dimensional dataset that best fits original data (Fig. 3.4).
SVD is an optimal transform if we aim to reconstruct data, because it minimizes the reconstruction error, but have two important drawbacks: (i) it needs a collection of time series to perform dimensionality reduction (it cannot operate on singular time series), because examines the whole dataset prior to transform. Moreover, the computational complexity is $O(\min(m^2n, mn^2))$. (ii) This transformation is not incremental, because a new data insertion requires a new global computation.

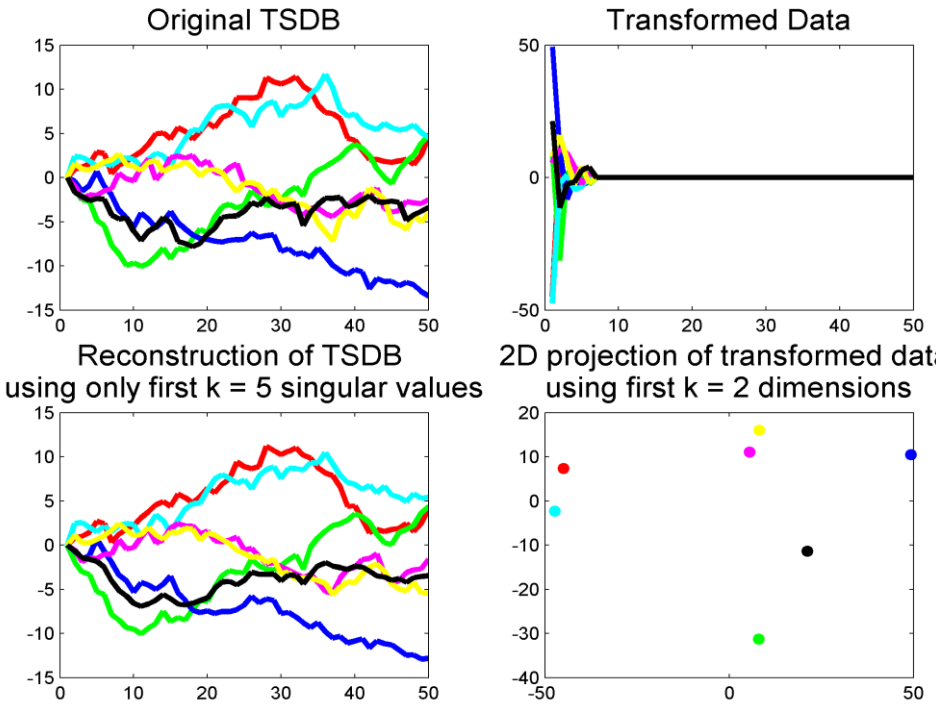


Fig. 3.4. ***SVD*** **for a** ***TSDB*** **of** m**=7 time series of length** n**=50. It is possible to note in the** ***transformed data*** **plot how only first** $k < 10$ **singular values are significant. In this example we heuristically choose to store only first** k**=5 diagonal elements of** V**, and their relative entries in** A**,** U **and** W**, because they represent about 95% of total variance. This allows reducing space complexity from** n **to** k**, still maintaining almost unchanged the information (see the reconstruction on the bottom-left plot).**

3.4 Dimensionality reduction via PAA

Given a time series T of length n, *PAA* divides it into w equal sized segments t_i ($1 < i \leq w$) and records values corresponding to the mean of each segment $mean(t_i)$ (Fig. 3.5) into a vector $PAA(T) = \{mean(t_1), mean(t_2), \ldots, mean(t_w)\}$, using the following formula:

$$mean(t_i) = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} t_j \tag{3.15}$$

When n is a power of 2, each $mean(t_i)$ essentially represents an *Averages coefficient* $A_L(i)$, defined in Section 3.2, and w corresponds in this case to:

$$w = \frac{n}{2^L} \tag{3.16}$$

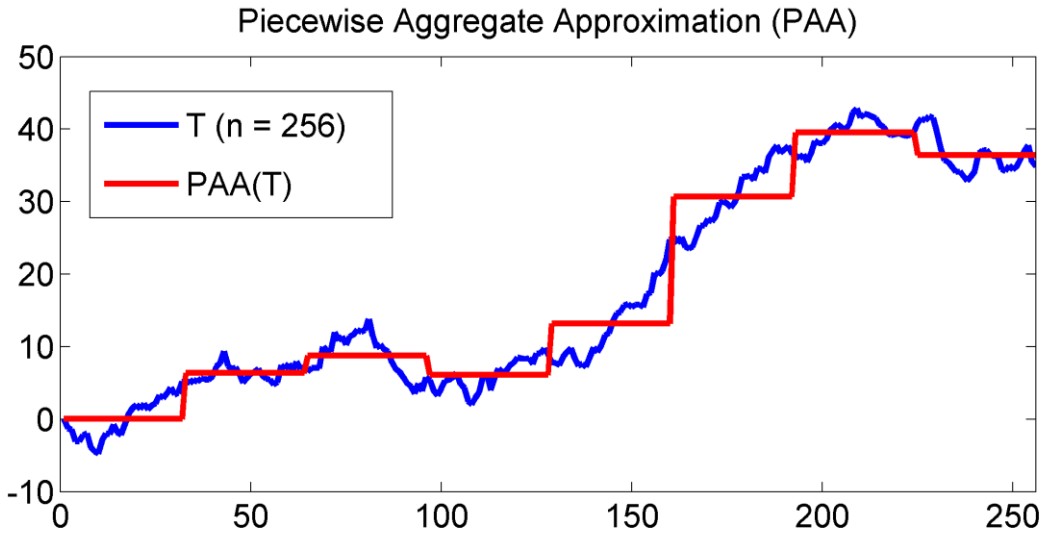


Fig. 3.5. An approximation via *PAA* technique of a time series *T* of length *n* = 256, with *w* = 8 segments.

The complexity time to calculate the mean values of Eq. 3.15 is $O(n)$. The *PAA* method is very simple and intuitive, moreover it is strongly competitive with other more sophisticated transforms such as *DFT* and *DWT* (Keogh and Kasetty, 2002; Yi and Faloutsos, 2000). Most of data mining researches make use of *PAA* reduction for its simplicity. It is simple to demonstrate how the distance on raw representation is bounded below by the distances calculated on *PAA* representation (even using Euclidean distance as reference point), satisfying Eq. 3.1. A limitation of such a reduction, in some contexts, can be the fixed size of the obtained segments.

3.5 APCA

In Section 3.2 we noticed that not all *Haar* coefficients in *DWT* are important for the time series reconstruction. Same thing for *PAA* in Section 3.4, where not all segment means are equally important for the reconstruction, or better, we sometimes need an approximation with no-fixed size segments. *APCA* is an adaptive model that, differently from *PAA*, allows defining segments of variable size. This can be useful when we find in time series areas of low variance and areas of high variance, for

which we want to have, respectively, few segments for the former, and many segments for the latter.

Given a time series $T = \{t_1, t_2, \ldots, t_n\}$ of length n, the *APCA* representation of T is defined as (Chakrabarti et al., 2002):

$$C = \left\{\langle cv_1, cr_1\rangle, \ldots, \langle cv_M, cr_M\rangle\right\}, \quad cr_0 = 0 \tag{3.17}$$

where cr_i is the last index of the ith segment, and

$$cv_i = mean\left(t_{cr_{i-1}+1}, \ldots, t_{cr_i}\right) \tag{3.18}$$

To find an optimal representation through the *APCA* technique, dynamic programming can be used (Pavlidis, 1976; Faloutsos et al., 1997). This solution requires $O(Mn^2)$ time. A better solution was proposed by Chakrabarti et al. (2002), which finds the *APCA* representation in $O(n \log n)$ time, and defines a distance measure for this representation satisfying the lower bounding property defined in Eq. 3.1. The proposed method bases on *Haar* wavelet transformation. As we have just seen in Section 3.2, the original signal can be reconstructed by only selecting bigger coefficients, and truncating the rest. The segments in the reconstructed signal may have approximate mean values (due to truncation) (Chakrabarti et al., 2002), so these values are replaced by the exact mean values of the original signal. Two aspects to consider before performing APCA:

1. Since *Haar* transformation deals only with time series length $n = 2^p$, we need to add zeros to the end of the time series, until it reaches the desired length.
2. If we held the biggest M Haar coefficients, we are not sure if the reconstruction will return an *APCA* representation of length M. We can know only that the number of segments will vary between M and $3M$ (Chakrabarti et al., 2002). If the number of segments is more than M, we will iteratively merge more similar adjacent pairs of segments, until we reach M segments.

```
Algorithm Compute_APCA(T,M)
begin
Padded = false;
if length(T) mod 2 <> 0 then
     Padding of T with zeros until length(T) mod 2 ==
     0;
     Padded = true;
Perform the Haar DWT on T;
Sort coefficients in order of decreasing normalized
magnitude, truncate after M;
C = Reconstruct approximation of T from retained
coeffs;
if Padded then
   Truncate C to the original length;
Replace approximate segment mean values with exact mean
values;
while the number of segments of C is greater than M do
   Merge the pair of segments with least rise in error;
end while;
end.
```

Table 3.2. An algorithm for the APCA, from Chakrabarti et al. (2002).

3.6 Time series segmentation using PLA

As with most computer science problems, representation of data is the key to efficient and effective solutions. A suitable representation of a time series may be *Piecewise Linear Approximation* (*PLA*), where the original points are reduced to a set of segments.

PLA refers to the approximation of a time series *T*, of length *N*, using *K* consecutive segments with *K* much smaller than *n* (Fig. 3.6). This representation makes the storage, transmission and computation of the data more efficient (Keogh et al., 2004). In the light of it, *PLA* may be used to support clustering, classification, indexing and association rule mining of time series data (e.g. Di Salvo et al., 2012).

The process of time series approximation using *PLA* is known as segmentation and is related to clustering process where each segment can be considered as a cluster (Salvador and Chan, 2004).

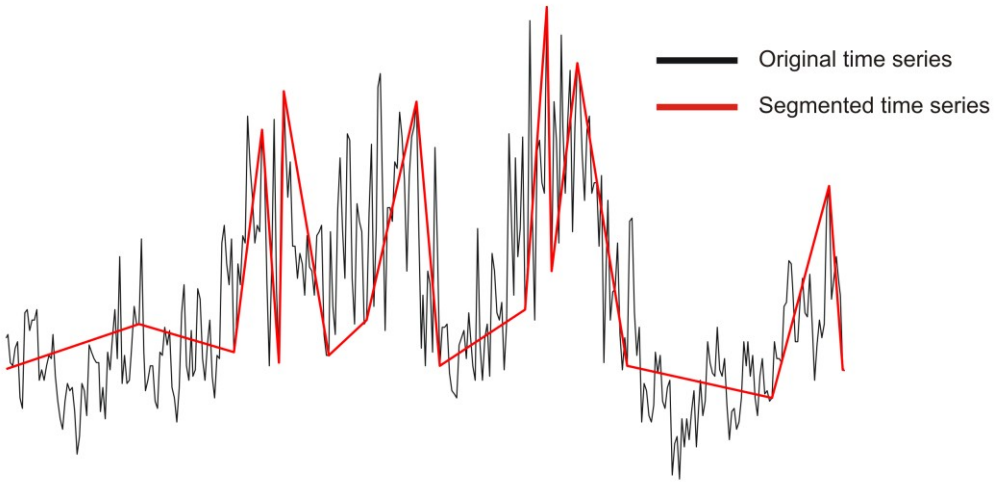


Fig. 3.6: The trend approximation (red line) of the original time series (black line) obtained by *PLA*.

There are several techniques to segment a time series and they can be distinguished into off-line and on-line approaches. In the former approach an error threshold is fixed by the user, while the latter uses a dynamic error threshold that changes, according to specific criteria, during the execution of the algorithm. Although off-line algorithms are simple to realize, they are less effective than the online ones. The classic approaches to time series segmentation are the sliding window (Table 3.3), bottom-up (Table 3.4) and top-down (Table 3.5) algorithms.

Sliding window is an on-line algorithm and works growing a segment until the error for the potential segment is greater than the user-specified threshold, then the subsequence is transformed into a segment; the process repeats until the entire time series has been approximated by its *PLA* (Keogh et al., 2004). A way to estimate error is by taking the mean of the sum of the square of vertical differences between the best-fit line and the actual data points. Another commonly used measure of goodness of fit is the distance between the best fit line and the data point furthest away in the vertical direction (Keogh et al., 2004).

```
Algorithm Seg_TS = Sliding_Window(T,max_error)
begin
anchor = 1;
while not finished segmenting time series
     i = 2;
     while (calculate_error(T[anchor:(anchor + i)]) <
     max_error)
          i = i + 1;
     end while;
     Seg_TS = concat(Seg_TS, create_segment(T[anchor:
     anchor + (i-1)]);
     anchor = anchor + i;
end while;
end.
```

Table 3.3. An algorithm for sliding window approach, from (Keogh et al., 2001).

```
Algorithm Seg_TS = Top_Down(T,max_error)
begin
best_so_far = inf;
for i = 2 to length(T) - 2
     // Find best place to split the time series.
     improvement_in_approximation =
     improvement_splitting_here(T,i);
     if improvement_in_approximation < best_so_far
          breakpoint = i;
          best_so_far = improvement_in_approximation;
     end if;
end for;
// Recursively split the left segment if necessary.
if calculate_error(T[1:breakpoint]) > max_error
     Seg_TS = Top_Down(T[1: breakpoint]);
end if;
// Recursively split the right segment if necessary.
if (calculate_error(T[(breakpoint+1):length(T)]) >
max_error)
     Seg_TS = Top_Down(T[breakpoint + 1: length(T)]);
end if.
end.
```

Table 3.4. An algorithm for top-down approach, from (Keogh et al., 2001).

```
Algorithm Seg_TS = Bottom_Up(T,max_error)
begin
for i = 1:2:length(T)
     // Create initial fine approximation.
     Seg_TS = concat(Seg_TS,create_segment(T[i:i+1]));
end for;

for i = 1:length(Seg_TS)-1
     // Find cost of merging each pair of segments.
     merge_cost(i) =
     calculate_error([merge(Seg_TS(i), Seg_TS(i+1))]);
end for;

while min(merge_cost) < max_error
     // Find "cheapest" pair to merge.
     index = min(merge_cost);

     // Merge them.
     Seg_TS(index) =
     merge(Seg_TS(index),Seg_TS(index+1)));

     // Update records.
     delete(Seg_TS(index+1));

     merge_cost(index) =
     calculate_error(merge(Seg_TS(index),Seg_TS(index+1
     )));

     merge_cost(index-1) =
     calculate_error(merge(Seg_TS(index-
     1),Seg_TS(index)));
end while;
end.
```

Table 3.5. An algorithm for bottom-up approach, from (Keogh et al., 2001).

In the top-down approach a segment, that represents the entire time-series, is recursively split until the desired number of segment or an error threshold is reached. Dually, the bottom-up algorithm starts from the finest approximation of the time series using $n/2$ segments and merging

the two most similar adjacent segments until the desired number of segment or an error threshold is reached.

However, an open question is the choice of best *k* number of segments. This problem involves a trade-off between compression and accuracy of time series representation. As suggested by Salvador and Chan (2004), the appropriate number of segments may be estimated using evaluation graph. It is defined as a two dimensional plot where x-axis is the number of segments, while y-axis is a measure of the segmentation error. The best number of segments is provided by the point of maximum curvature, also called "knee", of the evaluation graph (Fig. 3.7).

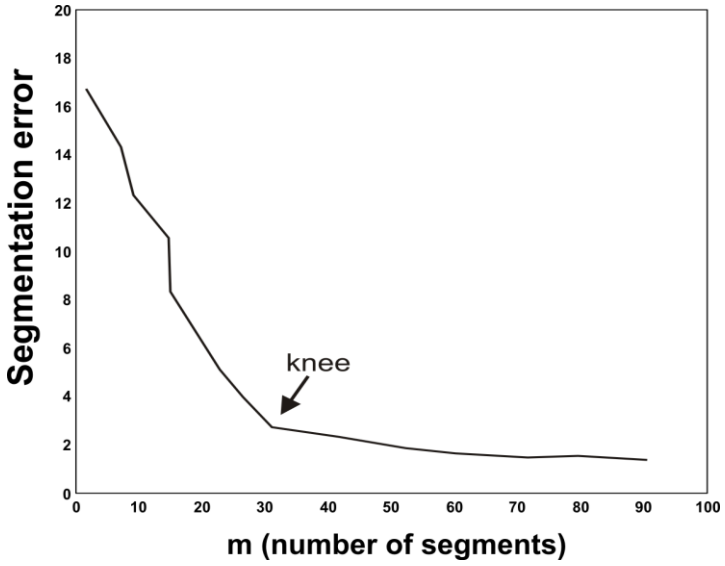


Fig. 3.7: Evaluation graph. The best number of segments is provided by the knee of the curvature.

3.7 Chebyshev Polynomials approximation

By this technique, the reduction problem is resolved by considering the values of the time series *T* as values of a function *f*, and approximating it with a polynomial function of degree *n* which well fits *f*:

$$P_n(x) = \sum_{i=0}^{n} a_i x^i \tag{3.19}$$

where each a_i corresponds to coefficients and x^i to the variables of degree *i*.

There are many possible ways to choose the polynomial: Fourier transforms (Section 3.1), splines, non-linear regressions, etc. Ng and Cai (2004) hypothesized that one of the best approaches is the polynomial that minimizes the maximum deviation from the true value, which is called the *minimax* polynomial. It has been shown that the *Chebyshev* approximation is almost identical to the optimal minimax polynomial, and is easy to compute (Mason and Handscomb, 2003). Thus, Ng and Cai (2004) explored how to use the *Chebyshev polynomials* (of the first kind) as a basis for approximating and indexing n-dimensional ($n \geq 1$) time series. The Chebyshev polynomial $CP_m(x)$ of the first kind is a polynomial of degree m ($m = 0, 1, \ldots$), defined as:

$$CP_m(x) = \cos(m \arccos(x)) \quad x \in [-1, 1] \tag{3.20}$$

It is possible to compute every $CP_m(x)$ using the following recurrence function (Ng and Cai, 2004):

$$CP_m(x) = 2xCP_{m-1}(x) - CP_{m-2}(x) \tag{3.21}$$

for all $m \geq 2$ with $CP_0(x) = 1$ and $CP_1(x) = x$. Since Chebyshev polynomials form a family of orthogonal functions, a function $f(x)$ can be approximated by the following Chebyshev series expansion:

$$S_\infty^{CP}(f(x)) = \sum_{i=0}^{\infty} c_i CP_i(x) \tag{3.22}$$

where c_i refer to the *Chebyshev coefficients*. We refer the reader to the paper (Ng and Cai, 2004) for the conversion of a time series, which represents a discrete function, to an interval function required for the computation of Chebyshev coefficients. Given two time series T and S, and their corresponding vectors of Chebyshev coefficients, C_1 and C_2, the key feature of their work is the definition of a distance function d_{Cheb} between the two vectors that guarantees the lower bounding property defined in Eq. 3.1. Since it results:

$$d_{cheb}(C_1, C_2) \leq d_{true}(T_1, T_2) \tag{3.23}$$

the indexing with Chebyshev coefficients admits no false negatives. The computational complexity of Chebishev approximation is $O(n)$, where n is the length of the approximated time series.

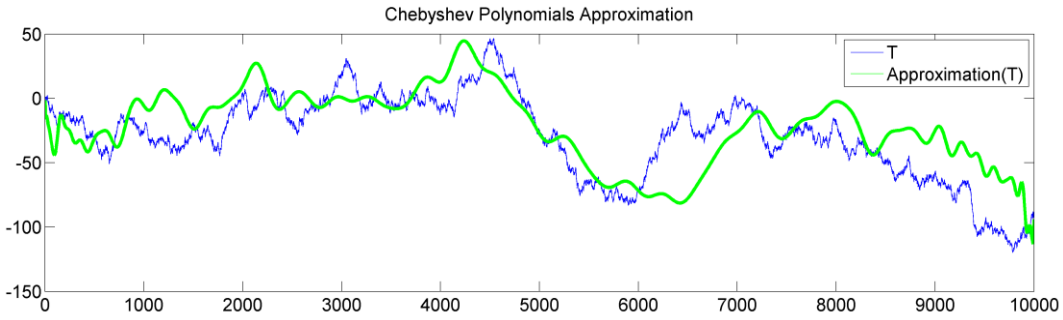


Fig. 3.8: An example of approximation of a time series T of length n = 10000 with a Chebyshev series expansion (Eq. 3.22) where i is from 0 to k = 100, using the *chebfun* toolbox for MATLAB (http://www2.maths.ox.ac.uk/chebfun/).

3.8 SAX

Many symbolic representations of time series have been introduced over the past decades. The challenge in this field is to create a real correlation between the distance measure defined on the symbolic representation, and that defined on original time series. *SAX* is the most known symbolic representation technique on time series data mining that ensures both a considerable dimensionality reduction, and the lower bounding property, allowing enhancing of time performances on most of data mining algorithm.

Given a time series T of length n, and an alphabet of arbitrary size a, *SAX* returns a string of arbitrary length w (typically $w << n$). The alphabet size a is an integer, where $a > 2$. *SAX* method is *PAA*-based (see Section 3.4), since it transforms *PAA* means into symbols, according to a defined transformation function.

To give a significance to the symbolic transformation, it is necessary to deal with a system producing symbols with equal probability, or with a Gaussian distribution. This can be achieved by normalizing time series, since normalized time series have generally a Gaussian distribution (Lin et al., 2007). This is the first assumption to consider about this technique.

However, for data not obeying to this property, the efficiency of the reduction is slightly deteriorated. Given the Gaussian distribution, it is simple to determine the "breakpoints" that will produce *a* equal-sized areas of probability under the Gaussian curve. What follows gives the principal definitions to understand *SAX* representation.

Definition 3.8.1. ***Breakpoints***: A sorted list of numbers $B = \beta_1, \ldots, \beta_{a-1}$ such that the area under a $N(0, 1)$ Gaussian curve from β_i to $\beta_{i+1} = 1/a$ (β_0 and β_a are defined as $-\infty$ and ∞, respectively) (Table 3.6). For example, if we want to obtain breakpoints for an alphabet of size $a = 4$, we have to compute the first (q_1), the second (q_2), and the third (q_3) quartiles of the inverse cumulative Gaussian distribution, corresponding to the 25%, 50% and 75% of the cumulative frequency: $\beta_1 = q_1$, $\beta_2 = q_2$, $\beta_3 = q_3$.

$\beta_i \setminus a$	3	4	5	6	7	8
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67
β_3		0.67	0.25	0	-0.18	-0.32
β_4			0.84	0.43	0.18	0
β_5				0.97	0.57	0.32
β_6					1.07	0.67
β_7						1.15

Table 3.6. A look-up table for breakpoints used for alphabet of size 2 < a < 9.

Definition 3.8.2. ***Alphabet***: A collection of symbols *alpha* = $\alpha_1, \alpha_2, \ldots, \alpha_a$ of size *a* used to transform mean frames into symbols.

Definition 3.8.3. ***Word***: A PAA approximation $PAA(T) = \{mean(t_1), mean(t_2), \ldots, mean(t_w)\}$ of length *w* can be represented as a word $SAX(T) = \{sax(t_1), sax(t_2), \ldots, sax(t_w)\}$, with respect to the following mapping function:

$$sax(t_i) = \alpha_j, \quad iff \ \beta_{j-1} \le mean(t_i) < \beta_j \quad (0 < i \le w, \quad 1 < j < a) \tag{3.24}$$

Lin et al. (2007) defined a distance measure for this representation, such that the real distance calculated on original representation is bounded

below from it. An extension of *SAX* technique, *iSAX*, was proposed by Shieh and Keogh (2008) which allows to get different resolutions for the same word, by using several combination of parameters a and w.

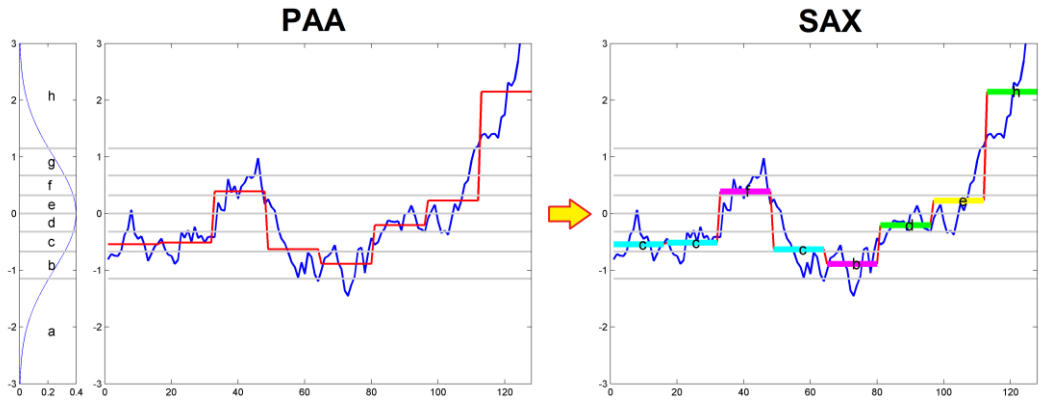


Fig. 3.9: An example of conversion of a time series T (blue line) of length n = 128, into a word of length w = 8, using an alphabet *alpha* = {*a*,*b*,*c*,*d*,*e*,*f*,*g*,*h*} of size a = 8. The left plot refers to the Gaussian distribution divided into equal areas of size 1/a. *PAA* mean frames falling into two consecutive cutlines (gray lines) will be mapped into the corresponding plotted symbol (colored segments). The *PAA* plot shows the *PAA* representation (red line), while *SAX* plot shows the conversion of *PAA*(*T*) into the word *SAX*(*T*) = {*c*,*g*,*e*,*g*,*f*,*g*,*a*,*a*}. Images generated by MATLAB and code provided by *SAX* authors (Lin et al., 2007).

Chapter 4

Classification and prediction

Classification process can be defined as the task in which objects are classified on the basis of a priori knowledge patterns. The main difference between clustering and classification process is the unsupervised nature of the clustering. While the clustering process attempts to derive meaningful classes directly from the data, the traditional classification methods involve a special input training set of classes in which known objects are placed. The choice of classification algorithm is strictly related to the data structure and is guided by the prediction performance obtained by the chosen model. In time series analysis, also the prediction of symbolic time series or sequences is sometimes called classification. Han and Kamber (2000) view that predicting class labels is *classification*, and predicting values (e.g. using regression techniques) is *prediction*.
In this section we will briefly introduce, in the first part, some basic methods for the discriminant analysis, the most known classification technique. The second part focuses on a prediction task, spending attention on Hidden Markov Models.

4.1 Classification

The discriminant analysis, known as *supervised classification*, is a classification technique proposed by Fisher (1936). It bases on identification of the attributes able to discriminate the observations of a sample and to classify them in different groups with respect to their attributes.

4.1.1 Fisher's discriminant analysis

Fisher's linear discriminant, used for *Linear Discriminant Analysis* (*LDA*) is a classification method that projects high-dimensional data onto a line and

performs classification in this one-dimensional space. The projection w is chosen to maximize the distance between the means of the two classes while minimizing the variance within each class. To that purpose *Fisher-LDA* considers maximizing the following objective:

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \tag{4.1}$$

where S_B is the *between classes scatter matrix* and S_W is the *within classes scatter matrix*. The definitions of the scatter matrices are:

$$S_B = \sum_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T \tag{4.2}$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{4.3}$$

where $\bar{x}$ is the overall mean of the dataset.

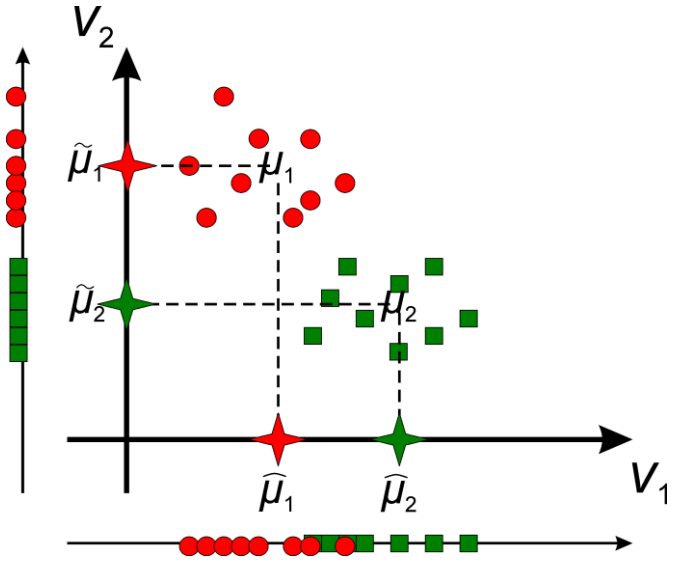


Fig. 4.1. Let be μ_1 e μ_2 the mean values of class 1 (represented by red objects) and 2 (represented by green objects), respectively. In this simple example, the couple of values ($\tilde{\mu}_1$, $\tilde{\mu}_2$) and ($\hat{\mu}_1$, $\hat{\mu}_2$) are the projected class means respectively on the first attribute (the horizontal axis) and on the second attribute (the vertical axis). The projection on second attribute returns a best separation among them because $|\tilde{\mu}_1 - \tilde{\mu}_2| > |\hat{\mu}_1 - \hat{\mu}_2|$.

Assume we have a set of samples partitioned into two classes (Fig. 4.1). Oftentimes you will see that for two classes S_B is defined as

$$S'_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tag{4.4}$$

This is the scatter of class 1 with respect to the scatter of class 2 and hence corresponds to computing the scatter relative to a different vector.

4.1.2 The perceptron criterion function

Another well known method for linear discriminant analysis is the perceptron criterion function. It is the first method using the neural network engineering (other best solutions are *SOM*, described in Section 5.3.6). Suppose to deal with a two dimensional dataset *D* with *n* objects, where each object is represented by a point with two attributes *x* and *y*, and we know the label *l* for each object: -1 or 1 (in Fig. 4.2 they are represented respectively by red and blue points).

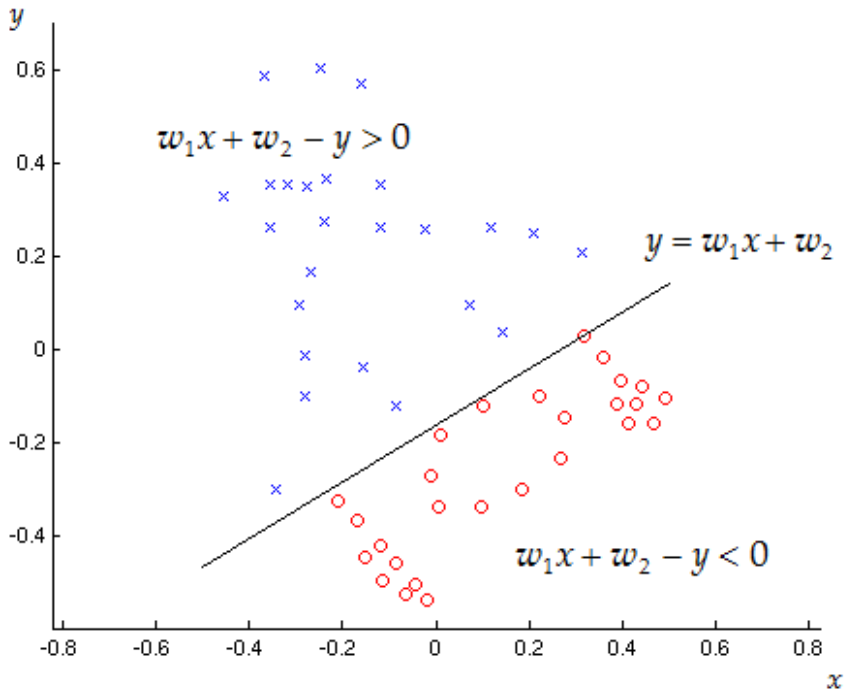


Fig 4.2. An example of 2D dataset with two classes represented by red and blue points. The perceptron function aim to discover the values of parameters w_1 and w_2 which best separates the two classes.

```
Perceptron learning phase(D,η)
begin
t = 0;
Randomly initialize w1,t and w2,t;
while convergence is reached do
     Pick (xt, yt, lt) from the training set D;
     if sign(w1,txt + w2,t − yt)·sign(lt) < 0 then
          δ-rule(w1,t,w2,t,xt,yt,lt,η)
     end if
     t = t + 1;
end while
end
```

Table 4.1 Learning phase of the perceptron. It calls the *δ-rule* to calibrate *w* values.

```
δ-rule(w1,t,w2,t,xt,yt,lt,η)
begin
if lt < 0 and (w1,txt + w2,t − yt) ≥ 0 then
     w1,t+1 = w1,t − ηxt;
     w2,t+1 = w1,t − η;
end if
if lt ≥ 0 and (w1,txt + w2,t − yt) < 0 then
     w1,t+1 = w1,t + ηxt;
     w2,t+1 = w1,t + η;
end if
end
```

Table 4.2 With the *δ-rule* if the line which linearly separates the two classes exists, then it will be discovered; else the perceptron learning phase (Table 6.1 will continue indefinitely.

A discriminant function wants to find the values of vector w (in this case containing w_1 and w_2) such that:

$$sign(w_1 x_i + w_2 - y_i) \cdot sign(l_i) > 0 \qquad (4.5)$$

for each object i of attribute (x_i, y_i), with $0 < i < n$. To do this a learning phase is needed. The procedure to calculate w is shown on Tables 4.1-2: it

needs a parameter η relative to the learning rate. At time t=0 the values of w are randomly initialized.
The perceptron method, like the Fisher's method, has several limits, among them the application on linearly separable cases (obviously), a high generalization error, and a high sensitivity to class boundary objects.

4.1.3 k-Nearest Neighbour (kNN)

The *k-Nearest Neighbour* algorithm is widely used for the classification of objects belonging to Euclidean spaces. It consists on to choice a class label for a new observation, basing on the known class of the k nearest neighbor samples (Fig. 4.3).

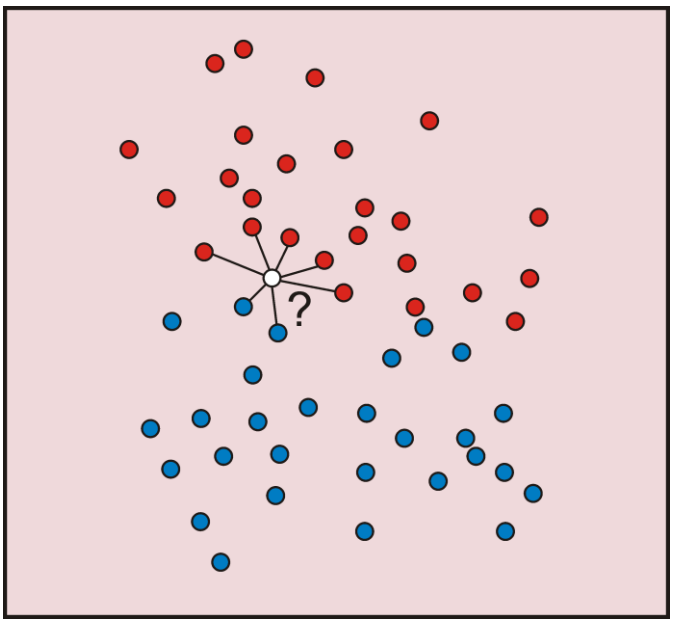


Fig. 4.3. A dataset with two classes (distinct with red and blue points). The new observation class (the white point) is chosen with respect to the majority class of its k (in this case k = 7) nearest neighbors.

4.1.4 Decision trees

Decision trees are predictive models and belong to the decisional theory field. In data mining, decision trees were introduced to solve problems connected to data classification.
Looking at the structure (Fig. 4.4), the tree nodes correspond to logical statements about the value of one or more attributes of the observation.

The first node is the *root* and contains the first (possibly more important) test for classification, while the *leaf* nodes contain the class labels.

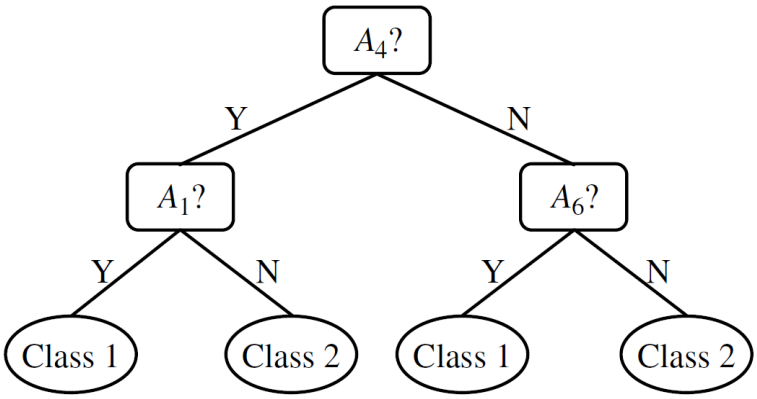


Fig. 4.4. Internal nodes represent tests on attributes. Leaf nodes represent classes (redrawn from Han and Kamber, 2000).

The construction of a decision tree consists of several stages:

1. Selection of the splitting attribute, or that most characterizing the observations. The algorithm identifies the most important attribute and splits the database into two partitions with respect to this. At each level the algorithm identifies the attribute that best performs the partition on the available observation. For this scope methods for feature selection are used, such as the *information gain*, described in Appendix C.
2. Determination of the *branch factor*: each *parent* node can generate one or more *child* nodes.
3. Evaluation of the best split: after the choice of the attribute, the algorithm identifies the split value which realizes the best division between observations.
4. Choice of the type of partition to realize: algorithms usually perform a recursive partition.
5. Choice of the tree dimensions, by establishing the minimum number of leaves or the tree depth, to avoid making estimation errors.
6. A pruning step, which try to optimize the decision process basing on certain criteria.

7. In a final step, the algorithms try to solve problems about missing values by ignoring them or, for example, creating a special class for them.

Among the most important classification algorithms implementing decision trees we mention:

- *Automatic Interaction Detection* (*AID*; Kass, 1980);
- *Chi-Square Automatic Interaction Detection* (*CHAID*; Hawkins and Kass, 1982);
- *Classification And Regression Trees* (*CART*; Breiman et al., 1984);
- *ID3* (Quinlan, 1986);
- *C4.5* (Quinlan, 1993).

This model achieves widespread success because has very small processing time, but presents some negative aspects, including the inability to examine continue attributes. Moreover, even if data can be perfectly divided into classes and uses only simple threshold tests, there is no foolproof way to predict the number of branches or spears that emit from decisions or sub-decisions.

4.1.5 Support Vector Machines (SVMs)

SVMs are a popular machine learning method for solving problems in classification and regression, able to guarantee high classification quality (Burges 1998). In recent years, novel applications of SVM have been performed in several research areas such as biology (e.g. Noble 2004; Cheng et al. 2006) and volcano seismology (e.g. Masotti et al. 2008; Langer et al. 2009). The SVM algorithm can be summarized as follows. It first uses a non-linear mapping to transform the original data set into a higher dimension space. Next, it identifies a hyperplane able to maximize the margin of separation among the classes of the training set. Such a hyperplane is called *maximum marginal hyperplane* (*MMH*). The margin in SVMs denotes the distance from the boundary to the closest data in the feature space (Fig. 4.5).

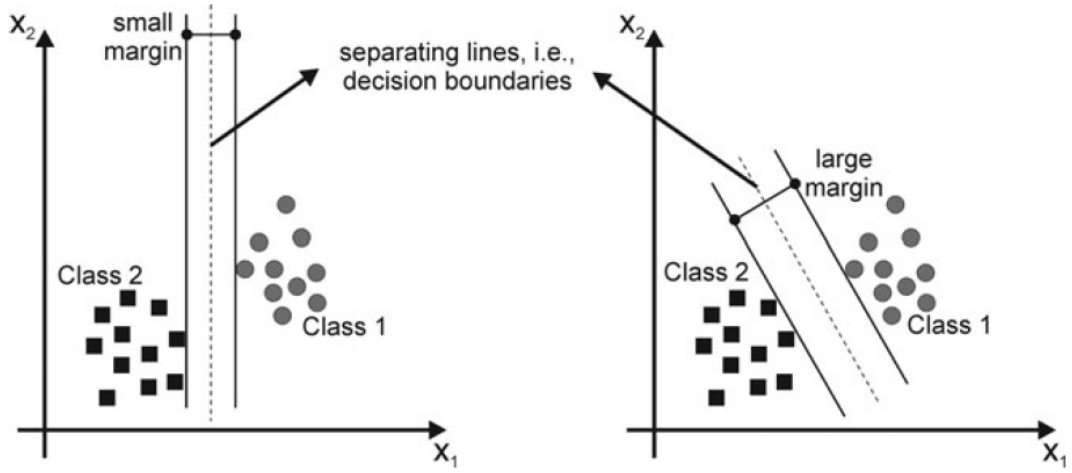


Fig. 4.5. Two-feature planes each of which with two classes of data (black squares and grey circles) and a separating line (dashed lines): the left one shows a small margin between clusters, the right one a larger margin (redrawn from Kecman 2001).

With appropriate mapping, data from two classes can always be separated by a hyperplane. The problem of computing the MMH can be formulated in terms of quadratic programming in the following way (Hwanjo et al. 2003):

$$W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \tag{4.6}$$

subject to:

$$\sum_{i=1}^{l} y_i \alpha_i = 0 \quad \forall i : 0 \le \alpha_i \le C. \tag{4.7}$$

The number of training data is denoted by l, α is a vector of l variables, where each component αi corresponds to a training data (x_i, y_i). C is the soft margin parameter controlling the influence of the outliers (or noise) in training data. The kernel for linear boundary function is $x_i y_i$, a scalar product of two data points. The non-linear transformation of the feature space is performed by replacing $k(x_i, y_i)$ with an advanced kernel φ, such as polynomial kernel:

$$(x^T x_i + 1)^p \tag{4.8}$$

or a radial basis function kernel:

$$\exp(-\frac{1}{2\sigma^2}\|x - x_i\|^2) \tag{4.9}$$

The use of an advanced kernel is an attractive computational shortcut, which avoids the expensive creation of a complicated feature space. An advanced kernel is a function that operates on the input data but has the effect of computing the scalar product of their images in a usually much higher-dimensional feature space (or even an infinite-dimensional space), which allows one to work implicitly with hyperplanes in such highly complex spaces (Fig. 4.6).

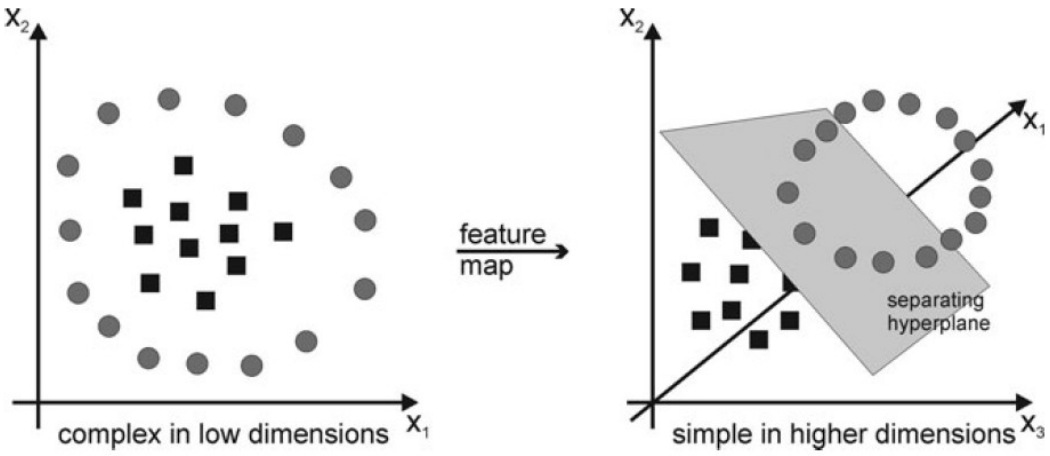


Fig. 4.6. Two classes of data in the original 2-D space (left) and in a higher-dimensional feature space (right) (from Cannata et al., 2011a).

The extension of SVM to multiclass problems can be performed using two different methods called one-against-one and one-against-all. The former constructs $k(k-1)/2$ classifier where each one is trained on data from two classes. The latter constructs k SVM classifier. In this last case, the ith SVM is trained using all training patterns belonging to ith class with positive labels and the other with negative labels. A point is assigned to the class for which the distance from margin is maximal. Finally, the output of one-

against-all method is the class that corresponds to SVM with highest output value (Weston and Watkins 1999; Hsu and Lin 2002).

4.2 Prediction

Prediction is usually understood as forecasting the next few values of a numeric or symbolic series. For the prediction of numeric time series there is a huge amount of literature especially in the field of statistics. Common methods for time series modeling and prediction are *ARIMA* (*Auto Regressive Integrated Moving Average*) (Box et al., 1994) models for stationary processes. More simple techniques like regression (Appendix D) or more complicated techniques like supervised neural nets (Tang and Fishwick, 1991; see also Section 5.3.6), or *Support Vector Machines* (Section 4.1.5; Burges, 1998) can also be used. In the following subsection we will describe *Markov Chains* and *Hidden Markov Models* (Rabiner, 1989) because they are commonly used.

4.2.1 Hidden Markov Models

A stochastic process is formally defined by a set of random variables $\{s(t)\}$, where parameter t corresponds to time. In general, the future behaviour of a system depends on its past, i.e. the states $s(t)$ and $s(t')$, corresponding to different times t and t', are dependent random variables. Markov process is an interesting class of stochastic processes, where a future state depends only on its most recent states (Bharucha-Reid, 1960). The first-order Markov Models are the most used for its simplicity, because a future state $s(t)$, is a function of only its nearest past state $s(t\text{-}1)$:

$$s(t) = f(s(t-1)) \tag{4.10}$$

If f is a deterministic function, then the Markov model is deterministic. In pattern recognition f is a probabilistic function, for which $s(t)$ is reached with a certain probability from the state $s(t\text{-}1)$. A typical diagram of a probabilistic first-order Markov model is in Fig. 4.7.

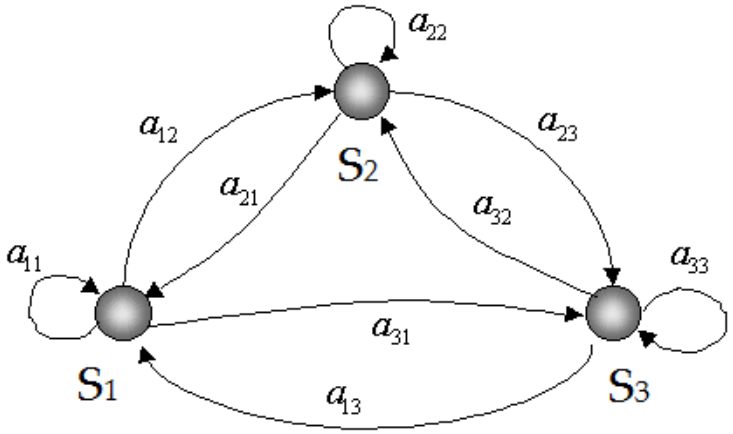


Fig 4.7. In this model there are three states (S_1, S_2, S_3). a_{ij} (0 < *i,j* < 3) indicates the probability, at a certain time *t*, of transition from the state S_i to a state S_j.

Assuming the system varies into a finite set of states, whose identity is not visible to the reader, the *Hidden Markov Model* (*HMM;* Rabiner, 1989) is a statistical model whose purpose is to recover the sequence of hidden states, by reading several observations (or *tokens*), which are the only outputs provided by the system (Fig 4.8).
For each state is associated a probability distribution over all possible output tokens. Transitions among the states are governed by a set of probabilities. Given a sequence of tokens, we learn an *HMM* and derive a sequence of hidden states that correspond to the sequence of tokens. Formally, an *HMM*, denoted by $\lambda = \{S, A, O, E\}$ is described with the following parameters:

1. A set of *m* states $S = \{S_1, S_2, \ldots, S_m\}$.
2. A *m* × *m* state transition matrix $A = \{a_{ij}\}$ whose *(i, j)* entry is the probability of a transition from state *Si* to state *Sj*.
3. A set of possible observations, or tokens, $O = \{O_1, O_2, \ldots, O_n\}$.
4. An *m* × *n* emission matrix $E = \{e_{ij}\}$ whose *(i,j)* entry gives the probability of emitting symbol O_j given that the model is in state *Si*.
5. Start probabilities π_i, where $\pi_1 + \pi_2 + \ldots + \pi_m = 1$, which denote the probability of starting at a given state in the first time point.

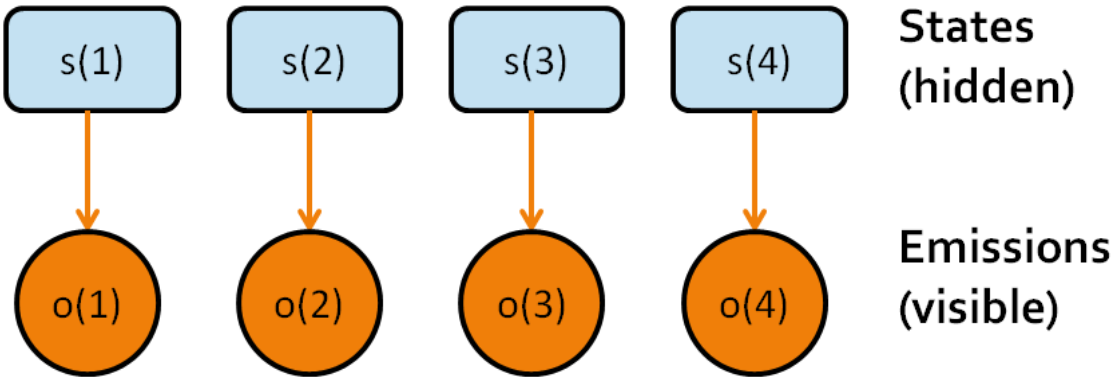


Fig. 4.8. Hidden Markov Models analysis seeks to recover the sequence of hidden states from the observed emissions.

Three main problems are associated to *HMM*s:

1) **Evaluation**: Given a model λ, and a sequence of observations o, find $P(o \mid \lambda)$.
2) **Decoding:** Given a model λ, and a sequence of observations o, find the sequence s of states that maximize $P(s \mid o, \lambda)$.
3) **Learning**: Given a model λ, with unspecified transition/emission probabilities, find parameters $\{S, A, O, E\}$ that maximize $P(o \mid \lambda)$.

Evaluation

There are two ways to compute the probability $P(o \mid \lambda)$ of an observed sequence o, given a model λ: *forward* or *backward* algorithm.

Forward algorithm. Let the forward variable $\alpha_t(i)$ be:

$$\alpha_T(i) = P(o(1)o(2)\ldots o(T), s(T) = S_i \mid \lambda) \tag{4.11}$$

the probability to observe a sequence $o = o(1),o(2),\ldots,o(T)$ and state in S_i at time T, given a model λ. This probability can be calculated recurrently:

1. Inizialization:

$$\alpha_1(i) = \pi_i e_{iI(o(1))}, \quad 1 \le i \le m \tag{4.12}$$

where $I(o(1))$ is the index of the token in O equal to $o(1)$.

2. Induction (Fig. 4.9):

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{m} \alpha_t(i)\, a_{ij} \right] e_{jI(o(t+1))}, \quad 1 \le j \le m, \quad 1 < t \le T-1 \tag{4.13}$$

3. Termination:

$$P(o \mid \lambda) = \sum_{i=1}^{m} \alpha_T(i) \tag{4.14}$$

Backward algorithm. Let the backward variable $\beta_t(i)$ be:

$$\beta_t(i) = P(o(t+1)o(t+2)\ldots o(T), s(t) = S_i \mid \lambda) \tag{4.15}$$

the probability to observe a sequence $o = o(t+1),o(t+2),\ldots,o(T)$ and state in S_i at time t, given a model λ. This probability can be calculated recurrently:

1. Inizialization:

$$\beta_T(i) = 1, \quad 1 \le i \le m \tag{4.16}$$

where $I(o_1)$ is the index of the token in O equal to o_1.

2. Induction (Fig. 4.10):

$$\beta_t(i) = \left[\sum_{j=1}^{m} a_{ij}\, e_{jI(o(t+1))}\, \beta_{t+1}(j) \right], \quad 1 \le i \le m, \quad 1 \le t \le T-1 \tag{4.17}$$

3. Termination:

$$P(o \mid \lambda) = \sum_{i=1}^{m} \pi_i\, e_{iI(o(1))}\, \beta_1(i) \tag{4.18}$$

Computational complexity of forward-backward algorithm is $O(Tm^2)$.

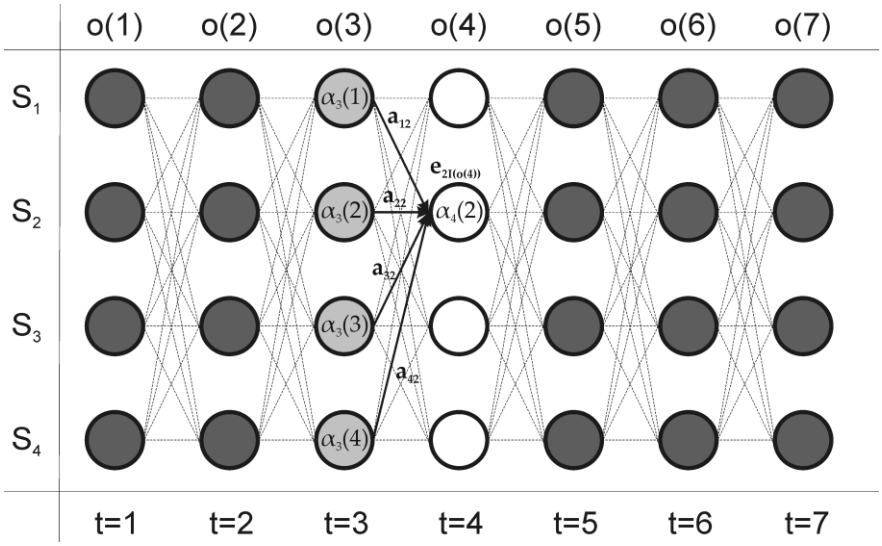


Fig. 4.9. The "trellis" for the forward algorithm. The forward variable $\alpha_{t+1}(j)$ is calculated as a function of the same variable at time t, according to Eq. 6.9.

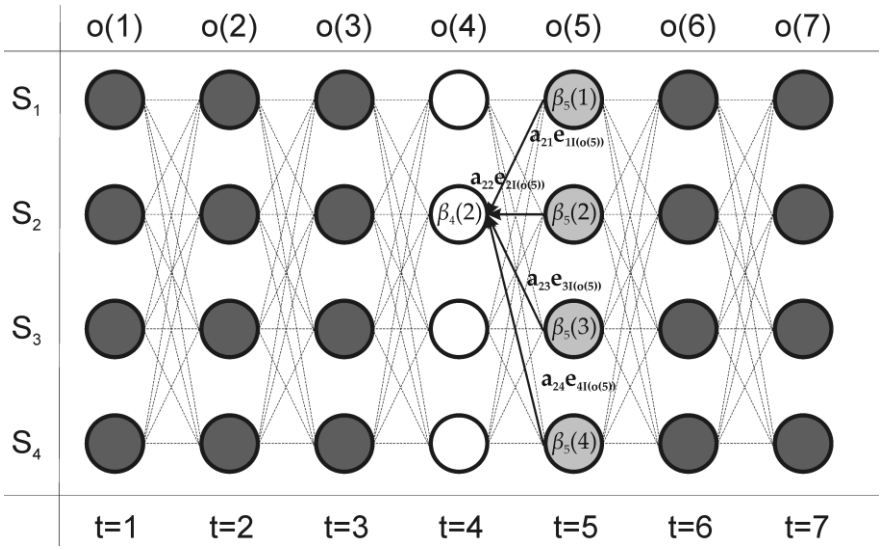


Fig. 4.10. The "trellis" for the backward algorithm. The backward variable $\beta_t(i)$ is calculated as a function of the same variable at time t+1, according to Eq. 6.13.

Decoding

Given an observed sequence $o = o(1),o(2),...,o(T)$ and a model λ, to find the most probable hidden state sequence $s = s(1), s(2), ..., s(T)$, the *Viterbi* algorithm (Viterbi, 1967; Forney, 1973) is applied. It is a dynamic

programming solution adapted to Markov model. Let us define the following variables:

$$\delta_t(i) = \max_{s(1),s(2),\ldots,s(t-1)} P(s(1)s(2)\ldots s(t) = S_i \mid o, \lambda) \tag{4.19}$$

that is the highest probability which accounts for the first t observations ending at time t in state S_i. By induction we have (Rabiner, 1989):

$$\delta_{t+1}(j) = [\max_i P(\delta_t(i)\, a_{ij})] \cdot e_{jI(o(t+1))} \tag{4.20}$$

To retrieve the state sequence, we need to keep track of the arguments maximizing Eq. 4.20 in the variable $\psi_t(j)$, for each t and j. *Viterbi* procedure for finding the best state sequence reads as follows:

1. Inizialization:

$$\delta_1(i) = \pi_i e_{iI(o(1))}, \quad 1 \le i \le m \tag{4.21}$$

$$\psi_1(i) = 0, \quad 1 \le i \le m \tag{4.22}$$

2. Recursion:

$$\delta_t(j) = \max_{1 \le i \le m} [\delta_{t-1}(i)\, a_{ij}] \cdot e_{jI(o(t))}, \quad 1 < t \le T, \quad 1 \le j \le m \tag{4.23}$$

$$\psi_t(j) = \operatorname{argmax}_{1 \le i \le m} [\delta_{t-1}(i)\, a_{ij}], \quad 1 < t \le T, \quad 1 \le j \le m \tag{4.24}$$

3. Termination:

$$P^* = \max_{1 \le i \le m} [\delta_T(i)] \tag{4.25}$$

$$q_T^*(j) = \operatorname{argmax}_{1 \le i \le m} [\delta_T(i)] \tag{4.26}$$

4. Path (state sequence) backtracking:

$$q_t^*(j) = \psi_{t+1}(q_{t+1}^*), \quad 1 \le t \le T-1 \tag{4.27}$$

Learning

Adjust parameters of the model λ to maximize $P(o \mid \lambda)$ for a given observed sequence $o = o(1),o(2),...,o(T)$. There is no optimal way to estimate the model parameters (Rabiner, 1989), but it is possible to choose them such that $P(o \mid \lambda)$ is locally maximized by means of a optimization algorithm, such as the *Baum-Welch* procedure (Baum et al., 1970), that is a particular case of a generalized *Expectation–Maximization* (*EM*) algorithm (Dempster et al., 1977). Let us define the variable:

$$\xi_t(i,j) = P(s(t) = S_i, s(t+1) = S_j \mid o, \lambda) \tag{4.28}$$

that represents the joint probability of state S_i at time t and state S_j at time t+1. From the definition of backward and forward variable we can write it as:

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} e_{jI(o(t+1))} \beta_{(t+1)}(j)}{P(o \mid \lambda)} = \frac{\alpha_t(i)\, a_{ij} e_{jI(o(t+1))} \beta_{(t+1)}(j)}{\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_t(i)\, a_{ij} e_{jI(o(t+1))} \beta_{(t+1)}(j)} \tag{4.29}$$

The posterior probability of state S_i at time t is:

$$\gamma_t(i) = \sum_{j=1}^{m} \xi_t(i,j) \tag{4.30}$$

This equation has the following properties: (i) the expected number of transitions from state S_i is equal to $\sum_{t=1}^{T-1} \gamma_t(i)$, while (ii) the expected number of transitions from state S_i to S_j is equal to $\sum_{t=1}^{T-1} \xi_t(i,j)$. Using above expressions one can re-estimate the parameters of *HMM*, closing the loop of the *EM*-type algorithm:

$$\bar{\pi}_i = \gamma_t(i) \tag{4.31}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{4.32}$$

$$\bar{e}_{ij} = \frac{\sum_{t=1,\, o(t)=Oj}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{4.33}$$

Chapter 5

Clustering

The main data mining target is to explore huge amount of data to retrieve patterns information. Clustering is an important task in data mining, dealing with division of data, in order to create groups of similar objects that are dissimilar to the objects belonging to the other groups. It makes use of tools and methodologies from statistics to numerical analysis, to model data or, informally, to summarize data information. Clustering plays an important role in a wide range of application areas, from biology, geophysics, information technology to marketing. Such applications usually deal with large datasets and many attributes that need to be explored with automatic or semi-automatic methods. The goal of this chapter is to provide a comprehensive description of the main clustering algorithms used in data mining. A final section will be devoted to the description of a development of a new clustering technique.

5.1 Introduction

Clustering can be considered the most important unsupervised learning process for the hidden patterns problem; it relates to the division of unlabeled data into groups (clusters) of similar objects. Clustering differs from classification because the latter assigns objects to predefined classes after a supervised learning process: clustering is an unsupervised form of *learning by observation*, while classification is *learning by example* (Han and Kamber, 2000). For real time classification, clustering spends a lot of time to explore the entire dataset and recognize classes, because learns on running; while a classifier makes use of acquired knowledge to perform fast classification. However, during classifier training, classes labeling can be expensive, in terms of human resources, and very difficult: in most cases, it is desirable to perform clustering on initial data to obtain groups

of similar objects for assigning labels, and then train a classifier by means of these labels.

Cluster analysis is an important activity in humans learning processes. As example, children learn, continuously and unconsciously, to distinguish between dogs and cats, or between animals and plants, applying clustering patterns, which for psychologists are known as *laws of Gestalt* (Hothersall, 2004). Humans perform competitively with automatic clustering procedures in two dimensions, but most real problems involve clustering in higher dimensions. It is difficult for humans to obtain an intuitive interpretation of data embedded in a high-dimensional space (Jain et al., 1999). For this reason, clustering algorithms play an important role in a broad range of applications in many fields. In marketing they can be used to find groups of customers with similar behavior; in biology for classification of plants and animals given their features; in seismology for earthquake studies: by clustering observed earthquake epicenters, to identify dangerous zones; on social networks to discover social circles based on links between users; on world wide web for document classification; in image processing for image segmentation. Each application uses a particular representation for data (images, graphs, time series, documents, etc.): in many cases, in fact, clustering term is associated with "segmentation" (for images) or "partitioning" (for graphs) or to "pattern (or motif) discovery" when dealing with time series.

Algorithms are applied in different manner, in relation to the managed data type, and to the purpose of results. A very important feature in this sense is the choice of a *distance* or *similarity* measure (see Chapter 2) for objects comparisons; if it does not exist, algorithms have to define some, which is not always easy, especially for data with high number of attributes or containing different types of attributes.

Managing different types of attributes is not the only requirement. Clustering algorithms need to overcome several issues: scalability; easy parameters setting; presence of noise data or outliers; dynamic systems; high-dimensional data. Unfortunately, it is very difficult to satisfy all these requirements. Indeed, current clustering techniques do not address all the requirements adequately, because there is no one universally applicable.

5.2 Definitions

The main goal of clustering is to return a partition $P = \{C_1, C_2, \ldots, C_k\}$ of dataset D, of cardinality k such that:

$$D = C_1 \cup C_2 \cup \ldots \cup C_k \,, C_i \cap C_j = \varnothing \quad i \neq j \tag{5.1}$$

The above definition (Berkhin, 2002) gives a strict sense to clustering, because one object can belong only to one cluster. There are other approaches, such as the *fuzzy* clustering (Yang, 1993), that relaxes this definition allowing an object to belong to more clusters, each with a corresponding probability. Generally, the definition of cluster combines several reasonable criteria (Gan et al., 2007): cluster objects share the same or closely related properties (attributes); show small mutual distance (or high similarity); and are clearly distinguishable from the complement (the rest of the objects in the dataset). For example, some researchers (Ester et al., 1996) suggested, for spatial databases, that datasets contain clusters if there are continuous and relative densely populated regions of the space, and these are surrounded by continuous and relatively empty regions of the space.

Distance measures play an important role in data clustering, since it is fundamental to the definition of a cluster. To compare a pair of objects, a definition of relationship is needed. On literature this relation is referred to *similarity* or *distance* measure. When grouping similar objects, using similarity, the goal is to maximize the measure value; on the contrary, using distance, the goal is to minimize the measure value.

5.3 Clustering methods

In literature a wide range of clustering techniques was proposed. In this section we want to provide a categorization of principal types of clustering, and focus on description of the main techniques. The first thing to consider, when classifying methods, is the type of clustering result. In this sense, can be distinguished *hierarchical* from *partitioning* methods, *exclusive* from *overlapping* and *fuzzy* methods, and *complete* versus *partial* methods (Tan et al., 2005).

Clustering returns a partition $P = \{C_1, C_2, \ldots, C_k\}$ of dataset D. *Partitioning* methods return a simple division of non-overlapping subsets, each of which is a cluster, according to (Eq. 5.1). *Hierarchical* methods divide the dataset in several levels of partitioning $P_1, P_2, \ldots, P_l$, and establish a hierarchy between clusters, in the form of a tree, also called *dendrogram* (Fig. 5.1): a cluster can be a subset of a greater cluster, or contains other subsets corresponding to minor clusters.

Exclusive clustering is different from *overlapping* clustering, because the former does not allow any intersection of objects between clusters (see Eq. 5.1) while, for the latter, objects can belong to different clusters. Similar concept to overlapping we found for *fuzzy* clustering, but in this case clusters are considered *fuzzy sets*. In mathematics, an object x belongs to a fuzzy set C_i with a certain probability p (Zadeh, 1965): $p(x \in C_i) \in [0,1]$, and the sum of all probabilities of x, calculated with respect to each cluster, must sum to 1:

$$\sum_{i=1}^{k} p(x \in C_i) = 1 \qquad (5.2)$$

Last distinction can be found between *complete* clustering and *partial* clustering. In the first case, every object in D is assigned to a particular cluster C_i in P. In the second case, there are unclassifiable objects that are assigned to any cluster: these objects are considered *outlier* or *noise* data.

However, since many concepts may overlap among clustering methods, it is useful to present an organized classification of different clustering methods, based on their definition of cluster:

1. Partitioning methods;
2. Hierarchical methods;
3. Density-based methods;
4. Graph-based methods;
5. Grid-based methods;
6. Model-based methods;
7. Subspace-clustering methods;
8. Neural-network clustering methods.

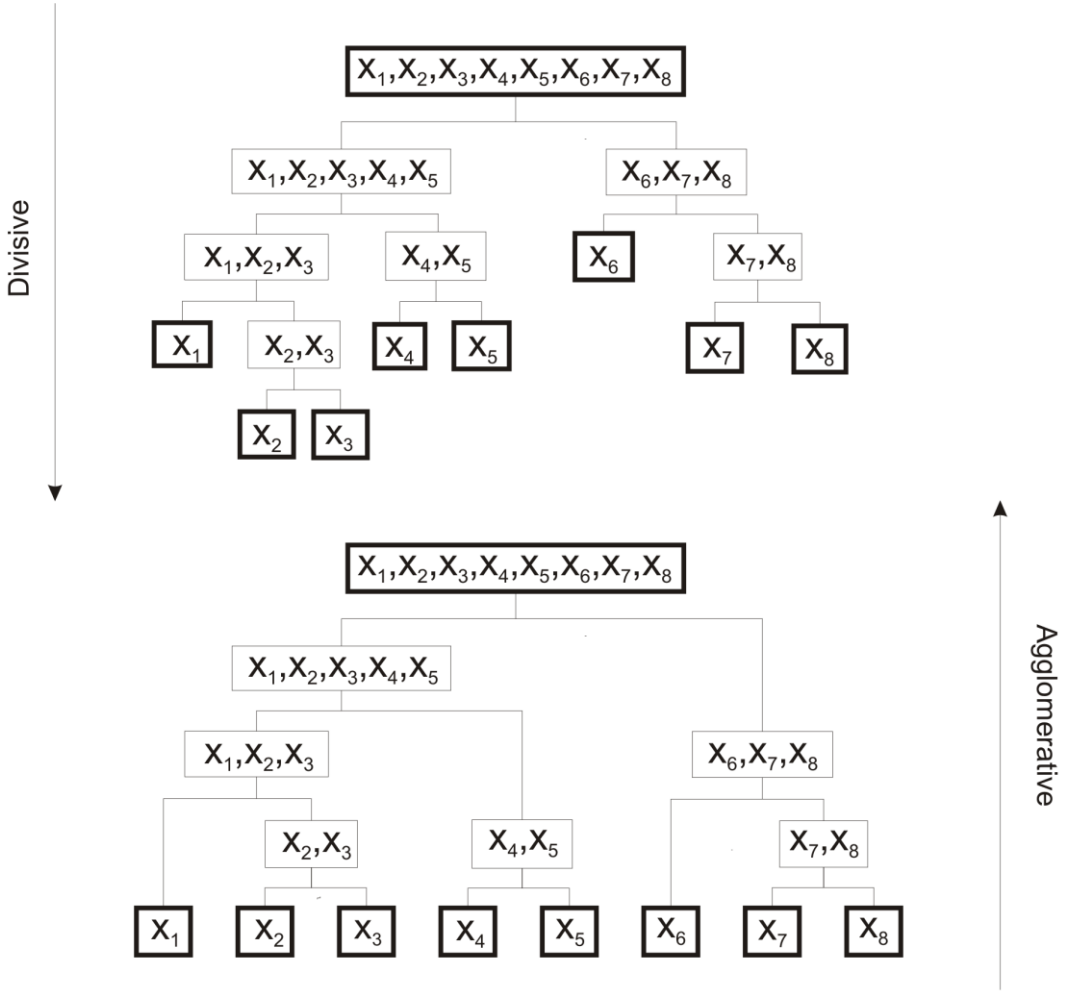


Fig. 5.1. Divisive versus agglomerative hierarchical clustering (see Section 3.2). An example of iterations for a dataset $D = \{x_1, x_2, \ldots, x_8\}$. The constructed hierarchy tree is often called *dendrogram*.

Partitioning, hierarchical, density-based, graph-based and grid-based methods will be explained respectively from Section 5.3.1 to Section 5.3.5. In Section 5.3.6 we will introduce the remaining approaches: model-based, subspace clustering, and neural network clustering. In Section 5.3.7 we will give a brief introduction to the techniques and measures used for validation of clustering results.

5.3.1. Partitional Clustering

Partitioning methods have long been the most popular algorithms before the advent of data mining. The most well-known and commonly used

partitioning methods are *k-Means*, *k-Medoids*, and their variations (Han and Kamber, 2000). These algorithms follow the same steps when perform clustering:

1. Choose arbitrarily k random objects from dataset of N objects, each representing a cluster center, as initial solution;
2. Assign an object to a cluster, according to the nearest center;
3. Compute the new cluster centers after each assignment;
4. Iterate the (2) and (3) steps, until the cluster centers do not change (convergence criterion).

This kind of algorithm has high computational complexity, and results are inadequate for clustering of large dataset. In literature, other partitioning algorithms try to address the scalability problem, typically using techniques of data sampling, among them: *CLARA* (*Clustering LARge Applications*; Rousseeuw and Kaufman, 1990), and *CLARANS* (*Clustering Large Applications based on RANdomized Search*; Ng and Han, 1994) which obtains good clustering results with a computational complexity of $O(N^2)$ in the worst case.

K-Means. For *k-Means* algorithm (MacQueen, 1967) the centers, treated at the above step (2), correspond to the calculated means of each cluster. So, the convergence is reached when the following value, called *square error function*, does not change during iterations:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2 \qquad (5.3)$$

where m_i denotes the mean of cluster C_i. The main drawback of this approach is the high sensitivity to outliers, because they can distort the distribution of calculated centers. The achievement of the convergence criterion is a *NP-complete* problem (Garey et al., 1982), so the real implementation of these algorithms needs to stop after a certain number of steps t. The total complexity of this algorithm is $O(Nkt)$.

K-Medoids. *k-Means* performances are much altered in presence of outliers. To diminish such sensitivity, instead of consider a cluster means as centers, is useful to pick a representative object (often the median) from cluster and elect it as center. In this case, the center is a real data object, not imaginary. Differently from k-Means, at each iteration, the centers are not calculated. Every time, a new representative object o_{random} is selected from data and replaced with another existent representative object o_j, only if o_{random} error results less than o_j error. The error function will be calculated as:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - o_i)^2 \qquad (5.4)$$

where o_i denotes the representative object of cluster C_i. *PAM* is the first algorithm to implement *k-Medoids* method. The complexity of each iteration is $O(k(N-k)^2)$ (Han and Kamber, 2000). For large values of N and k, such computation becomes very costly. To deal with larger data sets, it is necessary to sample data. *CLARA* is a more efficient implementation of *k-Medoids* method working only with a small portion, chosen from the data (a sample). If the selected sample is a good representative, the clustering result will be similar to PAM, but with a computational time complexity $O(ks^2+k(N-k))$, where s is the sample size (Han and Kamber, 2000). However, it is not always easy to make a good choice for the representative sample (Gonzalez, 1985).

Another improvement of *k-Medoids* algorithm is *CLARANS*. Respect to *CLARA*, where the selected sample is fixed for all iterations, *CLARANS* performs a selection of a random sample in each iteration.

One of the limits of partitioning methods is the a-priori establishing of the number k of clusters. It is not a simple choice to understand the optimal number of clusters into a dataset, and the literature offers a wide range of techniques to perform a comparison between different clustering results (an exhaustive review can be found in Halkidi et al., 2001). An interesting feature of this method is the determination of the "natural" number k_{nat} of clusters in a database. They propose to run *CLARANS* once for each k from 2 to N. For each clustering result, a clustering validity measure, called *silhouette coefficient* (Rousseeuw and Kaufman, 1990), is calculated. The

clustering with maximum score is chosen as the "natural" clustering. Unfortunately, this enhancement adds computational complexity, and the running time of this approach is also prohibitive for large N, because it is has $O(n^2)$ complexity.

Both *k-Means* and *k-Medoids* methods are centroid based, and their application is useful when clustering dataset contain spherical shape clusters. They obtain low quality of clustering when deal with clusters of arbitrary shape.

5.3.2. Hierarchical Clustering

Hierarchical clustering algorithms can be classified into two main types: *agglomerative* and *divisive* (Fig. 5.1). In agglomerative (or *bottom-up*) hierarchical clustering, the two most similar clusters are merged to form a large cluster at each step, and this processing is continued until the desired number of clusters is obtained. In a divisive (or *top-down*) hierarchical algorithm, the process is reversed by starting with all data points into one cluster, and subdividing it into smaller clusters by several criteria (Gan et al., 2007).

In both cases, it is necessary to establish a way to measure distance between an object and a cluster or between two clusters. Let us start by taking reference on agglomerative technique (for divisive technique will be the exact opposite). Let be D a dataset of size N, $P_0 = \{C_1, C_2, \ldots, C_{n0}\}$ the first partitioning of D, where each object corresponds to a cluster ($n_0 = N$), and $P_l = \{C_1, C_2, \ldots, C_{nl}\}$, the partitioning of D at level l ($n_l < n_{l+1} \leq n_0$). The iterative criterion is to find, at each level l, the most similar pair of cluster C_r and C_s ($0 < r, s < n_l$) such that the distance between them is minimized. The following subsections describe the most popular ways to form hierarchies between clusters.

Mean Distance (Fig. 5.2a). By this method, distance between clusters is calculated by calculating distance between cluster centers. It is used for the average-linkage hierarchical clustering. Let be μ the cluster center of C, calculated as:

$$\mu = \frac{1}{|C|}\sum_{i=1}^{|C|} x_i, \quad x_i \in C \tag{5.5}$$

Then, the distance between C_r and C_s is defined as:

$$d(C_r, C_s) = d(\mu_r, \mu_s) \tag{5.6}$$

Minimum Distance (Fig. 5.2b). Used for single-linkage hierarchical clustering. In this case, distance between clusters C_r and C_s is calculated taking into account the pair of object x_i and x_j, belonging respectively to C_r and C_s, which are closer:

$$d(C_r, C_s) = \min_{(0<i\leq|C_r|,\ 0<j\leq|C_s|)} \left| d(x_i, x_j) \right| \tag{5.7}$$

Maximum Distance (Fig. 5.2c). Opposite to the minimum distance criterion, it is used for complete-linkage hierarchical clustering. In such a way, the distance between clusters C_r and C_s is calculated taking account of the pair of object x_i and x_j, belonging respectively to C_r and C_s, that are farthest:

$$d(C_r, C_s) = \max_{(0<i\leq|C_r|,\ 0<j\leq|C_s|)} \left| d(x_i, x_j) \right| \tag{5.8}$$

Average Distance (Fig. 5.2d). In this case, the distance between clusters C_r and C_s corresponds to the average distance of all pair distances between x_i and x_j, belonging to C_r and C_s, respectively:

$$d(C_r, C_s) = \frac{1}{|C_r||C_s|}\sum_{i=1}^{|C_r|}\sum_{j=1}^{|C_s|} d(x_i, x_j) \tag{5.9}$$

Two algorithms which specify agglomerative and divisive methodologies are *AGNES* (AGglomerative NESting) and *DIANA* (DIvisive ANAlysis; Rousseeuw and Kaufman, 1990), among the first to implement the hierarchical clustering. *AGNES* and *DIANA* today are exceeded, because do not address to several issues, such as the recognition of clusters of arbitrary shape (natural clusters), and are very sensitive to the presence of

noise or outliers. Therefore, other sophisticated techniques were proposed. In particular, for hierarchical clustering, we want to mention *BIRCH* (*Balanced Iterative Reducing and Clustering using Hierarchies*; Zhang et al., 1997), *CURE* (*Clustering Using REpresentatives*; Guha et al., 1998), *ROCK* (*RObust Clustering using linKs*; Guha et al., 1999) and *CHAMELEON* (Karypis et al., 1999).

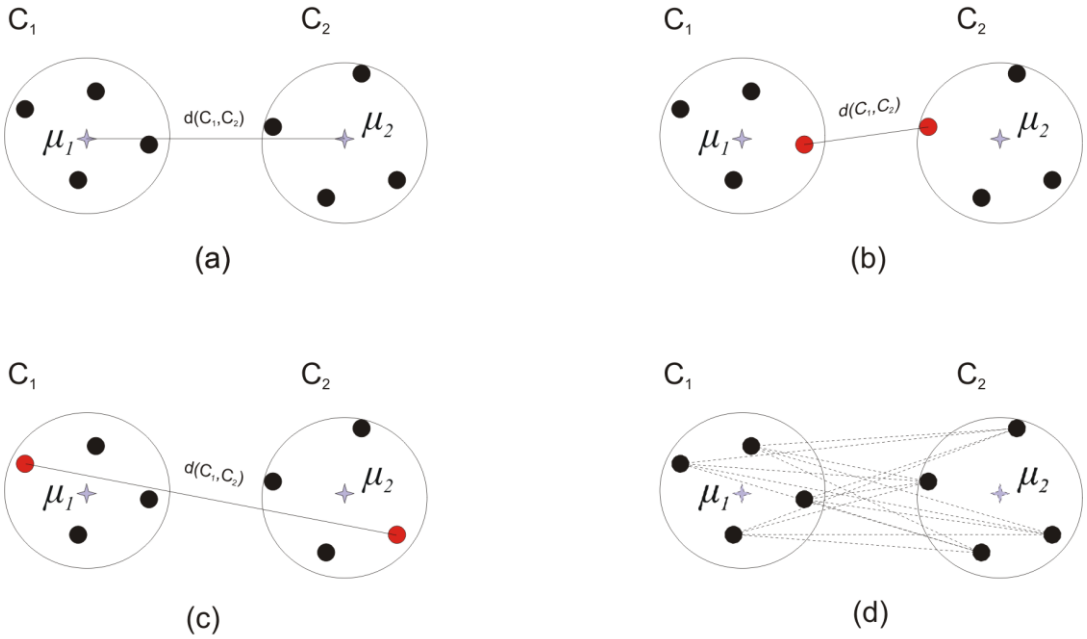


Fig. 5.2. Distance measure between clusters. (a) Mean distance. (b) Minimum distance. (c) Maximum distance. (d) Average distance.

BIRCH. *BIRCH* is a hierarchical clustering algorithm, whose best peculiarity is the high data compression level. It is best indicated when data objects are vectors of numbers.

The basic concept about this method is to group data in several sub-clusters, whose number is much less than dataset cardinality, into a balanced tree data structure, called *CFTree*. Each node of the tree, called *Cluster Feature* (*CF*), is a light data structure, saved on the host device central memory, containing statistics about sub-tree starting from it. These two structures allow speed-up operations of inserting, deleting and searching, especially for large dataset, since operations in balanced tree have $O(\log N)$ computational complexity. *CF* is a record $<n, LS, SS>$ summarizing as follow: number n of points of the sub-cluster, a vector *LS* representing the linear sum of sub-cluster's objects, and a vector *SS*

representing the square sum of sub-cluster's objects. These clustering features are sufficient to calculate all of the measurements needed for making clustering decisions in *BIRCH* (Han and Kamber, 2000).
The *CFTree* management requires the setting of two parameters: *B* (*branching factor*), and *T* (*threshold*), indicating respectively the maximum number of children for internal tree nodes, and the maximum diameter of sub-clusters contained in the leaf nodes. There are two main phases in the *CFTree*: (i) the initialization phase, where data are disposed into a hierarchy of clusters, corresponding to the tree structure; and (ii) the updating phase, where all operations of insertion are performed. If a leaf node have a diameter greater than *T*, it is necessary to perform a split of the sub-cluster.
The overall complexity of *BIRCH* algorithm is *O(N)*. The main drawbacks of this method are: (i) it has limited effectiveness when dealing with clusters of arbitrary shape, because of using notion of diameter for clusters boundary; (ii) outliers or noise data can bring to several undesirable splitting operations. It is more robust when managing spherical clusters.

CURE. Algorithms often make use of a central cluster object to represent the entire cluster (i.e. the mean object). *CURE* algorithm makes use of a set of well distributed points to represent a cluster, instead of a single object.
This feature produces two immediate positive results: (i) it is possible to discover also clusters of ellipsoidal shape (but not of arbitrary shape yet); (ii) the clustering is less sensible to noise, even if it is not specific for outlier discovery.
It is an agglomerative method, so initially each input object is a cluster. In order to compute the distance between a pair of clusters, for each cluster, *c* representative points are stored. These are determined by first choosing *c* well scattered points within the cluster, and then shrinking them toward the mean of the cluster by a fraction α (Han and Kamber, 2000). The distance between two clusters is then the distance between the closest pair of representative points (Eq. 5.7; Guha et al., 1998). Its computational complexity is $O(N^2 \log N)$ in the worst case. The parameter α is used to control the shapes of clusters.

ROCK. *ROCK* is born to deal with categorical data. *ROCK* performs distance comparisons for clustering, by considering the neighborhoods of each data object. The *ROCK* idea is that, if two objects x, y, have similar neighbors, then they are very similar and can be merged. The used similarity measure, called *Jaccard's coefficient*, is the exact opposite of the distance introduced in (Eq. 2.20). In this case we consider the number of same neighbors as the number of same attributes:

$$sim(x,y) = 1 - \frac{r+s}{q+r+s} = \frac{q}{q+r+s} \tag{5.10}$$

If the *Jaccard's coefficient* between x and y exceeds a specified threshold Θ (i.e. $sim(x,y) \geq \Theta$), then x and y are merged in the same cluster. Based on this definition, after calculating the similarity matrix for each pair of objects, *ROCK* first builds links between objects having similarity greater than Θ, to obtain a sparse graph of similarity. Then, it performs agglomerative hierarchical clustering on this graph. Guha et al. (1999) demonstrated that the worst-case time complexity of *ROCK* is $O(N^2+Nm_m m_a+N^2 logN)$, where m_m and m_a are the maximum and average number of neighbors, respectively, and N is the number of objects.

CHAMELEON. *CHAMELEON* guarantees better performances than previous agglomerative hierarchical clustering algorithms, since it overcomes the principal limitations of *ROCK* and *CURE* algorithms. *CURE* is able to find clusters of different shape and size, because it uses the *closeness* relation (Eq. 5.7), which emphasizes clusters proximity, to perform hierarchical agglomeration. However, it ignores information about the *interconnectivity* of items in two clusters. Contrariwise, *ROCK* ignores information about the *closeness* of two clusters, because it uses links (or similar neighbors) to define similarity: even if a pair of objects x and y in different clusters are neighbors, it is very unlikely that the pairs have a large number of common neighbors. For the definition of links, the *interconnectivity* between clusters is emphasized, but proximity between clusters is ignored. Chameleon tries to overcome weakness of both algorithms, by taking account of both *interconnectivity* and *closeness* when identifying clusters. The Chameleon algorithm has four main phases (Fig. 5.3):

1. Calculates for each data object, its first *k* more similar objects, or *kNN* (*k Nearest Neighbors*);
2. Builds the *k-nearest-neighbors* graph, where each node corresponds to a data object. An edge between nodes indicates that the two represented objects share more than *k* neighbors;
3. Partitions graph into clusters, by minimizing the *edge cut* (see Section 5.3.4): given a cluster *C*, it corresponds to find the minimum weight of edges, whose elimination makes two disconnected partitions C_i and C_j from *C*. Karypis et al. (1999) refer to this value as *absolute interconnectivity* $EC(C)$ or $EC(C_i, C_j)$;
4. Applies the agglomerative hierarchical clustering, by iteratively merging pairs of clusters whose *relative interconnectivity RI*, multiplied by the *relative closeness RC,* is the highest. The iteration stops when there are only a user-defined number of clusters.

The edge cut measures the absolute interconnectivity of two partitions. The relative interconnectivity $RI(C_i, C_j)$ between two clusters C_i and C_j, is calculated as the ratio of the absolute interconnectivity between the two clusters $EC(C_i, C_j)$, and the mean of internal interconnectivities of each single cluster $EC(C_i)$ and $EC(C_j)$:

$$RI(C_i, C_j) = \frac{\left|EC(C_i, C_j)\right|}{\frac{\left|EC(C_i)\right| + \left|EC(C_j)\right|}{2}} \tag{5.11}$$

By using relative interconnectivity, Chameleon overcomes the limitation of establish an absolute interconnectivity value for all clusters, making it flexible to the discovery of cluster of different shapes and densities.
In the same way we can calculate the relative closeness defining the absolute closeness of clusters $SEC(C_i, C_j)$, that is the average weight (opposed to the sum of weights for interconnectivity) of the edges that connect vertices in C_i to those in C_j. The relative closeness is defined as:

$$RC(C_i, C_j) = \frac{\left|SEC(C_i, C_j)\right|}{\frac{|C_i|}{|C_i|+|C_j|} SEC(C_i) + \frac{|C_j|}{|C_i|+|C_j|} SEC(C_j)} \tag{5.12}$$

The average weight computation makes the algorithm more tolerant to the presence of outliers and noise, especially if we consider that connections between clusters come from the k-nearest-neighbors graph. In the first phase of the algorithm, the graph construction highlights the identity of outliers and noise. The computational complexity of this algorithm is $O(N^2)$ in the worst case.

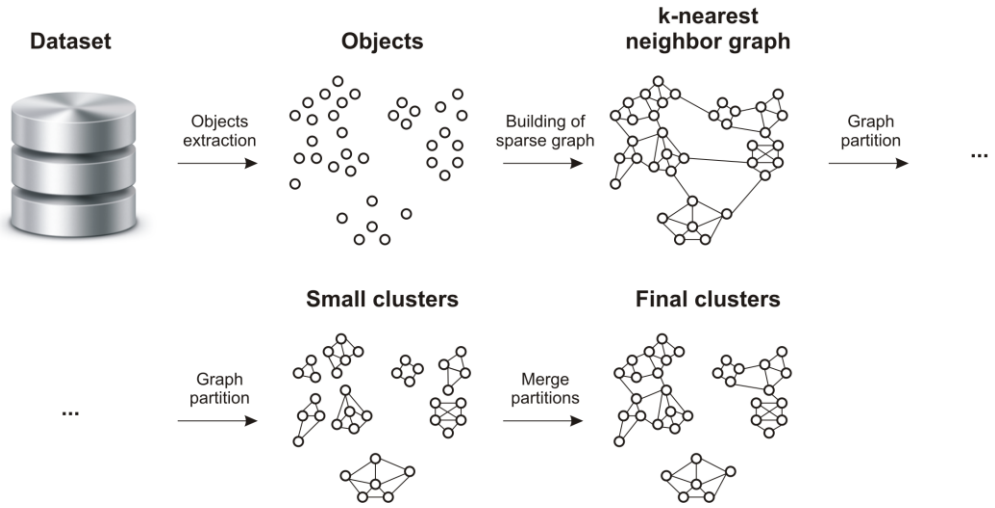


Fig. 5.3. An illustration of Chameleon phases. Redrawn from Karypis et al. (1999).

5.3.3. Density-based Clustering

Density-based clustering allows discovering clusters of arbitrary shape. Here, clusters are defined as dense regions of objects in the data space, separated by regions of low density, which represent noise. The basic idea of density-based algorithms, first proposed in *DBSCAN* (Ester et al., 1996), is to grow a given cluster guaranteeing that the density in its neighborhood, represented by the surrounding number of objects, exceeds some specified threshold. This kind of algorithm is able to detect clusters of arbitrary shape. A further product of clustering is to filter out outliers (or noise). Since outliers carry useful hidden knowledge related to a potential abnormal behavior, their detection has been applied in fields

such as fraud detection, intrusion discovery, marketing, and pharmaceutical testing.

In the following subsection we will examine three principal algorithms, basilar for the development of further enhanced density-based methods, focusing on cluster definitions rather than the computational complexity: *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*; Ester et al., 1996), *OPTICS* (*Ordering Points To Identify the Clustering Structure*; Ankerst et al., 1999) and *DENCLUE* (*DENsity based CLUstEring;* Hinneburg and Keim, 1998).

DBSCAN. *DBSCAN* relies on the idea that objects, which form dense regions, should be grouped together into clusters. Usually *DBSCAN* runs on data sets drawn from a metric space and uses a distance function to compare objects. Given a data set *D* of *N* objects, and two fixed threshold values, ε and *MinPts, DBSCAN* makes use of the following structures and definitions (Ester et al., 1996): (i) *ε-neighborhood*, (ii) *core object*, (iii) *directly density-reachable*, (iv) *density-reachable* and (v) *density-connected* clusters.

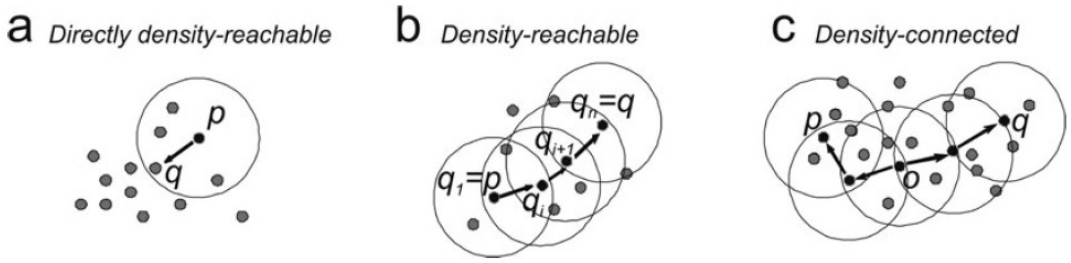


Fig. 5.4. Examples of (a) *directly density-reachable*, (b) *density-reachable* and (c) *density-connected* in density-based clustering (from Cannata et al., 2011a). Suppose *MinPts* = 3. Grey and black dots indicate the points to group into clusters, black circles delineate the area of radius ε around black dots, the arrow denotes the relation of direct density-reachability. In (a) dot *p* is the so-called core point, while *q* is directly density-reachable from *p*. In (b) dot *q* is density-reachable from *p*. In (c) dot *q* is density-connected to *p* and *o* is a point such that both *p* and *q* are density reachable from *o*.

DBSCAN defines the *ε-neighborhood* of an object *p* as the set of objects N_ε that fall within a circle of radius ε, centered in *p*. If $|N_\varepsilon| \geq MinPts$ then *p* is called a *core object*. All points in N_ε are called *directly density reachable* from *p* (Fig. 5.4a). An object *p* is *density-reachable* from an object *q* if there is a chain of objects $p_1,...,p_m$, $p_1 = q$, $p_m = p$ such that each p_{i+1} is directly density-

reachable from p_i, for $1 \leq i \leq m$ (Fig. 5.4b). A point p is *density-connected* to a point q if there is a point o such that both, p and q, are density reachable from o (Fig. 5.4c). A cluster is a maximal set of density-connected objects.

DBSCAN scans once object in D, and for each object p, it checks its N_ε. When N_ε contains more than *MinPts*, it creates a new cluster with p as core point, and iteratively collects directly density-reachable points from p. The process terminates when no new points can be added to any cluster. A point in D is an outlier if does not belong to any cluster. The space can be indexed to speed up the *kNN* search. In such a case the computational complexity of *DBSCAN* is $O(N \log N)$, where N is the number of points in the dataset. Otherwise, it is $O(N^2)$.

The basic structure of *DBSCAN* presents a particular shortcoming when clusters having different densities have to be discovered (Fig. 5.5). Since the definition of density is set at the beginning, by parameters *MinPts* and ε, such a global setting could cause low quality clustering. Usually, empirical parameters setting is difficult to determine, especially for real-world or high-dimensional data.

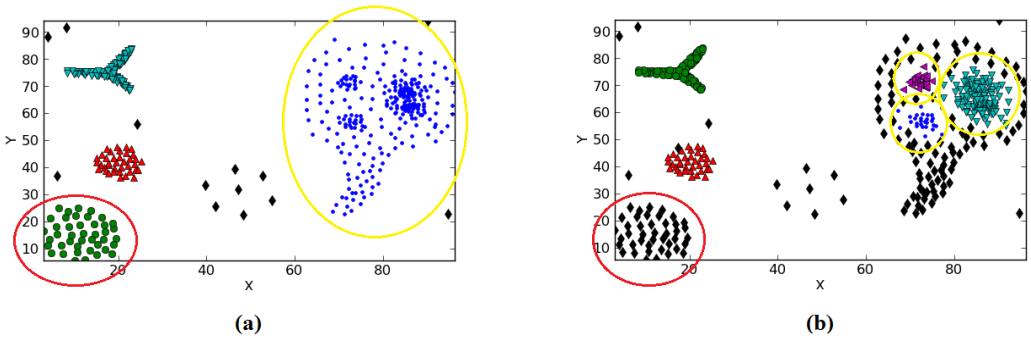


Fig. 5.5. An example of clustering with *DBSCAN*, using *PyDBSCAN* software (Cassisi et al., 2011a). The dataset presents clusters of different densities. (a) *MinPts* = 10 and ε = 8. The ε is large enough to permit *DBSCAN* to find the green cluster (into red circle), but is too large to distinguish different clusters into the blue cluster (into yellow circle). (b) *MinPts* = 10 and ε = 15. The ε is small enough to permit *DBSCAN* to find the three clusters circumscribed by yellow circles, but is too small to detect the cluster indicated by the red circle.

Many other sophisticated techniques have been proposed to overcome these limitations: *SNN* (Ertoz et al., 2003) meets the problem of variety in density and high-dimensionality. Other approaches, including automatic

parameter settings, can be found in (Vadapalli et al., 2006) and (Cassisi et al. 2012b; see also Section 5.5.2). The notion of density-connectivity of *DBSCAN* has been used also for subspace clustering (Zimek, 2008; see also Section 5.3.6).

OPTICS. In the previous subsection, it has been noted how parameter setting can affect clustering results on *DBSCAN*. In many cases, to understand the best combination of parameter values *MinPts* and ε, it should be convenient to run several executions of the algorithm. Unfortunately, the search for the different pairs of parameters can bring to long computation time. However, it has also been noted how a single pair of parameters can be adequate only partially, when different densities are present into the dataset. To overcome these problems, *OPTICS* algorithm is used. Even if it is not a real clustering algorithm, it makes an ordered list of the dataset objects, which reflects the density structure of dataset clusters. This structure contains information, which can be extractable running *DBSCAN* with a wide range of density levels.
Fixed a value for *MinPts*, and a sufficient large value for ε, it stores for each point two interesting values, the *core-distance* and the *reachability-distance*, first defined by Ankerst et al. (1999), representing the relative density value.
Let p be an object from a database D, let ε be a distance value, let $N_\varepsilon(p)$ be the *ε-neighbourhood* of p, let *MinPts* be an integer number and let $k_{dist}(p)$ be the distance from p of its kth nearest neighbour. The core-distance cd, with respect to ε and *MinPts*, of object p is defined as:

$$cd_{\varepsilon,MinPts}(p) = \begin{cases} UNDEFINED & |N_\varepsilon(p)| < MinPts \\ k_{dist}(p) & otherwise \end{cases} \tag{5.13}$$

The core-distance is the smallest distance ε' between p and an object in its *ε-neighbourhood* such that p would be a core object.
Let p and o be objects from a database D, the reachability-distance rd, of p with respect to ε, *MinPts*, and o is defined as:

$$rd_{\varepsilon,MinPts}(p,o) = \begin{cases} UNDEFINED & |N_\varepsilon(o)| < MinPts \\ \max[cd(o), dist(o,p)] & otherwise \end{cases} \tag{5.14}$$

The reachability-distance of p is the smallest distance such that p is density-reachable from a core object o.
The above values are useful for *DBSCAN* to extract clusters for each $\varepsilon' < \varepsilon$. Furthermore, the reachability plot offers a high informative content (Fig. 5.6). Algorithms based on its study, such as *ξ-clustering* (Ankerst et al., 1999), *cluster_tree* (Sander et al., 2003), or that proposed by Brecheisen et al. (2004), can catch, if exist, hierarchies between clusters and then build the relative *dendrogram*.

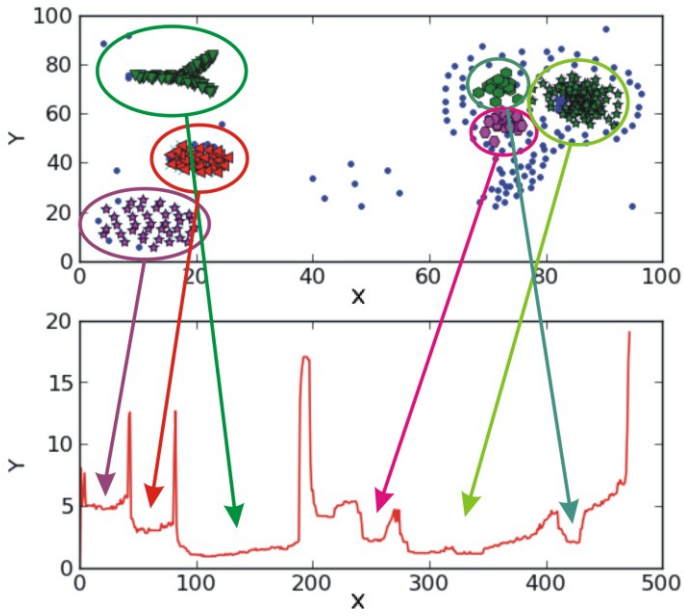


Fig. 5.6. An example of the reachability plot (red line on bottom plot) for a dataset of N = 473 objects. "Valleys" on the red line correspond to the clusters.

DENCLUE. The basic idea of this algorithm is to establish how objects can influence each other: greater the influence, the shorter the distance among them, and vice versa. The algorithm needs a definition of "influence" function for each point, which indicates this relation. At this point, it is intuitive to understand that clusters are disposed into space regions where the sum of objects influence is high. *DENCLUE* bases on the following definitions: (i) *basic influence function* (ii) *density function*, (iii) *gradient*, (iv) *density attractor*, and (v) *density attracted*.
Let be x and y two objects of dataset D, the *basic influence function* $f_B(x,y)$ describes the influence of a data object y, respect to x. Hinneburg and

Keim (1998) provided two examples of basic influence function, the *Square Wave* influence function and the *Gaussian* influence function:

$$f_{Square}(x,y) = \begin{cases} 0 & if\ d(x,y) > \sigma \\ 1 & otherwise \end{cases} \tag{5.15}$$

$$f_{Gauss}(x,y) = e^{\frac{d(x,y)^2}{2\sigma^2}} \tag{5.16}$$

Both functions depend on σ: greater its value, the greater the influence of y on x.
The *density function* $f_B^D(x)$ indicates the influence of the entire dataset on x. It can be calculated as the sum of the basic influence function of all data objects in D, respect to x:

$$f_B^D(x) = \sum_{i=1}^{N} f_B(x_i, x) \tag{5.17}$$

In real applications, the computation of density function $f_B^D(x)$ needs to be approximated to $f_B^{kNN(x)}(x)$, since it is very expensive to compute it for all objects in dataset: a density information for x can be taken by exploring only its neighbors, so (Eq. 35) can be transformed in:

$$f_B^D(x) \approx f_B^{kNN(x)}(x) = \sum_{x_i \in kNN(x)} f_B(x_i, x) \tag{5.18}$$

The *gradient* of the density function f_B^D is defined as:

$$\nabla f_B{}^D(x) = \sum_{i=1}^{N} (x_i - x) \cdot f_B(x_i, x) \tag{5.19}$$

Clusters can then be determined mathematically by identifying *density attractors*, where density attractors are local maxima of f_B^D. An object x is *density attracted* from a density attractor x^*, if there is a chain of object x_0, x_1, …, x_k, such that the gradient of x_{i-1} is in the direction of x_i (or have the same sign), for each $0 < i < k$. More formally:

$$x_0 = x, x_k = x^*, \quad x_i = x_{i-1} + \delta \cdot \frac{\nabla f_B^D(x_{i-1})}{\left\| \nabla f_B^D(x_{i-1}) \right\|}, \quad 0 < i < k \tag{5.20}$$

In what follows we can give two definitions of cluster for *DENCLUE* (Hinneburg and Keim, 1998):

1. Given parameters σ and ξ, a *center-defined* cluster of a density attractor x^*, is a subset of object C of D, such that for each x in C, x is density attracted from x^*, and $f_B^D(x^*) \geq \xi$. All objects in D density attracted by an x_0^* having $f_B^D(x^*) \leq \xi$, are considered *outliers*.
2. Given a set of density attractors X, parameters σ and ξ , an *arbitrary shape* cluster is a subset of object C of D, which satisfies the two following conditions: (i) for each x in C, exists a x^* in X, such that x is density attracted from x^*, and $f_B^D(x^*) \geq \xi$; (ii) for each pair x_1^* and x_2^*, exist a path P of objects in D, from x_1^* to x_2^*, such that $f_B^D(p) \geq \xi$, for each p in P.

It is interesting to observe the general formal structure of *DENCLUE*. Depending on the basic influence function, and on the chosen parameters σ and ξ, this algorithm traces other clustering methods. For example, using the square wave influence function, the search for arbitrary shape clusters coincides with *DBSCAN*, with *MinPts* = ξ, and $\varepsilon = \sigma$. Indeed, the density attractor definition equals to the core object definition, while density attracted definition equals to directly density-reachable definition. More, using different values of σ (Fig. 5.7), it is possible to produce hierarchies of cluster, typical in hierarchical clustering: starting with small σ values, for smaller clusters, going to greater values of σ, to find cluster even larger, until all objects are merged in only one cluster: the root of the dendrogram.

To speed up computations, space can be indexed, by creating a map of the space, and memorizing it into a tree indexing structure. The first step consists of data space subdivision into hypercubes with an edge length of 2σ. In this sense, it can be also considered a grid-based algorithm (see Section 3.5). In the light of it, the density function is calculated for each populated hypercube H. A hypercube H is said to be a highly populated

hypercube if $|H| \geq \xi_c$, where ξ_c is an outlier bound. Two hypercubes H_1 and H_2 are said to be connected if $d(mean(H_1), mean(H_2)) \leq 4\sigma$.

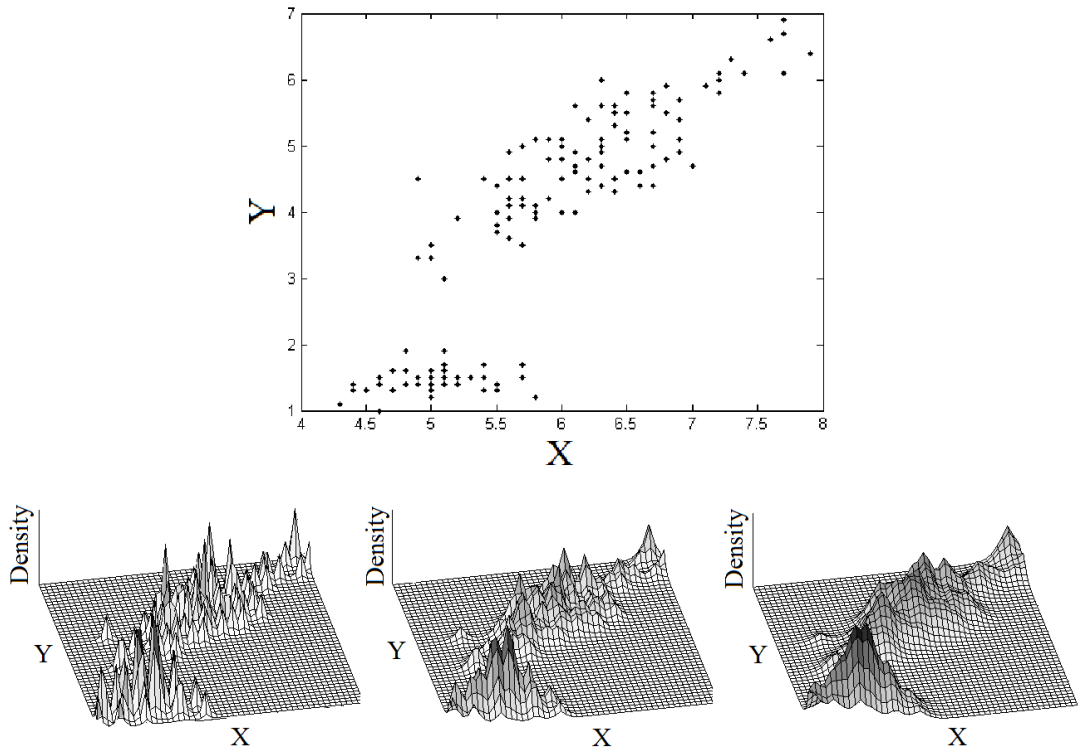


Fig. 5.7. On top, the plotting of *Fisher's Iris* dataset (Fisher, 1936), where *X* corresponds to the first attribute (sepal length), and *Y* to the third (petal length). On the bottom, from left to right, several density function distributions on grid, using an even larger σ. Redrawn from Hinneburg and Keim (1998).

5.3.4. Graph-based Clustering

Graph-based algorithms analyze dataset clusters, by representing the dataset as a particular data structure *G*, a *graph*, consisting in a set *V* of *vertices* (or *nodes*), and a set *E* of *edges*, connecting pairs of vertices (Fig. 5.8). The graph construction is usually based on the distance matrix (Eq. 2.6) calculated on the dataset. It is usual to map a dataset object *x* with a vertex *v*, and distance or similarity relation (satisfying several conditions) between objects with edges. The cluster analysis will come from studying the graph structure (as we have just seen for *CHAMELEON* algorithm in Section 5.3.2). In this section, we will define what is graph clustering, and we will present some of the most common techniques to perform it.

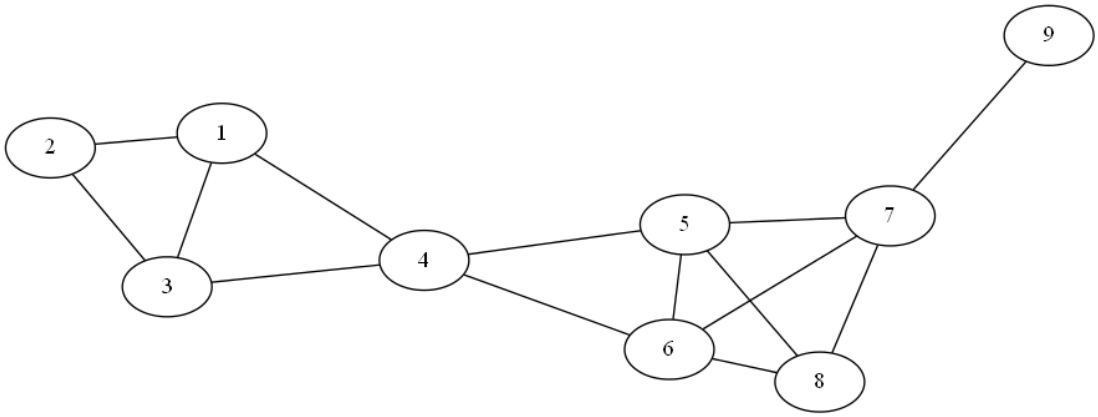


Fig. 5.8. An example of a graph *G* with 9 vertices. Redrawn from Tang and Liu (2010).

Graph clustering is the task of grouping the vertices of the graph into clusters, taking into consideration the edge structure of the graph, in such a way that there should be many edges *within* each cluster and relatively few *between* the clusters (Schaeffer, 2007). In literature, the graph clustering problem is also known as graph *community detection* problem (Tang and Liu, 2010; Fortunato, 2010). Approximately, graph clustering techniques are classified into four categories (Tang and Liu, 2010), depending on the community definitions:

1. Node-Centric Community;
2. Group-Centric Community;
3. Network-Centric Community;
4. Hierarchy-Centric Community;

Node-Centric Community. Vertices belonging to a cluster (or community) satisfy certain properties. It is usual to consider cluster a subset of vertices of the graph, where every two vertices in the cluster are connected by an edge (or forming a *clique*). Clique definition for clusters is very strict, and its detection is an *NP-complete* problem (Schaeffer, 2007). There are several methods belonging to this class, as the *Clique Percolation Method* (*CPM*; Palla et al., 2005), that relaxes this definition, allowing also overlapping between clusters. Given in input a parameter *k*, after discovering all cliques of size *k* in the graph *G*, the *CPM* target is to create a graph of cliques *G'*. In *G'* each found clique is a node, and edges are drawn for each

pair of cliques sharing k-1 vertices. The *connected components* (Cormen et al., 1990) of the resulting graph are considered clusters.

Group-Centric Community. The definition of cluster is relative to density, or the number of edges connecting cluster vertices. In this category, a sub-graph $G'(V', E')$ is considered to be a cluster if it is a *quasi-clique*, or γ*-dense*, that is:

$$\frac{|E'|}{|V'|(|V'|-1)/2} \geq \gamma \tag{5.21}$$

Consider that a clique of size k, have a total number of $k(k\text{-}1)/2$ edges connecting all its vertices. Thus, if the sub-graph G' is a clique, the left side of (Eq. 5.21) is equal to 1.

Network-Centric Community. Network-centric criterion needs to consider the connections within a network in a global sense (Tang and Liu, 2010). There are several approaches in this category, such as the clustering based on vertices similarity, usually defined from the *Jaccard's coefficient* (Eq. 5.10) or the *cosine similarity* (Eq. 2.11). Among them we will focus on *spectral clustering*, one of the most interesting technique for graph clustering (von Luxburg, 2006).

As we mentioned before, a graph G is usually represented by a $N \times N$ distance matrix (Eq. 2.6), where N is the number of vertices in G; we refer to this structure as A. Spectral clustering makes strong use of matrices, because global information about graph structure can be provided by the *spectrum* of the matrix. The matrix spectral analysis refers to the computation of the matrix *eigenvectors* (Friedberg et al., 1989), ordered by the magnitude (strength) of their corresponding *eigenvalues*.

All spectral clustering algorithms consist in four basic stages: (i) the construction of a matrix representation, the *utility matrix*, depending on the objective function; (ii) the computation of eigenvalues and eigenvectors of the utility matrix; (iii) the mapping of each vertex to a lower-dimensional representation, based on more or less important eigenvectors (again depending on the objective function); (iv) clustering

by applying algorithms as k-Means (Section 3.1.1) to cluster on low dimensional space.

When partitioning graphs, there are several models or objective functions to follow (Tang and Liu, 2010):

1. *Latent space model*. The objective function is to map vertices on a low dimensional space, where distances between vertices are preserved, respect to the original space. This can be done by using, for instance, the *MDS* (Multi-Dimensional Scaling) (Borg and Groenen, 2005). Given a distance matrix A for G, let be S a $N \times l$ matrix representing the dataset of cardinality N on the low dimensional space, with l attributes. It can be shown that (Borg and Groenen, 2005):

$$SS^T \approx -\frac{1}{2}(I - \frac{1}{N}\|1\|\|1\|^T)(A \circ A)(I - \frac{1}{N}\|1\|\|1\|^T) = P \qquad (5.22)$$

   where I is a $N \times N$ identity matrix (or the matrix containing all zero elements, except on the diagonal, where elements are set to 1), $||1||$ is a N-dimensional column vector, and ° indicates the element-wise matrix multiplication. The objective function is to minimize the difference $SS^T - P$. The *MDS* problem corresponds to find the top eigenvector V of P, since the optimal S corresponds to $V\Lambda^{1/2}$ (Tang and Liu, 2010), where Λ are the largest eigenvalues corresponding to V. The utility matrix to use for spectral clustering is P.

2. Minimum cut problem using the *edge cut* (*or minimum cut*). We remember that $cut(C_i, C_j)$ is the minimum weight of edges in a cluster C, whose elimination makes two disconnected partitions C_i and C_j (Section 5.3.1). If all edges have the same weight, it is the minimum number of edges disconnecting two partitions C_i and C_j. The edge cut often returns an unbalanced partition (Fig. 5.9). There are other different ways to estimate a cut. The *ratio* cut, and the *normalized* cut for a particular partitioning π are:

$$ratio\_cut(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, C_j)}{|C_i|} \qquad (5.23)$$

$$normalized\_cut(\pi) = \frac{1}{k}\sum_{i=1}^{k}\frac{cut(C_i, C_j)}{vol(C_i)} \tag{5.24}$$

where $vol(C_i)$ indicates the number of edges in C_i (or degree of C_i). Both ratio cut and normalized cut prefer balanced partitions (Fig. 5.9).

In this case spectral clustering can take advantage of two main types of utility matrix, corresponding to the name of *Laplacian* matrices. Before introducing them, we will define the *degree* matrix D, where each cell (i, i) indicates the degree of the ith vertex in G. The degree of a vertex v is denoted as the number of edges incident to it. D is a diagonal matrix, since non-zero elements are only on the trace. Both D and A are $N \times N$ matrices.

The first type of Laplacian matrix is the *unnormalized Laplacian matrix*, defined as:

$$L = D - A \tag{5.25}$$

usually used for the ratio cut approach.

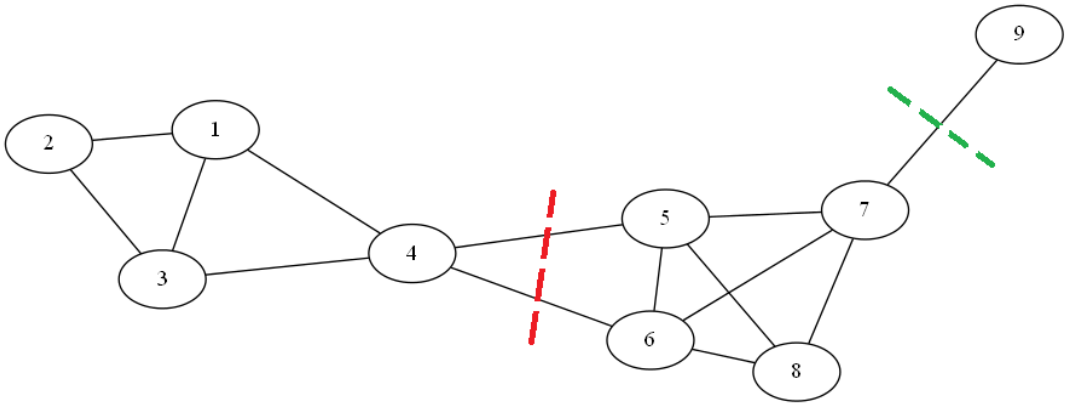


Fig. 5.9. An example of different partitions on a graph G of 9 vertices, where edges have the same weight. Redrawn from Tang and Liu (2010). The green dashed line represents the minimum cut (or edge cut), that produces an unbalanced partitioning π_1. The red dashed line represents another cut producing a balanced partitioning π_2. If we compute the normalized and the ratio cut for both partitions, we can see that both measures prefer the balanced partition π_2: $ratio\_cut(\pi_1)$ = 1/2 (1/1 + 1/8) = 9/16 = 0.56 > $ratio\_cut(\pi_2)$ = 1/2 (2/4 + 4/5) = 9/20 = 0.45; $normalized\_cut(\pi_1)$ = 1/2 (1/1 + 1/27) = 14/27 = 0.52 > $normalized\_cut(\pi_2)$ = 1/2 (2/12 + 2/16) = 7/48 = 0.15.

There are two versions of the second type, for the normalized cut approach, introduced by Shi and Malik (2000) and Ng et al. (2002), called *normalized Laplacian matrix*, and defined as:

$$L_{sym} = I - (D^{-1/2} A D^{-1/2}) \tag{5.26}$$

$$L_{rw} = I - (D^{-1} A) \tag{5.27}$$

The objective function corresponds to find the top eigenvectors of L, with the smallest eigenvalues.

3. *Modularity maximization*. Modularity measures the strength of a community partition by considering the degree distribution. The modularity for a cluster C is the fraction of the edges that fall within C, minus the expected ($d_i d_j / 2N$):

$$M(C) = \sum_{i \in C, j \in C} \left(A_{ij} - \frac{1}{N} \frac{d_i d_j}{2} \right) \tag{5.28}$$

where d_i and d_j are respectively the degrees of vertices i and j in C, and m is the total edges in the graph of size N. If the graph is partitioned into k clusters, the modularity for the entire graph will be:

$$M(G) = \sum_{l=1}^{k} \sum_{i \in C_l, j \in C_l} \left(A_{ij} - \frac{1}{N} \frac{d_i d_j}{2} \right) \tag{5.29}$$

For larger value of $M(G)$, the graph G presents a good community structure. Then, for spectral clustering, the utility matrix is the following *modularity* matrix B:

$$B = A - (DD^T / 2m) \tag{5.30}$$

and the objective function corresponds to find the top eigenvectors of B, with largest eigenvalues.

Hierarchy-Centric Community. The approach is similar to that introduced in Section 5.3.2. There are both divisive and agglomerative methods. For agglomerative methods, in the initial phase, each vertex is a cluster, and they are merged iteratively into large communities, following a certain criterion among those, proposed above, for graph clustering. For divisive methods, the entire graph is initially a community. A common technique, to iteratively build lower level communities, is to recursively remove the "weakest" edge, or the edge with highest *betweenness.* The betweenness is the number of shortest paths connecting any pair of vertices that pass through the edge (Schaeffer, 2007): edges with higher betweenness tend to be "bridges" within communities (Tang and Liu, 2010).

5.3.5 Grid-based Clustering

In general, a grid-based clustering algorithm consists of the following five basic steps (Gan et al., 2007): (i) partitioning the data space into a finite number of cells (or creating grid structure); (ii) estimating the cell density for each cell; (iii) sorting the cells according to their densities; (iv) identifying cluster centers; and (v) traversal of neighbor cells. The main feature of grid-based clustering is that it can take advantage of parallel processing, to significantly reduce the computational complexity. Among them we will discuss the most known *STING* (*STatistical INformation Grid-based method*) (Wang et al., 1997) and *Wavecluster* (Sheikholeslami et al., 1998) algorithms. Grid-based methods are fast and handle outliers well. Grid-based methodology is also used as an intermediate step in many other algorithms as *CLIQUE* (see Section 5.3.6).

STING. The algorithm *STING* is designed to deal with numerical attributes (spatial data) and to allow region oriented queries (Berkhin, 2002). *STING* builds a hierarchical tree, where each node is a grid cell having four children (default), each of which is a grid cell, recursively (Fig. 5.10). In a bottom-up way, from leaf nodes to root, it collects the following statistical information for each grid cell: number of contained objects, mean, standard deviation, minimum, maximum and distribution type. Higher level cells statistical information can be easily obtained from the lower level cells.

The type of distribution is measured by using a *chi-squared* (χ^2) test (Greenwood and Nikulin, 1996). If the aggregate distribution of an higher level cell does not correspond with the children cells' distribution, it is set to none.

STING offers several advantages: (i) the grid structure allows statistical information for each cell, and these are calculated once through the database in $O(N)$ time, in the initial phase; (ii) after generating the hierarchical structure, the query processing time is $O(g)$, where g is the total number of grid cells at the lowest level, which is usually much smaller than N. However, the quality of clustering is high sensitive to the grid resolution. When resolution gets lower, although it allows fast processing time, the clustering result may appear "isothetic" (Han and Kamber, 2000). During the clustering process, it considers spatial relationships between neighboring children, only on horizontal or vertical boundary, by excluding diagonal. Then, the lower the resolution, the lower the quality.

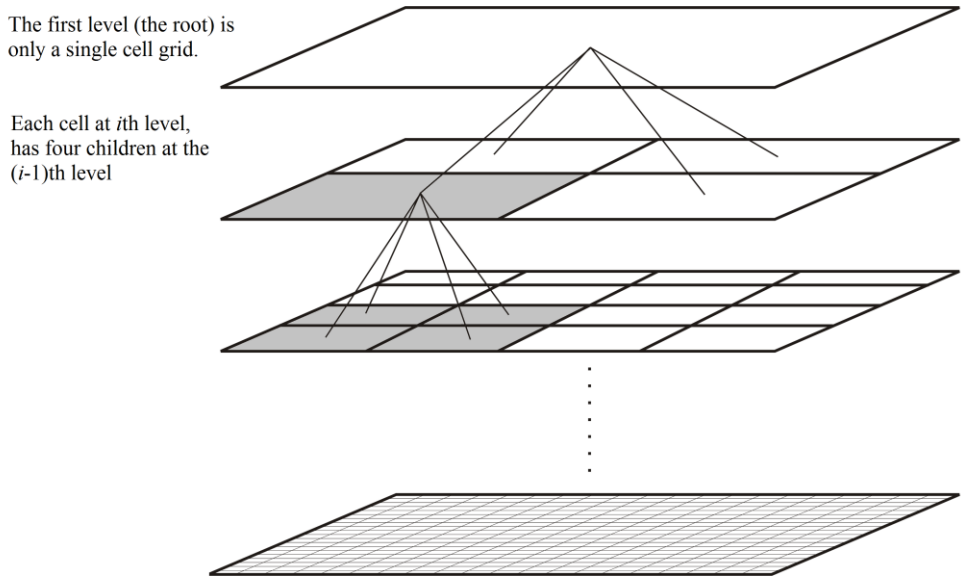


Fig. 5.10. Hierarchical grid structure of *STING*. Redrawn from Wang et al. (1997).

Wavecluster. This algorithm also works with numerical attributes, and supports an advanced multi-resolution, since it bases clustering analysis on the wavelet transform, a very common technique in signal processing (Chui, 1992; Graps, 1995). Wavelet transform has also been used to

compress data (Chan and Fu, 1999). To understand how to take advantage of this technique, it is necessary to discuss the relationship between spatial data and multidimensional signals (Sheikholeslami et al., 1998).

Multidimensional spatial data objects can be represented in an *m*-dimensional space, called also *feature space*. Numerical attributes of a spatial object are represented by a vector (see Chapter 2), where each element corresponds to a numerical attribute, or feature. Thus, an object with *m* numerical attributes represents a point in the *m*-dimensional feature space. Data clustering means to identify sparse and dense regions on this feature space.

Now, look at the feature space from a signal processing perspective (Sheikholeslami et al., 1998). Objects on the feature space form a *m*-dimensional signal. The high frequency sections of the signal correspond to the cluster boundaries on the feature space, while low frequency sections correspond to the regions of the feature spaces where objects are concentrated (cluster centers).

Wavelet transform is a signal processing technique that decomposes a signal into different frequency sub-bands (for example, high frequency and low frequency sub-bands). The key idea in this approach is to apply wavelet transform on the feature space to find low frequency parts that correspond to clusters.

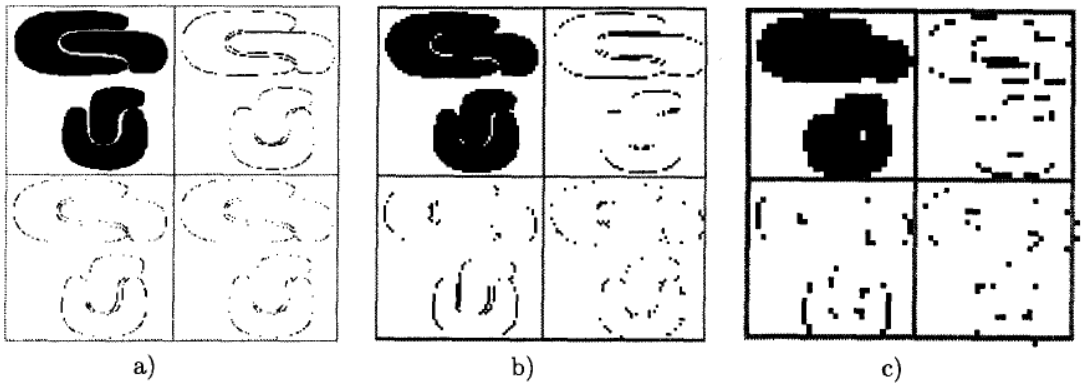


Fig. 5.11. Multi-resolution of a feature space. a) High resolution; b) medium resolution; c) low resolution. From Sheikholeslami et al. (1998).

Main features: (i) unsupervised clustering, since it does not need any input parameter; (ii) effective outliers removing, by using low pass filters; (iii) hierarchical clustering, because it finds clusters at different levels of

resolution (Fig. 5.11), by the multi-resolution decomposition property of wavelet transform (Chui, 1992); (iv) fast computation time, since wavelet transform computation needs $O(n)$ time, and can also be parallelized.

5.3.6 Other techniques

In this section we will give a concise introduction to the other approaches for clustering. We will start from model-based clustering, that defines cluster objects as samples coming from a particular statistical distribution. Then, we will present some suitable approaches for dealing with high-dimensional data: subspace clustering and neural networks.

Model-based Clustering. Model-based clustering methods try to find a statistic model for data. They assume that data are generated by a mixture of underlying probability distributions (Fraley and Raftery, 2002). Each cluster is assumed to come from each distribution. The clustering problem, in this case, becomes the estimation of the parameters of the assumed mixture model.

Let p_j (x_i | Θ_j) be the probability of finding an object x_i of dataset D, in the jth distribution, where Θ_j are the parameters of the jth distribution; and let k be the number of distributions (or clusters) in the mixture. The likelihood λ of the observed objects in D is:

$$\lambda(\theta_1, \theta_2, \ldots, \theta_k; \gamma_1, \gamma_2, \ldots, \gamma_k \mid D) = \prod_{i=1}^{N} \sum_{j=1}^{k} \gamma_j p(x_i \mid \theta_j) \tag{5.31}$$

where γ_j is the probability that an object belongs to the jth distribution.

In the mixture likelihood approach, the goal is to estimate the parameters Θ_j (for $0 < j \leq k$) that maximize the above value. If we assume that data come from a mixture of k Gaussian distribution, we aim to find mean and standard deviation of each distribution that maximize the likelihood λ (Fig. 5.12).

The *EM* (*Expectation-Maximization*; Dempster et al., 1977) algorithm is the most widely used method for estimating the parameters of a finite mixture probability density (Gan et al., 2007). Partitional algorithms, such as *k-means* (Section 5.3.1), are usually included in this class of clustering

algorithms, since the center refinement process, can be implemented using the *EM* algorithm. Among model-based algorithms we refer to *MCLUST* (Fraley and Raftery, 1999).

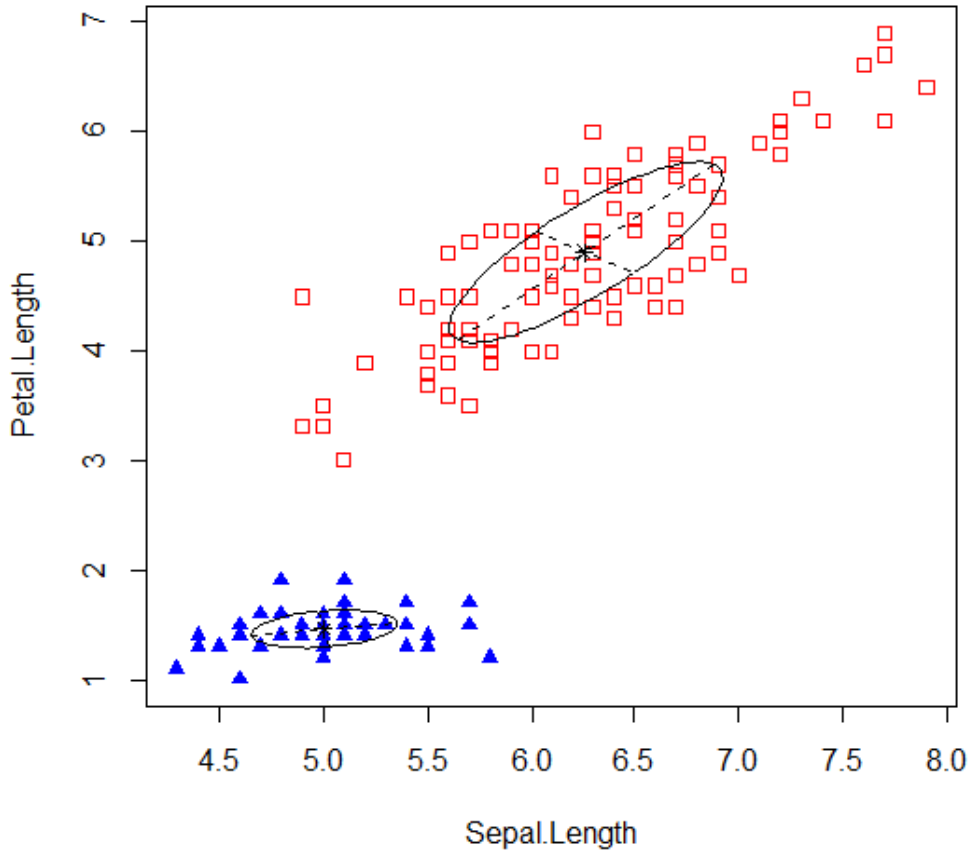


Fig. 5.12. Model based clustering, using *MCLUST* (Fraley and Raftery, 1999), on *Fisher's Iris* dataset (Fisher, 1936), by selection of sepal length and petal length attributes. In this case the selected model is Gaussian. Each cluster is overlaid by the relative shape of Gaussian distribution function.

Subspace Clustering. Subspace clustering deals with clustering of high dimensional data. A common way to overcome problems of high dimensional data spaces, is to map them into lower dimensional spaces, by selection of a small number of feature (*feature selection*), or extracting a set of new feature (*feature extraction*), where data variance is maximized, similarly to the *Principal Component Analysis* (*PCA*; Smith, 2002; also described in Appendix C). Unfortunately, these methods bring to a partial evaluation for clustering, because take in consideration only clusters extractable from a single combination of features. The idea behind this

approach is that different clusters can be discovered selecting different subsets of attributes (or *subspaces*). A naïve solution, to solve this limit, is to test all possible subspaces to find clusters. However, in real applications this search can be expensive, since all potential subspaces for a dataset with m attributes (features) are $2^m - 1$. The literature offers several methods to find clusters on subspaces. Here we sketch two of the most known algorithms: *CLIQUE* (Agrawal et al., 1998) and *PROCLUS* (Aggarwal et al., 1999).

CLIQUE was the first algorithm to treat the subspace clustering. It uses a grid-based approach, because divides the space into equi-sized cells of width ξ. Only cells containing at least τ objects are considered dense. A cluster is then defined as a maximal set of adjacent dense cells. The algorithm works in a bottom-up way. Starting with all 1-dimensional dense cells, (k+1)-dimensional dense cells are computed from the set of k-dimensional dense cells in an *APRIORI*-like style (Han and Kamber, 2000). For the downward closure property, if a (k+1)-dimensional cell contains a projection onto a k-dimensional cell that is not dense, then the (k+1)-dimensional cell can also not be dense (Zimek, 2008). There are some variants of *CLIQUE* such as *ENCLUS* (Cheng et al., 1999) and *MAFIA* (Nagesh et al., 2001).

PROCLUS (*PROjected CLUStering*) try to find the subsets of attributes (or projection) where the a considered set of points cluster best. It finds projected clusters by locating the cluster centers and finding the appropriate set of dimensions in which each cluster exists. The problem of finding cluster centers has been introduced by *k-medoids* method (see Section 5.3.1). The general approach is to find the best set of medoids by a hill climbing process similar to the one used in *CLARANS*, but generalized to deal with projected clustering.

There are many other algorithms in literature (e.g. Parsons et al., 2004), each proposing different approaches. An exhaustive review of density-based clustering applied to the subspace clustering can be found in Zimek (2008).

Neural Network Clustering. Another common approach, to deal with high-dimensional data, is the use of *artificial neural networks*. A neural network consists of a set of input/output *units* (or *neurons*), and a set of weighted

connections among them. The main properties that make popular neural networks for clustering are: (i) they can be implemented easily on parallel and distributed processing architectures, and (ii) they learn by adjusting their interconnection weights so as to best fit the data (Han and Kamber, 2000). One of the most known neural networks for clustering analysis are *SOM* (*Self Organizing Maps*) or *Kohonen's maps* (Kohonen, 2001).

A *SOM* consists basically of two layers. The input layer consists of N elementary computational units or neurons corresponding to vector objects of the input dataset D of size N. These units are connected to a second layer of output neurons U that form a map. A reference weight vector v, also called *prototype vector*, is associated with each output neuron in the map. *SOM* then maps high-dimensional input data vectors onto two-dimensional grid of prototype vectors that are easier to visualize and explore than the original data.

By means of lateral connections, the neurons in U form a lattice structure of dimensionality N' (Figs. 5.13-14), which is typically much smaller than N (Utlsch, 2000). The fundamentals of *SOM* are the competition between nodes in the output layer U.

Self-organization refers to the ability of a biological or technical system to adapt its internal organization to structures sensed in the input of the system. The neural network approach, at each time t, consists of two modalities: (i) a *training* step, and (ii) an *updating* step. Before starting, output vectors weights are set with random values.

In the training step, an input vector x is compared with each prototype vector on the map, using a distance measure, to find the most similar: the *Best Matching Unit* (*BMU*). In the updating step, the prototype vector of each output neuron i is updated following the rule:

$$v_i(t+1) = v_i(t) + \alpha(t) h_{bi}(t)[x - v_i(t)] \tag{5.32}$$

where $\alpha(t)$ is the learning rate, h_{bi} is a neighbor function between neuron i and the *BMU*. With this function, the closer a node is to the *BMU*, the more its weights become more like the input vector. The farther away the neighbor is from the *BMU*, the less it learns. Both the learning rate and the neighbor function decrease monotonically over time. In particular, assuming h_{bi} to be Gaussian, it can be expressed as:

$$h_{bi}(t) = e^{-\|r_b - r_i\|^2 / 2\sigma^2(t)} \tag{5.33}$$

where r_b and r_i are respectively the position of *BMU* and neurons i, while $\sigma(t)$ is the standard deviation of the Gaussian.

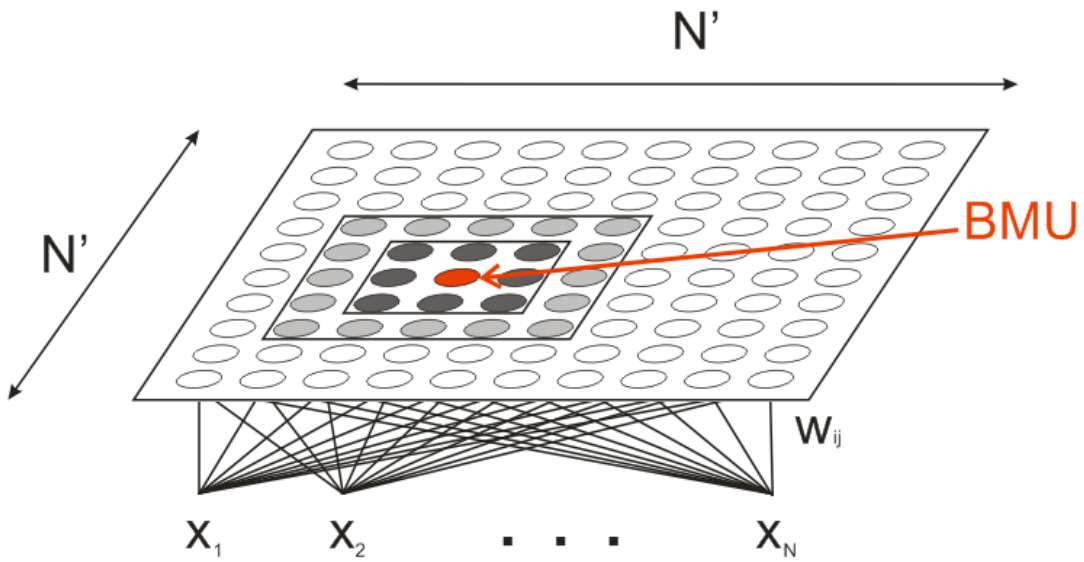


Fig. 5.13. *SOM* lattice structure. The red point indicates the *BMU*. Gray points are relative to the neighborhood of the *BMU*. Redrawn from Di Salvo et al. (2012).

Fig. 5.14. Different types of SOM grid structure, with different neighboring relation. Hexagonal on the left, rectangular on the right. Redrawn from Di Salvo et al. (2012).

To make an example of clustering, consider to pass, in the training step, objects from a dataset, once iteration. If the dataset presents several similar objects forming clusters (with similar attributes, or weights), they will be mapped in a particular region of the *SOM*, because they represent a repeated input for the map. Thus, a group of neighboring output vectors

will be influenced, and their weights will be accosted to input cluster objects. For this reason, *SOM* are also most used for visualization of dataset structures for high-dimensional data. The low-dimensional map obtained by *SOM* algorithm provides a 2D projection of the high dimensional data that can be visually inspected. A common way to visualize the presence of clusters after SOM learning process is the so called unified distance matrix (*U-matrix*). In order to calculate the *U-Matrix* the averaged distances between each neuron and its neighbors are computed. This method provides a color matrix representing distances between neighboring map units, and thus shows the cluster structure of the map: high values of *U-matrix* indicate a cluster border while uniform areas of low values indicate clusters themselves (Ultsch, 1993). An example of 3D features space and the *U-Matrix* obtained after a learning process of SOM is shown in Fig. 5.15.

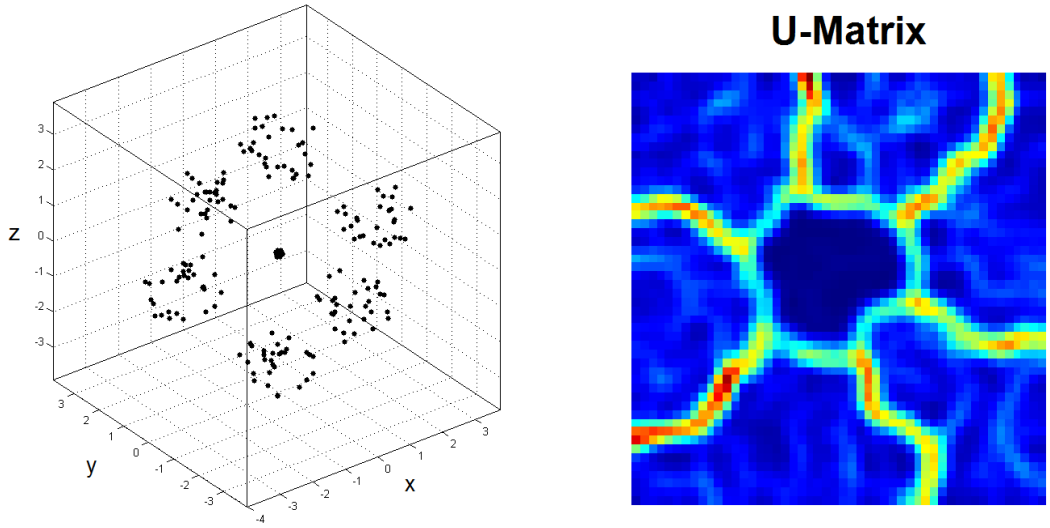


Fig. 5.15. a) 3D feature space with 7 clusters. b) A rectangular *U-Matrix* after training step: the blue regions are related to clusters, or regions where distance between neighboring neurons is small.

5.3.7 Evaluating clustering

Most algorithms base the validity of the results, by showing experiments on 2-dimensional or max 3-dimensional datasets. It is clear that they have more problems when try to visualize high-dimensional data: in this case, they need some adequate visualization techniques such as *SOM* (Section

5.3.6) or *Parallel coordinates* (Fig. 5.16; Inselberg and Dimsdale, 1990). However, assessment by visualization relates to the validator, and a numeric universal measure is needed to test the quality of clustering.

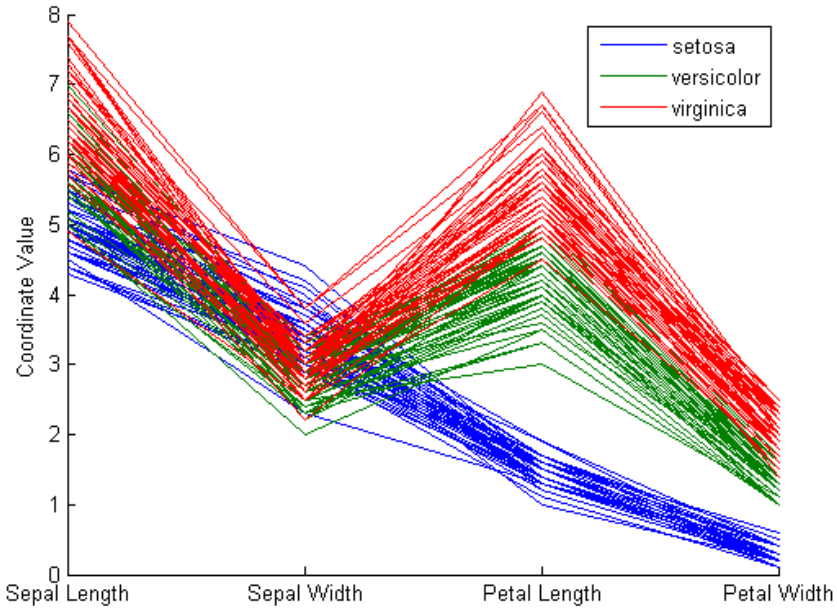


Fig. 5.16. Parallel coordinates visualization of *Fisher's Iris data* (Fisher, 1936) using MATLAB (http://www.mathworks.it/).

There are two general reasonable criterions for evaluation and selection of an optimal clustering scheme: (i) *compactness*, for which distances among members of each cluster should be minimized; (ii) *separation*, for which distances among clusters should be maximized. In literature three main types of clustering assessment are cited (Fig. 5.17; Gan et al., 2007). In particular, an *external* assessment of validity compares the recovered clustering structure *C*, with an *a priori* structure *P* and attempts to quantify match between the two.

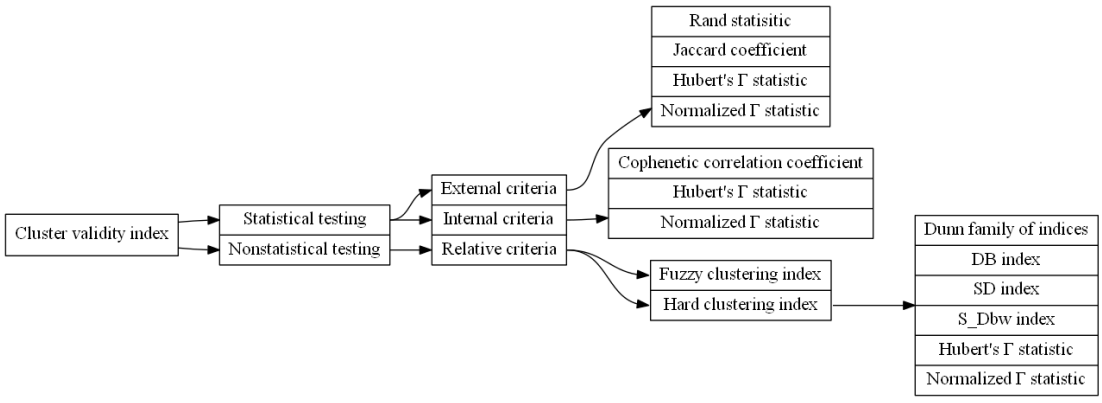


Fig. 5.17. A diagram of validity indices. Redrawn from Gan et al. (2007).

An essential task, when using this approach, is to test whether the dataset is randomly structured or not (or *Null Hypothesis*; Halkidi et al. 2001). If the dataset has a low cluster tendency, there is or no benefit in performing cluster analysis. An *internal* examination of validity tries to determine if the clustering structure is intrinsically appropriate for the data. This assessment considers whether a given cluster is unusually compact or isolated compared to other clusters of the same size in random data (Aldridge, 2006). These first two approaches involve statistical testing, which is computationally expensive (Gan et al., 2007). The third approach, the *relative* test, compares two structures and measures their relative merit (Jain et al., 1999). It does not involve statistical testing, but aims to find the best clustering scheme based on certain assumptions and parameters. Indices used for this comparison are discussed in detail in Jain and Dubes (1988) and Dubes (1993).

One common validation measure, using relative criteria, is the *Davies-Bouldin* (*DB*) index (Davies and Bouldin, 1979). Such an index is function of the number of clusters, the inter-cluster and within-cluster distances. Formally it is defined as follows:

$$DB = \frac{1}{N}\sum_{i=1}^{N} \max_{i \neq j} \left\{ \frac{S_n(C_i) + S_n(C_j)}{d(C_i, C_j)} \right\} \tag{5.34}$$

where $S_n(C)$ is the average distance of all objects in C to their cluster center, $D(Q_i, Q_j)$ is the distance between centers of clusters C_i and C_j, respectively.

Small values of *DB* correspond to compact clusters whose centers are far away from each other. In the light of it, the number of clusters that minimizes *DB* is taken as the optimal number of clusters. A possible approach may be the use of validity algorithms such as *Davies-Bouldin* (*DB*) index to validate *k-means* clustering results, explained in Section 5.3.1: the number of clusters that minimizes *DB* is taken as the optimal number.

An example of 3-class *k-means* together with *DB* index is shown in Figure 5.18. In particular, Figure 5.18a shows the best 3-clustering structure of the data set, while Figure 5.18b shows the value of *DB* index for increasing

value of *k*. The best cluster number is chosen on the basis of minimum value of DB index.

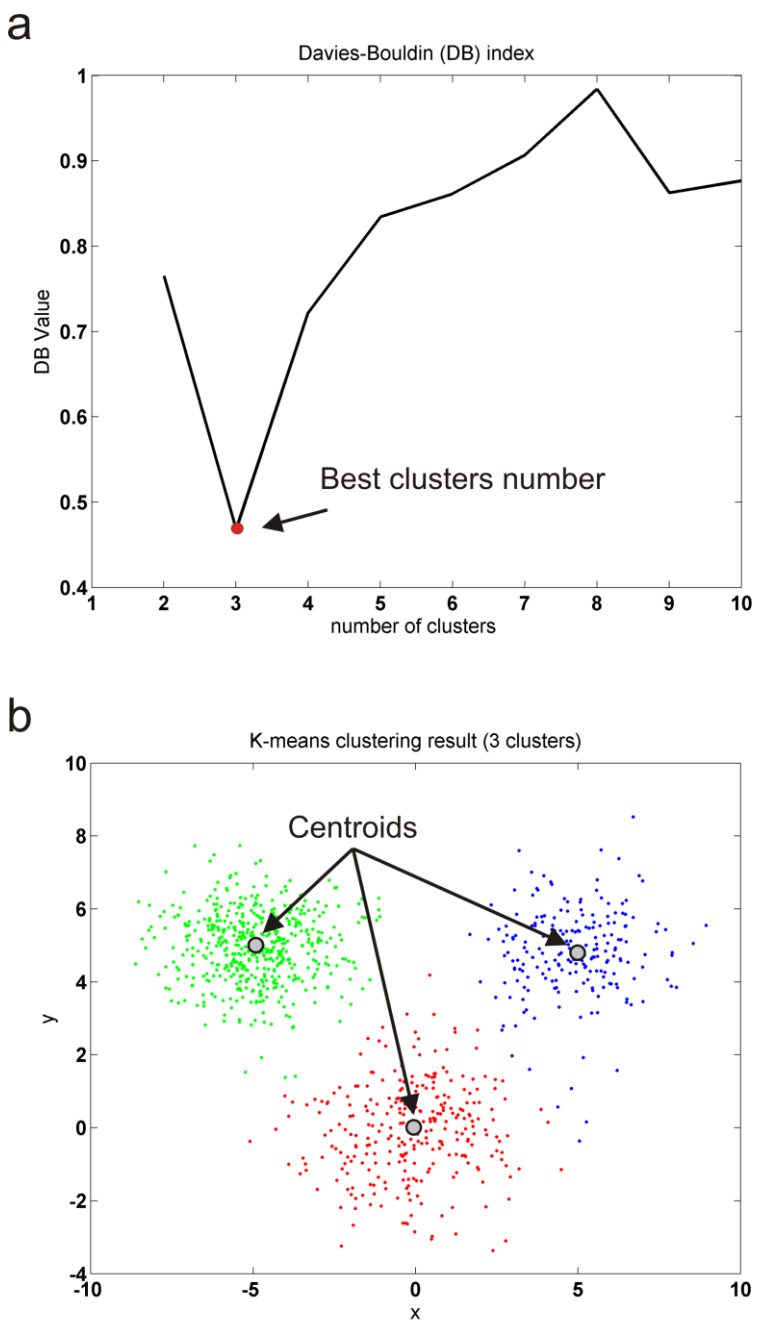


Fig. 5.18. a) Best clustering structure computed using *Davies-Bouldin* index for a feature space with 3 clusters. b) Clustering result using *k-means* with $k = 3$.

5.4 Outlier detection

All density-based clustering algorithms (Section 5.3.3) naturally deal with outliers by avoiding inserting them into clusters. They commonly use the distance to the *k*-th nearest neighbor to detect them. However, they are able to capture only certain types of noise when clusters of different densities are present. In fact, an object close to a tight cluster may be more likely to be an outlier than an object that is further away from a weaker cluster. Few approaches are directly concerned with outlier detection. These algorithms, in general, consider outliers from a more global perspective, assuming objects belonging to a known statistical distribution.

Recently, attention has been placed on local outlier detection (Fig. 5.19) (Breunig et al., 2000; Papadimitriou et al., 2003; Jin et al., 2006), in which outliers are locally compared with neighbors (i) by taking into account their density distribution or (ii) by measuring the similarity using symmetric relations (neighbors and reverse neighbors).

Breunig et al. (2000) introduce the *Local Outlier Factor* (*LOF*) to rank objects with respect to their outlierness. It uses a definition for the reachability-distance for an object p similar to that introduced in Section 5.3.3 for *OPTICS*:

$$rd_k(p,o) = \max[k_{dist}(o), dist(o,p)] \tag{5.35}$$

and defines *local reachability density* of an object p as:

$$lrd_k(p) = 1/\left(\frac{\sum_{o \in N_k(p)} rd_k(p,o)}{|N_k(p)|} \right) \tag{5.36}$$

where $N_k(p)$ represents the set of k-nearest neighbors of an object p. The local outlier factor of object p captures the degree to which we can call p an outlier. It is the average of the ratio of the local reachability density of p and those of p's k-nearest neighbors (Breunig et al., 2000):

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|} \tag{5.37}$$

Although LOF does not suffer from the local density problem, selecting a suitable k, for k-nearest neighbors search, is non-trivial. *LOCI* (Papadimitriou et al., 2003) overcomes this shortcoming by using statistical values derived from the data. Jin et al. (2006) propose an algorithm (*INFLO*) to efficiently discover top-n outliers using clusters, for a given value of k. Density distribution is estimated by considering both neighbors and reverse neighbors. This results in meaningful outliers detection (Fig. 5.20).

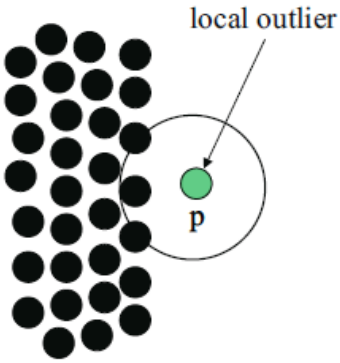


Fig. 5.19. p is a local outlier (from Jin et al., 2006).

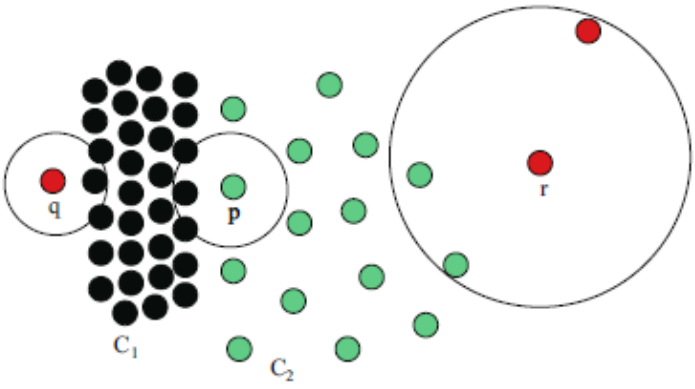


Fig. 5.20. Basing on local neighborhood, p is not so much outlier as in Fig. 5.19. Same consideration can be done for r. q is the most probable outlier (from Jin et al., 2006).

To define *INFLO*, let $k_{dist}(p)$ be the distance from p to its k-th nearest object. The inverse of $k_{dist}(p)$ is called k-local density. The reverse k-nearest

neighborhood of p is the set of all objects having p among their k nearest neighbors. The k-influence space of p is the collection of those objects among the k closest to p which also belong to its reverse k-nearest neighborhood. Then, $INFLO_k$ of p is defined as the average of k-local densities of objects belonging to the influence space of p. We formally explain such definitions.

Definition 5.4.1. The k-local density of p, denoted as $den_k(p)$, is the inverse of the $k_{dist}(p)$:

$$den_k(p) = \frac{1}{k_{dist}(p)} \tag{5.38}$$

Definition 5.4.2. The reverse k-nearest neighborhood of an object p is defined as:

$$RN_k(p) = \{q \in D \mid p \in N_k(q)\} \tag{5.39}$$

Definition 5.4.3. Following Jin et al. (2006), the density distribution around an object p can be estimated through the k-influence space defined as:

$$IS_k(p) = N_k(p) \cap RN_k(p) \tag{5.40}$$

The k-nearest-neighbors set $Nk(p)$ is always not empty, whereas the size of $RN_k(p)$ depends on the number of times p is classified as k nearest neighbor of an object.

Definition 5.4.4. The k-influenced outlierness of p is defined as:

$$INFLO_k(p) = \frac{\sum_{o \in IS_k(p)} den_k(p)}{|IS_k(p)| den_k(p)} \tag{5.41}$$

The higher $INFLO_k$ of p is, the more likely such an object is a local outlier.

5.5 Enhancing Density-Based Clustering

The basic structure of density-based clustering presents some common drawbacks: (i) parameters have to be set; (ii) the behavior of the algorithm is sensitive to the density of the starting object; and (iii) adjacent clusters of different densities could not be properly identified. Although these problems have been subject of intensive investigation (Ram et al., 2009) a satisfactory solution has not been found, yet.

In (Cassisi et al., 2012b), we propose a simple but effective method able to overcome the above shortcomings and boost the *DBSCAN* performances. We introduce the concept of space stratification which ranks the objects in the space according to density criteria. Usually, a reasonable density estimator relies on *k*-nearest neighbors (knn) distances. Objects with small knn distance values belong to dense regions of the space, while large values refer to sparse regions or outliers. The limit of a basic method is that it can detect only global outliers (i.e., objects lying far away from the rest of data). Those objects have large knn distance values. Whereas, local outliers are objects located near to dense regions, and therefore may have small knn distance values. Alternatively, the degree of outlierness can be used as sorting criterion to rank the objects in the space. As an example, the *INFLO* function (see Section 5.4) efficiently identifies local outliers in a generic space by using the concept of local outlier factor. However, a stratification based only on this method does not highlight different density areas (Figure 5.21(a)).

In our work, we use the stratification based on both *INFLO* function and knn distances. More precisely, the choice of *INFLO* is motivated by the following reasons. *INFLO* defines a new neighborhood relationship, called *Influence Space* (*IS*), allowing a better estimation of the neighborhoods density distribution (Hinneburg and Keim, 1998). Since *IS* uses both the nearest neighbors (N_k) and reverse nearest neighbors (RN_k), *INFLO* outperforms other measures (e.g. *LOF*, Breunig et al., 2000) in detecting local outliers. Nevertheless, our ranking procedure (stratification) based on a linear combination of *INFLO* and knn distances shows more robustness in detecting outliers than *INFLO*. Moreover, clustering based on *IS* makes the cluster expansion phase (Daszykowski et al., 2002) highly sensitive to local density changes. Our main contribution consists of

replacing the classic neighborhood relationship introduced in (Ester et al., 1996), called *ε-neighbourhood* (see Section 5.3.3), with a novel approach for density-based clustering that takes advantage of the *Influence Space*. In addition, in order to enhance performance, we exploit stratification by projecting a generic *k*-dimensional space into a (*k*+1)-dimensional space, where the new dimension refers to the relative ranking value. Consequently, the method results effective in distinguishing slightly different density areas and in detecting local and global outliers. A comprehensive evaluation of the method indicates that it outperforms *DBSCAN* and *OPTICS* in all the standard benchmark datasets.

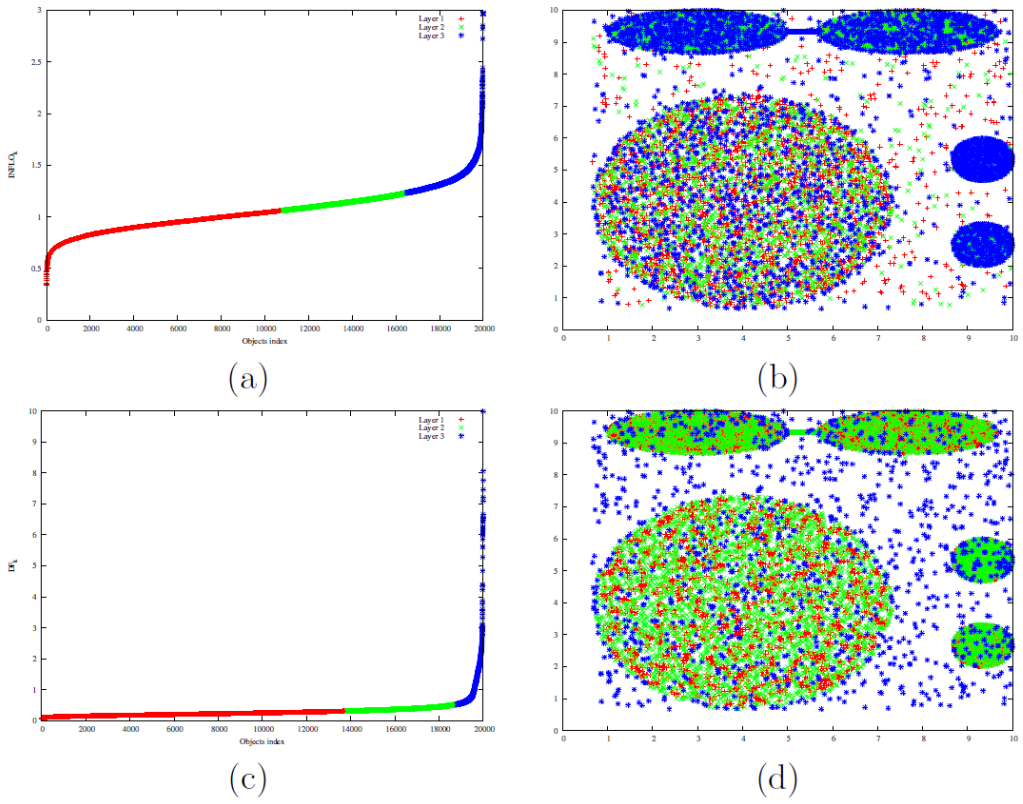


Fig. 5.21: Stratification based on both $INFLO_k$ and DF_k measures. Layers, representing different density regions of the dataset, are distinguished by different colors (red for first layer, green for the second, and so on). (a) Strata obtained using algorithm Stratify in Table 5.1 in connection with the $INFLO_k$ measure. (b) Stratification of the space related to (a). (c) Strata obtained using algorithm Stratify in connection with the DF_k measure. (d) Stratification of the space related to (c). From Cassisi et al. (2012b).

5.5.1. Stratification based outlier detection

Let (D, d) be a metric space, where D is the universe of data and d be the metric distance function. Let $S = \{x_i \mid x_i \in D, i = 1,2,...,n\}$ be a finite subset of U of size $n \geq 1$. Let $d(x, y)$ be the distance of object y from x, where $x, y \in S$. Let $w(x) = \sum_{y \in S} d(x, y)$ be the sum of distances of x from the remaining objects of S.

Stratification can be considered as a pre-processing step to analyze and discover main data properties. It divides data into layers, where objects belonging to the same layer have global similar characteristics. The stratification process consists of the following steps: (i) sorting the objects in S according to the function w; (ii) partition S into an ordered list of subsets on the basis of w. This ordered list of sets, P, is called stratification of S.

Definition 5.5.1. Given two subsets A and B of S, they satisfy the stratification
relation $\prec_w$ if and only if, $\forall x \in A$ and $\forall y \in B$, $w(x) \leq w(y)$.

Definition 5.5.2. Let $P = \{S_1, S_2, \ldots, S_h\}$ be a set of h disjoint subsets of S. We
call P a stratification of S if P is a linearly ordered set with respect to $\prec_w$.

By using the above definitions the space can be partitioned into subsets which are concentric (toroidal). The above definition can be generalized to the *knn-stratification*.

Let w_k be the function which maps each object x into the sum of the distances from x to the nearest k objects. Let be the set of k-nearest neighbors of x in S. For each object x in S, we compute the k-nearest neighbors and store the sum of their distances from x. Therefore:

$$w_k(x) = \sum_{y \in N_k(x)} d(x, y) \tag{5.42}$$

Definition 5.5.3. Let $P = \{S_1, S_2, \ldots, S_h\}$ be a set of h disjoint subsets of S. We call P a knn-stratification of S if P is a linearly ordered set with respect to $\prec_{wk}$.

The knn-stratification allows to partition the dataset with respect to the density of the k-nearest neighborhood. A partition of the stratification will contain objects that are similar with respect to the w_k function. The largest element S_h of P is a subset containing the outliers of S, indeed the objects of the dataset which are far from their k-nearest neighbors. Besides classical outlier analysis algorithms (Aggarwal and Yu, 2001), recent studies have focused on mining local outliers computing the density distribution of their neighbors, as we have just seen in Section 5.4. Although intuitive, when outliers are in the location where the density distributions in the neighborhood are significantly different, for example, in the case of objects from a sparse cluster close to a denser cluster, this may cause wrong estimations. Outlierness function such as INFLO are suitable for these purposes. In our method, in order to improve treatment of density variance, $INFLO_k$ is normalized with respect to the size of RN_k. We call the new measure $AINFLO_k$ (*Adjusted* $INFLO_k$). This function indicates how much an object is a local or a global outlier. A density measurement based on *AINFLOk* and *knn-stratification* improves *DBSCAN* clustering since distinguishes clusters having small density variance.

Definition 5.5.4. The *Adjusted* $INFLO_k$ of x is defined as:

$$AINFLO_k(x) = \frac{INFLO_k(x)}{|RN_k(x)|} \tag{5.43}$$

Definition 5.5.5. By linearly combining the knn-stratification and the $AINFLO_k$ measure, we obtain the following density function:

$$DF_k(x) = AINFLO_k(x) + w_k(x) \tag{5.44}$$

The DF_k function is a monotonic function with respect to the outlierness of an object. By sorting the objects in relation to the DF_k in an increasing order, we note that at some point the function becomes steeper. We

stratify the input dataset by using the density function DF_k in the algorithm *STRATIFY* of Table 5.1. The algorithm computes the average $AINFLO_k$ and uses such a measure to partition the dataset into layers of decreasing density. The last layer of the dataset will be the set of all candidate outliers.

In Figure 5.21, we compare the behavior of $INFLO_k$ and DF_k on the dataset *D1* obtained from Guha et al. 1998. The basic idea behind the DF_k function is to merge the knowledge carried by $INFLO_k$ and w_k. The shape of the $INFLO_k$ function is typically the one showed in Fig. 5.21(a). However, most of the objects having a high $INFLO_k$ are not outliers. By adding the function w_k to $INFLO_k$ we are able to reduce the false positive outliers. Furthermore, candidate outliers have always a very high DF_k values and they cause the exponential growth of the function.

```
Algorithm STRATIFY(D, k)
begin
n = |D|;
for i=1 to n do
     D[i].DF_k = DF(D[i].data,k);
     D[i].AINFLO_k = AINFLO(D[i].data,k);
end for
Sort D using DF_k field as sorting key;
cut = 1;
layer = 1;
δ = avg(D[1...n].AINFLO_k) + var(D[1...n].AINFLO_k);
while cut < n do
     cut = FindStrata(D, cut, layer, δ, n);
     layer = layer + 1;
end while
end
```

Table 5.1. The algorithm *STRATIFY*. *D* is the dataset of objects to be clustered. It is loaded into a data structure having as fields *data*, *layer*, $AINFLO_k$, DF_k. Given in input *k*, *DF* and *AINFLO* are subroutines calculating, respectively, the DF_k and the $AINFLO_k$ function value of *D*[*i*. δ is a particular threshold used in *FindStrata* (Table 5.2) to stop stratification. The objects in the last strata of *D* are candidate to be outliers. All the remaining objects are considered members of some cluster.

```
Algorithm FindStrata(D, start_idx, new_layer, δ, n)
begin
new_cut = start_idx; // cut point index
layer = new_layer;   // index of the new layer
μ_DF = avg(D[start_idx...n].DF_k);
while S[new_cut].DF_k < μ_DF do
     new_cut = new_cut + 1;
end while
μ_AINFLO = avg(D[start_idx...new_cut-1].AINFLO_k);
for i=start_idx to new_cut-1 do
     D[i].layer = layer;
end for
if μ_AINFLO < δ then
     return new_cut;
end if
noise = layer + 1; // noise is the last layer index
for i=new_cut to n do
     D[i].layer = noise;
end for
return n
end
```

Table 5.2. The *FindStrata* method extracts the strata from the current subset of *D*. Then it checks if the average $AINFLO_k$ of the computed strata is above δ; we use δ = avg(D[1…n].$AINFLO_k$) + var(D[1…n].$AINFLO_k$) (Table 5.1). The objects in the last layer of *D* are candidate to be outliers.

Through such an observation, the algorithm *STRATIFY*, in Table 5.1, can be adapted to identify only the outliers of the dataset. DF_k is a monotonic increasing function (see Fig. 5.21c), and the slope of lines can be computed. We name s_1 and s_2 the slope of lines approximating the first and the last layer values, respectively. The former characterizes data forming clusters, while the latter is relative to sparse regions. Then, starting from the first object in the last layer and following the ordering, we heuristically set as outliers all objects x_i such that:

$$\frac{DF_k(x_i) + DF_k(x_{i+1})}{2} > \frac{s_1 + s_2}{2} \tag{5.45}$$

In Figure 5.22, we show the results of such a method.

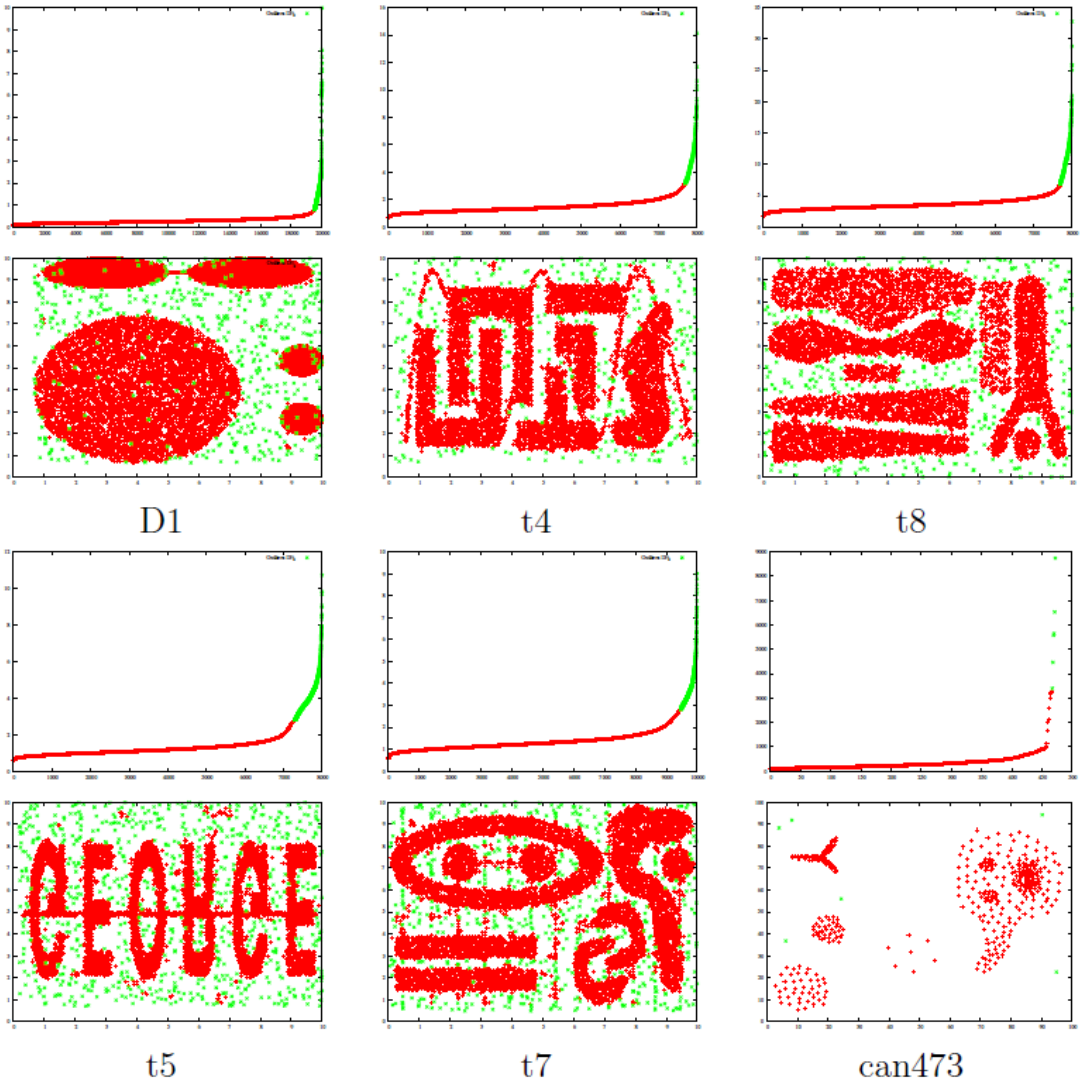


Fig. 5.22. Outlier detection based on the DFk function and STRATIFY algorithm. For each dataset, on the top, we show the partitions corresponding to the DFk curve. Following Breunig et al. (2000), k has been set to 10 for t8, D1, t4, and can473; k = 20 for t5 and t7. From Cassisi et al. (2012b).

5.5.2. Development of a new density-based algorithm

We propose the following method, to improve the quality of DBSCAN algorithm:

- Remove outliers from the dataset for clustering by applying the *STRATIFY* algorithm presented of Table 5.1;

- The knowledge inferred during the outlier detection phase is embedded into the new residual space by adding a new dimension, whose value, for each point, represents the sum of distances of IS_k;

- IS_k improves separation of clusters with different densities. Clusters having the same densities will be embedded into a common strata related to the new dimension.

- We apply the proposed density based clustering algorithm called *ISDBSCAN*, in the new residual dataset (see Table 5.3 for the pseudocode). For each object, we compute IS_k the neighborhood. Then a random point is selected and a depth-first cluster expansion procedure is applied. The method constructs a cluster around a point p until a border point or an outlier is reached. A border point is recognized by the algorithm by checking the size of IS_k. When the algorithm, reaches an object p whose size of IS_k (p) is below a certain threshold (a threshold valued of $2k/3$ has given experimentally satisfactory results), the subset is not processed and the point is classified as noise.

```
Algorithm ISDBSCAN(D, k)
begin
i = 1;
while D ≠ Ø do
     p = Randomly pick an object from D;
     Ci = MakeCluster(D, p, k, i);
     D = D \ Ci;
     if |Ci| > k then
          membership(p) = i; //assign p to i-th cluster
          i = i + 1;
     else
          membership(p) = noise; // assign p to noise
     end if
end while
return {C1, C2, ..., Ci-1};
end
```

Table 5.3. Pseudocode of the *ISDBSCAN* algorithm. The *MakeCluster* subroutine is shown in Table 5.4.

```
Algorithm MakeCluster(D, p, k, i)
begin
if |IS_k(p)| > 2k/3 then
     for each q in IS_k(p) do
          if membership(q) = -1 then
               membership(q) = i;
               C = C + {q};
               C = C + MakeCluster(D, q, k, i);
          end if
     end for
end if
return C;
end
```

Table 5.4. The *MakeCluster* subroutine is a depth-first algorithm. The threshold, which defines the size of IS_k, determines a stop condition during the *MakeCluster*. It has been set to $2k/3$. k is the number of neighbors, and is the only parameter to be set.

In Figure 5.23, we depict the embedding of two datasets provided by Fahim et al. (2009a). Pictures show contours representing the levels in which the points have been embedded by using IS_k and N_k, respectively. The pictures show that the embedding using the IS_k function is able to separate clusters and gives a smoother treatment of the objects lying in the clusters' border with respect to the N_k function.

Note that, the algorithm presented above needs only one parameter to be set which is k. This parameter represents the number of k-nearest neighbors needed to have a sound size of IS_k. In some cases, small dimension of IS_k sets can be viewed as gateways of close clusters. Finally, the algorithm is independent of the starting point and naturally discriminates clusters having different densities. This does not happen to DBSCAN and variants of it (Fahim et al., 2009b), for which it is convenient to start from points of highest density.

ISDBSCAN looks, for expanding clusters, at IS_k-neighbourhood instead of the ε-neighbourhood. Whereas the new neighbourhood relationship is symmetric, objects belonging to clusters of different densities, cannot be considered as neighbors. So, it can randomly select any object to start with cluster expansion, because this method naturally recognizes density

differences, overcoming classical *DBSCAN* limitations (see some example on Figs. 5.24-25).

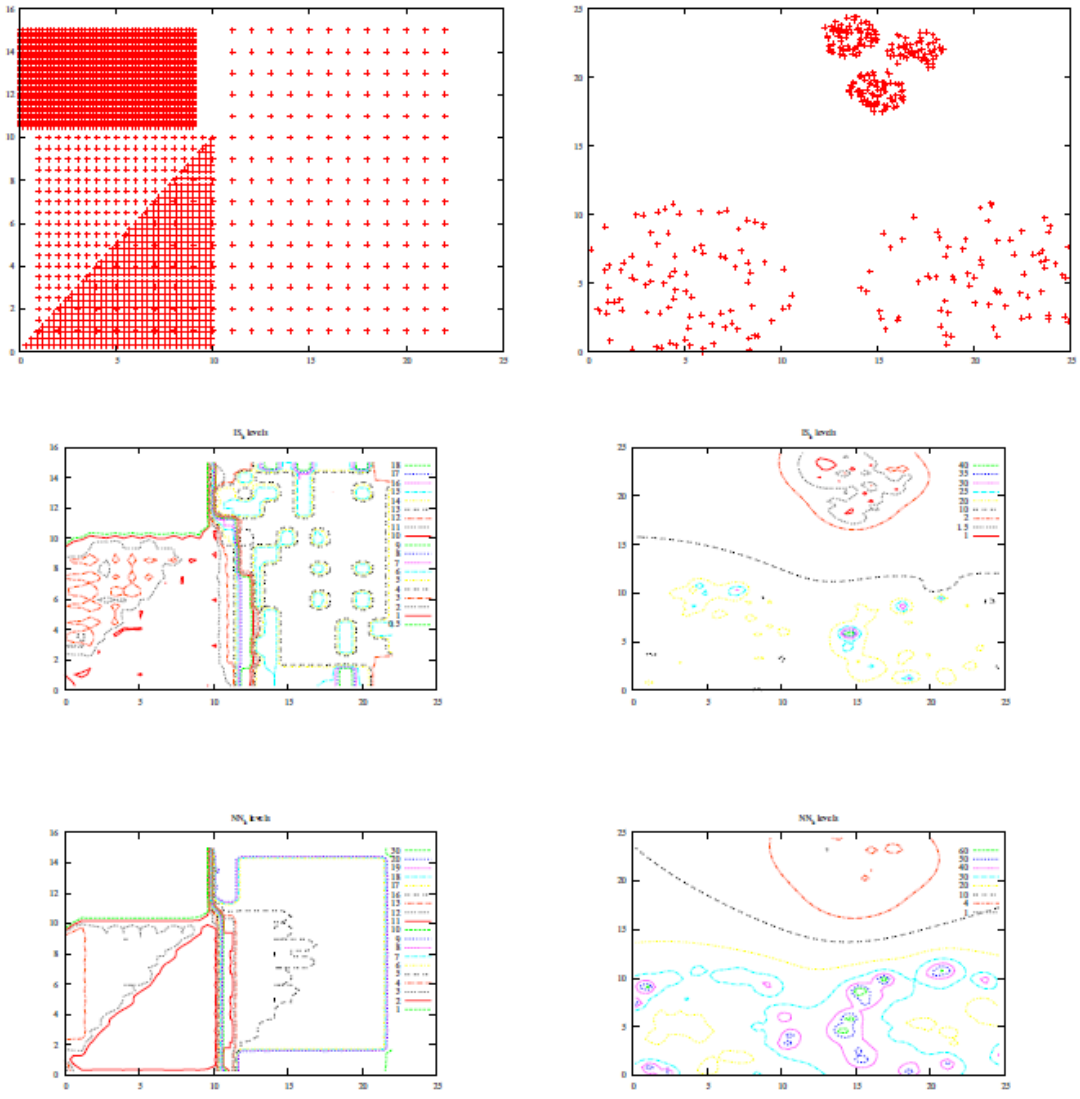


Fig. 5.23. Density levels on the dataset *can3147* (left column) and *can383* (right column). We have used as third dimension IS_k and N_k, respectively. The plots (second and third rows) represent two dimensional contour graphs of the three dimensional datasets given in the first rows, using IS_k and N_k, respectively. IS_k function separates clusters and gives a smoother treatment of the objects lying in the clusters' border with respect to the N_k function. We can observe that the difference between close two dimensional contour lines in N_k is much higher that the corresponding using IS_k. From Cassisi et al. (2012b).

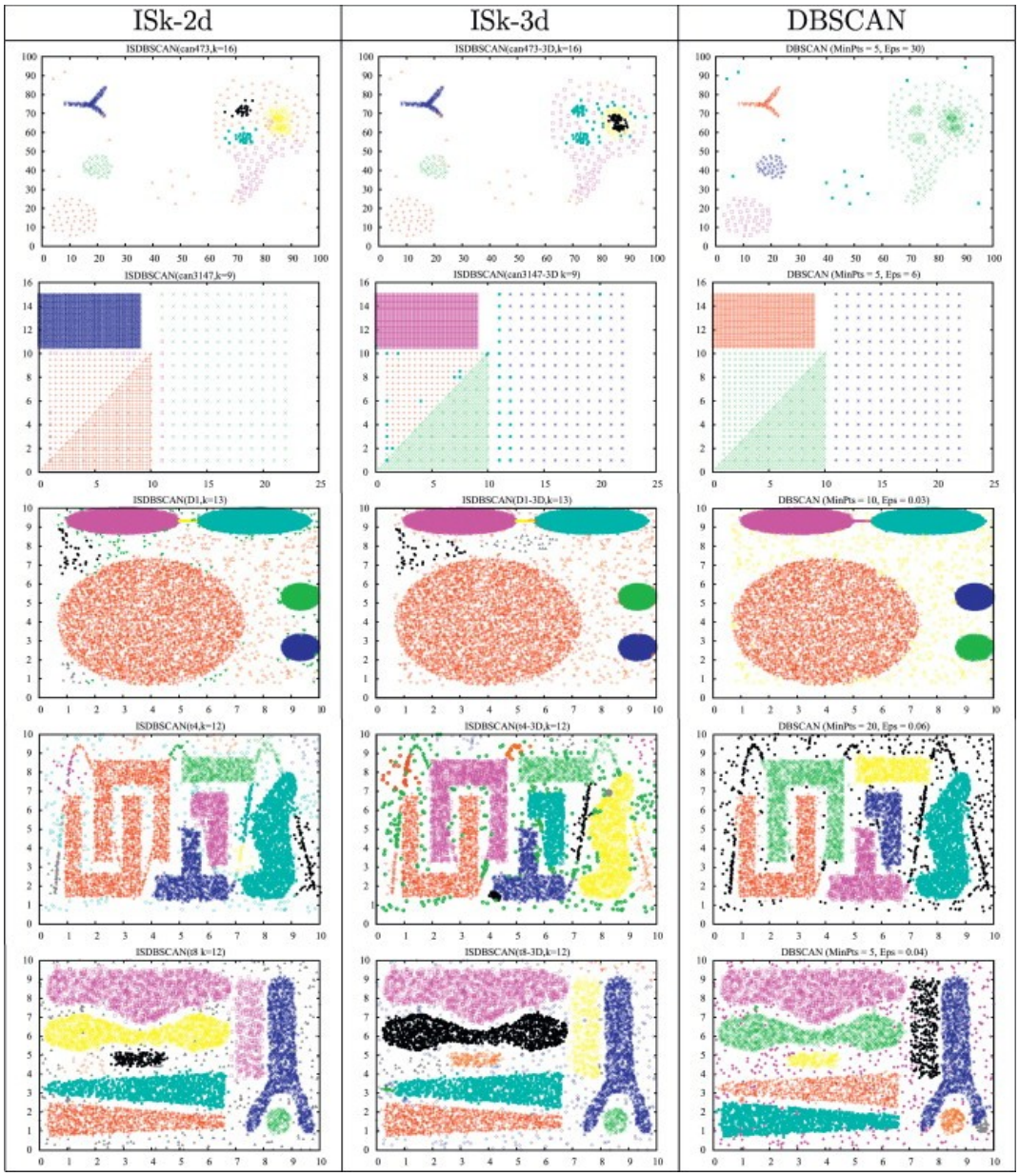


Fig. 5.24. Left column: ISDBSCAN applied on 2Ddatasets. Center column: ISDBSCAN performance and relative IS_k 3D projections. Right column: corresponding best DBSCAN results. From Cassisi et al. (2012b).

This system proposes a stratification process that contains a good heuristic to remove noise (global and local outliers). This step provides further information for enhancing both classic and proposed clustering implementations. For *DBSCAN*, it automatically determines the denser core-point from which to start expanding clusters; while given a fixed parameter *MinPts*, it allows to properly estimate the ε parameter value, by analyzing the DF_k curve. For *ISDBSCAN*, makes computation of IS_k-

neighborhood (useful for expanding cluster phase), and the relative distances, that can be used to project data in a new space to amplify density differences.

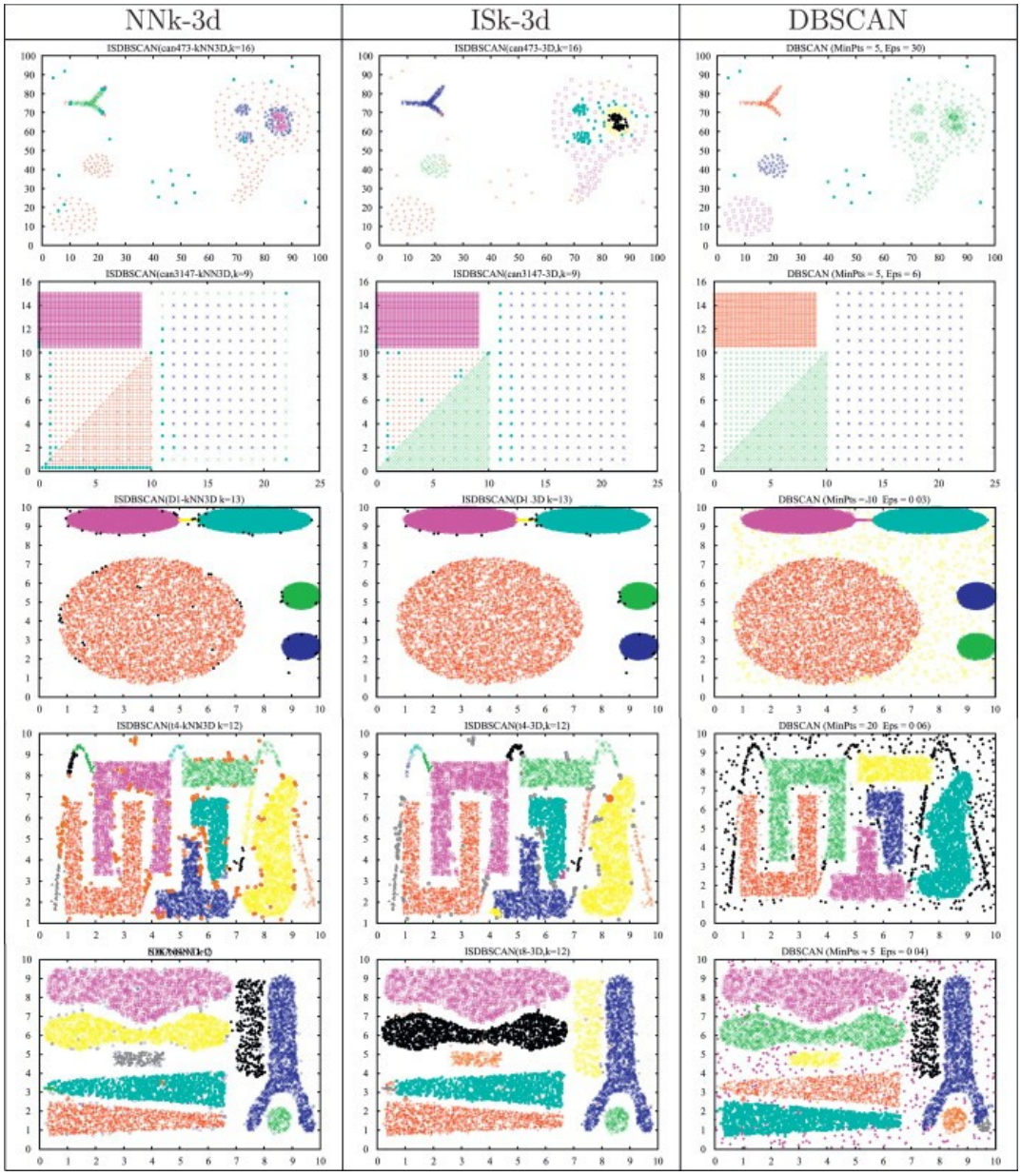


Fig. 5.25. Datasets after removing noise. Left column: ISDBSCAN on NNk 3D projections. Center: ISDBSCAN on ISk 3D projections . Right: comparison with best DBSCAN results. From Cassisi et al. (2012b).

Since *ISDBSCAN* suffers the presence of uniform distributed noise (for example by classifying it as a whole cluster; see Fig 5.24 on center column at row 3), stratification step allows also to efficiently remove it before

clustering. Even though the proposed method deals with low-dimensional data, the *ISDBSCAN* procedure can be used as subroutine of methods for solving subspace clustering problems in high-dimension (see Zimek, 2008 for a survey).

5.5.3 DBStrata

The proposed *ISDBSCAN* algorithm is implemented in a software system called *DBStrata* (Cassisi et al., 2011b). *DBStrata* has been developed in Python 2.6, it uses Python Scientific Library (Scipy/Numpy) and the PyQt Graphical User Interface Library. It interacts using a GUI, which allows to run density-based clustering, stratification pre-processing phase, outlier detection, IS_k-projection, and *ISDBSCAN* clustering. Moreover, for comparisons purpose a framework allows to run the *OPTICS* module (see Section 5.3.3).

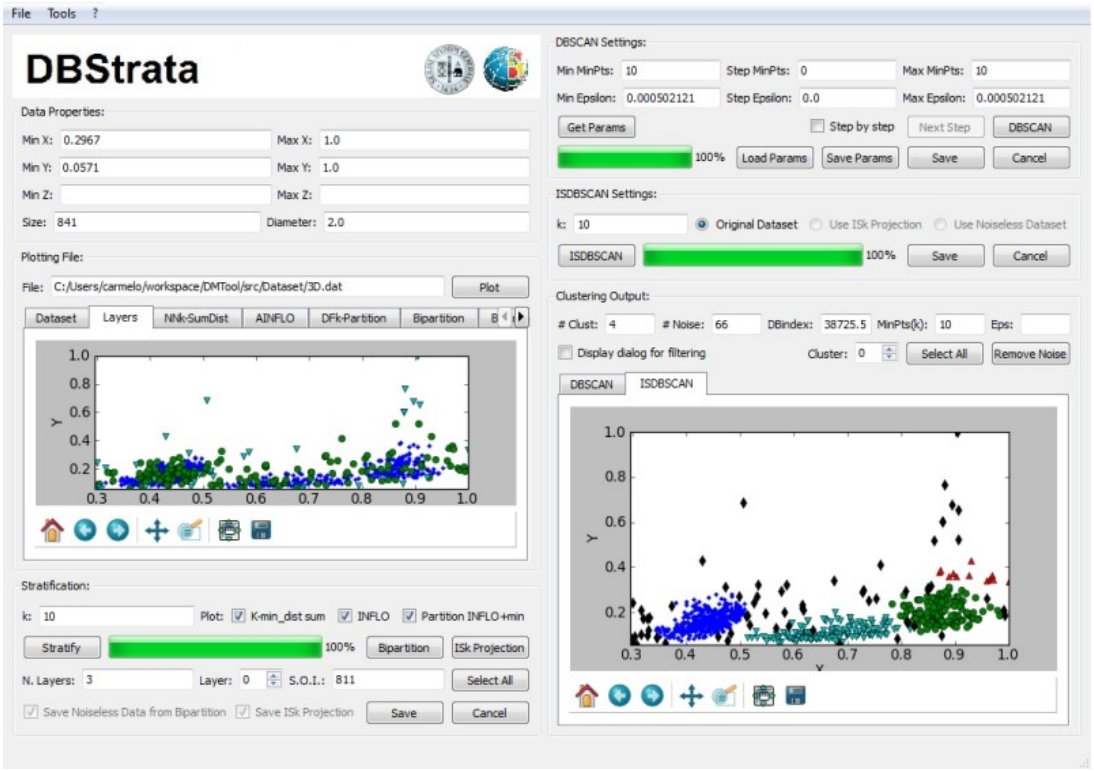


Fig. 5.25. The *DBStrata* main interface. The software available for download at the following web site http://www.dmi.unict.it/~cassisi/DBStrata/.

The main interface allows to apply density-based clustering to loaded files, including stratification pre-processing phase, useful to help user on *DBSCAN* parameters setting, outlier detection, IS_k-projection, and *ISDBSCAN* clustering (see the software map in Fig. 5.26). It can be considered the main program; from this interface, user can calls the other part of the software relative to *OPTICS* framework.

When the application starts, the only action allowed is the *Open* action under the *File* menu. This opens a dialog box to load *.tsv* (tab-separated-values) files, that stores data table in which columns of data are separated by tabs. The layout has six group boxes: 1) *Data Properties*; 2) *Plotting File*; 3) *Stratification*; 4) *DBSCAN Settings*; 5) *DBSCAN Output*. After the file is loaded, the first four group boxes are enabled.

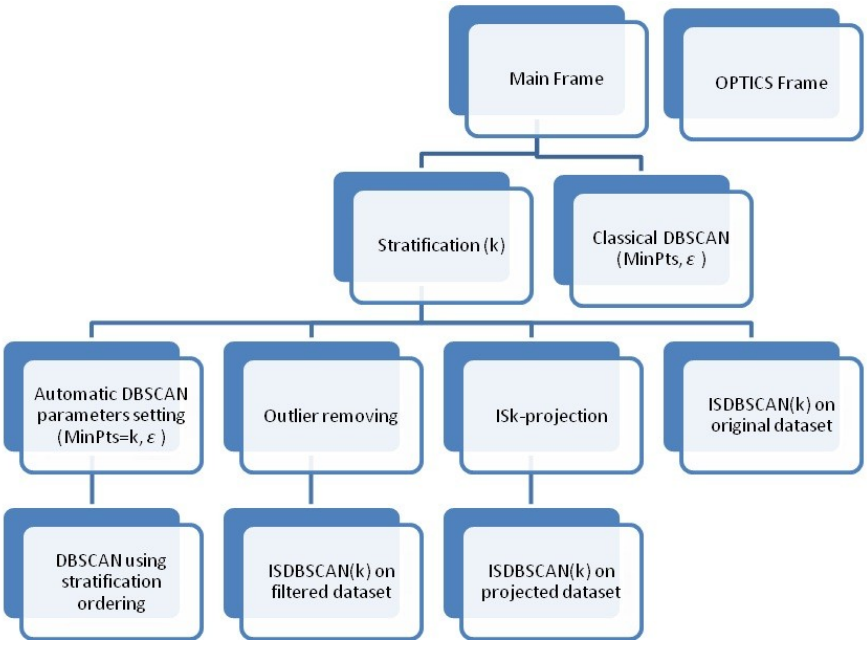


Fig. 5.26. Structure and flow diagram of the software. Main interface calls only another interface to clustering with *OPTICS* system.

Data Properties group shows minimum and maximum values for each coordinate, and the diameter of the space; it is a read-only group.

Plotting File group contains the plot of the loaded file. Our Python implementation (Python 2.6) uses the Matplotlib library that offers good quality in data plotting. In addition to the figures canvas is made available a navigation toolbar.

Stratification group allows editing of first input k to perform stratification (default $k=10$). The following plots are included:

1. *Stratification layers* with different colors;
2. *Ordering* based on sum of NN_k distances from p, ω_k;
3. Relative *AINFLO* values;
4. DF_k curve partitioned into the returned layers;
5. (Optional) Dataset *Bipartition* discriminating outliers;
6. (Optional) Dataset *ISk projection*.

Stratification returns the *S.O.I. (Start Outliers Index)*, the number of layers, and the range value for ε defined between ε_1 and ε_2, setting respectively the *"Min MinPts"* and *"Max MinPts"* line edit placed on the *DBSCAN Settings* group. In this last, the interval $[\varepsilon_1, \varepsilon_2]$ is by default divided into *10* intervals for a multiple execution of the algorithm. Although automatic setting can be useful and precise in many cases, before clustering execution it is always advisable to check the sum of NN_k distances ordering plot to ensure logical settings, and especially to verify that the *S.O.I.* has visual feedback with the surge of the curve. A not suitable setting can bring long running execution time for *DBSCAN*. Due to this, all edit lines can be edited for custom handling of parameters range. You can also choose to proceed step-by-step *(Next Step* button), or make a single run (*DBSCAN* button). Our implementation uses *kd-Tree* for data indexing, whose Python implementation is available on http://sites.google.com/site/mikescoderama/Home/kd-tree-knn.
Both optional *bipartition* and IS_k *projection* are modified versions of the original dataset and are saved into the working directory. *ISDBSCAN Settings* group also allows running of ISDBSCAN with input parameter k on each dataset version.
Clustering Output group shows the output of the algorithm relative to the input parameters: number of clusters, outliers found, and a validation index for clustering inspired to (Davies and Bouldin, 1979). It allows to display its results in a dedicated dialog *Other Filters* to filter the result of seismic signals clustering (only *2D* dataset), it includes: noise removing;

selection of a labeled cluster or of main clusters having a number of points greater than a percentage of dataset size; and moving average of each filtering, by defining the size of the moving average window.

The *Main Frame* (Fig. 5.25) offers the vast majority of the software tools. Under *Tools* action, user can call another interface relative to *OPTICS* framework. The routine to calculate *Reachability* and *Core Distance* plot was imported from http://chemometria.us.edu.pl/index.php?goto=downloads, referring to (Daszykowski, et al., 2002).

The window was designed to present the main dataset characteristics retrievable from *OPTICS* algorithm. In the left section, *OPTICS plots* group shows *reachability* plot, *core distance* plot, and a visual path of dataset ordering. In the right section, *Clustering* group includes the choice to cluster through *DBSCAN*, setting the relative parameter; or to use the system proposed in (Sander, et al., 2003) to discover, if exist, the hierarchies between clusters on different densities dataset.

Chapter 6

Geophysical application of data mining

This chapter contains a collection of works, concerning the application of data mining on geophysical data, that became object of publication (Aliotta et al., 2010; Cannata et al., 2011a; Lo Castro et al., 2011a; Lo Castro et al., 2011b; Cassisi et al., 2012a; Montalto et al., 2012). This was made possible thanks to the collaboration with the INGV (Istituto Nazionale di Geofisica e Vulcanologia), Section of Catania - Osservatorio Etneo, which makes available its data for this kind of research.

6.1 Clustering and classification of infrasonic events at Mount Etna using pattern recognition techniques

Active volcanoes generate sonic and infrasonic signals, whose investigation provides useful information for both monitoring purposes and the study of the dynamics of explosive phenomena.
Over the last decades, Mt. Etna volcano (Italy) has been characterized by a remarkable increase in the frequency of shortlived, but violent eruptive episodes at the summit craters. Between 1900 and 1970, about 30 paroxysmal eruptive episodes occurred at the summit craters, while there have been more than 180 since then (Behncke and Neri, 2003). The summit area of Mt. Etna is currently characterized by four active craters: Voragine, Bocca Nuova, Southeast Crater and Northeast Crater (hereafter referred to as VOR, BN, SEC and NEC, respectively; see Fig. 6.1). These craters are characterized by persistent activity that can be of different and sometimes coexistent types: degassing, lava filling or collapses, low rate lava emissions, phreatic, phreato-magmatic or strombolian explosions and lava fountains (e.g. Cannata et al. 2008). At Mt. Etna in 2006, a permanent infrasound network was deployed providing useful information to

monitor the explosive activity (Cannata et al. 2009a,b; Di Grazia *et al.* 2009). Unfortunately, sometimes during the winter
season owing to bad weather conditions, the lack of signals from some summit stations prevents applying the aforementioned location algorithms.

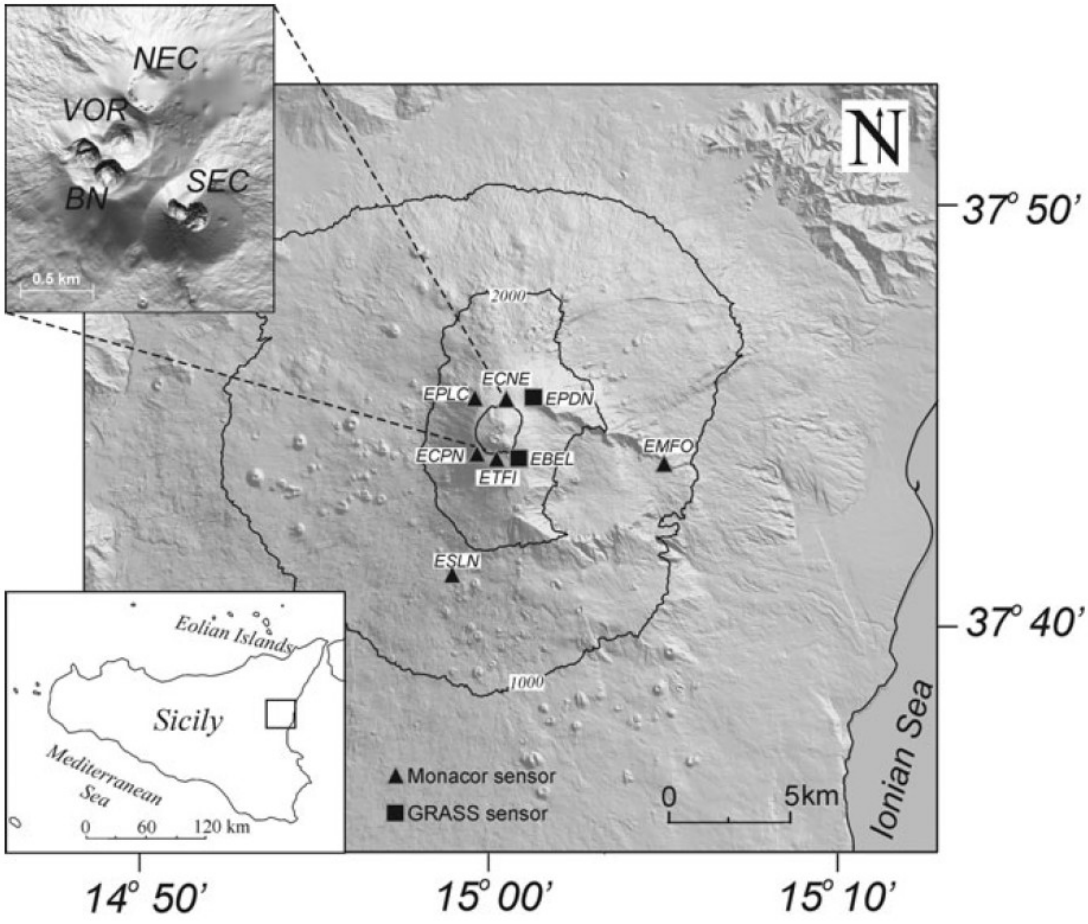


Fig. 6.1. Digital elevation model of Mt. Etna with the location of the infrasonic sensors (triangles and squares), composing the permanent infrasound network. The upper right inset shows the distribution of the four summit craters (VOR, Voragine; BN, Bocca Nuova; SEC, Southeast Crater; NEC, Northeast Crater) (from Cannata et al., 2011a).

Here, we propose a new system, based on pattern recognition techniques, able to identify at Mt. Etna the active summit crater from the infrasonic point of view using only the signal recorded by a single station.
First, by a parametric power spectrum method, the features describing and characterizing the infrasound events were extracted: peak frequency

and quality factor. Then, together with the peak-to-peak amplitude, these features constituted a 3-dimensional feature space; by means of *DBSCAN* algorithm (see Section 5.3.3) three clusters were recognized inside it. After the clustering process, by using a common location method (semblance method) and additional volcanological information concerning the intensity of the explosive activity, we were able to associate each cluster to a particular source vent and/or a kind of volcanic activity. Finally, for automatic event location, clusters were used to train a model based on *Support Vector Machine* (*SVM*; see Section 4.1.5), calculating optimal hyperplanes able to maximize the margins of separation among the clusters. After the training phase this system automatically allows recognizing the active vent with no location algorithm and by using only a single station. This work was partly performed with grants of the 'Flank project' (INGV-DPC 2007–2009).

6.1.1 Infrasound features at Mt. Etna

Some recent studies have shown that the infrasonic signal at Mt. Etna is generally composed of amplitude transients (named 'infrasonic events'), characterized by short duration (from 1 to over 10 s), impulsive compression onsets and peaked spectra with most of energy in the frequency range 1–5 Hz (Fig. 5.2; Gresta et al. 2004; Cannata et al. 2009a,b). Similar features are also observed at several volcanoes, though characterized by different volcanic activity, such as Stromboli (Ripepe et al. 1996), Klyuchevskoj (Firstov and Kravchenko 1996), Sangay (Johnson and Lees 2000), Karymsky (Johnson and Lees 2000), Erebus (Rowe et al. 2000), Arenal (Hagerty et al. 2000) and Tungurahua (Ruiz et al. 2006).
Since the deployment of the infrasound permanent network at Mt. Etna in 2006, two summit craters have been recognized as active from the infrasonic point of view: SEC and NEC (Cannata et al. 2009a,b). The former has been characterized by sporadic explosive activity with different intensity, from ash emission to lava fountaining, while the latter mainly by degassing. According to Cannata et al. (2009a,b), these craters generate infrasound signals with different spectral features and duration: 'SEC events', showing a duration of about 2 s, dominant frequency mainly higher than 2.5 Hz and higher peak-to-peak amplitude than the NEC

events (Fig. 5.6a); 'NEC events', lasting up to 10 s and characterized by dominant frequency generally lower than 2.5 Hz (Fig. 6.2b).

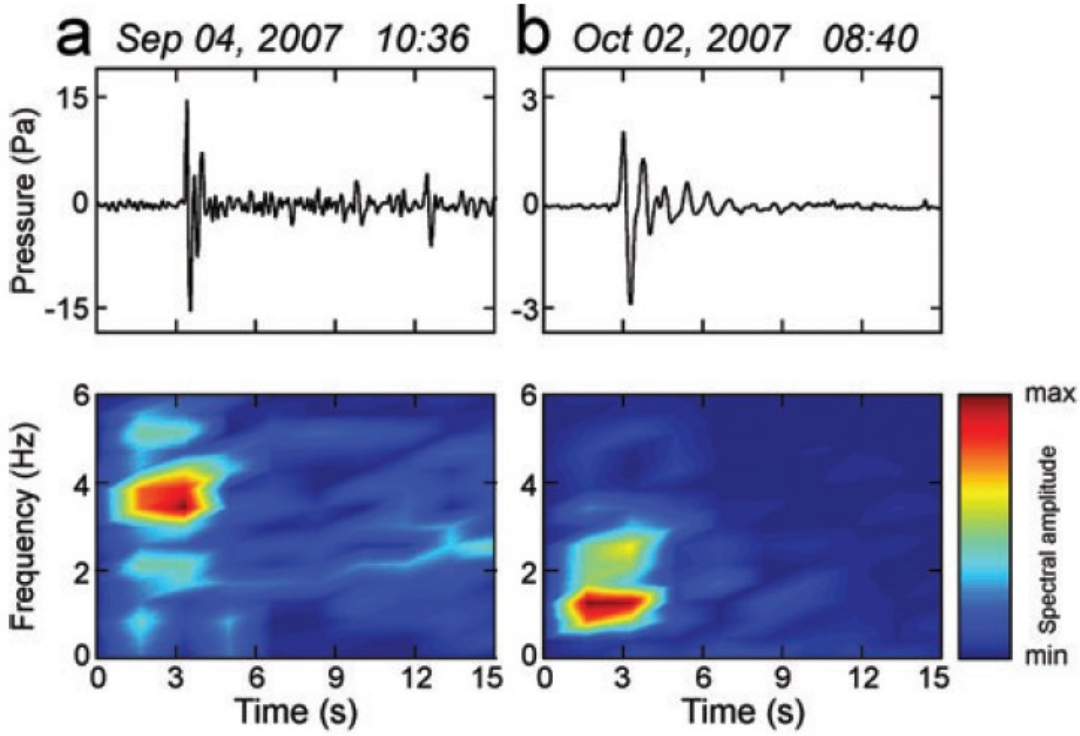


Fig. 6.2. Infrasonic events recorded by EBEL station and corresponding Short Time Fourier Transform, obtained by using 2.56-s long windows overlapped by 1.28 s. The event in (a) is a typical 'SEC event', the one in (b) a typical 'NEC event' (from Cannata et al., 2011a).

6.1.2 Data acquisition and infrasound signal characterization

In the following subsections (i) data acquisition and event detection, (ii) features extraction and (iii) the semblance algorithm are briefly described.

Data acquisition and event detection. Since 2006, the permanent infrasound network run by Istituto Nazionale di Geofisica e Vulcanologia, Section of Catania, has been composed of a number of stations ranging from one to eight depending on the considered period, located at distances ranging between 1.5 and 7 km from the centre of the summit area (Fig. 5.1). Today, some stations are equipped with Monacor condenser microphones MC-2005, with a sensitivity of 80 mV Pa−1 in the 1–20 Hz infrasonic band, while others with GRASS 40AN microphone with a flat

response with sensitivity of 50 mV Pa−1 in the frequency range 0.3–20000 Hz. The infrasonic signals are transmitted in real-time by means of radio link to the data acquisition centre in Catania where they are acquired at a sampling rate of 100 Hz. At Mt. Etna we use EBEL as reference station, because it generally shows a very good signal-to-noise ratio and, unlike the other summit stations, its maintenance is generally feasible even during the winter season. Once the infrasound signal is recorded, the signal portions of interest, that are the infrasonic events, have to be extracted. Then, the root mean square (rms) envelope of the infrasonic recordings is calculated by a moving window of fixed length. Successively, we calculate the percentile envelope on moving windows of rms envelope. For a given time-series, the pth percentile can be defined as the value such that at most (100 × p) per cent of the measurements are less than this value and 100(1 − p) per cent are greater. In light of this, the estimation of percentile enables us to efficiently detect amplitude transients and estimate background signal level. The percentage threshold should be chosen on the basis of both the amount of transients in the signal that have to be included or excluded in our calculations and the signal-to-noise ratio. The performance of this method was compared with the short time average/long time average (STA/LTA) technique (e.g. Withers 1997; Withers et al. 1998). The lengths of short and long windows, mainly depending on the frequency content of the investigated signal, were fixed respectively to 2.5 and 12.5 times the dominant period of the signal (equal to roughly 0.3 s), considered a reasonable compromise between sensitivity and noise reduction (Withers 1997), and the detection threshold to 1.7. As shown in Fig. 6.3, the trigger results obtained by the two methods were similar; nevertheless, the technique based on percentile was also able to detect transients very close to each other.

Infrasonic signal features extraction. Often the decomposition of a time series into purely harmonic components (Fourier transform case) can be impractical. In fact, the actual oscillations observed in geophysics often decay (or grow) exponentially with time, due to some mechanisms of energy dissipation (or supply), as if the frequency were complex (Kumazawa et al. 1990). Therefore, the spectral structure will be

reasonably represented in the complex frequency space (Kumazawa et al. 1990).

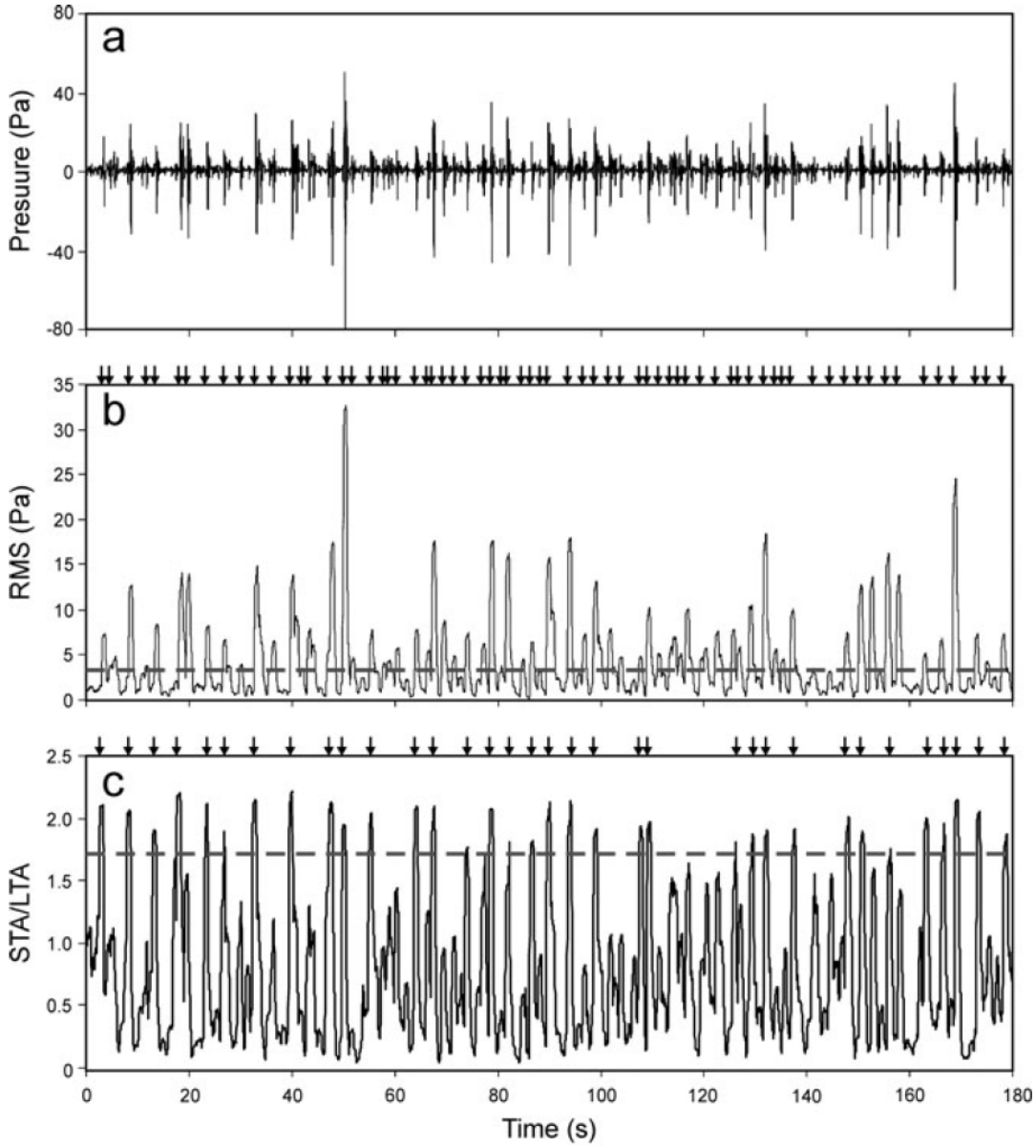


Fig. 6.3. (a) Three-minute long infrasound signal recorded by EBEL station, (b) corresponding rms envelope (black line) calculated by using a moving window of 0.7 s and (c) STA/LTA values. The horizontal grey dashed line in (b) indicates the detection threshold calculated by a percentile value of 5 multiplied by 5. The horizontal grey dashed line in (c) indicates the detection threshold fixed at 1.7. The arrows at top of (b,c) indicate the onset time of the detected events (from Cannata et al., 2011a).

Since infrasonic events can be represented as decaying complex exponential functions, to determine their complex frequency the *Sompi*

method can be used (Kumazawa et al. 1990, and references therein). This is a high-resolution spectral analysis method based on an autoregressive (AR) filter. By this method, a given time series is resolved into a number of 'wave elements' that consist of decaying harmonic components, and additional noise (more details about Sompi method are reported in the Appendix B). Each wave element is specified by two complex parameters z and α (Kumazawa et al. 1990):

$$z = \exp(\gamma + iw) \tag{6.1}$$

$$\alpha = Ae^{i\theta} \tag{6.2}$$

where γ and ω are the real and imaginary parts of the complex angular frequency, A and θ correspond to the real amplitude and phase of the wave element referred to some origin point and finally i is $\sqrt{-1}$. Another two parameters, ordinary real frequency and 'gradient' or 'growth rate', referred as to f and g, respectively (Kumazawa et al. 1990), are given by:

$$f = \omega / 2\pi \tag{6.3}$$

$$g = \gamma / 2\pi \tag{6.4}$$

Finally, the 'dissipation factor' or 'quality factor' Q is defined as:

$$Q = -f / 2g \tag{6.5}$$

Generally, to represent a set of complex frequencies, their locations are plotted on a 2-D plane with f and g axes. The wave elements scattering widely in the plot, as the AR order changes, are considered noise. It is also possible to identify some wave elements densely populated on the theoretical frequency lines that remain mainly stable as the AR order changes. They are considered dominant spectral components (Hori et al. 1989). An example of frequency-growth rate domain for an infrasound event recorded by EBEL station is reported in Fig. 6.4.

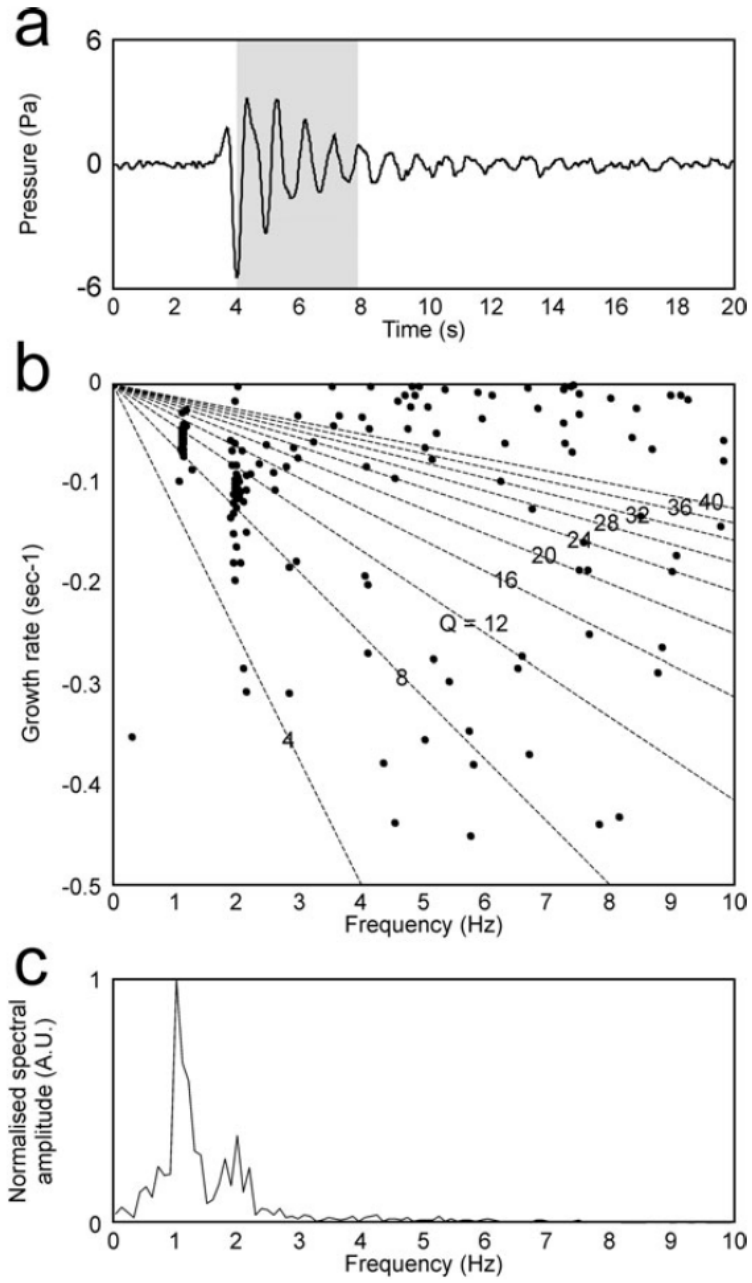


Fig. 6.4. (a) Infrasound event recorded by EBEL station, and corresponding (b) frequency-growth rate plot (AR order 2–60) and (c) amplitude spectrum. The grey area in (a) represents the window used to calculate the frequency-growth rate plot in (b). The dashed lines in (b) represent lines along which the quality factor (*Q*) is constant. Clusters of points in (b) indicate dominant spectral components of the signal; scattered points represent noise (from Cannata et al., 2011a).

Therefore, in summary, the spectral features of an infrasonic event can be described by the two parameters *Q* and *f*. Further, in addition to frequency and quality factor, the third feature used to characterize the infrasound

events is the peak-to-peak amplitude, depending on both distance source-station and energy of the infrasonic source.

Semblance algorithm. The location of the source of the infrasonic events, generally coinciding with active vents, is of great importance for volcanic monitoring. Therefore, different location techniques, generally based on grid searching procedures, were developed (e.g. Ripepe and Marchetti, 2002; Jones et al. 2008; Johnson et al. 2010; Montalto et al. 2010).
The semblance technique is based on the semblance function that is a measure of the similarity of multichannel data (Neidell and Taner 1971). For infrasonic events this method applies a 2-D grid searching procedure over a surface covering the summit area and coinciding with the topographic surface. The infrasonic source is assumed to be in each node of the grid, and for each node the theoretical travel times at the sensors are first calculated. Then, infrasonic signals at different stations are delayed and compared by the semblance function. Finally, the source is located in the node where the delayed signals show the largest semblance value. Therefore, the semblance function is assumed representative of the probability that a node has to be the source location (further details about the method are reported in Montalto et al. 2010). In Fig. 6.5 two examples of infrasound location are reported for a SEC event and a NEC event.

6.1.3 Learning phase

In the proposed system, the learning phase merges together results of clustering and classification analysis (Fig. 6.6). *DBSCAN* and *SVM* are applied on infrasound event features together with geophysical information used to 'label' the recognized clusters. About 665 events, recorded during 2007 September–November at EBEL station, were detected and filtered in frequency range 0.5–5 Hz. The feature extraction from the detected events was performed by *Sompi* method (see also Appendix B) using 2-s long windows of infrasonic signal recorded at EBEL station and AR order equal to two. The sharply monochromatic nature of the investigated signals justifies the choice of this low order (Lesage 2008).

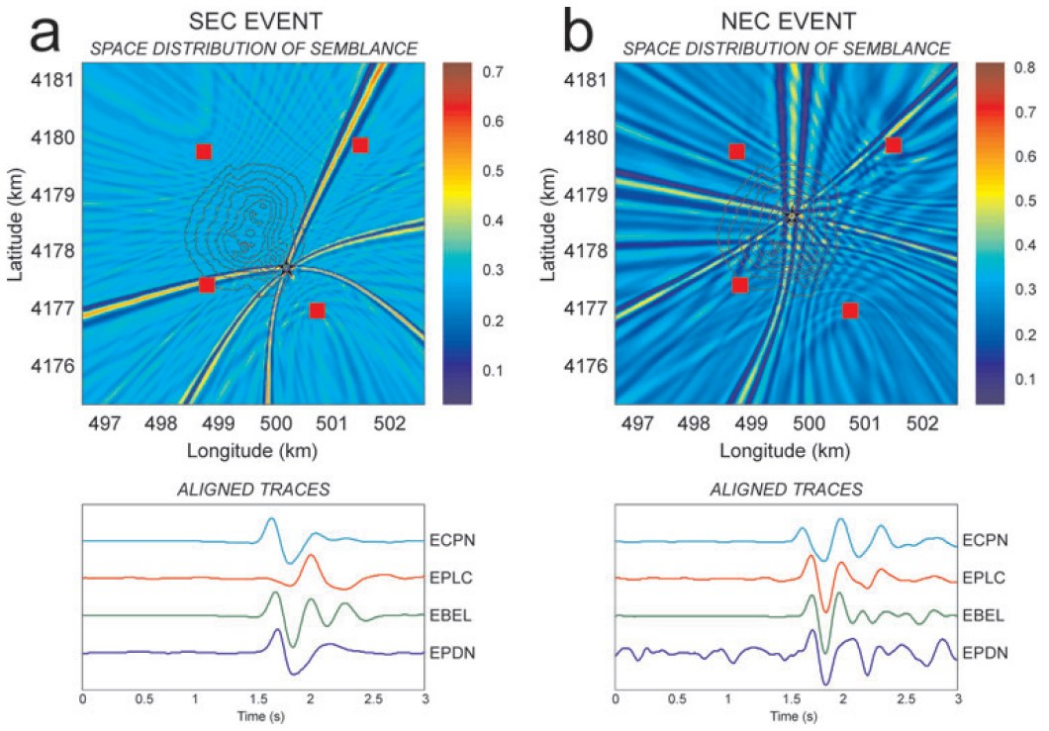


Fig. 6.5. Examples of space distribution of semblance values, calculated by locating two infrasonic events at Mt. Etna, and corresponding infrasonic signals at four different stations shifted by the time delay that allows obtaining the maximum semblance. The red squares and stars in the top plot indicate four station sites and the nodes with the maximum semblance value, respectively. The black lines in the top plot are the altitude contour lines from 3 to 3.3 km a.s.l. (from Cannata et al., 2011a).

Frequency and quality factor of the events, together with peak-to-peak amplitude, constituted the feature space and are plotted in Fig. 6.7. Then, to discover clusters in this space, 'data clustering' techniques based on *DBSCAN* algorithm (Section 5.3.3) were applied. Using such an algorithm we found three main clusters (called cluster 1, 2 and 3) and other outlier points that can be considered as noise (Fig. 6.8). Points belonging to each cluster are related to infrasonic events that were located using Semblance location method (Section 6.1.2). In accordance with Cannata et al. (2009b), during 2007 September–November, two infrasonic sources were found, NEC and SEC. In particular, a cluster was composed of events generated by NEC (cluster 1) and the other two by SEC. Such last two clusters were related to different kinds of explosive activity at SEC. In particular, the events belonging to cluster 3 were coincident with 'more visible' explosions, characterized by a relevant presence of ash, whereas the

events of cluster 2 were hardly visible in the monitoring video-camera recordings (Cannata et al. 2009b).

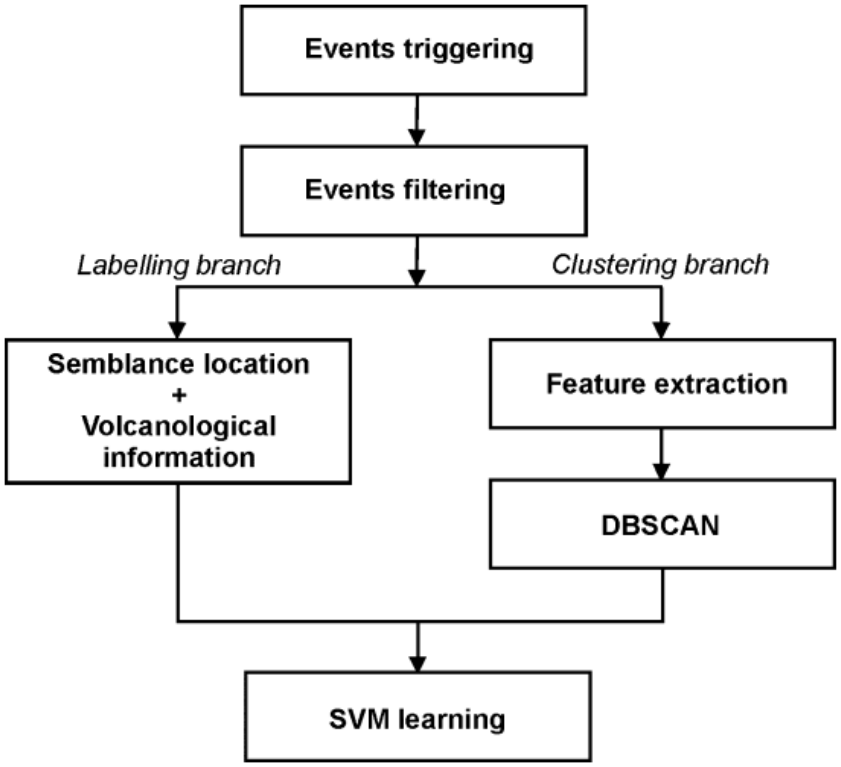


Fig. 6.6. Scheme of the learning system (from Cannata et al., 2011a).

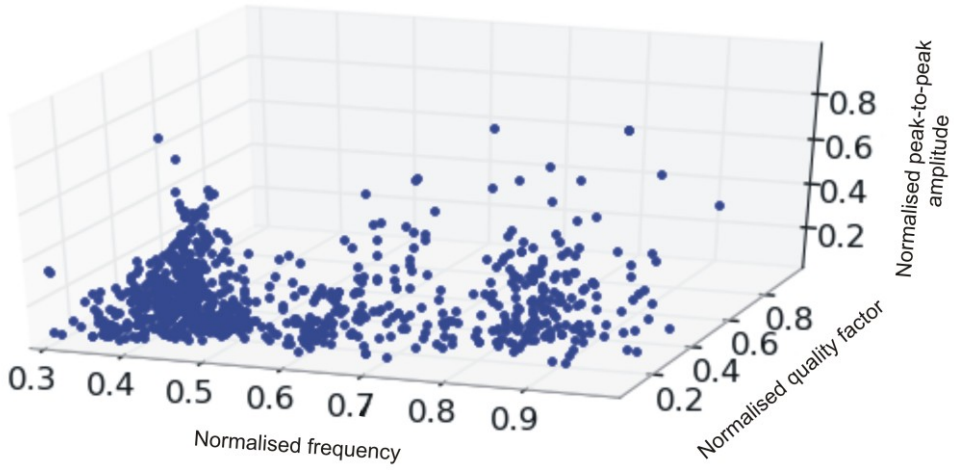


Fig. 6.7. Feature space with frequency, quality factor and peak-to-peak amplitude of the infrasound events recorded at EBEL station during 2007 September–November 2007.

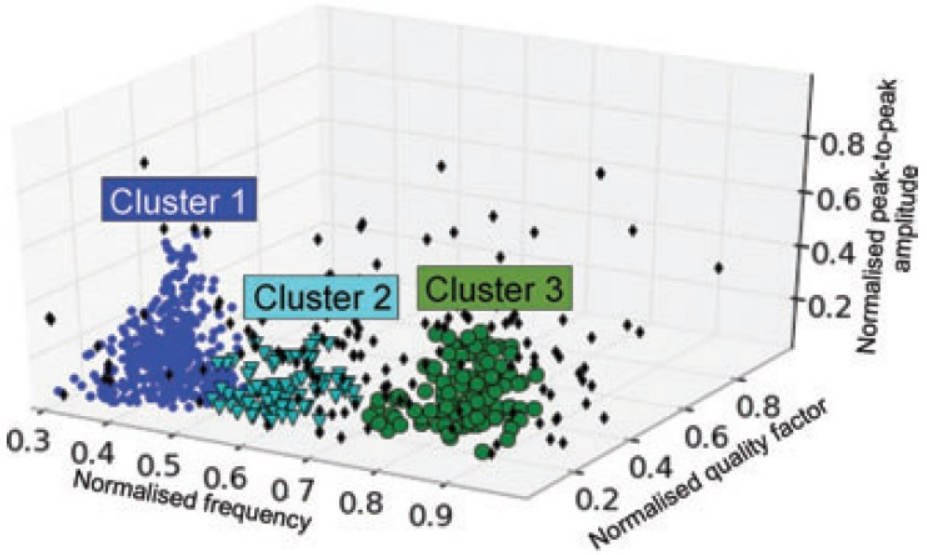


Fig. 6.8. Clustering of the feature space reported in Fig. 10. The clusters are indicated with blue (cluster 1) and green circles (cluster 3) and light green triangles (cluster 2), the outliers with black diamonds (from Cannata et al., 2011a).

Features clustering together with labels provide the patterns for SVM learning process. As mentioned in Section 4.1.5, optimization of parameters C (regularization parameter) and σ (radial basis function kernel parameter, see Eq. 4.9) is a key step in SVM learning because their values determine classification performance (Devos et al. 2009). As a consequence, model selection is applied with the aim of finding the best pair of parameters C and σ that minimizes the error rate estimated as the ratio between misclassified and hit patterns. These parameters can be chosen using a cross-validation (CV) approach (Hastie et al. 2002), which is a statistical method for learning algorithms evaluation and model selection. In particular, in K-fold CV the available data set is partitioned into K subsets or 'folds': K–1 folds are used for SVM learning purpose, and the remaining fold for model validation (Fig. 6.9). Thus, K iteration of learning and validation are performed and for each ith iteration the training process is carried out using K−1 folds and the ith fold for validation (Fig. 6.9).

All SVM training algorithms are computed using one-against-all method (see Section 5.2.1). Since we worked on a small data set, a simple exhaustive grid search can be performed (Hsu et al. 2007). In particular, C

was systematically changed in the range [1–100] with a step of 10, σ in the range [0.1–10] with a step of 0.5 and a *K*-fold CV with $K = 10$ was used.

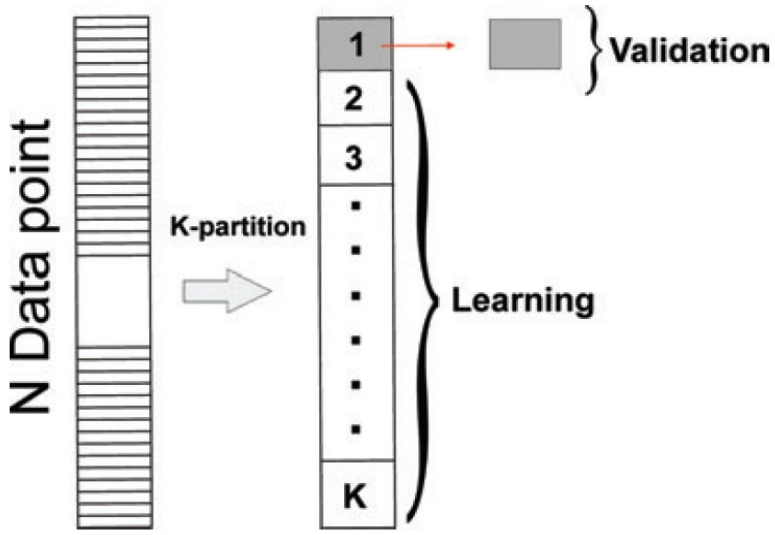


Fig. 6.9. Basic scheme of *K*-Fold Cross Validation (from Cannata et al., 2011a).

The entire procedure can be summarized as follows (Fig. 6.10): (1) a grid value of C and σ is defined; (2) for each pair of C and σ values, a mean error rate is computed averaging the error rate values obtained by the K SVM models; (3) the pair of C and σ with the minimum error rate is selected; (4) such a pair is used to train the final SVM model with the whole data set, comprising all the K folds. Here, the best parameter values were $C = 1$ and $\sigma = 0.1$, for which mean CV error minimized to 0.6 per cent.

6.1.4 Testing phase and final system

To verify the system, the trained SVM is tested by classifying new unknown infrasonic events and then assigning them to their source crater. The reliability is verified using events not analysed during the previous learning phase (Section 5.2.2). To this end, a new test data set of about 610 events, recorded during 2 months, 2007 August and December, was used and labelled by location algorithm based on semblance method (Section 5.1.3). Moreover, the events belonging to cluster 2 and cluster 3 were labelled using information related to the intensity of the explosive activity (Cannata et al. 2009b).

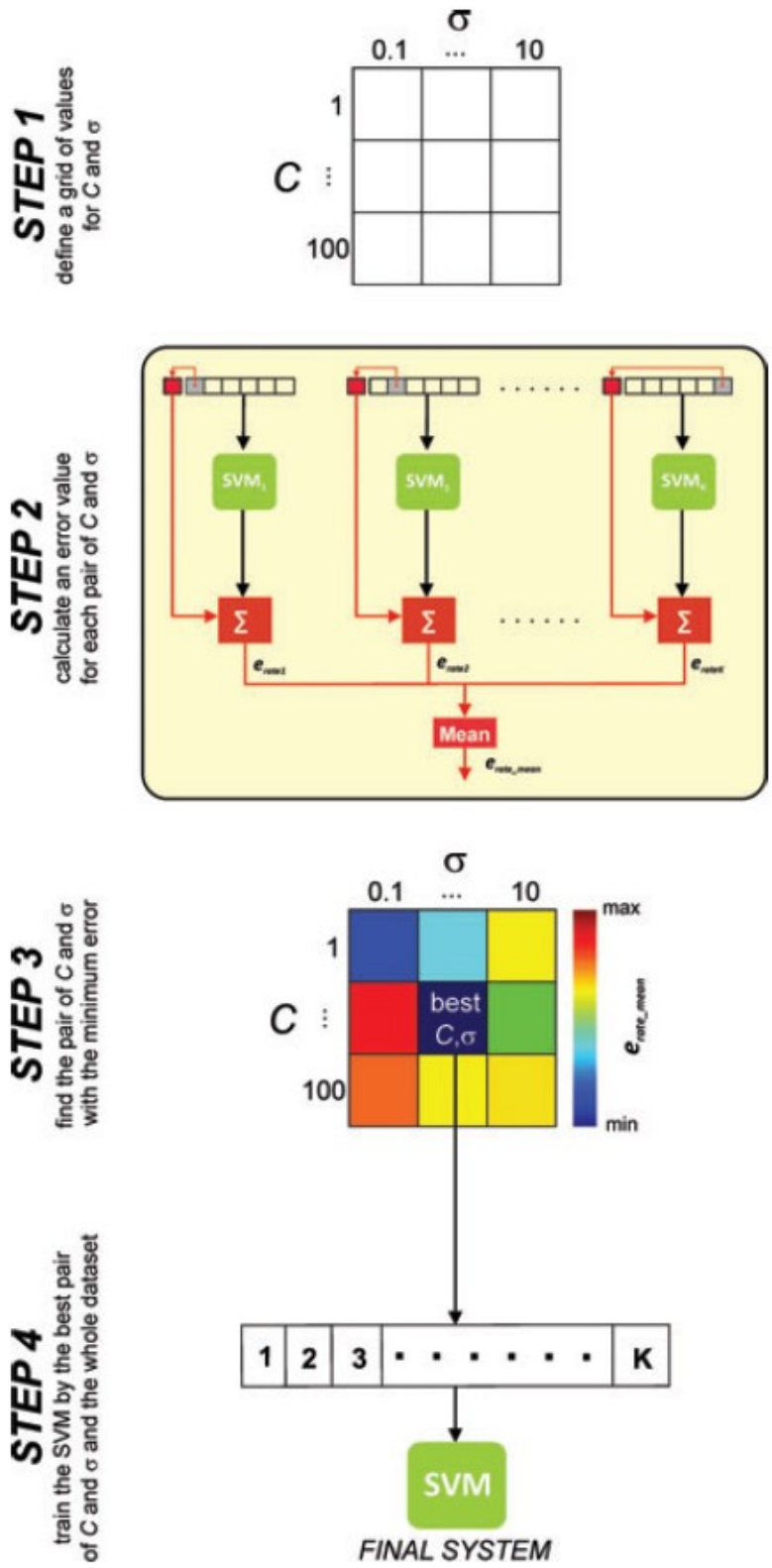


Fig. 6.10. Best SVM model selection using *K*-Fold Cross-Validation (from Cannata et al., 2011a).

The quality of classification is quantified using confusion matrix (Table 6.1), where each column represents the instances in the predicted class (based on the SVM model), while each row represents the instances in the actual class (based on the previously attributed labels). Thus, the entries on the diagonal count the events in which prediction agrees with known labels, whereas the other entries the misclassified events. 63 elements were wrong assigned, providing an error rate of about 11.97 per cent. Misclassifications were mostly concentrated in the second and third classes that are related to the two different explosion activities of SEC crater. Indeed, such a distinction is qualitative and not clearcut, hence many halfway events can be misclassified. If we do not take into account the distinction between clusters 2 and 3, and consider them as a single cluster, the error decreases to 5.25 per cent.

		PREDICTED		
		Cluster 1	***Cluster 2***	***Cluster 3***
ACTUAL	***Cluster 1***	**476**	9	6
	Cluster 2	9	**15**	8
	Cluster 3	8	33	**46**

Table 6.1. Confusion matrix calculated in the testing phase. Each column represents the instances in the predicted class (based on the SVM model), while each row represents the instances in the actual class (based on the previously attributed labels). Thus, the entries on the diagonal (bold numbers) count the events in which prediction agrees with known labels, whereas the other entries the misclassified events.

Finally, the proposed system can be summarized as follows (Fig. 6.11): (i) triggering procedures is performed on buffer of acquired signal; (ii) then, if events are found, the system evaluates whether there is a sufficient number of stations for semblance location algorithm; (iii) if the number of stations is not sufficient, alternative 'single station' location is performed by extracting signal features and classifying them using the trained SVM. It is also worth noting that SVM classifier is also applied offline on localizable events to evaluate its performance in distinguishing NEC events (cluster 1) from SEC events (clusters 2 and 3). In this case, events belonging to clusters 2 and 3 are simply considered SEC events and then

labelled based on the source vent, with no further distinction depending on the type of explosive activity. This task is carried out by comparing the results of the classifier with the location parameters provided by the semblance algorithm. By the inspection of the obtained error rate, a new clustering execution is necessary when classification of new signals is not aligned with that of infrasonic network classifier. This may be caused by the creation of a new active vent or by the changing activity of a pre-existing vent; in such a case the system must be updated.

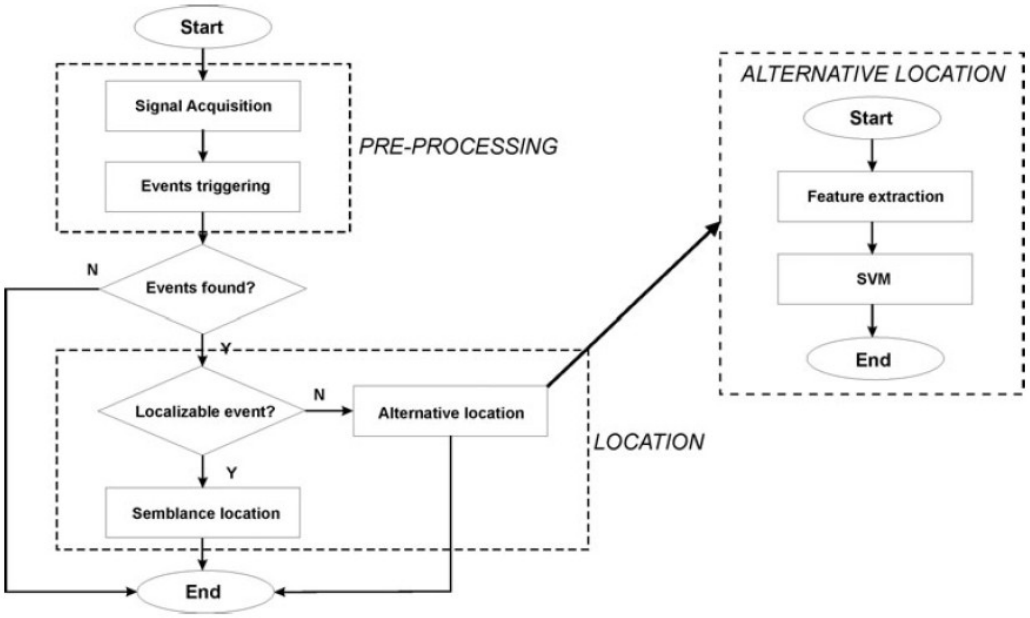


Fig. 6.11. Flow chart of the proposed location system (from Cannata et al., 2011a).

6.2 Characterization of particles shapes by CAMSIZER measurements and cluster algorithms

The shape is a very important feature affecting the properties and the physics behaviours of materials of different natures. In volcanological area, the study of the volcanic ash particles shape, emitted during explosive eruptions, allows to get information about:

- the origin and the fragmentation mechanisms of them;
- the post-eruptive processes, such as the alteration, deposition and transport (Riley et al., 2003);

- the residence time in the atmosphere.

Precisely because of the latter information, the shape parameter is used in some models for the scattering ashes (Scollo et al., 2008). The measurement and quantification of a particle shape are hard challenges, especially when the number of the particles to analyze is high and their size is small (i.e. sub-millimeter), as in the case of volcanic ash. The current methods in volcanology used for quantitative measurements of the particle shape are based on image processing, and are mainly achieved by manual outputs (e.g. investigations under microscope). The innovation of this procedure arises from the use of *CAMSIZER* (www.retsch-technology.com), an instrument developed by the German leader company *Retsch Technology*. This instrument, massively used in the industrial field for the quality control of many types of materials (Lo Castro and Andronico, 2008), permits to obtain very important information both on size and shape parameters of a high number of particles (hundreds of thousands data). Moreover, we used clustering and classification algorithms in order to group particles according to their morphologic characteristics.

6.2.1 Definition of the shape

In literature, the most commonly used definitions are often based on the notion of invariance property of the object shape respect to the basic geometric transformations: translation, rotations, scale factor (Dryden and Mardia, 1998). According to this definition, a set of different numerical "descriptors" are used to identify different shapes (ISO 9276-6, 2008).

Given a specific shape S, it is possible to describe it by a set of measures and properties, called *features*. For example, a shape can be described by its area, or perimeter, or by the number of cavities or picks. More formally, the shape characterization process implies a set of transformations T_i, such that the shape of each transformation can be represented by a set of scalar measures (features) F_k (Costa and Cesar, 2001) with $k = 1, 2, ..., n$ (Fig. 6.12), saved in an array $F = (F_1, F_2, ..., F_n)$.

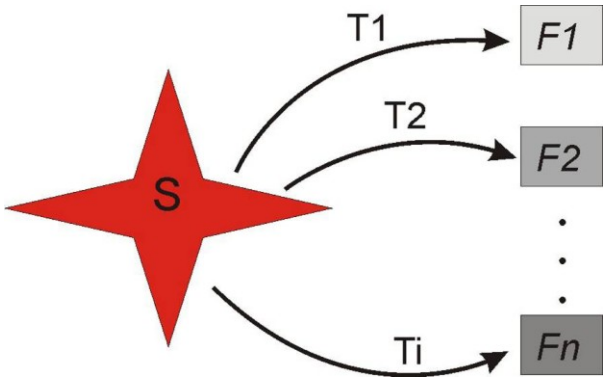


Fig. 6.12. Scheme illustrating the characterization of a generic shape *S* according to a series of features *F* (from Lo Castro et al., 2011a).

Each feature must have a strong discrimination power, and must emphasize the property of interest (for example, if we want to characterize polygons, the feature corresponding to the number of sides will be more significant than the number of cavities). Methods to calculate the shape descriptors can be classified in:

a) *qualitative methods*: refer to the visual appearance of a given particle (e.g. rounded, sub-angular and angular particles), and are generally based on comparative charts (Fig. 6.13).

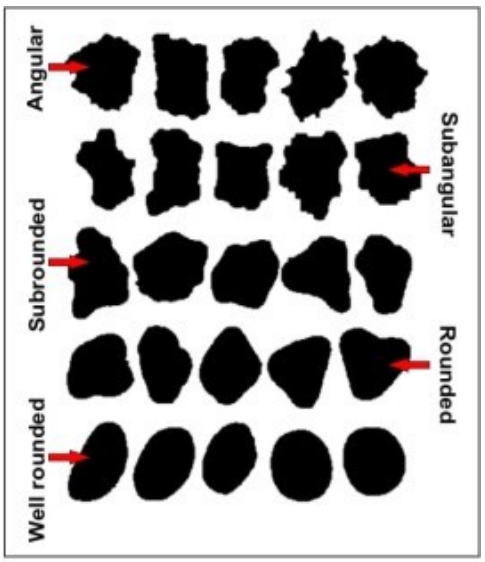


Fig. 6.13. Comparative chart of Russell, Taylor (1937) and Pettijohn (1972) for the qualitative characterization of shape (modified after Muller, 1967), from Lo Castro et al. (2011a).

b) *quantitative methods*: based on numeric values, which can be calculated from images or from physics properties of the particles.

Quantitative methods based on image analysis are used for our scope. The image analysis is a versatile technique, and is applied in a broad range of disciplines. Given a real (tridimensional) particle, the image analysis needs an input device to collect images (generally a camera, or microscope, or scanner). The acquired images will be transformed in bidimensional digital images (the projections of the particles), and given in input to a software, able to read them and to calculate information, such as the dimensional parameters and the shape (Fig. 6.14).

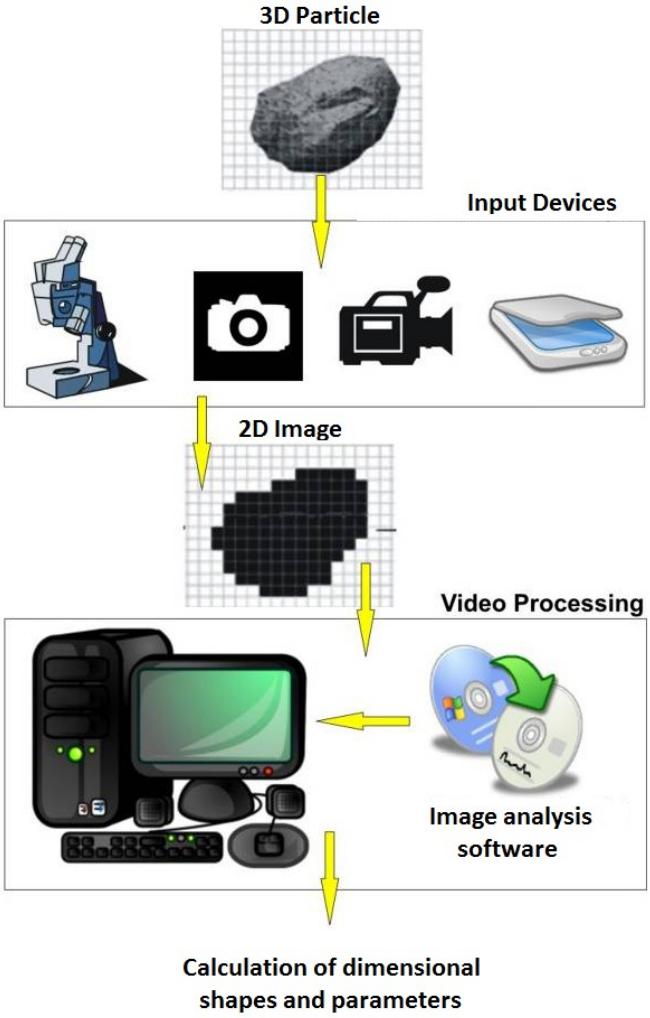


Fig. 6.14. Scheme showing the basic instruments for image analysis (redrawn from Lo Castro et al., 2011a).

There are several tools to perform the image analysis. Microscopic analysis has so far been the reference technique, because allows to directly measure the size and the shape of the particles. However, this manual technique involves many hours of work, and it is subject to human errors. Modern systems allow to more precisely analyze thousands of particles at time. Among them we can distinguish:

a) *Static image analysis*, in which the particles are stationed on a moving treadmill, framed by a camera and a microscope (Fig. 6.15). This method can manage only a limited number of data and, above all, particles are oriented according to their base.
b) *Dynamic image analysis*, where particles fall down at certain velocity v, into a corridor, and are framed by one or more cameras (Fig. 6.15b). Particles are arbitrarily oriented during the fall.

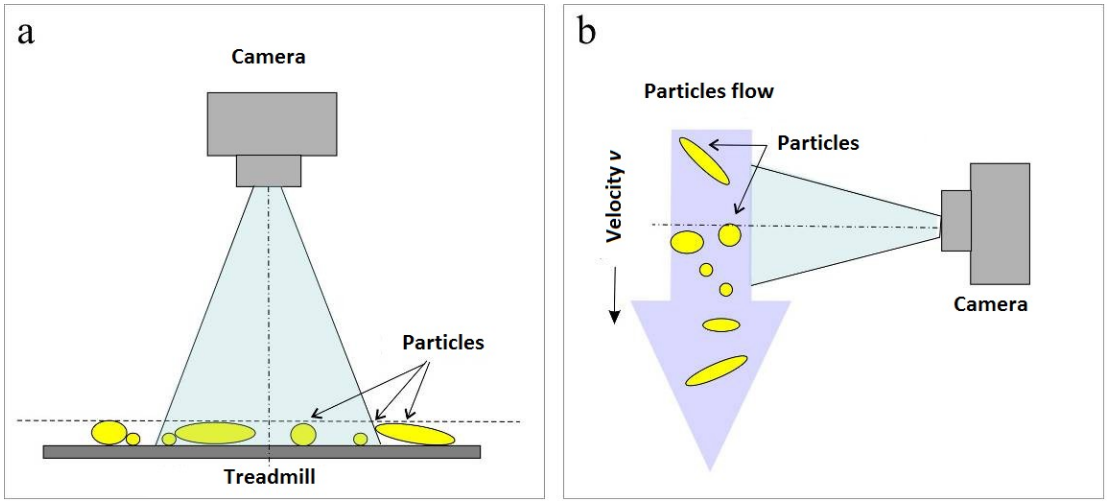


Fig. 6.15. Different Image Analysis methodologies: a) static and b) dynamic (from Lo Castro et al., 2011a).

6.2.2 Methodology

The proposed system, developed in collaboration with Deborah Lo Castro of INGV, bases on measures obtained by *CAMSIZER*, followed by automatic clustering and classification analysis. The complete process is shown on Fig. 6.16).

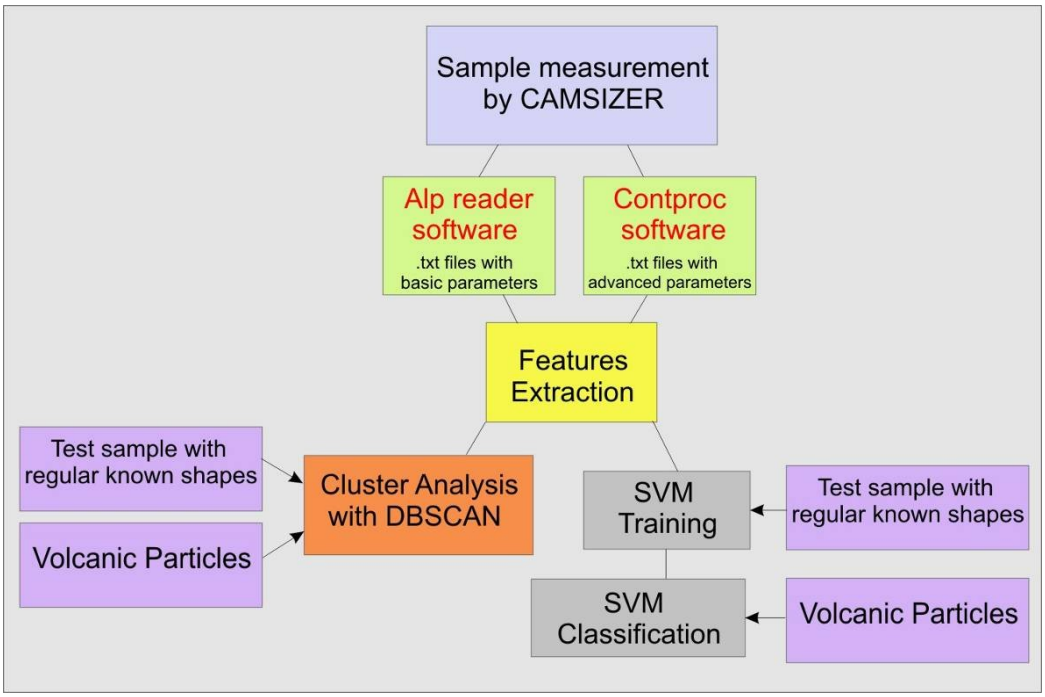


Fig. 6.16. Scheme of the methodology used in the research (from Lo Castro et al., 2011a).

CAMSIZER. In the first step, the *CAMSIZER*, communicating with external software (*Alp-reader* and *Contproc*, provided by *Retsch Technology*), returns a set of output files, relative to dimensional and shape parameters of the chosen sample for the analysis. *CAMSIZER* ® is a laboratory device, created by *Retsch Technology* (www.retsch-technology.com), which dynamically and concurrently analyzes the shape of solid particles of size ranging from 30 μm to 30 mm. The device (Fig. 6.17) is composed by a funnel for inputting samples, a vibrating plate for particles sliding (feeder), ending with a precipice where particles fall down into a monitored chamber. In the chamber each particle, illuminated by a white light, is captured by two cameras, exclusively calibrated for large (*CCD-Basic*) and small (*CCD-Zoom*) size particles (Fig. 6.18). The detected images are the projections of the shadows of each particle. For a more precision on measurements, each projection is scanned in 64 different directions of measure.

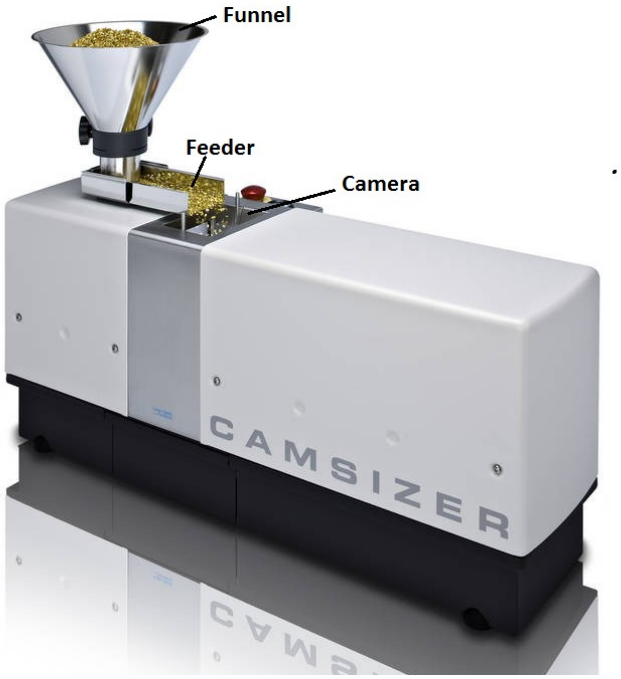


Fig. 6.17. Main components of CAMSIZER (from http://www.retsch-technology.com/rt/products/dynamic-image-analysis/camsizer/function-features/).

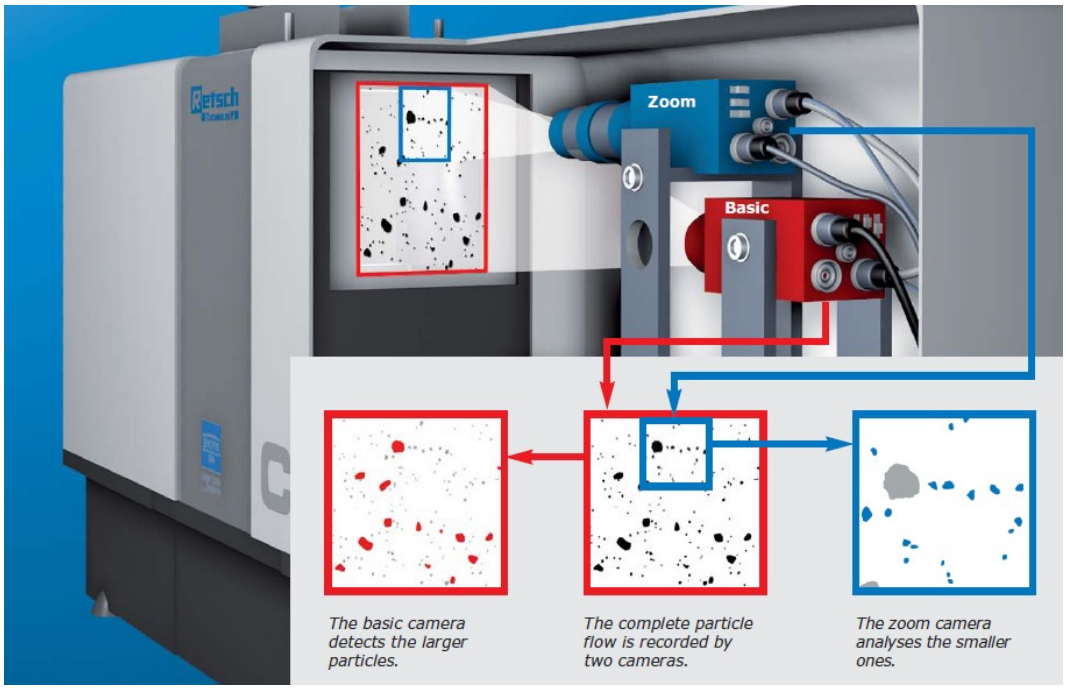


Fig. 6.18. Structure of the *CAMSIZER* camera (from http://www.horiba.com/fileadmin/uploads/Scientific/Documents/PSA/CAMSIZER_brochure.pdf).

Results are stored into the following output files:

- *Raw Data File* (**.rdf*): native file of the *CAMSIZER* software, storing all information about measures.
- *Excel File* (**.xls*): result table for parameters relative to each granulometric class.
- *Alp File* (**.alp*): stores dimensional and shape parameters of each measured particle. It is read from an external software (*Alpreader*).
- *Kon File* (**.kon*): stores information about particles contouring. It is read from a special software (*Contproc*).

Features. A simple method to characterize a particle shape consists on using the ratio between two dimensional measures (x_i and x_j), to obtain the *Conventional Shape Descriptor* (S_{ij}) (Hentschel and Page, 2003):

$$S_{ij} = x_i / x_j \tag{6.6}$$

According to the type of the chosen dimensional measures, the obtained parameter will be sensitive to a specific aspect of the shape. Hentschel and Page (2003) applied cluster analysis on different parameter combinations, in order to identify few parameters to describe particle shape.

From this research they found that the shape, for a range of commercial powders and bulk materials, can be efficiently described by two conventional shape descriptors*: Aspect Ratio* (*AR*) for particle stretch, and the *Form Factor* (*FF*) (Cox, 1927; Kuo et al., 1998) for sphericity and irregular contours. These parameters are normally used in volcanology for the description of the volcanic particles (Riley et al., 2003), and then are extracted from *.alp* file and used for this work. For our experiment, a further parameter, stored on *.kon* files, is used to describe the angularity of a particle.

The following formally explains the chosen parameter:

a) *Aspect Ratio (AR)* is the ratio between the width (xc_{min}) and the height (xFe_{max}) of the particle projection (Fig. 6.19), and describes its stretch degree (Fig. 6.19a):

$$xc_{\min} / xFe_{\max} \tag{6.7}$$

Globular particles have an *AR* value near to 1, while the *AR* value of a stretched particles is < 1;

b) *Form Factor (FF)* is defined as the ratio between the area of the projection of a given particle (*A*) and its perimeter (*P*), and refers to the degree of circularity (Fig. 6.19b):

$$\frac{4\pi A}{P^2} \tag{6.8}$$

In 2D, a perfect circle has *FF* = 1, while irregular shapes have *FF* < 1, because an irregular shapes have a greater perimeter.

c) Angularity refers to the particle contour and to its irregularity. From *.kon* file we considered the $E_{polygon}$ parameter, that measures the mean value of the vertices of a polygon (convex angles α), defined as the polygon which best fits with a given contour, multiplied by the relative height *h* (Fig. 6.19c):

$$E_{polygon} = \frac{\sum_i h_i \frac{\pi - \alpha_i}{\pi}}{\sum_i h_i} \tag{6.9}$$

This value depends on the number of vertices of the polygon: a perfect circle has $E_{polygon} = 0$, while a sharp contour has $E_{polygon} = 1$.

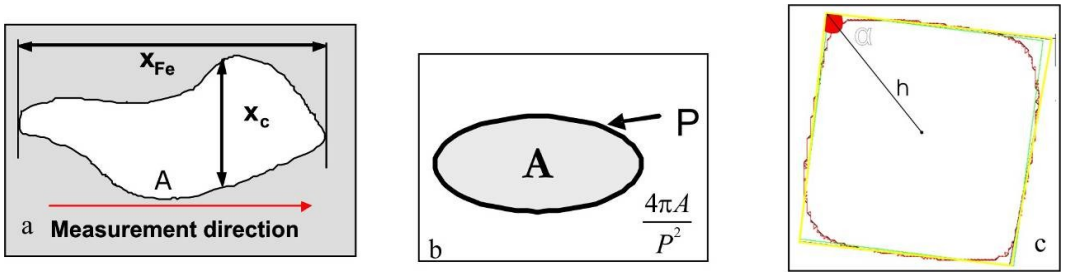


Fig. 6.19. Schemes describing the shape parameters used: a) aspect ratio; b) sphericity; c) angularity.

6.2.3 Data analysis

The step of calibration was performed using only well-known shapes, such as spheres, cubes and cylinders, created by a simulation software provided by the *CAMSIZER* (Fig. 6.20).

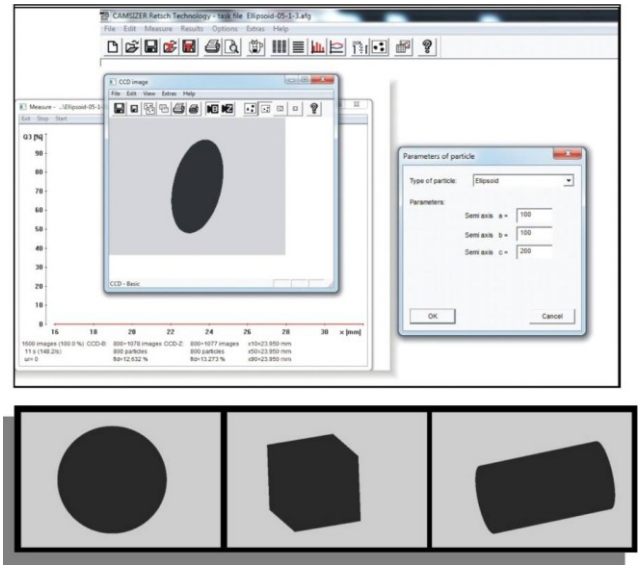


Fig. 6.20. Graphical interface of the simulation software of *CAMSIZER* (on top) and simulated shapes used in the experiments (bottom).

The measurements will be done both for the simulated shapes and for the real samples. The results, saved on the *.alp* and *.kon* files, were used to extract the 3 parameters we need to describe the shapes and the relative *labels* or class of belonging: sphere (*S*), cubes (*Cu*), and cylinders (*Ci*) (Table 5.2). The selected data was given in input to the *PyDBSCAN* software (Cassisi et al., 2011a).

The real samples (Fig. 6.21) are 1.2 × 1.2 × 1.2 cm standard rubber cubes, plastic spheres (diameter = 2 cm) and wood cylinders (length = 4 cm and diameter = 1 cm). Samples were analyzed with CAMSIZER, by performing tests on a single sample and on a set of samples.

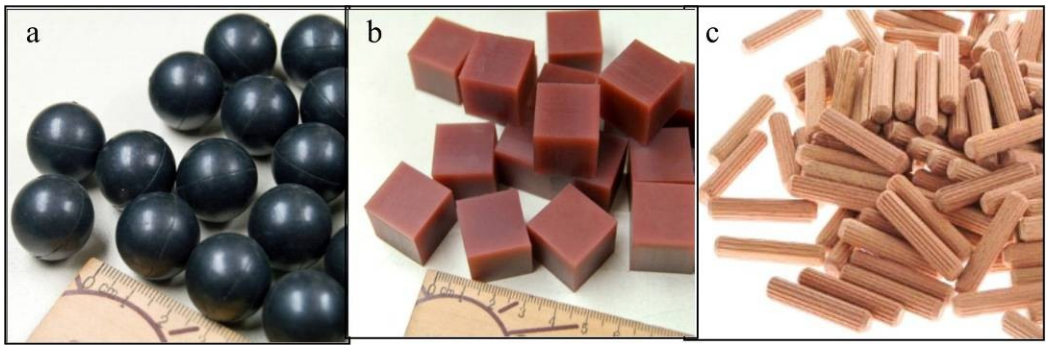


Fig. 6.21. Material used in the experiments: a) plastic spheres; b) standard rubber cubes; c) wood cylinders.

	b/l	SPHT	Epol
S	0.997	0.9971	0.0955
S	0.997	0.9979	0.0958
S	0.997	0.9961	0.0872
S	0.9969	0.9996	0.0871
S	0.997	10.011	0.0872
S	0.9971	0.9988	0.0954
…	…	…	…
Ci	0.4988	0.7682	0.2669
Ci	0.4987	0.7647	0.3759
Ci	0.4983	0.753	0.2591
Ci	0.4989	0.7947	0.2466
Ci	0.4984	0.7862	0.1777
Ci	0.4995	0.726	0.4422
…	…	…	…
Cu	0.7744	0.8645	0.3561
Cu	0.8413	0.9152	0.3356
Cu	0.6524	0.8538	0.4431
Cu	0.6499	0.779	0.4456
Cu	0.748	0.9173	0.3731
Cu	0.6625	0.802	0.4328
Cu	0.7045	0.8648	0.4042
…	…	…	…

Table 6.2. Input data interface for the clustering software. Labels of the shape typologies are shown in the first column (S=sphere; Ci=cylinders; Cu=Cubes). The other 3 columns report the shape parameters (b/l=aspect ratio; SPHT= sphericity; Epol= angularity).

6.2.4 Results

Fig. 6.22 shows a 3D feature space (b/l, SPHT, Epol), 3 different clouds of density and then 3 clusters for the density-based clustering definition (see Section 5.3.3), corresponding to the 3 different shapes:

- Spheres are grouped in a small region where b/l, SPHT, and Epol are all equal to 1.
- Cylinders have b/l value between 0.5 and 0.9, with a higher percentage around 0.6, because are more stretched than spheres. The SPHT value is practically constant, and ranges from 0.75 to 0.85, while Epol value ranges in a more large interval starting from 0.1 (more rounded particles) to 0.5 (more angular particles). Such a difference is due to the various directions and the relative projections of the cylinders during the "fall" into the measurement chamber of the instrument (Fig. 6.22b).
- Cubes present high b/l and SPHT values, ranging from 0.6 to 0.9, and a quasi-constant value for Epol (≈ 0.4), because any projections of the cube in its fall always maintains a certain angularity.

Using the same sample dataset of the clustering process, we tested a classification model based on *SVM* to compute the maximum marginal hyperplane (see Section 4.1.5) separating the obtained clusters. Fig. 6.23 shows a 2D projection of the dataset where the 3 cluster are divided into 3 well-defined areas.
To check the quality of the classification model we tested the system by using a new dataset representing real spheres (cluster 1), cubes (cluster 2) and cylinders (cluster 3), and storing the classification results into a confusion matrix (see Section 6.1.4), showed in Fig. 6.24.
In this case 619 spheres (cluster 1) are well classified, while only 10 are classified as cubes (cluster 2). Same thing for the cubes, where 365 sample are classified as cubes (cluster 2), while 78 as spheres (cluster 1). For cylinders we observed that 111 samples are well classified as cylinders (cluster 3), while 11 samples are labeled as spheres (cluster 1). From this matrix we can calculate a total error rate about 0.08 %.

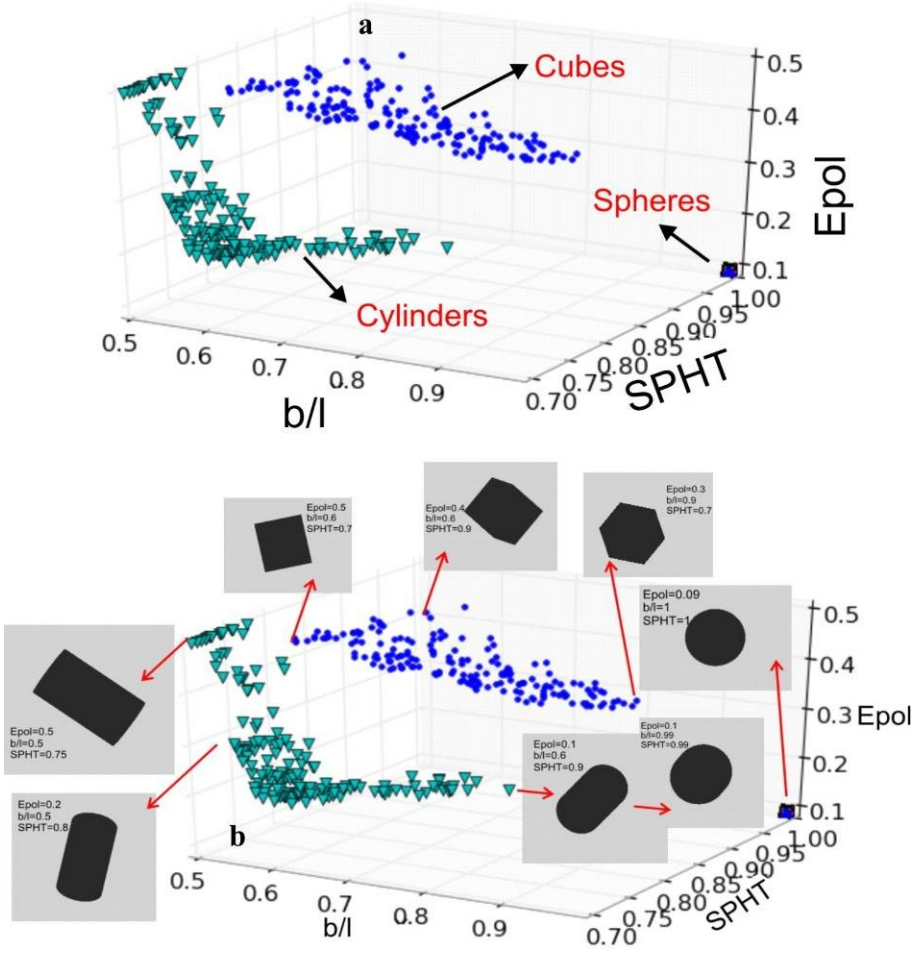


Fig. 6.22. a) Density clusters showing the 3 different analysed shapes in a 3D space defined by the descriptive features; b) the same diagram a with the different shape typologies.

6.2.5 Future works

This technique allows good classification for elementary shapes (spheres, cubes, cylinders). The next step of this research will be aimed to the realization of new experiments about volcanic materials (lapillus and ash) which, unlike the materials described herein, are composed by very irregular shapes and therefore more difficult to characterize. Preliminary tests highlight the exigency of using more particular shape descriptors, that can be based, for example, on the conversion of a two-dimensional shape to a one-dimensional "time series" (Want et al., 2008).

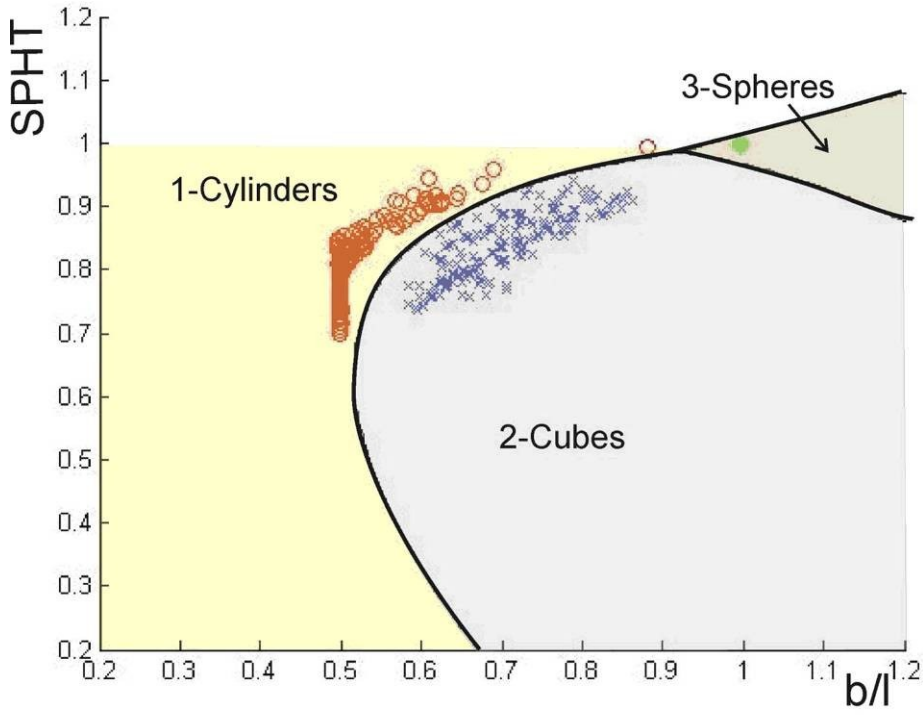
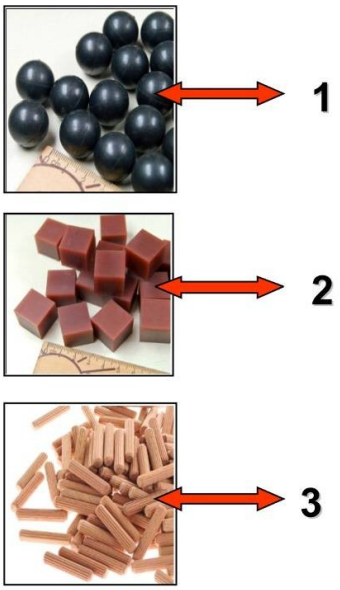


Fig. 6.23. 2D diagram showing the optimal hyperplane separating the 3 clusters, obtained by the SVM analysis.



		PREDICTED		
		Cluster 1	Cluster 2	Cluster 3
ACTUAL	Cluster 1	619	10	0
	Cluster 2	78	365	0
	Cluster 3	11	0	111

Fig. 6.24. Real materials used during the testing phase (on the left, with labels) and confusion matrix (on the right). The cluster 1 is relative to spheres, the cluster 2 to cubes, and the cluster 3 to cylinders.

6.3 Motif discovery on seismic amplitude time series: the case study of Mt. Etna 2011 eruptive activity

Mt. Etna volcano (Italy) is one of the most active volcanoes in the world. The summit area is currently characterized by four active craters: Voragine, Bocca Nuova, South-East Crater and North-East Crater (hereafter referred to as VOR, BN, SEC and NEC, respectively; see Fig. 5.1). Over time, the summit activity has consisted of persistent degassing and different, and sometimes coexistent, eruptive types also from its flanks such as: phreatic, phreato-magmatic, strombolian explosions, lava fountaining, and lava effusion (e.g., Chester et al., 1985; Branca and Del Carlo, 2005). After the 2008-2009 eruption at the volcano's eastern flank (Cannata et al., 2011b), the activity resumed in 2010 with minor explosive activities from BN, NEC and SEC (Spina et al., 2012), and continued in 2011 with several lava fountaining episodes taking place at a new crater, opened to the east of SEC and named as "new SEC". Each of the lava fountaining episodes showed an initial strombolian phase and lava effusion, that emplaced on the upper Valle del Bove (e.g., Bonaccorso et al., 2011a; Calvari et al., 2011). The paroxysmal behaviour of such short-lasting violent phases is not uncommon at Etna and during recent decades it has become more and more frequent. Indeed, between 1900 and 1970, about 30 paroxysmal eruptive episodes occurred at the summit craters, while there have been almost 200 since then (Behncke and Neri, 2003).

The geophysical surveillance of active volcanoes is routinely carried out mainly by observing the patterns of seismic activity and ground deformations (e.g. Scarpa and Gasparini, 1996). Commonly, seismic unrest in the form of earthquakes and tremor have almost always preceded and/or accompanied volcanic unrest phases at different types of volcanoes (McNutt, 2000). Seismic activity is considered a critical indicator, and often a reliable short- to midterm (days to weeks) eruption forecaster, and a marker of the level and evolution of ongoing volcanic activity (McNutt, 2000).

We can distinguish two different groups of seismic signals in volcanic areas—those associated with shear failures in the volcanic edifice, which are called volcano-tectonic (VT) or high-frequency earthquakes, and the seismic signals associated with fluid processes (Lahr et al., 1994; Chouet,

1996; McNutt, 2005). These include long period (LP) events, volcanic tremor, which share the same spectral components, and very long period (VLP) events, characterized by lower frequency content. The main difference between LP
events and volcanic tremors is their duration. Similarly to earthquakes, LP event duration is in the order of seconds whereas volcanic tremor can last from minutes to months. Alparone et al. (2003) qualitatively observed that some recurrent patterns can be recognized in the seismic amplitude time series. Specific volcano states, for example lava fountain activity, may have a recognizable amplitude pattern.

The task of extracting previously unknown recurrent patterns (also called motifs) from available data constitutes a crucial step in geophysical time series analysis. Many algorithms for efficiently mining motifs have been proposed, especially in bioinformatics (Lawrence et al., 1993; Bailey and Elkan, 1995; Pevzner and Sze, 2000; Tompa and Buhler, 2001). Since naïve algorithms are computationally time-expensive, as quadratic in the length of time series, most researchers have abandoned searching for exact solution methods, such as the cross-correlation approach, and have focused on investigating approximated solutions to the problem. Many of them have made use of data dimensionality reduction.

In this work, we apply an exact time series motif discovery technique to explore recurrent patterns within the seismic amplitude time series of Mt. Etna 2011 periodic eruptive activity (1 January – 16 November 2011). To this end, the seismic amplitude time series were computed using a root mean square (RMS), which provides information on the volcano states and/or external seismic sources. Although belonging to exact solution methods, this technique allowed us to reduce computation time in finding motifs in the investigated time series.

6.3.1 Data analysis

On Mt Etna, the broadband permanent seismic network consists of 32 stations equipped with broadband (40 s cutoff period), three-component Trillium seismometers (Nanometrics) acquiring at a sampling rate of 100 Hz in real-time. The signal recorded by the vertical component of the reference station EBEL (Fig. 6.1) from 1 January–16 November 2011 was

filtered in the 0.5–5.0 Hz band, which comprises most of the volcanic tremor energy of Mt Etna (Cannata et al. 2008, 2010). RMS was calculated on 10-minute-long moving windows, chosen as a compromise between acceptable time resolution and the fairly small time series (Fig. 6.25).

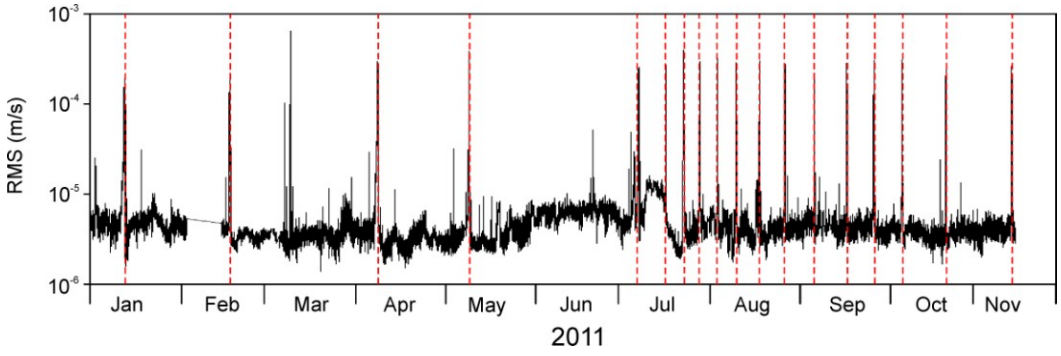


Fig. 6.25. RMS of seismic signal recorded by the vertical component of EBEL station and filtered in the band 0.5-5.0 Hz. The vertical dashed red lines indicate the episodes of lava fountains (from Cassisi et al., 2012a).

6.3.2 Motif discovery theory

The brute-force search algorithm is a simple solution that performs a sequential scan of the database. Suppose a simple case, where we have a time series database (*TSDB)* of L time series, each of length m. At each step, the current analyzed time series T_i ($0 < i \leq L$) is compared with all the following time series T_{i+1}, …, T_L (Fig. 6.26) by computing distances or similarity coefficients (e.g. cross-correlation) between time series, whether they use metric or non-metric measures such as *Euclidean* (Lie Hetland, 2004) or *Dynamic Time Warping* (Berndt and Clifford, 1994), respectively. For each comparison, if the distance (or similarity) between T_i with a T_j ($i < j \leq L$) satisfies some user-defined threshold, then the pair (T_i, T_j) is candidate to be a motif. This step is repeated for all T_i. Each step will perform L-1 comparisons at step 1, L-2 at step 2, until to 1 comparison at step L-1. Then, the total number of comparisons can be calculated as ($L + L - 1 + \ldots + 1$) = $L(L - 1) / 2$. This value is quadratic respect to the *TSDB* size. In computer science this computation is referred to achieve a complexity $O(L^2)$. Moreover, each comparison makes a number of operations that is proportional to m, for a total computational complexity of $O(mL^2)$. A typical application of such a method makes use of the cross-correlation to

quantify similarity between time series (e.g. Brown et al., 2008). Since this kind of algorithm is computationally time-expensive, some researchers have abandoned searching for exact solution methods, such as the cross-correlation approach, and have focused on investigating approximated solutions to the problem. Moreover, over the last years, novel time series algorithms, dealing with high dimensional data, have been developed to optimize the computational time even using the raw representation of data. One of such algorithms is the *Mueen-Keogh* (*MK*) algorithm (Mueen et al., 2009), that was here applied to perform exact motif discovery in seismic RMS time series. Such an algorithm makes use of Euclidean distance to carry out comparisons between time series, instead of more complex and accurate measurements used on time series classification/clustering, such as *DTW*. Recent works showed its effectiveness on various domains (Ding et al., 2008), especially when datasets expand and the difference between the two measures rapidly decreases (Shieh and Keogh, 2008).

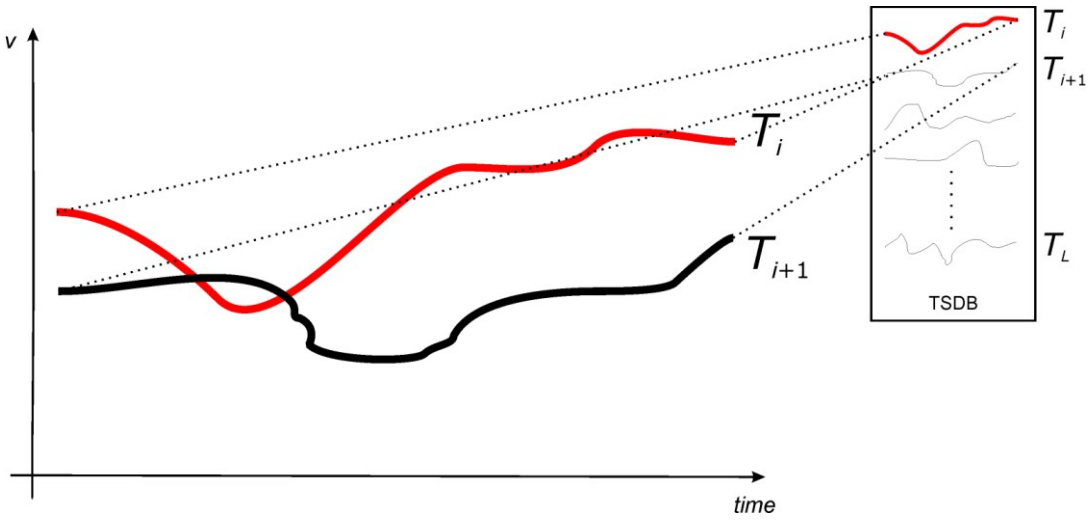


Fig. 6.26. At step i ($0 < i \leq L$), if the distance between T_i with a T_j ($i < j \leq L$; in this figure $j = i+1$) is less than some user-defined threshold, then the pair (T_i, T_j) is candidate to be a motif. This step is repeated for all T_i. The x-axis is relative to time, and the y-axis to a generic time dependent variable (indicated by v).

Before presenting the algorithm, we list some basic notations and definitions of the analysis method.

Definition 6.3.1. A *Time Series* is a sequence $T=(t_1, t_2,\ldots, t_m)$, which is an ordered set of m real valued numbers;

Definition 6.3.2. A *Time Series Database* (D) is an unordered set of time series, possibly of different lengths;

Definition 6.3.3. The *Time Series Motif* of D is a pair of time series $\{T_i, T_j\}$ in D, which is the most similar among all possible pairs. More formally, $\forall a,b,i,j$ the pair $\{T_i, T_j\}$ is the motif if $d(T_i,T_j) \le d(T_a,T_b)$, $i \neq j$ and $a \neq b$;

Definition 6.3.4. The k^{th}*-Time Series motif* is the k^{th} most similar pair in the database D. The pair $\{T_i, T_j\}$ is the k^{th} motif if there a set S of pairs of time series of size exactly k-1 exists, such that $\forall T_d \in D \quad \{T_i, T_d\} \notin S$ and $\{T_j, T_d\} \notin S$, and $\forall \{T_x, T_y\} \in S \quad \{T_a, T_b\} \notin S \quad d(T_x,T_y) \le d(T_i,T_j) \le d(T_a,T_b)$;

Definition 6.3.5. The *Range motif* with range r is the maximal set of time series with the property that the maximum distance between them is less than $2r$. More formally, S is a range motif with range r if $\forall \{T_x, T_y\} \in S$, $d(T_x, T_y) \le 2r$ and $\forall T_d \in D - S \quad \forall T_y \in S \;\; d(T_d,T_y) > 2r$;

Definition 6.3.6. A *subsequence* of length n of a time series $T=(t_1, t_2, \ldots, t_m)$ (with $n \ll m$) is a time series $T_{i,n}=(t_1, t_{i+1}, \ldots, t_{i+n-1})$ for $1 \le i \le m-n+1$;

Definition 6.3.7. The *Subsequence Motif* is a pair of subsequences $\{T_{i,n}, T_{j,n}\}$ of a long time series T that are most similar. More formally, $\forall a,b,i,j$ the pair $\{T_{i,n}, T_{j,n}\}$ is the subsequence motif if $d(T_{i,n}, T_{j,n}) \le d(T_{a,n}, T_{b,n})$, $|i-j| \ge w$ and $|a-b| \ge w$ for $w > 0$.

Let us explain the idea behind the algorithm. The first improvement for the brute-force method is the application of the *early abandoning* idea (Fig. 6.27). In order to search the nearest neighbour of a time series Q, it needs to compute the point-to-point distance from Q for each time series in the database. At each step, if we know the nearest neighbour distance from Q (or the *best-so-far*), we can stop computation as soon as the current distance

exceeds this value, and skip scanning the next time series. Initially the algorithm assumes the *best-so-far* value to be infinity.

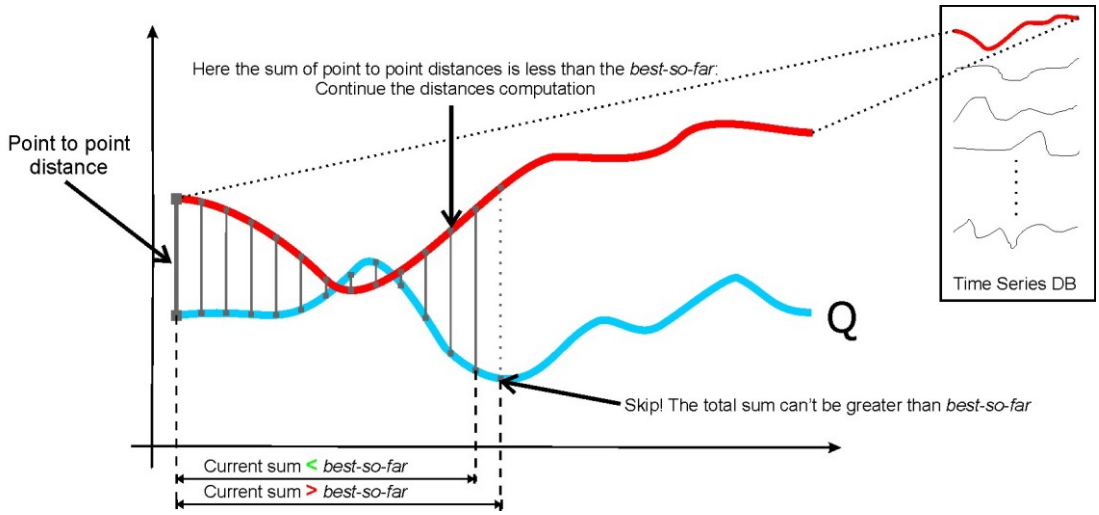


Fig. 6.27. *Q* is the time series query. To find the *Q* nearest neighbor, we compare *Q* with each time series in the database. At each step, with the early abandoning method, once the sum of accumulated distances (gray lines) exceeds the *best-so-far* value, the full Euclidean distance surely exceeds this threshold, and we can skip scanning the next time series (redrawn from Mueen et al., 2009).

The second improvement regards the creation of a *database index* (see also Section 2.2). In this case, the indexing consists in choosing a random object as "pivot" (or *reference point*; hereafter referred to as O_1; Fig. 6.28) and linearly ordering the space $\{O_1, O_2, O_3, \ldots, O_n\}$ by using the distance from it, $dist'(O_i) = dist(O_1, O_i)$ (with $0 < i \leq n$), considered as sorting key. We then record distances between adjacent pairs: $\{dist'(O_2) - dist'(O_1)\}$, $\{dist'(O_3) - dist'(O_2)\}$, …, $\{dist'(O_n) - dist'(O_{n-1})\}$. This transformation maintains two useful properties: (i) even if the saved distances cannot be true, they are always lower than the true distances, thanks to the triangle inequality property (Fig. 6.29); and (ii) if two objects are close in the original space, they must be close in the new projection too (but the contrary is not always true).

Thanks to these two improvements, *MK* algorithm is still worst case quadratic, but generally it reduces the computational time by three orders of magnitude with respect to the classical brute-force method (Mueen et al., 2009).

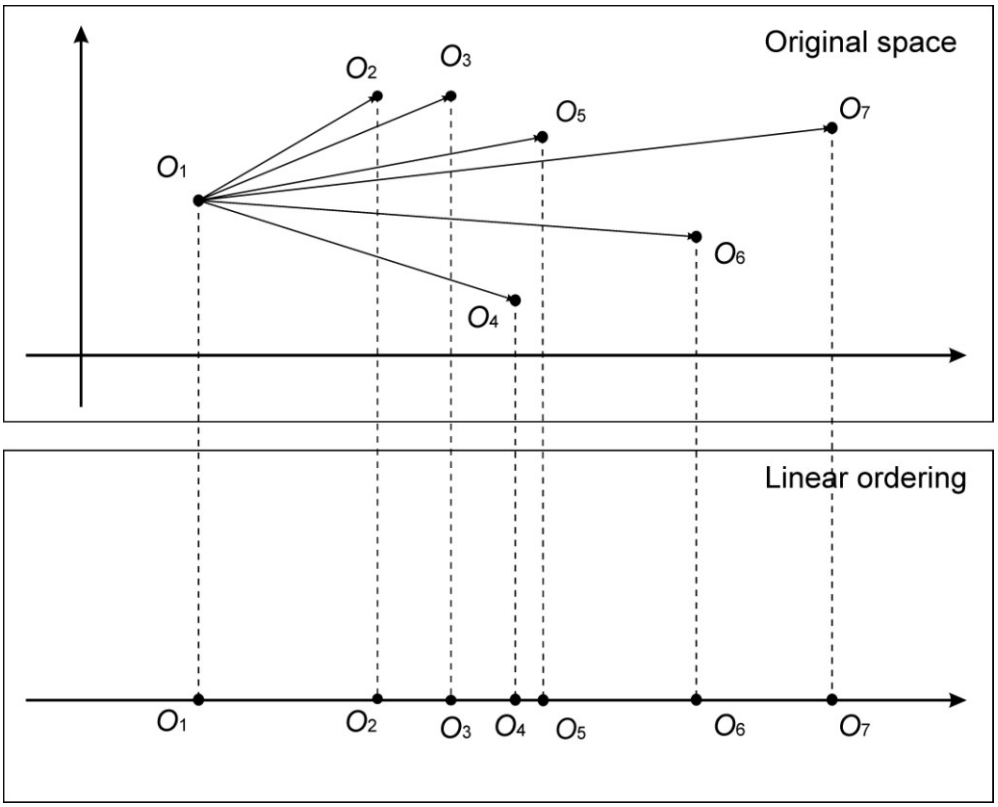


Fig. 6.28. A simple indexing example in 2D space. Objects can be arranged in a one-dimensional representation by measuring their distance from a randomly chosen point: the *reference point* (O_1 in this case) (redrawn from Mueen et al., 2009).

We now provide an example of the analytical procedure to show how the linear ordering, combined with the *best-so-far*, permits to make space pruning to enhance motif search. At first we choose O_1 as the current reference point, then we set the *best-so-far* value to the distance from its nearest neighbour O_2, scan across the linear ordering (Fig. 6.28), measure the true distances between adjacent pairs, and update them. As soon as a pair of objects having distance value less than the *best-so-far* is found, we update the *best-so-far* to this last value. Successively, if the found value does not match the true *best-so-far*, scanning again the linear ordering, we can prune off all pairs of objects with mutual distance greater than the resulting *best-so-far* value. In fact, the triangle inequality property ensures that these are lower than the true distances, thus they cannot be the candidates. Using a simple heuristic to find good reference points, and thus making multiple pruning rounds with multiple reference points,

Mueen et al. (2009) demonstrated how this technique speeds-up the computation time.

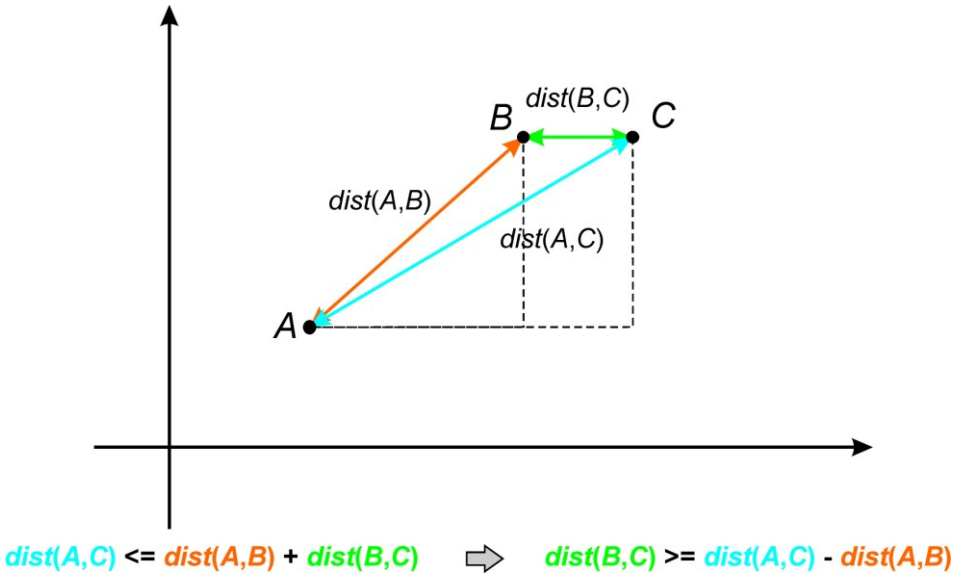


Fig. 6.29. Assume working with metric distance functions as the Euclidean distance. If we refer the *A*, *B* and *C* points to O_1, O_2 and O_3 of Fig. 5.29 respectively, we can use the triangle inequality property to assert that dist(O_2,O_3) is greater than the value {O_2,O_3} = dist(O_1,O_3) - dist(O_1,O_2) (saved on the linear ordering).

The power of *early-abandoning* method can be related to the familiar effect of the birthday paradox in probability theory (Brink, 2012), for which in a dataset consisting of 23 people, the chance of any two people sharing a birthday is 50.7%, while 99% probability can be reached in a set of just 57 people. There are so many possible ways for pairs to be similar, that it is reasonable to think that the algorithm can find a very low *best-so-far* rapidly.

5.3.3 Results

The used Matlab tool developed by Mueen et al. (2009; http://www.cs.ucr.edu/~eamonn/exact_motif/) allows applying the described technique to investigate likely subsequence motifs into a single time series. In order to run the tool we need 5 input values:

- *m*, the length of the time series;
- *R*, the number of reference points to use;

- n, the length of the searched motif;
- X, the coefficient used for motif range r (default X=2 in definition 5.3.5), in this case r corresponds to the *best-so-far*;
- K, the number of desired groups of motifs**.** The k^{th} group corresponds to the k^{th} subsequence motif (see definitions 4 and **7**), combined with all subsequences distant from it at most Xr (see definition 5).

Since the analyzed time series of seismic RMS starts from 00:00 on 1 January 2011 to 17:40 on 16 November 2011, with a 10 minutes time step, we theoretically deal with a time series of m = [6*24*(31+28+31+30+31+30+31+31+30+31+15)] + [6*17] + [4] = 46042. However, since data present missing values, we consider the real input number of points m = 43103. The parameter R was fixed to 10, as suggested by Mueen et al. (2009).

In order to choose the other three parameters we run experiments by systematically changing their values within defined ranges. For each experiment, and then for each n, X, K combination, we count the number of subsequences in each group and evaluated the similarity among them by using another similarity measure, the average cross correlation (indicated by ACC; Figs. 6.30-6.32).

The parameter n was fixed to 50, 100 and 150, roughly corresponding to 8, 17, and 25 hours. Such a range was chosen on the basis of the duration of the "seismo-volcanic phenomena" of interest. Indeed, the investigated period was characterized by lava fountains, that last from a few hours to a couple of days (taking into account also the strombolian phases preceding and following the lava fountain phase). As shown in Figs. 6.30-6.32, variations in n scarcely affect both number of subsequences and ACC. The analysis was focused on experiments having n = 100, because the most significant volcanic tremor amplitude changes during the analyzed lava fountains generally take place in time windows shorter than 17 hours. We explored X values fixed to 2, 3 and 4 because, already in this small range, the number of significant motif groups (i.e. those having high ACC) widely changed. Indeed, for $X > 4$ the method found a few groups including a very large number of subsequences (with relatively low ACC). Among the considered values for X, the choice of the optimum one is very delicate, and, similarly to the cross correlation threshold used to classify

earthquakes into families of similar events (e.g. Ferretti et al., 2005), there is no objective means to determine the right value. Such a threshold depends on how similar the motifs, belonging to a given group, are needed to be for a particular application. There is no similarity threshold value that has a general validity. We chose X=3, because it permits to obtain a good compromise between the number of groups, and the number of subsequences with fairly high ACC within each group (number of subsequences < 20, and ACC generally higher than 0.5). The last parameter is K, whose maximum value was chosen equal to 20, because the MK algorithm does not produce any motifs beyond this limit in all the performed experiments. Considering the plot with n=100 and X=3 (Fig. 6.31), we chose to take into account the first K=8 groups. Indeed, in correspondence to such a value the plot shows both the maximum downward slope of the ACC line (left plot) and the maximum upward slope of the subsequence number line (right plot). Once the values of the 5 input parameters were fixed, the MK algorithm was applied on the seismic RMS time series. Figures 6.33-6.34 report the 8 groups of motifs with their time locations within the original RMS series. The first 2 groups are related to the initial phases of the 2011 lava fountain episodes (Fig. 6.33), with group 1 showing quicker increases (~5 hours on average) in the volcanic tremor amplitude than group 2 (~10 hours on average). Groups 3 and 4 provide information on the final phases of the lava fountain episodes. These are characterized by volcanic tremor amplitude decreasing trends, which, also in this case, can be more or less rapid (motifs 3 and 4, ~30 and 60 minutes on average, respectively; Fig. 6.33). An example of seismic signal recorded during a lava fountain taking place on 10 April 2011, together with RMS time series calculated on 10-minute-long moving windows, is reported in Fig. 6.35. Groups 5 and 6 are composed of portions of RMS time series typified by very sharp increases and decreases (from seconds to a few minutes; Fig. 6.34), respectively. These sudden changes result from the occurrence of VT earthquakes clearly recorded at EBEL station (examples are reported in Fig. 6.36). Group 7 includes motifs of gradual decreases in the seismic RMS, taking place roughly in 5-10 hours, associated with decreases in the volcanic tremor amplitude, and sometimes related to waning phases of the strombolian activity (Figs. 6.34 and 6.37a,b).

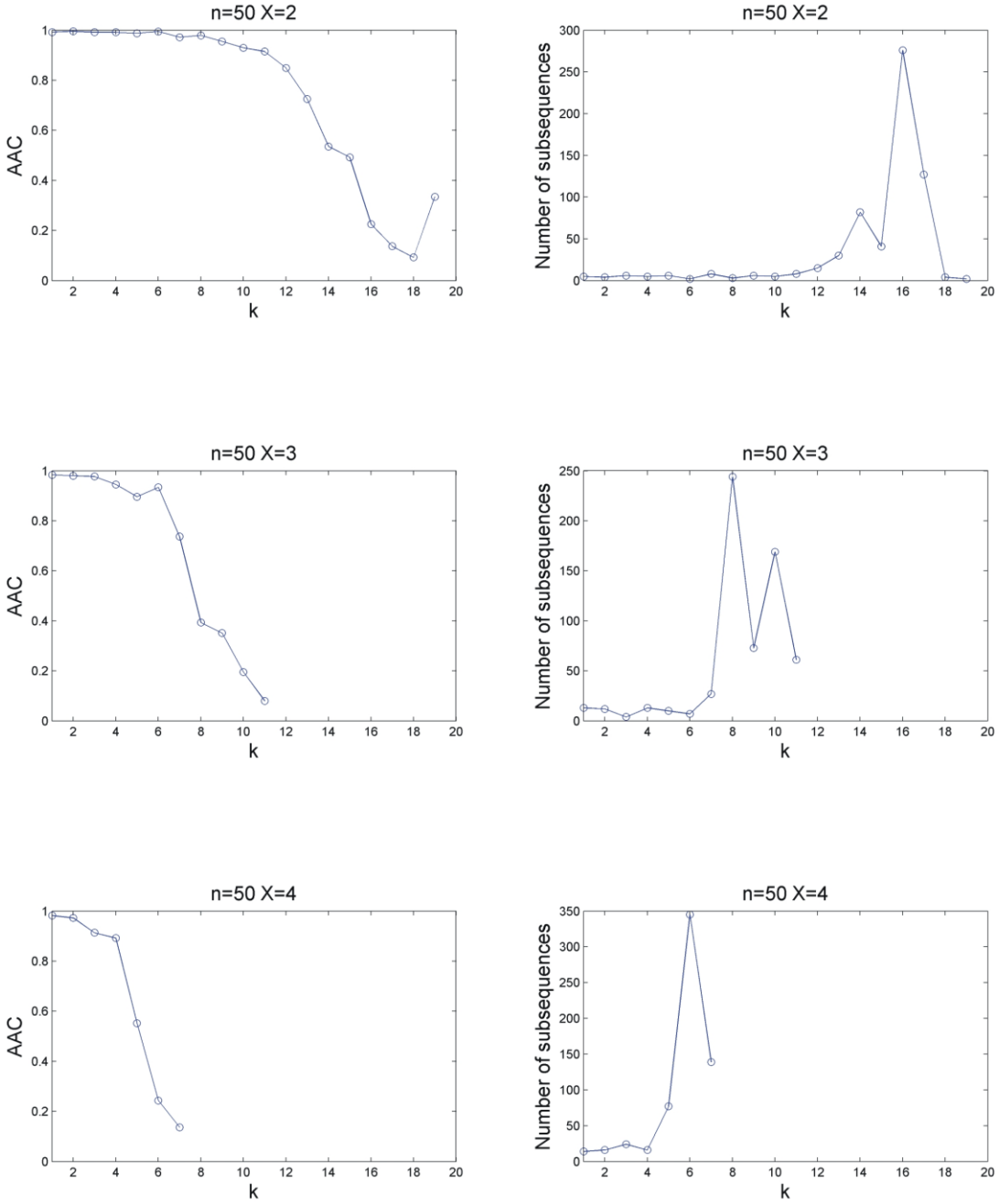


Fig. 6.30. On the left column the average correlation coefficient (ACC) for each group of motifs of size n = 50. On the right column the relative number of subsequences within each group. From top to bottom we show the experiments using X varying from 2 to 4. Points in the plots related to k values, producing no results (no motifs found), are missing.

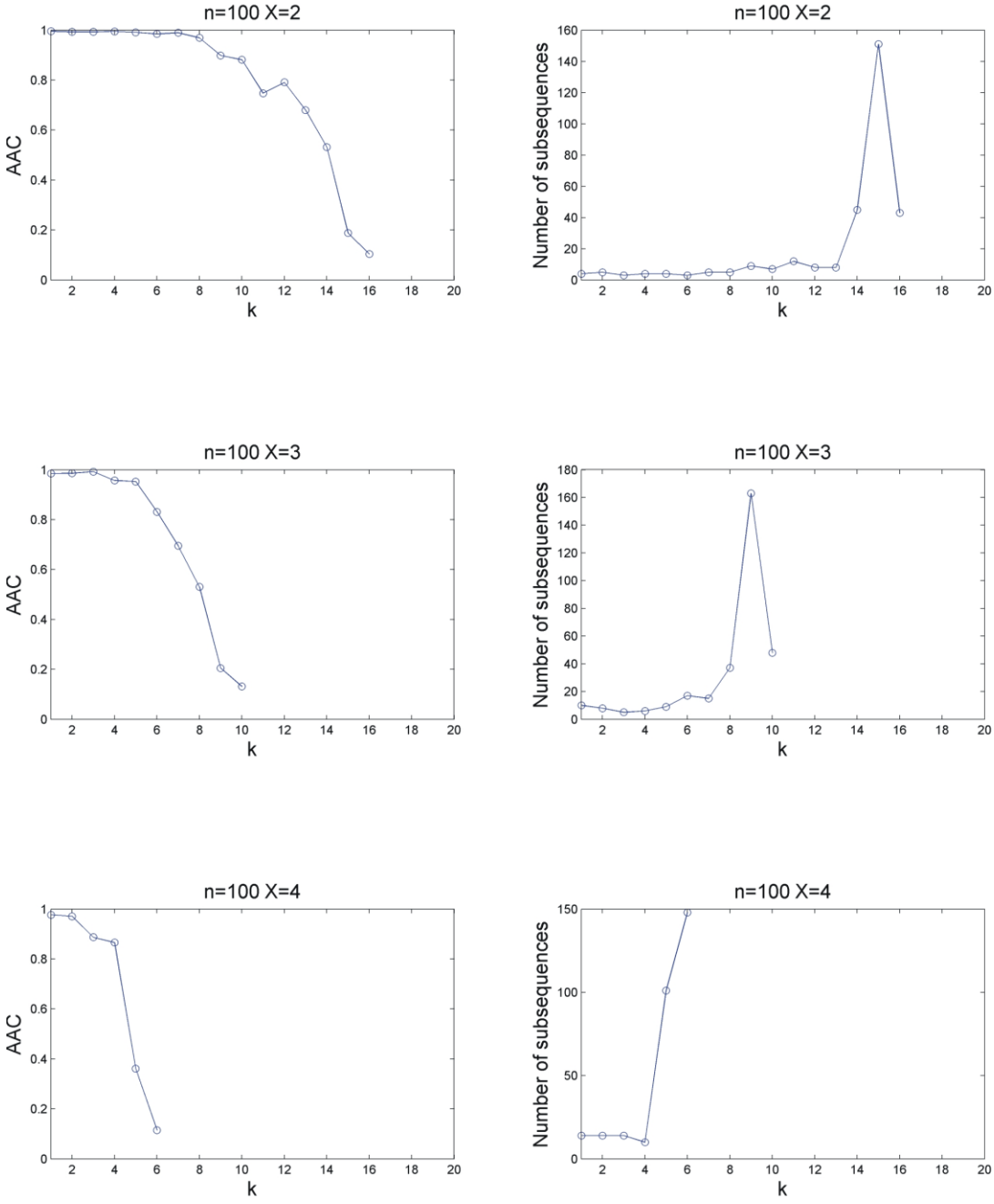


Fig. 6.31. On the left column the average correlation coefficient (ACC) for each group of motifs of size n = 100. On the right column the relative number of subsequences within each group. From top to bottom we show the experiments using X varying from 2 to 4. Points in the plots related to k values, producing no results (no motifs found), are missing.

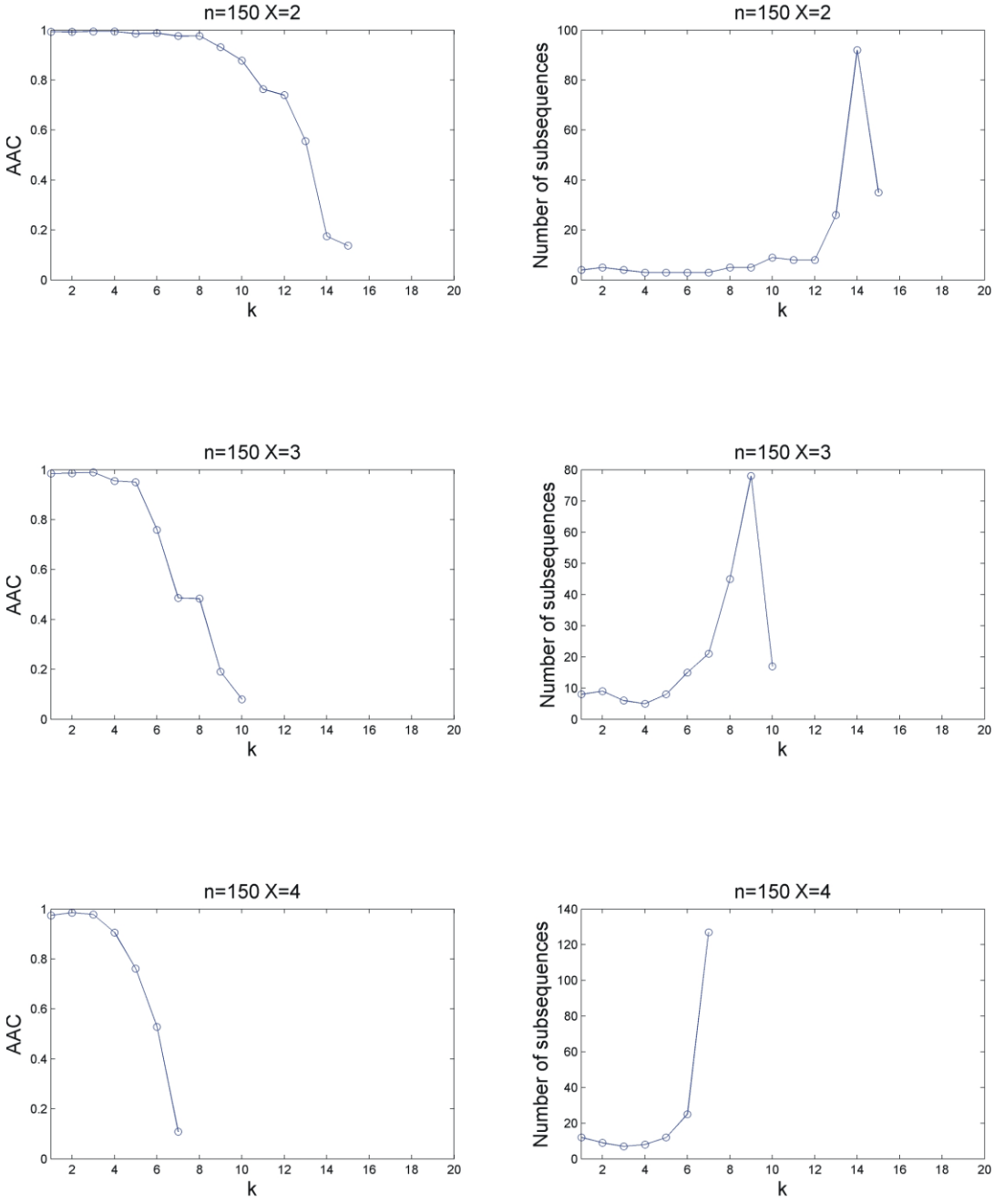


Fig. 6.32. On the left column the average correlation coefficient (ACC) for each group of motifs of size n = 150. On the right column the relative number of subsequences within each group. From top to bottom we show the experiments using X varying from 2 to 4. Points in the plots related to k values, producing no results (no motifs found), are missing.

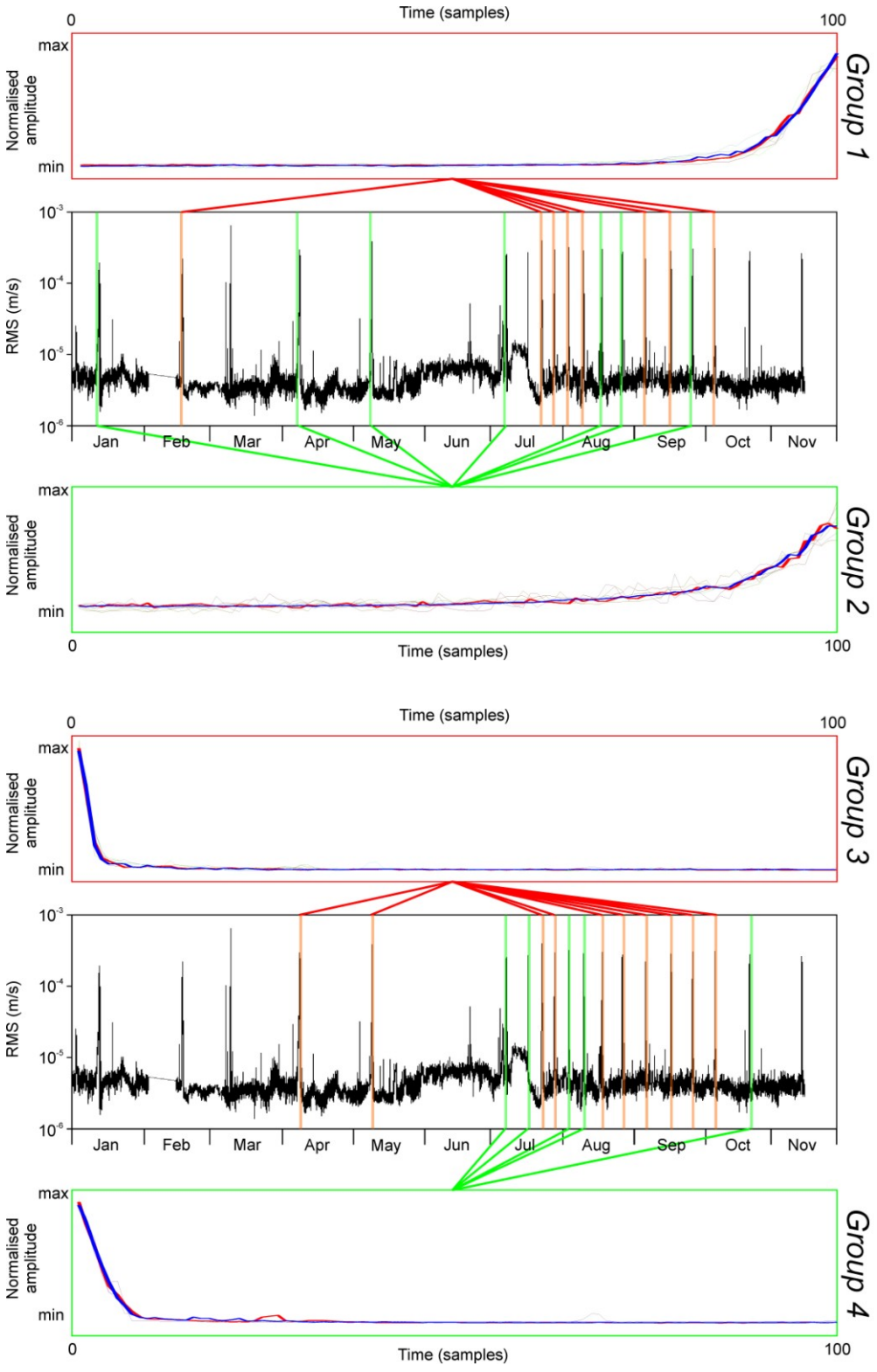


Fig. 6.33. Upper and bottom panels: motifs found in RMS time series (the bold blue and red lines indicate the first and second motifs, respectively). Middle panel: RMS time series with the time intervals corresponding to the found motifs (green and red areas).

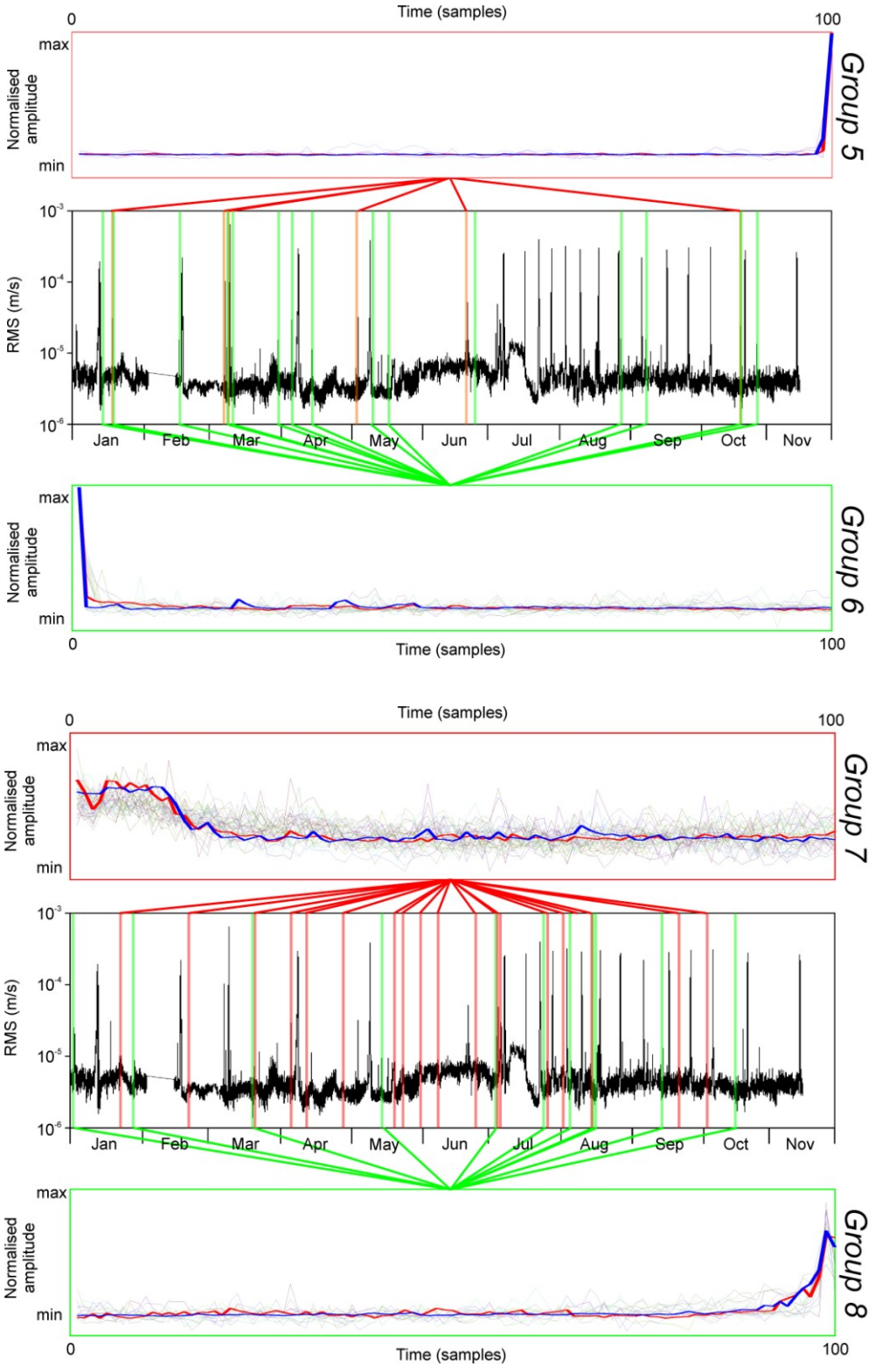


Fig. 6.34. Upper and bottom panels: motifs found in RMS time series (the bold blue and red lines indicate the first and second motifs, respectively). Middle panel: RMS time series with the time intervals corresponding to the found motifs (green and red areas).

Finally, group 8 clusters motifs of quick increases of RMS followed by decreasing trends, reflecting either earthquakes or brief variations of the volcanic tremor amplitude or sequences of energetic LP events (Figs. 6.36 and 6.37c,d). Similarly to groups 5 and 6, the variations of the group 8 generally take place in a few minutes.

5.3.4 Discussion and conclusions

Seismic RMS, together with helicorder and RSAM (real-time seismic amplitude measurements), is one of the most used geophysical parameters in volcano activity monitoring (e.g., Endo and Murray, 1991; Qamar et al., 2008). Over time, it has been observed that many eruptions were preceded and/or accompanied by increases in seismic energy that might have resulted from increases in the volcanic tremor amplitude and/or rate and amplitude of seismic transients such as VT earthquakes and LP events (e.g., Lahr et al., 1994; McNutt, 2000; Moran et al., 2008). In particular, at Mt. Etna many authors highlighted the close relationships between volcanic activity changes and variations of the features of volcanic tremor/long period events (e.g. Alparone et al., 2003; Alparone et al., 2007; Patanè et al., 2008; Aiuppa et al., 2010). Alparone et al. (2003) carried out a qualitative study of the seismic amplitude time series, finding three different repeating patterns of increase of volcanic tremor energy corresponding to the onset of lava fountain phases. However, variations in seismic amplitude time series can also be associated with phenomena external to volcanic eruptions such as distant earthquakes.
In order to discriminate between the two cases, we applied an exact time series motif discovery technique on seismic RMS time series to quantitatively search for recurrent patterns (Figs. 6.33-6.34). This permitted us to observe that different phenomena are characterized by distinct RMS trends, i.e. earthquakes are accompanied by sharp increase and decrease in RMS (lasting from seconds to minutes; Figs.6.34 and 6.36), while lava fountains by slower changes (taking place in hours, sometimes even days; Figs. 6.33, 6.35). The opportunity to assess such a difference between the two kinds of motifs, and hence to distinguish the two source

phenomena might represent a complementary information for the monitoring purposes in volcanic areas.

Analysis of long RMS time series can provide motifs related to specific phenomena (earthquakes, lava fountain initial phases and so on), that can be used as templates, with which to compare the RMS calculated on real-time seismic data. Then, if a good match is found, the evolution of an ongoing eruptive activity could be inferred without any further information. The opportunity to understand the RMS behavior in its increasing phase might be of primary importance, especially in cases of violent and short-time evolving eruptive phases such as lava fountaining episodes. Indeed, lava fountains release copious amounts of volcanic ash and gases into the atmosphere, which may cause danger to the aviation (e.g., Scollo et al., 2009).

At volcano observatories, the routine surveillance and monitoring of volcanic activity can also be carried out by networks of visible and infrared cameras (e.g., Spampinato et al., 2011). These allow the real-time, automatic ground observations that provide information on volcano activity changes. However, these observations are heavily dependent on external effects such as weather conditions, presence of gas and ash along the line-of-sight, which can inhibit and sometimes preclude the visibility. Hence, in these cases the possibility of assessing the kind of phenomenon occurring at the surface by seismic parameters becomes of primary importance. In order to be used as templates for comparison with the RMS calculated in real-time, motifs need to be tested on long time series. Indeed, according to what has been qualitatively reported by Alparone et al. (2003), not all the lava fountains showed exactly the same RMS behavior. The initial phase of the eruptive phenomena can be accompanied either by gradual or quicker RMS increases, and likewise the final phases may show more or less rapid RMS decreases. Integrating our results with volcanological observations of the eruptive activity at the surface, we suggest that the kind of patterns with which lava fountain phases started, i.e. gradual or sharp, depend on the duration of the Strombolian activity that characterized the early stages of all the 2011 lava fountains. In particular, we observed that the longer the strombolian activity the lower the slope of the motif (see groups 1 and 2 for comparison; Fig. 6.33).

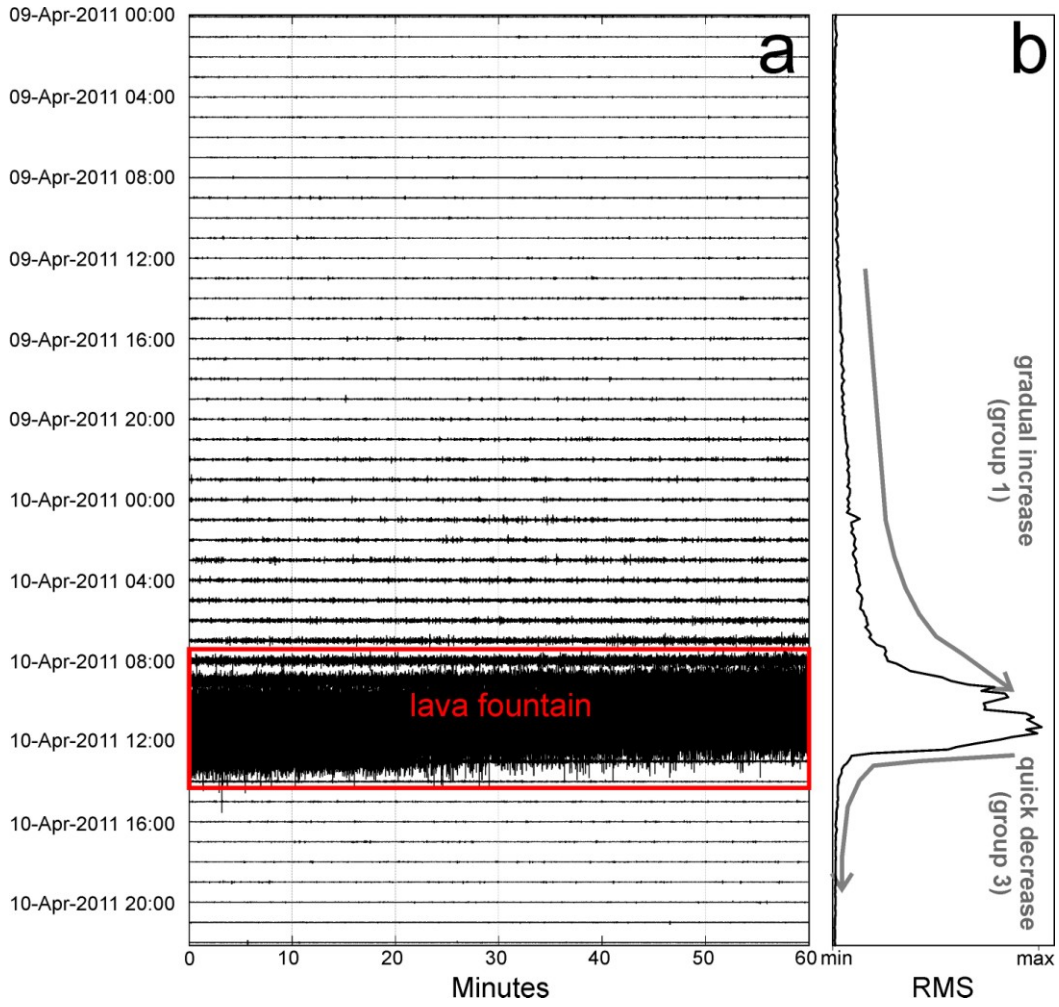


Fig. 6.35. (a) Seismic signal recorded by the vertical component of EBEL station from 9 to 10 April 2011 and (b) corresponding RMS time series calculated on 10-minute-long moving windows.

Similarly, we believe that the contrasting behavior of the lava fountaining waning stages might result from the combination between the duration of the strombolian phase and that of lava effusion following the paroxysmal phase (see groups 3 and 4 for comparison; Fig. 6.33). If this is the case, and assuming the 2011 lava fountaining events have been fed by constant magma supply, we can infer that the different shape of RMS motifs observed might relate to the modalities with which the energy of the event is released at the surface, and thus to the magma transport mechanisms occurring in the volcano shallow feeder system.

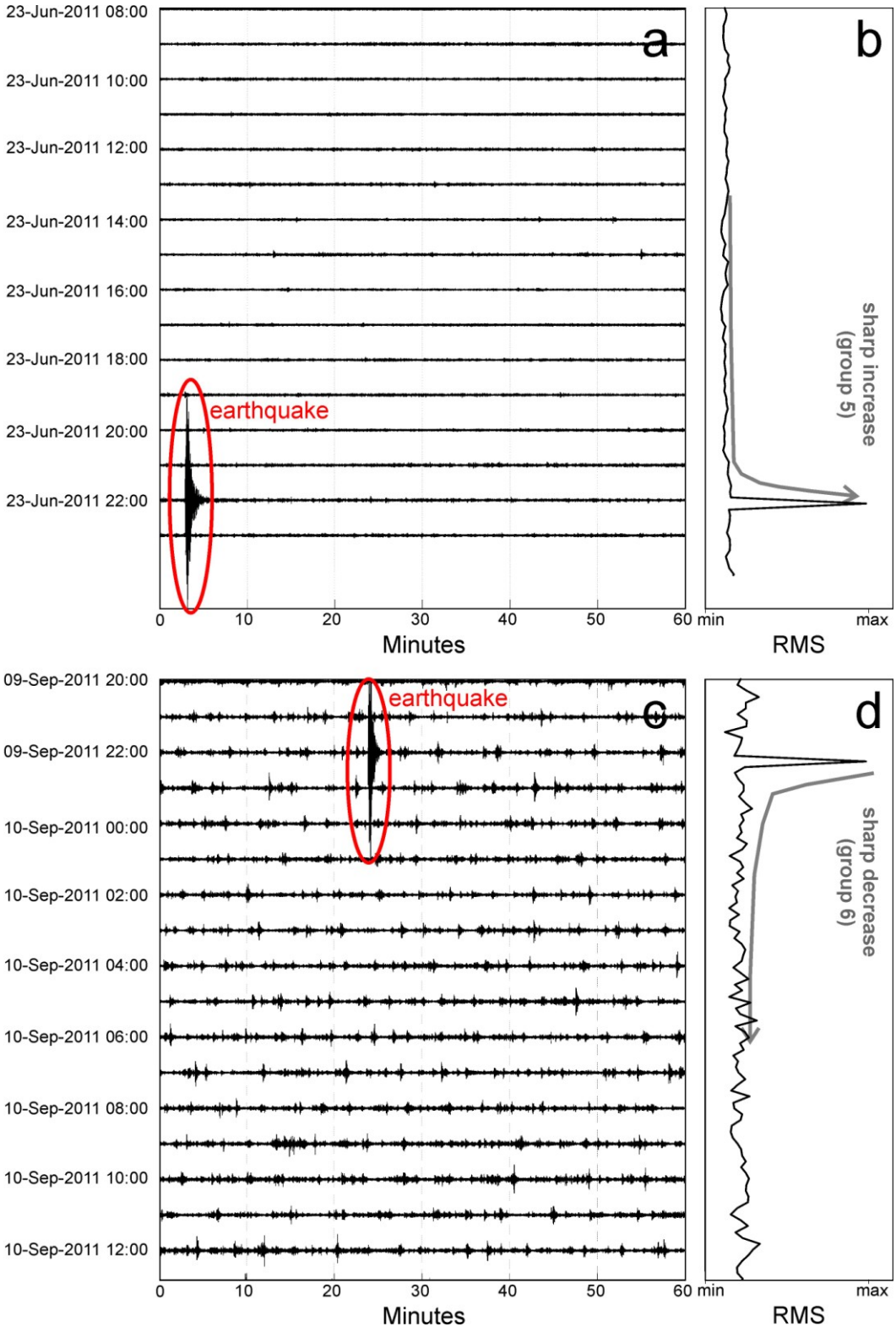


Fig. 6.36. (a,c) Seismic signal recorded by the vertical component of EBEL station on 23 June and 9 September 2011 and (b,d) corresponding RMS time series calculated on 10-minute -long moving windows.

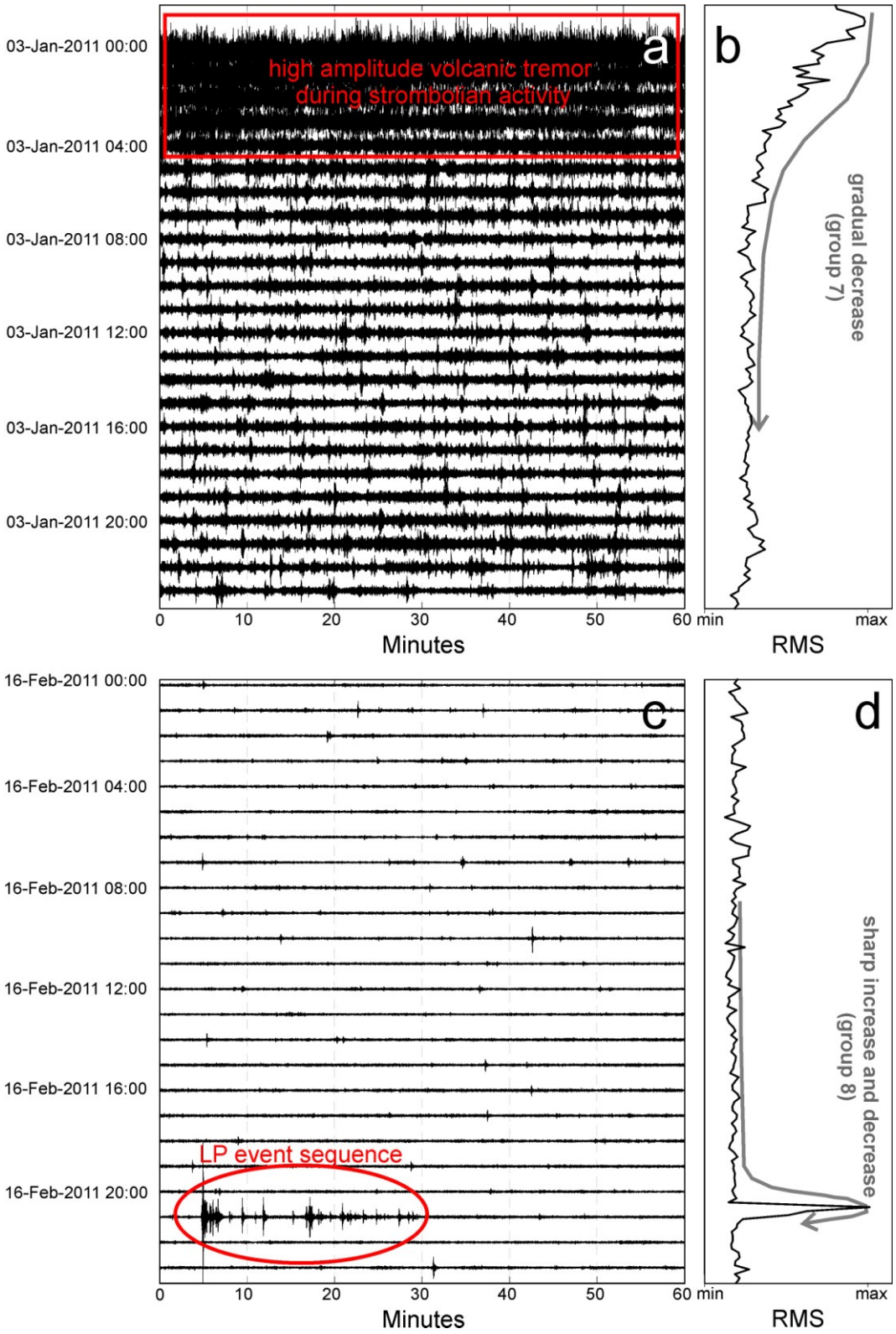


Fig. 6.37. (a,c) Seismic signal recorded by the vertical component of EBEL station on 3 January and 16 February 2011 and (b,d) corresponding RMS time series calculated on 10-minute -long moving windows.

This is only one example of prospective application of this fairly novel MK technique in seismology. In future, a further useful application might be the detection of repeating earthquakes within a continuous seismic signal, for instance used to detect variations of attenuation (e.g. Antolik et al., 1996). Many Authors demonstrated how the common detecting techniques, based on the comparison between the power of a short time window with the power of a long time window (STA/LTA; e.g. Withers et al., 1998), are sometimes ineffective to detect seismic events especially in conditions of low signal to ratio (e.g. Gibbons and Ringdall, 2006; Schaff, 2008, 2009). For this reason, techniques based on the cross-correlation detectors were developed. Such techniques have been used when the waveforms of the events of interest are known (sometimes called "templates"; Shelly, 2010), as well as when there is no a-priori knowledge (e.g. running auto-correlation; Brown et al., 2008).

However, the drawback of these methods, limiting or even preventing their applications on very big datasets, is the computational complexity, that in the latter case (no a-priori knowledge) is equal to run a brute-force search algorithm, using cross-correlation coefficient as similarity function. *MK* algorithm overcomes such limitation by means of two optimization techniques i.e. the *early-abandoning* basic concept and the space indexing. Even if their implementation does not reduce the searching theoretical complexity, with respect to the brute-force algorithm, it is able to speed-up in practice the computation time, especially when dealing with huge datasets (Mueen et al., 2009). The *early-abandoning* is a simple and intuitive enhancement for searching, allowing it to skip many unnecessary steps. Since *MK* algorithm uses a metric (the Euclidean) distance function to perform similarity among time series, it can take advantage of the triangle inequality metric property to build an indexing structure corresponding to a pivot-based linear ordering. This is a well-known architecture in similarity searching area, because it enhances performances allowing pruning operations in the searching space (Zezula et al., 2005).

6.4 An application of segmentation method on seismo-volcanic time series

In this section we show an application of segmentation algorithm (see Section 3.6). Segmentation on seismo-volcanic time series data (Montalto et al., 2012) provides a good method for data compression, allowing faster transmission and visualization.

The time series used for the experiments come from the seismic RMS signal, acquired from summit stations on Mt Etna. In particular, the RMS was calculated on 10-minute-long moving windows, from which the automatic trigger of seismic transients is performed. Skipping details on data elaboration procedure, the final information are relative to the volcanic tremor amplitude and to the number of seismo-volcanic events.

These information, currently used for volcano monitoring, are managed and stored into databases that, over the time, get larger and larger. Moreover, analysts may request to display such a data on very large time intervals. In this case the segmentation method to obtain and display a 'light' version of time series, which does not lose the information content.

The scheme in Fig. 6.38 provides a representation of the system used for the elaboration of the above time series. It starts from the extraction of the data from the database containing the RMS time series and the number of seismo-volcanic transient. Then, data are input to the segmentation algorithm to obtain a compressed version. Segmented data can be stored again into a new database structure used for transmission and visualization purposes.

Fig. 6.39 shows the result of two segmentation processes, relative to the time series (of length m = 8760) showing the number (per hour) of seismo-volcanic events, registered during 2009, and using two different error thresholds. The first experiment, which uses a relative high error threshold, returns a representation of m' = 42 points (Fig. 6.39a). Although points follow the trend of the original time series, the considered representation offers little detail. To refine the approximation degree, we used a smaller error threshold. Fig. 6.39b shows the resulting time series of m'' = 199 points. In this last case the improved approximation allows to see also faster variations.

The segmentation was applied also to the RMS time series. In the reported examples, the RMS time series was calculated on 10-minute-long moving windows and filtered in frequency range 0.5–5 Hz. Fig. 6.40 shows the result of two different approximations of a time series of length m = 52560, with 532 (Fig. 6.40a) and 171 (Fig. 6.40b) points.

Both representations well maintain information about the original time series trend. Smaller the error threshold is chosen, more precise will be the approximation of the time series. Differently from a simple moving average, the information content remains unaltered, so information about fast variations are not lost.

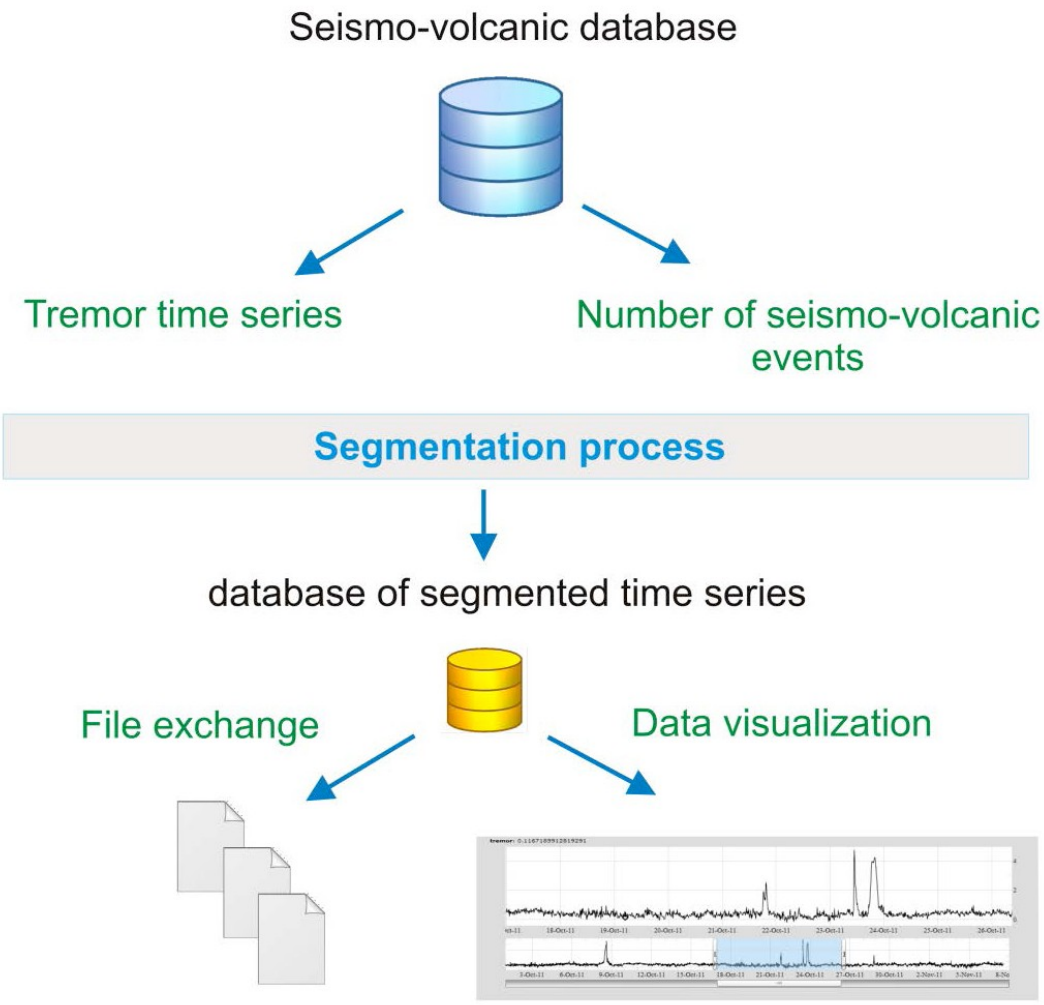


Fig. 6.38. Scheme of the system using segmentation process.

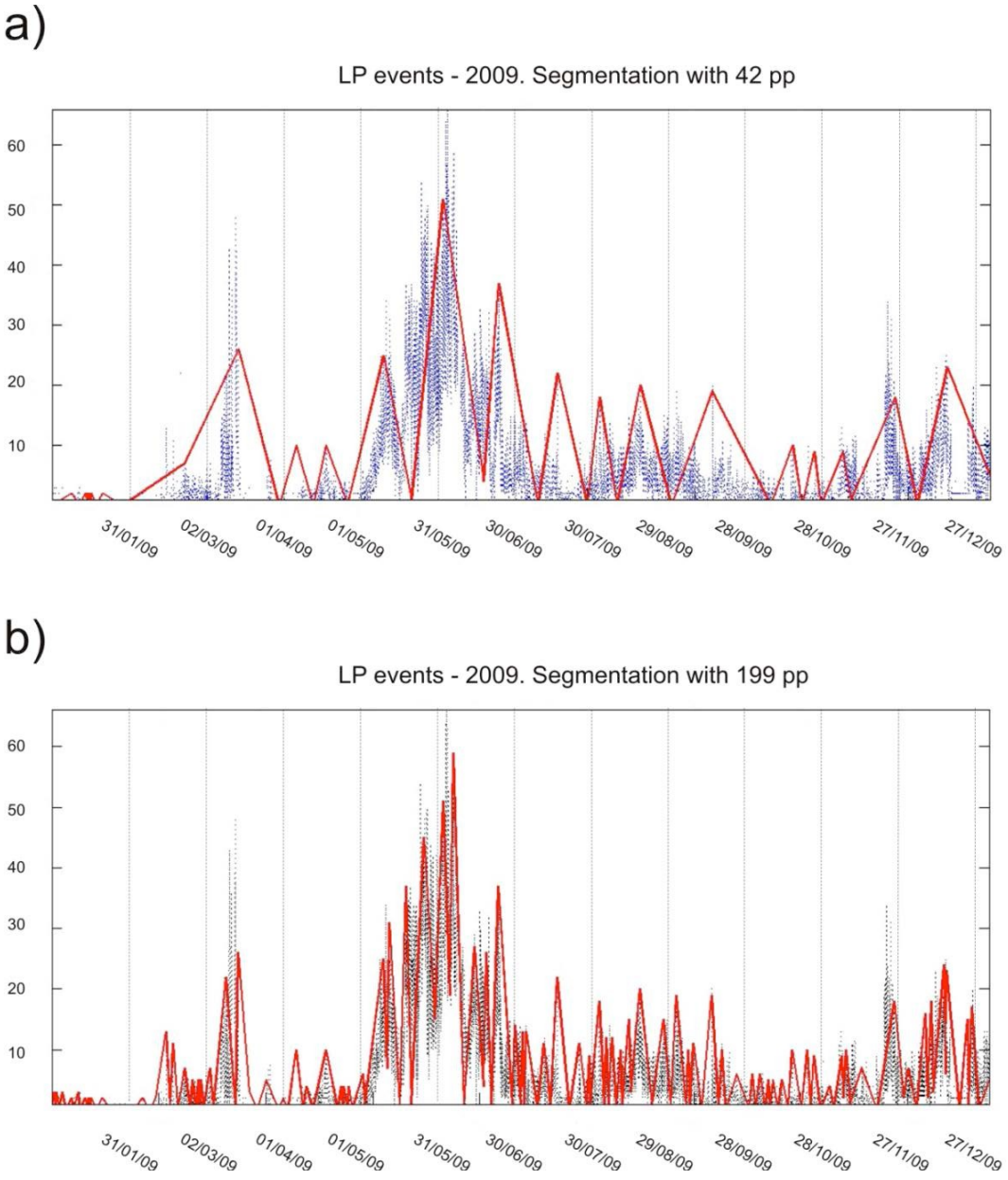


Fig. 6.39. Time series representing the number of LP events registered during 2009. The dashed line indicates the original time series, while the red line represents the segmented time series. (a) Segmentation with 42 points (high error threshold); (b) Segmentation with 199 points (smaller error threshold).

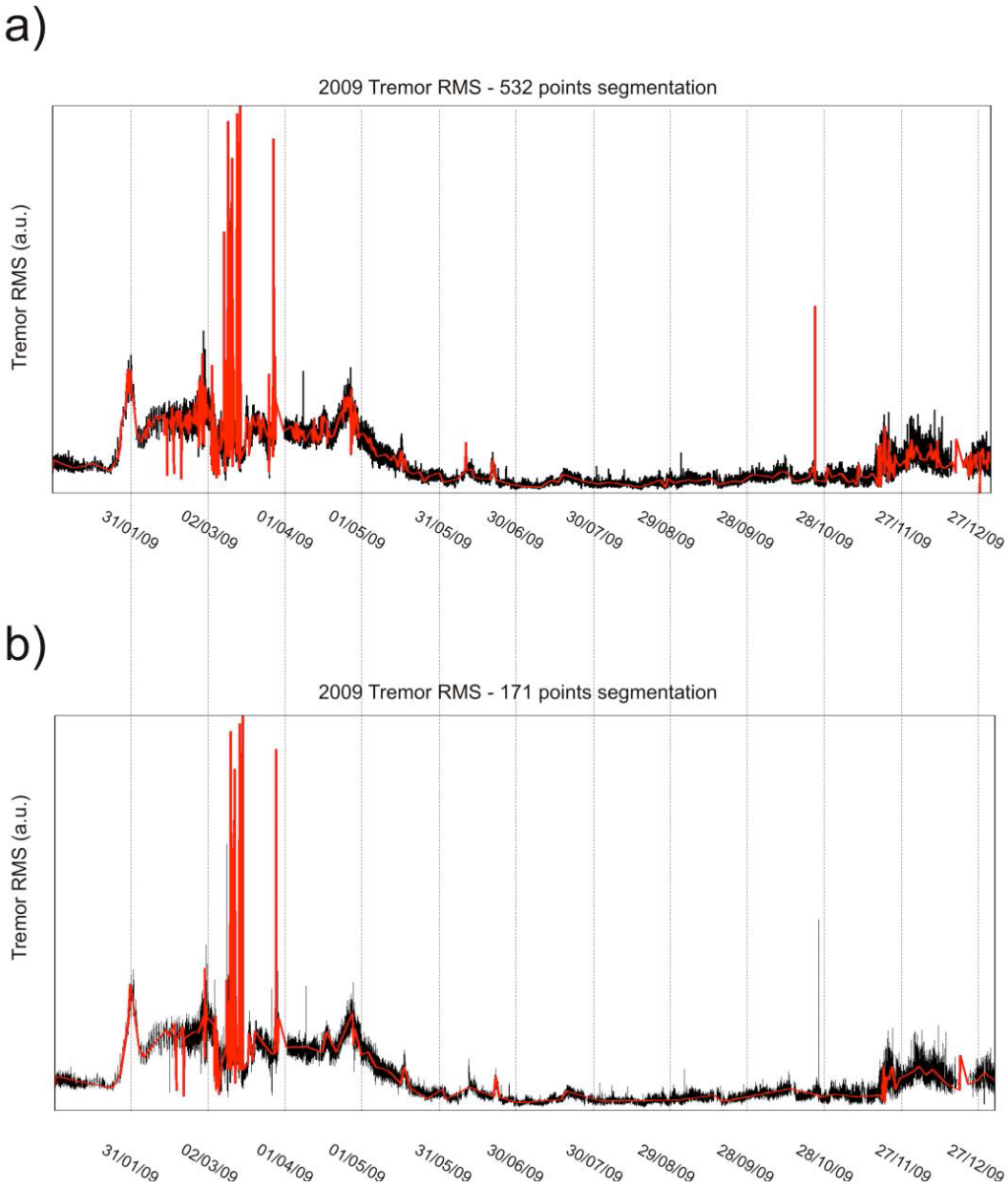


Fig. 6.40. Time series representing RMS of seismic signal recorded by the vertical component of EBEL station and filtered in the band 0.5-5.0 Hz (black line). The number of point of the time series is 52560. The red line represent the segmented time series. (a) Segmentation with 532 points; (b) Segmentation with 171 points.

6.5 Monitoring volcano activity through Hidden Markov Model

During 2011, Mt. Etna was mainly characterized by cyclic occurrences of lava fountains, totaling to 18 episodes. During this time interval Etna volcano's states ("Quiet", "Pre-fountain", "Fountain", "Post-fountain"; Fig. 6.41), whose automatic recognition is very useful for monitoring purposes, turned out to be strongly related to the trend of RMS of the seismic signal recorded by stations close to the summit area. Since RMS behavior is considered to be stochastic, we can try to model the system, assuming to be a Markov process, by using *Hidden Markov models* (*HMMs*). *HMMs* analysis seeks to recover the sequence of hidden states from the observed emissions (Fig. 4.8). In our framework, observed emissions are characters generated by the *SAX* (Lin et al., 2007; see also Section 3.8) technique, which maps RMS time series values with discrete literal emissions. The experiments show how it is possible to guess volcano states by means of *HMMs* and SAX.

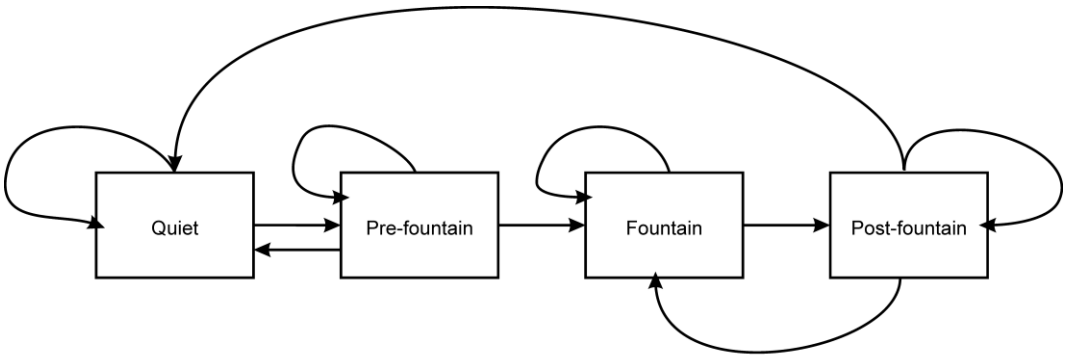


Fig. 6.41. Diagram of volcano's states transitions.

6.5.1 Modelling RMS values distribution

As we have just seen in Section 3.8, to give significance to the symbolic transformation, it is necessary to deal with a system producing symbols with equal probability, or with a Gaussian distribution. This is the first assumption to use the *SAX* algorithm, because breakpoints (Def. 3.8.1) correspond to quantiles of the cumulative of a normal distribution function. However, this is a limit for our purposes: the probability

distribution of RMS values are not approximated by a Gaussian (Fig. 6.42), and even if Lin et al. (2007) advised that this limit can be overcome by normalizing data, we don't want to perform any transformation to maintain the correctness as well as the significance of the signal.

Distribution fitting

We performed a chi-square goodness-of-fit test by assuming the null hypothesis that the data in the RMS are random samples from a known distribution with parameters estimated from the samples (i.e. the RMS values).
The chi-square test compares the observed frequency distribution in the sample, *O*, with the expected frequency distribution, *E*. The difference between the observed and expected (*O* - *E*) is squared to remove negative signs, and then standardized by the expected frequency in that class or range, in order to obtain a standardized measure of the difference between the two distributions. The sum of these standardized differences is then calculated and compared to a chi-square distribution with *n*-1 degrees of freedom, where *n* is the number of frequency classes used in the calculation. The formula used is thus of the form:

$$\chi^2_{n-1} = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{6.10}$$

To do this we used *chi2gof* from *MATLAB Statistic Toolbox* (http://www.mathworks.it/it/help/stats/chi2gof.html). The result *h* of this procedure is 1 if the null hypothesis can be rejected at the 5% significance level. The result *h* is 0 if the null hypothesis cannot be rejected at the 5% significance level. The null distribution can be an arbitrary discrete or continuous distribution (e.g. normal, Poisson, gamma, lognormal, exponential). The test is performed by grouping the data into bins, calculating the observed and expected counts for those bins, and computing the chi-square test statistic. *chi2gof* sets the number of bins, *nbins*, to 10 by default, and compares the test statistic to a chi-square distribution with *nbins* – 3 degrees of freedom to take into account the two estimated parameters.

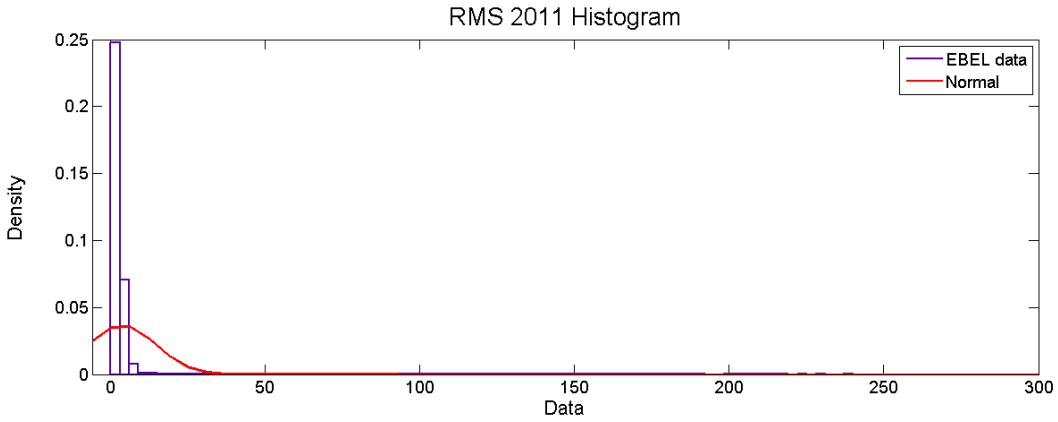


Fig. 6.42. Histogram of RMS values calculated on the seismic signal recorded at EBEL station in the period starting from 00:00 on 1 January 2011 to 00:00 on 1 December 2011. The red shape is relative to the Gaussian distribution fitted to the RMS values distribution.

We tested RMS distribution with several known distributions, and for each test the above procedure rejected the null hypothesis, so we decided to make a non-parametric estimation through the kernel density estimation (KDE) of the *dfittool* (*Distribution Fitting Tool* of *MATLAB Statistic Toolbox*). There are some techniques to evaluate the number of bins (Appendix E) to use for the histogram computation (Fig. 6.42). We chose to apply the *Freedman-Diaconis* rule (Eq. E.3), because the *Sturges'* rule (Eq. E.1) is not indicated when the number of samples is > 200, while the *Scott*'s rule (Eq. E.2) requires knowledge about the distribution of the data, and assumes its normality, which we don't have.

Symbolization

To get breakpoints we picked quantiles from the custom *cdf* (*cumulative distribution function*) of the RMS distribution (Fig. 6.43). Since values corresponding to fountain activity are much rarer than values corresponding to quiet periods, they can be considered outliers. In this case the regions between zero and the fifth percentile and between the ninety-fifth and one hundredth percentiles are of great interest (Lodder and Hieftje, 1988). If we want to ensure a symbolization with at least a Gaussian distribution, we cannot ignore them and then we have to redefine the breakpoints definition (Def. 3.8.1) by adding to the original

breakpoints (initially we set the alphabet size to 4, and then kept the first, the second and the third quantile) also the 1%, 5% and the 95%, 99% of the cumulative frequency. With this system we will deal with a number of symbols equal to 8. The results of this choice are shown in Fig 6.44.

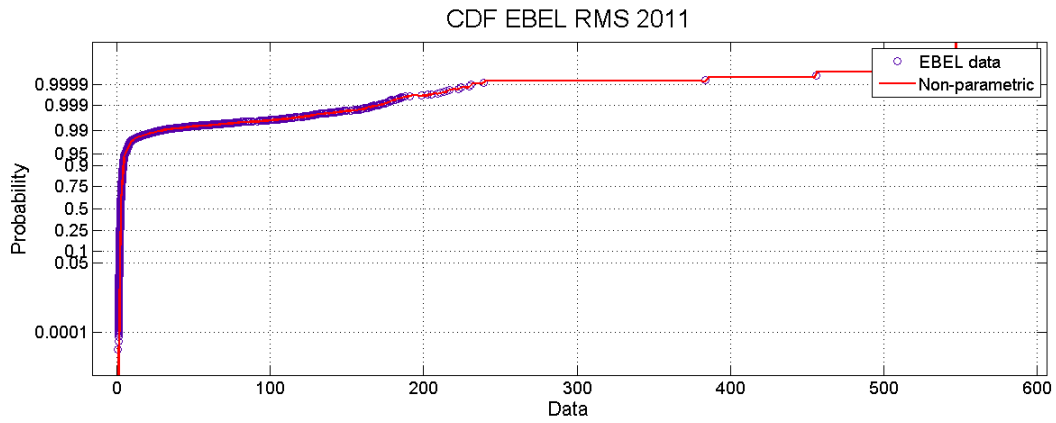


Fig. 6.43. The cumulative distribution function plot of RMS data registered at EBEL station in the period starting from 00:00 on 1 January 2011 to 00:00 on 1 December 2011. The red shape is relative to the calculated (with *dfittool* of the *MATLAB Statistics Toolbox*) non-parametric distribution fitted to the RMS values distribution.

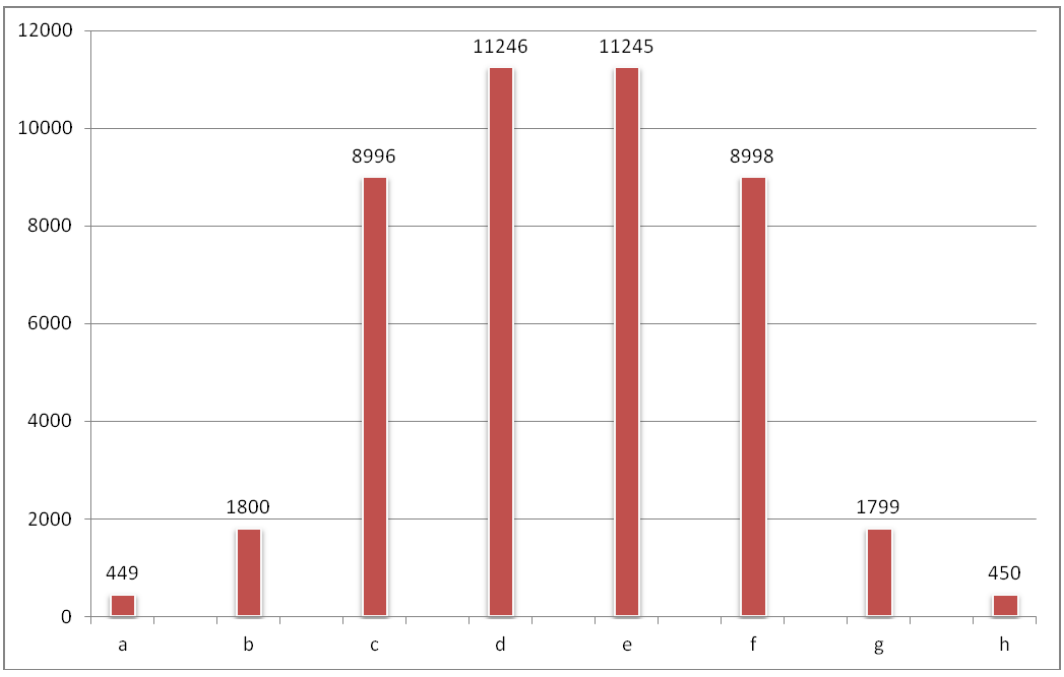


Fig. 6.44. Histogram relative to the occurrences of each symbol generated by *SAX* algorithm on the above mentioned RMS time series, using our breakpoints definition. We can notice the normal distribution tendency.

6.5.2 Implementing the framework

After having calculated custom breakpoints, we were able to produce symbols using the *SAX* technique. For each symbol we added a '+' or a '-' basing on the value corresponding to the symbol: if it is greater than the previous on the time series, then we add a '+', else we add a '-' (Fig. 6.45).

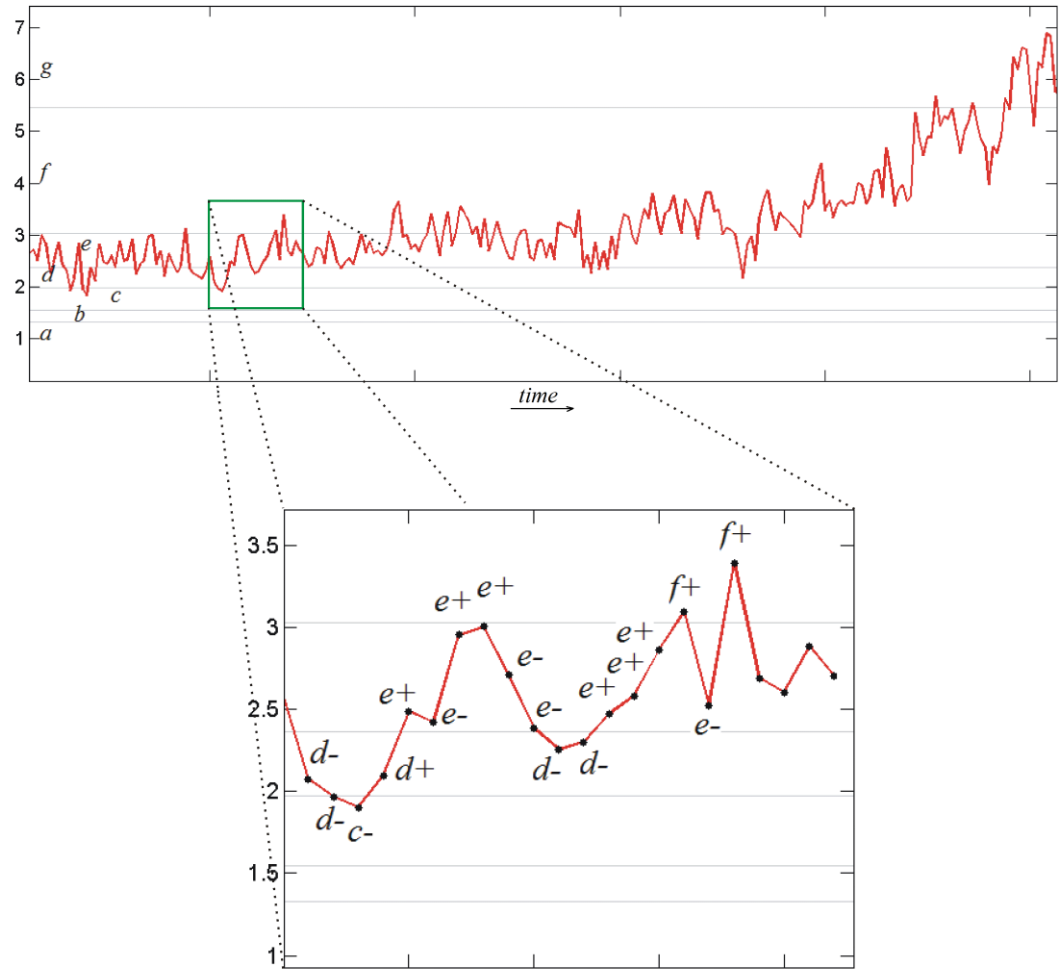


Fig. 6.45. Our symbolization includes the use of an additional symbol ('+', '-') basing on the value corresponding to the symbol: if it is greater than the previous on the time series, then we add a '+', else we add a '-'.

We also add to our alphabet the symbol '_' relative to the RMS values equal to -1 indicating the lack of signal (Fig. 6.46). This further addition makes our framework more robust on states classification, without significantly altering the symbols distribution (Fig. 6.47). Then, the

complete alphabet is the following: O = {'*a-*', '*a+*', '*b-*', '*b+*', '*c-*', '*c+*', '*d-*', '*d+*', '*e-*', '*e+*', '*f-*', '*f+*', '*g-*', '*g+*', '*h-*', '*h+*', '_'}.

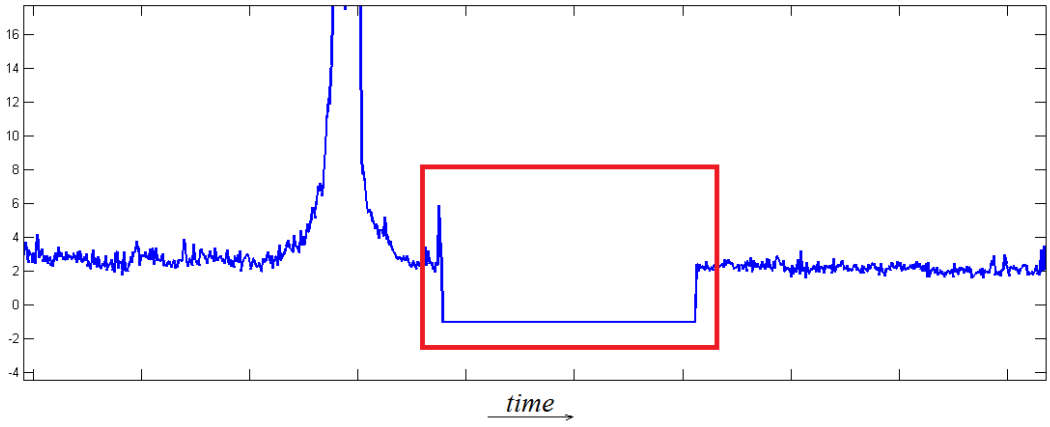


Fig. 6.46. The red rectangle highlights a period of absence of signal (lack), indicated with -1 values. We convert them with the literal symbol '_'.

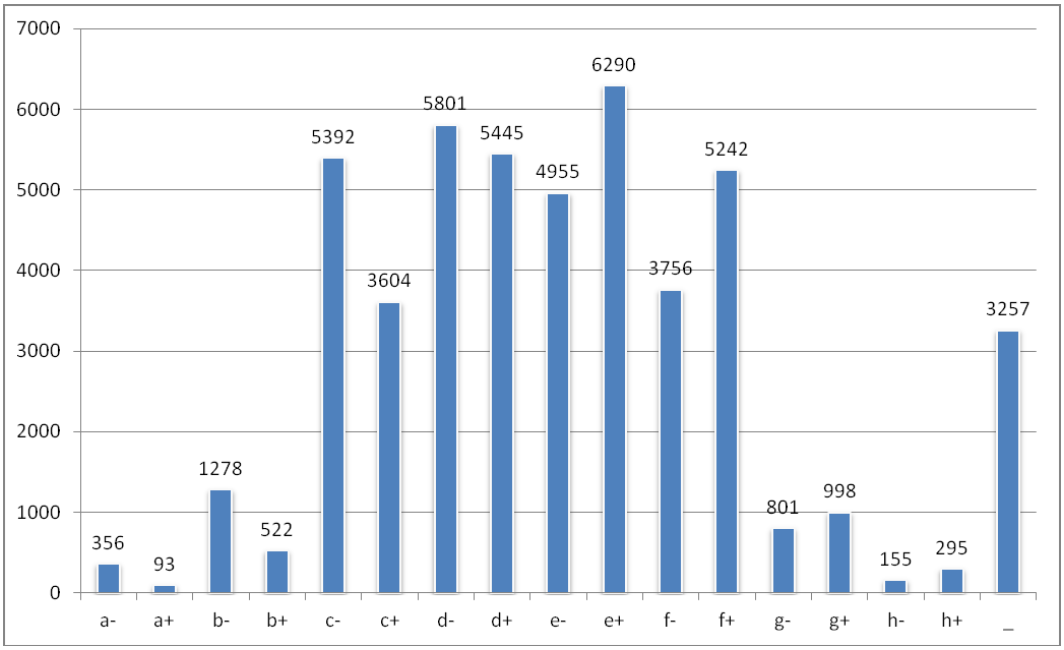


Fig. 6.47. Histogram relative to the occurrences of each symbol generated by our framework on the above mentioned RMS time series. Without considering the bar relative to the '_' symbol (which varies in relation to the 'holes' in the signal), we can notice how the normal distribution tendency is not altered respect to that shown in Fig. 6.44.

HMM settings

As we saw in Section 4.2.1, an *HMM* is denoted by $\lambda = \{S, A, O, E\}$:

1. Volcano states are "Quiet", "Pre-fountain", "Fountain", "Post-fountain". We decided to include other two states to set our *HMM*, because the managed RMS time series shows values relative to other two phenomena: a) peaks relative to earthquakes; b) values equal to -1 when there is no signal, then *S* = {'*Quiet*', '*Pre-fountain*', '*Fountain*', '*Post-fountain*', '*No Signal*', '*Earthquake*'} (Fig. 6.48).
2. The state transition matrix $A = \{a_{ij}\}$ relative to the diagram in Fig. 6.48 is shown in Table 6.4.
3. The set of possible observations is *O* = {'*a-*', '*a+*', '*b-*', '*b+*', '*c-*', '*c+*', '*d-*', '*d+*', '*e-*', '*e+*', '*f-*', '*f+*', '*g-*', '*g+*', '*h-*', '*h+*', '_'}.
4. The emission matrix $E = \{e_{ij}\}$ is shown in Table 6.5.
5. We denoted the state '*Quiet*' as starter states, so π = {1, 0, 0, 0, 0, 0}.

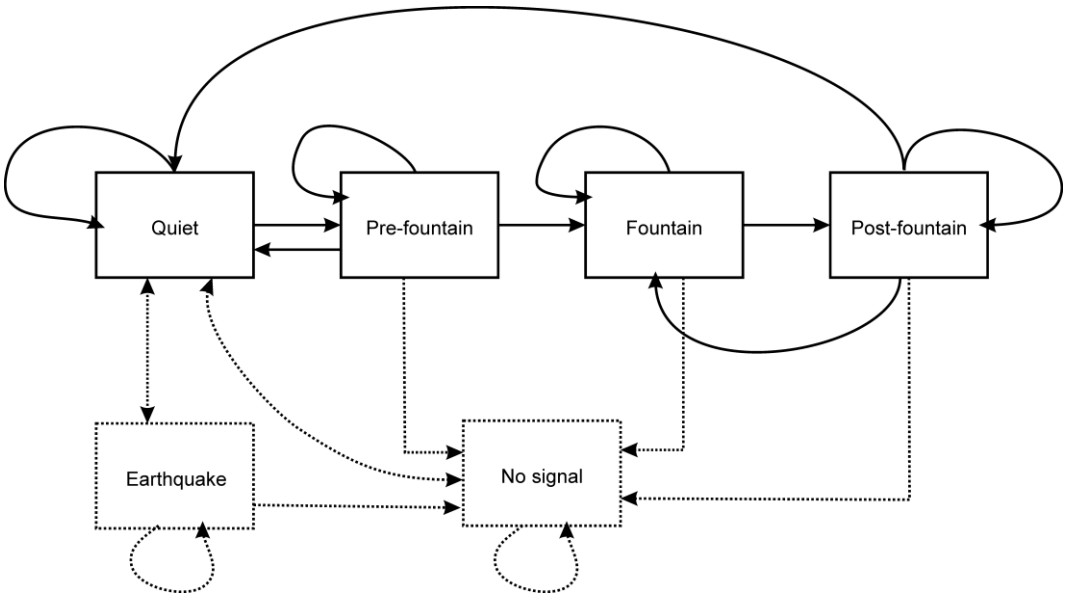


Fig. 6.48. The diagram of states transition used in our framework. It includes other two 'symbolic' states respect that shown in Fig. 6.9: "Earthquake" and "No signal".

Both emission and state transition matrices (Tables 6.4 and 6.5) are based on a statistical analysis conducted on the period relative to the first month of the signal (January 2011), which provided information about the

studied states. The testing phase, described in the following paragraph, is applied on the rest of the signal (from February to December 2011).

6.5.3 Classification results

To recover the sequence of the volcano states from the RMS signal we applied the Viterbi algorithm of the *HMM R*-package (http://cran.r-project.org/web/packages/HMM/index.html) on the part of time series not learned by the model. Figs. 6.49-52 show some snapshots of the results.
To test the quality of states classification, we selected the most important episodes visible from RMS and saved their HMM classification into a confusion matrix (Table 6.3; see also Section 6.1.4), where each column represents the state predicted by the HMM, while each row represents states attributed by an accurate event classification conducted by the analysts. The signal relative to the tested period was characterized by several peaks: some corresponding to 17 eruptive episodes, and other (averagely smaller than previous) due to local, regional and teleseismic earthquakes. Table 6.3 allows to evaluate the classification result: we are interested in the blue-filled part, related to fountains phenomena. We completely ignored the '*Noise*' state, given the banality of its classification.
The row '*Quiet*', in the confusion matrix, contains instances relative to periods of small increments of tremor or to LP events. In all cases they were related to '*Quiet*' states, according to our aims. In only two cases the classification conducted to the '*Earthquake*' state, because noise in the signal caused very high peaks (see the [*Quiet*, *Eq*] entry in the confusion matrix of Table 6.3).
The most important row, for monitoring purposes, is the '*Pre*' state. Almost all eruptive episodes are preceded by an increasing in tremor. In the tested signal this happens for all episodes. All real '*Pre*' fountain states were classified as '*Pre*', included an "aborted" episode on 7 July (Fig. 6.52).
We obtained good results for the '*Fountain*' state. Even if we can notice 11 occurrences of '*Post*' classification for '*Fountain*' state (see the [*Fountain*, *Post*] entry), they are a subset of the 17 occurrences in the [*Fountain*, *Fountain*] entry, because they just relates to a quickly identification of the '*Post*' state, when the fountain is going to its final phase, which is represented by ever lower RMS values.

Regarding the identification of the '*Post*' fountain state, an ordinary mistake was noted: since values belonging to '*Post*' fountain are very similar to values relative to '*Pre*' fountain, they can bring the HMM to assign them to '*Pre*' state (see the [*Post*, *Pre*] entry). This can also be seen, for example, in Figs. 6.49 and 6.51: in the descending phase of the signal peak, corresponding to the fountain, there are some yellow circles, representing '*Pre*' state classification, between blue and green circles, representing respectively '*Post*' and '*Quiet*' states classification. This is not worried because, if this assignment occurs, it follows systematically a previous identification of the '*Post*' state. So, a specific rule can be applied at a later stage.

The blank-filled part of the confusion matrix, in Table 6.3, is relative to the earthquakes identification, which is out of our intentions. In any case, most of the earthquakes are related to the '*Quiet*' state (see the [*Eq*, *Quiet*] entry), or to the '*Earthquake*' state (see the [*Eq*, *Eq*] entry). It is worth noting that the heavy influence of the earthquakes on the signal can sometimes lead to the identification of a '*Pre*' state (5 instances in the [*Eq*, *Pre*] entry). However, this can be solved by the application of adequate filters on the RMS signal, in a previous step of the data acquisition (see the *KDD* process in Fig. 1.1). If we calculate statistics on the focused part (blue in Table 6.3), we can estimate a 'hit' rate of 63.51%. This could be daunting if we give importance to the [*Fountain*, *Post*] and the [*Post*, *Pre*] entries. By excluding them we reach an 'hit' rate of 95,91%.

		PREDICTED				
		Quiet	***Pre***	***Fountain***	***Post***	***Eq***
ACTUAL	***Quiet***	**6**	0	0	0	2
	Pre	0	**18**	0	0	0
	Fountain	0	2	**17**	11	0
	Post	0	14	0	**6**	0
	Eq	9	5	1	1	**3**

Table 6.3. Confusion matrix calculated on events occurring during February – December 2011. The '*Eq*' label refers to '*Earthquake*'.

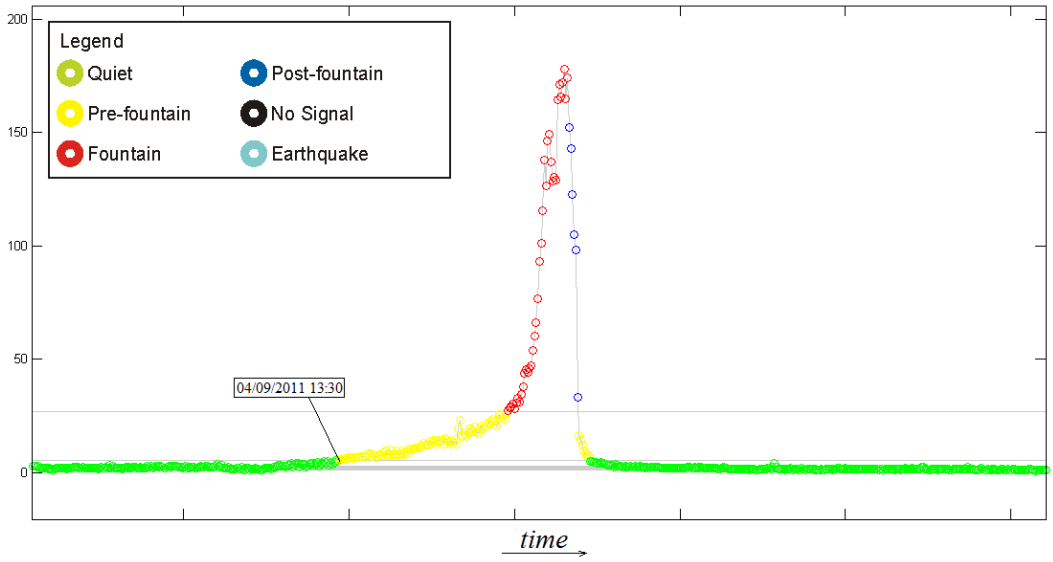


Fig. 6.49. A typical trend of RMS time series when a fountain occurs. In almost all cases, the HMM can distinguish when the volcano state is going to fountain (red circles), by passing in the "Pre-fountain" state (yellow circles) and discriminating the "Post-fountain" state (blue circles) when the RMS value goes down.

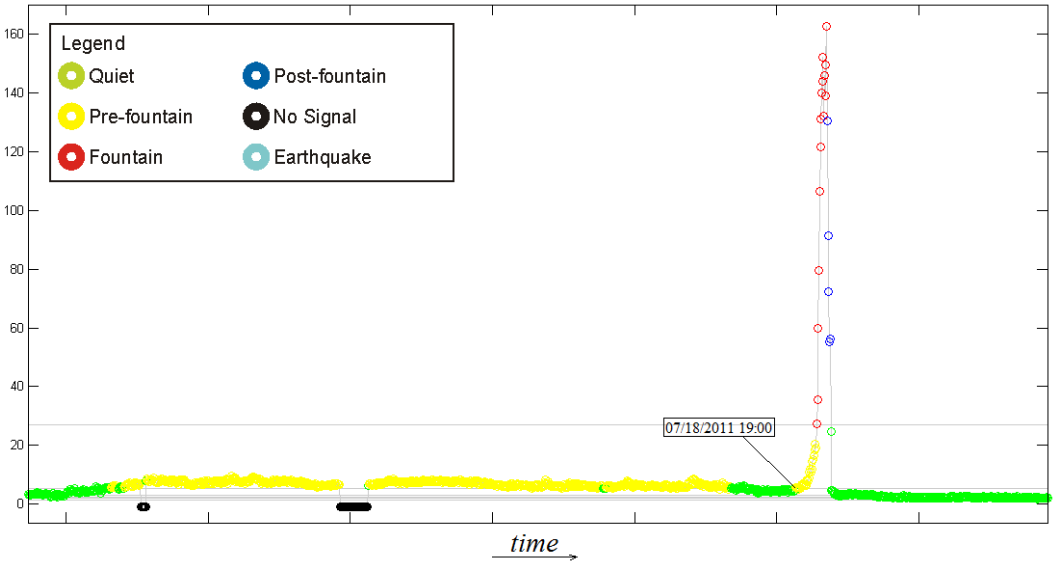


Fig. 6.50. In this case, before the fountain (peak in the time series) there is a period with relatively high RMS values, which lead the HMM to assume the "Pre-fountain" state.

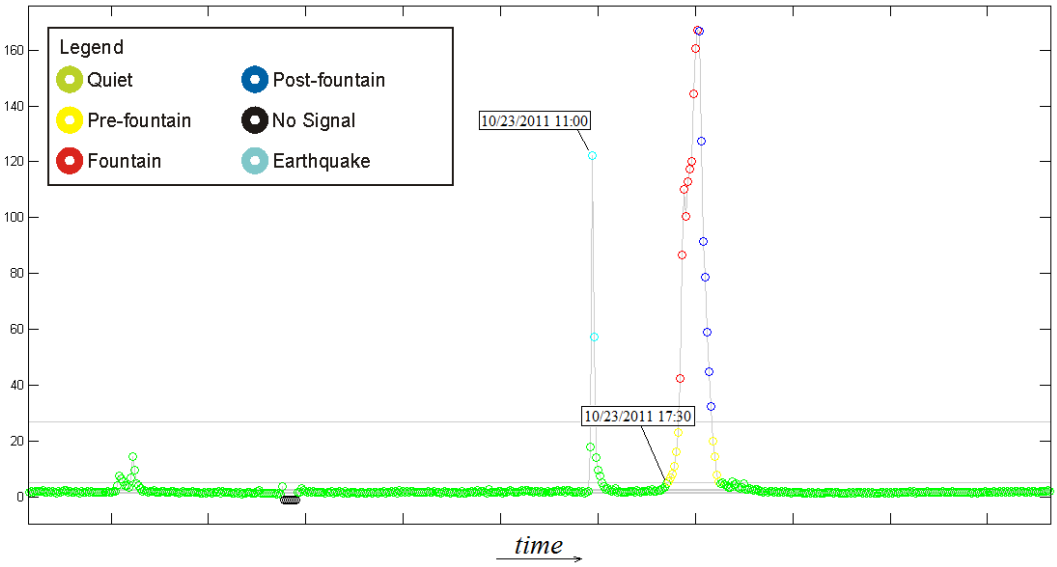


Fig. 6.51. Another fountain (right peak in the time series) preceded by an earthquake (left peak in the time series). Also in this case the HMM recognizes the two different phenomena.

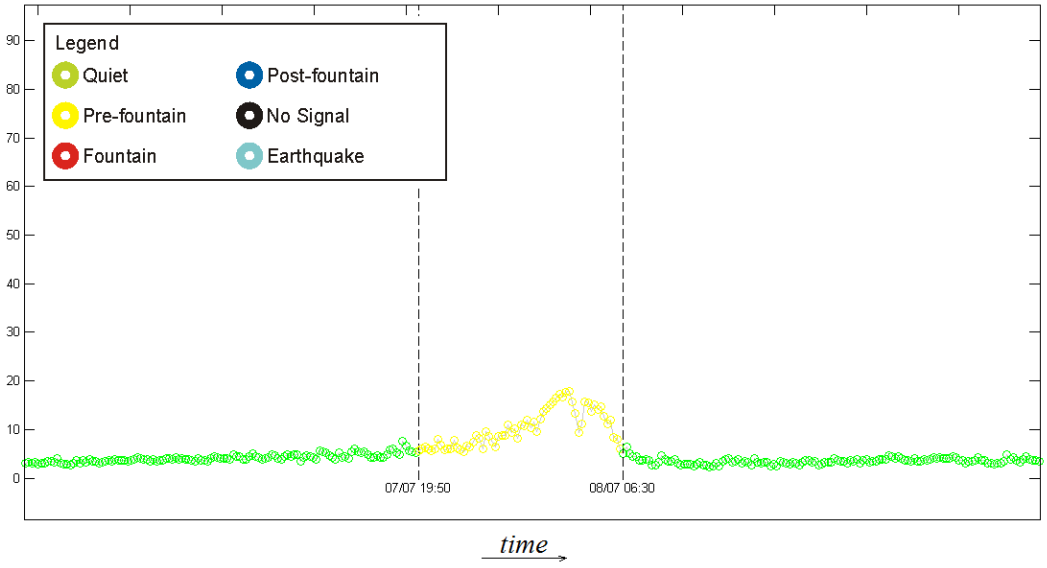


Fig. 6.52. RMS signal relative to the aborted fountain on 7 July 2011.

Trans Matrix	QUIET	PRE	FOUNTAIN	POST	NOS	EQ	Sum Probs
QUIET	0,93	0,01	0	0	0,01	0,05	**1**
PRE	0,3	0,58	0,1	0	0,01	0,01	**1**
FOUNTAIN	0	0	0,69	0,29	0,01	0,01	**1**
POST	0,3	0,09	0	0,59	0,01	0,01	**1**
NOS	0,1	0	0	0	0,9	0	**1**
EQ	0,79	0	0	0	0,01	0,2	**1**

Table 6.4. Guess transition matrix for the HMM model.

Emiss Matrix	a-	a+	b-	b+	c-	c+	d-	d+	e-	e+	f-	f+	g-	g+	h-	h+	_	Sum Probs
QUIET	0,01	0,01	0,02	0,02	0,05	0,05	0,12	0,12	0,12	0,12	0,12	0,11	0,05	0,05	0,01	0,01	0,01	**1**
PRE	0,02	0,02	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,2	0,34	0,3	0,01	0,01	**1**
FOUNTAIN	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,3	0,55	0,01	**1**
POST	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,05	0,02	0,5	0,3	0,01	**1**
NOS	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,84	**1**
EQ	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,43	0,42	0,01	**1**

Table 6.5. Guess emission matrix for the HMM model.

Conclusions

This thesis submits the application and engineering of data mining algorithms. The discussion seeks to propose both purely theoretical solutions, to improve the efficiency of certain techniques, and applicative solutions in geophysics.

The thesis mainly focuses on the issue of "similarity matching" in connection with mining. We introduced this concept, exposing the usefulness of appropriate data structures to perform similarity search, and the choice of suitable similarity/distance measures to enhance the efficiency and the effectiveness. We highlighted the importance of some techniques to reduce the data dimensionality, because data are often represented by a large number of attributes (as in the case of time series, which is the main representation of the phenomena studied in a geophysical context). We then provided experiments and comparisons among several data compression methods, and proposed a new data-storage engine able to speed the elaboration and visualization of geophysical time series.

Concerning the similarity matching field, an important section was devoted to the clustering, with a detailed review of the main techniques, which allow the unsupervised analysis of similar objects in datasets. We described a new algorithm of density-based clustering, developed by the thesis' author, which is particularly competitive with other algorithms constituting the state-of-art in this area.

We also demonstrated the importance of the applications of data mining on geophysical data which, for our experiments, were kindly provided by the Istituto Nazionale di Geofisica e Vulcanologia (INGV), Section of Catania - Osservatorio Etneo, which also granted funds for the author's doctorate cycle. Since the amount of data is very large and there is a standard representation for each type of signal, we were able to process and analyze them by means of data mining techniques. We performed real-time data analysis, by

developing several systems addressing to the monitoring task of the volcanoes. Important results were achieved on infrasonic signals monitoring on Mount Etna. In particular, we proposed a novel infrasonic events location system based on DBSCAN and SVM, which make use of the infrasonic signal features to find associations among signals and active craters. Another application based on Hidden Markov Models was developed for volcanic tremor analysis with the aim of finding critical states of Mount Etna volcano. Other techniques, such as the Mueen-Keogh algorithm, were applied on historical seismic data for the extraction of recurrent patterns, which is a crucial step in the analysis of geophysical time series. Its application allowed increasing the amount of useful information for monitoring purposes, and a promising tool to use for the analysis of other geophysical data.

Appendices

Appendix A

Covariance Matrix

Covariance is a well-known concept in statistics. Let D be a data set with n objects, each of which is described by m attributes $v_1, v_2, \ldots, v_m$. The attributes $v_1, v_2, \ldots, v_m$ are also referred to as variables. The covariance between two variables v_i and v_j (with $0 < i, j \leq m$) is defined to be the ratio of the sum of the products of their deviation from the mean to the number of objects:

$$\mathrm{cov}(v_i, v_j) = \frac{1}{n}\sum_{k=1}^{n}(x_k^{(i)} - \mu_i)(x_k^{(j)} - \mu_j) \tag{A.1}$$

where $x_k^{(i)}$ denotes the ith attribute of x_k (See Eq. 2.1), and μ_i is the mean of all data points in the ith variable:

$$\mu_i = \frac{1}{n}\sum_{k=1}^{n} x_k^{(i)} \tag{A.2}$$

The covariance matrix is a $m \times m$ matrix in which the entry (i,j) contains the covariance between variable v_i and v_j:

$$\begin{pmatrix} \mathrm{cov}(v_1, v_1) & \mathrm{cov}(v_1, v_2) & \cdots & \mathrm{cov}(v_1, v_m) \\ \mathrm{cov}(v_2, v_1) & \mathrm{cov}(v_2, v_2) & & \\ \vdots & & \ddots & \\ \mathrm{cov}(v_m, v_1) & & & \mathrm{cov}(v_m, v_m) \end{pmatrix} \tag{A.3}$$

Appendix B

SOMPI method

Time-series modelling consists of estimating the governing dynamics of the hypothetical linear system that has yielded the given time-series data (Kumazawa et al. 1990). In these approaches, a signal is considered as the impulse response of an AR or an autoregressive moving average (ARMA) filter. In general, ARMA filter is a discrete-time system that takes an input sequence x_n and produces an output sequence y_n. This kind of system can be described by a linear-constant difference equation:

$$x_n = \sum_{k=1}^{p} a_k x_{n-k} - \sum_{k=1}^{q} b_k y_{n-k} \qquad \text{(B.1)}$$

where $\{a_k\}$ and $\{b_k\}$ are the system coefficients, p and q are the order of the AR and MA parts of the filter, respectively. The coefficients of the AR filter can be obtained by solving the modified Yule–Walker equation (Marple 1987) and the coefficients of the MA filter can be estimated using the Durbin method (Kay 1988; Mars et al. 2004). As argued in Lesage (2008), this process is affected by numerical instabilities and long computation time. Furthermore, the deconvolution of the AR part alone gives good estimation of the duration and spectral content of the considered signals (Lesage 2008).

To estimate the AR coefficients, the Sompi method (Kumazawa et al. 1990) can be implemented. Unlike the traditional spectral estimators in real frequency space, this method yields a line-shaped spectrum in complex frequency space. The basic concepts of the AR model and the formulation based on the maximum likelihood principle lead to a model estimation algorithm different from other AR methods (Fukao and Suda 1989; Kumazawa et al. 1990). By Sompi analysis, a time-series is deconvoluted into

a linear combination of coherent oscillation with decaying amplitude and additional noise. Let (x_n) time-series that can be considered the sum of signal (u_n) and Gaussian white noise (e_n):

$$x_n = u_n + e_n \tag{B.2}$$

where u_n is described as a set of decaying sinusoids:

$$u_n = \sum_k \left\{ C_k (z_k)^n + C_k^{\,*} (z_k^{\,*})^n \right\} \tag{B.3}$$

and z_k is defined as:

$$z_k = \exp(2\pi(g_k + if_k)\Delta t) \tag{B.4}$$

where Δt is the sampling step and the symbol $*$ represents the complex conjugate. In eq. (B.3) C_k represents the complex amplitude of the kth sinusoid at the complex frequency given by $f_k - ig_k$ and i is $\sqrt{-1}$. The time-series (u_i) is defined as the sequence satisfying the AR equation:

$$\sum_{j=-m}^{m} a_j u_{i-j} = 0 \tag{B.5}$$

where (a_j; $j=-m, \ldots, m$) are real AR coefficients. An exhaustive treatment about a_j coefficients estimations is reported in Hori et al. (1989), Fukao and Suda (1989) and Kumazawa et al. (1990). Briefly, a way to compute the coefficients a_j that satisfy eq. (B.5) is the minimization of the functional S:

$$S = \sum_{i=N+m}^{N-m} \left(\sum_{j=-m}^{m} a_j x_{i-j} \right)^2 \tag{B.6}$$

under the condition:

$$\sum_{j=-m}^{m} a_j^{\,2} = 1 \qquad \text{(B.7)}$$

This minimization problem leads to an eigenvalue problem where coefficients a_j are the eigenvectors corresponding to minimum eigenvalues. Now, once the a_j are calculated, the Sompi characteristic equation is defined as:

$$\sum_{j=-m}^{m} a_j z^{-j} = 0 \qquad \text{(B.8)}$$

The roots z_k and z_k* of eq. B.8 give the complex frequencies expressed in eq. B.4. Let (x_i) a time-series, Sompi method extracts m wave elements characterized by a complex frequency $f_k - ig_k$ where f_k is the frequency, g_k is the growth rate.

Appendix C

Data transformation

In Chapter 1 we introduced how the all data mining techniques suffer the *GIGO* problem, for which if the input data is not good, also the results will be bad. Data transformation is a preprocessing step on data mining which aims to avoid that data representation "fakes" the real information.

Data normalization

The normalization (or standardization), used for quantitative data types (see Section 2.1), is a typical data transformation. Let us suppose, for example, to calculate the distance between two dataset objects, which attributes range over different scales (Fig. C.1). If we calculate the Euclidean distance (Eq. 2.13) between them, the attribute defined on the greater range will be dominant in the computation of the distance value, of course. The problem is that all variables have the same representation, but not the same valence, and then we may consider an attribute more important than others, even when we do not want. We so need to homogenize all attributes, giving to all the same weight.
A common used normalization technique is the *z-score normalization* that, for each attribute, subtracts its mean value (calculated on the dataset), and divides it for the attribute standard deviation (Gan et al., 2007):

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \qquad \text{(C.1)}$$

The new dataset attributes will have mean = 0 and variance = 1.

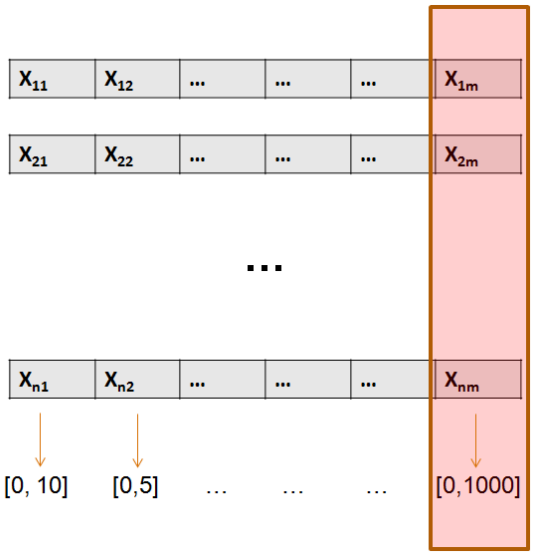


Fig. C.1. An example dataset, where attributes values range on different scales.

Another common used technique is the *min-max normalization*, which subtracts to each attribute the minimum value on it, and divides it for the range amplitude (maximum minus minimum value), to obtain the same range for each attribute, between 0 and 1 (Gan et al., 2007):

$$x_{ij}' = \frac{x_{ij} - \min_j}{\max_j - \min_j} \qquad \text{(C.2)}$$

When attribute values are not arranged on a linear scale, (i.e. the ratio-scaled attributes, Section 2.1), but have exponential trend, is better to make a logarithmic transformation (Han and Kamber, 2000):

$$y_{ij} = \log(x_{ij}) \qquad \text{(C.3)}$$

Another kind of transformation consists in the reduction of the number of attributes describing the dataset objects. The target is to input only essential attributes needed from data mining algorithms to work efficiently and effectively. There are two main ways to obtain this: the *feature selection* and the *feature extraction*.

The *information gain* (Han and Kamber, 2000; Quinlan, 1986) is one of the principal tools for feature selection, especially for classification by decision trees. It bases on entropy concept (Shannon and Weaver, 1949) and calculates the relevance of each attribute for the classification. Let be the dataset divided into k classes, the dataset entropy is computed as:

$$Info(D) = -\sum_{i=1}^{k} p_i \log_2 p_i \qquad \text{(C.4)}$$

where p_i is the probability of a dataset object to belong to a specific class C_i. To calculate the information associated to an attribute A, let us suppose to divide its interval value into v intervals $\{a_1, a_2, \ldots, a_v\}$, and let us create a dataset partition $\{D_1, D_2, \ldots, D_v\}$, for which D_i contains all tuples having a value belonging to the interval a_i for the attribute A. The information associated to the attribute A will be:

$$Info_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} Info(D_j) \qquad \text{(C.5)}$$

where $Info(D_j)$ is the entropy value of the partition D_j. The lower the value, the greater its informative contribution for the classification. The information gain of an attribute is computed in the following way:

$$Gain(A) = Info(D) - Info_A(D) \qquad \text{(C.6)}$$

The attribute A_i with higher $Gain(A_i)$ is more discriminant for classification.

The most known method for the feature extraction is the *PCA* (*Principal Component Analysis*). Let be the dataset represented by a $N \times m$ matrix (Eq. 2.1) where each object is represented by m attributes, constituting the *feature space*. Geometrically each attribute is a dimension (or reference axis) of this space (Fig. C.2). *PCA* aims to perform a rotation of the reference axes so that they maximize the variance of each dimension.

The first principal component is the dimension (or attribute) with maximum variance, the second principal component, orthogonal to the first, is the dimension with second maximum variance, and so on. A dimensionality reduction technique consists into select the first k components, with $k<<m$, which contains the main information about the data structure (see Section 3.3).

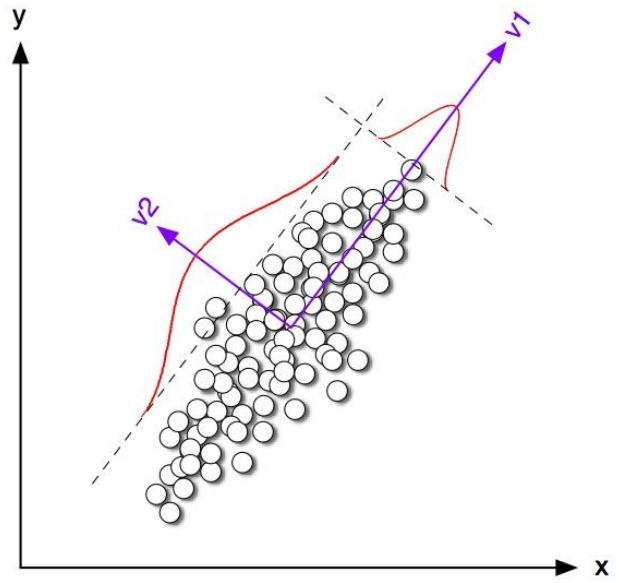


Fig C.2. x and y correspond to original features. v1 and v2 are the new axes corresponding to the new PCA features (image from http://www.cs.cornell.edu/courses/cs322/2008sp/schedule.html).

A typical procedure for the principal components analysis is the following, and is also known as *Karhunen-Loève transformation* (there exists other principal components analysis techniques such as the *SVD*, see Section 3.3):

1. Input are normalized data, so that all attributes range in the same interval values. This step ensures that attributes defined on large intervals do not dominate on small intervals defined attributes.
2. Calculate the covariance matrix (Appendix A) on normalized dataset.
3. Calculate eigenvectors and eigenvalues of the covariance matrix. Eigenvectors are the principal components: they are orthogonal with each

other, and constitute the base of normalized data. Their linear combination with the relative eigenvalues allow to obtain the original input data.

4. Principal components are sorted according to the relative eigenvalues (which represent their variance): the larger is the value, the more important is the component.
5. Since components are sorted in decreasing mode, data dimensionality can be reduced by deleting less important components, i.e. with smaller variance. The data reconstruction, using only more important components, appears to be a good approximation of original data.

Appendix D

Regression

Regression is a widespread technique for prediction of quantitative variables. It aims to approximate the function generating observations. The prediction is validated calculating the error between the model predicted values and the true values. Regression models are classified in *linear* and *non linear*. The linear regression model studies the relation between two variables, trying to find a possible dependence and its nature. In other words, the linear regression tries to plot into a graph a straight line interpolating points (the observations), whose attributes are the two variables values, so that the line minimizes the distance from points (Fig. D.1).

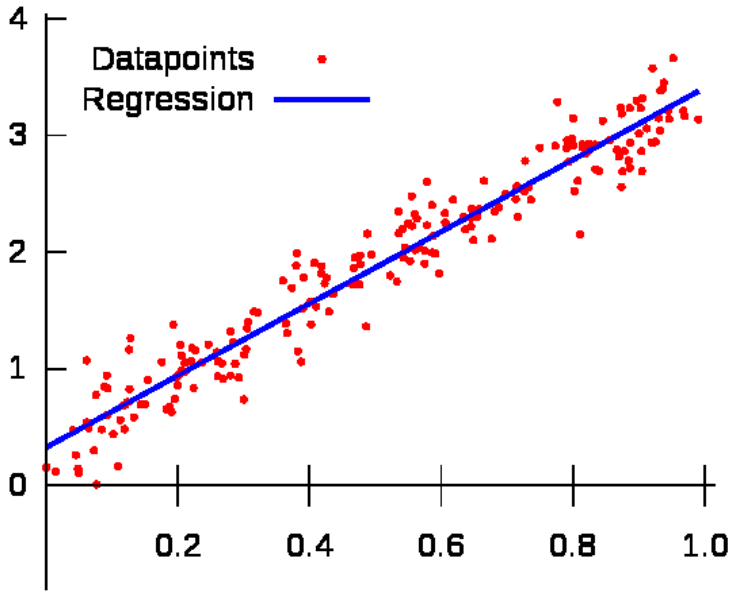


Fig. D.1. Example of linear regression. Image from http://it.wikipedia.org/wiki/Regressione_lineare.

A regression function can be represented through the following formula:

$$y = a + bx + \varepsilon$$

where y is the dependent variable, x is the independent variable, a is the y-intercept of the line, b is the slope and ε is the error term and represents the distance between the line and each point (the residual). a and b are the *regression coefficients*.

The goal of the linear regression is to minimize the value of ε, by choosing the appropriate values for a and b. There are several methods to estimate these by observing x and y values. The most used is the *means square error* method, which calculates the line minimizing the sum of the squares of the residuals. In analytical terms the problem consists on the minimization of the following function:

$$S = S(a,b) = \sum_{i=1}^{n} \varepsilon_i^{\,2} = \sum_{i=1}^{n} \left(y_i - a - bx_i\right)^2 \qquad \text{(D.1)}$$

where n is the number of observation, and then:

$$\{a,b\} = \underset{a,b}{\operatorname{argmin}}\, S(a,b) \qquad \text{(D.2)}$$

The solution can be calculated by equating to zero the partial derivatives of S with respect to a and b:

$$\frac{\partial S}{\partial a} = -2\sum_{i=1}^{n}\left(y_i - a - bx_i\right) = 0 \quad \rightarrow \quad \sum_{i=1}^{n} y_i = an + b\sum_{i=1}^{n} x_i \qquad \text{(D.3)}$$

$$\frac{\partial S}{\partial b} = -2\sum_{i=1}^{n}\left(y_i - a - bx_i\right)x_i = 0 \quad \rightarrow \quad \sum_{i=1}^{n} x_i y_i = a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} x_i^{\,2} \qquad \text{(D.4)}$$

Dividing by n the Equations (D.3-4) and making the system with them, we obtain:

$$\begin{cases} \overline{y} = a + b\overline{x} \\ \frac{1}{n}\sum_{i=1}^{n} x_i y_i = a\overline{x} + \frac{b}{n}\sum_{i=1}^{n} x_i{}^2 \end{cases} \rightarrow \begin{cases} a = \overline{y} - b\overline{x} \\ \frac{1}{n}\sum_{i=1}^{n} x_i y_i = a\overline{x} + \frac{b}{n}\sum_{i=1}^{n} x_i{}^2 \end{cases} \rightarrow$$

$$\rightarrow \begin{cases} a = \overline{y} - b\overline{x} \\ \frac{b}{n}\sum_{i=1}^{n} x_i{}^2 + \overline{x}\overline{y} - b(\overline{x})^2 - \frac{1}{n}\sum_{i=1}^{n} x_i y_i = 0 \end{cases} \rightarrow \begin{cases} a = \overline{y} - b\overline{x} \\ b = \dfrac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i - \overline{x}\overline{y}}{\frac{\sum_{i=1}^{n} x_i{}^2}{n} - (\overline{x})^2} \end{cases} \quad \text{(D.5)}$$

By defining:

$$S_{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \overline{x}\overline{y} \quad \text{(D.6)}$$

$$S_{xx} = \frac{\sum_{i=1}^{n} x_i{}^2}{n} - (\overline{x})^2 \quad \text{(D.7)}$$

we can calculate the values for *a* and *b* by using Equations (D.5-7):

$$b = \frac{S_{xy}}{S_{xx}} \quad \text{(D.8)}$$

$$a = \overline{y} - b\overline{x} \quad \text{(D.9)}$$

The applied method provides the parameters of the line best approximating data, but does not provide the correlation coefficient. It is possible to determine it with the formula in Eq. 2.12.

The linear regression is efficient only when data are linearly dependent. It presents some limits, such as the sensitivity to outliers, and the assumption of considering the errors as belonging to a Gaussian distribution. The *Generalized Linear Models* (*GLM*; Nelder and Wedderburn, 1972) overcome this last limitation, but still maintaining the same analysis structure. A special case of *GLM* is the *logistic regression*, described in (Wooff, 2004). When *y*

variable is not a linearly dependent from *x*, then is the case of *non linear least squares* (*NLS*; George et al.,2003).

Appendix E

Determining the width of histogram bars

A histogram plot is a natural way to represent samples drawn from a univariate variable. It shows the range, central tendencies and shape of the data distribution. However, to make a histogram plot, an important decision is relative to the number of bars (or *bins*) to use.

Sturge's rule

Most statistical software use Sturges (1926) rule which says the data range should be split into k equally spaced classes where:

$$k = \lceil 1 + \log_2 n \rceil \tag{E.1}$$

Sturges' rule is not good when data exhibits skewness or any other non-normality.

Scott's rule

Scott (1979) proposed that the bar width w should be determined as follows:

$$w = 3.49 \frac{\sigma}{\sqrt[3]{n}} \tag{E.2}$$

where σ is the sample standard deviation of the n data values. With this equation the rule tries to minimize the bias in variance of the histogram compared with the data set. It requires knowledge about the distribution form of the data, which we rarely have, so the above equation assumes normality. It is restrictive in practice.

Freedman – Diaconis rule

Freedman and Diaconis (1981) proposed that the bar width w should be determined as follows:

$$w = 2\frac{IQR}{\sqrt[3]{n}} \qquad \text{(E.3)}$$

where IQR is the sample inter-quartile range of the n data values, i.e. the difference between the 75^{th} and 25^{th} percentile of the data.

References

Aggarwal, Wolf, Yu, Procopiuc, Park (1999). "Fast algorithms for projected clustering". In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 61-72. ACM Press.

Aggarwal, Yu (2001). "Outlier detection for high dimensional data". In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, page 46. ACM, 2001.

Agrawal, Faloutsos, Swami (1993). "Efficient similarity search in sequence databases". *Proc. of the 4th Conference on Foundations of Data Organization and Algorithms*, pp. 69–84.

Agrawal, Gehrke, Gunopulos, Raghavan (1998). "Automatic subspace clustering of high dimensional data for data mining applications". In *SIGMOD Record ACM Special Interest Group on Management of Data*, pages 94–105. New York: ACM Press.

Aiuppa, Cannata, Cannavò, Di Grazia, Ferrari, Giudice, Gurrieri, Liuzzo, Mattia, Montalto, Patanè, Puglisi (2010). "Patterns in the recent 2007-2008 activity of Mount Etna volcano investigated by integrated geophysical and geochemical observations", *Geochem. Geophys. Geosyst.* 11, 9, doi: 10.1029/2010GC003168.

Aliotta, Cannata, Cassisi, Montalto, Privitera, Pulvirenti (2010). "Unsupervised clustering of infrasonic events at Mount Etna using DBSCAN and SVM". In: *Geophysical Research Abstracts- EGU General Assembly 2010*. Vienna, 2-7 May 2010, vol. 12.

Aldridge (2006). “Clustering an overview”, in *Lecture Notes in Data Mining*, Berry M.W., and Browne, M. eds., pp. 99-107, World Scientific.

Alparone, Andronico, Lodato, Sgroi (2003). “Relationship between tremor and volcanic activity during the Southeast Crater eruption on Mount Etna in early 2000”, *J. Geophys. Res.* 108(B5), 2241, doi:10.1029/2002JB001866.

Alparone, Cannata, Gresta (2007). “Time variation of spectral and wavefield features of volcanic tremor at Mt. Etna (January–June 1999)”, *J. Volcanol. Geotherm. Res.* 161, 318-332, doi:10.1016/j.jvolgeores.2006.12.012.

Ankerst, Breunig, Kriegel, Sander (1999): “OPTICS: Ordering Points To Identify the Clustering Structure”. *Proc. ACM SIGMOD ‘99 Int. Conf. on Management of Data*, Philadelphia.

Antolik, Nadeau, Aster, McEvilly (1996). “Differential analysis of coda Q using similar microearthquakes in seismic gaps, part 2: Application to seismograms recorded by the Parkfield high resolution seismic network”, *Bull. Seismol. Soc. Am*. 86, 890-910.

Bailey, Elkan (1995). “Unsupervised learning of multiple motifs in biopolymers using expectation maximization”. *Machine Learning Journal 21*, 51–80.

Baum, Petrie, Soules, Weiss (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains", *Ann. Math. Statist.*, vol. 41, no. 1, pp. 164–171, 1970.

Beckmann, Kriegel, Schneider, Seeger (1990). “The R*-tree: An Efficient and Robust Access Method for Points and Rectangles”. *Proc. ACM SIGMOD Int. Conf. on Bentley Management of Data*, Atlantic City, NJ. p. 322-331.

Behncke, Neri (2003). “Cycles and trends in the recent eruptive behaviour of Mount Etna (Italy)”, *Can. J. Earth Sci.*, 40, pp. 1405–1411.

Bentley (1975). "Multidimensional binary search trees used for associative searching". *Communications of ACM*. Vol. 18., 509-517.

Berkhin (2002). "Survey of clustering data mining techniques". *Technical report*, *Accrue Software*, San Jose, CA.

Berndt, Clifford (1994). "Using dynamic time warping to find patterns in time series", *AAAI-94 workshop on knowledge discovery in databases*, 229–248.

Berry, M. J. A. and G. Linoff (1997). *Data Mining Techniques for Marketing, Sales and Customer Support*. New York: John Wiley & Sons.

Bharucha-Reid (1960). *Elements of the Theory of Markov Processes and Their Applications*. New York: McGraw-Hill, 1960.

Bonaccorso, Caltabiano, Currenti, Del Negro, Gambino, Ganci, Giammanco, Greco, Pistorio, Salerno, Spampinato, Boschi (2011a). "Dynamics of a lava fountain revealed by geophysical, geochemical and thermal satellite measurements: The case of the 10 April 2011 Mt Etna eruption", *Geophys. Res. Lett.* 38, L24307, doi:10.1029/2011GL049637.

Borg, Groenen (2005). *Modern Multidimensional Scaling: theory and applications* (2nd ed.). New York: Springer-Verlag. pp. 207–212.

Box, Jenkins, Reinsel (1994). *Time Series Analysis: Forecasting & Control*. Prentice Hall, 3rd edition.

Branca, Del Carlo (2005). "Types of eruptions of Etna volcano AD 1670-2003: implications of short-term eruptive activity", *Bull Volc*, 67, 732-742.

Brecheisen, Kriegel, Kröger, Pfeifle (2004). "Visually mining through cluster hierarchies". *Proc. SIAM Int. Conf. on Data Mining* (*SDM'04*), Lake Buena Vista.

Breiman, Friedman, Olshen, Stone (1984). *Classification and Regression Trees*. Wadsworth International Group.

Breunig, Kriegel, Ng, Sander (2000). "LOF: identifying density-based local outliers". *Sigmod Record*, 29(2):93-104.

Brink (2012). "A (probably) exact solution to the Birthday Problem", *Ramanujan Journal*, doi: 10.1007/s11139-011-9343-9.

Brown, Beroza, Shelly (2008). "An autocorrelation method to detect low frequency earthquakes within tremor", *Geophys. Res. Lett*. 35, L16305, doi:10.1029/2008GL034560

Burges (1998). "A tutorial on support vector machines for pattern recognition", *Data Min. Knowl. Discov.*, 2, 121–167.

Calvari, Salerno, Spampinato, Gouhier, La Spina, Pecora, Harris, Labazuy, Biale, Boschi (2011). "An unloading foam model to constrain Etna's 11–13 January 2011 lava fountaining episode", *J. Geophys. Res.* 116, B11207, doi:10.1029/2011JB008407.

Cannata, Catania, Alparone, Gresta (2008). "Volcanic tremor at Mt. Etna: inferences on magma dynamics during effusive and explosive activity", *J. Volc. Geotherm. Res.*, 178, doi:10.1016/j.jvolgeores.2007.11.027.

Cannata, Montalto, Privitera, Russo, Gresta, (2009a). "Tracking eruptive phenomena by infrasound: May 13, 2008 eruption at Mt. Etna", *Geophys. Res. Lett.*, **36**, doi:10.1029/2008GL036738.

Cannata, Montalto, Privitera, Russo (2009b). "Characterization and location of infrasonic sources in active volcanoes: Mt. Etna, September–November 2007", *J. geophys. Res.*, **114**, doi:10.1029/2008JB006007.

Cannata, Giudice, Gurrieri, Montalto, Alparone, Di Grazia, Favara, Gresta (2010). "Relationship between soil CO2 flux and volcanic tremor at Mt Etna: implications for magma dynamics". *Env. Earth Sci*. doi:10.1007/s12665-009-0359-z.

Cannata, Montalto, Aliotta, Cassisi, Pulvirenti, Privitera, Patanè (2011a). "Clustering and classification of infrasonic events at Mount Etna using pattern recognition techniques". *Geophysical Journal International*, 185: 253–264.

Cannata, Sciotto, Spampinato, Spina (2011b). "Insights into explosive activity at eruptive fissure closely-spaced vents by infrasound signals: example of Mt. Etna 2008 eruption", *J. Volcanol. Geotherm. Res.* 208, 1-11.

Cantone, Ferro, Pulvirenti, Reforgiato (2005). "Antipole tree indexing to support range search and k-nearest neighbour search in metric spaces". *IEEE Transactions on Knowledge and Data Engineering*. Vol. 17, p. 535-550.

Cassisi, Montalto, Pulvirenti, Aliotta, Cannata (2011a). "Pydbscan un software per il clustering di dati". *Rapporti Tecnici INGV*, n. 182.

Cassisi, Giugno, Montalto, Pulvirenti, Aliotta, Cannata (2011b). "DBStrata: a system for density-based and outlier detection based on stratification". *In Proceedings of the Fourth International Conference on SImilarity Search and APplications (SISAP '11)*. ACM, New York, NY, USA, 107-108.

Cassisi, Aliotta, Cannata, Montalto, Patanè, Pulvirenti, Spampinato (2012a). "Motif discovery on seismic amplitude time series: the case study of Mt. Etna 2011 eruptive activity". *Pure and Applied Geophysics*. 0033-4553, pp. 1-17.

Cassisi, Ferro, Giugno, Pigola, Pulvirenti (2012b). "Enhancing density-based clustering: parameter reduction and outlier detection". *Information Systems*, Elsevier, ISSN 0306-4379, 10.1016/j.is.2012.09.001.

Cassisi, Montalto, Aliotta, Cannata, Pulvirenti (2012c). "Similarity measures and dimensionality reduction techniques for time series data mining". In *Advances in Data Mining Knowledge Discovery and Applications*, Adem Karahoca (Ed.), ISBN: 978-953-51-0748-4, InTech. 10.5772/49941

Chakrabarti, Keogh, Mehrotra, Pazzani (2002). "Locally adaptive dimensionality reduction for indexing large time series databases". *ACM Trans. Database Syst.* 27, 2, pp 188-228.

Chan, Fu (1999). "Efficient time series matching by wavelets". In *proceedings of the 15th IEEE Int'l Conference on Data Engineering*. Sydney, Australia, Mar 23-26. pp 126-133.

Cheng, Fu, Zhang (1999). "Entropy-based subspace clustering for mining numerical data". In *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, San Diego, CA, pages 84–93.

Cheng, Randall, Baldi (2006). "Prediction of protein stability changes for single-site mutations using support vectormachines", *Proteins*, 62, 1125–1132, doi:10.1002/prot.20810.

Chester, Duncan, Guest, Kilburn (1985). *Mount Etna*, Stanford University Press, Stanford, California.

Chouet (1996). "Long-period volcano seismicity: its source and use in eruption forecasting". *Nature* 380, 309–316.

Chui (1992). *An Introduction to Wavelets*. San Diego: Academic Press.

Cooley, Tukey (1965). "An algorithm for the machine calculation of complex Fourier series", *Math. Comput.* 19, 297–301.

Cormen, Leiserson, Rivest (1990). *Introduction to Algorithms*. The MIT Press, McGraw-Hill Book Company.

Costa, Cesar (2001). *Shape analysis and classification: theory and practice*. CRC Press, Boca Raton.

Cox (1927). "A method of assigning numerical and percentage values to the degree of roundness". *J. Paleont.*, 1, pp.61-73.

Daszykowski , Walczak, Massart (2002). "Looking for natural patterns in analytical data. Part 2. Tracing local density with OPTICS". *Journal of Chemical Information and Computer Sciences* - 2002. - Vol. 42. - p. 500-507.

Davies, Bouldin (1979). "A Cluster Separation Measure". *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2): 224.

Dempster, Laird, Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society. Series B (Methodological)* 39 (1): 1–38. JSTOR 2984875. MR 0501537.

Devos, Ruckebusch, Durand, Duponchel, Huvenne (2009). "Support vector machines (SVM) in near infrared (NIR) spectroscopy: focus on parameters optimization and model interpretation", *Phys. Chem. B*, 113, 6031–6040.

Di Salvo, Montalto, Nunnari, Neri, Puglisi (2012). "Multivariate time series clustering on geophysical data recorded at Mt. Etna from 1996 to 2003". *Journal of Volcanology and Geothermal Research.*

Ding, Trajcevski, Scheuermann, Wang, Keogh (2008). "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures", *Proceedings of the VLDB Endowment*, doi: 10.1145/1454159.1454226.

Dryden, Mardia (1998). *Statistical Shape Analysis*. Wiley, Chichester.

Dubes (1993). "Cluster analysis and related issues". *Handbook of Pattern Recognition and Computer Vision*, World Scienti5c Publishing Co., Inc., River Edger, NJ, USA, pp. 3–32.

Duda, Hart, Stork (2001). *Pattern Classification*. John Wiley & Sons.

Endo, Murray (1991). "Real-time seismic amplitude measurement (RSAM): A volcano monitoring and prediction tool", *Bulletin of Volcanology* 53, 533-545.

Ertöz, Steinbach, Kumar (2003). "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data". In *SIAM international conference on data mining*. Volume 47.

Ester, Kriegel, Sander, Xu (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". *In Proceedings of the 2nd ACM SIGKDD*, pages 226–231, Portland, Oregon, USA.

Fahim, Saake, Salem, Torkey, Ramadan (2009a). "Enhanced density based spatial clustering of application with noise". In *Proceedings of the 2009 International Conference on Data Mining*, pp. 517-523.

Fahim, Saake, Salem, Torkey, Ramadan (2009b). "Improved dbscan for spatial databases with noise and di_erent densities". *Georgian Electronic Scientific Journal: Computer Science and Telecommunications*, 3:53-60.

Faloutsos, Ranganathan, Manolopoulos (1994). "Fast subsequence matching in time-series databases". *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*. Minneapolis.

Faloutsos, Jagadish, Mendelzon, Milo (1997). "A signature technique for similarity-based queries". In *Proceedings of the SEQUENCES 97* (Positano-Salerno, Italy).

Fayyad, Piatetsky-Shapiro, Smyth (1996). "From Data Mining to Knowledge Discovery in Databases". *AI Magazine*, 17(3):37-54.

Ferretti, Massa, Solarino (2005). "An improved method for the recognition of seismic families: application to the Garfagnana-Lunigiana area (Italy)". *Bull. Seism. Soc. Am.* 95, 1903–1915.

Fink, Pratt (2004). "Indexing of compressed time series**".** In Mark Last, Abraham Kandel, and Horst Bunke, editors, *Data Mining in Time Series Databases*, pages 43-65. World Scientific, Singapore.

Firstov, Kravchenko (1996). "Estimation of the amount of explosive gas released in volcanic eruptions using airwaves", *Volcanol. Seismol.*, 17, 547–560.

Fisher (1936). "The Use of Multiple Measurements in Taxonomic Problems". *Annals of Eugenics* **7** (2): 179–188.

Forney (1973). "The Viterbi Algorithm", *Proceedings of the IEEE*, March 1973; 61 (3) : pp. 268 - 278.

Fortunato (2010). "Community detection in graphs". *Physics Reports* 486, pp. 75-174. Eprint arXiv: 0906.0612.

Fraley, Raftery (1999). "MCLUST: Software for Model-Based Cluster Analysis". *Journal of Classification*, 16, 297-306.

Fraley, Raftery (2002). "Model-based clustering, discriminant analysis, and density estimation". *Journal of the American Statistical Association*, 97(458):611–631.

Frawley, Piatetsky-Shapiro, Matheus (1992). "Knowledge Discovery in Databases - An Overview", in *Knowledge Discovery in Databases 1991*, pp. 1--30. Reprinted in AI Magazine, Fall 1992.

Freedman, Diaconis (1981). "On this histogram as a density estimator: L2 theory". *Zeit. Wahr. ver. Geb.*, 57, 453–476.

Friedberg, Insel, Spence (1989), *Linear algebra* (2nd ed.), Englewood Cliffs, NJ 07632: Prentice Hall.

Fukao, Suda (1989). "Core modes of the Earth's free oscillations and structure of the inner core", *Geophys. Res. Lett.*, 16, 401–404.

Gan, Ma, Wu (2007). "Data Clustering: Theory, Algorithms, and Applications", *ASA-SIAM Series on Statistics and Applied Probability,* volume 20.

George, Seber, Wild (2003). *Nonlinear Regression*. Wiley-Interscience.

Gibbons, and Ringdal (2006). "The detection of low magnitude seismic events using array-based waveform correlation", *Geophys. J. Int.* 165, 149–166.

Golub, Van Loan (1996). *Matrix Computations*, 3rd edition. Baltimore, MD: Hopkins University Press.

Gonzalez (1985). "Clustering to minimize the maximum intercluster distance". *Theoret. Computer Science*, 38(1985), 293-306.

Graps (1995). *An introduction to wavelets*. IEEE.

Greenwood, Nikulin (1996). *A guide to chi-squared testing*. Wiley, New York.

Gresta, Ripepe, Marchetti, D'Amico, Coltelli, Harris, Privitera (2004). "Seismoacoustic measurements during the July–August 2001 eruption at Mt. Etna volcano, Italy". *J. Volc. Geotherm.Res.*, 137, 219–230.

Guha, Rastogi, Shim (1998). "CURE: A clustering algorithm for large databases". *ACM SIGMOD International Conference on Management of Data*, pages 73–84.

Guha, Rastogi, Shim (1999). "Rock: a robust clustering algorithm for categorical attributes". *In Proc. of the 15th Int'l Conf. on Data Eng.*, 1999.

Hagerty, Schwartz, Garces, Protti (2000). "Analysis of seismic and acoustic observations at Arenal Volcano, Costa Rica, 1995–1997". *J. Volc.Geotherm. Res.*, 101, 27–65.

Halkidi, Batistakis, Vazirgiannis (2001). "On clustering validation techniques". *Journal of Intelligent Information Systems*, Vol. 17 (2001), pp. 107-145.

Han, Kamber (2000). *Data Mining: Concepts and Techniques (2nd Edition),* Morgan Kaufmann, San Francisco, CA.

Hastie, Tibshirani, Friedman (2002). *The Elements of Statistical Learning*, p. 533, Springer, New York.

Hawkins, Kass, (1982). "Automatic Interaction Detection", in *Topics in Applied Multivariate Analysis*, Hawkins, Douglas M. (ed), Cambridge University Press, Cambridge, pp. 269–302.

Hentschel, Page (2003). "Selection of Descriptors for Particle Shape Characterization". *Part. Part. Syst. Charact*. 20 (2003) 25 ± 38.

Hinneburg, Keim (1998): "An Efficient Approach to Clustering in Large Multimedia Databases with Noise". *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, New York.

Holsheimer, Siebes (1994). *Data Mining: The Search for Knowledge in Databases*. pp. 1-78.

Hori, Fukao, Kumazawa, Furumotom, Yamamoto (1989). "A new method of spectral analysis and its application to the Earth's free oscillations: the 'Sompi' method", *J. geophys. Res.*, 94(B6), 7535–7553.

Hothersall (2004). *History of Psychology* (4th Edition), McGraw-Hill.

Hsu, Lin (2002). "A comparison of methods for multi-class support vector machines", *IEEE Trans. Neural Netw.*, 13, 415–425.

Hsu, Chang Lin (2007). *A practical guide to support vector classification*, http://www.csie.ntu.edu.tw/~cjlin/papers/ guide/guide.pdf (last accessed 2011 January 31).

Hwanjo, Yang, Han (2003). "Classifying large data sets using SVMs with hierarchical clusters", in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC.

Inselberg, Dimsdale (1990). "Parallel coordinates: A tool for visualizing multidimensional geometry". In *VIS '90: Proceedings of the 1st conference on visualization*, pages 361–378. Los Alamitos, CA: IEEE Computer Society.

ISO 9276-6 (2008). *Representation of results of particle size analysis -- Part 6: Descriptive and quantitative representation of particle shape and morphology*.

Itakura (1975). "Minimum prediction residual principle applied to speech recognition". *IEEE Trans Acoustics Speech Signal Process*. ASSP 23:52–72.

Jain, Dubes (1988). *Algorithms for Clustering Data*. Prentice-Hall Englewood Cliffs, N.J.

Jain, Murty, Flynn (1999). "Data clustering: A survey". *ACM Comput. Surv.*, 31:264–323.

Jin, Tung, Han, Wang (2006). "Ranking outliers using symmetric neighborhood relationship". *Advances in Knowledge Discovery and Data Mining*, pp. 577-593.

Johnson, Lees (2000). "Plugs and chugs: seismic and acoustic observations of degassing explosions at Karymsky, Russia and Sangay, Ecuador", *J. Volc. Geotherm. Res.*, 101, 67–82.

Johnson, Lees, Varley (2010). "Characterizing complex eruptive activity at Santiaguito, Guatemala using infrasound semblance in networked arrays", *J. Volc. Geotherm. Res.*, 199, doi:10.1016/j.jvolgeores.2010.08.005.

Jones, Johnson, Aster, Kyle, McIntosh (2008). "Infrasonic tracking of large bubble bursts and ash venting at Erebus volcano, Antarctica", *J. Volc. Geotherm. Res.*, 177, doi:10.1016/j.jvolgeores.2008.02.001.

Karypis, Han, Kumar (1999). "CHAMELEON: Hierarchical clustering using dynamic modeling". *IEEE Computer*, 32(8):68–75.

Kay (1988). *Modern Spectral Estimation, Theory and Application*, Prentice-Hall, Englewood Cliffs, NJ.

Kass (1980). "An Exploratory Technique for Investigating Large Quantities of Categorical Data", *Applied Statistics*, Vol. 29, No. 2, pp. 119–127.

Keogh, Chakrabarti, Pazzani, Mehrotra (2000). "Dimensionality reduction for fast similarity search in large time series databases". *Journal of Knowledge and Information Systems.*

Keogh, Chu, Hart, Pazzani (2001). "An Online Algorithm for Segmenting Time Series". In *Proc. IEEE Intl. Conf. on Data Mining*, pp. 289-296, 2001.

Keogh, Kasetty (2002). "On the need for time series data mining benchmarks: a survey and empirical demonstration". *Proceedings of the Eighth ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 102-111.

Keogh, Ratanamahatana (2002). "Exact indexing of dynamic time warping". In *proceedings of the 26th Int'l Conference on Very Large Data Bases*. Hong Kong. pp 406-417.

Keogh, Chu, Hart, Pazzani (2004). "Segmenting time series: a survey and novel approach". In: Last, M., Kandel, A., Bunke, H. (Eds.), *Data mining in time series database*. World Scientific Publishing Company, pp. 1-21.

Kohonen (2001). *Self-Organizing Maps* (3rd edition). Springer Series in Information Sciences, Vol. 30, Springer, New York.

Korn, Jagadish, Faloutsos (1997). "Efficiently supporting ad hoc queries in large datasets of time sequences". *Proceedings of SIGMOD '97*, Tucson, AZ, pp 289-300.

Kumazawa, Imanishi, Fukao, Furumoto, Yamamoto (1990). "A theory of spectral analysis based on the characteristic property of a linear dynamic system", *Geophys. J. Int.*, 101, 613–630.

Kuo, Rollings, Lynch (1998). "Morphological study of coarse aggregates using image analysis". *Journal of Materials in Civil Engineering*, 10(3), 135-142.

Lahr, Chouet, Stephens, Power, Page (1994). "Earthquake classification, location, and error analysis in a volcanic environment: implications for the magmatic system of the 1989–1990 eruptions" at *Redoubt Volcano*, Alaska. J. Volcanol. Geotherm. Res. 62, 137–152.

Langer, Falsaperla, Masotti, Campanini, Spampinato, Messina, (2009). "Synopsis of supervised and unsupervised pattern classification techniques applied to volcanic tremor data at Mt Etna, Italy", *Geophys. J. Int.*, 178, doi:10.1111/j.1365–246X.2009.04179.x.

Lawrence, Altschul, Boguski, Liu, Neuwald, Wootton (1993). "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment". *Science 262*, 208–14.

Lesage (2008). "Automatic estimation of optimal autoregressive filters for the analysis of volcanic seismic activity", *Nat. Hazards Earth Syst. Sci.*, 8, 369–376.

Lie Hetland (2004). "A survey of recent methods for efficient retrieval of similar time sequences", in *Data Mining in Time Series Databases* (eds. Last M., Kandel, A., Bunke H.), World Scientific, Singapore.

Lin, Keogh, Wei, Lonardi (2007). "Experiencing SAX: a novel symbolic representation of time series". Data Mining Knowledge Discovery, 15(2).

Lo Castro, Andronico (2008). "Operazioni di base per la misura della distribuzione granulometrica di particelle vulcaniche tramite il CAMSIZER". *Rapporti Tecnici INGV*, n. 79.

Lo Castro, Andronico, Cassisi, Montalto, Prestifilippo (2011a). "Implementazione di una nuova procedura per caratterizzare la forma di particelle mediante misure al CAMSIZER e algoritmi di clustering". *QUADERNI DI GEOFISICA*, ISSN: 1590-2595.

Lo Castro, Andronico, Cassisi, Montalto, Prestifilippo, Beckmann, Dueffels, Westermann (2011b). "A New Technique for the Characterization of Volcanic Particle Shape by CAMSIZER and Cluster Algorithms". In: *Conferenza A. RITTMANN "Per Giovani Ricercatori"*. Nicolosi (Catania), 7-9 Giugno 2011.

Lodder , Hieftje (1988). "Quantile analysis: a method for characterizing data distributions". Applied Spectroscopy 42:1512–1520.

MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations", *Proceedings of 5-th Berkeley Symposium on*

Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297.

Marchetti, Ripepe, Ulivieri, Caffo, Privitera (2009). "Infrasonic evidences for branched conduit dynamics at Mt. Etna volcano, Italy", *Geophys. Res. Lett.*, **36**, L19308, doi:10.1029/2009GL040070.

Marple (1987). *Digital Spectral Analysis with Applications*, Prentice Hall, Englewood Cliffs.

Mars, Lacoume, Mari, Glangeaud (2004). "Traitement du Signal Pour Géologues et géophysiciens", *Techniques avancées*, Vol. 3, Technip.

Mason, Handscomb (2003). *Chebyshev Polynomials*. Chapman & Hall.

Masotti, Campanini, Mazzacurati, Falsaperla, Langer, Spampinato (2008). "TREMOrEC: a software utility for automatic classification of volcanic tremor", *Geochem. Geophys.Geosyst.*, 9(1), Q04007, doi:10.1029/2007GC001860.

McNutt (2000). "Seismic Monitoring", In *Encyclopedia of Volcanoes* (eds. Sigurdsson, H., B. Houghton, S.R. McNutt, H. Rymer, and J. Stix) (Academic Press, San Diego, CA) pp. 1095-1119.

McNutt (2005). "Volcanic seismology", *Annu. Rev. Earth Planet. Sci*. 32, 461–491.

Montalto, Cannata, Privitera, Gresta, Nunnari, Patanè (2010). "Towards an automatic monitoring system for infrasonic events at Mt. Etna: strategies for source location and modeling", *Pure appl. Geophys.*, 167, doi:10.1007/s00024–010-0051-y.

Montalto, Aliotta, Cannata, Cassisi (2012). "Segmentazione delle serie temporali nell'analisi dei dati: un esempio di applicazione a dati sismo-vulcanici". *Rapporti Tecnici INGV*, n. 224.

Moran, Malone, Qamar, Thelen, Wright, Caplan-Auerbach (2008). "Seismicity associated with renewed dome building at Mount St. Helens, 2004–2005", In *A Volcano Rekindled: The Renewed Eruption of Mount St. Helens*, 2004–2006 (eds. D. R. Sherrod, W. E. Scott, and P. H. Stauffer) (USGS Professional Paper 1750) pp. 27–50.

Mueen, Keogh, Zhu, Cash, Westover (2009). "Exact Discovery of Time Series Motif". *SIAM International Conference on Data Mining SDM 2009*.

Muller (1967). *Methods in sedimentary petrology*. Hafner, New York.

Nagesh, Goil, Choudhary (2001). "Adaptive grids for clustering massive data sets". In *Proceedings of the 1st SIAM International Conference on Data Mining (SDM)*, Chicago, IL.

Neidell, Taner (1971). "Semblance and other coherency measures for multichannel data". *Geophysics*, 36, 482–497, doi:10.1190/1.1440186.

Nelder, Wedderburn (1972). "Generalized Linear Models". *Journal of the Royal Statistical Society. Series A (General)* (Blackwell Publishing) 135 (3): 370–384.

Ng, Han (1994). "Efficient and Effective Clustering Methods for Spatial Data Mining". *Proc. 20th Int. Conf. on Very Large Data Bases*, 144-155. Santiago, Chile.

Ng, Jordan, Weiss (2002). "On spectral clustering: analysis and an algorithm". In *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani (Eds.), 14 (pp. 849 – 856). MIT Press.

Ng, Cai (2004): "Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials". *SIGMOD 2004*.

Noble (2004). Support vector machine applications in computational biology, in *Kernel Methods in Computational Biology*, pp. 71–92, eds Schölkopf, B., Tsuda, K., Vert, J., The MIT Press, Cambridge, MA.

Palla, Derényi, Farkas, Vicsek (2005). "Uncovering the overlapping community structure of complex networks in nature and society". *Nature* 435, 814.

Papadimitriou, Kitagawa, Gibbons, Faloutsos (2003). "LOCI: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings 19th International Conference* on, pp. 315-326. IEEE.

Parsons, Haque, Liu (2004). "Subspace clustering for high dimensional data: A review". *SIGKDD Explorations*, 6(1):90–105.

Patanè, Di Grazia, Cannata, Montalto, Boschi (2008). "The shallow magma pathway geometry at Mt. Etna volcano", *Geochem. Geophys. Geosyst*. 9, 12, doi:10.1029/2008GC002131.

Pavlidis (1976). "Waveform segmentation through functional approximation". *IEEE Trans.Comput. C-22*, 7 (July).

Perng, Wang, Zhang, Parker (2000). "Landmarks: a newmodel for similarity-based pattern querying in time series databases". *Proc.2000 ICDE*, pp. 33–42.

Pettijohn, Potter, Siever (1972). *Sand and Sandstone*. Springer-Verlag, Berlin, pp. 618. Wadell.

Pevzner, Sze (2000). "Combinatorial approaches to finding subtle signals in DNA sequences", *Proc. 8th Int. Conf. Intelligent Systems for Molecular Biology*, 269–78.

Qamar, Malone, Moran, Steele, Thelen (2008). "Near-real-time information products for Mount St. Helens—tracking the ongoing eruption", In *A Volcano*

Rekindled: The Renewed Eruption of Mount St. Helens, 2004–2006 (eds. D. R. Sherrod, W. E. Scott, and P. H. Stauffer) (USGS Professional Paper 1750) pp. 61–70.

Quinlan (1986). "Induction of decision trees". *Machine Learning*, 1:81–106, 1986.

Quinlan (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rabiner (1989). "Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". *Proceedings of the IEEE*, 77 (2), p. 257–286.

Ram, Sharma, Jalal, Agrawal, Singh (2009). "An enhanced density based spatial clustering of applications with noise". In *Advance Computing Conference, 2009. IACC 2009*. IEEE International, pp. 1475-1478.

Russell, Taylor (1937). "Roundness and shape of Mississippi River sands". *J. Geol.*, 45, 225–267.

Ratanamahatana, Lin, Gunopulos, Keogh, Vlachos, Das (2010): "Mining Time Series Data". *Data Mining and Knowledge Discovery Handbook*, pp. 1049-1077.

Riley, Rose, Bluth (2003). "Quantitative shape measurements of distal volcanic ash". *Journal of Geophysical Research*, vol. 108, no. b10, 2504, doi:10.1029/2001jb000818.

Ripepe, Poggi, Braun, Gordeev (1996)." Infrasonic waves and volcanic tremor at Stromboli", *Geophys. Res. Lett.*, 23, 181–184.

Ripepe, Marchetti (2002). "Array tracking of infrasonic sources at Stromboli volcano", *Geophys. Res. Lett.*, **29**(22), 2076, doi:10.1029/2002GL015452.

Rousseeuw, Kaufman (1990). *Finding groups in data*. JohnWiley & Sons, Inc, New York.

Rowe, Aster, Kyle, Dibble, Schlue (2000). "Seismic and acoustic observations at Mount Erebus Volcano, Ross Island, Antarctica, 1994–1998", *J. Volc. Geotherm. Res.*, 101, 105–128.

Ruiz, Lees, Johnson (2006). "Source constraints of Tungurahua volcano explosion events", *Bull. Volcanol.*, 68, 480–490.

Sakoe, Chiba (1978). "Dynamic programming algorithm optimization for spoken word recognition". *IEEE Trans Acoustics Speech Signal Process*. ASSP 26:43-49.

Salvador, Chan, (2004). "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms". *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pp. 576-584.

Sander, Qin, Lu, Niu, Kovarsky (2003). **"**Automatic extraction of clusters from hierarchical clustering representations ". *Advances in Knowledge Discovery and Data Mining*. Springer, p. 567.

Scarpa, Gasparini (1996). "A review of volcano geophysics and volcano-monitoring methods, in Monitoring and mitigation of volcano hazards" (eds. Scarpa, Tilling) (Springer, Heidelberg), pp 3–22.

Scollo, Folch, Costa (2008). "A parametric and comparative study of different tephra fallout models". *Journal of Volcanology and Geothermal Research*, 176, 199–211.

Scollo, Prestifilippo, Spata, D'Agostino, Coltelli (2009). "Monitoring and forecasting Etna volcanic plumes", *Nat. Hazards Earth Syst. Sci*. 9, 1573–1585.

Schaeffer (2007). "Graph Clustering". *Computer Science Review* 1(1):27-64.

Schaff (2008). “Semiempirical statistics of correlation-detector performance”, *Bull. seism. Soc. Am.* 98, 1495–1507.

Schaff (2009). “Broad-scale applicability of correlation detectors to China seismicity”, *Geophys. Res. Lett.* 36, L11301, doi:10.1029/2009GL038179.

Scott (1979). “On optimal and data-based histograms”. *Biometrika*, 66, 605–610.

Scott (2009). "Sturges' rule". *WIREs Computational Statistics* **1**: 303–306.

Shannon,Weaver (1949). *The mathematical theory of communication*. University of Illinois Press, Urbana, IL.

Sheikholeslami, Chatterjee, Zhang (1998). “Wavecluster: A multiresolution clustering approach for very large spatial databases”. In *Proceedings of the 24th Conference on VLDB*, pages 428–439, New York, NY.

Shelly (2010). “Periodic, chaotic, and doubled earthquake recurrence intervals on the deep San Andreas Fault”, *Science* 328, 1385-1388.

Shi, Malik (2000). “Normalized cuts and image segmentation”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (8), 888 – 905.

Shieh and Keogh (2008). “iSAX: Indexing and Mining Terabyte Sized Time Series”. *SIGKDD*, pp 623-631.

Smith (2002). *A Tutorial on Principal Components Analysis* . [Online] Available: http://www.itu.dk/courses/SIGB/F2011/untitled%20folder/Reading/pca-smithTutorial.pdf

Spampinato, Calvari, Oppenheimer, Boschi (2011). "Volcano surveillance using infrared cameras", *Earth Science Reviews*, doi: 10.1016/j.earscirev.2011.01.003.

Spina, Lo Castro, Sciotto, Andronico (2012). "Investigation of 2010 ash emission episodes at Mt Etna by combining volcanological and seismo-acoustic analyses", *EGU General Assembly 2012*, Geophys. Res. Abstr. vol. 14, EGU2012-5534-1, 2012.

Sturges (1926). "The choice of a class-interval". *J. Amer. Statist. Assoc.*, 21, 65–66.

Tan, Steinbach, Kumar (2005). *Introduction to Data Mining* (1st Edition), Addiso

Tang, Fishwick (1991). "Feed-forward neural nets as models for time series forecasting". *Technical Report TR91-008*, University of Florida, Gainesville, FL, USA.
n-Wesley Longman Publishing Co., Inc., Boston, MA.

Tang, Liu (2010). *Community Detection and Mining in Social Media*, Morgan & Claypool Publishers, Synthesis Lectures on Data Mining and Knowledge Discovery.

Tompa, Buhler (2001). "Finding motifs using random projections". *In proceedings of the 5th Int'l Conference on Computational Molecular Biology*. Montreal, Canada, Apr 22-25. pp 67-74.

Ultsch (1993). "Self organized features planes for monitoring and knowledge acquisition of a chemical process". *International Conference on Artificial Neural Networks*, Springer-Verlag,London, pp. 864-867.

Ultsch (2000). "The neuro-data-mine". Symposia on *Neural Computation* (*NC'2000*), Berlin, Germany.

Vadapalli, Valluri, Karlapalem (2006). "A simple yet effective data clustering algorithm". *ICDM '06. Sixth International Conference on Data Mining*.

Vergniolle, Brandeis (1994). "Origin of the sound generated by Strombolian explosions", *Geophys. Res. Lett.*, **21**, 1959–1962.

Viterbi (1967). "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Transactions on Information Theory*, April 1967; IT - 13 (2) : pp. 260 - 269.

Vlachos, Kollios, Gunopulos (2002). "Discovering similar multidimensional trajectories". *Proc. 2002 ICDE*, pp. 673–684.

von Luxburg (2006). "A tutorial on spectral clustering". *Technical Report 149, Max Planck Institute for Biological Cybernetics*.

Von Storch, Zwiers (2001). *Statistical analysis in climate research.* Cambridge Univ Pr. ISBN 0521012309.

Wang, Yang, Muntz (1997). "STING: a statistical information grid approach to spatialdata mining". In *Proceedings of the 23rd Conference on VLDB*, pages 186–195, Athens, Greece.

Wang, Ye, Keogh, Shelton (2008). "Annotating Historical Archives of Images". *JCDL 2008*.

Weston, Watkins (1999). *Multi-class support vector machines*, presented at the *Proceedings ESANN99*, ed. M.Verleysen, Brussels,Belgium.

Withers (1997). *An automated local/regional seismic event detection and location system using waveform correlation*, PhD thesis, New Mexico Tech., Socorro, NM.

Withers, Aster, Young, Beiriger, Harris, Moore, Trujillo (1998). "A comparison of select trigger algorithms for automated global seismic phase and event detection", *Bull. seism. Soc. Am.*, 88, 95–106.

Yang (1993). "A survey of fuzzy clustering", *Math. Comput. Modelling 18*, l-16.

Yi and C. Faloutsos (2000). "Fast Time Sequence Indexing for Arbitrary Lp Norms". *VLDB.*

Wooff (2004). "Logistic Regression: a Self-learning Text, 2nd edn". *Journal of the Royal Statistical Society: Series A* (*Statistics in Society*), 167: 192–194.

Zadeh (1965). "Fuzzy sets". *Information and control*. 8 (3) 338–353.

Zezula, Amato, Dohnal, Batko (2005). "Similarity Search: The Metric Space Approach", *Advances in Database Systems*, 32. Springer.

Zhang, Ramakrishnan, Livny (1997). "BIRCH: A new data clustering algorithm and its applications". *Journal of Data Mining and Knowledge Discovery*, 1(2):141–182.

Zimek (2008). *Correlation Clustering*. PhD Thesis, Ludwig-Maximilians-Universität München, Munich, Germany.