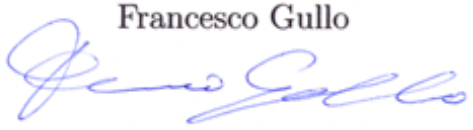
Università della Calabria

Dottorato di Ricerca in
Ingegneria dei Sistemi e Informatica
XXII Ciclo

Ph.D. Thesis

Overcoming Uncertainty and the Curse of Dimensionality in Data Clustering

Francesco Gullo



Director
Prof. Luigi Palopoli

Advisor
Prof. Sergio Greco

Dipartimento di Elettronica, Informatica e Sistemistica
Settore Scientifico Disciplinare: ING-INF/05

Acknowledgments

Several people have contributed to allow this thesis to exist.

I acknowledge my advisor, prof. *Sergio Greco*, for the guidance over these years and the opportunity to carry on the research of this thesis without constraining it.

I would like to thank the two people who shared with me much more time than anyone else: my "de facto" second advisor *Andrea Tagarelli* and my roommate *Giovanni* "*Gianni*" *Ponti*.

Thanks to the "UNICZ team", *Pierangelo Veltri*, *Giuseppe Tradigo*, and *Mario Cannataro*. Besides the collaboration in some works, they have the great merit to have made the days spent in the "wonderful" Catanzaro better days!

I am grateful to *Carlotta Domeniconi* for the chance to experience the US, and, along with *Sean Luke* and *Roberto Avogadri*, for having made such an experience nice and homely.

I cannot forget all the people who, during these years, "housed" the fifth floor of the 41C cube: who was here before my arrival and is still here right now (*Andrea*, *Bettina*, *Cristian*, who they call "*Cristiano*", *Filippo*, *Francesco*, *Irina*, *Lillo*, *Sergio*), who was here but has been "unfaithful" and moved to higher or lower floors (*Ester*, *Luciano*, *Massimo*), and the "new entries" (*Andrea*, *Francesca*, *Giuseppe*). Among them, a special thank goes to *Giovanni Costabile* for having tolerated and satisfied me in every my "technical request"!

Finally, I am going to conclude by writing a sentence which, I am sure, has not ever previously ended any acknowledgment section: LAST BUT NOT LEAST, I want to thank my best friends and, in particular, my family: thanks *mum*, thanks *dad*, thanks *Rita*!

To my family

Abstract

Uncertainty and *the curse of dimensionality* are two crucial problems that usually affect data clustering.

Uncertainty in data clustering may be typically considered at *data level* or *clustering level*. Data level uncertainty is inherently present in the representation of several kinds of data objects from various application contexts (e.g., sensor networks, moving objects databases, biomedicine). This kind of uncertainty should be carefully taken into account in a clustering task in order to achieve adequate accuracy; unfortunately, traditional clustering methods are designed to work only on deterministic vectorial representations of data objects. *Clustering uncertainty* is related to the output of any clustering algorithm. Indeed, the ill-posed nature of clustering leads to clustering algorithms that cannot be generally valid for any input dataset, i.e., their output results are necessarily uncertain. *Clustering ensembles* has been recognized as a powerful solution to overcome clustering uncertainty. It aims to derive a single clustering solution (i.e., the *consensus partition*) from a set of clusterings representing different partitions of the same input dataset (i.e., the *ensemble*). A major weakness of the existing clustering ensembles methods is that they compute the consensus partition by equally considering all the solutions in the ensemble.

The curse of dimensionality in data clustering concerns all the issues that naturally arise from data objects represented by a large set of features and are responsible of poor accuracy and efficiency achieved by traditional clustering methods working on high dimensional data. Classic approaches to the curse of dimensionality include *global* and *local* dimensionality reduction. Global techniques aim at reducing the dimensionality of the input dataset by applying the same algorithm(s) to all the input data objects. Local dimensionality reduction acts by considering subsets of the input dataset and performing dimensionality reduction specific for any of such subsets. *Projective clustering* is an effective class of methods falling into the category of local dimensionality reduction. It aims to discover clusters of objects along with the corresponding *subspaces*, following the principle that objects in the same cluster are close to each other if and only if they are projected onto the subspace associated to that cluster.

The focus of this thesis is on the development of proper techniques for overcoming the crucial problems of uncertainty and the curse of dimensionality arising from data clustering. This thesis provides the following main contributions.

Uncertainty. Uncertainty at a representation level is addressed by proposing:

- *UK-medoids*, which is a new partitional algorithm for clustering uncertain objects, which is designed to overcome efficiency and accuracy issues of some existing state-of-the-art methods;
- *U-AHC*, i.e., the first (agglomerative) hierarchical algorithm for clustering uncertain objects;
- a methodology to exploit U-AHC for clustering *microarray* biomedical data with *probe-level* uncertainty.

Clustering uncertainty is addressed by focusing on the problem of *weighted consensus clustering*, which aims to automatically determine weighting schemes to discriminate among clustering solutions in a given ensemble. In particular:

- three novel *diversity*-based, general schemes for weighting the individual clusterings in a given ensemble are proposed, i.e., *Single-Weighting* (SW), *Group-Weighting* (GW), and *Dendrogram-Weighting* (DW);
- three algorithms, called *WICE*, *WCCE*, and *WHCE*, are defined to easily involve clustering weighting schemes into any clustering ensembles algorithm falling into one of the main classes of clustering ensembles approaches, i.e., *instance-based*, *cluster-based*, and *hybrid*.

The curse of dimensionality. Global dimensionality reduction is addressed by focusing on the *time series* data application context:

- the *Derivative time series Segment Approximation* (DSA) model is proposed as a new time series dimensionality reduction method designed for accurate and fast similarity detection and clustering;
- *Mass Spectrometry Data Analysis* (MaSDA) system is presented; it mainly aims at analyzing *mass spectrometry* (MS) biomedical data by exploiting DSA to model such data according to a time series-based representation;
- DSA is exploited for profiling low-voltage electricity customers.

Regarding local dimensionality reduction, a unified view of projective clustering and clustering ensembles is provided. In particular:

- the novel *Projective Clustering Ensembles* (PCE) problem is addressed and formally defined according two specific optimization formulations, i.e., *two-objective PCE* and *single-objective PCE*;
- *MOEA-PCE* and *EM-PCE* algorithms are proposed as novel heuristics to solve two-objective PCE and single-objective PCE, respectively.

Absolute accuracy and efficiency performance achieved by the proposed techniques, as well as the performance with respect to the prominent state-of-the-art methods are evaluated by performing extensive sets of experiments on benchmark, synthetically generated, and real-world datasets.

Contents

List of Tables

List of Figures

Part I

Preliminaries

1

Introduction

1.1 Knowledge Discovery in Databases, Data Mining, Clustering

Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying novel, valid, potentially useful, and ultimately understandable patterns in data [FPS96]. The term "pattern" refers to a subset of the data expressed in some language or a model exploited for representing such a subset. KDD aims to discover patterns that *(i)* do not result in straightforwardly computing predefined quantities (i.e., non-trivial), *(ii)* can apply to new data with some degree of certainty (i.e., valid), *(iii)* are previously unknown (i.e., novel), *(iv)* provide some benefit to the user or further task (i.e., potentially useful), and *(v)* lead to insight, immediately or after some post-processing (i.e., understandable).

The KDD process is an iterative and interactive sequence of the following main steps (Fig. 1.1):

- *selection*, whose main goal is to create a target data set from the original data, i.e., selecting a subset of variables or data samples, on which discovery has to be performed;
- *preprocessing*, which aims to "clean" data by performing various operations, such as noise modeling and removal, defining proper strategies for handling missing data fields, accounting for time sequence information;
- *transformation*, which is responsible of reducing and projecting the data, in order to derive a representation suitable for the specific task to be performed; it is typically accomplished by involving transformation techniques or methods that are able to find invariant representations for the data;
- *data mining*, which deals with extracting the interesting patterns by choosing *(i)* a specific data mining *method* or *task* (e.g, summarization, classification, clustering, regression, and so on), *(ii)* the proper *algorithm(s)* for performing the task at hand, and *(iii)* the appropriate representation for the output results;

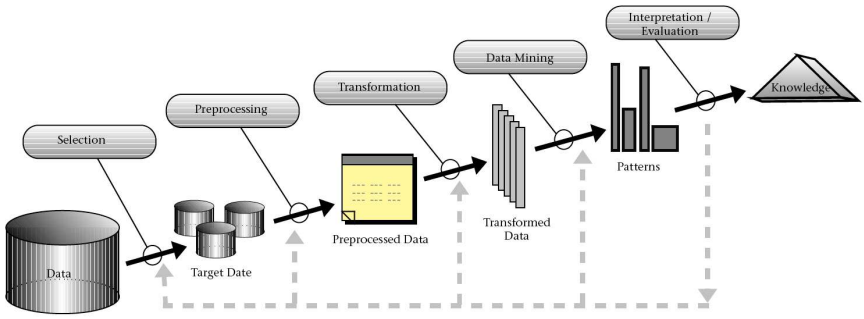


Fig. 1.1. The Knowledge Discovery in Databases (KDD) process

- *interpretation/evaluation*, which is exploited by the user to interpret and extract knowledge from the mined patterns, by visualizing the patterns; this interpretation is typically carried out by visualizing the patterns, the models, or the data given such models and, in case, iteratively looking back at the previous steps of the process.

Data mining represents the "core" step of the KDD process, so much so that the "data mining" and "KDD" terms are often treated as synonyms [HK01]. Data mining tasks are classified into *predictive* and *descriptive* [FPSU96]. Predictive tasks refer to building a model useful for predicting future behavior or values for certain features. These comprise *association analysis*, i.e., discovering *association rules* that show attribute-value conditions occurring frequently together in a given set of data; *classification* and *prediction*, i.e., deriving some models (or functions) which describe data classes or concepts by a set of data objects whose class label is known (i.e., the *training set*); such models have the main goal of being used to predict the class of objects whose class label is unknown as accurately as possible; *deviation detection*, i.e., dealing with *deviations* in data, which are defined as differences between measured values and corresponding references such as previous values or normative values; *evolution analysis*, i.e., detecting and describing regular patterns in data whose behavior changes over time. On the other hand, in a descriptive data mining task, the model built has to describe the data in an understandable, effective, and efficient form. Relevant examples of descriptive tasks are *data characterization*, whose main goal is to summarize the general characteristics or features of a target class of data, *data discrimination*, i.e., a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes, and *clustering*.

Given a set of data objects, clustering aims to identify a finite set of groups of objects, i.e., *clusters*, so that the objects within the same cluster are "sim-

ilar" to each other, whereas the objects belonging to different clusters are "dissimilar". The degrees of (dis)similarity among data objects are computed and evaluated according to a similarity/distance measure that can be either specified by the user or inherently employed in the specific clustering algorithm. In a clustering task, there is no prior knowledge of the class labels associated to the objects to be grouped; according to this feature, clustering is often also referred to as *unsupervised classification*, to emphasize the difference with respect to the (supervised) classification task, in which the class labels of the objects in the training set are known.

A large number of clustering algorithms has been proposed in the literature [JD88, KR90]. Traditionally, these algorithms share to each other the following main features: *(i)* they work on a set of a data objects which are described according to a "deterministic" vectorial representation, i.e., as tuples of *attribute* or *feature* values; *(ii)* they assume and, consequently, guarantee high performances only if the number of features describing each data object is sufficiently small; *(iii)* they make the final decision about the output result without taking into account additional information, possibly given by, e.g., varying parameters and features and/or exploiting different algorithms or similarity measures. Unfortunately, due to recent advances in database applications which lead to a significant increasing of the complexity of the data to be treated, the above features make the traditional clustering algorithms not easily applicable to a lot of newly emerged application contexts. Within this view, major challenges in data clustering are focusing on overcoming issues related to the crucial notions of *uncertainty* and *the curse of dimensionality*.

1.2 Uncertainty

Uncertainty in Data Representation. Handling uncertainty in data management has been recently requiring more and more importance in a wide range of application contexts [Agg09]. Data uncertainty naturally arises from, e.g., implicit randomness in a process of data generation/acquisition, imprecision in physical measurements, application of approximation methods, and data staling. This makes uncertainty inherently present in several application domains. For instance, sensor measurements may be imprecise at a certain degree due to the presence of various noisy factors (e.g., signal noise, instrumental errors, wireless transmission) [CLL06, CKP03]. To address this issue, it is advisable to model sensor data as continuous probability distributions [FGB02, DGM$^+$05]. Another example is given by data representing moving objects, which continuously change their location so that the exact positional information at a given time instant may be unavailable [LHY04]. Moreover, some methods have recently been defined to handle uncertainty in gene expression data [MFNL03, HRC$^+$05, LMLR05]. Further examples come from distributed applications, privacy preserving data mining, and forecasting or other statistical techniques used to generate data attributes [AY09].

Dealing with uncertain data has raised several issues in data management and knowledge discovery. In particular, clustering uncertain data is especially challenging and has been attracting increasing interest in recent years [KP05a, KP05b, CCKN06, NKC+06, S. 07, KLC+08]. Indeed, in order to produce meaningful results, uncertain data clustering algorithms have to necessarily deal with the non-trivial issue of carefully considering and modeling uncertainty. This represents a crucial point, because, if data uncertainty is not effectively taken into account and represented, any clustering algorithm may probably fail in discovering accurate cluster structures.

Clustering Uncertainty. Clustering is typically affected by a different kind of uncertainty, i.e., *clustering uncertainty*, which is somehow related to the output results. Clustering uncertainty is mainly motivated by the "ill-posed" nature of clustering: it is well-known that there do not exist definitely best algorithms, similarity measures, and parameter settings for providing general valid solutions to the clustering problem [JMF99]. Instead, such choices have to be made depending on the specific data to be clustered. Within this view, any clustering result outputted by a clustering algorithm equipped with a specific similarity measure and parameter values cannot be recognized as "certain"; rather, many different configurations of algorithm, similarity measure, and parameter setting that lead to different output results (and, hence, to uncertainty in clustering results) should be in principle taken into account.

Clustering ensembles [SG02], also known as *consensus clustering* or *aggregation clustering*, has recently emerged as a powerful tool to face the issues due to uncertainty in clustering results. In particular, clustering ensembles aims to make a clustering solution more robust against the bias due to the peculiarities of the specific clustering algorithm. Basically, clustering ensembles resorts to the idea of combining multiple classifiers, which has received increased attention in the last years [BK99]. Given a data collection, a set of clustering solutions, or *ensemble*, can be generated by varying one or more aspects, such as the clustering algorithm, the parameter setting, and the number of features, objects or clusters. Given an ensemble, a major goal is to extract a *consensus partition*, i.e., a clustering solution that maximizes some objective function (the *consensus function*) defined by taking into account different information available from the given set of clustering solutions.

1.3 The Curse of Dimensionality

The term "*curse of dimensionality*", as originally coined in [Bel61], refers to the impossibility of optimizing a function of many variables by a brute force search on a discrete multidimensional grid [SEK04]. However, in data management, the curse of dimensionality is more generally referred to all the accuracy and efficiency issues due to high dimensional data, i.e., data

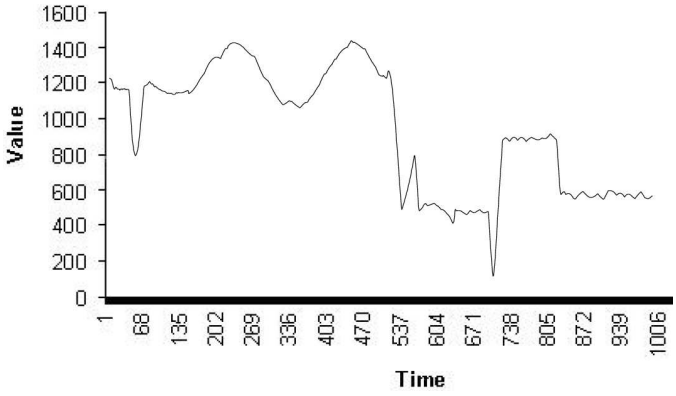


Fig. 1.2. An example of time series data

whose representation is given by a large number of variables (i.e, features). Major issues arising from the curse of dimensionality concern the sparsity of data represented in high dimensional spaces and the meaningfulness of the distance measure applied to this kind of data. Indeed, it has been shown that, for certain data distributions, the relative distances of the closest and farthest data points of an independently selected point goes to zero as the dimensionality increases [BGRS99]; a similar result involving the absolute distances and the L_p norm ($p \geq 3$) has been proved in [HAK00]. Such results make the curse of dimensionality phenomenon particularly critical especially in a task of clustering, since it is well-known that clustering puts its basis on the notion of similarity/distance between data objects.

Nowadays, high dimensional data arise from a lot of various application domains. Relevant examples are biomedical data, text data, data managed by e-commerce applications, web data, xml and semistructured data, *time series* data. In particular, time series, i.e., sequences of (real) numeric values upon which a total order based on timestamps is defined, are generally used to represent the temporal evolution of objects (cf. Fig. 1.2[1]). To this purpose, managing time series data by means of data mining techniques, and, in particular, clustering, is challenging due to the enormous amounts of such data naturally available from several sources of different domains, e.g., speech recognition, financial and market data analysis, telecommunication and telemetry, sensor networking, motion tracking, meteorology [WY05, Lia05].

Global Dimensionality Reduction. A well-known class of methods and algorithms which aim to alleviate the issues arising from high dimensional data is the so-called *dimensionality* or *dimension reduction* [Fod02]. The main goal of dimensionality reduction is to compute, for each data object having m dimensions (i.e., represented by a number of features equal to m), a different

[1] Figure 1.2 is borrowed from [KS97].

representation given by m' features, where $m' \ll m$. The new m'-dimensional representation is typically carried out globally to the entire set of objects, i.e., the same algorithm or scheme is exploited for all the objects.

Dimensionality reduction techniques can be classified into two main categories, namely *domain-driven* and *general*. Domain-driven techniques are specific and specialized only for the application context that are designed to; hence, they are not applicable or do not guarantee high performances when involved in more general scenarios. As an example, there has been proposed a lot of dimensionality reduction techniques for time series data [DTS+08]. Although such techniques fulfill some crucial requirements arising from the time series application context, such as, e.g., *time-warping awareness*, they cannot be recognized as generally valid for different application domains. General dimensionality reduction aims to define methods that are suitable for dealing with any kind of data. General techniques comprise *feature selection* [MBN02] and *feature extraction* [GGNZ06]. According to feature selection, the dimensionality is reduced by selecting a subset of the original features of the data objects, while discarding the remaining ones. The crucial problem of choosing the features to be maintained can be solved by taking into account different criteria. For instance, according to *wrapper models* [KJ97, KSM00], the relevant features are selected by evaluating the results obtained by running one or more clustering algorithms on the reduced space. *Entropy-based* criteria [DLY97, DCSL02] involve the evaluation of the entropy of the distribution of the feature values in each dimension. On the other hand, feature extraction methods construct a new, smaller set of features by exploiting the information available from the original one. Well-established feature extraction techniques include *Principal Component Analysis* (PCA) [DH73, Fuk90, Jol02], which computes the new features as linear combinations of the original ones, and is a special case of a more general matrix decomposition method called *Singular Value Decomposition* (SVD) [Str88, Dep89, Sch91].

Local Dimensionality Reduction. In the context of data clustering, it may happen that any cluster to be discovered "exists" in a specific subspace, i.e., its objects are close to each other if (and only if) they are projected onto that subspace. Thus, performing dimensionality reduction globally to the entire set of objects may lead to meaningless clustering results. Indeed, global dimensionality reduction might eliminate or transform one o more dimensions that are potentially relevant to at least one cluster; also, since the subspaces associated to each cluster may be in general different to each other, the global reduced representation can be clearly not consistent with the subspaces of the various clusters. An illustrative example of this scenario is depicted in Fig. 1.3;[2] the three clusters C_1, C_2, and C_3 exist in the subspaces given by the sets of features $\{x_1\}$, $\{x_2\}$, and $\{x_3\}$, respectively; therefore, the subspaces associated to each cluster are all different to each other and each dimension is relevant to at least one cluster.

[2] Figure 1.3 is borrowed from [PJAM02].

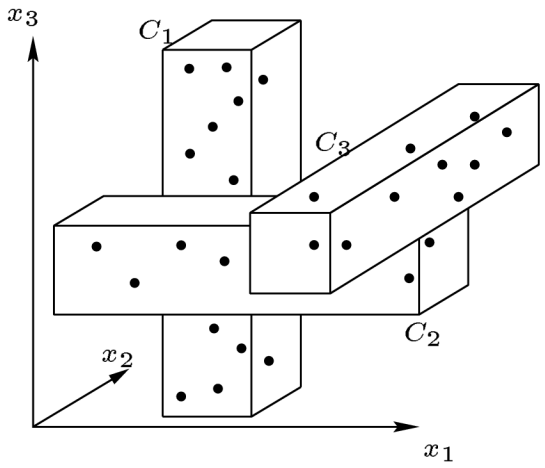


Fig. 1.3. Three clusters existing in different subspaces

Within this view, for a clustering task, it is advisable to perform a dimensionality reduction that is *local* to the single cluster. This is the main goal of *subspace clustering* and *projective* (or *projected*) *clustering* [PHL04, KKZ09], which aim to discover the cluster structure along with the subspaces corresponding to each cluster. As a consequence, subspace and projective clustering tend to be less noisy—because each group of data is represented over a subspace which does not contain irrelevant or redundant dimensions—and more understandable—because the exploration of a cluster is much easier when only few dimensions are involved. Projective clustering and subspace clustering problems are strictly related to each other.[3] A major difference between the two problems is that projective clustering outputs a single partition of the input set of data objects, whereas subspace clustering aims to find a set of clustering solutions, each one having clusters defined in a specific subspace. Indeed, subspace clustering algorithms typically aim to find clustering structures in every possible "interesting" subspace.

1.4 Contributions

The focus of this thesis is on the crucial problems of uncertainty and the curse of dimensionality arising from data clustering. The main contributions of this thesis are summarized in the following.

Uncertainty. The problem of clustering data affected by uncertainty at a representation level is deeply investigated. In particular, main contributions in this respect include:

[3] The terms "projective clustering" and "subspace clustering" are not used in a unified way in the literature.

- *UK-medoids*: a new partitional algorithm for clustering uncertain data, which is designed to overcome efficiency and accuracy issues of some existing partitional uncertain data clustering algorithms;
- *U-AHC*: the first (agglomerative) hierarchical algorithm for clustering uncertain data;
- as a special case of uncertain data, the problem of clustering *microarray* biomedical data with *probe-level* uncertainty is addressed and solved by exploiting the U-AHC algorithm.

Clustering uncertainty is addressed from a clustering ensembles perspective, i.e., by focusing on the problem of *weighted consensus clustering*, which aims to automatically determine weighting schemes to discriminate among clustering solutions in a given ensemble. In particular, contributions of this thesis to clustering ensembles include:

- *Single-Weighting* (SW), *Group-Weighting* (GW), and *Dendrogram-Weighting* (DW): three novel *diversity*-based, general schemes for weighting the individual clusterings in a given ensemble;
- *WICE*, *WCCE*, and *WHCE*: three algorithms to easily involve clustering weighting schemes into any clustering ensembles algorithm falling into one of the main classes of clustering ensembles approaches, i.e., *instance-based*, *cluster-based*, and *hybrid*.

The curse of dimensionality. Global dimensionality reduction is addressed by focusing on a domain-driven scenario; in particular, the application context taken into account is that of time series data. Contributions in this regard are the following:

- *Derivative time series Segment Approximation* (DSA) model: a new time series dimensionality reduction method (i.e, representation model) designed for accurate and fast similarity detection and clustering of time series data;
- *Mass Spectrometry Data Analysis* (MaSDA): a system for analyzing *mass spectrometry* biomedical data, whose core exploits DSA to model mass spectrometry data according to a time series-based representation;
- exploiting DSA for low-voltage electricity customer profiling.

Regarding local dimensionality reduction, the projective clustering and clustering ensembles problems are viewed for the first time as two faces of the same coin. In particular:

- the novel *Projective Clustering Ensembles* (PCE) problem is investigated and formally defined by providing two specific optimization formulations, i.e., *two-objective PCE* and *single-objective PCE*;
- *MOEA-PCE* and *EM-PCE* algorithms are proposed as novel heuristic solutions to the two-objective PCE and single-objective PCE problems, respectively.

1.5 Outline of the Thesis

This thesis is organized in five parts:

- Part I, *Preliminaries*, provides an introduction and the background to clustering data mining task;
- Part II, *Uncertainty in Data Clustering – data level*, deals with the problem of uncertainty at a data representation level arising from data clustering;
- Part III, *Uncertainty in Data Clustering – clustering level*, addresses the so-called clustering uncertainty, by focusing, in particular, on the problem of clustering ensembles;
- Part IV, *The Curse of Dimensionality in Data Clustering – global dimensionality reduction*, focuses on global dimensionality reduction as a solution for overcoming the curse of dimensionality problem in data clustering;
- Part V, *The Curse of Dimensionality in Data Clustering – local dimensionality reduction*, deals with the curse of dimensionality in data clustering from a local dimensionality reduction perspective; in particular, the focus is on the problem of projective clustering.

The five parts are articulated around the following chapters.

Chapter 1 informally describes the context of this thesis. It firstly introduces the KDD process, paying particular attention on data mining, as a major step of KDD, and clustering, i.e., the specific data mining task addressed in this thesis. Then, it provides an overview of the problems of uncertainty and the curse of dimensionality in data clustering, along with a brief summary of the main existing techniques for overcoming these problems. Finally, the main contributions and the outline of the thesis conclude the chapter.

Chapter 2 provides background notions to the clustering task in data mining. The focus of the chapter is on the algorithms/criteria needed for a clear explanation of the proposals discussed in the remainder of the thesis. It firstly overviews the main existing classes of clustering approaches, namely, *partitional relocation*, *density-based*, and *hierarchical*. Afterward, the problem of *soft* clustering is introduced, along with the cluster validity criteria aimed at assessing the quality of a clustering solution.

Chapter 3 deals with the problem of uncertainty in data representation, which is analyzed from a clustering perspective. It provides a background to the problem at hand. First, the main models used for representing data uncertainty are discussed, particularly focusing on the model adopted by the so-called *uncertain objects*, which are the specific kind of uncertain data taken into account in this thesis. Furthermore, the prominent state-of-the-art methods for clustering uncertain objects are briefly reviewed.

Chapter 4 presents a new algorithm for clustering uncertain objects, called *UK-Medoids*, which is mainly conceived to overcome accuracy and efficiency issues of some of the current partitional algorithms for clustering uncertain objects. It first defines the distance functions exploited by the proposed algorithm, and, then, describes the K-Medoids-based scheme at the basis of

UK-Medoids. The chapter ends by discussing the experiments carried out to show accuracy and efficiency performances of UK-medoids with respect to the other prominent state-of-the-art methods.

Chapter 5 introduces U-AHC, i.e., the first agglomerative hierarchical algorithm for clustering uncertain objects. Before describing the prototype link-based agglomerative scheme of U-AHC, it provides the definitions of the prototypes used for summarizing the uncertain objects in a cluster and the new information-theoretic criterion for comparing such prototypes. Moreover, the chapter presents the experimental evaluation aimed at validating U-AHC in terms of accuracy and efficiency, and with respect to the other existing algorithms for clustering uncertain objects. The chapter concludes by presenting an application of the U-AHC algorithm to clustering microarray biomedical data with probe-level uncertainty.

Chapter 6 provides background to the clustering ensembles problem as a valid and powerful solution for overcoming clustering uncertainty. It concerns the main notions at the basis of clustering ensembles, along with a brief overview of the major existing methods proposed in the literature for solving such a problem.

Chapter 7 addresses the weighted consensus clustering problem by proposing three general clustering weighting schemes, called Single-Weighting (SW), Group-Weighting (GW), and Dendrogram-Weighting (DW). The main goal of the proposed schemes is to assign a proper weight to each clustering solutions in a given ensemble, in order to discriminate among such solutions when performing a clustering ensembles task. After explaining the details of the proposed SW, GW, and DW schemes, three further algorithms, i.e., WICE, WCCE, and WHCE, are presented. Such algorithms are designed to easily involve clustering weighting schemes into any instance-based, cluster-based, and hybrid clustering ensembles algorithm, respectively. Finally, an extensive experimental evaluation is presented. The main goal of these experiments is to assess the impact of employing the proposed weighting schemes in clustering ensembles, by comparing the performances of a large number of clustering ensemble algorithms with and without each of the proposed schemes.

Chapter 8 focuses on time series data management. It provides background definitions for the context at hand and a brief description of the state-of-the-art on time series similarity detection and time series dimensionality reduction.

Chapter 9 presents a new time series representation model, called DSA, which performs domain-driven dimensionality reduction in the context of time series data in an effective and efficient way. Firstly, the novelty at the basis of DSA, along with its main differences with respect to the competing methods, are discussed. Secondly, the three main steps of the DSA model, i.e., *derivative estimation*, *segmentation*, and *segment approximation*, are described in detail. Finally, DSA accuracy and efficiency performances are extensively evaluated and compared to those of the other existing time series dimensionality reduction techniques.

Chapter 10 shows as the DSA model can be exploited for solving two problems from real-world applications: *(i)* clustering of mass spectrometry biomedical data, and *(ii)* low-voltage electricity customers profiling. The first problem is addressed by describing the Mass Spectrometry Analyzing System (MaSDA) and presenting several experiments aimed at assessing the validity of the innovative approach proposed. As regards the second problem, the specific approach to electricity customers profiling is presented, along with the relative experimental evaluation.

Chapter 11 deals with local dimensionality reduction in data clustering. It summarizes the main notions at the basis of subspace clustering and projective clustering problems, putting particular emphasis on the latter. In this respect, the problem of projective clustering is formalized and existing research on projective clustering is briefly reviewed.

Chapter 12 addresses for the first time the Projective Clustering Ensembles (PCE) problem, whose objective is to define methods for clustering ensembles that are able to deal with ensembles of projective clustering solutions. Firstly, PCE is formally defined according to two different optimization formulations, namely a two-objective and a single-objective formulation. For each of the proposed formulations, proper heuristic algorithms, i.e, MOEA-PCE and EM-PCE, respectively, are described. MOEA-PCE and EM-PCE are eventually evaluated in terms of accuracy by performing a large set of experiments on several publicly available benchmark datasets.

Finally, *Conclusion* chapter ends the thesis by reviewing the main contributions to uncertainty and the curse of dimensionality in data clustering, and considering open problems and directions of future research.

2

Clustering

Abstract This chapter provides an insight into the problem of clustering, which represents the focus of this thesis. In particular, it reports on the major details about clustering needed for the purposes of this thesis. The basis of partitional and hierarchical clustering approaches are provided. The discussion about partitional clustering is twofold: it concerns both partitional algorithms exploiting a relocation scheme and density-based algorithms. Hierarchical clustering is treated by focusing on the standard agglomerative scheme (AHC) and the classic linkage metrics typically used in any AHC approach (i.e., *single link*, *complete link*, *average link*, and *prototype link*). Finally, the chapter describes the problem of soft clustering and the criteria used in the remainder of the thesis for assessing the quality of any clustering solution.

2.1 Clustering Solution

Definition 2.1 (clustering solution). *Let $D = \{o_1, \ldots, o_n\}$ be a set of data objects and $f : D \times D \rightarrow \Re$ be a distance function between the objects in D. A* clustering solution *or simply a* clustering *$\mathcal{C} = \{C_1, \ldots, C_K\}$ defined over D is a partition of D into K groups, i.e.,* clusters*, computed by properly exploiting the information available from f.*

Traditionally, the objects in the input set D are represented in terms of deterministic vectors of *numerical* or *categorical* feature values. In that case, each object o_i is coupled with the corresponding vector $\boldsymbol{\omega}_i = [\omega_{i1}, \ldots, \omega_{im}]$, $\forall i \in [1..n]$. If not differently specified, this thesis hereinafter refers to this kind of representation for data objects. Also, in the remainder of this thesis, any distance function f between data objects in D is assumed to satisfy the following conditions:

1. $f(o_i, o_{i'}) \geq 0, \quad \forall i, i' \in [1..n]$
2. $f(o_i, o_{i'}) = 0$ if and only if $o_i = o_{i'}$
3. $f(o_i, o_{i'}) = f(o_{i'}, o_i), \forall i, i' \in [1..n]$

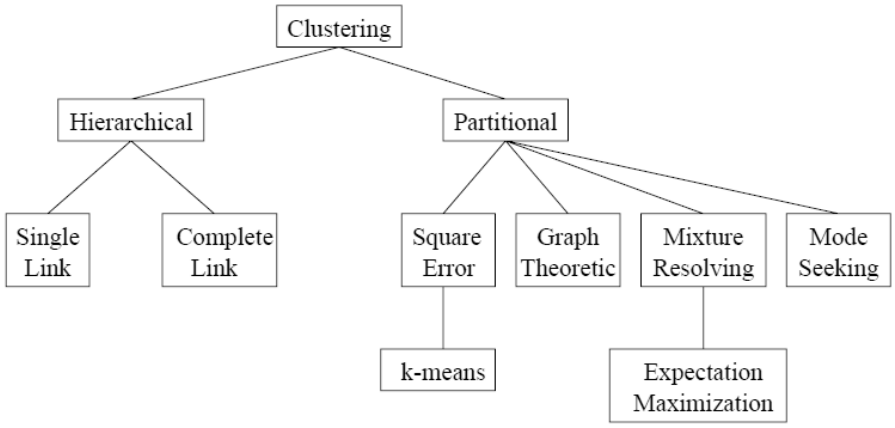


Fig. 2.1. A taxonomy of clustering approaches

Note that if f also satisfies the *triangle inequality* condition, i.e., $f(o_i, o_{i'}) \leq f(o_i, o_{i''}) + f(o_{i''}, o_{i'})$, $\forall i, i', i'' \in [1..n]$, then f is a *metric* [LS74].

As mentioned in Chap. 1, any clustering is built in such a way that cluster cohesiveness and separation, measured according to the input function f, are maximized. Clearly, such a criterion is too general; therefore, clustering methods typically provide a specific objective function to minimize/maximize, in order to formally define clusters that are compact and well-separated from each other. Since these formulations usually leads to NP-hard problems, any specific clustering method should define the corresponding heuristic algorithm to find good approximations of the optimal solution.

In the literature, there has been defined a huge number of clustering methods and algorithms, which differ to each other for the optimization criterion, the resolution strategy, and the computation of the distance between the input objects. These algorithms can be classified according to a lot of different taxonomies, such as, e.g., that reported in Fig. 2.1 [JD88]. Typically, according to the top level of such taxonomies clustering approaches are classified into two main categories, i.e., *partitional* (or *partitioning*) and *hierarchical*.

2.2 Partitional Clustering

Partitional clustering algorithms compute a single partition of the input dataset. A significant subset of partitional algorithms exploits the *relocation* scheme [Ber02], i.e., the objects are iteratively re-assigned to the clusters, until a stop criterion is met. Another approach is that exploited by the *density-based* methods [HK01], which try to discover dense connected components of data objects.

Algorithm 2.1 K-Means

Input: a set $D = \{o_1, \ldots, o_n\}$ of data objects;
the number K of clusters in the output clustering solution;
Output: a clustering solution $\mathcal{C}^* = \{C_1^*, \ldots, C_K^*\}$ defined over D

1: $\mathcal{V}^* \leftarrow randomSelect(D, K)$
2: **repeat**
3: compute $\mathcal{C}^*$ according to (2.4) {*object assigning*}
4: compute $\mathcal{V}^*$ according to (2.5) {*centroid updating*}
5: **until** *convergence*

2.2.1 Partitional Relocation Methods

Relocation scheme is at the basis of the well-known *K-Means* and *K-Medoids* algorithms.

K-Means

The basic K-Means algorithm [Mac67] works on data objects represented by deterministic vectors of numerical features.[1] It is based on the minimization of the *Sum of Squared Error* (SSE) [LC03] between each object in a cluster and the corresponding cluster *representative* or *prototype*, which, in case of K-Means, is called *centroid*. Formally, given an input dataset $D = \{o_1, \ldots, o_n\}$, a partition (clustering) $\mathcal{C} = \{C_1, \ldots, C_K\}$ of D, and a set $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$ of centroids, such that $\mathbf{v}_k$ is the centroid of cluster C_k, $\forall k \in [1..K]$, the objective function to be minimized by K-Means is the following:

$$J(D, \mathcal{C}, \mathcal{V}) = \sum_{k=1}^{K} \sum_{i=1}^{n} I[o_i \in C_k] \sum_{j=1}^{m} (\omega_{ij} - v_{ij})^2 \tag{2.1}$$

where $I[A]$ is the *indicator function*, which is equal to 1 when the event A occurs, 0 otherwise.

The outline of K-Means algorithm is reported in Alg. 2.1. Essentially, K-Means is based on a relocation scheme composed by two alternating steps. In the *object assigning* step, the current clustering $\mathcal{C}^* = \{C_1^*, \ldots, C_K^*\}$ is computed, i.e., the objects are assigned to the proper cluster according to the minimum distance from the centroids. In the *centroid updating* step, the set $\mathcal{V}^* = \{\mathbf{v}_1^*, \ldots, \mathbf{v}_K^*\}$ of cluster centroids is re-computed according to the assignments performed in the previous step. $\mathcal{C}^*$ and $\mathcal{V}^*$ are computed by solving the following system of equations, which follow directly from the optimization function reported in (2.1):

[1] Several variants of the basic K-Means have been proposed in the literature [And73].

$$\frac{\partial\ J}{\partial\ C_k} = 0 \tag{2.2}$$

$$\frac{\partial\ J}{\partial\ v_{kj}} = 0 \tag{2.3}$$

It can be proved that the solutions of the previous equations are, respectively:

$$C_k^* = \left\{ o_i \in D \mid \mathbf{v}_k = \arg\min_{\mathbf{v}_{k'} \in \mathcal{V}} \sum_{j=1}^{m} (\omega_{ij} - v_{k'j})^2 \right\} \tag{2.4}$$

$$v_{kj}^* = \frac{1}{|C_k^*|} \sum_{i=1}^{n} I[o_i \in C_k^*]\ \omega_{ij} \tag{2.5}$$

In the initialization step of K-Means, the set of initial centroids is computed by randomly choosing K objects from the input dataset. The convergence of the algorithm can be easily proved since, according to the way how (2.4) and (2.5) have been derived, a gradient descent on the objective function J is performed; hence, it holds that the alternating procedure at the basis of K-Means converges to a local optimum of function J. The convergence criterion can be precisely specified according to one of the following [MIH08]: the algorithm may run until *(i)* the set of centroids does not change, *(ii)* the assignments of the objects to the clusters do not change, or *(iii)* the current value of J is equal to that computed in the previous iteration.

Finally, the computational complexity of K-Means is $\mathcal{O}(I\ K\ n\ m)$, where I is the number of iterations needed for the convergence.

K-Medoids

K-Medoids algorithm shares with K-Means two main features. Firstly, the alternating procedure at the basis of the relocation scheme is the same for both the algorithms. Also, in both K-Medoids and K-Means, the object assigning criterion involves the distance between objects and cluster prototypes. However, unlike K-Means, cluster prototypes in K-Medoids are given by the so-called *medoids*, i.e., specific objects in the cluster that satisfy proper criteria.

Defining medoids as cluster prototypes has two main advantages with respect to K-Means centroids [Ber02]. Firstly, while K-Means requires the squared Euclidean norm to compute the distance between objects and centroids, K-Medoids can in principle work with any distance function f provided in input; this leads to the possibility of easily extending K-Medoids to be used for data objects represented in a way more general than the numerical vectorial form required by K-Means. Moreover, the choice of medoids is dictated by the location of a predominant fraction of points inside a cluster and, therefore, it is less sensitive to the presence of outliers. On the other hand, centroids

have the advantages of clear geometric and statistical meaning, and easier computation, in terms of both procedural and computational complexity.

One of the most general way to define medoids consists in taking into account the sum of the distances between any object in a cluster and all the other objects in the same cluster. According to such a criterion, a basic version of K-Medoids can be easily defined by resorting to a scheme similar to that employed by K-Means (Alg. 2.1), in which different ways for *(i)* computing the distance between objects and prototypes, and *(ii)* defining prototypes are exploited. In particular, the distance between objects and prototypes is measured according to the input function f, and, in the definition of cluster prototypes v_k^*, $\forall k \in [1..K]$, (2.5) is replaced with the following:

$$v_k^* = \arg \min_{o \in C_k^*} \sum_{\substack{o' \in C_k^*, \\ o' \neq o}} f(o, o') \tag{2.6}$$

Since K is obviously $\mathcal{O}(n)$, the basic K-Medoids scheme leads to a computational complexity of $\mathcal{O}(I\ F\ n^2)$, where F is the cost of computing the distance between any pair of objects according to function f; it should be noted that F is typically $\Omega(m)$. Therefore, K-Medoids is computationally more expensive than K-Means.

More refined versions of K-Medoids comprise, e.g., *Partitioning Around Medoids* (PAM) [KR87], *Clustering LARge Applications* (CLARA) [KR90], *Clustering Large Applications based upon RANdomized Search* (CLARANS) [NH94] and its extension working on spatial very large databases [EKX95].

2.2.2 Density-based Methods

The main idea underlying density-based clustering algorithms is that an open set in an Euclidean space can be split into a number of connected components. Such a concept is exploited for recognizing clusters as *dense* subsets of the input dataset, where the notion of *density* requires a metric space to satisfy soundness properties. In particular, any cluster is built incrementally by starting from an initial point (representing any object in the input dataset) and including, at each step, a set of neighbor objects; in this way, cluster grows towards the direction of the density region.

Density-based algorithms have several advantageous features with respect to relocation algorithms. Indeed, they perform well in detecting clusters having irregular shapes, are robust to outliers, may effectively handle noisy data, and have high scalability. On the other hand, density-based approaches may suffer from some issues. First, they may fail in discovering the actual cluster structure when the clusters have densities not equally-distributed. Moreover, the discovered clusters may be not easily understandable, since they are composed by connected objects which carry a great variety of feature values within the cluster; this aspect may affect pattern identification and cluster characterization.

Most popular density-based clustering algorithms comprise *DBSCAN* and *OPTICS*.

DBSCAN

DBSCAN (*Density Based Spatial Clustering of Application with Noise*) [EKSX96] is one of the earliest density-based clustering algorithms. The outline of DBSCAN is shown in Alg. 2.2.

Algorithm 2.2 DBSCAN

Input: a set D of n data objects;
a real number ϵ representing the radius size;
an integer μ representing the minimum number of objects in the neighborhood of any core point
Output: a clustering solution $\mathcal{C}^*$ defined over D

```
1: C* ← ∅
2: for all o ∈ D such that o has not been visited do
3:    mark o as visited
4:    N_o ← getNeighbors(o,ε)
5:    if |N_o| ≥ μ then
6:       C ← {o}
7:       expandCluster(C, N_o, ε, μ)
8:       C* ← C* ∪ {C}
9:    else
10:      mark o as outlier
11:   end if
12: end for
```

DBSCAN requires two input parameters, i.e., a real value ϵ which represents the radius of the hypersphere within which the neighbors of any object are searched for, and μ, which is an integer representing the minimum number of points within the hypersphere of radius ϵ needed for recognizing any object as a *core point*. Basically, any object $o \in D$ is recognized as a core point if and only if the number of objects having distance from o lower than ϵ is greater than or equal to μ.

The algorithm is essentially based on two main steps. In the first one, it searches for core points among the objects that have not already been visited (i.e., among the objects in D that do not yet belong to any cluster and have not been previously marked as *outliers*). Once a new core point o has been discovered, a new cluster C is built around o by means of the procedure *expandCluster*. Such a procedure aims to iteratively look for core points among the neighbor list of o. The procedure ends when the cluster C cannot be further expanded, i.e, there are no more core points to be recognized. Using proper data structures, the overall time complexity of DBSCAN is $\mathcal{O}(n \log n)$.

OPTICS

Effectiveness and efficiency of DBSCAN are both highly related to the two input parameters ϵ and μ which are typically hard to set. Moreover, there might exist different densities among clusters; thus, in order to properly discover clusters of different densities, ϵ and μ should not be global thresholds, rather their values should vary depending on the specific cluster to be discovered.

In order to overcome these issues, the OPTICS (*Ordering Points To Identify the Clustering Structure*) [ABKS99] algorithm aims to build an augmented ordering of the objects in the input dataset by varying ϵ parameter in order to cover a spectrum of all different $\epsilon' \leq \epsilon$. To build the ordering, OPTICS requires to store additional information for each object, i.e., the *core-distance* and the *reachability-distance*. The core-distance of any object o is the minimum distance ϵ' such that the neighbor list of $o \in D$ computed according to ϵ' has size exactly equal to μ. The reachability distance of any object o with respect to any other object $o' \in D$, $o \neq o'$, is the minimum distance ϵ' such that o is "ϵ'-reachable" from o', i.e., o' is a core point according to ϵ', and the cluster C, which is built by means of procedure *expandCluster* invoked over o' and ϵ', contains o.

The augmented ordering is finally exploited to efficiently extract all the cluster structures based on any density threshold $\epsilon' \leq \epsilon$. Thus, by setting ϵ equal to a number sufficiently large, OPTICS has a very low sensitivity to parameter ϵ. A similar conclusion can be drawn for parameter μ.

OPTICS outputs a *reachability plot*, which stores information for extracting a cluster hierarchy based on density, i.e., a set of clustering solutions whose clusters have been computed by taking into account different density thresholds. Due to this peculiarity, OPTICS is sometimes referred to as an "hybrid" algorithm, in the sense that it shares features with both partitional density-based and hierarchical algorithms.

2.3 Hierarchical Clustering

Instead of a single partition of the input dataset, hierarchical clustering approaches output a hierarchy of clustering solutions that are organized into a so-called *dendrogram* [JD88], i.e., is a tree aimed at graphically describing such a hierarchy (cf. Fig. 2.2[2]).

Definition 2.2 (dendrogram). *A* dendrogram *defined over a set D of data objects is a set $\mathbf{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$ of cluster pairs, where $\mathcal{T}_t = \langle \mathcal{T}_t', \mathcal{T}_t'' \rangle$, $\mathcal{T}_t' \subseteq D$, $\mathcal{T}_t'' \subseteq D$, $\forall t \in [1..T]$, and:*

1. $\mathcal{T}_t'' \subset \mathcal{T}_t', \quad \forall t \in [1..T]$

[2] Figure 2.2 is borrowed from [Ols95].

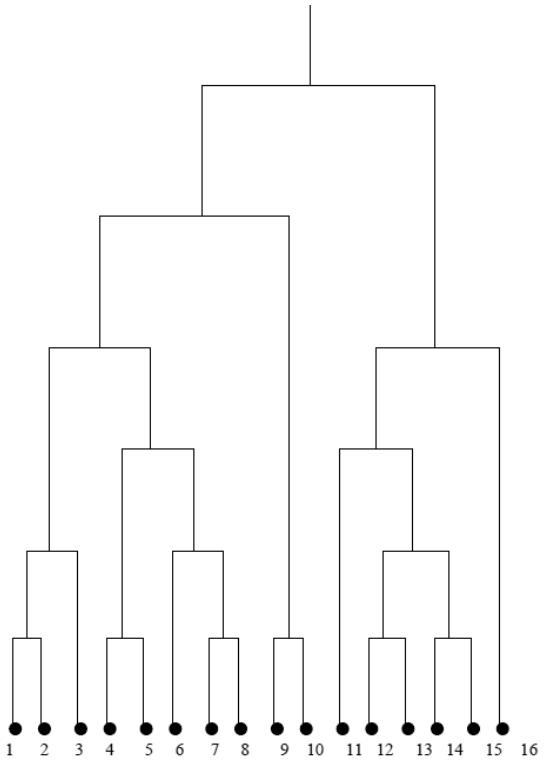


Fig. 2.2. A dendrogram showing a possible cluster hierarchy built upon a simple dataset of 16 data objects

2. $\mathcal{T}_t'' \cap \mathcal{T}_{t'}'' = \emptyset$ *and* $\mathcal{T}_t' \supseteq \mathcal{T}_t'' \cup \mathcal{T}_{t'}''$, $\quad \forall t, t' \in [1..T]$ *such that* $t \neq t'$ *and* $\mathcal{T}_t' = \mathcal{T}_{t'}'$

Any dendrogram defined according to Def. 2.2 can be alternatively defined by taking into account the levels of the tree. The top level $\mathcal{L}_Q$ (i.e., the root of the tree) contains a single cluster composed by all the objects in the input dataset, whereas in the bottom level $\mathcal{L}_0$ (i.e., the level containing the leaves of the tree) there are singleton clusters, i.e., clusters composed by exactly one input object. Intermediate levels contain partitions of the input dataset at a different granularity. In particular, each level $\mathcal{L}_q$, $\forall q \in [1..Q-1]$ comprises a number of clusters lower than that of level $\mathcal{L}_{q-1}$ and greater than that of level $\mathcal{L}_{q+1}$; this property clearly leads to clusters having average size directly proportional to the corresponding level number q. A *level-organized* dendrogram can be formally defined as follows.

Definition 2.3 (level-organized dendrogram). *Let* $\mathbf{T}$ *be a dendrogram defined over a set* D *of data objects. A* level-organized dendrogram *derived*

Algorithm 2.3 standard AHC

Input: a set $D = \{o_1, \ldots, o_n\}$ of data objects;
Output: a level-organized dendrogram $\mathbf{T}_\ell = [\mathcal{L}_0, \ldots, \mathcal{L}_Q]$ defined over D

1: $\mathcal{C} \leftarrow \{\{o_1\}, \ldots, \{o_n\}\}$
2: $\mathcal{L}_0 \leftarrow \mathcal{C}$
3: **repeat**
4: $\quad \langle C', C'' \rangle \leftarrow closestClusters(\mathcal{C})$
5: $\quad \mathcal{C} \leftarrow \mathcal{C} \setminus \{C', C''\} \cup \{C' \cup C''\}$
6: $\quad append(\mathbf{T}_\ell, \mathcal{C})$
7: **until** $|\mathcal{C}| = 1$

from $\mathbf{T}$ *is a list* $\mathbf{T}_\ell = [\mathcal{L}_0, \ldots, \mathcal{L}_Q]$*, where each* $\mathcal{L}_q$*,* $q \in [0..Q]$*, is a partition of* D*, and:*

1. $|\mathcal{L}_0| = |D|$
2. $|\mathcal{L}_Q| = 1$
3. $|\mathcal{L}_q| > |\mathcal{L}_{q+1}|, \quad \forall q \in [0..Q-1]$

Hierarchical clustering algorithms can be classified into two main categories, namely *Agglomerative Hierarchical Clustering* (AHC) and *Divisive Hierarchical Clustering* (DHC) [JD88]. The difference between these two kinds of approaches lies in the way how the dendrogram is computed. Agglomerative algorithms start from the bottom level of the tree, and build the dendrogram in a bottom-up way. According to divisive approaches the dendrogram is built in a top-down way, by starting from the root of the tree.

The standard AHC scheme follows a greedy approach (cf. Alg. 2.3). Given an input dataset D of size n, the starting point is a partition of D composed by n singleton clusters. Such a partition composes the bottom level of the dendrogram. At each iteration, the current level of the dendrogram is computed by merging the "closest" pair of clusters in the previous level. This strategy clearly leads to a dendrogram whose levels contain a number of clusters equal to that of the previous level minus one. Hence, the algorithm terminates after $n - 1$ iterations, i.e., when the whole tree has been built.

Linkage Metrics

A crucial point in the standard AHC scheme is the choice of a proper *linkage metric*, i.e., a criterion to decide which is the closest pair of clusters to be merged at each iteration [Mur83, Mur85, DE84]. Classic linkage metrics comprise *single link* (SL) [Sib73], *complete link* (CL) [Def77], *average link* (AL) [Voo86], and *prototype link*.

According to SL, CL, and AL metrics, also known as *graph* linkage metrics [Ols95], the pair of clusters to be merged is chosen by looking at the minimum, maximum, and average among the pairwise distances between the

objects in the two clusters, respectively. Formally, given a clustering $\mathcal{C}$ defined over a set D of data objects, and a distance function $f : D \times D \rightarrow \Re$ between data objects, the pair of closest clusters $\langle C', C'' \rangle$, $C', C'' \in \mathcal{C}$, selected by SL, CL, or AL is defined according to the following formulas, respectively:

$$\langle C', C'' \rangle = \arg\min_{\substack{\langle \hat{C}', \hat{C}'' \rangle \in \mathcal{C} \times \mathcal{C}, \\ \hat{C}' \neq \hat{C}''}} \min_{\substack{o' \in \hat{C}', \\ o'' \in \hat{C}''}} f(o', o'') \tag{2.7}$$

$$\langle C', C'' \rangle = \arg\min_{\substack{\langle \hat{C}', \hat{C}'' \rangle \in \mathcal{C} \times \mathcal{C}, \\ \hat{C}' \neq \hat{C}''}} \max_{\substack{o' \in \hat{C}', \\ o'' \in \hat{C}''}} f(o', o'') \tag{2.8}$$

$$\langle C', C'' \rangle = \arg\min_{\substack{\langle \hat{C}', \hat{C}'' \rangle \in \mathcal{C} \times \mathcal{C}, \\ \hat{C}' \neq \hat{C}''}} \frac{1}{|\hat{C}'||\hat{C}''|} \sum_{\substack{o' \in \hat{C}', \\ o'' \in \hat{C}''}} f(o', o'') \tag{2.9}$$

Let us now discuss the computational complexity of any clustering algorithm involving the standard AHC scheme along with one among SL, CL, or AL metric. The function f is assumed to be able to compute the distance between any pair of data objects in $\mathcal{O}(F)$, where F is typically $\Omega(m)$. Clearly, naïve implementations lead to algorithms working in $\mathcal{O}(F\ n^2 + n^3)$, for all SL, CL, and AL. This happens if the closest clusters to be merged are recognized by scanning, at each iteration, all the pairwise distances between the clusters in the current partition. If ad-hoc data structures are used to store these pairwise distances (e.g., priority queues whose insert/extract/delete operations are computed in logarithmic time), the complexity becomes $\mathcal{O}(n^2\ (F + \log n))$ [MRS08]. However, by using more complex data structures, it is possible to define algorithms working in $\mathcal{O}(F\ n^2)$ for each of the above metrics, i.e., SL [Sib73], CL [Def77], and AL [GM07].

Unlike SL, CL, and AL, prototype link metric does not involve any distance f among data objects. Prototype link is instead based on the comparison between cluster prototypes, which can be properly defined depending on the specific context. More precisely, given a clustering $\mathcal{C} = \{C_1, \ldots, C_K\}$, a set $\mathcal{P} = \{P_1, \ldots, P_K\}$ such that P_k is the prototype of cluster C_k, $\forall k \in [1..K]$, and a distance function $g : \mathcal{P} \times \mathcal{P} \rightarrow \Re$ between prototypes, the pair of closest clusters $\langle C', C'' \rangle$ selected according to the prototype link metric is defined as follows:

$$\langle C', C'' \rangle = \arg\min_{\substack{\langle \hat{C}', \hat{C}'' \rangle \in \mathcal{C} \times \mathcal{C}, \\ \hat{C}' \neq \hat{C}''}} g(P', P'') \tag{2.10}$$

As a particular case of prototype link metrics, the so-called *geometric* linkage metrics have been proposed in the literature [Ols95]. This kind of metrics deals with specific definitions of cluster prototypes, such as prototypes defined according to a *centroid*, *median*, or *minimum* variance criterion [Mur83]. As an example, according to a centroid link metric, the prototype of a cluster is defined as the arithmetic mean of the objects in the clusters, in a way similar to that employed by K-Means algorithm (cf. (2.5)). Centroid link clearly works only for data objects represented by vectors of numerical features.

Regarding the computational complexity of a hierarchical algorithm involving the standard AHC scheme and the prototype link metric, it generally depends on the time needed for computing *(i)* cluster prototypes, and *(ii)* the distance between prototypes according to the function g.

2.4 Soft Clustering

All the discussions carried out so far focused on the *hard* clustering problem, whose main goal is to produce *crisp* partitions of the input dataset. A partition is called "crisp" if any input data object belongs to exactly one cluster of the partition.

However, for many application contexts, dealing with crisp partitions may lead to loss of accuracy, since there might exist more than one "true" partition to be discovered for the data at hand. As a result of focusing only on one of these true partitions, like hard clustering algorithms do, critical information about hidden relationships among data objects may be potentially disregarded. Within this view, it is advisable to resort to *soft* (or *fuzzy*) clustering algorithms [Ped90, BP92, Yan93, BB99a, BB99b, VP07, MIH08], whose main goal is to output *fuzzy* partitions. A major feature of this kind of partitions, which resort to the notion of *fuzzy sets* [Zad65], is that they are composed by possibly *overlapping* clusters, i.e., each input data object may belong to more than one cluster. To this purpose, a fuzzy partition is necessarily coupled with a *membership* function [Pao89], which aims to numerically quantify the degree of belonging of input data objects to the various clusters. Thus, the set of clusters of the partition and the membership function are the two ingredients of the definition of a *soft clustering solution*, which contrasts that of (hard) clustering solution reported in Def. 2.1.

Definition 2.4 (soft clustering solution). *Let $D = \{o_1, \ldots, o_n\}$ be a set of data objects. A* soft clustering solution *or simply a* soft clustering *defined over D is a pair $\langle \mathcal{L}, \Gamma \rangle$, where $\mathcal{L} = \{\ell_1, \ldots, \ell_K\}$ is a set of labels which uniquely represent the K clusters, and $\Gamma : \mathcal{L} \times D \rightarrow \Re$ is a membership function which numerically quantifies the degree of belonging of object o_i to cluster labeled with ℓ_k, $\forall i \in [1..n], k \in [1..K]$, such that:*

1. $\Gamma_{ki} \in [0, 1], \quad \forall k \in [1..K], i \in [1..n]$
2. $0 < \sum_{i=1}^{n} \Gamma_{ki} < n, \quad \forall k \in [1..K]$
3. $\max_{k \in [1..K]} \Gamma_{ki} > 0, \quad \forall i \in [1..n]$

where Γ_{ki} hereinafter refers to $\Gamma(\ell_k, o_i)$.

The three requirements reported in Def. 2.4 make function Γ falling into the category of *possibilistic* or *absolute* membership functions [KK93, DK97]. A more restricted category comprises *probabilistic* or *relative* membership functions [KK93, DK97], for which the following additional condition holds:

$$\sum_{k=1}^{K} \Gamma_{ki} = 1, \quad \forall i \in [1..n]$$

Fuzzy *C*-Means

Fuzzy C-Means (FCM) [Dun74] is a modified version of the popular K-Means algorithm which handles fuzzy partitions of the input dataset and deals with probabilistic membership functions. Essentially, the key idea is to modify the objective function J (cf. (2.1)) of K-Means, in order to obtain soft clustering solutions. More precisely, the optimization problem at the basis of FCM is the following:

$$\langle \Gamma^*, \mathcal{V}^* \rangle = \arg\min_{\langle \Gamma, \mathcal{V} \rangle} J_{\mathcal{F}}(D, \mathcal{L}, \Gamma, \mathcal{V}) \tag{2.11}$$

$$s.t.$$

$$\sum_{k=1}^{K} \Gamma_{ki} = 1, \quad \forall i \in [1..n] \tag{2.12}$$

$$\Gamma_{ki} \geq 0, \quad \forall k \in [1..K], i \in [1..n] \tag{2.13}$$

where $D = \{o_1, \ldots, o_n\}$ is a set of m-dimensional data objects, $\mathcal{L} = \{\ell_1, \ldots, \ell_K\}$ is a set of cluster labels, $\Gamma : \mathcal{L} \times D \rightarrow \Re$ is real-valued function, $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$ is a set of centroids, such that $\mathbf{v}_k$ is the centroid of cluster labeled with ℓ_k, $\forall k \in [1..K]$, and

$$J_{\mathcal{F}}(D, \mathcal{L}, \Gamma, \mathcal{V}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \Gamma_{ki}^{\alpha} \sum_{j=1}^{m} (\omega_{ij} - v_{ij})^2 \tag{2.14}$$

Note that $\alpha > 1$ is an integer user-defined parameter that guarantees the nonlinearity of $J_{\mathcal{F}}$ with respect to Γ_{ki}, which is needed for ensuring that Γ_{ki} range within $[0, 1]$ (instead of $\{0, 1\}$), for any solution of the problem defined in (2.11)-(2.13).

FCM exploits a relocation partitional scheme similar to that of its corresponding hard version. The main equations at the basis of such a scheme are derived by resorting to the relaxed objective function $J'_{\mathcal{F}}$, which is computed according to the conventional Lagrange multipliers method [Bal70, TW81]:

$$J'_{\mathcal{F}}(D, \mathcal{L}, \Gamma, \mathcal{V}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \Gamma_{ki}^{\alpha} \sum_{j=1}^{m} (\omega_{ij} - v_{ij})^2 + \sum_{i=1}^{n} \lambda_i \left(\sum_{k=1}^{K} \Gamma_{ki} - 1 \right) \tag{2.15}$$

For a fixed assignment of $\mathcal{V}$, the optimal Γ^*_{ki} is computed by solving the following system of equations:

$$\frac{\partial J'_{\mathcal{F}}}{\partial \Gamma_{ki}} = 0 \tag{2.16}$$

$$\frac{\partial\ J'_{\mathcal{F}}}{\partial\ \lambda_i} = 0 \tag{2.17}$$

whose solution is:

$$\Gamma^*_{ki} = \left[\sum_{k'=1}^{K} \left(\frac{\sum_{j=1}^{m} (\omega_{ij} - v_{kj})^2}{\sum_{j=1}^{m} (\omega_{ij} - v_{k'j})^2} \right)^{\frac{1}{\alpha-1}} \right]^{-1} \tag{2.18}$$

Analogously, for a fixed assignment of Γ, the optimal v^*_{kj} is computed by solving the following equations:

$$\frac{\partial\ J'_{\mathcal{F}}}{\partial\ v_{kj}} = 0 \tag{2.19}$$

$$\frac{\partial\ J'_{\mathcal{F}}}{\partial\ \lambda_i} = 0 \tag{2.20}$$

which are solved by the following:

$$v^*_{kj} = \frac{\sum_{i=1}^{n} \Gamma^{\alpha}_{ki}\ \omega_{ij}}{\sum_{i=1}^{n} \Gamma^{\alpha}_{ki}} \tag{2.21}$$

The basic scheme of FCM is that exploited for K-Means algorithm (cf. Alg. 2.1), in which (2.4) and (2.5) are replaced with (2.18) and (2.21), respectively.

Finally, it can be easily noted that the considerations about convergence and computational complexity of K-Means also hold for FCM.

Fuzzy *C*-Medoids

Fuzzy C-Medoids (FCMdd) [KJY99, KJNY01] is a soft clustering algorithm that follows the scheme of K-Medoids described in Sect. 2.2. Like the corresponding hard version, FCMdd is not required to work only on objects represented by numerical features or use the squared Euclidean norm to compute the distances between objects and cluster prototypes. Instead, FCMdd may in principle work on any kind of object representations, by exploiting a function $f : D \times D \rightarrow \Re$ designed for properly quantifying the distance between the objects into the input dataset D.

Each iteration of the relocation scheme of FCMdd outputs a soft clustering solution $\langle \mathcal{L}^*, \Gamma^* \rangle$, where $\mathcal{L}^* = \{\ell^*_1, \ldots, \ell^*_K\}$ is a set of cluster labels, and $\mathcal{V}^* = \{v^*_1, \ldots, v^*_K\}$ is a set of medoids. In particular, v^*_k, $\forall k \in [1..K]$, is defined as:

$$v^*_k = \arg\min_{o \in D} \sum_{\substack{o' \in D, \\ o' \neq o}} \big(\Gamma(\ell^*_k, o')\big)^{\alpha} f(o, o') \tag{2.22}$$

whereas the values Γ^*_{ki}, $\forall k \in [1..K], i \in [1..n]$, of the membership function Γ^* are computed according to the following:

$$\Gamma^*_{ki} = \left[\sum_{k'=1}^{K} \left(\frac{f(o_i, v_k)}{f(o_i, v_{k'})} \right)^{\frac{1}{\alpha-1}} \right]^{-1} \tag{2.23}$$

Like in FCM algorithm, $\alpha > 1$ is an integer user-defined parameter needed for ensuring that Γ^*_{ki} range within $[0, 1]$ (instead of $\{0, 1\}$).

Finally, it easy to observe that the computational complexity of FCMdd is equal to that of the basic K-Medoids algorithm.

2.5 Cluster Validity

The effectiveness of hard clustering algorithms is evaluated by exploiting *cluster validity* methods. As regards soft clustering, there have been defined in the literature several methods for evaluating fuzzy partitions [WZ07]. However, one of the most common ways to evaluate soft clustering solutions is to firstly transform fuzzy partitions into crisp partitions, and then apply (hard) cluster validity methods. To this purpose, this section focuses only on methods for evaluating hard clusterings, which can be classified into three main categories, i.e., *external*, *internal*, and *relative* criteria [TK99].

Relative cluster validity criteria aim to assess the accuracy of a clustering algorithm by comparing the different results obtained by varying the values of the input parameters. The descriptions of external and internal criteria are provided in the following, along with the definitions of the specific external and internal measures involved into the experiments of this thesis.

2.5.1 External Cluster Validity Criteria

According to external cluster validity criteria, the results of any clustering algorithm is evaluated by resorting to some prior-knowledge available from the input dataset. This thesis focuses on external criteria aimed at comparing the clustering $\mathcal{C}$ obtained by a specific algorithm with respect to a *reference classification* $\widetilde{\mathcal{C}}$ available for the data at hand. The reference classification of a given dataset represents the "true" partition that should be ideally recognized by any clustering algorithm. Therefore, the main goal of external criteria is to evaluate how well a clustering fits a predefined scheme of known classes (i.e., natural clusters).

F1-Measure

F1-Measure (F1) [van79] is one the most commonly used Information Retrieval external criteria. It is defined as the harmonic mean of values that

express the notions of *Precision* and *Recall*. Given a set D of data objects, and two clustering solutions $\mathcal{C} = \{C_1, \ldots, C_K\}$, $\widetilde{\mathcal{C}} = \{\widetilde{C}_1, \ldots, \widetilde{C}_{K'}\}$ defined over D, Precision $\mathrm{P}_{kk'}$ of cluster $C_k \in \mathcal{C}$ with respect to cluster $\widetilde{C}_{k'} \in \widetilde{\mathcal{C}}$ is the fraction of the objects in C_k that are contained into $\widetilde{C}_{k'}$. Recall $\mathrm{R}_{kk'}$ of cluster $C_k \in \mathcal{C}$ with respect to cluster $\widetilde{C}_{k'} \in \widetilde{\mathcal{C}}$ is the fraction of the objects in $\widetilde{C}_{k'}$ that are contained into C_k. Formally:

$$\mathrm{P}_{kk'} = \frac{|C_k \cap \widetilde{C}_{k'}|}{|C_k|} \tag{2.24}$$

$$\mathrm{R}_{kk'} = \frac{|C_k \cap \widetilde{C}_{k'}|}{|\widetilde{C}_{k'}|} \tag{2.25}$$

If $|C_k| = 0$ (resp. $|\widetilde{C}_{k'}| = 0$), it can be reasonably assumed that $\mathrm{P}_{kk'} = 0$ (resp. $\mathrm{R}_{kk'} = 0$).

Definition 2.5 (F1-Measure). *Let D be a set of data objects, and $\mathcal{C} = \{C_1, \ldots, C_K\}$, $\widetilde{\mathcal{C}} = \{\widetilde{C}_1, \ldots, \widetilde{C}_{K'}\}$ be two clustering solutions defined over D.* F1-Measure *of $\mathcal{C}$ with respect to $\widetilde{\mathcal{C}}$ is defined as follows:*

$$F1(\mathcal{C}, \widetilde{\mathcal{C}}) = \frac{1}{|D|} \sum_{k'=1}^{K'} |\widetilde{C}_{k'}| \max_{k \in [1..K]} F1_{kk'} \tag{2.26}$$

where

$$F1_{kk'} = \frac{2\ \mathrm{P}_{kk'}\ \mathrm{R}_{kk'}}{\mathrm{P}_{kk'} + \mathrm{R}_{kk'}}$$

$F1$ is not symmetric and ranges within $[0, 1]$, where higher values refer to higher similarity between the clusterings to be compared.

Normalized Mutual Information

Normalized Mutual Information (NMI) [CT06] is a measure able to quantify the statistical information shared between two distributions. NMI can be used to express a sound indication of the degree of shared information between any pair of clustering solutions. Thus, NMI can be exploited to measure the similarity between any two clusterings $\mathcal{C}$ and $\widetilde{\mathcal{C}}$ defined over the same set D of data objects.

Definition 2.6 (Normalized Mutual Information). *Let D be a set of data objects, and $\mathcal{C} = \{C_1, \ldots, C_K\}$, $\widetilde{\mathcal{C}} = \{\widetilde{C}_1, \ldots, \widetilde{C}_{K'}\}$ be two clustering solutions defined over D.* Normalized Mutual Information *between $\mathcal{C}$ and $\widetilde{\mathcal{C}}$ is defined as follows:*

$$NMI(\mathcal{C},\widetilde{\mathcal{C}}) = \frac{\sum_{k=1}^{K}\sum_{k'=1}^{K'} |C_k \cap \widetilde{C}_{k'}| \log\left(\frac{|D|\ |C_k \cap \widetilde{C}_{k'}|}{|C_k||\widetilde{C}_{k'}|}\right)}{\sqrt{\left(\sum_{k=1}^{K} |C_k| \log\frac{|C_k|}{|D|}\right)\left(\sum_{k'=1}^{K'} |\widetilde{C}_{k'}| \log\frac{|\widetilde{C}_{k'}|}{|D|}\right)}} \tag{2.27}$$

It can be easily proved that $NMI(\mathcal{C},\widetilde{\mathcal{C}}) \in [0,1]$, and $NMI(\mathcal{C},\widetilde{\mathcal{C}}) = NMI(\widetilde{\mathcal{C}},\mathcal{C})$, $\forall \mathcal{C},\widetilde{\mathcal{C}}$ defined over D.

Conditional Entropy

Conditional Entropy (or simply *entropy*) [Dom01] is a non-symmetric external cluster validity criterion based on information theoretic concepts similar to those exploited by NMI. Given two clusterings $\mathcal{C}$, $\widetilde{\mathcal{C}}$ to be compared, entropy is based on the probability that an object in the cluster C_k belongs to the cluster $\widetilde{C}_{k'}$, i.e., $\Pr(\widetilde{C}_{k'}|C_k)$, for each $C_k \in \mathcal{C}$, $\widetilde{C}_{k'} \in \widetilde{\mathcal{C}}$.

Definition 2.7. *Let D be a set of data objects, and $\mathcal{C} = \{C_1,\ldots,C_K\}$, $\widetilde{\mathcal{C}} = \{\widetilde{C}_1,\ldots,\widetilde{C}_{K'}\}$ be two clustering solutions defined over D.* Entropy *of $\mathcal{C}$ with respect to $\widetilde{\mathcal{C}}$ is defined as follows:*

$$entr(\mathcal{C},\widetilde{\mathcal{C}}) = \frac{1}{|D|}\sum_{k=1}^{K} -\frac{|C_k|}{\log K'} \sum_{k'=1}^{K'} \Pr(\widetilde{C}_{k'}|C_k)\ \log\Pr(\widetilde{C}_{k'}|C_k) \tag{2.28}$$

where

$$\Pr(\widetilde{C}_{k'}|C_k) = \begin{cases} \dfrac{|C_k \cap \widetilde{C}_{k'}|}{|C_k|} & \text{if } |C_k| > 0 \\ 0 & \text{otherwise} \end{cases}$$

According to the above definition, entropy ranges within $[0,1]$, and the lower $entr(\mathcal{C},\widetilde{\mathcal{C}})$, the higher the similarity between the clustering solutions $\mathcal{C}$ and $\widetilde{\mathcal{C}}$.

2.5.2 Internal Cluster Validity Criteria

Internal cluster validity criteria aim to evaluate a clustering solution using only quantities and features inherent to the input dataset. This goal is typically accomplished by directly accessing the feature values of the data objects and exploiting the distance measure used for producing the results.

Intra-cluster and Inter-cluster Distances

Intra-cluster distance and *inter-cluster* distance are two popular internal criteria to evaluate the quality of clustering solutions in terms of cluster cohesiveness and cluster separation, respectively. Given a clustering solution $\mathcal{C}$, the intra-cluster distance of any cluster in $\mathcal{C}$ is defined as the mean of the pairwise distances between the objects in the cluster. The inter-cluster distance between any pair of clusters in $\mathcal{C}$ is computed by averaging the pairwise distances between the objects in the two clusters. The overall intra-cluster ($intra(\mathcal{C})$) and inter-cluster ($inter(\mathcal{C})$) distances are computed by averaging the local values.

Definition 2.8 (intra-cluster distance). *Let D be a set of data objects, $\mathcal{C} = \{C_1, \ldots, C_K\}$ be a clustering solution defined over D, and $f : D \times D \rightarrow \Re$ be a distance function between the objects in D. The* intra-cluster distance *of $\mathcal{C}$ is defined as follows:*

$$intra(\mathcal{C}) = \frac{1}{|\mathcal{C}|} \sum_{k=1}^{K} \frac{1}{|C_k|(|C_k| - 1)} \sum_{o \in C_k} \sum_{\substack{o' \in C_k, \\ o' \neq o}} f(o, o') \tag{2.29}$$

Definition 2.9 (inter-cluster distance). *Let D be a set of data objects, $\mathcal{C} = \{C_1, \ldots, C_K\}$ a clustering solution defined over D, and $f : D \times D \rightarrow \Re$ a distance function between the objects in D. The* inter-cluster distance *of $\mathcal{C}$ is defined as follows:*

$$inter(\mathcal{C}) = \frac{1}{|\mathcal{C}|(|\mathcal{C}| - 1)} \sum_{\substack{\langle C, C' \rangle \in \mathcal{C} \times \mathcal{C}, \\ C \neq C'}} \frac{1}{|C||C'|} \sum_{o \in C} \sum_{o' \in C'} f(o, o') \tag{2.30}$$

It is easy to note that the lower $intra(\mathcal{C})$ and the higher $inter(\mathcal{C})$, the higher the quality of clustering $\mathcal{C}$. Intra-cluster and inter-cluster distances can be combined in order to measure the accuracy of any clustering in terms of a single numerical value. As an example, the *quality* $qual(\mathcal{C})$ of the clustering solution $\mathcal{C}$ is defined as:

$$qual(\mathcal{C}) = inter(\mathcal{C}) - intra(\mathcal{C}) \tag{2.31}$$

Cophenetic Correlation Coefficient

Hierarchical clustering results can be evaluated in terms of the *cophenetic correlation coefficient* [SR62]. This measure (ranging within $[-1, 1]$) aims to evaluate a dendrogram produced by any hierarchical algorithm according to how it preserves the pairwise distances between the original data points. Intuitively, the higher the cophenetic correlation value for a dendrogram, the higher the compactness and the better the quality achieved by the hierarchical algorithm.

Definition 2.10 (cophenetic correlation coefficient). *Let $D = \{o_1, \ldots, o_n\}$ be a set of data objects, $\mathbf{T}_\ell = [\mathcal{L}_0, \ldots, \mathcal{L}_Q]$ a level-organized dendrogram representing a cluster hierarchy defined over D, and $f : D \times D \to \Re$ a distance function between the objects in D. The* cophenetic correlation coefficient *with respect to D and $\mathbf{T}_\ell$ is defined as:*

$$CPCC(D, \mathbf{T}_\ell) = \frac{\sum_{i=1}^{n-1} \sum_{i'=i+1}^{n} \Big(f(o_i, o_{i'}) - \overline{f}\Big)\Big(\tau(o_i, o_{i'}) - \overline{\tau}\Big)}{\sqrt{\left(\sum_{i=1}^{n-1} \sum_{i'=i+1}^{n} \Big(f(o_i, o_{i'}) - \overline{f}\Big)^2\right)\left(\sum_{i=1}^{n-1} \sum_{i'=i+1}^{n} \Big(\tau(o_i, o_{i'}) - \overline{\tau}\Big)^2\right)}} \tag{2.32}$$

where $\tau(o, o')$ denotes the dendrogrammatic distance between o and o', which indicates the level of the dendrogram where o and o' are first joined together:

$$\tau(o, o') = \min\left\{q \in [1..Q] \;\middle|\; \sum_{L \in \mathcal{L}_q} I[o \in L \cap o' \in L] = 1\right\}$$

and

$$\overline{f} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{i'=i+1}^{n} f(o_i, o_{i'})$$

$$\overline{\tau} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{i'=i+1}^{n} \tau(o_i, o_{i'})$$

Part II

Uncertainty in Data Clustering

– *Data Level* –

3

Uncertainty in Data Representation: Background

Abstract This chapter provides background to uncertainty in data representation. The problem of properly modeling data uncertainty has been addressed and solved in various ways in the literature. We are particularly interested in the model exploited for representing the so-called uncertain objects, which are at the basis of one of the topics of this thesis. In this respect, a further goal is to provide a rough overview on the main algorithms for clustering uncertain objects so far defined in the literature. Such algorithms fall into two main categories, namely partitional relocation and density-based.

3.1 Modeling Uncertainty in Data Representation

Depending on the application domain, several definitions of uncertainty in data representation have been provided in the literature (e.g., [IL84, AKG87, Sad91, LLRS97, DS04, GT06, Agg07]).

In general, for relational databases, uncertainty can be considered at different levels of granularity, i.e., at *table*, *tuple* or *attribute* level [TXC07]. Table-based uncertainty refers to the "coverage" of a table, i.e., how many tuples are present in a table [Wid05, BSHW06]. According to tuple-based granularity, the uncertainty is due to the fact that each individual tuple in the database may exist or may not [Fuh95, DS04, DS05]. Finally, attribute-based models concern uncertainty about the values of single attributes (features) of the tuples in the database [WSCY99, PJ99, CKP03, DGMH04].

Regardless of the granularity, the specific way to express uncertainty can involve *fuzzy models* [GUP06], *evidence-oriented models* [Lee92, LSS96], or *probabilistic models* [SBHW06]. In fuzzy models, fuzzy entities, fuzzy attributes, fuzzy relationships, fuzzy aggregation, fuzzy constraints, and so on are used to model uncertainty [ZLPZ08]. The *Dempster-Shafer Theory of Evidence* [Dem67, Sha76] is exploited for modeling uncertainty according to evidence-oriented models. Probabilistic models exploit probability values and distributions in order to quantify uncertainty.

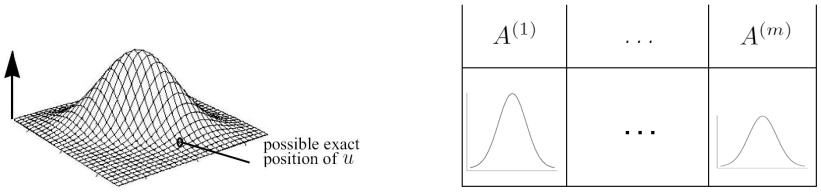


Fig. 3.1. Graphical representation of (a) a multivariate uncertain object and (b) a univariate uncertain object

The focus of this thesis is on data containing attribute-level uncertainty modeled according to probabilistic models. In particular, the specific probabilistic representation taken into account in this thesis resorts to *probability distributions*, which describe the likelihood that any given object appears at each position in a multidimensional space [CKP03, KP05a, CCKN06]. Probabilistic models may in principle involve other forms to represent uncertainty, such as statistical properties (e.g., error percentage or deviation from an expected value); however, while these properties provide a concise information about an uncertain set of values, probability distributions offer a more-accurate solution in uncertainty modeling.

Data objects described in terms of probability distributions are hereinafter referred to as *uncertain objects*.

Uncertain Objects

The two basic models generally used for representing uncertain objects are the *multivariate* uncertainty model and the *univariate* uncertainty model.

In a multivariate uncertainty model, an m-dimensional uncertain object is defined in terms of an m-dimensional region and a multivariate *probability density function* (pdf), which stores the probability according to which the exact representation of the object coincides with any point in the region. In a univariate uncertainty model, an m-dimensional uncertain object has, for each attribute, an interval and a univariate pdf that assigns a probability value to each point within the interval. Depending on the specific application context, an uncertain object can be modeled according to one of the two models. An example of multivariate and univariate uncertain objects is shown in Fig. 3.1.

It should be noted that, in general, it is not straightforward to map multivariate objects into univariate objects, and vice versa. Indeed, deriving a multivariate representation from a univariate one cannot be performed without knowing in advance the conditional pdfs or making specific statistical assumptions (e.g., statistical independence). On the other hand, transforming a multivariate model into a univariate one requires the computation of the marginal pdfs from the joint distribution; this can be a very complex and

inefficient operation depending on the form and/or the dimensionality of the distribution.

Definition 3.1 (multivariate uncertain object). *A* multivariate uncertain object u *is a pair* $(\mathcal{R}, p)$*, where* $\mathcal{R} = [a^{(1)}, b^{(1)}] \times \cdots \times [a^{(m)}, b^{(m)}]$ *is the* m*-dimensional region in which* u *is defined and* $p : \Re^m \rightarrow \Re_0^+$ *is the probability density function of* u *at each point* $\mathbf{x} \in \Re^m$*, such that:*

$$\int_{\mathbf{x} \in \Re^m \setminus \mathcal{R}} p(\mathbf{x}) \mathrm{d}\mathbf{x} = 0 \tag{3.1}$$

$$p(\mathbf{x}) > 0, \quad \forall \mathbf{x} \in \mathcal{R} \tag{3.2}$$

Definition 3.2 (univariate uncertain object). *A* univariate uncertain object u *is a tuple* $(A^{(1)}, \ldots, A^{(m)})$*. Each attribute* $A^{(j)}$*,* $j \in [1..m]$*, is a pair* $(\mathcal{I}^{(j)}, p^{(j)})$*, where* $\mathcal{I}^{(j)} = [a^{(j)}, b^{(j)}]$ *is the interval of definition of* $A^{(j)}$*, and* $p^{(j)} : \Re \rightarrow \Re_0^+$ *is the probability density function that assigns a probability value to each* $x \in \Re$*, such that:*

$$\int_{x \in \Re \setminus \mathcal{I}^{(j)}} p^{(j)}(x) \mathrm{d}x = 0 \tag{3.3}$$

$$p^{(j)}(x) > 0, \quad \forall x \in \mathcal{I}^{(j)} \tag{3.4}$$

The definitions above involve the region/intervals of definition for the pdf(s) associated to any uncertain object. This represents a key aspect, since in many real applications these objects are defined according to limited region/intervals. Moreover, this allows for defining an uncertain object in a more general way, since the case of uncertain object with unlimited region/intervals is also implicitly taken into account in these models.

Also, Defs. 3.1-3.2 refer to the most general case of uncertain objects modeled as continuous random variables and described by probability density functions. Nevertheless, without loss of generality, these definitions include the discrete case. Indeed, for any discrete multivariate uncertain object $u = (\mathcal{R}, p)$, the m-dimensional region $\mathcal{R} = [a^{(1)}, b^{(1)}] \times \cdots \times [a^{(m)}, b^{(m)}]$ bounds a discrete set S of m-dimensional points, and the probability distribution function p is a multivariate discrete pdf, also known as multivariate *probability mass function*, defined as $p : S \rightarrow \Re_0^+$. Analogously, for any discrete univariate uncertain object $u = (A^{(1)}, \ldots, A^{(m)})$, the j-th attribute ($j \in [1..m]$) is defined over an interval $\mathcal{I}^{(j)}$ which bounds a discrete set $S^{(j)}$, and has a function $p^{(j)} : S^{(j)} \rightarrow \Re_0^+$ which is a univariate probability mass function.

In the following, we assume statistical independence between any actual location $\mathbf{x}', \mathbf{x}'' \in \Re^m$ of any two multivariate/univariate uncertain objects u', u'' belonging to the same dataset D, i.e., we assume that

$$\Pr(u' \equiv \mathbf{x}' \cap u'' \equiv \mathbf{x}'') = \Pr(u \equiv \mathbf{x}') \ \Pr(u'' \equiv \mathbf{x}''), \ \ \forall u', u'' \in D, \ \forall \mathbf{x}', \mathbf{x}'' \in \Re^m$$

where $u \equiv \mathbf{x}$ denotes the event "*the actual location of uncertain object u is in the point* $\mathbf{x} \in \Re^m$".

Also, for any univariate object $u = ((\mathcal{I}^{(1)}, p^{(1)}), \ldots, (\mathcal{I}^{(m)}, p^{(m)}))$, we assume statistical independence among its m dimensions:

$$\Pr(u \equiv \mathbf{x}) = \prod_{j=1}^{m} p^{(j)}(x_j)$$

for each point $\mathbf{x} = [x_1, \ldots, x_m] \in \Re^m$.

3.2 Clustering of Uncertain Objects: State of the Art

In the context of uncertain data management, a lot of research has been mainly focused on data representation and modeling, indexing, query processing, and data mining (e.g., [DS07, ZLPZ08, AY09]). In particular, data mining applications have involved various tasks, such as classification [BZ04], outlier detection [AY08], association analysis [CKH07], and clustering [KP05b, KP05a, CCKN06, NKC+06, S. 07, KLC+08]. The most relevant approaches to clustering uncertain objects so far defined in the literature fall into two main categories, namely partitional relocation and density-based.

3.2.1 Partitional Relocation Methods

One of the earliest partitional relocation-based attempts to solve the problem of clustering uncertain objects is the *UK-Means* algorithm [CCKN06], which is an adaptation of the popular K-Means (cf. Chapt. 2) designed for handling uncertain objects. UK-Means suffers from two major weakness. The first one is related to an accuracy issue. Indeed, cluster centroids in UK-Means are computed as deterministic objects using the expected values of the pdfs of the clustered objects. Moreover, the efficiency of UK-Means is limited by the expensive computation of the *expected distance* (ED) between uncertain objects and cluster centroids (which are defined as deterministic objects), at each iteration of the algorithm.

In order to improve the UK-Means efficiency, [NKC+06] proposes some pruning techniques to avoid the computation of redundant EDs. Such techniques make use of lower- and upper-bounds ad-hoc defined for each ED to be calculated, in order to define a specific bounding-box for each uncertain object; these boxes allow for eliminating some candidate assignments of objects to clusters and avoiding the corresponding ED computation. However, a major problem of this approach is that it cannot guarantee high pruning (hence, high efficiency), as it depends on the features of the objects in the specific dataset.

Another pruning method to reduce the number of EDs calculation is described in [KLC+08]. Such a method exploits Voronoi diagrams and has been recognized as more effective than the basic bounding-box-based techniques.

In [S. 07], the *CK-Means* algorithm is proposed as a variant of UK-Means that exploits the moment of inertia of rigid bodies in order to reduce the execution time needed for computing EDs. Unfortunately, the soundness of the CK-Means criterion for the ED computation is guaranteed only if the mean squared error for the definition of the EDs is used and the distance function is based on the Euclidean norm.

The methods proposed in [NKC+06, S. 07, KLC+08] attempt to reduce the execution time required for computing the EDs. The Approximation by Single Gaussian (ASG) method [XH07] belongs to a more general approach, which aims to approximate the distance between an uncertain object and another uncertain object. ASG has been proved to achieve accuracy close to the exact distance calculation while improving the efficiency. Another approach to the distance computation consists in defining a univariate pdf, or *fuzzy distance function*, for each pair of uncertain objects; this univariate pdf stores a probability for each distance value for two given objects. The final distance, called *fuzzy distance* (FD), between the objects is computed by deriving an aggregated, representative value (e.g., expected value) from the pdf of those objects. This method has been presented originally in [KP05b] and proved to be more effective than the standard Euclidean distance applied to vectors of deterministic values.

3.2.2 Density-based Methods

Devising a fuzzy distance function is a key aspect in density-based approaches to clustering uncertain objects [KP05a, KP05b]. In [KP05a], the $\mathcal{F}DBSCAN$ is proposed as a fuzzy version of the popular DBSCAN (cf. Chapt. 2, Sect. 2.2.2), which uses fuzzy distance functions to compute the core object and reachability probabilities. A similar approach is presented in $\mathcal{F}OPTICS$ [KP05b]. Like the well-known density-based clustering algorithm OPTICS (cf. Chapt. 2, Sect. 2.2.2), $\mathcal{F}$OPTICS produces an augmented ordering of the objects based on the notion of fuzzy object reachability-distance; this ordering can eventually be used to derive a cluster hierarchy.

While the majority of algorithms proposed for clustering uncertain objects are based on either partitional or density-based schemes, it should be noted that there is poor research on (agglomerative) hierarchical clustering of uncertain data. $\mathcal{F}$OPTICS is close to a hierarchical scheme, although is significantly different in constructing the cluster hierarchy. Indeed, $\mathcal{F}$OPTICS outputs a reachability plot, whereas the classic result of any hierarchical clustering algorithm is a dendrogram. While reachability plots may be easier to read than dendrograms for very large datasets, a dendrogram gives a clearer view of the cluster membership of individual objects than a reachability plot for a wide set of real cases [J. 03]; also, when a reachability plot is produced, building the corresponding cluster hierarchy needs for a further step, which is typically performed by employing either visualization or automatic techniques [ABKS99].

4

Clustering Uncertain Objects via K-Medoids

Abstract The centroid-based approach to clustering uncertain objects used in the UK-Means algorithm presents two major weaknesses that are related to: *(i)* an accuracy issue, since cluster centroids are computed as deterministic objects using the expected values of the pdfs of the clustered objects; and, *(ii)* an efficiency issue, since the expected distance between uncertain objects and cluster centroids is computationally expensive.

In this chapter, we address the problem of clustering uncertain objects by proposing a new algorithm, called *UK-Medoids*, which is designed to overcome the above issues. UK-Medoids employs distance functions properly defined for uncertain objects, and exploits a K-Medoids scheme. Experiments have shown that UK-Medoids outperforms existing algorithms from an accuracy viewpoint while achieving reasonably good efficiency.

4.1 Introduction

As mentioned in Chap. 3, the popular K-Means algorithm has been recently adapted to the uncertain objects domain [CCKN06]. However, the resulting algorithm, called UK-Means, has two major weak points. First, cluster centroids are defined as deterministic objects and computed as the mean of the expected values over the pdfs of the uncertain objects in the cluster; defining centroids in this way may result in loss of accuracy, since only the expected values of the pdfs of the uncertain objects are taken into account. Second, the computation of the Expected Distance (ED) between cluster centroids and uncertain objects is computationally expensive, as it requires non-trivial numerical integral estimations; this represents an efficiency bottleneck at each iteration of the algorithm.

In this chapter, we present *UK-Medoids* [GPT08a], a new algorithm for clustering uncertain objects based on the K-Medoids clustering scheme. The proposed algorithm exploits a distance function for uncertain objects, which is not limited to consider only scalar values derived from the pdfs associated to the objects (e.g., pdf expected values). This allows for better estimating

the real distance between two uncertain objects, leading to significant improvement of the clustering quality. Also, UK-Medoids does not require any expensive operation to be repeated at each iteration; indeed, the computation of the distances between uncertain objects in the dataset is performed only once, thus guaranteeing a significant improvement of the efficiency with respect to UK-Means.

4.2 Uncertain Distance

To measure the distance between uncertain objects, we need to devise a suitable notion of *uncertain distance*, which is involved into the proposed clustering algorithm. Uncertain distance is defined in terms of an *uncertain distance function*. In order to make the uncertain distance independent from the chosen uncertainty model, we provide definitions of uncertain distance function for both multivariate and univariate uncertainty models (cf. Chap. 2, Sect. 3.1).

Definition 4.1 (uncertain distance function). *Given a set* $D = \{u_1, \ldots, u_n\}$ *of uncertain objects, the* uncertain distance function *defined over* D *is a function* $\Theta : D \times D \times \Re \rightarrow \Re_0^+$*, for which the following conditions hold:*

$$\int_{x \in \Re} \Theta(u_i, u_{i'}, x)\ \mathrm{d}x = 1, \quad \forall u_i, u_{i'} \in D,$$

$$\Theta(u_i, u_{i'}, x) = \begin{cases} 1, & \textit{if}\ \ i = i', x = 0 \\ 0, & \textit{if}\ \ i = i', x \neq 0 \end{cases}$$

For any pair of uncertain objects $u_i, u_{i'}, i \neq i'$, Θ can be derived from the pdfs associated to the uncertain objects. The definition of Θ depends on the uncertainty model used for representing u_i and $u_{i'}$.

Uncertain Distance Function for Multivariate Objects

If $u_i = (\mathcal{R}_i, p_i)$, $u_{i'} = (\mathcal{R}_{i'}, p_{i'})$ are multivariate uncertain objects, Θ is defined as:

$$\Theta(u_i, u_{i'}, x) = \int_{\mathbf{x} \in \mathcal{R}_i} \int_{\mathbf{x}' \in \mathcal{R}_{i'}} I[f(\mathbf{x}, \mathbf{x}') = x]\ p_i(\mathbf{x})\ p_{i'}(\mathbf{x}')\ \mathrm{d}\mathbf{x}\ \mathrm{d}\mathbf{x}' \tag{4.1}$$

where $f(\mathbf{x}, \mathbf{x}')$ is a distance function between any pair $\mathbf{x}, \mathbf{x}' \in \Re^m$ (e.g., Euclidean norm), and $I[A]$ is the *indicator function*, which is equal to 1 when the event A occurs, 0 otherwise.

Uncertain Distance Function for Univariate Objects

If $u_i = ((\mathcal{I}_i^{(1)}, p_i^{(1)}), \ldots, (\mathcal{I}_i^{(m)}, p_i^{(m)}))$, $u_{i'} = ((\mathcal{I}_{i'}^{(1)}, p_{i'}^{(1)}), \ldots, (\mathcal{I}_{i'}^{(m)}, p_{i'}^{(m)}))$ are univariate uncertain objects, Θ is defined as:

$$\Theta(u_i, u_{i'}, x) = \int_{x_1 \in \Re} \cdots \int_{x_m \in \Re} I[f'(x_1, \ldots, x_m) = x] \prod_{j=1}^{m} \Psi^{(j)}(u_i, u_{i'}, x_j)\ \mathrm{d}x_1 \cdots \mathrm{d}x_m \tag{4.2}$$

where

- $\Psi^{(j)} : D \times D \times \Re \to \Re$,
- $\Psi^{(j)}(u_i, u_{i'}, x_j) = \int_{r \in \mathcal{I}_i^{(j)}} \int_{s \in \mathcal{I}_{i'}^{(j)}} I[|r - s| = x_j]\ p_i^{(j)}(r)\ p_{i'}^{(j)}(s)\ \mathrm{d}r\ \mathrm{d}s, \quad j \in [1..m]$,
- $f' : \Re^m \to \Re$ is a function that computes a scalar value from the components of a vector $(x_1, \ldots, x_m)$. In this thesis, this function is defined as $f' = \sqrt{\sum_{j=1}^{m} {x_j}^2}$.

It can be proved that the condition $\int_{x \in \Re} \Theta(u_i, u_{i'}, x)\ \mathrm{d}x = 1$ holds for both the definitions of Θ, for all $u_i, u_{i'}$ in the dataset D.

Given an uncertain distance function Θ, it can be provided a definition of uncertain distance by extracting a single, well-representative numerical value from Θ.

Definition 4.2 (uncertain distance). *Given a set $D = \{u_1, \ldots, u_n\}$ of uncertain objects, let Θ be the uncertain distance function defined over D. The* uncertain distance *is a function $\theta : D \times D \to \Re_0^+$, which is defined as:*

$$\theta(u_i, u_{i'}) = \int_{x \in \Re} x\ \Theta(u_i, u_{i'}, x)\ \mathrm{d}x \tag{4.3}$$

According to (4.3), $\theta(u_i, u_{i'})$ is the expected value of the uncertain distance function θ between u_i and $u_{i'}$. Note that, if u_i, $u_{i'}$ are multivariate uncertain objects, $\theta(u_i, u_{i'})$ can be directly computed as:

$$\theta(u_i, u_{i'}) = \int_{\mathbf{x} \in \mathcal{R}_i} \int_{\mathbf{x}' \in \mathcal{R}_{i'}} f(\mathbf{x}, \mathbf{x}')\ p_i(\mathbf{x})\ p_{i'}(\mathbf{x}')\ \mathrm{d}\mathbf{x}\ \mathrm{d}\mathbf{x}' \tag{4.4}$$

whereas, if u_i, $u_{i'}$ are univariate uncertain objects, $\theta(u_i, u_{i'})$ can be calculated as:

$$\theta(u_i, u_{i'}) = f'(\psi^{(1)}(u_i, u_{i'}), \ldots, \psi^{(m)}(u_i, u_{i'})) \tag{4.5}$$

where

$$\psi^{(j)}(u_i, u_{i'}) = \int_{x \in \mathcal{I}_i^{(j)}} \int_{x' \in \mathcal{I}_{i'}^{(j)}} |x - x'|\ p_i^{(j)}(x)\ p_{i'}^{(j)}(x')\ \mathrm{d}x\ \mathrm{d}x', \quad j \in [1..m].$$

4.3 UK-Medoids Algorithm

In this section the proposed K-Medoids-based algorithm for clustering uncertain objects, called *UK-Medoids* is presented. The outline of UK-Medoids is given in Algorithm 4.1.

Algorithm 4.1 UK-Medoids

Input: a set $D = \{u_1, \ldots, u_n\}$ of uncertain objects; the number K of output clusters
Output: a clustering $\mathcal{C}$ of D

1: compute distances $\theta(u_i, u_{i'}), \forall u_i, u_{i'} \in D$
2: compute the set $\mathcal{V} = \{v_1, \ldots, v_K\}$ of initial medoids
3: **repeat**
4: $\quad \mathcal{V}' \leftarrow \mathcal{V}$
5: $\quad \mathcal{V} \leftarrow \emptyset$
6: $\quad \mathcal{C} = \{C_1, \ldots, C_K\} \leftarrow \{\emptyset, \ldots, \emptyset\}$
7: $\quad$ **for all** $u \in D$ **do**
8: $\quad\quad$ {*assign each object to the closest cluster, based on its uncertain distance to cluster medoids*}
9: $\quad\quad v_k \leftarrow \arg\min_{v' \in \mathcal{V}'} \theta(u, v')$
10: $\quad\quad C_k \leftarrow C_k \cup \{u\}$
11: $\quad$ **end for**
12: $\quad$ **for all** $C \in \mathcal{C}$ **do**
13: $\quad\quad$ {*recompute the medoid of each cluster*}
14: $\quad\quad v \leftarrow \arg\min_{u \in C} \sum_{u' \in C} \theta(u, u')$
15: $\quad\quad \mathcal{V} \leftarrow \mathcal{V} \cup \{v\}$
16: $\quad$ **end for**
17: **until** $\mathcal{V} = \mathcal{V}'$

The input for the UK-Medoids algorithm is a dataset D of n uncertain objects and the number K of clusters to be discovered, and the output is a set $\mathcal{C}$ of K clusters, i.e., a partition of D. Initially, all the uncertain distances between any pair of objects $u_i, u_{i'} \in D$ are computed (Line 1). The distances are calculated only once and are used at each iteration of the algorithm. Afterward, the set of K initial medoids is computed (Line 2). The initial medoids can be selected by means of either random chance or a suitable procedure aimed to choose well-separated medoids (e.g., that proposed for the Partitioning Around Medoids (PAM) algorithm [KR87]).

After the initialization steps, the algorithm performs the main loop (starting from Line 3) which comprises two phases. In the first phase (Lines $7-11$), each object u in D is assigned to the cluster represented by the medoid v closest to u. In the second phase, the medoids in the set $\mathcal{V}$ are recomputed according to the objects assigned to each cluster (Lines $12-16$). Such phases are iteratively repeated until a local optimum has not been reached, i.e., there is no change in the current $\mathcal{V}$ with respect to the previous iteration (Line 17).

Given a dataset D of n uncertain objects, it can be proved that the "online" computational complexity of Alg. 4.1 (i.e., the complexity of the main loop, Lines $3-16$) is $\mathcal{O}(I\ n^2)$, where I is the number of iterations needed for the convergence of the algorithm.

4.4 Experimental Evaluation

We devised an experimental evaluation aimed to assess the ability of UK-Medoids in clustering uncertain objects, both in terms of accuracy and efficiency. We also compared UK-Medoids to K-Means-based algorithms, i.e., UK-Means and its variant CK-Means (cf. Chap. 3).

4.4.1 Evaluation Methodology

Datasets

The experimental analysis was performed on benchmark datasets from the UCI Machine Learning Repository [ANml]. We chose four datasets with numerical real-value attributes, namely Iris, Wine, Glass, and Ecoli (cf. Appendix A).

All the selected datasets originally contain deterministic values, hence the uncertainty was synthetically generated for each object of any dataset. In particular:

- Generation of univariate uncertainty — For each univariate object u, we generated the uncertain interval $\mathcal{I}^{(j)}$ and the pdf $p^{(j)}$ defined over $\mathcal{I}^{(j)}$, for each attribute $A^{(j)}$, $j \in [1..m]$. The interval $\mathcal{I}^{(j)}$ was randomly chosen as a subinterval within $[min_{u_j}, max_{u_j}]$, where min_{u_j} (resp. max_{u_j}) is the minimum (resp. maximum) deterministic value of the j-th attribute over all the objects belonging to the same ideal class of u.
  As concerns $p^{(j)}$, we considered two continuous density functions, namely *Uniform* and *Normal* pdfs, and *Binomial* as a discrete mass function. We set the parameters of Normal and Binomial pdfs in such a way that their mode corresponded to the deterministic value of the j-th attribute of the object u
- Generation of multivariate uncertainty — Similarly to the univariate case, for each multivariate object u we generated the uncertainty region $\mathcal{R}$ as the product of the intervals randomly generated for each attribute of u. The distributions involved were (multivariate) Uniform and Normal pdfs. The strategy for setting the parameters of the pdfs was the same as for the univariate case.

Since univariate and multivariate models gave similar results, here we report only results on the univariate models.

Table 4.1. UK-Medoids vs. competing methods: clustering quality results (F1-Measure)

dataset	*pdf*	UK-Means	CK-Means	**UK-Medoids**
Iris	Uniform	0.45	*0.50*	**0.84**
	Normal	0.84	*0.85*	**0.88**
	Binomial	*0.62*	0.58	**0.87**
Wine	Uniform	0.46	*0.50*	**0.80**
	Normal	0.69	*0.70*	**0.70**
	Binomial	*0.63*	0.58	**0.73**
Glass	Uniform	0.26	*0.29*	**0.71**
	Normal	*0.63*	0.59	**0.68**
	Binomial	0.27	*0.29*	**0.67**
Ecoli	Uniform	0.30	*0.33*	**0.73**
	Normal	0.73	*0.74*	**0.77**
	Binomial	*0.50*	0.44	**0.72**

Cluster Validity

To assess the quality of clustering solutions we exploited the availability of reference classifications for the datasets. The objective was to evaluate how well a clustering fits a predefined scheme of known classes (natural clusters). To this purpose, we resorted to the F1-Measure external validity criterion (cf. Def. 2.5).

Settings

In K-Means-based approaches, the set of initial centroids is randomly selected. Therefore, to avoid that clustering results were biased by random chance, we averaged accuracy and efficiency measurements over 100 different runs. We made a similar choice also for UK-Medoids, since we noted that the use of a refined strategy for selecting initial medoids (e.g., the procedure proposed in [KR87]) gave no significant improvement with respect to random selection.

We computed the integrals involved into the distances calculation by taking into account lists of samples derived from the pdfs. To accomplish this, we employed the classic *Monte Carlo* sampling method.[1] We also performed a preliminary tuning phase to properly set the number of samples S; in particular, for each method and dataset, we chose S in such a way that there was no significant improvement in accuracy for any $S' > S$. In general, the optimal S depended on the width of the uncertainty interval/region; however, according to our experiments, 50 and $400 \div 500$ samples represented a reasonably good choice, for univariate and multivariate uncertainty model, respectively.

[1] We used the SSJ library, available at http://www.iro.umontreal.ca/∼simardr/ssj/

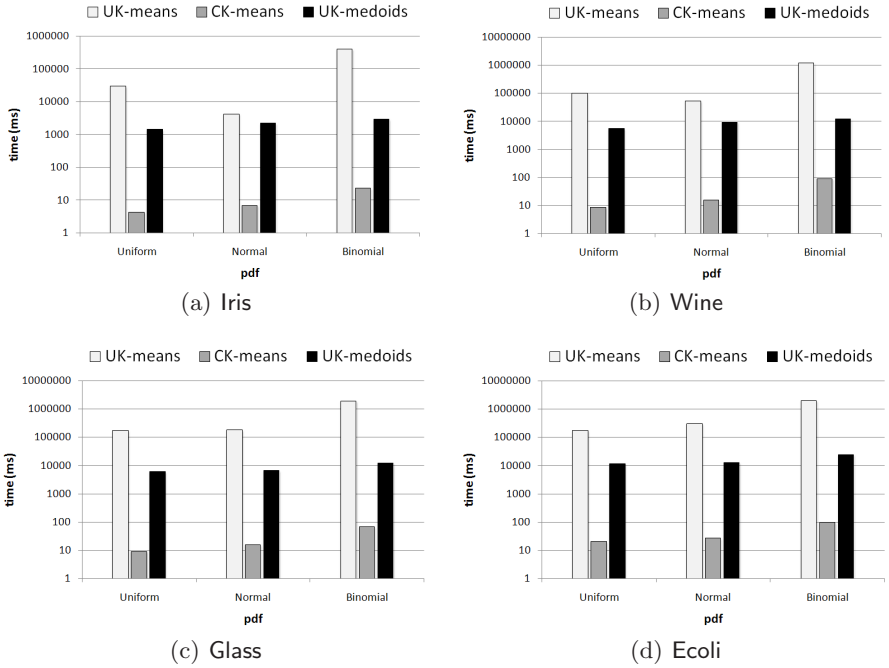


Fig. 4.1. UK-Medoids vs. competing methods: clustering time performances

4.4.2 Results

Accuracy

Table 4.1 summarizes the F1-Measure results obtained by UK-Medoids and the other methods. We can observe that UK-Medoids drastically outperformed UK-Means and CK-Means on all the datasets, with Uniform and Binomial pdfs. In particular, compared to the best competing method(s), the accuracy improvement obtained by UK-Medoids was from 34% to 42% with Uniform pdfs and from 10% to 38% with Binomial pdfs. In case of Normal pdfs, UK-Medoids performed $3 \div 5\%$ better than the other methods on three datasets, whereas all the methods behaved similarly in Wine. The reduction of gap between UK-Medoids and K-Means-based approaches on Normal pdfs can be explained in that, according to our uncertainty generation scheme, the expected value of a Normal pdf associated to any attribute of each uncertain object was set equal to the deterministic value of the attribute for that object. This allowed the centroid generation strategy of UK-Means and CK-Means to perform well in that case.

It should be also noted that UK-Means and CK-Means performed similarly for all the pdfs and datasets, as expected, since they employ a similar clustering scheme; the only differences between the two methods are due to

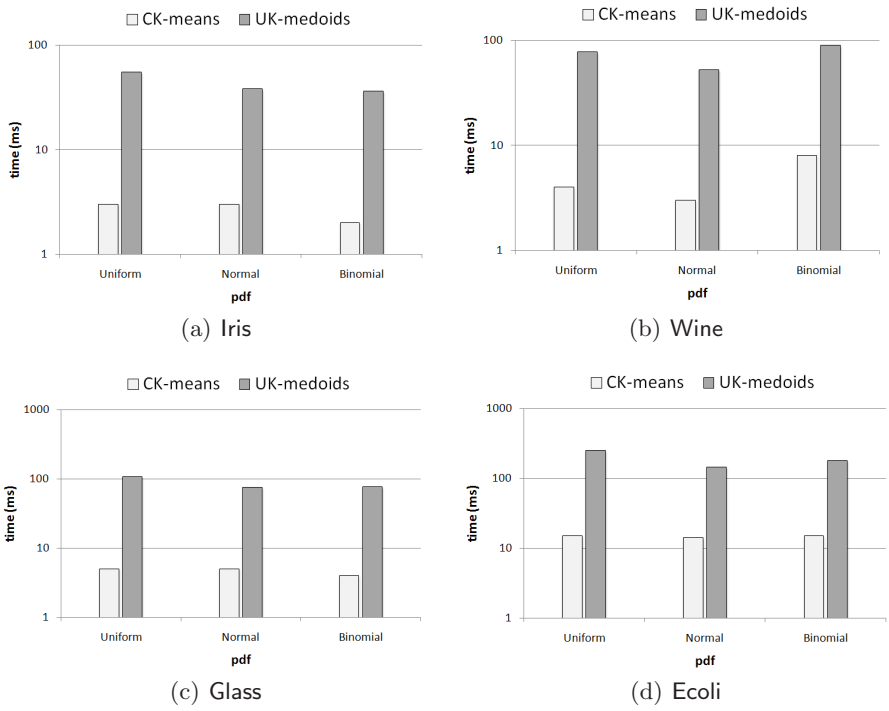


Fig. 4.2. UK-Medoids vs. CK-Means: performance of the algorithm runtimes (pre-computing phases are ignored)

random choices, such as selection of initial centroids and pdf sampling for the computation of the integrals.

Efficiency

To evaluate the efficiency of UK-Medoids and the competing methods, we measured their time performances in clustering uncertain objects.[2] Fig. 4.1 shows the total execution times (in milliseconds) obtained by the methods on the various datasets. For UK-Medoids and CK-Means, we calculated the sum of the times obtained for the pre-computing phase (i.e., uncertain distances computation for UK-Medoids and cluster centroids computation for CK-Means), along with the algorithm runtimes.

In the figure, it can be noted that UK-Medoids was $1 \div 2$ orders of magnitude faster than UK-Means, which was the slowest method on all datasets. The slowness of UK-Means is mainly due to the EDs computation needed for each object in the dataset, at each iteration of the algorithm.

[2] Experiments were conducted on a platform Intel Pentium IV 3GHz with 2GB memory and running Microsoft WinXP Pro

As expected, CK-Means outperformed UK-Medoids on all datasets, which is explained by a difference between the computational complexities of the two algorithms. Indeed, both the phases of pre-computing and algorithm execution are quadratic (resp. linear) with the number of objects in the dataset for UK-Medoids (resp. CK-Means). However, it should be emphasized that the CK-Means algorithm is less general than the other methods, as it works only if the mean squared error for the definition of the EDs is used and the distance function is based on the Euclidean norm.

We also measured separately the times of the pre-computing phases, which involve the calculation of uncertain distances (in UK-Medoids) and cluster centroids (in CK-Means). Fig. 4.2 shows that the gap between UK-Medoids and CK-Means was reduced with respect to that measured by including the total runtimes (Fig. 4.1). This result confirms that the major difference between UK-Medoids and CK-Means is given by the pre-computing phase. Thus, in case of multiple runs of the two algorithms, we can state that the performance of UK-medoids and CK-means are comparable, since the pre-computing phase has to be performed once.

5

Information-Theoretic Hierarchical Clustering of Uncertain Objects

Abstract In the contest of clustering uncertain objects, a special emphasis has been put on partitional relocation and density-based approaches, whereas hierarchical clustering has drawn less attention. In this chapter, we present a prototype link-based agglomerative hierarchical algorithm for clustering uncertain objects, called *U-AHC*. A major novelty of U-AHC lies in a well-founded information-theoretic approach to the computation of distance between cluster prototypes, which is at the basis of the definition of a new proposed prototype link-based metric. Experiments conducted on various datasets have shown that U-AHC outperforms state-of-the-art methods for clustering uncertain objects from an accuracy viewpoint while achieving efficiency comparable to density-based methods. U-AHC has achieved good results also in clustering microarray data whose probe-level uncertainty is represented according to a univariate model.

5.1 Introduction

In recent years, clustering of uncertain objects has been deeply investigated from the point of view of both partitional algorithms exploiting a relocation scheme (i.e., approaches K-Means- or K-Medoids-based) [CCKN06, S. 07, GPT08a] and density-based algorithms [KP05a, KP05b]. On the other hand, to the best of our knowledge, no hierarchical approach has been so far proposed (cf. Chapt 3).

While the proposed clustering algorithms mainly differ on the clustering strategy and the cluster model, the adopted notions of distance between uncertain objects come into two main approaches. The first approach consists in computing the distance between aggregated values extracted from the probability distributions of the uncertain objects (e.g., expected values), which can be performed with a complexity linear with S.[1] The second approach instead

[1] S may denote either the dimensionality of a discrete probability mass function, or the number of statistically independent samples employed for approximating a continuous probability density function.

involves the computation of the so-called expected distance (ED) between distributions (computed, e.g., as reported in (4.1)-(4.2)),[2] which requires to somehow compare the whole distributions and works in $\mathcal{O}(S^2)$. Both these approaches have some drawbacks in their own, as stated in, e.g., [KKPR05]. The first one, though particularly efficient, has clearly an accuracy issue, since all the information available from the distributions is collapsed into a single, representative numerical value. An opposite consideration holds for the ED-based approach: it is accurate, but inefficient. Within this view, it should be not advisable to involve the existing definitions of distance between uncertain objects into any of the classic single link (SL), complete link (CL), or average link (AL) metrics[3] (cf. Chapt. 2, Sect. 2.3), and exploit one of these classic linkage metrics into an agglomerative hierarchical clustering (AHC) scheme. Instead, any AHC algorithm for clustering uncertain objects should involve a linkage metric that is accurate, i.e., it should exploit all the information available from the probability distributions, as well as efficient, i.e., it should work in $\mathcal{O}(S)$.

Information-Theory (IT) represents a fruitful research area to devise distance measures for comparing probability distributions. IT measures typically allows for computing the distance between distributions accurately and, most of them, are able to work in linear time with respect to S. However, most of the prominent existing IT measures, such as the popular ones falling into the Ali-Silvey class [AS66], cannot be used to directly define distances for uncertain objects. Indeed, such measures may have a number of shortcomings (e.g., do not satisfy the symmetric property, do not range within a bounded interval, etc.) that limit their applicability. Most importantly, IT measures commonly require that the probability distributions being compared hold for random variables defined over a common event space (i.e., common domain region); unfortunately, the domain regions of the probability distributions associated to uncertain objects may not have wide intersections.

In this chapter, we present the first (agglomerative) hierarchical algorithm for clustering uncertain objects, called *U-AHC* [GPTG08, GPTG09b], which exploits a new prototype link-based metric, whose definition is specific to the contest of uncertain objects.

The main contributions of this chapter can be summarized as follows.

1. We present U-AHC, i.e., a prototype link-based agglomerative hierarchical algorithm for clustering uncertain objects, which brings for the first time a hierarchical approach to group uncertain objects, by exploiting a standard AHC scheme.

[2] In this chapter, the term "expected distance" (ED) is used to denote a distance computed between probability distributions, instead of a distance between a probability distribution and a deterministic vector, like in the UK-Means algorithm.

[3] The term "metric", here and in the following, is not used with the classic mathematical meaning of distance in a metric space; it instead refers to a criterion for comparing two clusters.

2. As a major novelty of U-AHC, we propose a new prototype link-based metric to choose the pair of clusters to be merged at each iteration of U-AHC. It allows for overcoming the issues due to exploiting naïve linkage metric definitions based on straightforwardly applying classic SL, CL, or AL to the context of uncertain objects. Indeed, the proposed metric allows for comparing prototypes effectively, as proved by some theoretical results, and efficiently, since it works in $\mathcal{O}(S)$. The two main ingredients of the new metric are:
   - the prototype of any given cluster, which is computed as a mixture model that summarizes the probability distributions of all the objects within that cluster;
   - a new IT distance measure for comparing prototypes, whose definition is allowed by the definition of cluster prototypes as mixture densities. The proposed IT distance is a compound measure which is established based on two different ways of comparing probability distributions that represent prototypes: *(i)* measuring the distance by involving the whole probability distributions, and *(ii)* computing the difference between the expected values of the distributions. The intuition behind this definition of distance lies in the fact that comparing two distributions by an IT distance is powerful but, in principle, not always applicable; on the contrary, expected value of distributions is always computable but represents a concise information which, in general, is not able to capture the real proximity (which also depends on the shapes) between probability distributions.
3. We provide a deep insight into the properties of the proposed compound distance measure, by demonstrating its effectiveness and soundness in comparing probability distributions of prototypes. In particular:
   - we introduce a notion of *adequacy* of computing the distance between any two probability distributions (of cluster prototypes) by means of a given IT measure. Intuitively, this notion expresses to what degree an IT distance measure is worth comparing two prototypes by involving only their distributions. Indeed, a high value of adequacy implies that the distance computation can exploit most of the information into the probability distributions of the prototypes, thus ensuring high accuracy in detecting dissimilarities;
   - based on the notion of adequacy, we show that the proposed compound distance exploits an advantageous characteristic of the cluster prototypes defined as mixture densities, i.e., the overlaps between the prototypes' domain regions are generally larger than the overlaps between the individual objects' regions. Such a characteristic is employed to theoretically prove an important result: a strong relationship holds between the main ingredients that entail our proposal, i.e., the compound distance, the definition of cluster prototype, and the prototype link-based AHC scheme. Such a result allows us to claim that the pro-

posed linkage metric is well-suited for effectively (as well as efficiently) choosing the clusters to be merged.

4. We have conducted an extensive experimental evaluation on several datasets (which include benchmark and real-world data collections) in order to assess effectiveness of U-AHC algorithm in clustering uncertain data. Compared to state-of-the-art partitional and density-based algorithms, U-AHC achieved the highest quality on all the datasets. Moreover, a study on the runtime performances has shown that U-AHC behaves as good as or better than the density-based algorithms.

5.2 Uncertain Prototype

In the following, we discuss the first part of the prototype link-based metric involved into the proposed U-AHC algorithm, i.e., the definition of *uncertain prototype*, or simply *prototype*, as a new uncertain object computed to properly summarize the features of all the uncertain objects in a given set. The second part, i.e., the measure for comparing uncertain prototypes, is described in the next section.

Since uncertain objects are represented by probability distributions, it is reasonable to represent an uncertain prototype as a *finite mixture*, whose components are the pdfs associated to the objects within the set to be summarized.

Finite mixtures have long been used to model various phenomena in which more (independent) variables sum to a whole, being this characterized by the fraction of each variable (component) [MP00]. Applications have arisen in disjoint scientific disciplines and led to the development of several variants and extensions for special cases. As concerns data clustering, finite mixture models provide a foundation for probabilistic clustering (e.g., mixtures are often used to model the form of cluster members) [JD88]. Also, defining uncertain prototypes as mixture densities is an efficient operation, since it can be carried out linearly with respect to the size of the set of objects to be summarized. Other more complex definitions (such as, e.g., those based on the probability that at least one uncertain object is located at any point $\mathbf{x} \in \Re^m$) inevitably increase the computational complexity.

Definition 5.1 (multivariate uncertain prototype). *Let* $C = \{u_1, \ldots, u_n\}$ *be a set of multivariate uncertain objects, where* $u_i = (\mathcal{R}_i, p_i)$, $\mathcal{R}_i = [a_i^{(1)}, b_i^{(1)}] \times \ldots \times [a_i^{(m)}, b_i^{(m)}]$, *for each* $i \in [1..n]$. *The* multivariate uncertain prototype *of* C *is a pair* $P_C = (\mathcal{R}_C, p_C)$, *where*

$$\mathcal{R}_C = \left[\min_{i \in [1..n]} a_i^{(1)}, \max_{i \in [1..n]} b_i^{(1)} \right] \times \cdots \times \left[\min_{i \in [1..n]} a_i^{(m)}, \max_{i \in [1..n]} b_i^{(m)} \right] \tag{5.1}$$

$$p_C(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} p_i(\mathbf{x}) \tag{5.2}$$

Definition 5.2 (univariate uncertain prototype). *Let* $C = \{u_1, \ldots, u_n\}$ *be a set of univariate uncertain objects, where* $u_i = ((\mathcal{I}_i^{(1)}, p_i^{(1)}), \ldots, (\mathcal{I}_i^{(m)}, p_i^{(m)}))$, $\mathcal{I}_i^{(j)} = [a_i^{(j)}, b_i^{(j)}]$, *for each* $j \in [1..m]$, $i \in [1..n]$. *The* univariate uncertain prototype *of* C *is a tuple* $P_C = ((\mathcal{I}_C^{(1)}, p_C^{(1)}), \ldots, (\mathcal{I}_C^{(m)}, p_C^{(m)}))$ *such that, for each* $j \in [1..m]$*:*

$$\mathcal{I}_C^{(j)} = \Big[\min_{i \in [1..n]} a_i^{(j)}, \max_{i \in [1..n]} b_i^{(j)} \Big] \tag{5.3}$$

$$p_C^{(j)}(x) = \frac{1}{n} \sum_{i=1}^{n} p_i^{(j)}(x) \tag{5.4}$$

Proposition 5.3. *Let* $C = \{u_1, \ldots, u_n\}$ *be a set of multivariate uncertain objects. The multivariate uncertain prototype* P_C *is a multivariate uncertain object.*

Proof. Being C a set of multivariate uncertain objects, to prove that $P_C = (\mathcal{R}_C, p_C)$ is a (multivariate) uncertain object, we need to demonstrate that:

1. p_C is a pdf,
2. (3.1) of Def. 3.1 holds, and
3. (3.2) of Def. 3.1 holds.

Condition (1) is true since p_C represents a mixture of pdfs of the form $\sum_{i=1}^{n} \alpha_i p_i(\mathbf{x})$, where $\alpha_i = 1/n$. As concerns condition (2), we have that:

$$\begin{aligned} &\int_{\mathbf{x} \in \Re^m \setminus \mathcal{R}_C} p_C(\mathbf{x}) \mathrm{d}\mathbf{x} = \\ &= \int_{\mathbf{x} \in \Re^m \setminus \mathcal{R}_C} \frac{1}{n} \sum_{i=1}^{n} p_i(\mathbf{x}) \mathrm{d}\mathbf{x} \;=\; \frac{1}{n} \sum_{i=1}^{n} \int_{\mathbf{x} \in \Re^m \setminus \mathcal{R}_C} p_i(\mathbf{x}) \mathrm{d}\mathbf{x} \end{aligned}$$

Since $\mathcal{R}_C \supseteq \mathcal{R}_i$, $\forall i \in [1..n]$ (cf. (5.1)), we have:

$$\begin{aligned} &\frac{1}{n} \sum_{i=1}^{n} \int_{\mathbf{x} \in \Re^m \setminus \mathcal{R}_C} p_i(\mathbf{x}) \mathrm{d}\mathbf{x} = \\ &= \frac{1}{n} \sum_{i=1}^{n} \Bigg(\int_{\mathbf{x} \in \Re^m \setminus \mathcal{R}_i} p_i(\mathbf{x}) \mathrm{d}\mathbf{x} \;-\; \int_{\mathbf{x} \in \mathcal{R}_C \setminus \mathcal{R}_i} p_i(\mathbf{x}) \mathrm{d}\mathbf{x} \Bigg) \\ &= \frac{1}{n} \sum_{i=1}^{n} (0 - 0) = 0 \end{aligned}$$

which proves condition (2).

To prove condition (3), it can be straightforwardly noted that, due to the definition of prototype region in (5.1), $\forall \mathbf{x} \in \mathcal{R}_C$, there must exist at least one p_i, $i \in [1..n]$, such that $p_i(\mathbf{x}) > 0$. Consequently, the following holds: $(1/n)\sum_{i=1}^{n} p_i(\mathbf{x}) > 0,\ \forall \mathbf{x} \in \mathcal{R}_C$, i.e.,

$$p_C(\mathbf{x}) > 0,\ \forall \mathbf{x} \in \mathcal{R}_C \tag{5.5}$$

which proves condition (3). □

A result similar to that stated in Prop. 5.3 can be shown for the univariate case, i.e., it can be proved that a univariate uncertain prototype is a univariate uncertain object. We omit the details of the proof, since it is similar to that exploited for proving Prop. 5.3. Also, we point out that this observation holds in general: all the proofs concerning the univariate case may have reference to the corresponding ones concerning the multivariate case. To this purpose, we hereinafter report the proofs only for the multivariate case.

The following propositions are introduced to describe what is the form of a multivariate/univariate uncertain prototype for a new cluster resulting from the union of any two given clusters. Note that the proof of each of these propositions is trivial since they can be straightforwardly derived from Defs. 5.1-5.2.

Proposition 5.4. *Given two sets C_k, $C_{k'}$ of m-dimensional multivariate uncertain objects, such that $C_k \cap C_{k'} = \emptyset$, and the corresponding prototypes $P_{C_k} = (\mathcal{R}_{C_k}, p_{C_k})$, $P_{C_{k'}} = (\mathcal{R}_{C_{k'}}, p_{C_{k'}})$, let $\overline{C} = C_k \cup C_{k'}$ be the set composed by the objects in C_k and $C_{k'}$, and $P_{\overline{C}} = (\mathcal{R}_{\overline{C}}, p_{\overline{C}})$ the resulting prototype. It holds that:*

$$\mathcal{R}_{\overline{C}} = \Big[\min_{\hat{k}\in\{k,k'\}} a_{\hat{k}}^{(1)}, \max_{\hat{k}\in\{k,k'\}} b_{\hat{k}}^{(1)}\Big] \times \cdots \times \Big[\min_{\hat{k}\in\{k,k'\}} a_{\hat{k}}^{(m)}, \max_{\hat{k}\in\{k,k'\}} b_{\hat{k}}^{(m)}\Big] \tag{5.6}$$

$$p_{\overline{C}} = \frac{|C_k|}{|\overline{C}|} p_{C_k} + \frac{|C_{k'}|}{|\overline{C}|} p_{C_{k'}} \tag{5.7}$$

Proposition 5.5. *Given two sets C_k, $C_{k'}$ of m-dimensional univariate uncertain objects, such that $C_k \cap C_{k'} = \emptyset$, and the corresponding prototypes $P_{C_k} = ((\mathcal{I}_{C_k}^{(1)}, p_{C_k}^{(1)}), \ldots, (\mathcal{I}_{C_k}^{(m)}, p_{C_k}^{(m)}))$, $P_{C_{k'}} = ((\mathcal{I}_{C_{k'}}^{(1)}, p_{C_{k'}}^{(1)}), \ldots, (\mathcal{I}_{C_{k'}}^{(m)}, p_{C_{k'}}^{(m)}))$, let $\overline{C} = C_k \cup C_{k'}$ be the set composed by the objects in C_k and $C_{k'}$, and $P_{\overline{C}} = ((\mathcal{I}_{\overline{C}}^{(1)}, p_{\overline{C}}^{(1)}), \ldots, (\mathcal{I}_{\overline{C}}^{(m)}, p_{\overline{C}}^{(m)}))$ the resulting prototype. For each $j \in [1..m]$, it holds that:*

$$\mathcal{I}_{\overline{C}}^{(j)} = \Big[\min_{\hat{k}\in\{k,k'\}} a_{\hat{k}}^{(j)}, \max_{\hat{k}\in\{k,k'\}} b_{\hat{k}}^{(j)}\Big] \tag{5.8}$$

$$p_{\overline{C}}^{(j)} = \frac{|C_k|}{|\overline{C}|} p_{C_k}^{(j)} + \frac{|C_{k'}|}{|\overline{C}|} p_{C_{k'}}^{(j)} \tag{5.9}$$

5.3 Comparing Uncertain Prototypes

5.3.1 Distance Measures for pdfs

Probability density functions are usually compared by using *information-theoretic* (IT) measures, such as those falling into the Ali-Silvey class of distance functions [AS66]. These functions have been widely used in several application contexts, such as signal processing, pattern recognition, and speech recognition [AJ56, Bas89].

Two of the most frequently used distance measures for probability densities are the *Kullback-Leibler* (KL) divergence [KL51, Kul59] and the *Chernoff* distance [Che52]. From a similarity viewpoint, a very useful notion is represented by the *Bhattacharyya coefficient* [Bha43, Kai67]. Given any two continuous pdfs g_1 and g_2, the Bhattacharyya coefficient (ρ) is defined as:

$$\rho(g_1, g_2) = \int_{\mathbf{x} \in \Re^m} \sqrt{g_1(\mathbf{x})\ g_2(\mathbf{x})}\ \mathrm{d}\mathbf{x} \tag{5.10}$$

The original "discrete" definition of ρ in [Bha43] considers g_1 and g_2 as multinomial populations, each one consisting of K classes with associated probabilities. In any case, ρ has a relevant geometric interpretation: it can be seen as the cosine between the two vectors for g_1 and g_2, whose components are the square root of the probabilities of the K classes that compose g_1 and g_2. This interpretation also holds in the extended definition reported in (5.10), which defines Bhattacharyya coefficient for continuous pdfs.

Based on the Bhattacharyya coefficient, various distance functions can in principle be defined [Kai67]. In particular, the following measure

$$\mathrm{B}(g_1, g_2) = \sqrt{1 - \rho(g_1, g_2)} \tag{5.11}$$

has a number of advantages with respect to other Bhattacharyya distances, such as the commonly used definition of the form $-\log \rho$, and other information-theoretic measures, including as Kullback-Leibler or Chernoff. Unlike all the other mentioned measures, B ranges within the interval $[0, 1]$, which makes it particularly suitable to be combined with measures that capture other aspects when comparing two pdfs (cf. next subsection). Also, unlike the Chernoff distance (which is a more general case), B is easier and less expensive to compute and satisfies the additive property for probability distributions even though the random variables are not identically distributed. Such a property states that the distance between two joint distributions of statistically independent random variables equals the sum of the marginal distances. Finally, B is symmetric (unlike Kullback-Leibler) and satisfies the triangle inequality (unlike $-\log \rho$, Kullback-Leibler, and Chernoff).

5.3.2 Combining Information-Theoretic Notions and Expected Value on pdfs

Using an IT proximity measure represents a natural solution for devising a notion of distance between uncertain prototypes; in particular, this choice is essential to establish a function that is able to compare two pdfs by exploiting the whole information stored in the pdfs. Also, IT measures are typically fast to compute (linear with respect to S). However, this holds provided that the comparison makes sense: indeed, it should be taken into account that IT measures work out for pdfs that share a common event space (domain region). By contrast, if the two pdfs do not have any intersection in their event spaces (i.e., there is no common region in which both pdfs are greater than zero), the distance according to any IT measure is always equal to the maximum value possible (e.g., one for B, ∞ for KL).

We introduce a notion, called *IT-adequacy*, which quantifies to what degree an information-theoretic distance measure is worth comparing two uncertain prototypes by involving only their pdfs.

Definition 5.6 (IT-adequacy). *Let g_1 and g_2 be two m-dimensional pdfs ($m \geq 1$), and $\mathcal{R}_1 \subseteq \Re^m$, $\mathcal{R}_2 \subseteq \Re^m$ be two m-dimensional regions such that (for $i \in \{1, 2\}$):*

$$\int_{\mathbf{x} \in \Re^m \setminus \mathcal{R}_i} g_i(\mathbf{x}) \mathrm{d}\mathbf{x} = 0 \quad \textit{and} \quad g_i(\mathbf{x}) > 0 \ \ , \ \forall \mathbf{x} \in \mathcal{R}_i$$

The IT-adequacy *between g_1 and g_2 with respect to $\mathcal{R}_1$ and $\mathcal{R}_2$ is defined as:*

$$\Upsilon_{\mathcal{R}_1, \mathcal{R}_2}(g_1, g_2) = \frac{1}{2} \Bigg(\int_{\mathbf{x} \in \mathcal{R}_1 \cap \mathcal{R}_2} g_1(\mathbf{x}) \ \mathrm{d}\mathbf{x} \ + \int_{\mathbf{x} \in \mathcal{R}_1 \cap \mathcal{R}_2} g_2(\mathbf{x}) \ \mathrm{d}\mathbf{x} \Bigg) \tag{5.12}$$

$\Upsilon_{\mathcal{R}_1, \mathcal{R}_2}(g_1, g_2)$ (which ranges within $[0, 1]$) expresses the adequacy of computing the distance between g_1 and g_2 by using a certain IT measure. In particular, the higher $\Upsilon_{\mathcal{R}_1, \mathcal{R}_2}(g_1, g_2)$, the more the information coming from g_1 and g_2 that is exploited in the comparison.

For the sake of simplicity of notation, we will use the symbols Υ (resp. $\Upsilon^{(j)}$) to denote the IT-adequacy relative to the comparison of any two multivariate (resp. univariate) uncertain prototypes. Formally:

$$\Upsilon(P_k, P_{k'}) = \Upsilon_{\mathcal{R}_k, \mathcal{R}_{k'}}(p_k, p_{k'}) \tag{5.13}$$

for any two multivariate uncertain prototypes $P_k = (\mathcal{R}_k, p_k)$, $P_{k'} = (\mathcal{R}_{k'}, p_{k'})$, and

$$\Upsilon^{(j)}(P_k, P_{k'}) = \Upsilon_{\mathcal{I}_k^{(j)}, \mathcal{I}_{k'}^{(j)}}(p_k^{(j)}, p_{k'}^{(j)}) \tag{5.14}$$

for any two univariate uncertain prototypes $P_k = ((\mathcal{I}_k^{(1)}, p_k^{(1)}), \ldots, (\mathcal{I}_k^{(m)}, p_k^{(m)}))$, $P_{k'} = ((\mathcal{I}_{k'}^{(1)}, p_{k'}^{(1)}), \ldots, (\mathcal{I}_{k'}^{(m)}, p_{k'}^{(m)}))$.

It should be emphasized that a poor IT-adequacy may be computed when the pdfs of uncertain prototypes being compared have small (or empty) overlapping areas. To address such cases, it may be advisable to express the proximity between pdfs by resorting to the difference of their expected values. Within this view, the main intuition underlying our notion of distance measure between uncertain prototypes is to suitably combine an IT measure (which in principle is not always applicable) with a concise (but always available) information given by the expected values. Formally, we propose a distance measure Θ for uncertain prototypes P_k and $P_{k'}$ which is expressed as a function of two different terms:

$$\Theta(P_k, P_{k'}) = \mathrm{f}(\Theta_{IT}(P_k, P_{k'}), \Theta_{ED}(P_k, P_{k'})) \tag{5.15}$$

where Θ_{IT} involves a comparison by means of a certain IT measure, and Θ_{ED} measures the distance proportionally to the difference of the expected values.

5.3.3 Distance Measure for Uncertain Prototypes

We now provide the definitions of the distance measures for comparing uncertain prototypes involved into the proposed prototype link-based metric. More precisely, according to (5.15), we specify the choices for *(i)* the IT measure used for computing Θ_{IT}, and *(ii)* the way of combining Θ_{IT} and Θ_{ED}.

As regards the first point, we chose the Bhattacharyya distance (B), as defined in (5.11). In Sect. 5.3.1, we have already given motivations for which this measure has been preferred to other IT measures (such as, e.g., $-\log\rho$, Kullback-Leibler or Chernoff). We point out that B ranges within $[0, 1]$, which makes this measure easily comparable and combinable with other distance functions, which represents a major focus on this work. A further notable remark is that B can be proved to be strictly related to Υ (Def. 5.6). Indeed, B is based on ρ (the Bhattacharyya coefficient), for which the following nice property holds.

Proposition 5.7. *Let g_1 and g_2 be two m-dimensional pdfs ($m \geq 1$), and $\mathcal{R}_1 \subseteq \Re^m$, $\mathcal{R}_2 \subseteq \Re^m$ be two m-dimensional regions such that (for $i \in \{1, 2\}$):*

$$\int\limits_{\mathbf{x}\in\Re^m\setminus\mathcal{R}_i} g_i(\mathbf{x})\mathrm{d}\mathbf{x} = 0, \quad \textit{and} \quad g_i(\mathbf{x}) > 0,\ \forall \mathbf{x} \in \mathcal{R}_i$$

It holds that:

$$\rho(g_1, g_2) \leq \Upsilon_{\mathcal{R}_1,\mathcal{R}_2}(g_1, g_2)$$

Proof. Since g_1 and g_2 have a limited region of definition (i.e., $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively), the following statement holds:

$$\rho(g_1, g_2) = \int\limits_{\mathbf{x}\in\Re^m} \sqrt{g_1(\mathbf{x})\ g_2(\mathbf{x})}\ \mathrm{d}\mathbf{x} = \int\limits_{\mathbf{x}\in\mathcal{R}_1\cap\mathcal{R}_2} \sqrt{g_1(\mathbf{x})\ g_2(\mathbf{x})}\ \mathrm{d}\mathbf{x}$$

Hence, to prove that $\rho(g_1, g_2) \leq \Upsilon_{\mathcal{R}_1,\mathcal{R}_2}(g_1, g_2)$, it is sufficient to demonstrate that:

$$\int_{\mathbf{x}\in\mathcal{R}_1\cap\mathcal{R}_2} \sqrt{g_1(\mathbf{x})\ g_2(\mathbf{x})}\ \mathrm{d}\mathbf{x} \leq \int_{\mathbf{x}\in\mathcal{R}_1\cap\mathcal{R}_2} \frac{1}{2}\Big(g_1(\mathbf{x}) + g_2(\mathbf{x})\Big)\ \mathrm{d}\mathbf{x}$$

which can be straightforwardly proved to be a true statement, since the geometric mean is never greater than the arithmetic mean. □

Prop. 5.7 shows that the upper bound of the computation of ρ for any two given pdfs is equal to the IT-adequacy between those pdfs. This represents a key aspect in our framework since it supports the theoretical validity for combining Θ_{IT} and Θ_{ED}, as we shall show in a while.

Once established the IT distance measure to be used in Θ_{IT}, we have to define a way to properly combine Θ_{IT} and Θ_{ED}. A first way is to introduce a weighting factor ranging within $[0, 1]$ to control the contribution due to B with respect to the difference between the expected values, and define Θ as a linear combination of Θ_{IT} and Θ_{ED}. This weighting factor can be defined proportionally to the width of the domain region shared between the pdfs of the uncertain objects being compared.

The way of combining Θ_{IT} and Θ_{ED} described above may represent a reasonable choice to a certain extent, since the larger is the portion of the pdfs involved into the Bhattacharyya distance computation, the smaller is the need for comparing the pdfs by also considering the corresponding expected values. However, it may happen the case in which two pdfs share a large domain region but have most of their values (main portions of their areas) over different intervals.

Within this view, a more robust way to weight the contribution due to Θ might be based on the degree of overlap of the pdf areas. As previously stated, this property is at the basis of the notion of IT-adequacy, which also represents an upper bound for ρ. For this purpose, we define Θ as a linear combination of Θ_{IT} and Θ_{ED}:

$$\Theta = 1 - ((1 - \Theta_{IT}) + \beta\ (1 - \Theta_{ED}))$$

which is controlled by a factor β that is proportional to the IT-adequacy of the pdfs of the objects to be compared. Indeed, since $\Theta_{IT} = \mathrm{B} = \sqrt{1-\rho}$ and $\rho \leq \Upsilon$ (cf. Prop. 5.7), it holds that:

$$1 - \Theta_{IT} \leq 1 - \sqrt{1 - \Upsilon}$$

Thus, we can define factor β as:

$$\beta = 1 - (1 - \sqrt{1 - \Upsilon})$$

which leads to the following formula for the overall Θ

$$\begin{aligned}\Theta &= 1 - ((1 - \Theta_{IT}) + 1 - (1 - \sqrt{1 - \Upsilon})\ (1 - \Theta_{ED})) = \\ &= \Theta_{IT} - \sqrt{1 - \Upsilon}\ (1 - \Theta_{ED})\end{aligned}$$

and to the following formal definitions.

Definition 5.8 (multivariate uncertain distance). *The* multivariate uncertain distance *between two multivariate uncertain prototypes* $P_k = (\mathcal{R}_k, p_k)$ *and* $P_{k'} = (\mathcal{R}_{k'}, p_{k'})$ *is defined as*

$$\Theta(P_k, P_{k'}) = \mathrm{B}(p_k, p_{k'}) - \sqrt{1 - \Upsilon(P_k, P_{k'})} \times \mathrm{e}^{-f(E[p_k], E[p_{k'}])} \tag{5.16}$$

In (5.16), $f : \Re^m \rightarrow \Re_0^+$ is a function that measures the distance between m-dimensional points (e.g., Euclidean norm), and $E[p]$ denotes the expected value of the pdf p. Moreover, note that Θ_{ED} is defined as equal to $1 - \mathrm{e}^{-f(E[p_k], E[p_{k'}])}$. The exponential function is used to make Θ_{ED} ranging within $[0, 1]$, and, therefore, comparable to Θ_{IT} (which also ranges within $[0, 1]$).

Definition 5.9 (univariate uncertain distance). *The* univariate uncertain distance *between two univariate uncertain prototypes* $P_k = ((\mathcal{I}_k^{(1)}, p_k^{(1)}), \ldots, (\mathcal{I}_k^{(m)}, p_k^{(m)}))$ *and* $P_{k'} = ((\mathcal{I}_{k'}^{(1)}, p_{k'}^{(1)}), \ldots, (\mathcal{I}_{k'}^{(m)}, p_{k'}^{(m)}))$ *is defined as*

$$\Theta(P_k, P_{k'}) = f'(\theta^{(1)}, \ldots, \theta^{(m)}) \tag{5.17}$$

where

$$\theta^{(j)} = \mathrm{B}(p_k^{(j)}, p_{k'}^{(j)}) - \sqrt{1 - \Upsilon^{(j)}(P_k, P_{k'})}\ \mathrm{e}^{-|E[p_k^{(j)}] - E[p_{k'}^{(j)}]|}$$

for each $j \in [1..m]$.

In (5.17), $f' : \Re^m \rightarrow \Re_0^+$ is any function that computes a scalar value from the components of an m-dimensional vector. In particular, we define $f'(\theta^{(1)}, \ldots, \theta^{(m)}) = (1/m)\sqrt{\sum_{j=1}^{m} (\theta^{(j)})^2}$.

Proposition 5.10. *Given any two uncertain prototypes* $P_k = (\mathcal{R}_k, p_k)$ *and* $P_{k'} = (\mathcal{R}_{k'}, p_{k'})$, $\Theta(P_k, P_{k'})$ *assumes values within* $[0, 1]$.

Proof.

$$\begin{aligned}
&\Theta(P_k, P_{k'}) = \mathrm{B}(p_k, p_{k'}) - \sqrt{1 - \Upsilon(P_k, P_{k'})} \times \mathrm{e}^{-f(E[p_k], E[p_{k'}])} = \\
&= \sqrt{1 - \rho(p_k, p_{k'})} - \sqrt{1 - \Upsilon(P_k, P_{k'})} \times \mathrm{e}^{-f(E[p_k], E[p_{k'}])} \\
&= 1 - [\Theta'_{kk'} + \Theta''_{kk'}]
\end{aligned}$$

where

$$\Theta'_{kk'} = 1 - \sqrt{1 - \rho(p_k, p_{k'})} \quad \text{and}$$

$$\Theta''_{kk'} = \left(1 - \left(1 - \sqrt{1 - \Upsilon(P_k, P_{k'})}\right)\right) \times \mathrm{e}^{-f(E[p_k], E[p_{k'}])}$$

Since $\rho(p_k, p_{k'}) \leq \Upsilon(P_k, P_{k'})$ (Prop. 5.7), we have that $\Theta'_{kk'} \leq 1-\sqrt{1-\Upsilon(P_k, P_{k'})}$, consequently:

$$\begin{aligned}
&\min \Theta(P_k, P_{k'}) = \\
&= 1 - [\max \Theta'_{kk'} + \max \Theta''_{kk'}] \\
&= 1 - \left[1 - \sqrt{1-\Upsilon(P_k, P_{k'})} + \right. \\
&\left. + \left(1 - \left(1 - \sqrt{1-\Upsilon(P_k, P_{k'})}\right)\right)\right] = 0
\end{aligned}$$

Moreover, since $\Theta'_{kk'}$ and $\Theta''_{kk'}$ are minimum if and only if $\rho(p_k, p_{k'})$ and $\mathrm{e}^{-f(E[p_k],E[p_{k'}])}$ are individually minimum (i.e., $\rho(p_k, p_{k'}) = 0$ and $f(E[p_k], E[p_{k'}]) \rightarrow +\infty$), then:

$$\begin{aligned}
&\max \Theta(P_k, P_{k'}) = \\
&= 1 - [\min \Theta'_{kk'} + \min \Theta''_{kk'}] \\
&= 1 - \left[0 + \sqrt{1-\Upsilon(P_k, P_{k'})} \times \mathrm{e}^{-\infty}\right] = 1
\end{aligned}$$

□

We now give an insight into the behavior of the proposed distance function Θ, when the two terms Θ_{IT} and Θ_{ED} are close to their extreme values, i.e., they are close to either zero or one. We show a number of properties only for the multivariate definition of Θ, since it is easy to prove that they also hold for each $\theta^{(j)}$ term ($j \in [1..m]$) of the univariate version of Θ.

1. Since $\mathrm{B}(p_k, p_{k'}) = 0$ if and only if $p_k = p_{k'}$, i.e., $P_k = P_{k'}$, and $\mathrm{B}(p_k, p_{k'}) = 0$ implies that $\Upsilon(P_k, P_{k'}) = 1$, we have:

$$\lim_{\mathrm{B}(p_k,p_{k'})\rightarrow 0} \Theta(P_k, P_{k'}) = 0 \tag{5.18}$$

   Indeed, if $P_k = P_{k'}$ then $\Theta(P_k, P_{k'})$ is equal to zero.
2. Let us denote $\Theta^{(2)} = 1 - \sqrt{1-\Upsilon(P_k, P_{k'})} \times \mathrm{e}^{-f(E[p_k],E[p_{k'}])}$. It is easy to see that:

$$\lim_{\mathrm{B}(p_k,p_{k'})\rightarrow 1} \Theta(P_k, P_{k'}) = \Theta^{(2)} \tag{5.19}$$

   To prove the soundness of this property, it should be noted that:

$$\lim_{\Upsilon(P_k,P_{k'})\rightarrow 0} \Theta^{(2)} = 1 - \mathrm{e}^{-f(E[p_k],E[p_{k'}])}$$

   i.e., if the IT-adequacy between P_k and P_k is low, the only information exploited for computing Θ is the distance between $E[p_k]$ and $E[p_{k'}]$, which is clearly reasonable. Moreover:

$$\lim_{\Upsilon(P_k,P_{k'})\rightarrow 1} \Theta^{(2)} = 1$$

which is intuitively correct since, if $\Upsilon(P_k, P_{k'})$ is high, Θ would tend to the distance value computed by B, which tends to one in this case.

3. Let $\Theta^{(3)} = \mathrm{B}(p_k, p_{k'}) - \sqrt{1 - \Upsilon(P_k, P_{k'})}$. It holds that:

$$\lim_{f(E[p_k], E[p_{k'}]) \to 0} \Theta(P_k, P_{k'}) = \Theta^{(3)} \tag{5.20}$$

This property holds due to the following additional considerations.

$$\lim_{\Upsilon(P_k, P_{k'}) \to 0} \Theta^{(3)} = 0$$

Indeed, if P_k and $P_{k'}$ have low IT-adequacy, Θ would take into account only $f(E[p_k], E[p_{k'}])$, which tends to 0 in this case. Also:

$$\lim_{\Upsilon(P_k, P_{k'}) \to 1} \Delta^{(3)} = \mathrm{B}(p_k, p_{k'})$$

In this case, since the IT-adequacy is high, it is correct to focus on the shapes of the pdfs, i.e., to compute Θ by only considering the term $\mathrm{B}(p_k, p_{k'})$;

4. Finally:

$$\lim_{f(E[p_k], E[p_{k'}]) \to +\infty} \Theta(P_k, P_{k'}) = \mathrm{B}(p_k, p_{k'}) \tag{5.21}$$

which holds since P_k and $P_{k'}$ may be similar to a certain degree (in a way inversely proportional to $\mathrm{B}(p_k, p_{k'})$) even if the expected values of the corresponding pdfs p_k and $p_{k'}$ are dissimilar. This highlights as Θ allows for overcoming limitations of the distance between expected values and further supports for the robustness of Θ.

5.4 U-AHC Algorithm

We present here the proposed AHC-based algorithm for clustering uncertain objects, named *U-AHC*. The outline of U-AHC is given in Alg. 5.1.

The input for U-AHC algorithm is a dataset D of n uncertain objects and a positive integer S, which denotes the number of pdf samples used for integral estimations; the output is a hierarchy $\mathbf{T}$ of clusters (a dendrogram). The algorithm follows the classic AHC scheme. The proposed implementation makes use of a priority queue ($\mathbf{Q}$) to store the inter-cluster distances in order to improve the computational efficiency—the lower the distance between a pair of clusters, the higher the corresponding priority in $\mathbf{Q}$.

The initialization steps (Lines 1-11) are in charge of computing the list of samples for each object (procedure *object_sampling*) and the initial set of clusters composed by n singletons along with the corresponding prototypes (Lines 2-6). The initial pair-wise distances are computed by the *prototype_distance* procedure (Funct. 5.1) and inserted into the priority queue (Lines 8-11). Note

Algorithm 5.1 U-AHC

Input: a set $D = \{u_1, \ldots, u_n\}$ of uncertain objects, the number S of samples used for integral estimations
Output: a set of partitions $\mathbf{T}$ (i.e., a dendrogram)

```
    {initialization}
 1: C ← ∅, Q ← ∅
 2: for all u ∈ D do
 3:    object_sampling(u, S)
 4:    C ← {u}, P_C ← u
 5:    C ← C ∪ {C}
 6: end for
 7: T ← {C}
 8: for all C_k, C_k' ∈ C, C_k ≠ C_k' do
 9:    d ← prototype_distance(C_k, C_k', P_{C_k}, P_{C_k'})
10:    Q.insert(C_k, C_k', d)
11: end for
    {main loop}
12: repeat
13:    (C_k, C_k') ← Q.removeMin()
14:    C̄ ← C_k ∪ C_k'
15:    P_C̄ ← compute_prototype(C_k, C_k', P_{C_k}, P_{C_k'})        {(5.6)–(5.9)}
16:    for all C ∈ C, C ≠ C_k, C ≠ C_k' do
17:       Q.remove(C, C_k)
18:       Q.remove(C, C_k')
19:       d ← prototype_distance(C, C̄, P_C, P_C̄)
20:       Q.insert(C, C̄, d)
21:    end for
22:    C ← C \ {C_k, C_k'} ∪ {C̄}
23:    T ← T ∪ {C}
24: until |C| = 1
```

that, in order to estimate the integrals numerically, each uncertain object (including the uncertain prototypes) is associated with its corresponding list of samples during the whole execution of the algorithm (procedure *get_samples*).

The main loop of the algorithm (Lines 12-24) is repeated until the whole hierarchy has been built, i.e., the number of clusters in the current clustering $\mathcal{C}$ is equal to one. At each iteration, the pair of the two clusters having the minimum distance is extracted from the priority queue (Line 13) and the two clusters are merged to form a new cluster $\overline{C}$ (Line 14). Afterward, the procedure *compute_prototype* (Line 15) updates the prototype of $\overline{C}$ by computing the new region/intervals of definition according to (5.6) and (5.8), pdfs according to (5.7) and (5.9), and samples. More precisely, computing the new list of samples is accomplished by involving the *Markov Chain Monte Carlo* (MCMC) sampling method [W. 70], which is suitable for mixture densities. Once the merging step has been completed, the priority queue is updated (Lines 16-21). For each cluster C in the current clustering, the distances be-

tween C and the earlier clusters $C_k, C_{k'}$ are removed from $\mathbf{Q}$ (Lines 17-18) and replaced with the distance from the new cluster $\overline{C}$ (Lines 19-20). We assume that any removal operation $\mathbf{Q}.remove(C, C')$ may implicitly perform also $\mathbf{Q}.remove(C', C)$.

Function 5.1 prototype_distance

Parameters: a pair C, C' of clusters and their respective prototypes $P_C, P_{C'}$
Returned data: a real value d within $[0, 1]$

1: $U_B \leftarrow$ compute_ub_values$(C, C', P_C, P_{C'})$ {*(5.12)* }
2: $Pr \leftarrow \emptyset, \mathcal{S} \leftarrow \emptyset$
3: **if** $|C| < |C'|$ **then**
4: $\mathcal{S} \leftarrow$ get_samples(P_C)
5: **for all** $S' \in \mathcal{S}$ **do**
6: $Pr \leftarrow Pr \cup$ compute_probability(C', S')
7: **end for**
8: **else**
9: $\mathcal{S} \leftarrow$ get_samples$(P_{C'})$
10: **for all** $S' \in \mathcal{S}$ **do**
11: $Pr \leftarrow Pr \cup$ compute_probability(C, S')
12: **end for**
13: **end if**
14: $D_{IT} \leftarrow$ compute_B_values$(P_C, P_{C'}, \mathcal{S}, Pr)$ {*(5.11)*}
15: $D_{ED} \leftarrow$ compute_ED_values$(P_C, P_{C'})$ {*(5.16)–(5.17)*}
16: $d \leftarrow$ uncertain_distance(D_{IT}, D_{ED}, U_B) {*(5.16)–(5.17)*}
17: **return** d

The *prototype_distance* function (Funct. 5.1) computes a distance value d between the prototypes of any two given clusters C and C' according to either Def. 5.8 or Def. 5.9. This function consists of three main phases. First, a set U_B is computed to contain the upper-bound values of the Bhattacharyya coefficient (i.e., the IT-adequacy values) between the pdfs of the prototypes of C and C' (Line 1); these values are calculated according to (5.12), and involve the cumulative distribution functions (cdfs) of the objects within the clusters C and C'. Afterward, the *compute_B_values* procedure computes the set D_{IT} of Bhattacharyya distance values according to (5.11) (Line 14); this procedure estimates the integrals needed for deriving the Bhattacharyya coefficient by taking into account the set of samples $\mathcal{S}$ and the set Pr of "new" probabilities (Lines 3-13). The sets Pr and $\mathcal{S}$ are created as follows: the set $\mathcal{S}$ is built up with the samples of the prototype of C (resp. prototype of C') if $|C|$ is smaller than (resp. greater than or equal to) $|C'|$, and Pr contains the probabilities of each sample in $\mathcal{S}$ computed according to the pdfs of the objects in C' (resp. C). In the third phase, the *compute_ED_values* procedure computes the set D_{ED} of distances based on the expected values; in particular, D_{ED} is populated by computing the exponential of the negative of the distances between the expected values of the prototypes (Line 15). Finally, the *uncertain_distance*

procedure computes the final distance value d according to either (5.16) (multivariate case) or (5.17) (univariate case), by considering the values in the sets D_{IT}, D_{ED} and U_B (Line 16). Note that $|U_B| = |D_{IT}| = |D_{ED}| = 1$, in the multivariate case, whereas $|U_B| = |D_{IT}| = |D_{ED}| = m$, in the univariate case.

Computational Aspects

Let us now discuss the computational complexity of Alg. 5.1, given a dataset D of n uncertain objects and a number S of samples. As previously discussed, Alg. 5.1 uses the *prototype_distance* function (Funct. 5.1), whose complexity can be trivially proved to be $\mathcal{O}(S \ \min\{|C|, |C'|\})$. Furthermore, we assume that *(i)* the operations of insertion/deletion/extraction of any object into/from the priority queue $\mathbf{Q}$ are performed in $\mathcal{O}(\log |\mathbf{Q}|)$, and *(ii)* the number m of dimensions of the uncertain objects is a constant. The costs of the various steps of Alg. 5.1 are summarized as follows:

- the initialization steps, i.e., computing the lists of samples for each uncertain object and the set of initial clusters (Lines 2-6), and initializing the priority queue (Lines 8-11), are executed in $\mathcal{O}(S \ n)$ and $\mathcal{O}(n^2 \ \log n)$, respectively;
- the main loop (Lines 12-24) is repeated $n - 1$ times; therefore, each step of this loop has the following total cost:
  - extracting from $\mathbf{Q}$ the pairs of clusters having the minimum distance (Line 13) is $\mathcal{O}(n \ \log n)$;
  - merging the clusters $C_k, C_{k'}$ (Line 14) is $\mathcal{O}(\sum_{i=1}^{n-1} \max_{\hat{C} \in \mathcal{C}^{(r)}} |\hat{C}|) = \mathcal{O}(\sum_{i=1}^{n-1} i) = \mathcal{O}(n^2)$, where $\mathcal{C}^{(r)}$ is the set of clusters computed at the r-th iteration;
  - in the *compute_prototype* function, the most expensive operation is resampling from the new mixture density (i.e., the new prototype), which is $\mathcal{O}(S \ (|C_k| + |C_{k'}|))$. Therefore, computing a new prototype for all the iterations of the algorithm (Line 15) is $\mathcal{O}(\sum_{i=1}^{n-1} S \max_{\hat{C} \in \mathcal{C}^{(r)}} |\hat{C}|) = \mathcal{O}(S \sum_{i=1}^{n-1} i) = \mathcal{O}(S \ n^2)$;
  - in the internal loop (Lines 16-21), the operations of insertion/deletion into/from the priority queue (Lines 17, 18, and 20) are performed in $\mathcal{O}(n^2 \ \log n)$. The prototype distance (Line 19) is computed by taking into account the list of samples of one of the two clusters, which has a cost $\mathcal{O}(\sum_{i=1}^{n-1} \sum_{\hat{C} \in \mathcal{C}^{(r)}} S \ |\hat{C}|) = \mathcal{O}(S \sum_{i=1}^{n-1} \sum_{\hat{C} \in \mathcal{C}^{(r)}} |\hat{C}|) = \mathcal{O}(S \ n^2)$;
- updating $\mathcal{C}$ (Line 22) and $\mathbf{T}$ (Line 23) is obviously $\mathcal{O}(n)$.

In conclusion, we can state that Alg. 5.1 works in $\mathcal{O}(n^2(S + \log n))$. For many real cases, the condition that S is $\Omega(\log n)$ typically holds; under this assumption, the computational complexity of Alg. 5.1 can be rewritten as $\mathcal{O}(S \ n^2)$.

5.5 Impact of IT-adequacy on the Behavior of U-AHC

Proposition 5.3 states that uncertain prototypes are uncertain objects. Thus, the distance measures proposed in Defs. 5.8-5.9 in principle work for any multivariate and univariate uncertain object, respectively. However, we provide in the following some theoretical results aimed at assessing that the proposed distances satisfy the accuracy requirement only if they are exploited into an AHC scheme to compare uncertain prototypes defined as mixture densities. Hence, adopting the measures in Defs. 5.8-5.9 for measuring the distance between any two uncertain objects may be inappropriate. We point out that this is not a limitation of our work, since we are not interested in defining a new distance measure between uncertain objects, but rather a measure for comparing prototypes defined as mixture densities into an AHC algorithm.

Before presenting the main results of this section, let us define a simple lemma and provide some background notations—to avoid overloading the notation, we assume that clustering and related items refer to the r-th iteration of the U-AHC algorithm, with $r \in [1..n-1]$.

Lemma 5.11. *Let $u = (\mathcal{R}, p)$ be a multivariate uncertain object. Given any $\mathcal{R}' \subset \Re^m$, it holds that:*

$$\int_{\mathcal{R} \cap \mathcal{R}'} p(\mathbf{x}) \mathrm{d}\mathbf{x} = \int_{\mathcal{R}'} p(\mathbf{x}) \mathrm{d}\mathbf{x}$$

Proof. To prove the lemma, it is easy to see that:

$$\int_{\mathcal{R} \cap \mathcal{R}'} p(\mathbf{x}) \mathrm{d}\mathbf{x} = \int_{\mathcal{R}'} p(\mathbf{x}) \mathrm{d}\mathbf{x} - \int_{\mathcal{R}' \setminus \mathcal{R}} p(\mathbf{x}) \mathrm{d}\mathbf{x} = \int_{\mathcal{R}'} p(\mathbf{x}) \mathrm{d}\mathbf{x}$$

If $\mathcal{R}' \subset \mathcal{R}$, then the above equality holds trivially, since $\mathcal{R} \cap \mathcal{R}' = \mathcal{R}'$ and p is also defined over $\mathcal{R}'$. Otherwise, if $\mathcal{R} \subset \mathcal{R} \cup \mathcal{R}'$, it holds that

$$\int_{\mathcal{R}' \setminus \mathcal{R}} p(\mathbf{x}) \mathrm{d}\mathbf{x}) = 0$$

Finally, as a special case, if $\mathcal{R}' \cap \mathcal{R} = \emptyset$ then the integral of p over the empty set is equal to zero as well as the integral of p over any external region $\mathcal{R}'$. □

- $\mathcal{P} = \{P_1, \ldots, P_{n-r+1}\}$ is a set of prototypes of the form $P_q = (\mathcal{R}_q, p_q)$, for $q \in [1..n-r+1]$, which correspond to the clustering solution $\mathcal{C} = \{C_1, \ldots, C_{n-r+1}\}$ obtained by U-AHC (at the r-th iteration);
- $k, k' \in [1..n-r+1]$ are two indices such that $C_k, C_{k'} \in \mathcal{C}$ is the pair of clusters to be merged, and
- $\overline{C} = C_k \cup C_{k'}$ is the new cluster formed;

- $\overline{P} = (\overline{\mathcal{R}}, \overline{p})$ is the prototype of $\overline{C}$.

Theorem 5.12. *Let $\alpha \in [0,1]$ be a constant, $\Psi_q(\mathcal{C}, \alpha) = \Upsilon(P_q, \overline{P}) - (\alpha\Upsilon(P_q, P_k) + (1-\alpha)\Upsilon(P_q, P_{k'}))$. Given $\hat{\alpha} = |C_k|/(|C_k| + |C_{k'}|)$, for each $q \in [1..n-r+1]$, $q \neq k$, $q \neq k'$, it holds that:*

$$\Psi_q(\mathcal{C}, \hat{\alpha}) = \frac{1}{2}\left((1-\hat{\alpha})\int_{\mathcal{R}_k} p_q(\mathbf{x})\mathrm{d}\mathbf{x} + \hat{\alpha}\int_{\mathcal{R}_{k'}} p_q(\mathbf{x})\mathrm{d}\mathbf{x}\right)$$

Proof. Let us denote with $\mathbf{I}_\phi(\Phi)$ an integral of the form $\int_\Phi p_\phi(\mathbf{x})\mathrm{d}\mathbf{x}$. By setting α equal to $\hat{\alpha}$, we have that:

$$\begin{aligned}
&\Psi_q(\mathcal{C}, \hat{\alpha}) = \\
&= \frac{1}{2}\left(\mathbf{I}_q(\mathcal{R}_q \cap \overline{\mathcal{R}}) + \overline{\mathbf{I}}(\mathcal{R}_q \cap \overline{\mathcal{R}})\right) + \\
&- \left[\frac{\hat{\alpha}}{2}\left(\mathbf{I}_q(\mathcal{R}_q \cap \mathcal{R}_k) + \mathbf{I}_k(\mathcal{R}_q \cap \mathcal{R}_k)\right) + \right. \\
&\left. + \frac{1-\hat{\alpha}}{2}\left(\mathbf{I}_q(\mathcal{R}_q \cap \mathcal{R}_{k'}) + \mathbf{I}_{k'}(\mathcal{R}_q \cap \mathcal{R}_{k'})\right)\right]
\end{aligned}$$

Since for any multivariate uncertain object $u = (\mathcal{R}, p)$ and any $\mathcal{R}' \subset \Re^m$ it holds that (Lemma 5.11):

$$\int_{\mathcal{R} \cap \mathcal{R}'} p(\mathbf{x})\mathrm{d}\mathbf{x} = \int_{\mathcal{R}'} p(\mathbf{x})\mathrm{d}\mathbf{x}$$

and $\overline{p} = \hat{\alpha}p_k + (1-\hat{\alpha})p_{k'}$ (cf. (5.7)), then:

$$\begin{aligned}
&\Psi_q(\mathcal{C}, \hat{\alpha}) = \\
&= \frac{1}{2}\left(\mathcal{I}_q(\mathcal{R}_k) + \mathbf{I}_q(\mathcal{R}_{k'}) + \hat{\alpha}\mathbf{I}_k(\mathcal{R}_q) + \right. \\
&\left. + (1-\hat{\alpha})\mathbf{I}_{k'}(\mathcal{R}_q)\right) - \left[\frac{\hat{\alpha}}{2}\left(\mathbf{I}_q(\mathcal{R}_k) + \mathbf{I}_k(\mathcal{R}_q)\right) + \right. \\
&\left. + \frac{1-\hat{\alpha}}{2}\left(\mathbf{I}_q(\mathcal{R}_{k'}) + \mathbf{I}_{k'}(\mathcal{R}_q)\right)\right] \\
&= \frac{1}{2}\left((1-\hat{\alpha})\mathbf{I}_q(\mathcal{R}_k) + \hat{\alpha}\mathbf{I}_q(\mathcal{R}_{k'})\right) \\
&= \frac{1}{2}\left((1-\hat{\alpha})\int_{\mathcal{R}_k} p_q(\mathbf{x})\mathrm{d}\mathbf{x} + \hat{\alpha}\int_{\mathcal{R}_{k'}} p_q(\mathbf{x})\mathrm{d}\mathbf{x}\right)
\end{aligned}$$

□

Corollary 5.13. *For each $q \in [1..n-k+1]$, $q \neq k$, $q \neq k'$, if $\Upsilon(P_q, P_k) \neq 0$ or $\Upsilon(P_q, P_{k'}) \neq 0$, then $\Psi(\mathcal{C}, \hat{\alpha}) > 0$; otherwise $\Psi(\mathcal{C}, \hat{\alpha}) = 0$.*

The above theorem and corollary state that the notion of IT-adequacy plays an important role when comparing the (prototype of) clusters in every iteration of the proposed U-AHC algorithm. In particular, given any two clusters $C_k, C_{k'}$ being merged and any other cluster C_q, there is a strong relation between the individual IT-adequacy of P_k and $P_{k'}$ with respect to P_q, and the IT-adequacy of the prototype of the (new) merged cluster with respect to P_q. More precisely, the IT-adequacy of the prototype of the (new) merged cluster to P_q is not lower than (the combination of) the individual IT-adequacy of P_k and $P_{k'}$ with respect to P_q. As a special case, if there is an overlap between the region of P_k (or $P_{k'}$) and P_q, the IT-adequacy resulting from the merging step will increase. Thus, the accuracy in comparing the pair of clusters in U-AHC is not decreasing (and, in many cases, strictly increasing) as the iterations of U-AHC increase.

5.6 Experimental Evaluation

The proposed U-AHC algorithm was evaluated in performing effective clustering of uncertain data. The experimental evaluation was also conducted to give a comparison of U-AHC with existing partitional relocation algorithms (i.e., UK-Means [CCKN06], CK-Means [S. 07], and UK-Medoids [GPT08a]) and density-based algorithms (i.e., $\mathcal{F}$DBSCAN [KP05a] and $\mathcal{F}$OPTICS [KP05b]).

In the next two subsections, we discuss the evaluation methodology and the main experimental results from both accuracy and efficiency viewpoints, respectively.

5.6.1 Evaluation Methodology

Datasets

Experiments were performed on benchmark datasets from UCI Machine Learning Repository [ANml]. In particular, we selected eight datasets with numerical real-valued attributes, namely Iris, Wine, Glass, Ecoli, Yeast, ImageSegmentation, Abalone, and LetterRecognition (cf. Appendix A). These datasets were originally established as collections of data with deterministic values. We synthetically generated uncertainty in these collections using pdfs of different forms (i.e., *Uniform*, *Normal* and *Gamma*), obtaining both univariate and multivariate uncertain objects. We follow the methodology described in Chapt. 4, Sect. 4.4.1.

Cluster Validity

To assess the quality of clustering solutions we exploited the availability of reference classifications for the datasets. The objective was to evaluate how

well a clustering fits a predefined scheme of known classes (natural clusters). To this purpose, we resorted to the F1-Measure external validity criterion (cf. Def. 2.5).

Settings

The integrals involved into the distances calculation were computed by taking into account lists of samples derived from the pdfs. For this purpose, we employed the classic *Monte Carlo* and *Markov Chain Monte Carlo* sampling methods.[4]

Computing the set of initial centroids and medoids (in K-Means- and K-Medoids-based methods) and computing the set of samples (for all methods) correspond to non-deterministic operations; therefore, to avoid that clustering results were biased by random chance, the accuracy and efficiency measurements for all methods were averaged over 100 different runs.

We performed a tuning phase for the parameters ϵ (i.e., the threshold for the distance of the neighbors of an object) and μ (i.e., the minimum number of points within the neighborhood of an object) required by the density-based approaches $\mathcal{F}$DBSCAN and $\mathcal{F}$OPTICS. We set these parameters to the values that allowed each method to achieve the best accuracy results.

For our U-AHC and $\mathcal{F}$OPTICS, we also needed for selecting a partition of proper size from the clustering solution obtained by each of these methods. In particular, we considered the partition obtained by cutting the dendrogram at the level corresponding to the desired number of output clusters (e.g., equal to the desired number of ideal classes for each benchmark dataset). For $\mathcal{F}$OPTICS, we initially derived a cluster hierarchy by converting the reachability plot into a dendrogram according to the automatic procedure described in [SQL$^+$03].

5.6.2 Results

Accuracy

Table 5.1 and Table 5.2 summarize the F1-Measure results obtained by U-AHC and the other methods on the various datasets for the univariate and multivariate models, respectively. In either table, the last two rows contain the average F1-Measure score obtained by each method and the average percentage of quality gain obtained by U-AHC against each other method.

A first important remark is that U-AHC achieved the highest quality on all the datasets. On average, U-AHC outperformed the other methods, with average gains that ranged from 8.2% (vs. UK-Medoids) to about 18.4% (vs. $\mathcal{F}$DBSCAN), in the univariate case, and from 10% (vs. UK-Medoids) to 18% (vs. $\mathcal{F}$OPTICS), in the multivariate case.

[4] We used the SSJ library, available at http://www.iro.umontreal.ca/∼simardr/ssj/

Table 5.1. U-AHC vs. competing methods: accuracy results (F1-Measure) for univariate models

dataset	*pdf*	UK-means	CK-means	UK-medoids	$\mathcal{F}$DBSCAN	$\mathcal{F}$OPTICS	**U-AHC**
Iris	Uniform	0.841	*0.963*	0.886	0.919	0.886	**0.993**
	Normal	0.849	0.849	0.855	0.871	*0.907*	0.905
	Gamma	0.622	0.501	0.848	0.893	*0.905*	0.628
Wine	Uniform	0.500	0.724	*0.810*	0.664	0.695	**0.984**
	Normal	0.500	0.704	0.578	0.653	*0.713*	0.954
	Gamma	0.500	0.581	0.581	0.692	*0.713*	0.595
Glass	Uniform	0.639	0.670	0.697	*0.768*	0.718	**0.828**
	Normal	*0.577*	0.552	0.513	0.514	0.438	0.822
	Gamma	0.379	0.314	*0.644*	0.468	0.438	0.550
Ecoli	Uniform	0.653	*0.795*	0.696	0.436	0.477	**0.915**
	Normal	0.609	*0.741*	0.528	0.544	0.477	0.726
	Gamma	0.533	0.412	*0.693*	0.401	0.477	0.450
Yeast	Uniform	0.497	0.562	*0.618*	0.515	0.543	**0.719**
	Normal	*0.471*	0.458	0.288	0.291	0.316	0.577
	Gamma	0.403	0.306	*0.469*	0.331	0.316	0.406
Image	Uniform	*0.810*	0.798	0.769	0.426	0.419	0.552
	Normal	0.623	*0.655*	0.451	0.416	0.419	**0.836**
	Gamma	0.545	0.353	*0.656*	0.339	0.419	0.503
Abalone	Uniform	0.331	0.294	*0.590*	0.447	0.439	**0.719**
	Normal	*0.288*	0.217	0.265	0.136	0.209	0.577
	Gamma	0.360	0.200	0.313	0.565	*0.607*	0.406
Letter	Uniform	0.529	0.629	*0.776*	0.344	0.318	**0.792**
	Normal	0.449	0.451	*0.490*	0.247	0.318	0.531
	Gamma	0.432	0.215	*0.584*	0.265	0.318	0.603
	avg. score	0.539	0.539	0.608	0.506	0.521	0.690
	avg. gain	15.1%	15.1%	8.2%	18.4%	16.9%	—

Table 5.2. U-AHC vs. competing methods: accuracy results (F1-Measure) for multivariate models

dataset	*pdf*	UK-means	CK-means	UK-medoids	$\mathcal{F}$DBSCAN	$\mathcal{F}$OPTICS	**U-AHC**
Iris	Uniform	0.948	*0.962*	0.907	0.929	0.907	**1**
	Normal	0.859	0.897	0.888	*0.929*	0.907	0.962
Wine	Uniform	0.735	0.747	0.761	*0.767*	0.713	**0.826**
	Normal	0.707	0.705	*0.749*	0.691	0.713	0.795
Glass	Uniform	0.677	*0.703*	0.653	0.575	0.636	0.779
	Normal	0.540	0.551	0.579	*0.868*	0.828	**0.891**
Ecoli	Uniform	0.787	*0.790*	0.728	0.443	0.477	0.743
	Normal	*0.745*	0.740	0.560	0.416	0.477	**0.795**
Yeast	Uniform	0.533	0.538	*0.622*	0.599	0.528	**0.684**
	Normal	0.455	*0.457*	0.318	0.374	0.420	0.486
Image	Uniform	0.780	*0.801*	0.765	0.482	0.419	**0.837**
	Normal	0.628	0.637	*0.649*	0.415	0.419	0.684
Abalone	Uniform	0.288	0.290	*0.531*	0.499	0.439	0.492
	Normal	0.215	0.217	0.288	0.497	*0.558*	**0.572**
Letter	Uniform	0.637	0.636	*0.763*	0.320	0.318	**0.798**
	Normal	0.442	0.435	*0.595*	0.353	0.318	0.613
	avg. score	0.624	0.632	0.647	0.571	0.567	0.747
	avg. gain	12.3%	11.5%	10.0%	17.6%	18.0%	—

Overall, we observe that all the clustering methods obtained their respective best performances according to the uniform distribution in nearly all datasets. Moreover, in general, the algorithm performances improved in the multivariate case.

Among the competing methods, UK-Medoids behaved better than the other ones—10 out of 24 times (in the univariate case) and 6 out of 16 times (in the multivariate case)—obtaining an average quality gain up to 10% and 8% in the univariate and multivariate cases, respectively. The partitional algorithms performed slightly better than the density-based algorithms in the univariate case, whereas this gap tended to increase in the multivariate case. Moreover, ignoring the results that refer to uniform pdfs, the density-based algorithms (especially $\mathcal{F}$OPTICS) performed as good as or better than the partitional ones. It should also be noted that $\mathcal{F}$OPTICS and $\mathcal{F}$DBSCAN behaved closely, as well as UK-Means and CK-Means. This was not surprising since the two couples of algorithms employ similar clustering schemes.

Efficiency

As regards as the efficiency evaluation, we were actually interesting in measuring the runtime performance of U-AHC in relation to the other methods. For the sake of brevity of presentation, here we present results concerning the univariate case on the four largest datasets, namely Yeast, ImageSegmentation, Abalone, and LetterRecognition; however, the performance trends observed on the rest of the datasets and for the multivariate case were roughly similar to those presented in the following.[5] Note also that we left out of consideration times for operations which are usually carried out in an off-line mode, i.e., computing the list of samples for every uncertain object (required by all algorithms), the matrix of pair-wise distances (UK-Medoids and U-AHC), the gravity centers (expected values) and the distances between each uncertain object and its gravity center (CK-Means).

Figure 5.1 shows the clustering time performances (in milliseconds) of the various methods on the selected datasets. We can see that, as we expected, CK-Means outperformed all the other methods, including UK-Means; clearly, in this case, the significant gain observed is mainly due to the optimization employed by CK-Means for the EDs calculation. On the other hand, it should be recalled that the CK-Means algorithm works only if the mean squared error for the definition of the EDs is used and the distance function is based on the Euclidean norm.

UK-Medoids was the second fastest method, even better than UK-Means; we indeed observed that UK-Medoids in general required less iterations for the convergence than UK-Means. Our U-AHC compared closely to the density-based algorithms. More precisely, $\mathcal{F}$DBSCAN performed better than U-AHC, although in times of the same order of magnitude, and U-AHC as close as or slightly better than $\mathcal{F}$OPTICS in turn.

It is also easy to see how the plots in Fig. 5.1 are very similar over the various datasets. In general, it should be noted that the clustering performances

[5] Experiments were conducted on a platform Intel Pentium IV 3GHz with 2GB memory and running Microsoft WinXP Pro.

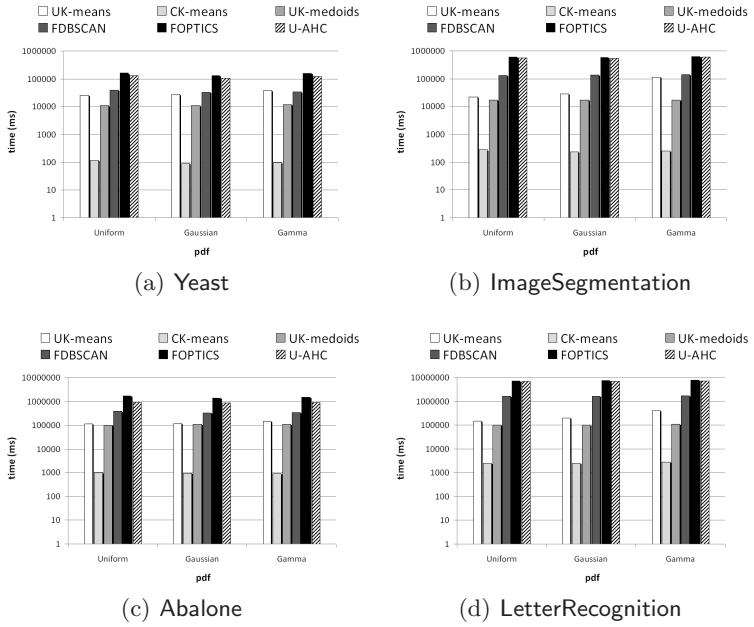


Fig. 5.1. U-AHC vs. competing methods: clustering time performances in the univariate model

followed the (on-line) computational complexities of the corresponding algorithm, namely $\mathcal{O}(I\ S\ n)$ for UK-Means, $\mathcal{O}(I\ n)$ for CK-Means, $\mathcal{O}(I\ n^2)$ for UK-Medoids—I is the number of iterations required for the algorithm convergence—$\mathcal{O}(n^2)$ for $\mathcal{F}$DBSCAN, and $\mathcal{O}(S\ n^2)$ for $\mathcal{F}$OPTICS and U-AHC.

5.7 Exploiting U-AHC for Clustering Microarray Data

5.7.1 Microarray Data and Probe-level Uncertainty

A major goal in genomics [UCA09, BC09] is to discover gene relationships and their role in diseases (*functional genomics*). *DNA microarray* [TBA+02] is a technology used in molecular biology and medicine which is able to trap and measure the relative quantity of a large number of genes with a single experiment. Probes that are able to trap genes (*targets*) consist of thousands of microscopic spots organized as a matrix and placed on a glass or silicon chip. The probes-target hybridization is quantified through techniques based on fluorescence. When an experiment is performed, the spots of the microarray matrix generate intensity values, which measure the *expression levels* of the genes. Recently, the Affymetrix company introduced an advanced chip version,

called Human Gene 1.0 ST chip, by which the analysts can explore the whole mRNA transcript in a single experiment [PPT+08].

Microarray data analysis is challenging. In particular, there are two major issues to face. A first issue is related to the high dimensionality of microarray data, which makes it necessary to resort to computer-based methods. Another issue is that the spots of the array trap information generally do not have a straightforward meaning; rather, the spots have to be compared and analyzed by possibly using statistical techniques.

Many approaches to microarray data analysis have been proposed by the research community [Dra03]. Most of them are essentially based on data mining techniques, in particular *clustering* methods [GMW07, JD88]. Clustering allows for understanding the huge mass of data in microarrays by grouping them in homogeneous subsets (clusters). In this way, cluster analysis aims to discover natural structures within the data and to help the analyst in identifying common structures and patterns in microarrays; for instance, finding similar expression patterns (i.e., co-expressed genes) which are related to cellular functions.

Microarray clustering approaches can be divided into three main categories [JTZ04]: *(i) gene-based clustering*, which treats genes as objects and samples as clustering features; *(ii) sample-based clustering*, where samples are the objects to be clustered and genes are the features; *(iii) co-clustering* approaches, where genes and samples are treated symmetrically (samples and genes can be both objects and features).

After several years of quantitative measurements of microarray probe-level data, new models have been proposed in order to manage the uncertainty of gene expression levels both in a single chip and across multiple chips. A novel probabilistic modeling approach is presented in [LMLR05], where the binding affinity of probe-pairs across multiple chips is modeled through a probabilistic model using Gamma distributions. In [LLAR07], a gene expression clustering algorithm has been proposed, which exploits the probabilistic modeling described above in order to improve performances with respect to classic techniques.

Within this view, we propose a new approach to modeling probe-level uncertainty in microarray data [GPT+09b], in order to achieve better quality in clustering results than traditional approaches. Basically, we propose to model microarray data with probe-level uncertainty as univariate uncertain objects (cf. Def. 3.2) and exploit U-AHC algorithm to discover a proper cluster hierarchy of genes.

5.7.2 Experimental Evaluation

Methodology

The U-AHC algorithm was evaluated in performing effective clustering of microarray data. In particular, the experiments were carried out on four large

microarray datasets, namely Leukaemia, Neuroblastoma, Myelodysplastic, and Mouse (cf. Appendix A).

The probe-level uncertainty for each dataset was extracted by exploiting the multi-mgMOS method [LMLR05].[6] For each dimension, the multi-mgMOS method yields a set of information that includes mean, standard deviation and principal percentiles (i.e., 5%, 25%, 50%, 75%, 95%). The information outputted by multi-mgMOS was exploited to model uncertainty according to the univariate uncertainty model. In particular, the univariate pdfs of each uncertain object (i.e., each row in the microarray matrix) was built by employing two different methods:

- *Normal* method, where Normal pdfs were easily derived from a combination of mean values with standard deviations;
- *Percentiles-based* method, where suitable statistical models were involved to fit pdfs to percentiles [Sil86].

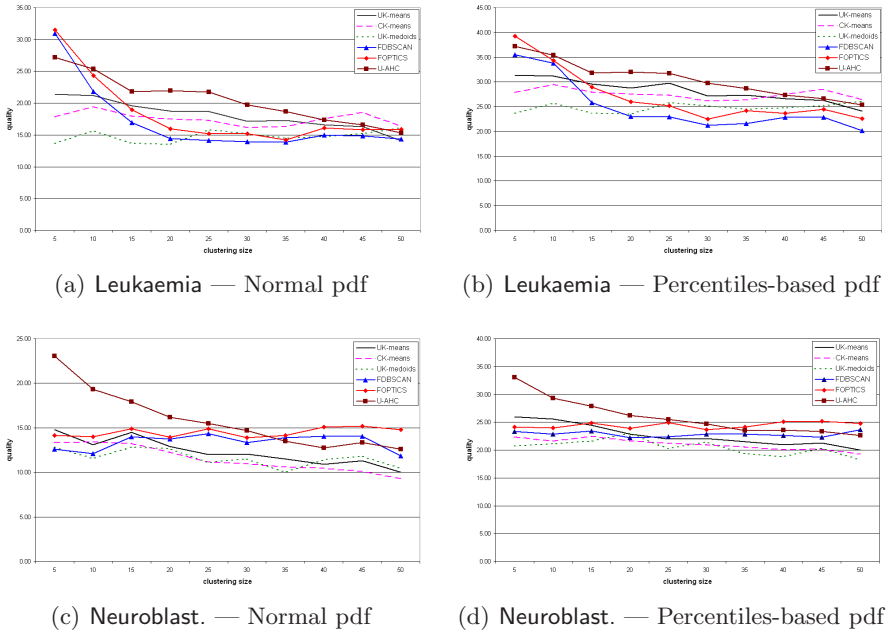
We performed a *gene-based clustering* in such a way that each group describes a particular macroscopic phenotype, such as cancer expressions or biological states [JTZ04]. Since there is no available reference classification for such data, we resorted to internal validity criteria based on the cophenetic correlation coefficient, which was exploited to evaluate a dendrogram according to how it preserved the pairwise distances between the original data objects (cf. Def. 2.10), and to the notion of quality (*qual*), defined in terms of intra- and inter-cluster distances (cf. (2.31)).

Results

For Leukaemia and Neuroblastoma datasets, we carried out multiple runs of U-AHC and its competing methods by varying the clustering size from a minimum of 5 to a maximum of 50 clusters. The clustering solutions obtained were evaluated by means of the internal criterion *qual*.

Figure 5.2 shows the quality results (*qual*) obtained by the clustering methods on the selected microarray datasets. As we can see, U-AHC achieved the best results averaged over the cluster sizes. In particular, U-AHC achieved the highest quality on Leukaemia (up to around 8% against the density-based methods), whereas behaved on average better than the other methods on Neuroblastoma; here, it can be noted that the density-based methods had a rapid decrease of the performances by increasing the number of clusters. By contrast, this decreasing trend was less evident in partitional methods on both datasets, which performed quite closely each other and tended to maintain a roughly constant quality on average. Also, the uncertainty generation based on percentiles (Fig. 5.2 (b) and (d)) generally led to higher quality results than the case in which normal pdfs are used to model the uncertain objects (Fig. 5.2 (a) and (c)).

[6] We used the Bioconductor package PUMA (*Propagating Uncertainty in Microarray Analysis*), available at http://www.bioinf.manchester.ac.uk/resources/puma/



(a) Leukaemia — Normal pdf (b) Leukaemia — Percentiles-based pdf

(c) Neuroblast. — Normal pdf (d) Neuroblast. — Percentiles-based pdf

Fig. 5.2. Accuracy results of U-AHC and competing methods in clustering microarray data (quality)

Table 5.3. Accuracy results of U-AHC in clustering microarray data (cophenetic coefficient)

dataset	*pdf form*	*cophenetic value*
Leukaemia	Normal	0.76
	Percentiles-based	0.82
Neuroblastoma	Normal	0.67
	Percentiles-based	0.75
Myelodysplastic	Normal	0.80
	Percentiles-based	0.89
Mouse	Normal	0.84
	Percentiles-based	0.92

We also evaluated the performance of U-AHC on Leukaemia, Neuroblastoma, Myelodysplastic, and Mouse in terms of cophenetic correlation coefficient. Such results are summarized in Table 5.3, where it can be noted that U-AHC obtained good accuracy results on all the datasets, from 67% to 84% with Normal pdfs. Also, the uncertainty generation based on percentiles generally

led to higher quality results than the previous case (about 8% on average); this improvement can be easily explained by the fact that percentiles provide a more refined representation of the uncertainty than the summarized information of mean value and standard deviation used for Normal pdf modeling.

Part III

Uncertainty in Data Clustering

– *Clustering Level* –

6

Clustering Ensembles: Background

Abstract This chapter provides background to the problem of clustering ensembles, which represents the solution to the clustering uncertainty problem addressed in this thesis. We firstly discuss the main notions that characterize clustering ensembles, i.e., the notions of ensemble, consensus partition, and diversity. Afterward, the state of the art in clustering ensembles is briefly reviewed by discussing the main methods falling into direct, instance-based, cluster-based, and hybrid classes of approaches.

6.1 Basic Definitions

Definition 6.1 (ensemble). *Given a set D of data objects, an* ensemble *defined over D is a set $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$, where $\mathcal{C}_h$ is a clustering solution defined over D, for each $h \in [1..H]$.*

Definition 6.2 (consensus partition). *Given a clustering ensemble $\mathcal{E}$, a* consensus partition *derived from $\mathcal{E}$ is a clustering solution $\mathcal{C}^*_{\mathcal{E}}$ that maximizes a given* consensus function *by exploiting information available from $\mathcal{E}$.*

For the sake of generality, any clustering ensembles method should be required to derive the consensus partition without accessing the original features of the objects in the data collection.

Building up an ensemble can be addressed by various ways, such as using different subsets of features [SG02, GTBC04], using different clustering algorithms [YY04], varying one or more (random) parameters of the clustering algorithm [BF98, FB03a, TJP03], or using different datasets obtained, e.g., by re-sampling the original dataset [SG02, DF03, FJ03, MTP04]. A crucial factor in the ensemble generation is the notion of *diversity*, which is used to quantify how the various clustering solutions in an ensemble are dissimilar to each other. This notion has been recognized as highly related to the accuracy of the consensus partition derived from an ensemble [FB03a, KH04, HKT06].

Definition 6.3 (partition-through diversity). *Given a clustering ensemble $\mathcal{E}$, a* partition-through diversity *measure defined over $\mathcal{E}$ is a function*

$\delta_P : \mathcal{E} \times \mathcal{E} \rightarrow \Re$ that quantifies, for each pair of clustering solutions $\mathcal{C}_h, \mathcal{C}_{h'} \in \mathcal{E}$, how $\mathcal{C}_h$ and $\mathcal{C}_{h'}$ are dissimilar to each other.

In the literature, partition-through diversity functions have been defined by resorting to external criteria used for assessing the quality of a clustering solution, such as NMI or F1, which are similarity measures that range within $[0, 1]$ (cf. Sect. 2.5). More precisely, the NMI- and F1-based partition-through diversity measures between two clustering solutions $\mathcal{C}'$ and $\mathcal{C}''$ can be defined according to the following formulas, respectively:

$$1 - NMI(\mathcal{C}', \mathcal{C}'') \tag{6.1}$$

$$1 - \sqrt{F1(\mathcal{C}', \mathcal{C}'') \times F1(\mathcal{C}'', \mathcal{C}')} \tag{6.2}$$

Note that the geometric mean is used in the definition of the F1-based partition-through diversity since F1 is originally not symmetric, unlike NMI.

Starting from the notion of partition-through diversity, the definitions of *clustering* and *ensemble* diversity notions are provided in the following.

Definition 6.4 (clustering diversity). *Given a clustering ensemble $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$ and a partition-through diversity measure δ_P defined over $\mathcal{E}$, the* clustering diversity *measure is a function $\delta_\mathcal{C} : \mathcal{E} \rightarrow \Re$ such that:*

$$\delta_\mathcal{C}(\mathcal{C}_h) = \frac{1}{H-1} \sum_{\substack{h' \in [1..H], \\ h' \neq h}} \delta_P(\mathcal{C}_h, \mathcal{C}_{h'}), \quad h \in [1..H]$$

Definition 6.5 (ensemble diversity). *Given a clustering ensemble $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$ and a partition-through diversity measure δ_P defined over $\mathcal{E}$, the* ensemble diversity *of $\mathcal{E}$ is defined as:*

$$\delta_\mathcal{E} = \frac{2}{H(H-1)} \sum_{h=1}^{H-1} \sum_{h'=h+1}^{H} \delta_P(\mathcal{C}_h, \mathcal{C}_{h'})$$

6.2 State of the Art

Clustering ensembles methods can be classified into four main approaches, namely *direct*, *instance-based*, *cluster-based*, and *hybrid*.

6.2.1 Direct Methods

Direct clustering ensembles methods are defined according to optimization criteria that involve a direct comparison between the solutions in the ensemble and the consensus partition. The algorithms proposed in [DWH01, DF03, FB03b] explicitly solve the *label correspondence problem* to find a correspondence between the cluster labels across the clusterings. In [LDJ07], clustering ensemble is mapped to a *Nonnegative Matrix Factorization* (NMF) problem.

6.2.2 Instance-based Methods

Instance-based methods are developed by involving a direct comparison between data objects. Most instance-based methods operate on the *co-occurrence* or *co-association* matrix $\mathbf{M}$. For each pair of objects (o', o''), this matrix stores the number of partitions of the ensemble in which o' and o'' appear in the same cluster divided by the size of the ensemble. In the *Majority Voting* (MV) algorithm [Fre01], $\mathbf{M}$ is "cut" at a given threshold, i.e., all the objects whose pairwise entry in $\mathbf{M}$ is greater than the threshold are joined into the same cluster. This approach has been proved to be equivalent to applying an AHC algorithm with single link metric on $\mathbf{M}$, cutting the resulting dendrogram according to the threshold [FJ02].

Other algorithms are based on using $\mathbf{M}$ either as a new data matrix [KHT06] or as a pair-wise distance matrix involved into a specific clustering algorithm. The *Agglomeration* (AGGL) algorithm [GMT07] uses Expectation Maximization or AHC with average linkage, whereas the *Iterative Voting Consensus* (IVC) algorithm [NC07] uses K-Means. In [ZTGG02], the AHC algorithm is applied to a pair-wise distance matrix derived from $\mathbf{M}$ by taking into account the statistical "signal" of the clusters in the ensemble.

In [SG02], clustering ensembles is mapped to a graph/hypergraph partitioning problem. The authors present two instance-based clustering ensembles methods, namely the *Cluster-based Similarity Partitioning Algorithm* (CSPA) and the *HyperGraph Partitioning Algorithm* (HGPA). CSPA induces a weighted graph from $\mathbf{M}$ and partitions it using the well-known graph partitioning algorithm METIS [KK98]. HGPA builds a hypergraph whose vertices are the data objects and the hyperedges are given by the clusters of all the clustering solutions in the ensemble; the consensus partition is then obtained by partitioning the hypergraph using HMETIS [KAKS97].

More recent graph-partitioning-based approaches are proposed in [AK03, DA09]. In [AK03], the weight of each edge (o', o'') in the induced graph is defined in terms of the size of the nearest neighbor list shared between the data objects o' and o''. In [DA09], the *Weighted Similarity Partitioning Algorithm* (WSPA) is proposed to combine multiple partitions that result from different runs of the projective clustering algorithm *Locally Adaptive Clustering* (LAC) [DGM$^+$07].

In [TJP05, WSB09], the features of the input data objects are re-defined according to the information available from the ensemble (e.g., by considering the specific cluster label, for each clustering of the ensemble) and involved into EM-like procedures.

6.2.3 Cluster-based Methods

Cluster-based clustering ensembles approaches are based on the principle "to cluster clusters". The key idea is to run a clustering algorithm on the set of all the clusters produced by the clustering solutions in the ensemble,

in order to compute a set of *meta-clusters*. The consensus partition is finally computed to assign each data object to the meta-cluster that maximizes some assignment criterion (e.g., majority voting).

The study in [BF98] proposes a two-stage clustering procedure. In the first stage, clustering solutions are obtained by multiple runs of the K-Means algorithm. Then, the output centroids from these clustering solutions are clustered by a further run of K-Means, and the resulting meta-centroids are used for the data assignment step.

The *Meta-CLustering Algorithm* (MCLA) [SG02] builds a graph whose vertices are the clusters of the various clustering solutions in the ensemble, and each edge (C', C'') has a weight equal to the Jaccard similarity coefficient [JD88] between the clusters relatively associated to the vertices C' and C''. The set of meta-clusters is computed by applying METIS on the graph, whereas the objects are assigned to the meta-clusters according to a majority voting criterion.

In [BO04], a *MetaCluster Search* (MCS) algorithm is formulated as a linear optimization problem to compute the optimum set of meta-clusters. The inter-cluster similarity is defined in terms of the Jaccard coefficient, and the assignment of the objects to the meta-clusters is accomplished by majority voting.

6.2.4 Hybrid Methods

Hybrid clustering ensembles methods attempt to combine ideas coming from both instance-based and cluster-based approaches. The objective is to build a *hybrid* bipartite graph whose vertices belong to the sets of objects and clusters.

The *Hybrid Bipartite Graph Formulation* (HBGF) algorithm [FB04] builds a bipartite graph whose edges (v_i^O, v_j^C) have weights equal to 1, if the object v_i^O belongs to the cluster v_j^C according to the clustering $\mathcal{C} \supset v_j^C$; otherwise, the weights are equal to zero. The clustering ensembles result is obtained by partitioning the graph according to standard methods (e.g., METIS) or spectral graph partitioning algorithms (e.g., [NJW01]).

The *Weighted Bipartite Partitioning Algorithm* (WBPA) [DA09] follows the same overall scheme of HBGF, although it allows for extending the range of weight values from $\{0, 1\}$ to $[0, 1]$.

In [DA09], the authors also propose the *Weighted Subspace Bipartite Partitioning Algorithm* (WSBPA). Given a set of projective clustering solutions computed by the LAC projective clustering algorithm [DGM$^+$07], WSBPA yields a consensus partition whose clusters are coupled with real-valued weights that quantify the relevance of each feature for that cluster.

6.2.5 Cluster Ensemble Selection and Weighted Consensus Clustering

Recently, there has been an increasing interest for some problems related to clustering ensembles; in particular, the *cluster ensemble selection* problem [CENS06, FL08] is to select a proper subset of solutions from an ensemble, and the *weighted consensus clustering* problem [LD08] is to automatically determine a proper weight for each solution in the ensemble. The key motivation for both problems arises from the fact that selecting a proper subset of clustering solutions (resp. assigning a proper weight to each clustering solution) allows for extracting a more accurate consensus partition than using the whole ensemble (resp. the unweighted version of the algorithm).

7

Diversity-based Weighting Schemes for Clustering Ensembles

Abstract Current methods of clustering ensembles typically fall into *instance-based*, *cluster-based*, or *hybrid* approaches; however, most of such methods fail in discriminating among the various clusterings that participate to the ensemble. In this chapter, we address the problem of weighting clustering ensembles by proposing general weighting approaches based on different implementations of the notion of diversity. We introduce three weighting schemes for clustering ensembles, called *Single Weighting*, *Group Weighting* and *Dendrogram Weighting*, which are independent of the particular method of clustering ensembles and designed to take into account correlations among the individual clustering solutions in different ways. We show how these schemes can be instantiated into any instance-based, cluster-based and hybrid clustering ensembles methods. Experiments have shown that the performance of the clustering ensembles algorithms increases when the proposed weighting schemes are employed.

7.1 Introduction

In recent years, various consensus functions for clustering ensembles have been defined and coupled with heuristic algorithms for maximizing them. Heuristic clustering ensembles algorithms are commonly based on three main approaches, namely *instance-based clustering ensembles*, *cluster-based clustering ensembles*, and *hybrid clustering ensembles* (cf. Chapt. 6). A common limit of all these approaches is that the consensus functions that they optimize are defined by equally considering the various clustering solutions in the ensemble. This is clearly a weak assumption for a number of reasons; for instance, an ensemble may be comprised of very different clusterings, as well as clusters that are somehow correlated with each other may appear in distinct clusterings of the ensemble. As a consequence, treating the constituent solutions of an ensemble equally and averaging over them to extract the consensus partition may not be effective.

In this chapter, we address the *weighted consensus clustering* problem by leveraging the importance of employing weighting schemes to discriminate

among the clustering solutions in an ensemble in extracting a proper consensus partition [GTG09]. Such a problem has been firstly introduced in [LD08]. However, that work proposes an optimization of an objective function which is derived from a specific formulation of the problem of clustering ensembles based on *Nonnegative Matrix Factorization* (NMF) [LDJ07]. By contrast, our work focuses for the first time on the development of general schemes for weighting clusterings which can be applied to any clustering ensembles method regardless of a specific approach. Thus, our proposed schemes are not related to any specific formulation of the clustering ensembles problem.

The contributions in this respect can be summarized as follows:

1. we propose three weighting schemes, called *Single Weighting*, *Group Weighting*, and *Dendrogram Weighting*, which are designed to take into account correlations among the individual clustering solutions to different levels;
2. we describe how any instance-based, cluster-based, and hybrid clustering ensembles approach can be easily reformulated to include weighting for the clustering solutions that participate to an ensemble;
3. we experimentally evaluated various state-of-the-art methods, for each one of the mentioned clustering ensembles approaches, with and without employing weighting schemes. Results have shown the beneficial impact of using a weighting scheme in improving the quality of the consensus partition from clustering ensembles.

7.2 Clustering Ensembles Weighting Schemes

Given a clustering ensemble $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$, we are interested in defining a vector $W = (w_1, \ldots, w_H)$ of weights , in such a way that each component w_h in W is assigned to the clustering solution $\mathcal{C}_h$ and reflects the relevance of $\mathcal{C}_h$ in determining the consensus partition.

In principle, W can be defined by resorting to traditional criteria that assess the validity of a clustering solution (cf. Chapt. 2). Unfortunately, this way is not applicable in this context, since neither external nor internal clustering validity criteria can be employed. Indeed, external criteria require prior knowledge of the ideal classification, whereas internal criteria can be used only if the original features of the clustered objects are available.

in the following, we propose three general schemes to compute the vector W, called *Single Weighting* (SW), *Group Weighting* (GW) and *Dendrogram Weighting* (DW). Each of these schemes is based on theoretical considerations on ensemble diversity (cf. Chapt. 6) and computes the vector $W = (w_1, \ldots, w_h)$ in such a way that $w_h \in [0, 1]$, for each $h \in [1..H]$, and $\sum_{h=1}^{H} w_h = 1$.

7.2.1 Single Weighting

The SW scheme takes into account each $\mathcal{C}_h \in \mathcal{E}$ individually. The key idea consists in evaluating the ensemble diversity of $\mathcal{E} \setminus \{\mathcal{C}_h\}$, i.e., $\delta_{\mathcal{E}\setminus\{\mathcal{C}_h\}}$, and defining w_h proportionally to $\delta_{\mathcal{E}\setminus\{\mathcal{C}_h\}}$.

Most research works focusing on clustering ensembles diversity suggest to generate ensembles according to a *maximum diversity criterion*, which states that the higher the ensemble diversity, the better the accuracy of the consensus partition extracted from the ensemble [FB03a, KH04, DA09]. This is an empirical assumption, which may not hold in general. Indeed, the study in [HKT06] shows that, in some cases, ensembles generated by a *median diversity criterion* (i.e., ensembles that exhibit a moderate level of diversity) produce a more accurate consensus partition than ensembles having higher diversity.

We take into account both the above intuitions and define W in such a way that it follows a mixture density composed by

- a linearly increasing distribution, W', which defines weights according to a maximum diversity criterion,
- a Normal distribution, W'', which computes weights according to a median diversity criterion.

Precisely, we define W as:

$$W = \alpha\ W' + (1 - \alpha)\ W'' \tag{7.1}$$

$W' = (w'_1, \ldots, w'_H)$ is a vector whose component values w'_h linearly increase as the quantity $\delta_{\mathcal{E}\setminus\{\mathcal{C}_h\}}$ decreases. By contrast, the normally distributed components of $W'' = (w''_1, \ldots, w''_H)$ are defined in such a way that the maximum value in W'' corresponds to the clustering solution $\mathcal{C}_h$ having the median $\delta_{\mathcal{E}\setminus\{\mathcal{C}_h\}}$. The importance of the two vectors W' and W'' in computing W is established by the user-defined parameter α (ranging within $[0, 1]$). Formally, each w'_h of W' ($h \in [1..H]$) is defined as:

$$w'_h = \left(1 - \frac{\delta_{\mathcal{E}\setminus\{\mathcal{C}_h\}}}{\sum_{h'=1}^{H} \delta_{\mathcal{E}\setminus\{\mathcal{C}_{h'}\}}}\right) \Bigg/ \Big(H - 1\Big) \tag{7.2}$$

and each w''_h of W'' ($h \in [1..H]$) is defined as:

$$w''_h = \frac{N_{\mu,\sigma}(\delta_{\mathcal{E}\setminus\{\mathcal{C}_h\}})}{\sum_{h'=1}^{H} N_{\mu,\sigma}(\delta_{\mathcal{E}\setminus\{\mathcal{C}_{h'}\}})} \tag{7.3}$$

where $N_{\mu,\sigma}$ is the Normal probability density function having mean μ and standard deviation σ, i.e.,

$$N_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Let us denote with Δ the set $\{\delta_{\mathcal{E}\setminus\{\mathcal{C}\}} \mid \mathcal{C} \in \mathcal{E}\}$, and with $\overline{d}$, d_{min}, d_{max} the median, the minimum and the maximum value in Δ. We define $\mu = \overline{d}$ and $\sigma = \hat{d}/3$, where $\hat{d} = \max_{d \in \{d_{min}, d_{max}\}} |\overline{d} - d|$. This choice of σ guarantees that the condition $\int_{\overline{d}-\hat{d}}^{\overline{d}+\hat{d}} N_{\mu,\sigma}(x)\mathrm{d}x \approx 1$ holds.

7.2.2 Group Weighting

The SW scheme leads to the construction of the vector W by considering each clustering solution individually. However, an ensemble may not contain only solutions that are totally dissimilar to each other; instead, in a real scenario, an ensemble comprises a number of subsets of (highly) correlated clusterings, which tend to bias the consensus partition.

Within this view, an intuitive refinement of SW can work as follows. The subsets of correlated clusterings are initially detected, then a *macro*-weight is preliminarily assigned to each of these subsets to quantify the importance of the whole corresponding group of clusterings. Based on the macro-weight assigned to the specific subset, a *micro*-weight is finally computed for each clustering of that subset.

The aforementioned idea is at the basis of the proposed GW scheme, whose outline can be summarized as follows:

1: partition the ensemble $\mathcal{E}$ into a set of clusters (of clusterings) $\mathbf{C} = \{\mathrm{C}_1, \ldots, \mathrm{C}_K\}$
2: compute the weight vector $W_{\mathbf{C}} = (w_{\mathbf{C}}^{(1)}, \ldots, w_{\mathbf{C}}^{(K)})$, where each $w_{\mathbf{C}}^{(k)}$, $k \in [1..K]$, is assigned to the cluster (of clusterings) $\mathrm{C}_k \in \mathbf{C}$
3: compute the weight vector $W = (w_1, \ldots, w_H)$ from $W_{\mathbf{C}}$, in which each w_h, $h \in [1..H]$, is assigned to the clustering solution $\mathcal{C}_h \in \mathcal{E}$

As shown in the outline, the task of detecting the subsets of correlated clusterings is accomplished by *clustering the clustering solutions*. This idea is not new in the context of clustering ensemble, since it has been previously involved into the cluster ensemble selection problem [CENS06, FL08]. However, in this work we bring out for the first time this idea for solving the weighted consensus clustering problem.

Once the "to cluster clusterings" step has been performed, the vector $W_{\mathbf{C}}$ is computed in a way similar to the SW scheme. The only difference is that GW considers the ensemble diversity of the sets obtained by subtracting the clustering solutions in the various clusters from the whole ensemble; by contrast, SW takes into account the diversity of the ensemble when a single clustering solution is subtracted from it. Formally, we compute $W_{\mathbf{C}}$ as:

$$W_{\mathbf{C}} = \alpha\ W'_{\mathbf{C}} + (1 - \alpha)\ W''_{\mathbf{C}} \tag{7.4}$$

where each $w_{\mathbf{C}}^{(k)'}$ of $W'_{\mathbf{C}}$ ($k \in [1..K]$) is defined as:

$$w_{\mathbf{C}}^{(k)'} = \left(1 - \frac{\delta_{\mathcal{E}\setminus\mathrm{C}_k}}{\sum_{k'=1}^{K} \delta_{\mathcal{E}\setminus\mathrm{C}_{k'}}}\right) \Big/ \left(K - 1\right) \tag{7.5}$$

and each $w_{\mathbf{C}}^{(k)''}$ of $W''_{\mathbf{C}}$ ($k \in [1..K]$) is defined as:

$$w_{\mathbf{C}}^{(k)''} = \frac{N_{\mu,\sigma}(\delta_{\mathcal{E}\setminus \mathrm{C}_k})}{\sum_{k'=1}^{K} N_{\mu,\sigma}(\delta_{\mathcal{E}\setminus \mathrm{C}_{k'}})} \tag{7.6}$$

From $W_{\mathbf{C}}$, we compute the vector of *micro*-weights W, which is the output of the GW scheme. Each w_h of W ($h \in [1..H]$) is defined as:

$$w_h = w_{h,k}^{SW} \times \hat{w}_k \tag{7.7}$$

where

- $w_{h,k}^{SW}$ is the weight assigned to the clustering solution $\mathcal{C}_h$ according to the SW scheme, when the ensemble is given by $\mathrm{C}_k \in \mathbf{C}$. C_k is the cluster such that $\mathcal{C}_h \in \mathrm{C}_k$,
- $\hat{w}_k \in W_{\mathbf{C}}$ is the weight assigned to C_k in the first step of GW.

7.2.3 Dendrogram Weighting

A major issue in the GW scheme is the requirement of a clustering algorithm to partition the ensemble and its relative parameter settings, such as the number of output clusters. To this purpose, we define a further weighting scheme, named Dendrogram Weighting (DW), to maintain the advantageous features of the GW scheme while overcoming the problem of choosing a clustering algorithm. The DW scheme is based on theoretical considerations on the dendrogram which can be built over the clustering solutions in the ensemble. In particular, DW consists of two main steps. First, the clusterings in the ensemble are clustered by using a hierarchical algorithm in order to organize them into a dendrogram. Then, the dendrogram is used as an intuitive tool for understanding relationships among the clusterings; this information is eventually exploited for properly defining the clustering weights.

Once a dendrogram $\mathbf{T}_\ell$ has been defined over the ensemble $\mathcal{E}$,[1] the weight vector W is finally computed by associating each clustering solution $\mathcal{C}_h \in \mathcal{E}$ with a coefficient γ_h. This coefficient expresses the correlation of $\mathcal{C}_h$ with the other clusterings in the ensemble, based on the set S_h (i.e., the set of different clusters of the dendrogram that contain $\mathcal{C}_h$). Precisely, γ_h is defined as directly proportional to the size of S_h and inversely proportional to the sum of the dendrogram levels that contain the clusters in S_h:

$$\gamma_h = \sum_{q=1}^{Q-1} (Q-q) I(\mathbf{T}_\ell, \mathcal{C}_h, \mathcal{L}_q) \tag{7.8}$$

where $I(\mathbf{T}_\ell, \mathcal{C}_h, \mathcal{L}_q)$ is an indicator function that returns 1 if there exists some "new" cluster at the level $\mathcal{L}_q$ of dendrogram $\mathbf{T}_\ell$ that contains $\mathcal{C}_h$, otherwise

[1] We refer to level-organized dendrograms as defined in Def. 2.3.

the function returns 0. Formally, let $\hat{L}_q \in \mathcal{L}_q$ and $\hat{L}_{q-1} \in \mathcal{L}_{q-1}$ be the clusters such that $\mathcal{C}_h \in \hat{L}_q$ and $\mathcal{C}_h \in \hat{L}_{q-1}$:

$$I(\mathbf{T}_\ell, \mathcal{C}_h, \mathcal{L}_q) = \begin{cases} 1 \text{ if } \hat{L}_q \neq \hat{L}_{q-1} \\ 0 \text{ otherwise} \end{cases}$$

The intuition underlying γ_h can be explained as follows. If the clustering $\mathcal{C}_h$ belongs to a large number of different clusters in the dendrogram, then $\mathcal{C}_h$ is expected to be correlated with a large number of other clusterings in the ensemble; therefore, γ_h should be low. On the other hand, the higher the dendrogram level of the clusters containing $\mathcal{C}_h$, the lower the correlation of $\mathcal{C}_h$ with the other clusterings in the ensemble; indeed, a high dendrogram level means that the corresponding clusters are less compact than the clusters formed at the lower levels.

In order to define the final weight vector W, we resort to a similar approach used for the SW scheme. In particular, we employ the same equations used for SW (i.e., (7.1)-(7.3)), where $\delta_{\mathcal{E}\setminus\{\mathcal{C}_h\}}$ is replaced with the coefficients γ_h, $h \in [1..H]$.

Computational Aspects.

Given an ensemble of size H, the computational complexity of the proposed weighting schemes is the following.

- *Single Weighting* performs in $\mathcal{O}(H^2)$
- *Group Weighting* performs in $\mathcal{O}(A + KH^2)$, where A is the execution cost required by the "to cluster clusterings" step, and K is the number of output clusters of clusterings. Under the reasonable assumption that K is a constant and A is $\mathcal{O}(KH^2)$, the complexity of the GW scheme is $\mathcal{O}(H^2)$
- *Dendrogram Weighting* performs in $\mathcal{O}(H^2)$

7.3 Involving Weights in Clustering Ensembles Algorithms

In this section we provide a formulation of the instance-based, cluster-based and hybrid clustering ensembles methods which takes into account weights for the clustering solutions in the ensembles.

7.3.1 Weighted Instance-based Clustering Ensembles

Algorithm 7.1 outlines the general scheme of a weighted instance-based clustering ensembles method (WICE).

Initially, each data object $o_i \in D$ is replaced with a new one o'_i which is defined over the space of features according to the information stored in

Algorithm 7.1 WICE: Weighted Instance-based Clustering Ensembles

Input: a set $D = \{o_1, \ldots, o_n\}$ of data objects, where $o_i = (\omega_{i1}, \ldots, \omega_{im})$, $i \in [1..n]$;
an ensemble $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$ defined over D;
a weight vector $W = (w_1, \ldots, w_H)$
Output: the consensus partition $\mathcal{C}^*_\mathcal{E}$

1: **for all** $i \in [1..n]$ **do**
2: replace o_i with $o'_i = (\omega'_{i1}, \ldots, \omega'_{iH})$
3: **end for**
4: **for all** $a \in [1..n], b \in [1..n]$ **do**
5: $(\mathbf{M}')_{ab} = \Phi(w_1\phi(\omega'_{a1}, \omega'_{b1}), \ldots, w_H\phi(\omega'_{aH}, \omega'_{bH}), \Gamma(o'_a, o'_b))$
6: **end for**
7: $\mathcal{C}^*_\mathcal{E} \leftarrow cluster(D, \mathbf{M}')$

$\mathcal{E}$ (Lines 1-3). Each feature ω'_{ih} of o'_i ($h \in [1..H]$) is defined according to the clustering solution $\mathcal{C}_h \in \mathcal{E}$ and depends on the specific instance-based algorithm. Once the objects o'_i have been defined, the matrix $\mathbf{M}'$ storing the (weighted) pair-wise distances for the data objects is computed (Lines 4-6). Each entry $(\mathbf{M}')_{ab}$ is computed in terms of the functions $\Phi : \Re^{H+1} \rightarrow \Re$, $\phi : \Re^2 \rightarrow \Re$, and $\Gamma : \Re^{2H} \rightarrow \Re$. Note that Φ, ϕ and Γ are properly defined depending on the specific instance-based algorithm. The entries in $\mathbf{M}'$ should give more weight to the information coming from clusterings whose associated weights are higher. Finally, the output consensus partition $\mathcal{C}^*_\mathcal{E}$ is computed by performing a further clustering task on the objects in D, where $\mathbf{M}'$ is used as the pair-wise distance matrix (Line 7).

According to Alg. 7.1, any instance-based method provides a specific way of computing the objects o'_i, and the functions Φ, ϕ and Γ. As an example, a clustering ensembles algorithm using a co-occurrence matrix with Euclidean distance values (e.g., [GMT07, NC07]) should be equipped with $o'_i = (\lambda_1(o_i), \ldots, \lambda_H(o_i))$, where each $\lambda_h(o)$ returns the identifier of the cluster in $\mathcal{C}_h$ that contains o, whereas $\Phi(y_1, \ldots, y_{H+1}) = \sqrt{y_1 + \cdots + y_H}$, $\phi(y_1, y_2) = (y_1 - y_2)^2$, and $\Gamma(y_1, \ldots, y_{2H}) = 0$.

7.3.2 Weighted Cluster-based Clustering Ensembles

The weighted cluster-based clustering ensembles algorithm, WCCE, consists of two main phases (Algorithm 7.2).

First, a preliminary task of clustering is performed over the union set $D_\mathcal{M}$ of all the clusters belonging to the clustering solutions in $\mathcal{E}$, in order to obtain a set $\mathcal{M}$ of *meta-clusters* (Lines 2-4). The clustering procedure involves $\mathbf{M}_{D_\mathcal{M}}$ as a matrix storing the distances between the pairs of clusters in $D_\mathcal{M}$ (Line 3). $\mathbf{M}_{D_\mathcal{M}}$ is properly defined according to the specific cluster-based algorithm (e.g., by using the Jaccard coefficient). Then, the output consensus partition is derived by assigning each $o_i \in D$ to one and only one meta-cluster in $\mathcal{M}$ (Lines 5-8) based on: *(i)* some criterion of object-to-meta-cluster assignment (which

Algorithm 7.2 WCCE: Weighted Cluster-based Clustering Ensembles

Input: a set $D = \{o_1, \ldots, o_n\}$ of data objects;
an ensemble $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$ defined over D;
a weight vector $W = (w_1, \ldots, w_H)$
Output: the consensus partition $\mathcal{C}^*_\mathcal{E}$

1: $\mathcal{C}^*_\mathcal{E} = \{C^*_1, \ldots, C^*_K\} \leftarrow \{\emptyset, \ldots, \emptyset\}$
2: $D_\mathcal{M} \leftarrow \bigcup_{\mathcal{C} \in \mathcal{E}} \mathcal{C}$
3: $\mathbf{M}_{D_\mathcal{M}} \leftarrow pair\text{–}wise\text{–}distances(D_\mathcal{M})$
4: $\mathcal{M} = \{\mathrm{M}_1, \ldots, \mathrm{M}_K\} \leftarrow cluster(D_\mathcal{M}, \mathbf{M}_{D_\mathcal{M}})$
5: **for all** $i \in [1..n]$ **do**
6: find $\mathrm{M}_k \in \mathcal{M}$ such that $\mathrm{M}_k \leftarrow assign(o_i, \mathcal{M}, W)$
7: $C^*_k \leftarrow C^*_k \cup \{o_i\}$
8: **end for**

depends on the specific cluster-based method) and *(ii)* the weight vector W defined over the ensemble.

Most cluster-based algorithms adopt the so-called *majority voting* [SG02, BO04] as an object-to-meta-cluster assignment criterion. Precisely, each $o_i \in D$ is assigned to the meta-cluster $\hat{\mathrm{M}} = \arg\max_{\mathrm{M} \in \mathcal{M}} \sum_{C \in \mathrm{M}} I[o_i \in C]$.

A weighted version of the majority voting criterion can be easily derived inasmuch as each $o_i \in D$ is assigned to the meta-cluster

$$\hat{\mathrm{M}} = \arg\max_{\mathrm{M} \in \mathcal{M}} \sum_{C \in \mathrm{M}} wI[o_i \in C]$$

where w is the weight associated to the clustering $\mathcal{C} \in \mathcal{E}$ such that $C \in \mathcal{C}$.

7.3.3 Weighted Hybrid Clustering Ensembles

Any hybrid clustering ensembles method exploits information coming from both instance-based and cluster-based approaches, and can be described by the outline reported in Alg. 7.3.

Initially, a *hybrid* bipartite graph G is built (Lines 1-9). The vertex set of G contains both the data objects in D (the set $\mathcal{V}_o$) and the clusters of each clustering solution in $\mathcal{E}$ (the set $\mathcal{V}_c$) (Lines 1-2). The weighted edge set E comprises links between vertices in $\mathcal{V}_o$ and vertices in $\mathcal{V}_c$, whereas the weight of each edge is defined according to the specific hybrid algorithm and takes into account the weight vector W (Lines 3-8). The weights in W are used to enhance the edge weights; precisely, given any two nodes $v_1 \in \mathcal{V}_o$, $v_2 \in \mathcal{V}_c$ and the corresponding edge $e = (v_1, v_2)$, the (new) associated weight $\hat{w}$ is obtained by multiplying w (i.e., the weight originally assigned to the edge e by the specific hybrid method) to $(1 + w_h)$, where w_h in W is the weight associated to the clustering $\mathcal{C}_h$ such that $v_2 \in \mathcal{C}_h$. Finally, the output consensus partition is derived by partitioning G by means of a suitable procedure (e.g., METIS [KK98]) that depends on the specific hybrid algorithm.

Algorithm 7.3 WHCE: Weighted Hybrid Clustering Ensembles

Input: a set $D = \{o_1, \ldots, o_n\}$ of data objects;
an ensemble $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$ defined over D;
a weight vector $W = (w_1, \ldots, w_H)$
Output: the consensus partition $\mathcal{C}^*_\mathcal{E}$

$\mathcal{V}_o \leftarrow D$
$\mathcal{V}_c \leftarrow \bigcup_{\mathcal{C} \in \mathcal{E}} \mathcal{C}$
$E \leftarrow \emptyset$
for all $v_o \in \mathcal{V}_o$ **do**
  for all $v_c \in \mathcal{V}_c$ **do**
    $\hat{w} = weight(v_o, v_c, \mathcal{E}, W)$
    $E \leftarrow E \cup \{(v_o, v_c, \hat{w})\}$
  end for
end for
$G \leftarrow \langle \mathcal{V}_o \cup \mathcal{V}_c, E \rangle$
$\mathcal{C}^*_\mathcal{E} \leftarrow partition(G)$

7.4 Experimental Evaluation

We devised an experimental evaluation in order to assess the impact of employing the proposed weighting schemes in clustering ensembles. To this purpose, we evaluated and compared the performances of instance-based, cluster-based and hybrid clustering ensemble algorithms with and without each weighting scheme. Specifically, in the experiments we involved the following clustering algorithms which have been discussed in Chapt. 6:

- CSPA [SG02], HGPA [SG02], WSPA [DA09], MV [Fre01], AGGL [GMT07], and IVC [NC07], as instance-based methods;
- MCLA [SG02] and MCS [BO04], as cluster-based methods;
- HBGF [FB04] and WBPA [DA09], as hybrid methods.

Note that, in case of weighted clustering ensembles, we adapted each of the selected clustering methods according to the algorithm schemes presented in Sect. 7.3.

In the following, we discuss the evaluation methodology used in the experiments, and present the main experimental results obtained on the various datasets.

7.4.1 Evaluation Methodology

Datasets

We used five benchmark datasets from the UCI Machine Learning Repository [ANml], namely Glass, Ecoli, ImageSegmentation, ISOLET, and Letter-Recognition. In addition to UCI datasets, we used two time series datasets

coming from different application domains [KXWRta], namely Tracedata and ControlChart. For a description of the selected datasets, see Appendix A.

Cluster Validity

To assess the quality of a consensus partition belonging to an ensemble, we exploited the availability of reference classifications for the datasets. The objective was to evaluate how well a clustering fits a predefined scheme of known classes (natural clusters). To this purpose, we resorted to NMI (cf. Def. 2.6) and F1 (cf. Def. 2.5) measures.

Ensemble Generation

To generate an ensemble we varied the clustering algorithm, the setting of the selected clustering algorithms, the number of features of the original data objects, and the number of output clusters. Precisely, the ensemble for each dataset was built up as follows:

1. We computed a set of clustering solutions, which were obtained by performing multiple runs of the K-Means algorithm on the specific dataset with different random initializations.
2. We completed the ensemble by adding a further set of clustering solutions obtained by varying the feature selection of the original data, the clustering algorithm, and the number of output clusters. In particular, the feature set was varied by randomly selecting subsets having 40%, 50%, 60%, 70%, 80%, 90%, and 100% size of the original feature space. The K-Means and AHC with average link algorithms were used for the clustering task, and the output clustering solutions were composed by a number of clusters equal to 2 and 50%, 75%, 100%, 150%, and 200% of the number of ideal classes of the specific dataset.

Setting of Weighting Schemes

For each of the proposed weighting schemes, we used both NMI and F1 in order to measure the ensemble diversity (cf. (6.1)-(6.2)). We also performed a preliminary phase of tuning of the parameter α and finally presented the clustering performance corresponding to the relative best setting of α for each weighting scheme. We noted that varying α seemed not to have a significant impact on the overall results—a generally valid setting was $\alpha = 0.65$.

In addition to the setting of α, the GW scheme also requires the selection of a clustering algorithm for the step "to cluster clusterings". We tried different well-known algorithms, such as K-Means and AHC with single link, complete link, and average link. In the following, we present results obtained by using the AHC with average link.

Setting of Clustering Ensembles Methods

For each method and dataset, we set the number of output clusters K equal to the number of ideal classes of the specific dataset. Also, as far as the graph-partitioning-based methods (i.e., CSPA, HGPA, WSPA, MCLA, HBGF, WBPA), we set the METIS parameters as suggested in [KK98]; WSPA and WBPA additionally require the number of LAC iterations, which was set equal to the size of the ensemble generated for each dataset.

It should be remarked that setting the clustering methods had a marginal importance in this work, since the main focus of our experimental evaluation was on assessing the effectiveness of the methods with and without employing the proposed weighting schemes.

7.4.2 Results

Tables 7.1–7.4 show the accuracy results obtained by the various clustering ensembles algorithms, with and without employing weighting schemes, on the selected datasets. Accuracy results are reported in terms of NMI and F1.

Evaluation of Weighted Clustering Ensembles

Looking at the tables, a first important remark is that, for each of the clustering algorithms, weighted settings led to better performance in general.

Regardless of the specific weighting scheme or clustering ensembles algorithm, we observed the following maximum improvements of clustering quality with respect to the case no weighting scheme was used: 24% on ControlChart, 22% on ISOLET, 18% on ImageSegmentation, 15% on Glass, 12% on LetterRecognition, 11% on Tracedata, and 8% on Ecoli.

Evaluation of Weighting Schemes

Comparing the proposed weighting schemes, the DW scheme led to the maximum quality improvements on all the datasets. Moreover, the DW-based weighted version of each clustering ensembles method performed as good as or better than the original (unweighted) clustering method in most cases (i.e., except WSPA on Glass and MV on LetterRecognition, with F1 and NMI as diversity measures, respectively).

As far as the other two weighting schemes, the adoption of GW led to better maximum performance than SW in nearly all datasets. However, considering the average performance (i.e., the average increase in accuracy with respect to the unweighted settings, over all the algorithms), GW behaved less reliably than SW. This can be explained since GW requires a phase of parameter tuning which is more critical than in the SW case; however, GW is in principle designed as a refinement of SW and is really effective in improving the performance of the clustering ensembles algorithms.

Table 7.1. Evaluating SW, GW, and DW clustering weighting schemes: (a) Glass, and (b) Ecoli

(a)

method	*diversity*	*accuracy*			
		no weights	*SW*	*GW*	*DW*
CSPA	NMI	0.56	0.61	0.65	0.66
	F1	0.77	0.78	0.74	0.78
HGPA	NMI	0.55	0.57	0.56	0.57
	F1	0.66	0.71	0.75	0.72
WSPA	NMI	0.65	0.67	0.73	0.71
	F1	0.75	0.71	0.73	0.72
MV	NMI	0.65	0.65	0.64	0.70
	F1	0.71	0.72	0.69	0.74
AGGL	NMI	0.67	0.67	0.65	0.68
	F1	0.70	0.70	0.70	0.70
IVC	NMI	0.55	0.68	0.65	0.70
	F1	0.71	0.77	0.75	0.80
MCLA	NMI	0.56	0.61	0.62	0.65
	F1	0.64	0.70	0.65	0.67
MCS	NMI	0.58	0.58	0.57	0.60
	F1	0.70	0.68	0.67	0.70
HBGF	NMI	0.64	0.64	0.62	0.64
	F1	0.76	0.77	0.76	0.77
WBPA	NMI	0.66	0.69	0.70	0.73
	F1	0.68	0.70	0.73	0.71

(b)

method	*diversity*	*accuracy*			
		no weights	*SW*	*GW*	*DW*
CSPA	NMI	0.66	0.68	0.65	0.69
	F1	0.70	0.70	0.68	0.70
HGPA	NMI	0.64	0.64	0.66	0.65
	F1	0.70	0.71	0.74	0.71
WSPA	NMI	0.59	0.62	0.59	0.62
	F1	0.62	0.64	0.63	0.63
MV	NMI	0.68	0.69	0.66	0.69
	F1	0.85	0.85	0.84	0.86
AGGL	NMI	0.63	0.67	0.67	0.69
	F1	0.74	0.75	0.77	0.75
IVC	NMI	0.65	0.63	0.66	0.68
	F1	0.80	0.74	0.86	0.80
MCLA	NMI	0.59	0.62	0.65	0.67
	F1	0.69	0.70	0.73	0.73
MCS	NMI	0.58	0.58	0.57	0.60
	F1	0.70	0.68	0.67	0.70
HBGF	NMI	0.62	0.62	0.59	0.62
	F1	0.72	0.74	0.70	0.75
WBPA	NMI	0.71	0.72	0.74	0.72
	F1	0.73	0.75	0.75	0.77

For instance, on ISOLET (Table 7.2), GW allowed the clustering ensembles algorithms to achieve up to 22% (resp. 21%) of maximum quality improvement according to NMI (resp. F1), against the 16% (resp. 17%) improvement obtained by employing the SW scheme. However, on the same dataset, no benefit resulted from the adoption of the GW scheme in six out of twenty cases (over all the algorithms and the reported performance).

Table 7.2. Evaluating SW, GW, and DW clustering weighting schemes: (a) Image-Segmentation, and (b) ISOLET

(a)

method	*diversity*	*accuracy*			
		no weights	*SW*	*GW*	*DW*
CSPA	NMI	0.49	0.51	0.53	0.51
	F1	0.55	0.56	0.60	0.59
HGPA	NMI	0.38	0.50	0.53	0.56
	F1	0.45	0.54	0.55	0.59
WSPA	NMI	0.53	0.63	0.63	0.61
	F1	0.61	0.71	0.70	0.68
MV	NMI	0.45	0.45	0.46	0.45
	F1	0.69	0.78	0.77	0.75
AGGL	NMI	0.58	0.58	0.58	0.58
	F1	0.68	0.69	0.66	0.69
IVC	NMI	0.51	0.57	0.55	0.60
	F1	0.59	0.65	0.60	0.69
MCLA	NMI	0.51	0.52	0.52	0.54
	F1	0.63	0.63	0.68	0.67
MCS	NMI	0.58	0.58	0.57	0.60
	F1	0.70	0.68	0.67	0.70
HBGF	NMI	0.48	0.53	0.51	0.53
	F1	0.57	0.58	0.60	0.59
WBPA	NMI	0.51	0.52	0.53	0.52
	F1	0.53	0.55	0.57	0.57

(b)

method	*diversity*	*accuracy*			
		no weights	*SW*	*GW*	*DW*
CSPA	NMI	0.55	0.55	0.58	0.58
	F1	0.62	0.62	0.65	0.64
HGPA	NMI	0.45	0.59	0.55	0.61
	F1	0.50	0.62	0.57	0.63
WSPA	NMI	0.55	0.62	0.63	0.66
	F1	0.64	0.70	0.70	0.74
MV	NMI	0.49	0.50	0.46	0.50
	F1	0.75	0.81	0.74	0.84
AGGL	NMI	0.66	0.67	0.63	0.68
	F1	0.72	0.72	0.71	0.72
IVC	NMI	0.44	0.60	0.66	0.66
	F1	0.53	0.70	0.74	0.73
MCLA	NMI	0.55	0.62	0.60	0.62
	F1	0.67	0.67	0.71	0.73
MCS	NMI	0.58	0.58	0.57	0.60
	F1	0.70	0.68	0.67	0.70
HBGF	NMI	0.56	0.69	0.68	0.66
	F1	0.63	0.69	0.69	0.65
WBPA	NMI	0.58	0.60	0.63	0.61
	F1	0.65	0.66	0.71	0.70

Evaluation of Diversity Measures

Using F1 as diversity criterion, the accuracy results were generally higher than in the NMI setting, i.e., the maximum quality of the consensus partition observed on all the datasets always referred to F1 values. However, from the perspective of the advantages that can be derived from using a weighting scheme, the highest average gains (over the performance of all the methods)

Table 7.3. Evaluating SW, GW, and DW clustering weighting schemes: LetterRecognition

method	*diversity*	*accuracy*			
		no weights	*SW*	*GW*	*DW*
CSPA	NMI	0.40	0.48	0.47	0.48
	F1	0.51	0.60	0.61	0.62
HGPA	NMI	0.41	0.40	0.38	0.41
	F1	0.51	0.48	0.52	0.53
WSPA	NMI	0.43	0.45	0.45	0.45
	F1	0.52	0.53	0.49	0.53
MV	NMI	0.72	0.70	0.68	0.70
	F1	0.80	0.80	0.80	0.80
AGGL	NMI	0.63	0.64	0.63	0.65
	F1	0.68	0.70	0.74	0.74
IVC	NMI	0.38	0.43	0.41	0.43
	F1	0.46	0.56	0.55	0.58
MCLA	NMI	0.45	0.49	0.51	0.53
	F1	0.56	0.59	0.59	0.62
MCS	NMI	0.48	0.50	0.50	0.53
	F1	0.50	0.52	0.48	0.55
HBGF	NMI	0.40	0.41	0.42	0.42
	F1	0.51	0.52	0.50	0.55
WBPA	NMI	0.46	0.48	0.50	0.51
	F1	0.52	0.52	0.56	0.57

were obtained in terms of NMI on four out of seven datasets (i.e., Glass, Ecoli, ISOLET, and ControlChart).

For instance, on ControlChart, using the DW scheme led to a maximum increase in quality (with respect to unweighted clustering methods) which was equal to 24% and 19% in terms of NMI and F1, respectively; the average increase in quality was 8% (NMI) and 6.5% (F1). On ImageSegmentation, the maximum gain was achieved in terms of NMI (18%, against 14% by F1) by using the DW scheme; the average improvement instead referred to the F1 diversity (5.2%, against 4.3% by NMI).

Evaluation of Clustering Ensembles Methods

Instance-based methods showed better performance with respect to methods belonging to the other two clustering ensembles approaches, on all datasets (except for Tracedata). For instance, considering the results based on NMI, we observed the following differences between the maximum NMI values scored by the best and the worst approach: 19% on LetterRecognition, 16% on ControlChart, 10% on ImageSegmentation, 9% on Ecoli, 8% on Tracedata and Glass, and 7% on ISOLET.

Concerning the algorithms, MV ranked first followed by IVC and HGPA, according to the F1-based diversity criterion; by contrast, in the NMI-based evaluation, more algorithms alternated with each other as best performer on the various datasets.

However, looking at the average performance over all the methods for each clustering approach and dataset, we observed that there was no ap-

Table 7.4. Evaluating SW, GW, and DW clustering weighting schemes: (a) Tracedata, and (b) ControlChart

(a)

method	*diversity*	*accuracy*			
		no weights	*SW*	*GW*	*DW*
CSPA	NMI	0.50	0.51	0.48	0.50
	F1	0.53	0.54	0.51	0.54
HGPA	NMI	0.53	0.55	0.56	0.58
	F1	0.64	0.65	0.67	0.67
WSPA	NMI	0.50	0.50	0.50	0.50
	F1	0.52	0.55	0.55	0.57
MV	NMI	0.50	0.54	0.57	0.54
	F1	0.53	0.59	0.62	0.63
AGGL	NMI	0.50	0.57	0.58	0.57
	F1	0.54	0.64	0.60	0.64
IVC	NMI	0.50	0.58	0.56	0.59
	F1	0.54	0.63	0.60	0.64
MCLA	NMI	0.58	0.60	0.63	0.64
	F1	0.71	0.70	0.73	0.75
MCS	NMI	0.57	0.58	0.57	0.60
	F1	0.63	0.68	0.65	0.66
HBGF	NMI	0.50	0.51	0.53	0.54
	F1	0.53	0.60	0.62	0.62
WBPA	NMI	0.45	0.50	0.53	0.56
	F1	0.52	0.53	0.56	0.62

(b)

method	*diversity*	*accuracy*			
		no weights	*SW*	*GW*	*DW*
CSPA	NMI	0.78	0.81	0.75	0.82
	F1	0.82	0.82	0.80	0.82
HGPA	NMI	0.68	0.80	0.77	0.81
	F1	0.72	0.87	0.83	0.85
WSPA	NMI	0.51	0.61	0.72	0.75
	F1	0.60	0.75	0.79	0.79
MV	NMI	0.83	0.84	0.86	0.84
	F1	0.84	0.86	0.87	0.87
AGGL	NMI	0.74	0.74	0.75	0.74
	F1	0.76	0.77	0.78	0.77
IVC	NMI	0.69	0.74	0.82	0.82
	F1	0.75	0.76	0.80	0.82
MCLA	NMI	0.63	0.64	0.70	0.66
	F1	0.69	0.71	0.75	0.75
MCS	NMI	0.58	0.58	0.57	0.60
	F1	0.70	0.68	0.67	0.70
HBGF	NMI	0.72	0.72	0.76	0.73
	F1	0.78	0.77	0.80	0.78
WBPA	NMI	0.56	0.69	0.72	0.73
	F1	0.62	0.78	0.79	0.78

proach prevailing against the remaining ones. In particular, the best average results were achieved by the instance-based methods on LetterRecognition and ControlChart, the hybrid methods on Glass, Ecoli and ISOLET, and the cluster-based methods on ImageSegmentation and Tracedata.

Part IV

The Curse of Dimensionality in Data Clustering

– *Global Dimensionality Reduction* –

8

Time Series Data Management: Background

Abstract A time series is a (large) sequence of (real) numeric values upon which a total order based on timestamps is defined. Detecting similarity between time series data in an effective and efficient way is particularly challenging in a clustering scenario. In this respect, the most used distance measure is *Dynamic Time Warping* (DTW), which allows for fulfilling some critical requirements needed when comparing two time series, while having some issues, such as the poor efficiency. The other significant state-of-the-art similarity measures are based on either refinements of DTW, or well-established string matching techniques. Since time series data typically have large dimensionality, another crucial aspect is the dimensionality reduction, which is commonly performed on time series to satisfy efficiency requirements.

8.1 Time Series Data and Dynamic Time Warping

Definition 8.1 (time series). *A* time series *is a sequence* $X = [(y_1, z_1), \ldots, (y_j, z_m)]$*, where each couple* (y_j, z_j) *is composed by a real numerical value* (y_j) *and an integer* (z_j) *denoting a timestamp, where* z_1 *is assumed to be equal to* 0*.*

As is often the case by assuming a fixed sampling period, X can be simply rewritten as $X = [y_1, \ldots, y_m]$. We hereinafter refer to time series having fixed sampling period.

In a contest of clustering, a crucial aspect is the similarity detection among time series data. A trivial way to quantify the similarity between two equal-length time series $X = [y_1, \ldots, y_m]$ and $X' = [y'_1, \ldots, y'_m]$ is to resort to the squared Euclidean norm:

$$L_2(X, X') = \sum_{j=1}^{m} (y_j - y'_j)^2$$

This approach is used in several early works on time series data management (e.g., [AFS93]), essentially because the Euclidean norm is fast to compute and

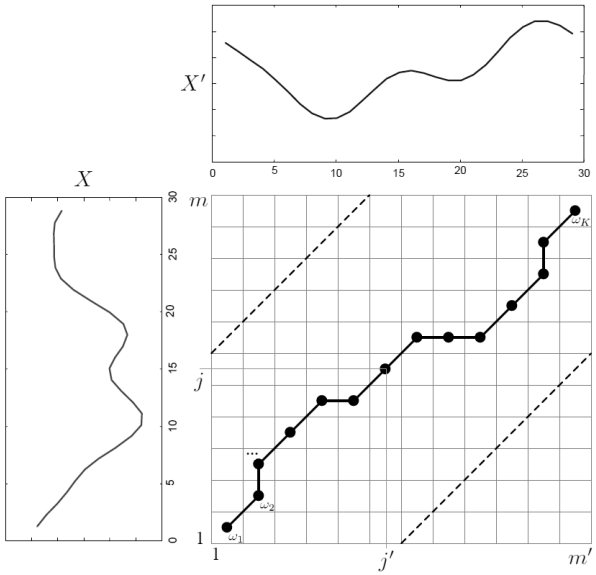
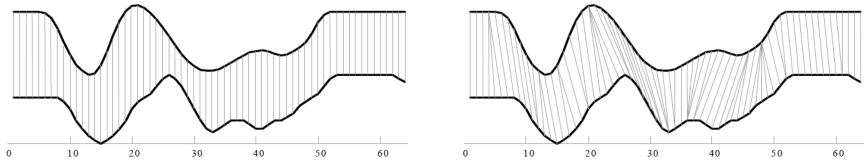


Fig. 8.1. An example warping path



(a) (b)

Fig. 8.2. Two time series aligned according to (a) Euclidean norm and (b) DTW

is a metric. Nevertheless, it has some important drawbacks, i.e., it is unable to deal with noisy time series and time series with different lengths or shifted in the time axis.

A more refined way to compare two time series is "warping" the time axis in order to achieve an alignment between the data points of the series. The *Dynamic Time Warping* (DTW) algorithm has long been known in speech recognition [RJ93], and shown to be an effective solution for measuring the distance between time series [BC94].

Given two time series $X = [y_1, \ldots, y_m]$ and $X' = [y'_1, \ldots, y'_{m'}]$, DTW performs a non-linear mapping of one sequence to another by minimizing the total distance between them. Initially, a $(m \times m')$-matrix Ω is built to contain the squared Euclidean distances between X points and X' points, that is $\Omega_{jj'}$ stores the value $(y_j - y'_{j'})^2$, $\forall j \in [1..m], j' \in [1..m']$. To find the

best alignment between the two time series, a *warping path* (i.e., a sequence of matrix elements) $\omega_1, \omega_2, \ldots, \omega_k, \ldots, \omega_K$ (where $\max\{m, m'\} \leq K \leq m+m'+1$) is computed in such a way that:

1. $\omega_1 = \Omega_{11}$
2. $\omega_K = \Omega_{mm'}$
3. $\omega_k = \Omega_{jj'} \Rightarrow \omega_{k-1} = \Omega_{\hat{j}\hat{j}'}, \quad 0 \leq j - \hat{j} \leq 1, \; 0 \leq j' - \hat{j}' \leq 1, \quad \forall k \in [2..K]$

An example of warping path is depicted in Fig. 8.1, whereas Fig. 8.2 shows the alignments between two example time series according to Euclidean norm and DTW.[1] The DTW distance between X and X' is finally defined as:

$$DTW(X, X') = \min_{\omega_1, \omega_2, \ldots, \omega_k, \ldots, \omega_K} \frac{\sqrt{\sum_{k=1}^{K} \omega_k}}{K} \tag{8.1}$$

Hence, the DTW distance involves the computation of the minimal warping path, which can be retrieved by using a dynamic programming algorithm, working in $\mathcal{O}(m\ m')$.[2]

Unlike Euclidean norm, DTW allows elastic shifting of a sequence to provide a better match with another sequence, thus it can handle time series with local time shifting and different lengths. On the other hand, a major drawback is due to its high computational complexity, which is quadratic with respect to the lengths of the series to be compared.

8.2 State of the Art

In the following, we discuss the state-of-the-art for similarity search/detection and dimensionality reduction in time series data.

8.2.1 Similarity Measures

A major weakness of DTW is that it tends to produce "singularities", i.e., alignments of a single point of a series with multiple points of another series. This phenomenon becomes undesirable when unexpected singularities are produced. An effective variant of DTW, called *Derivative Dynamic Time Warping* (DDTW) [KP01], has been proposed to reduce the singularity phenomenon. Basically, DDTW considers new features in the sequences while maintaining a computational complexity equal to DTW. The novelty of DDTW is that local derivatives of the data points are estimated to capture information on the trends in the sequences and to find a warping more robust to singularities. For instance, two data points having identical values, one with a negative

[1] Figures 8.1 and 8.2 are borrowed from [KP01].

[2] The dynamic programming algorithm exploits the following recursive formula: $\gamma(j, j') = \Omega_{jj'} + \min\{\gamma(j-1, j'-1), \gamma(j-1, j'), \gamma(j, j'-1)\}, \; \forall j \in [1..m], j' \in [1..m']$, where $\gamma(0,0) = \gamma(1,0) = \gamma(0,1) = \infty$, and $\gamma(m, m') = DTW(X, X')$.

slope (i.e., part of a falling trend) and the other one with a positive slope (i.e., part of a rising trend), are correctly not mapped each other when DDTW is used. In a sense, DDTW can be seen as DTW equipped with a preliminary preprocessing step, in which the original data points are replaced with their derivatives.

An alternative, although not computationally more convenient approach to similarity search and detection in time series is based on edit distance-like string matching measures. The *Longest Common SubSequence* (LCSS) algorithm [VGK02] is a variant of the edit distance that uses the length of the longest common subsequence of two sequences to define the distance between them. LCSS can deal with noisy time series by performing approximate matching rather than exact matching of time series, although it suffers from large-grained similarity. *Edit Distance with Real sequences* (EDR) [CÖO05] performs the same distance quantization of LCSS (which is parametric with respect to a tolerance threshold) in order to remove noisy effects. Unlike LCSS and EDR, *Edit distance with Real Penalty* (ERP) [CN04b] is a metric and still supports local time shifting. ERP can be seen as a variant of EDR and DTW, although it does not require a noise-tolerance threshold like EDR, and does not replicate previous data points to add a gap like DTW.

8.2.2 Dimensionality Reduction Techniques

To address the high dimensionality issue in time series, there are mainly two basic approaches: approximating a time series by a piecewise discontinuous function or applying a low-order continuous function to a time series.

The first approach includes *Discrete Wavelet Transform* (DWT) [CF99, WAA00], *Swinging Door* (SD) [Bri90], *Piecewise Linear Approximation* (PLA) [PH74, KP98], *Piecewise Aggregate Approximation* (PAA) [KCPM01, KP00, YF00], *Adaptive Piecewise Constant Approximation* (APCA) [CKMP02], and *Symbolic Aggregate approXimation* (SAX) [LKLC03]. Using DWT, a time series is represented in terms of a finite length, fast decaying, oscillating and discretely sampled waveform (mother wavelet), which is scaled and translated in order to create an orthonormal wavelet basis. Each function in the wavelet basis is related to a real coefficient: the original series is reconstructed by computing the weighted sum of all the functions in the basis, using the corresponding coefficient as weight. The Haar basis [BGG97] is the most widely used in wavelet transformation. The DWT representation of a time series of length m consists in identifying m wavelet coefficients, whereas a dimensionality reduction is achieved by maintaining only the first d coefficients (with $d \ll m$).

SD is a data compression technique that belongs to the family of piecewise linear trending functions. Recently, SD has been adopted in several PI data analysis scenarios (e.g., [TCS04]). Also, in [ICS07], SD has been compared to wavelet compression. The SD algorithm employs a heuristic to decide whether

a value is to be stored within the segment being grown or it is to be the beginning of a new segment. Given a pivot point, which indicates the beginning of a segment, two lines (the "doors") are drawn from it to envelop all the points up to the next one to be considered. The envelop has the form of a triangle according to a parameter that specifies the initial amplitude of the lines. The setup of this parameter has impact on the data compression level.

In the PLA method, a time series is represented by a piecewise linear function, i.e., a set of line segments. Several methods have been proposed to recognize PLA segments (e.g., [PH74, KP98]); among these methods, the most efficient ones are able to produce a PLA representation with computational complexity linear with the length of the time series.

PAA transforms a time series of m points in a new one composed by d segments (with $d \ll m$), each of which is of size equal to m/d and is represented by the mean value of the data points falling within the segment. Like PAA, APCA approximates a time series by a sequence of segments, each one represented by the mean value of its data points. A major difference from PAA is that APCA can identify segments of variable length. Also, the APCA algorithm is able to produce high-quality approximations of a time series by resorting to solutions adopted in the wavelet domain.

The SAX representation of a time series involves three steps. Initially, the PAA version of a time series is computed, then the PAA coefficients are quantized, and finally each quantization level is represented by a single character, called SAX symbol.

It should be noted that representing a time series of m points according to DWT, SD, (the fastest versions of) PLA, PAA, and SAX can be performed in $\mathcal{O}(m)$, whereas the complexity of APCA is $\mathcal{O}(m \log(m))$.

Dimensionality reduction techniques based on piecewise discontinuous approximations can be combined with existing similarity measures, in order to decrease the computational cost in similarity searches. In particular, the use of DTW on the coefficients obtained by segmenting a time series has been investigated in the literature (e.g., [KP00]), and several lower bounding measures operating on segmented versions of a time series have been defined. Among these methods, the *Fast search method for dynamic Time Warping* (FTW) [SYF05] has been proposed as one of the most effective methods that use the time warping distance on a coarse version of the original sequences.

The other approach to dimensionality reduction, which approximates a time series with a continuous polynomial, includes *Singular Value Decomposition* (SVD) [KJF97, KAS98], *Discrete Fourier Transforms* (DFT) [RM98, RM97], and *Chebyshev polynomials* [CN04a, MH03]. SVD consists of space rotation and truncation applied on a data matrix and is computationally more expensive than all the other discussed methods for dimensionality reduction. DFT and Chebyshev approaches are quite close to DWT: they are based on the use of a set of orthonormal functions, whose contributions to the whole representation are given by the relating coefficients. Major differences among these representations regard the functions that compose the orthonormal ba-

sis (i.e., sine waves for DFT, and Chebyshev polynomials for Chebyshev) and the computational cost (i.e., $\mathcal{O}(m \log m)$ for DFT, and $\mathcal{O}(m)$ for Chebyshev). Also, Chebyshev approximation is very close to the optimal minimax polynomial, which represents an approximation able to minimize the maximum deviation from the original data points.

9

Fast and Accurate Similarity Detection in Time Series Data

Abstract Similarity search and detection is a central problem in time series data processing and management. Most approaches to this problem have been developed around the notion of dynamic time warping, whereas several dimensionality reduction techniques have been proposed to improve the efficiency of similarity searches. Due to the continuous increasing of sources of time series data and the cruciality of real-world applications that use such data, there is a challenging demand for supporting similarity detection in time series in a both accurate and fast way. This chapter proposes a concise yet feature-rich representation of time series, on which the dynamic time warping can be applied for effective and efficient similarity detection of time series. We present the *Derivative time series Segment Approximation (DSA)* representation model, which originally features derivative estimation, segmentation and segment approximation to provide both high sensitivity in capturing the main trends of time series and data compression. We extensively compare DSA with state-of-the-art similarity methods and dimensionality reduction techniques in a clustering framework. Experimental evidence from effectiveness and efficiency tests on various datasets shows that DSA is well-suited to support both accurate and fast similarity detection.

9.1 Introduction

Most research on time series data management and knowledge discovery has been devoted to the similarity search and detection problem, which arises in many tasks such as indexing and query processing, change detection, frequent pattern mining, classification, and clustering. In particular, clustering of time series data has been attracting a growing interest in several scenarios. For instance, in the biomedical domain, frequently posed problems include finding groups of genes with similar expression profiles across a number of experiments, organizing patients according to different healthy/disease conditions, and finding groups of similar functional activities of the human brain in response to a given stimulus. In the socio-economics domain, clustering energy/power consumption patterns can support applications of fraud detec-

tion. Other challenging scenarios involve, for instance, seasonality patterns of retail data, personal income data, models of ecological dynamics, multimedia data streams. A more exhaustive list of applications which demand for time series clustering can be found in [Lia05].

The common approach to compare two time series is the Dynamic Time Warping (DTW) algorithm, which allows to "warp" the time axis in order to achieve an alignment between the data points of the series (cf. Chapt. 8).

Besides the similarity problem in time series, another issue concerns the high dimensionality that characterizes time series data in many application domains. To address this issue, various dimensionality reduction techniques have been proposed, following two main approaches in which a (continuous) time series is approximated with either a piecewise discontinuous function or a low-order continuous function (cf. Chapt. 8).

Dimensionality reduction methods are useful for modeling time series into a more compact form. However, while this can help to compare time series efficiently, dimensionality reduction methods may lose significant information about the main trends in a time series, which are essential to effective similarity detection. Indeed, in many real-world applications there is a growing interest in developing methods that are able to fit an emerging demand for both *accurate and fast similarity detection*. In this respect, we believe there is a number of special requirements that should be satisfied by any representation model to support accurate and fast similarity detection in time series, which are summarized as follows:

- *Time warping-awareness.* Time series should be modeled into a form that can be naturally mapped to the time domain. This will make it feasible to benefit from using dynamic time warping for similarity detection.
- *Low complexity.* Due to the high dimensionality of time series data, modeling time series should be performed maintaining a reasonably low complexity, which is possibly linear with the series length.
- *Sensitivity to relevant features.* It is clearly desirable that time series approximation is able to preserve as much information in the original series as possible. For this purpose, approximating a time series should be accomplished in such a way that it tailors itself to the local features of the series, in order to capture the important trends of the series.
- *Absence of parameters.* Most representation models and dimensionality reduction methods require the user to specify some input parameters, such as, e.g., the number of coefficients or symbols. However, prior domain knowledge is often unavailable, and the sensitivity to input parameters can seriously affect the accuracy of the representation model or dimensionality reduction method.

In this chapter, we present a time series representation model which is conceived to support accurate and fast similarity detection. This model is called *Derivative time series Segment Approximation (DSA)* [GRT06, GPTG07, GPTG09a], as it yields a concise yet feature-rich time series representation

by combining the notions of *derivative estimation*, *segmentation* and *segment approximation*.

DSA involves a segmentation scheme that employs the paradigm based on a piecewise discontinuous function. However, in contrast to any other technique of dimensionality reduction, the segmentation step is performed on the derivative version of the original time series, rather than directly on the raw time series. The derivative estimates represent a new feature space that enables the identification of the trends of the original series. Moreover, the final step of segment modeling allows for concisely fitting the detected trends in a low-dimensional, time warping-aware representation of the original time series. As we proved experimentally, the intuition underlying the DSA model works out very advantageously in supporting accurate and fast similarity detection; indeed, DSA is able to fulfill all of the desiderata mentioned above:

- DSA sequences can be compared by using DTW directly;
- the derivative-based feature generation allows for representing a time series by focusing on the characteristic trends in the series;
- the segmentation step in DSA has a computational complexity which is linear with the series length, and is adaptive with respect to the identified trends of the series;
- the absence of mandatory input parameters in DSA addresses the unavailability of prior domain knowledge.

We conducted an extensive experimental evaluation of DSA within a clustering framework, by considering aspects of effectiveness as well as efficiency. This evaluation necessarily involved the prominent state-of-the-art methods for time series representation and dimensionality reduction. Experimental evidence has shown that DSA supports accurate and fast similarity detection, in terms of a number of results.

9.2 Derivative time series Segment Approximation (DSA)

In this section, we describe the proposed *Derivative time series Segment Approximation (DSA)* model to represent time series into a concise form which is designed to capture the significant variations in the time series profile. More precisely, a DSA sequence is the result of a transformation that applies to a time series and yields a shorter sequence of values approximating the segments identified in the derivative version of the original series. DSA entails derivative estimation, segmentation and segment modeling to map a time series into a different value domain which allows for maintaining information on the significant features of the original series in a dense and concise way.

Given a time series of length m, DSA computes a new sequence χ of d values, with $d \ll m$, by three main steps:

1. Derivative estimation — the original time series is transformed into a new one in which each point is replaced with its first derivative estimate.
2. Segmentation — the derivative time series is decomposed into variable-length segments, each of which is comprised of a subsequence of points having close slopes.
3. Segment approximation — the individual segments are substantially mapped to angular values, which represent synthetic information on the average slopes within the segments.

9.2.1 Derivative Estimation

Given a time series $X = [y_1, \ldots, y_m]$, the derivative estimation step yields a sequence $\dot{X} = [\dot{y}_1, \ldots, \dot{y}_m]$, whose elements are first derivative estimates of the points in X.

A simple yet effective derivative estimation model is that exploited in [KP01]—we hereinafter refer to it as DDTW estimation model—which computes, for each point (except the first and the last one in the series), the mean value between the slope of the line from the left neighbor to the point and the slope of the line from the left neighbor to the right neighbor. Formally:

$$\dot{y}_j = \begin{cases} \dot{y}_{j+1} & \text{if } j = 1 \\ \frac{1}{2}[(y_j - y_{j-1}) + \frac{1}{2}(y_{j+1} - y_{j-1})] & \text{if } j \in [2..m\text{-}1] \\ \dot{y}_{j-1} & \text{if } j = m \end{cases} \tag{9.1}$$

We slightly modify the DDTW estimation model by also considering the slope of the line from the point to the right neighbor; actually, this modification leads to an algebraic simplification producing an expression that is equivalent to consider only the slope of the line from the left neighbor to the right neighbor. The derivatives of the first and the last point in the series are computed by taking into account their respective neighbors as well. Formally:

$$\dot{y}_j = \begin{cases} y_{j+1} - y_j & \text{if } j = 1 \\ \frac{1}{2}(y_{j+1} - y_{j-1}) & \text{if } j \in [2..m\text{-}1] \\ y_j - y_{j-1} & \text{if } j = m \end{cases} \tag{9.2}$$

We investigated how the performances of DSA and DDTW may vary depending on the derivative estimation model. As we describe in Appendix B, the DSA derivative estimation model reported in (9.2) leads to a better derivative-based feature space than the DDTW derivative estimation model defined in (9.1).

9.2.2 Segmentation

Segmenting a time series of length m consists in identifying $d-1$ delimiter points ($d \ll m$) to partition the series into d contiguous subsequences of points having similar features.

In our approach, segmentation is computed on the derivative version of a time series. Precisely, a derivative time series $\dot{X} = [\dot{y}_1, \ldots, \dot{y}_m]$ is transformed into a sequence $S_{\dot{X}} = [s_1, \ldots, s_d]$ of variable-length segments of the form $s_l = s_{l,1}, \ldots, s_{l,\kappa_l}] = [\dot{y}_{l_1}, \ldots, \dot{y}_{l_{\kappa_l}}]$, such that:

- $s_{1,1} = \dot{y}_1$ and $s_{d,\kappa_d} = \dot{y}_m$, and
- for each $l \in [1..d\text{-}1]$, s_{l,κ_l} immediately precedes $s_{l+1,1}$ in the time axis.

A critical aspect in segmentation is how to determine the segment delimiters. For this purpose, we follow a sliding windows approach, i.e., a segment is grown until it exceeds an error threshold, and the process continues from the next point not yet considered. Although more refined segmentation schemes could be devised (e.g., top-down or bottom-up schemes) [KCHP01], in this work we chose to pursue the above idea for the sake of its simplicity.

The key idea in our segmentation method is to break a series according to the first point such that the absolute difference between it and the mean of the previous points is above a certain threshold ϵ; this point becomes the anchor for the next segment to be identified in the rest of the series. Formally, let $\mu(s_l)$ denote the mean of the points in a sequence s_l of $S_{\dot{X}}$:

$$\mu(s_l) = \frac{\sum_{h=1}^{\kappa_l} \dot{y}_{l_h}}{\kappa_l}, \quad \text{for each } l \in [1..d\text{-}1]$$

The sequence s_l is identified as a segment if and only if

$$|\mu([s_{l,1}, \ldots, s_{l,h}]) - s_{l,h+1}| \leq \epsilon, \ \forall h \in [1..\kappa_l - 1]$$

and

$$|\mu([s_{l,1}, \ldots, s_{l,\kappa_l}]) - s_{l+1,1}| > \epsilon$$

Intuitively, this condition allows for aggregating subsequent data points having very close derivatives; in this way, the new segment s_l represents a subsequence of points with a specific trend.

To estimate the threshold ϵ, we resort to an index of spread of the (derivative) data points within a sequence. The objective is to produce a number of segments that is large enough to capture the "characteristic" trends in the original series (i.e., subsequences of points having close derivative estimates), but small enough to guarantee a reasonably good degree of compression. Within this view, we initially devised three definitions of ϵ, namely *dataset-oriented*, *series-oriented*, or *segment-oriented*.

The dataset-oriented definition of ϵ aims to express this threshold according to a given collection of time series. Given a dataset D of n time series, ϵ can be defined as:

$$\epsilon(D) = \frac{1}{n} \sum_{i=1}^{n} \frac{|\dot{X}_i|}{\max\{|\dot{X}| \mid X \in D\}} \sigma(\dot{X}_i)$$

where $\sigma(\dot{X}_i)$ denotes the standard deviation over the points in the i-th derivative series, and normalization of the series lengths is provided to deal with variable-length series.

The above definition is reasonably adequate when most time series in the collection show similar shapes; however, this may not necessarily hold in several real domains (e.g., sensor network measurements). A different way of computing ϵ can be series-oriented, i.e., globally to each individual time series X:

$$\epsilon(\dot{X}) = \sigma(\dot{X})$$

Another definition of ϵ may involve the individual segments being identified in each series. We can hence define a segment-oriented ϵ for each segment s_l as follows:

$$\epsilon(s_l) = \sigma(s_l)$$

It is easy to observe that, regardless the definition of ϵ, the segmentation step on a dataset of n series can be performed in $\mathcal{O}(n \times m_{max})$, where m_{max} is the maximum of the series lengths. Since the segment-oriented definition is tailored to the local features of an individual series, we chose to adopt it in the segmentation step of our DSA model.

It is worth noting that the segmentation step in DSA does not require any user-specified parameter, since the threshold ϵ is automatically computed by analyzing information of each series. By contrast, this step is not automatic for other methods of dimensionality reduction, e.g., APCA, SAX, SD, PAA, PLA, Chebyshev, DWT, and DFT (cf. Chapt. 8), where the user is required to specify an input, such as the number of segments or coefficients being computed.

9.2.3 Segment Approximation

The individual segments of a derivative time series are represented with a synthetic information capturing their respective main features. More precisely, each segment s_l is mapped to a pair formed by the value z_l+1, where z_l is the timestamp of the last point ($\dot{y}_{l_{\kappa_l}}$) in s_l, and an angle that explains the average slope of the portion of time series bounded by s_l. This is mathematically expressed by the notion of arctangent applied to the mean of the (derivative) points in each segment.

Given a segmented derivative time series $S_{\dot{X}} = [s_1, \ldots, s_d]$, the final step of segment approximation yields a sequence $\chi = [(\beta_1, t_1), \ldots, (\beta_d, t_d)]$ such that

$$\begin{aligned} \beta_l &= \arctan(\mu(s_l)), \; l \in [1..d] \\ t_l &= t_{l-1} + \kappa_l, \qquad l \in [1..d] \end{aligned}$$

where we assume $t_0 = 0$ for any DSA sequence.

Modeling a given time series by means of the DSA representation hence leads to a new sequence whose elements (pairs angle-timestamp) still maintain

a direct association to the original time domain, while concisely representing the features of original points. This makes the DSA model able to fully support dynamic time warping, i.e., (dis)similarities between DSA sequences can be computed by using DTW-based measures.

As a final remark, it is easy to observe that the time complexity of computing a DSA sequence from a time series of length m has a total cost $\mathcal{O}(m)$, since the three steps, namely differentiation, segmentation, and segment approximation, are $\mathcal{O}(m)$, $\mathcal{O}(m)$, and $\mathcal{O}(d)$ ($d \ll m$), respectively.

9.3 Experimental Methodology

We devised an experimental evaluation to assess the ability of the proposed DSA in supporting effective and efficient similarity detection within a clustering framework. We compared DSA against state-of-the-art methods for modeling and comparing time series data, which include LCSS, EDR, ERP, DTW, DDTW, and FTW as distance measures, and APCA, SAX, PAA, PLA, SD, Chebyshev, DWT, and DFT as dimensionality reduction methods (cf. Chapt. 8). Since DSA and the competing dimensionality reduction methods are not similarity/distance measures, we chose to apply DTW over the segments/coefficients computed by each particular representation scheme in the time domain (i.e., APCA, SAX, PAA, PLA, SD, and DSA), whereas we used the Euclidean distance (L_2) to compare the sequences obtained by Chebyshev, DWT, and DFT, as suggested in their respective works.

Before going into the details of the experimental results, in this section we introduce the selected datasets, the clustering algorithms, and the validity criteria used in the experimental evaluation. We also discuss the preliminary task of preprocessing of the raw time series and the setup of the various methods that compete with our DSA.

9.3.1 Datasets

We selected six benchmark time series datasets, namely GunX, Trace-data, CBF, ControlChart, Twopat, and Mixed-BagShapes. In addition, we also used OvarianCancer, which contains proteomic spectra generated by *Surface-Enhanced Laser Desorption and Ionization - Time Of Flight Mass Spectrometry* (SELDI-TOF MS). Figure 9.1 shows the shapes of sample representative instances in each dataset, whereas Appendix A reports on a description of the selected datasets.

It should be emphasized that OvarianCancer data, like most of MS datasets, are huge-dimensional and largely affected by noisy factors. Noise is typically due to a number of reasons, such as sample preparation, insertion of the samples into the mass spectrometer, and instrumental and measurement errors. For this purpose, OvarianCancer spectra were subject to a preliminary preprocessing phase specific for MS data. MS preprocessing has been recognized

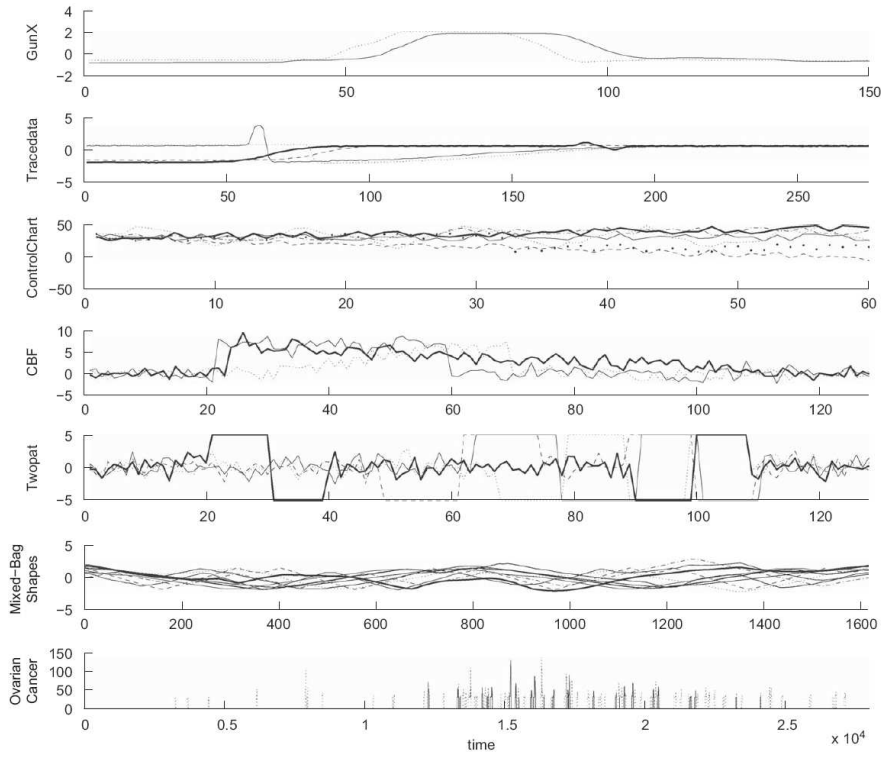


Fig. 9.1. Sample instances from the test datasets used for evaluating DSA performance. One time series from each class is displayed for each dataset

as a crucial step for tasks of MS data management and knowledge discovery, and mainly consists of operations such as noise reduction, baseline subtraction, and peak detection. The interested reader can find details about the preprocessing steps carried out in [GPT+08b].

It is interesting to take a look at the impact on the time series dimensionality by using DSA. Table 9.1 shows that DSA achieves a 59% compression of the original series lengths on average, with a maximum compression percentage of 97% in the OvarianCancer dataset. As we shall discuss later in this section, the reasonably good rate of compression achieved by DSA does not have a negative impact on the accuracy in detecting similarities.

9.3.2 Cluster Validity

To assess the effectiveness of clustering algorithms, we assess how well a clustering solution fits a given scheme of known classes, thanks to the availability of reference classifications for all the test datasets. In particular, we resort to the F1-Measure as defined in Def. 2.5.

Table 9.1. Segmentation and compression using DSA

dataset	*# of time steps*	*avg. # of DSA segments*	*compression*
GunX	150	68	55%
Tracedata	275	118	57%
ControlChart	60	35	42%
CBF	128	77	40%
Twopat	128	38	70%
Mixed-BagShapes	1,614	816	49%
OvarianCancer	28,000	943	97%

9.3.3 Algorithms

Finding the best strategy of time series clustering is not an objective of this work; rather, we are interested in assessing the impact of the proposed time series representation model in similarity detection, and hence we conceived a standard clustering framework for time series data. Specifically, we resorted to well-known paradigms, namely partitional clustering and agglomerative hierarchical clustering (cf. Chapt. 2). As stated in [Lia05], partitional and hierarchical clustering methods have been extensively used in the context of time series clustering.

Partitional Clustering

The partitional clustering paradigm is characterized by simplicity and low computational and memory requirements. In this work, we use the popular K-Means algorithm (cf. Chapt. 2). It is worth noting that choosing the number of output clusters does not represent a drawback in our evaluation context, since we selected datasets for which reference classifications are available, and hence we were able to fix the number of clusters equal to the actual number of classes in each clustering experiment. Also, we addressed the random selection of the initial cluster centroids by performing multiple runs of the K-Means algorithm to avoid that the quality results were due to random chance.

In order to define the cluster centroids, we adopted two strategies depending on whether or not the representation model produces variable-length segments. Concerning SAX, PAA, PLA, SD, Chebyshev, DWT, and DFT, we compute the cluster representatives by simply averaging the corresponding coefficients over the time series in any specific cluster. In the following, we present a method for computing cluster representatives of DSA sequences; we remark that although this method has been originally conceived for DSA clusters, it can be easily adapted to any representation model that is able to produce variable-length segments (coefficients), such as APCA.

Computing Cluster Representatives of DSA sequences in K-Means

Let us denote with $C_\chi = \{\chi_1, \ldots, \chi_n\}$ a cluster of DSA sequences, where each χ_i has the form $[(\beta_{i_1}, t_{i_1}), \ldots, (\beta_{i_{d_i}}, t_{i_{d_i}})]$, and with $C = \{X_1, \ldots, X_n\}$

the cluster of original time series such that each $X_i \in C$ is associated with a unique $\chi_i \in C_\chi$. The objective is to compute a DSA sequence prototype $rep(C_\chi)$ as the representative of cluster C.

We identify a fixed number V of segments over which the average series is defined. We can reasonably define the number of segments V as the closest integer to the mean $(\sum_{i=1}^{n} d_i)/n$ over all the series $\chi_i \in C_\chi$. The timestamps associated to the new V segments are defined as $\hat{t}_v = t_{max} \times v/V$, for each $v \in [1..V]$, where $t_{max} = \max\{t_{i_{d_i}} \mid \chi_i \in C_\chi\}$ $(i \in [1..n])$ and $\hat{t}_0 = 0$. For each χ_i, the angle β'_{i_v} corresponding to the timestamp $\hat{t}_v$ (with $\hat{t}_v \leq t_{i_{d_i}}$) is computed to be equal to the angle β_{i_u} of the u-th segment including the point sampled at time $\hat{t}_v$. Formally, β'_{i_v} is equal to the angle β_{i_u} such that $(\beta_{i_u}, t_{i_u}) \in \chi_i$ and $t_{i_{u-1}} < \hat{t}_v \leq t_{i_u}$, for all $i \in [1..n], v \in [1..V]$. Note that any pair $(\beta'_{i_v}, \hat{t}_v)$ is introduced only if the condition $\hat{t}_v \leq t_{i_{d_i}}$ holds, i.e., if the i-th time series is defined in the timestamp $\hat{t}_v$.

For each χ_i the new V_i pairs $(\beta'_{i_v}, \hat{t}_v)$ are then included in the rewritten DSA sequence $\chi''_i = [(\beta''_{i_1}, t''_{i_1}), \ldots, (\beta''_{i_{q_i}}, t''_{i_{q_i}})]$ which is computed as

$$\textit{time-sort}\{(\beta_{i_1}, t_{i_1}), \ldots, (\beta_{i_{d_i}}, t_{i_{d_i}}), (\beta'_{i_1}, \hat{t}_1), \ldots, (\beta'_{i_{V_i}}, \hat{t}_{V_i})\}$$

where $\hat{t}_{V_i}$ is the new timestamp with maximum value defined over the i-th sequence.

Formally, for each $\chi_i \in C_\chi$, the sequence $\hat{\chi}_i = [(\hat{\beta}_{i_1}, \hat{t}_1), \ldots, (\hat{\beta}_{i_V}, \hat{t}_{V_i})]$ is computed, where

$$\hat{\beta}_{i_v} = \frac{\sum_{(\beta''_{i_u}, t''_{i_u}) \in \chi''_i \,\wedge\, t''_{i_u} \in (\hat{t}_{v-1}, \hat{t}_v]} [\ \beta''_{i_u} \times (t''_{i_u} - t''_{i_{u-1}})\ ]}{\hat{t}_v - \hat{t}_{v-1}}, \quad v \in [1..V_i]$$

The DSA representative $rep(C_\chi)$ is finally computed as:

$$rep(C_\chi) = [(\overline{\beta}_1, \hat{t}_1), \ldots, (\overline{\beta}_V, \hat{t}_V)], \quad \text{where} \quad \overline{\beta}_v = \frac{\sum_{\hat{t}_v \leq \hat{t}_{V_i} \wedge i \in [1..n]} \hat{\beta}_{i_v}}{|\{\hat{t}_v | \hat{t}_v \leq \hat{t}_{V_i}\}|}$$

for each $v \in [1..V]$. Note that $\Delta t = \hat{t}_v - \hat{t}_{v-1}$ is a constant for each $v \in [1..V]$.

Hierarchical Clustering

The agglomerative hierarchical clustering paradigm allows us to test the competing methods in a clustering framework which does not rely on a notion of cluster prototype and on the cluster initialization. For this purpose, we use the UPGMA algorithm (Unweighted Pair Group Method using arithmetic Averages), which exploits the standard AHC scheme, along with an average link metric (cf. Chapt. 2).

9.3.4 Preprocessing Time Series

Raw time series are usually preprocessed by *smoothing* data points in order to reduce the noise in the data. Moving average represents the simplest family of smoothing models, as it is a compromise between the mean and the random walk model. Given a raw time series $X = [y_1, \ldots, y_m]$ and a smoothing degree λ (i.e., the maximum width of the moving average), the *centered λ-point moving average* recomputes the data points by considering both the previous and next observations around a center:

$$y_j^{smoothed} = \begin{cases} \mu([y_1, \ldots, y_{j+\varrho}]) & \text{if } j-\varrho \leq 0 \\ \mu([y_{j-\varrho}, \ldots, y_{j+\varrho}]) & \text{if } j-\varrho > 0 \text{ and } j+\varrho \leq m \\ \mu([y_{j-\varrho}, \ldots, y_m]) & \text{if } j+\varrho > m \end{cases}$$

where $\varrho = (\lambda-1)/2$ denotes the maximum number of back and forward points that are taken into account for smoothing the j-th point.

More refined models, such as exponential smoothing models, compute the weighted average of past observations on the basis of previously smoothed observations. Given a smoothing factor $\varphi \in [0, 1]$, the simple *exponential smoothing* is computed as:

$$y_j^{smoothed} = \begin{cases} y_j & \text{if } j = 1 \\ \varphi\ y_j + (1-\varphi)\ y_{j-1}^{smoothed} & \text{if } j > 1 \end{cases}$$

It should be emphasized that denoising is essential to make the data amenable to further analysis tasks, regardless of the specific representation method or distance measure used. In particular, in derivative-based feature spaces, denoising time series data (e.g., via a smoothing function) before differentiating them is necessary to avoid that the approximation of derivatives by finite differences will amplify the noise present in the data. In this respect, the combination of smoothing prior to the step of derivative estimation in our DSA approach (as well as in DDTW) can be seen as somehow similar to the regularization of a differentiation process [TA77], although potentially less accurate and general.

9.3.5 Settings

Unlike our DSA, most of the competing methods require one or more parameters to be set. In some cases, which include LCSS, EDR, ERP, and Chebyshev, typical settings are suggested in their respective works; specifically, the matching thresholds for LCSS and EDR are assumed to be equal to $(\max \sigma(X_i))/4$ and $\min \sigma(X_i)$ respectively (for all the series X_i in a dataset), the constant gap for ERP is set to 0, and the number of coefficients for Chebyshev is set to 20. Such parameter settings revealed to be good enough to enable the respective methods to achieve their best performances in accuracy. In particular, we took care in monitoring the behavior of the Chebyshev method,

and finally found no significant improvement in accuracy by increasing the number of Chebyshev coefficients.

In other cases, to make a comparative evaluation possible in terms of accuracy and efficiency, we aimed to prepare the various methods to perform at levels of data compression which were as close as possible. We tried several values for the parameters of APCA, SAX, PAA, PLA, DWT, DFT, and FTW. More precisely, for each dataset and algorithm, we varied the setting of each of these methods in such a way that it achieved the same compression (i.e., number of segments) obtained by DSA, and ±5%, ±10%, and ±20% of the DSA compression; then, we measured the relative clustering quality (F1-Measure) scores and finally chosen the setting corresponding to the best score. Analogously, the alphabet length (i.e., the number of symbols) required by SAX was chosen, for each dataset and clustering algorithm, as the value that led to the best trade-off between clustering quality and time performance.

A final remark concerns SD, which requires a deviation threshold (i.e., the "doors" amplitude); however, setting this parameter is even more difficult, since the compression factor (i.e., the number of segments) cannot be specified directly in swinging door compression. In [TCS04, XCCH02], many calibration trials are conducted to find the deviation thresholds corresponding to a given compression factor, for each dataset. We followed this approach and set the deviation threshold in such a way that the number of segments produced by SD was as close as possible to the number of segments produced by DSA, finally choosing the value that led to the best clustering quality.

9.4 Experimental Results

9.4.1 Effectiveness Evaluation

We measured the ability of DSA and the competing methods in supporting time series clustering effectively. We investigated how clustering results can be influenced by choosing different alternatives for data preprocessing and setting the parameters therein involved; then, we assessed the performance of DSA and the other methods according to their respective best settings.

Tuning Preprocessing Parameters

We used the smoothing functions previously described in Sect 9.3.4 to preprocess the time series in each dataset. In the case of centered moving average, the smoothing degree $\lambda(= 2\varrho + 1)$ was varied within a typical range, namely [5..9], whereas φ in exponential smoothing settings was varied from 0 to 1 by a 0.1 step. Moreover, we tried to perform zero, one or more iterations of smoothing (up to 5), on the various datasets and for each preprocessing scheme; the rationale here is that smoothing should be avoided to prevent loss of information for low-noise series and, conversely, excessive noise might

Table 9.2. DSA vs. competing methods: avg quality results (F1-Measure) for K-Means

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed-BagSh.	Ovarian Cancer
LCSS	0.59	0.30	0.50	0.79	0.36	0.32	0.34
EDR	0.54	0.74	0.88	0.86	0.42	0.70	0.58
ERP	0.72	0.62	0.76	0.58	0.39	0.48	0.34
DTW	0.66	0.78	0.87	0.89	*0.95*	*0.77*	0.60
DDTW	*0.89*	*1*	*0.89*	*0.97*	*0.95*	0.76	*0.62*
FTW	0.74	0.90	0.81	0.67	0.55	0.73	0.58
L_2 on DFT	0.63	0.77	0.78	0.67	0.39	0.70	0.36
L_2 on DWT	0.61	0.67	0.76	0.74	0.36	0.68	0.36
L_2 on CHEBY	0.57	0.72	0.70	0.69	0.38	0.72	0.34
DTW on SD	0.67	0.95	0.85	0.87	0.89	0.76	0.69
DTW on PLA	0.73	0.77	*0.89*	0.87	0.75	0.74	0.60
DTW on PAA	0.68	0.78	0.87	0.86	0.73	0.75	0.59
DTW on SAX	0.73	0.77	0.83	0.87	0.69	0.71	0.58
DTW on APCA	0.77	0.81	*0.89*	0.83	0.91	0.74	0.55
DTW on DSA	**0.92**	**1**	**0.90**	**0.96**	**0.97**	**0.78**	**0.75**

be treated with multiple smoothing. It should be noted that, in the case of K-Means evaluation, we performed multiple runs of the K-Means algorithm and finally averaged the quality results over the runs to obtain a single value of F1-Measure.

We observed that smoothing helped to improve the performance of the various methods on all the datasets, except OvarianCancer; OvarianCancer represented an exception since, in this case, data was preliminarily subject to domain-specific preprocessing steps (cf. Sect. 9.3.1), hence further preprocessing via smoothing would have tended to cause loss of information on potentially significant data features.

Exponential smoothing revealed to be more effective than moving average, as it was selected 74 out of 90 times as the best preprocessing way. Parameter φ was set to low values in most cases, thus suggesting the need for a greater smoothing effect (which is indeed achieved by values of φ closer to zero). Also, the number of smoothing iterations appeared to be not relevant in practice; three iterations of smoothing were enough in most cases, except for SD which always required four or five iterations. In Appendix C, we report further details about the preprocessing stage, including the best settings and an evaluation of the impact of smoothing on the various datasets.

Table 9.3 reports on the quality results obtained by the UPGMA algorithm. A first remark on these results is that the F1-Measure scores for the various methods were generally much lower than the corresponding results obtained by the K-Means algorithm on all the datasets, except OvarianCancer; in particular, for DTW on DSA, there was a quality decrease from 19% (in GunX) to 36% (in CBF and ControlChart). However, like in K-Means clustering, DTW on DSA revealed to behave as good as or better than the other methods and, in some cases (i.e., CBF and Mixed-BagShapes) we observed quality improvements with respect to the corresponding results by K-Means.

Table 9.3. DSA vs. competing methods: quality results (F1-Measure) for UPGMA

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed-BagSh.	Ovarian Cancer
LCSS	0.47	0.23	0.38	0.39	0.24	0.24	0.33
EDR	0.49	0.38	0.41	0.47	0.28	0.28	0.56
ERP	0.49	0.37	0.40	0.40	0.33	0.28	0.54
DTW	0.61	0.48	0.48	0.51	0.61	0.38	0.61
DDTW	*0.72*	*0.76*	*0.54*	0.49	*0.64*	0.41	*0.63*
FTW	0.60	0.52	0.44	0.44	0.50	0.40	*0.63*
L_2 on DFT	0.60	0.40	0.29	0.47	0.39	0.28	0.62
L_2 on DWT	0.60	0.40	0.29	0.47	0.39	0.28	0.62
L_2 on CHEBY	0.60	0.40	0.29	0.47	0.39	0.28	0.62
DTW on SD	0.51	0.55	0.40	0.49	0.43	*0.42*	0.60
DTW on PLA	0.61	0.63	0.40	0.51	0.43	0.29	0.61
DTW on PAA	0.61	0.61	0.36	0.51	0.56	0.41	0.61
DTW on SAX	0.61	0.60	0.48	*0.56*	0.44	0.31	0.61
DTW on APCA	0.61	0.63	0.40	0.51	0.43	0.29	0.61
DTW on DSA	**0.73**	**0.82**	**0.54**	**0.60**	**0.67**	**0.51**	**0.73**

Accuracy in Time Series Clustering

We evaluated DSA and the other methods in two clustering frameworks, namely K-Means and UPGMA. In relation to the selected clustering algorithm, each method was used with the best preprocessing setup for the specific dataset. In any case, the quality of the obtained clustering solutions was calculated in terms of F1-Measure.

Table 9.2 refers to K-Means clustering and shows the quality results averaged over 100 runs of this algorithm. Looking at the table we can see that DTW on DSA sequences (for short, DTW on DSA) was the first ranked method in all the datasets except CBF; however, in this dataset, DTW on DSA was only 1% below the performance of DDTW, which turned out to be the best method among the competing ones. Also, DTW on DSA always led to better results than DTW alone. It should be emphasized that the comparison with DDTW is particularly important in order to gain an insight into the role of derivative-based features in time series representation and the impact of combining derivative estimation and segmentation in the accuracy of similarity detection.

DTW on DSA performed as good as or better than the remaining methods, and the performance difference was quite evident in some cases. In particular, DTW on DSA led to quality improvements up to about 59% with respect to DWT, DFT, and Chebyshev, and up to 25% with respect to SAX, PAA, PLA, SD, and APCA. Among these competing methods, it can be noted that DTW on APCA and SD obtained the best results; in general, DTW on APCA, SD, SAX, PAA, and PLA achieved higher quality clustering than Chebyshev, DWT, and DFT.

9.4.2 Efficiency Evaluation

We measured the time performances of DSA and the other methods in accomplishing the tasks of modeling and clustering time series. For each dataset,

we randomly selected samples of sizes equal to 25%, 50%, 75%, and 100% of the size of the entire dataset and, for each sample and method, we used the respective best preprocessing setup.[1] We left the string matching based approaches (i.e., LCSS, EDR, and ERP) out of this presentation since they revealed to be drastically slower than all the other methods.[2]

Performances in Time Series Modeling

Table 9.4 summarizes the best performances (in milliseconds) in modeling time series on the various datasets. PAA, PLA, SAX, SD, and DWT performed as the fastest methods; actually, this result was not surprising since simpler models obviously lead to higher efficiency and, at the same time, lower accuracy. DFT and APCA were always by far slower than the other methods. Our DSA was close to the fastest methods in most cases; in particular, compared to Chebyshev, the larger the dataset the faster was modeling by DSA with respect to modeling by Chebyshev polynomials. It is important to note that, since DSA shares with PAA, PLA, SD, and SAX the same asymptotic time complexity (i.e., linear with the series length), the time differences between DSA and these fast methods should be not really relevant in practical contexts.

Table 9.4. DSA vs. competing methods: best time performances (msecs) in time series modeling

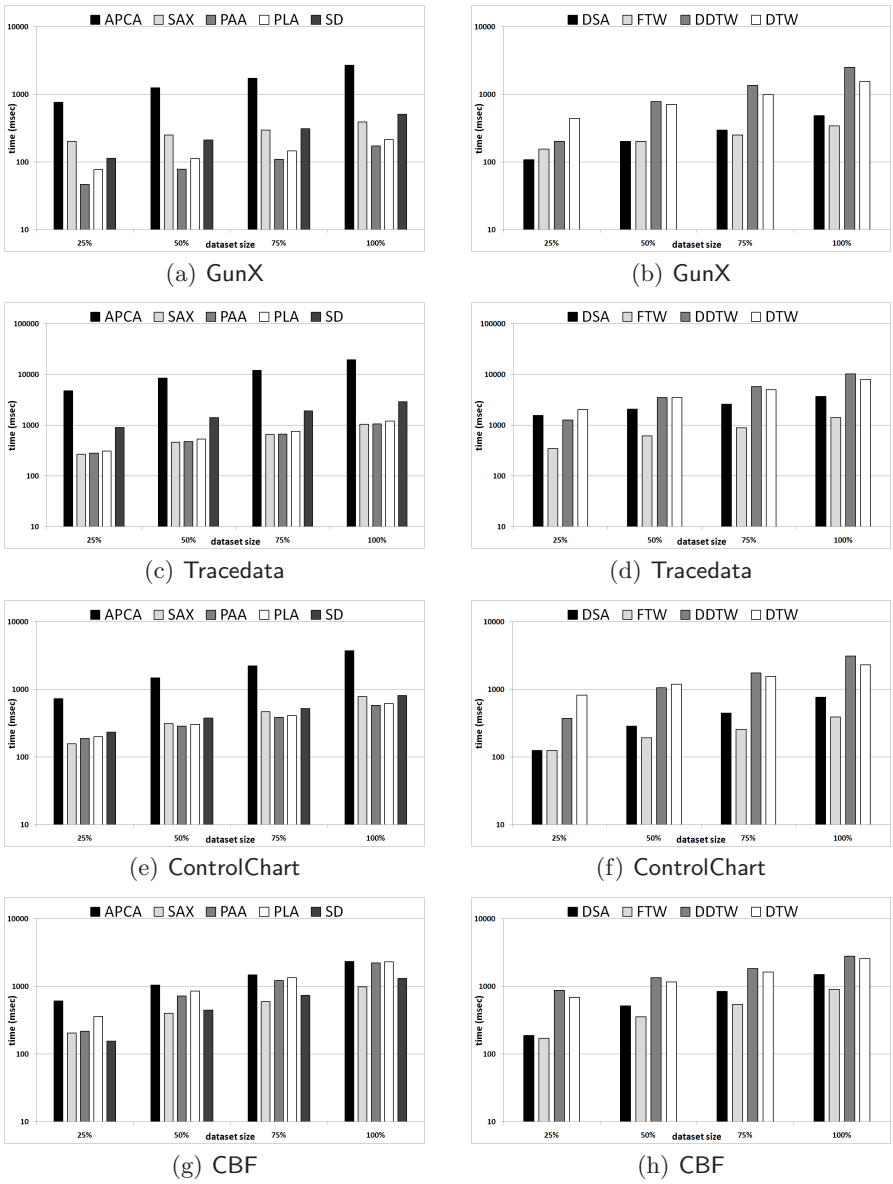
	GunX	Trace data	Control Chart	CBF	Twopat	Mixed-BagSh.	Ovarian Cancer
DFT	2,835	8,276	1,652	3,679	4,701	286,829	570,056
DWT	8	13	9	5	17	48	181
CHEBY	58	58	173	87	232	46	*14*
SD	7	15	14	17	24	65	262
PLA	4	7	*2*	3	*4*	21	46
PAA	*2*	*3*	*2*	*2*	*4*	*14*	41
SAX	8	13	11	11	19	56	67
APCA	1,758	7,151	412	657	2,358	68,739	135,982
DSA	**15**	**40**	**27**	**31**	**52**	**143**	**391**

Performances in Time Series Clustering

We also evaluated the time performances for the clustering task, including in this stage the time required by the series modeling task as well; for the sake of brevity, we present here results obtained by the K-Means algorithm, and we focus on time warping-aware representations.

[1] For each of the smaller samples of the datasets, we performed a preprocessing stage for the various methods, as we did for the entire datasets (see Sect. 9.4.1).

[2] Experiments were conducted on a platform Intel Pentium IV 3GHz with 2GB memory and running Microsoft WinXP Pro.



(a) GunX (b) GunX

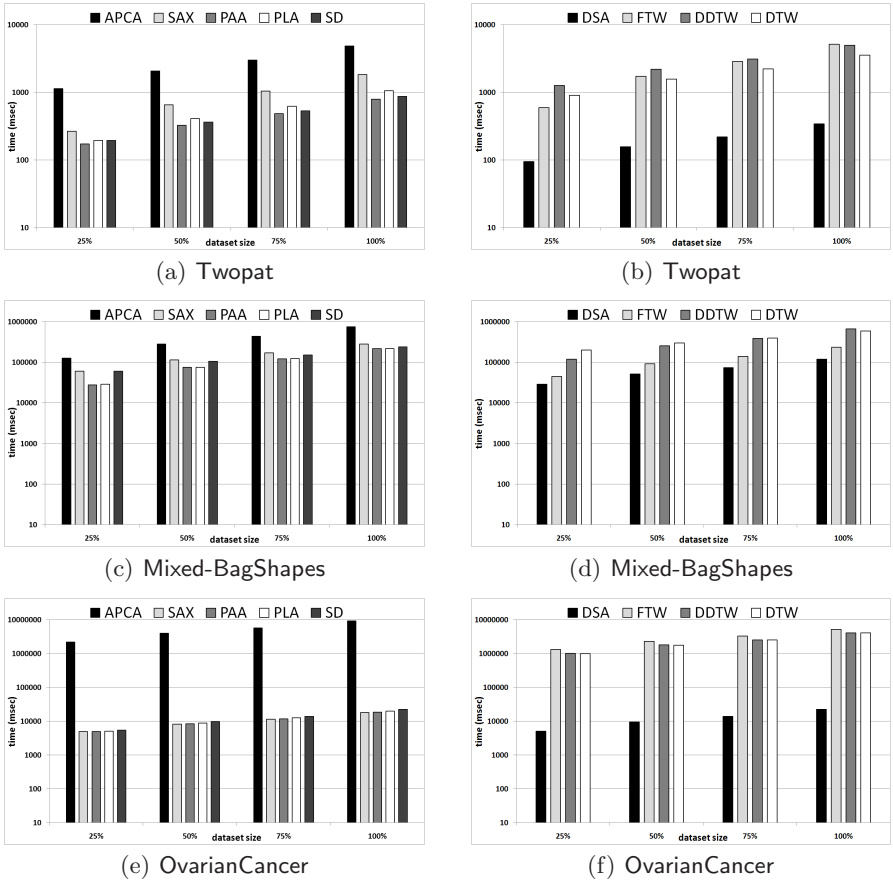
(c) Tracedata (d) Tracedata

(e) ControlChart (f) ControlChart

(g) CBF (h) CBF

Fig. 9.2. DSA vs. competing methods: time performances in time series modeling and clustering (GunX, Tracedata, ControlChart, CBF)

Figures 9.2–9.3 and the summary reported in Table 9.5 show that DTW on DSA drastically improved the clustering performances of basic DTW and DDTW; clearly, this was a consequence of the dimensionality reduction due



(a) Twopat (b) Twopat

(c) Mixed-BagShapes (d) Mixed-BagShapes

(e) OvarianCancer (f) OvarianCancer

Fig. 9.3. DSA vs. competing methods: time performances in time series modeling and clustering (Twopat, Mixed-BagShapes, OvarianCancer)

to the segmentation performed by DSA. More surprisingly, DSA behaved very close to the fastest competing methods: indeed, it is interesting to note that the performance difference between DSA and PAA, SAX, PLA, and SD was not as evident as in the modeling performances previously observed; in particular, DSA-based clustering was even faster than PAA, SAX, PLA, and SD on Twopat (the largest dataset) and Mixed-BagShapes (the dataset with the highest number of classes). This suggests that our DSA is able to yield a time series representation that might require more time to be computed, but generally is more accurate yet convenient to fit the whole task of clustering.

Table 9.5. DSA vs. competing methods: summary of best time performances (msecs) in time series modeling and clustering

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed-BagSh.	Ovarian Cancer
DTW	1,548	8,018	2,298	2,564	3,548	594,446	4,071,815
DDTW	2,518	10,345	3,117	2,789	4,971	664,885	4,098,329
FTW	368	1,491	*405*	*926*	5,254	237,823	5,252,045
DTW on SD	511	2,912	814	1,317	870	240,744	22,655
DTW on PLA	253	1,286	652	2,342	1,103	218,272	20,588
DTW on PAA	*197*	*1,103*	601	2,262	*812*	*216,196*	18,685
DTW on SAX	413	1,113	805	1,132	1,941	282,312	*18,403*
DTW on APCA	2,865	19,503	3,793	2,563	4,864	749,003	9,282,532
DTW on DSA	**521**	**3,733**	**792**	**1,587**	**377**	**121,402**	**23,821**

9.4.3 Summary of Results and Discussion

We evaluated the capabilities of our DSA as well as the competing methods in supporting similarity detection within a clustering framework. Of course, the extent to which such a framework actually led to good solutions depend on how each of the following critical aspects was devised: *(i)* the preprocessing scheme, *(ii)* the similarity method and its application in relation to a representation model, and *(iii)* the clustering algorithms.

Since the focus of this work is on a compact representation of derivative-based features of time series and its impact on similarity detection, it should be emphasized that the evaluation framework is indeed "parametric" with respect to the algorithms. In order to provide a complete specification of our evaluation framework, we conducted experiments using standard clustering algorithms mainly because of their simplicity, applicability, and relatively less dependence on algorithmic parameters.

Facing with the experimental results presented in the previous sections and having the focus on point *(ii)*, we can summarize the main remarks of our study as follows:

- Applying the dynamic time warping (DTW) to DSA sequences leads to clustering solutions that are more accurate than those obtained by using DTW on the original time series. The advantage taken by DSA is essentially due to the combination of a derivative-based feature generation with dimensionality reduction by segmentation. In this way, DTW on DSA performs as good as or better than the major methods for time series representation and dimensionality reduction (i.e., SAX, APCA, PLA, SD, PAA, Chebyshev polynomials, etc.) and even than DDTW alone.
- Modeling a time series into a DSA sequence is reasonably fast if compared to other methods of dimensionality reduction, which is supported by a computational complexity that is linear with the series length; most importantly, the trade-off between accuracy and compactness of DSA sequences makes performing similarity detection more advantageous.

10

Involving the DSA Model into Real Case Applications

Abstract This chapter presents some results aimed at assessing the effectiveness and the efficiency of the DSA time series representation model introduced in the previous chapter when exploited for representing data from scenarios that are apparently very different from that for which it is originally conceived (i.e., the context of time series data management). In particular, the focus here is on *mass spectrometry* (MS) biomedical data, and *low-voltage* (LV) electricity *load profile* data. In this respect, we present *MaSDA*, a system performing advanced analysis on MS data represented according to the DSA model. MaSDA system provides a wide set of *MS preprocessing* operations and embeds a number of tools implementing various tasks of data mining and knowledge discovery, in order to assist the user in taking critical clinical decisions. Furthermore, we focus on the characterization of LV customers based on their consumption data, by describing a clustering framework for detecting groups of customers having similar consumption behavior. The proposed framework exploits the DSA model for representing load profile data and was evaluated on a real application concerning the characterization of Enel (i.e., a large international energy utility) customers.

10.1 Mass Spectrometry Data Management

Mass spectrometry (MS) is a powerful analytical technique that can be applied to tissue or serum samples in order to extract interesting biological information in the form of proteomic patterns [GV03, AM03]. A mass spectrometer is able to produce and separate ions of different masses from a sample, so that the output spectral data consists of a vector of counts, where each count is the number of ions hitting the spectrometer detector during a small, fixed interval of time. A *mass spectrum* is typically represented as a plot of ion abundance (*intensity*) versus the mass-to-charge ratio (m/z). A sound analysis of mass spectra allows for identifying macromolecules contained in the original compounds by associating (portions of) proteins to their peak expressions in a spectrum. Techniques for spectra generation usually depend on sample preparation and ionization, used instrument and ions/macromolecules detec-

tions. Mostly used methods are *Matrix-Assisted Laser Desorption/Ionization - Time Of Flight Mass Spectrometry* (MALDI-TOF MS) [KH88] and *Surface-Enhanced Laser Desorption/Ionization - Time Of Flight Mass Spectrometry* (SELDI-TOF MS) [HY93].

Mass spectrometry approaches have been recently coupled with advanced data analysis techniques in order to discover useful knowledge that can aid clinicians in early detecting disease-related biological states. A common task of MS knowledge discovery consists in classifying spectra to discriminate them based on their biological states (e.g., healthy or diseased individuals). This task has to cope with huge dimensionality and frequently occurring noise in MS data, and becomes even harder when a training set of positive/negative examples from data is poorly available. In this case, the goal is to organize a collection of spectra (whose classification is unknown) into meaningful groups (i.e., clusters), based on interesting relationships discovered in the data. Clustering of MS data finds natural application to many real MS scenarios, since the various pathologic states from clinical studies need to be identified and discriminated in an unsupervised way.

A further crucial point in mass spectrometry is preprocessing. MS data preprocessing has been recognized as a mandatory phase in mass spectra data analysis [CBM07]. The need for preprocessing mass spectra arises since *(i)* data obtained from a mass spectrometer have very large dimensionality and *(ii)* MS data are naturally corrupted by various noisy factors. Several research studies have been proposed on the development of preprocessing steps for MS data (e.g., [CBM07, WNP03]), and in some cases they have focused on specific steps, such as baseline subtraction [WCD+05, SS04, RJFD99], peak identification [WKG04, YMA+03], and peak alignment [WCC05, Jef05, SS04]. Also, there has been recently a growing interest for developing MS data preprocessing systems that are able to fulfill the following main requirements: filtering data and highlighting relevant spectra portions with respect to non-relevant ones (e.g., noise), and allowing the user to perform the various preprocessing stages iteratively and interactively.

In this section, we present *MaSDA – Mass Spectrometry Data Analysis*, a system for advanced analysis of MS data [GPT+09c]. The general objective of MaSDA is to assist the user in discovering useful knowledge from MS data. The discovered patterns of knowledge might eventually support the user (e.g., the clinician) to take critical decisions; for instance, if interesting relationships on certain biological conditions referring to a given disease have been found out by analyzing MS data, then one might use this new information to design new therapies.

The key idea underlying our approach to MS data analysis, which is implemented in the MaSDA system, is to exploit the temporal information implicitly contained in mass spectra and model such data as compact *time series* by employing the DSA model (cf. Chapt. 9). The proposed MS data representation model is aimed to take some advantages with respect to the traditional count-vector-based approaches to MS data representation, in particular:

- The problem of high dimensionality in MS data is addressed by identifying variable-length segments in the time series representing mass spectra. Each one of these segments is conceived to be comprised of locally tight points, and is finally mapped to a synthetic information. This enables to drastically reduce the number of noisy dimensions while preserving relevant features (i.e., trends in the series profile).
- The critical task of preprocessing MS data is relatively simplified by employing major existing techniques for similarity detection in time series (cf. Chapt. 8), which allow for dealing with mass spectra in a way more robust to noise and suited to different profiles of the spectra.

Another important aspect of our MS data analysis system is that it offers a graphical tool for preprocessing the raw mass spectra, with the following main features [GPT+08b]:

- Wide set of supported preprocessing operations – it is designed to cover most of the MS data preprocessing steps that have been recognized as relevant in the literature;
- Efficiency – it guarantees high performance in MS preprocessing, by adopting fast algorithms for each step. This allows for efficiently dealing with high dimensional data;
- Support for user interaction – it enables the user to monitor and control the whole preprocessing task; in particular, the user can choose which preprocessing steps have to be performed and their execution order, and she/he can properly set the parameters involved into each step;
- Ease-to-use – it provides a user-friendly graphical interface and a simple wizard which guides the user in each preprocessing step;
- Web-based access – it makes use of the Java™ Web Start technology,[1] which allows for launching the tool directly from the Web.

Besides the functionalities of MS preprocessing and time series based MS modeling, MaSDA system is designed to perform various tasks of MS data analysis, by employing data mining and knowledge discovery techniques, and to evaluate and visualize the patterns of knowledge discovered from the input MS data. As experimentally proved on publicly available datasets, our system has been shown to be a valid support for the user interested in effectively and efficiently analyzing MS data. As experimentally proved on publicly available datasets, our system has been shown to be a valid support for the user interested in effectively and efficiently analyzing MS data.

10.1.1 Prior Work in Mass Spectrometry Data Management

In the last few years several systems for MS data management have been developed, ranging from domain-specific tools to general-purpose platforms.

[1] http://java.sun.com/products/javawebstart/

For instance, the database management system presented in [TSD+06] involves a statistical data analysis software to compare healthy and sick patients. MS-Analyzer [CV07] is a system that allows the ontology-based design of distributed applications for management, preprocessing and graphical analysis of MS data.

Recently, there has been a lot of research confirming the need for extracting knowledge from MS data. This has mainly involved various tasks aimed at identifying biological patterns and organize them at different degrees of automation.

Data mining techniques have been recognized as a valuable support to discovering significant patterns from MS data. The main focus in the MS domain has been on the task of *supervised classification*, or simply classification, that is learning how to assign each instance with a category from a set of predefined categories (classes). The problem of MS data classification has been addressed by using different machine learning approaches, including decision trees [GFdS+05, PKS+04], discriminant analysis [MEZ+05], support vector machines [PKS+04, WNP+04], and genetic algorithms [BMW+03, PAH+02]. A comparison of different classification methods has been presented in [WAF+03, WNP03].

Clustering of MS data has been attracting a growing interest in various MS applications. A common way to address this problem is to apply classic clustering schemes and equip them with the Euclidean distance. For instance, [PWJ+04] uses an average link agglomerative hierarchical clustering equipped with Euclidean distance in order to identify groups of proteins that show similar patterns of proteins copurifying with components of TFIID. In [SSML06], hierarchical clustering is applied to consolidate peak lists of GC-MS (*Gas Chromatography-Mass Spectrometry*) metabolic profiling data acquired on Leishmania mexicana, and to separate the wild-type and two mutant parasite lines based on their metabolic profile.

The study presented in [SHR+03] is focused on comparing spectra with respect to their peak heights. This is accomplished by employing principal component analysis to compare relative orders of the peak heights rather than directly peak heights.

[ZAHB00] uses a grid-based clustering algorithm to discover the functional molecules for determining structures of the pharmacological compounds. [BGM+05] proposes to map spectra to a complex space using discrete Fourier transformation. A thresholding approach is then adopted to denoise and reduce the length of each spectrum, and Bayesian model-based clustering is applied to the reconstructed data. However, poor experimental tests (only on one, small dataset of 70 patients) do not support the understanding of practical impact of this advanced approach.

It should be noted that most approaches to clustering MS data mainly differ each other with respect to the preprocessing steps and the clustering schemes used, whereas spectra representation models have been rarely inves-

tigated. Within this view, the basic idea presented here is to model spectra emphasizing the natural temporal evolution of protein profiles.

Also, most of the above works on MS data classification/clustering assume that clinical studies have been conducted on the data collections being available a-priori knowledge on the data domain. This typically drives the development of classification/clustering methodologies such that they may focus on only some portions of the original spectra, that is portions which likely contain potential discriminatory patterns (biomarkers). By contrast, our approach can be in principle applied to the entire spectra as well; nevertheless our underlying MS representation model is able to automatically performs dimensionality reduction on the spectra, thus preserving their relevant information.

10.1.2 The Mass Spectrometry Data Analysis (MaSDA) System

The MaSDA system consists of five main modules, which are sketched in Fig: 10.1 and described below:

1. MS Data Preprocessing: it performs one or more preprocessing steps on the raw spectra in order to make them amenable to the further analysis stage. In particular, this module includes at least the following preprocessing operations: range cut, peak smoothing, detection of valid peaks, baseline correction, quantization, and normalization.
2. Time Series based MS Data Modeling: this module transforms the preprocessed MS data into time series, using a model conceived to maintain the significant trends (peak profiles) while reducing the data dimensions.
3. MS Data Analysis: it includes a number of submodules each performing a certain task of knowledge discovery, such as cluster analysis, frequent pattern discovery, data summarization, and so on. The input for this module is the preprocessed spectra, which is of the form either original (output of module 1) or based on time series (output of module 2).
4. Pattern Evaluation: it is in charge of assessing the validity of the discovered knowledge patterns.
5. Knowledge Presentation: this module finally presents the discovered knowledge by using visualization tools.

10.1.3 Pre-analysis Processing

A raw spectrum generated by a mass spectrometer is substantially a combination of three components: the true signal, a baseline signal, and noise [CBM07]; in particular, the true signal contains biological information, whereas the base intensity level (baseline) varies from point to point across the m/z axis, so that intensity values that are under the baseline represent ground noise and should be hence filtered out. Separating and reconstructing such individual components from a raw spectrum is a hard task, since their

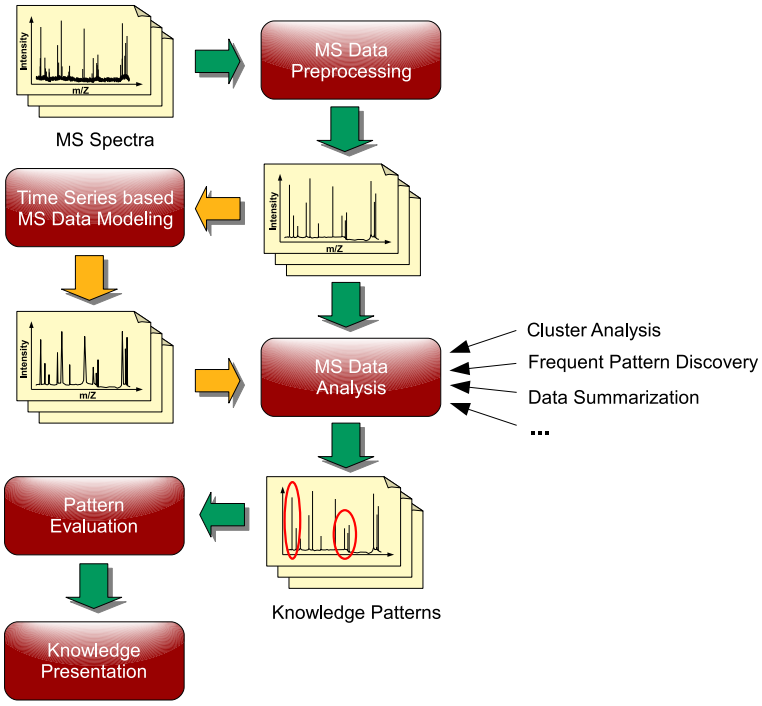


Fig. 10.1. The overall conceptual architecture of the MaSDA system

analytical forms are not known. Thus, spectra usually need to be subject to one or more preprocessing operations, in order to make them amenable to further analysis phases.

Since the variety of spectrometry platforms, experimental conditions and clinical studies, there exists a number of preprocessing operations (see, e.g., [CBM07, WNP03]). While there has not been shared agreement about a preprocessing scheme, a reasonable list of preprocessing steps on mass spectra can be given as follows:

- *calibration*, which is used to map the observed time of flight into the inferred mass-to-charge ratio;
- *filtering or denoising*, which aims to reduce random noise generated by electronic or chemical causes;
- *baseline correction*, which is in charge of recognizing and filtering out the baseline signal of mass spectra;
- *normalization*, which makes peak intensities understandable over a uniform range;

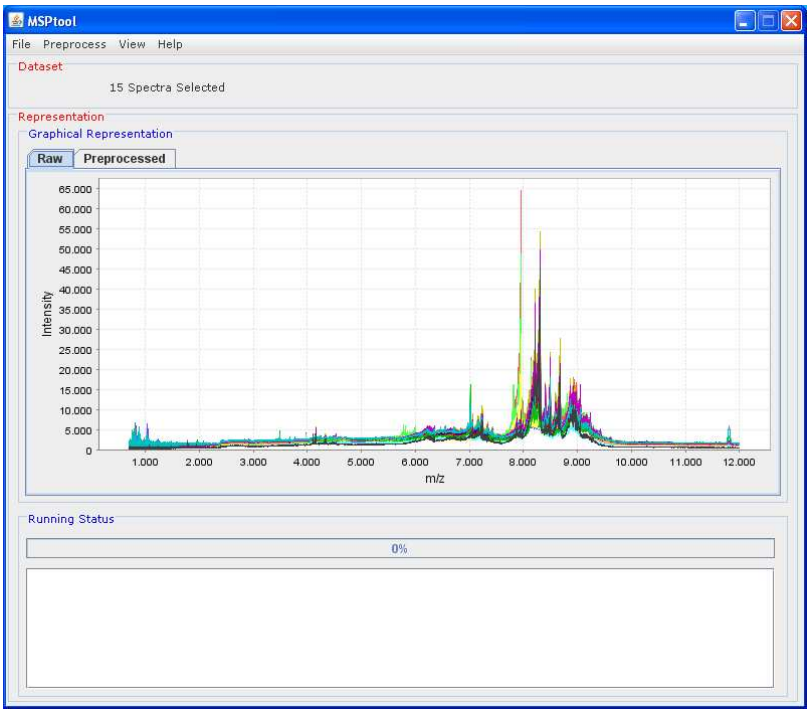


Fig. 10.2. A sample screenshot of the MaSDA system for MS preprocessing

- *peak detection*, which is in charge of locating specific proteins or peptides on the identified locations on the m/z axis and typically involves an assessment of the spectra local maxima and their signal-to-noise ratio (S/N);
- *peak quantification*, which represents each detected peak by means of a concise information (e.g., peak heights or areas);
- *peak matching/alignment*, which aims to recognize the peaks in different samples that correspond to the same biological molecule.

In this section we describe the capabilities of the MaSDA module for MS preprocessing we called *MSPtool*. MSPtool is a Java™ based tool that implements most of the MS preprocessing operations discussed above (cf. Fig. 10.2). This tool offers its features visually in order to assist the user in performing an MS preprocessing task, i.e., observing the raw spectra, selecting an appropriate sequence of preprocessing steps, and choosing the parameter setting for each of the selected preprocessing steps.

MSPtool is able to deal with various formats storing the raw spectrum/spectra to be preprocessed, including plain-text files, comma separated values files (CSV), and XML data. Also, the tool allows the user to graphically represent preprocessed spectra, which is useful to visually explore (and compare) the spectra profiles before and after the preprocessing step. Figure 10.3

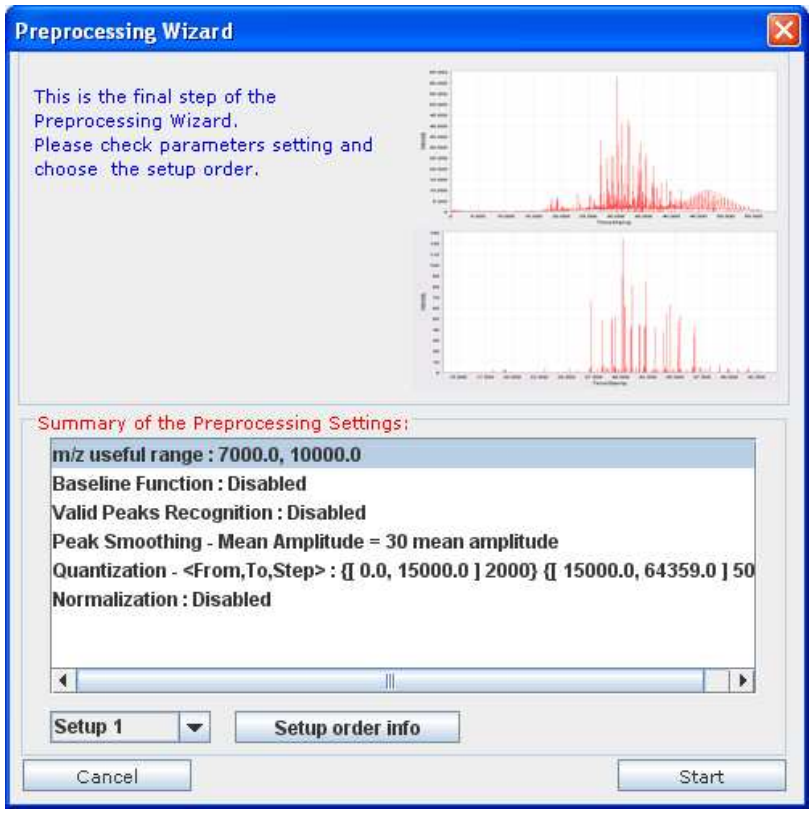


Fig. 10.3. MSPTool preprocessing wizard - summary of the preprocessing settings

shows a screenshot of the last step of the preprocessing wizard, which reports a summary of the preprocessing setting; in this step, it is also possible to change the order of the selected preprocessing operations.

It should be noted that, although MSPtool has been originally designed as a standalone application, we have also provided a Web-based version using the Java™ Web Start technology. This feature of the tool is mainly motivated by our intention to make MSPtool publicly available and to simplify the processes of deployment and upgrade of the tool.[2] In the following, we describe the main steps of MS data preprocessing involved into MSPtool.

[2] A beta version of the MS data preprocessing tool is available at the following Web address: http://polifemo.deis.unical.it/~gtradigo/jnlp/msptool/

Range Cut

This step performs a cut of the m/z range of the spectra, in order to filter out those portions of spectra that do not contain relevant biological information.

Peak Smoothing

Peak smoothing falls into the category of peak detection/quantification operations. This step aims to smooth the peak profiles in the spectra and to reconstruct the theoretical Gaussian profile of the peaks. An ideal peak profile comprises two parts: a monotonic ascending side and a monotonic descending side. We call *M-peak* a spectrum peak having its intensity higher than both the previous and the next point, i.e., a local maximum in the spectrum (cf. Fig 10.4 (a)–(b)).

The peak smoothing algorithm has a parameter w_p, *peak amplitude*, which is a function of the mass spectrometer resolution. This parameter can be initially set to the average width of peaks in the spectrum, or modified according to the data features. Basically, the algorithm works as follows: first, it detects all the M-peaks in the spectrum; each M-peak (except the last one) is compared with the next M-peak. If the distance between these two M-peaks is lower than $w_p/2$ then either a descending phase or an ascending phase can occur, and the spectrum is modified such that the resulting peak has the expected pseudo-Gaussian shape for both the ascending and the descending sides (cf. Fig. 10.4 (c)–(d)).

Valid Peaks Recognition

Valid peaks recognition is a further step of peak detection/quantification. This step aims to recognize as valid peaks the local maxima into a mass spectrum that satisfy specific requirements. In particular, the algorithm for valid peaks recognition implemented into MSPtool takes into account the signal-to-noise ratio (S/N) and works as follows: for each spectrum, the S/N at each local maximum of the spectrum is computed as the ratio of the intensity at the maximum to the local noise estimate; then, only the local maxima having S/N greater than a user-defined threshold (*multiplicative factor*) are recognized as valid peaks. The non-valid peaks in a spectrum are discarded from the further analysis.

Baseline Correction

This step aims to identify the baseline signal in the spectra and filter out all spectra intensity values below the baseline. The user can choose a function that approximates the baseline (i.e., the baseline function) and setting the parameters for each function. MSPtool offers the following baseline functions:

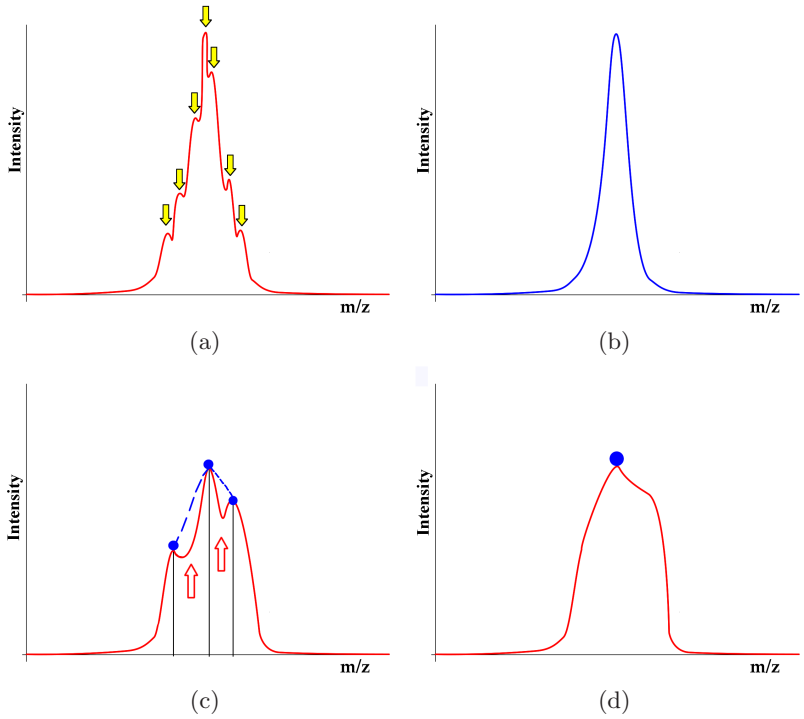


Fig. 10.4. Peak smoothing: (a) example M-peaks and (b) the corresponding ideal peak; (c) three local M-peaks and (d) the resulting profile after smoothing

linear function, *logarithmic function*, *exponential function* and *piecewise linear function*. The first three functions approximate the baseline as a linear, logarithmic and exponential function, respectively, whereas the definition of the piecewise linear function is as follows. The m/z range of each spectrum is divided into a user-defined number of equally-sized windows. The final piecewise linear function is composed by a number of linear functions, each of them properly defined according to the associated window. For each window, the corresponding linear function is computed by solving a line fitting problem to the local minima in the window.

Quantization

This step performs a quantization of the spectra, i.e, a discretization of the original intensity values according to specific quantization levels. A non-uniform quantization model is used in the MPStool in such a way that two or more ranges in the intensity axis are identified and subject to different fine-grained quantization.

Normalization

Spectra normalization changes spectra shapes by transforming original intensity values into new ones proportionally calculated according to a certain fixed range. MSPtool implements various normalization techniques, including z-normalization and min-max normalization. The former subtracts the mean over all the spectra intensities from each intensity value and then divides this difference by the standard deviation over all the spectra intensities; the latter scales the intensity values such that, for each m/z and over all the spectra, the smallest intensity value becomes zero and the largest intensity becomes one.

10.1.4 Time Series-Based Modeling of MS Data

A (preprocessed) mass spectrum is a sequence of paired values $S = [((m/z)_1, I_1), \ldots, ((m/z)_m, I_m)]$, where each pair is comprised of a mass-to-charge-ratio value and the associated intensity value. A mass spectrum so defined can be trivially modeled as a time series $X = [(y_1, z_1), \ldots, (y_m, z_m)]$ whose y_j correspond to the spectrum intensity values I_j, and the timestamps z_j correspond to the values $(m/z)_j$. Indeed, the notion of time implicitly lies in the sequence of mass-to-charge values.

Time series representing mass spectra are typically high dimensional data. Thus, it is desirable to model such time series into a compact representation that synthesizes the significant variations in the time series profile. For this purpose, we exploit the DSA representation model introduced in Chapt. 9.

We remark that the number d of segments produced by DSA is usually much smaller than the number of original points in the series (i.e., $d \ll m$). This enables a significant increment of performance efficiency for the various data analysis algorithms that will be performed on DSA representations of mass spectra.

10.1.5 Using the MaSDA System for Organizing MS Data

A major task of MS knowledge discovery consists in classifying spectra in order to discriminate them on the basis of their biological information (e.g., healthy or diseased individuals). To cope with huge dimensionality and frequently occurring noise in MS data, this task requires careful preprocessing and modeling of the data. However, the organization task is particularly difficult when a-priori knowledge on the predefined set of categories or a training set of positive/negative examples from data is poorly or not available at all. In this case, the goal is to infer an organization of a collection of MS data into meaningful groups (i.e., clusters), based on interesting relationships discovered in the data. Clustering of MS data finds natural application to many real MS scenarios, since the various pathologic states from clinical studies might

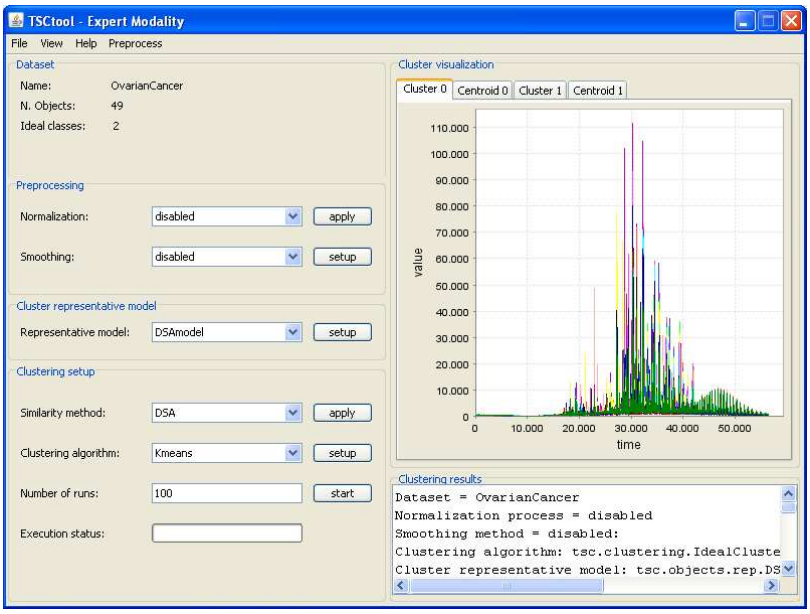
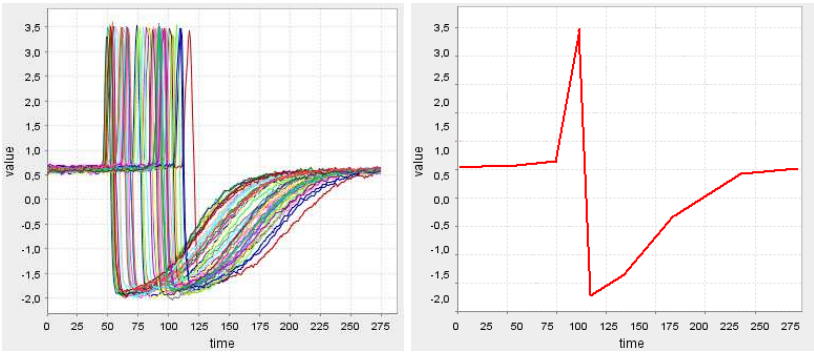


Fig. 10.5. A screenshot of the MaSDA tool for clustering MS data

require to be discovered in an unsupervised way. In the following we describe how MaSDA can be used to organize MS data by a task of cluster analysis.

Figure 10.5 shows a screenshot of a Java™ based tool embedded in MaSDA for clustering MS data (available from the OvarianCancer dataset [PAH+02], cf. Appendix A). On the left of this figure, we can observe a number of component panels devoted to the configuration of a clustering experiment, which involves the choice of the preprocessing (smoothing) function, the model of cluster representative, the method of representation and similarity between series, and the algorithm of clustering. On the right of the figure, each of the output clusters and relating representative can be explored using different choices of visualization; the clustering results can be also saved into a file for further reloading. Also, we can observe in the menu bar the presence of a command for launching the preprocessing tool (MSPtool) previously described.

Another important task allowed by MaSDA is *data summarization*. Given a set of MS time series, the objective here is to generate a summary, or prototype sequence that is able to capture the most relevant features of the series in the given set. Since the input series may have different length and scales, the task is not trivial (i.e., we cannot directly resort to the computation of an "average" time series); rather, a concise representation is desirable to include the significant trends in the set as well as to filter out irrelevant information [GPTG09a]. Figure 10.6 shows an example of summarization of a certain set of time series data.



(a) (b)

Fig. 10.6. An example of data summarization: (a) a set of time series and (b) the computed prototype

10.1.6 Experimental Evaluation

The MaSDA system has been tested on various real MALDI/SELDI-TOF MS data obtained using different clinical studies under different mass spectrometry platforms and experimental conditions; in particular, some of the used collections are publicly available from authoritative sources (e.g., the NCI's Center for Cancer Research [fCRsp]), other ones have been provided by the proteomics laboratory at the Magna Græcia University of Catanzaro.[3]

A major goal of the experiments conducted on MS data by using MaSDA was to identify groups of subjects that show similar characteristics according to the expected pathological states (e.g., in the Prostate dataset [POP+02] (cf. Appendix A) different cancer or benign conditions at various levels of PSA). Moreover, in this context a challenge is represented by the discovery of the proteomic profiles that distinguish disease-related or cancer conditions from the healthy ones. For instance, some discriminatory patters might be found out around early m/z values, other ones might be detected according to sequences of peaks at a certain intensity level. Intuitively, this issue can be more easily addressed by exploiting our time-series-based modeling of MS data: indeed, the compact representation that is substantially comprised of relevant features in the data (while discarding various noisy factors) favors the identification of significant patterns in the spectra.

In this section we describe the experimental evaluation aimed to assess effectiveness of the proposed system. We first present the data used in the main experiments, the cluster validity criteria adopted, and our evaluation methodology. Then we discuss the experimental results from both a quantitative and qualitative point of view.

[3] http://proteomics.unicz.it

Datasets

Typical datasets in real-world MS application domains contain just tens or hundreds of spectra [CBM07]. We used datasets available from the NCI's Center for Cancer Research [fCRsp], namely Cardiotoxicity, Pancreatic, and Prostate. All these datasets contain SELDI-TOF spectra and were obtained using different clinical studies under different mass spectrometry platforms and experimental conditions (cf. Appendix A).

Cluster Validity

To evaluate clustering quality, we exploit the availability of a reference classification for the data and use measures that allow for computing how well a clustering solution fits a predefined scheme of known classes (natural clusters). In particular, we resort to the F1-Measure (Def. 2.5) and entropy (Def. 2.7) external cluster validity criteria.

Evaluation Methodology

The objective of our evaluation methodology was to automatically measure the quality of a clustering solution. This requires to define the following elements for each experimental test: the data, the way(s) to preprocess the data, the clustering algorithm along with the distance/similarity measure and the choice of number of clusters, and the quality measure(s). In this work, we are given the previously presented datasets, and F1-Measure and Entropy as the quality measures.

Regarding clustering algorithm(s), it should be emphasized that the proposed system is parametric with respect to the clustering scheme. In this work, we resort to centroid-based partitional K-Means clustering algorithm (cf. Chapt. 2), due to the advantages offered in terms of simplicity, execution time and space requirement. We equipped the K-Means with either the Euclidean distance (L_2) or the Dynamic Time Warping (DTW) (cf. Chapt. 8); the latter has been used on the spectra modeled as time series by means of our DSA, whereas the Euclidean distance is referred to as the baseline method for computing distance among MS data.

As far as the data preprocessing, we did not fix a unique way to preprocess data but rather we identified the following sequences of operations as different *preprocessing setups*:

(S1) baseline subtraction, peak detection, normalization;
(S2) peak detection, baseline subtraction, normalization;
(S3) baseline subtraction, normalization;
(S4) peak detection, normalization;
(S5) peak detection, baseline subtraction;
(S6) baseline subtraction, peak detection;

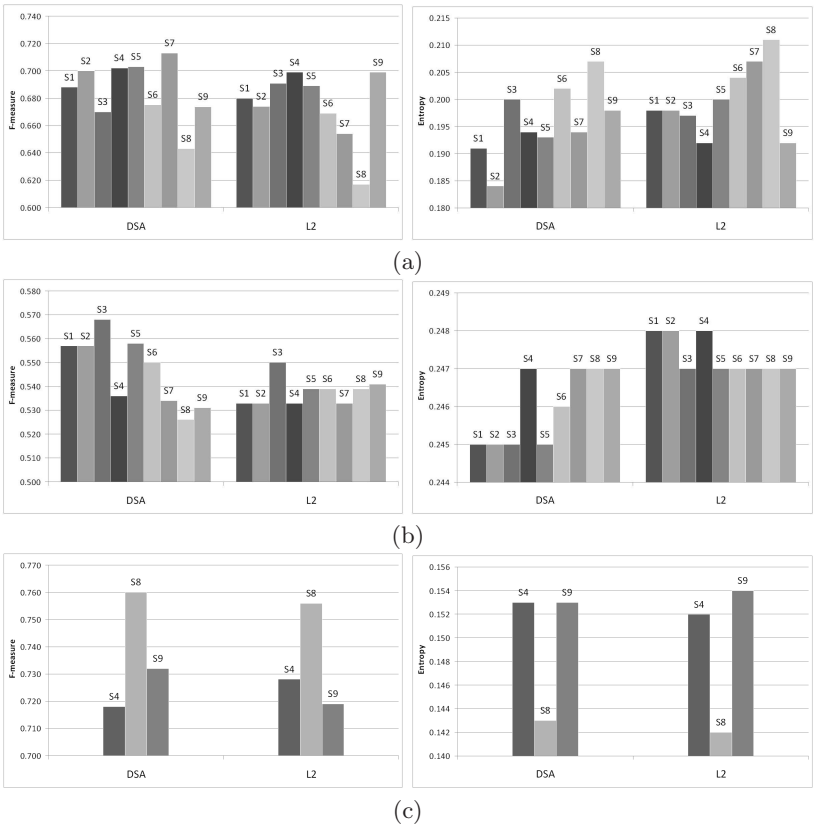


Fig. 10.7. Evaluating MaSDA system—clustering quality results (F1-Measure on the left, Entropy on the right): (a) Cardiotoxicity, (b) Pancreatic, and (c) Prostate

(S7) baseline subtraction;
(S8) peak detection;
(S9) normalization.

Results

For each test dataset, we conducted various runs of K-Means algorithm, by varying the preprocessing setup, and compared the results obtained by our time series based approach with the standard Euclidean approach.

Quantitative Evaluation

We initially focused on testing the ability of the framework to detect and distinguish all the meaningful groups in the data, that is 4 for Cardiotoxicity and Prostate and 2 for Pancreatic. Thus, for all experiments on a specific

Table 10.1. Evaluating MaSDA system: summary of clustering results on the various test datasets

	# clust. (K)	DSA			L_2		
		F1 (*prec./rec.*)	*entr*	*preproc. setting*	*F1* (*prec./rec.*)	*entr*	*preproc. setting*
Cardiotoxicity	4	.72 (.75 / .68)	.19	$S7$	.69 (.70 / .69)	.19	$S4/9$
Cardiotoxicity	2	.76 (.78 / .73)	.38	$S2$	.67 (.67 / .67)	.45	$S4/9$
Pancreatic	2	.57 (.58 / .56)	.48	$S3$	.55 (.57 / .53)	.49	$S3$
Prostate	4	.76 (.84 / .69)	.14	$S8$	.75 (.84 / .68)	.14	$S8$
Prostate	2	.78 (.79 / .78)	.34	$S8$	.77 (.79 / .76)	.34	$S8$

dataset, we fixed the number of clusters exactly to the number of classes associated with that dataset.

Figure 10.7 shows the quality results obtained on the various datasets, and compares our DSA-based approach to the standard Euclidean distance.[4] Clustering performances were generally affected by the selected preprocessing setup, while baseline subtraction revealed to be essential for improving the clustering quality in most cases. Notice that only setups $S4$, $S8$ and $S9$ were considered for Pancreatic, since this dataset was been already subject to baseline subtraction.

Our DSA-based approach achieved reasonably good clustering results at least in Cardiotoxicity and Prostate; in Pancreatic, clustering inevitably resulted in lower performances mainly due to the dominant presence of m/z values with very low intensity values and just a few characteristic trends in the spectra. Anyway, for all datasets the DSA-based approach was able to achieve higher F1-Measure and lower entropy scores than the standard Euclidean approach.

Table 10.1 summarizes the quality results (i.e., F1-Measure along with corresponding precision and recall, and entropy) referring to the best preprocessing setups; for each dataset and method, the best preprocessing setup is that leading to the highest quality in terms of F1-Measure. This table also includes results obtained by a two-class task of clustering; precisely, for Cardiotoxicity and Prostate, we also tried to select only the data assigned with the definite cancer (diseased) or the definite non-cancer (healthy) classes, and then we performed clustering on this subsets. The objective here was to give emphasis on distinguishing solely the extreme classes.

Performing K-Means with DTW on DSA sequences behaved as good as or better than standard Euclidean distance on the original spectra, up to a 10% improvement (Cardiotoxicity, 2 classes). It should be noted that the advantage of using the DTW on DSA sequences becomes important since the DSA model yields compact yet dense representations of the original spectra.

[4] Reported results were averaged over 100 runs of K-Means algorithm with a very low standard deviation (ranging between 0.001 and 0.008).

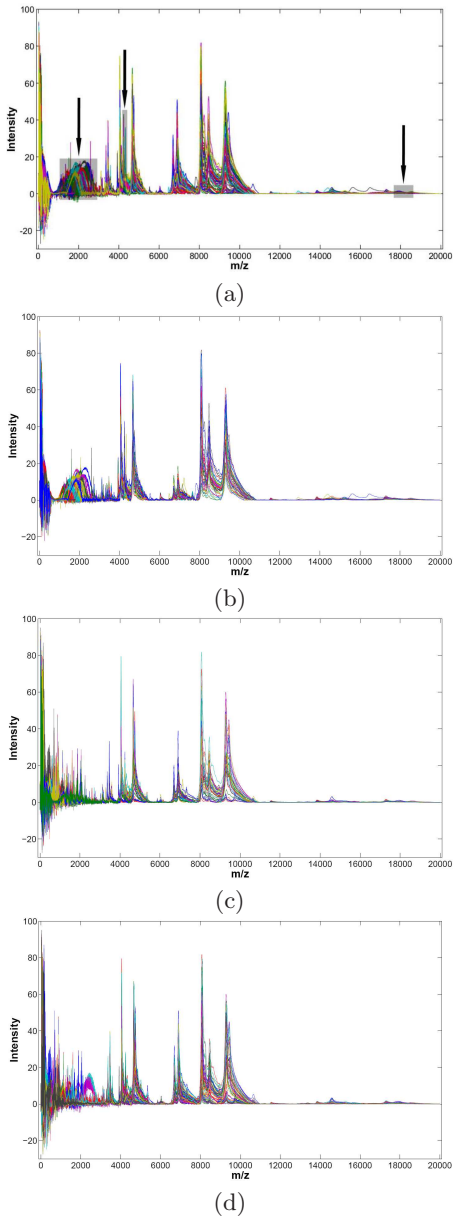


Fig. 10.8. Evaluating MaSDA system—clusters vs. natural classes from Prostate: (a) cluster and (b) class of cancer with PSA>10 ng/ml; (c) cluster and (d) class of no evidence of disease

The lower dimensionality of the spectra-series achieved by DSA is beneficial for the efficiency of the clustering task (and further post-processing analysis),

while not affecting negatively the clustering effectiveness. To report some details, the following data compression ratios were achieved (on the preprocessed spectra): 64% on Cardiotoxicity, 65% on Pancreatic, and 97% on Prostate. The latter result is particularly significant since it shows that a very high compression was obtained for a dataset on which DSA still performs very closely to (slightly better than) the standard approach based on L_2.

Qualitative Evaluation

It is useful to gain an insight into the output clusters with respect to the real classes. In particular, here we concentrate on Prostate and compare the definite cancer and no-evidence-of-disease clusters to their respective groupings in the reference classification (Figure 10.8)—for the sake of comparison, original spectra have been displayed for both clusters and classes. In Prostate, as discovered by authors of the study in [POP+02], there is a number of main discriminatory patterns, mostly distributed in the early m/z values. Figures 10.8(a) and (b) plot spectra belonging to the cluster and the class of definite cancer, respectively. At a first glance, the two plots look quite similar, suggesting that most cancer spectra have been correctly recognized. Also, we have highlighted on Figs. 10.8(a) some of the most evident trends that distinguish the cancer conditions from the healthy ones. Analogously, we can observe similar graphs for the healthy cluster/class (Figs. 10.8(c)/(d)).

Other Preliminary Results

In addition to the ones presented above, further experiments were performed on two more datasets [GPT+07]: OvarianCancer [PAH+02] (cf. Appendix A), and a smaller dataset consisting of MALDI data generated at the proteomics laboratory at Magna Græcia University of Catanzaro. In both datasets, spectra fall into either the healthy class or the diseased class. Figure 10.9 shows raw and preprocessed spectra of OvarianCancer dataset.

Noise reduction was performed considering a linear model for the baseline. A peak was recognized as valid if its maximum intensity value is at least 2.5 times the corresponding noise intensity value. Quantization step was set to 2,000 counts for intensity values within the range [0..10,000] and to 500 counts for values greater than 10,000. Moreover, the range of (m/z) values was reduced to focus only on significant portions of the spectra. In particular, the cut involved two m/z intervals: the first interval corresponds to intensity values close to zero (i.e., with m/z ranging within [0..15,000]) and the second one corresponds to intensity values identifying spectrum contaminants such as the polymer between [43,000..56,384] m/z.

Clustering results on OvarianCancer highlighted high effectiveness provided by the proposed system. In particular, the expected two classes were well-recognized with precision, recall and F1-Measure values measured in 0.88, 0.86 and 0.87, respectively. Note that such values are sufficiently high (i.e., they are close to 1) proving that our system was capable both to identify

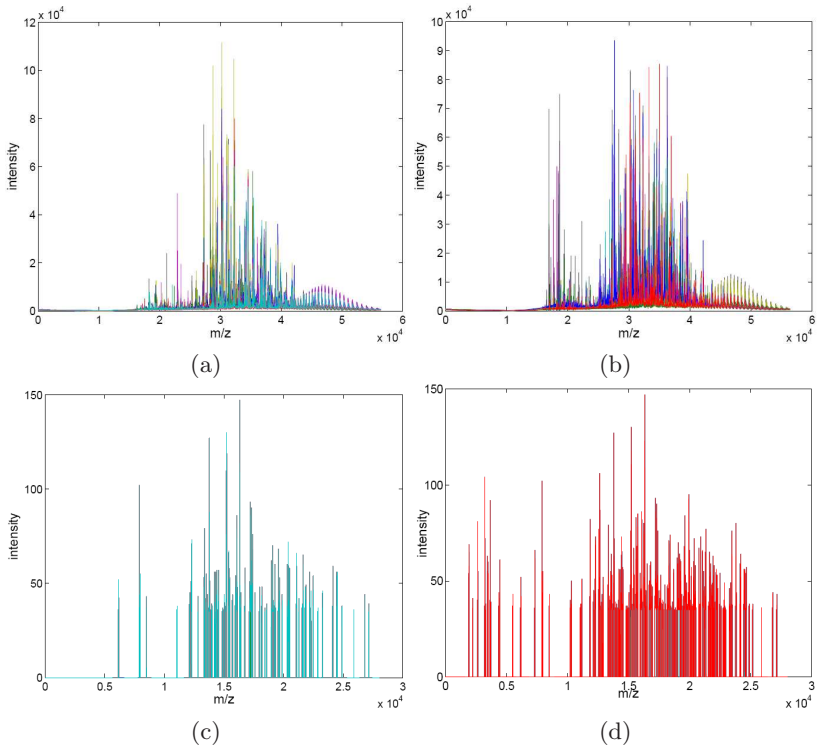


Fig. 10.9. Evaluating MaSDA system on OvarianCancer dataset: on top, raw spectra of (a) control class and (b) diseased class; on bottom, preprocessed spectra of (c) control class and (d) diseased class

homogeneous groups of MS data and to separate different data in distinct classes, according to biomarkers associated to discriminant peaks.

The framework was also tested on the MALDI dataset produced at Magna Græcia University of Catanzaro. It is worthy to noticing that we encountered different issues in the preprocessing phase with respect to OvarianCancer, which are due to their different originating laboratories. Nevertheless, MaSDA system obtained comparable quality results.

10.2 Low-voltage Electricity Customer Profiling Based on Load Data Clustering

Today energy markets are characterized by a growing insecurity in the wake of their liberalization. Due to an increasing customer volatility, it is becoming more difficult for utilities to plan their investments for the next decades. The new deregulated energy market needs utilities that have to face

with challenging issues, which mainly arise from the new way of conceiving frameworks as customer-centric instead of early supplier-centric ones. In addition, the problem of characterizing and predicting their customers' behavior and fitting a proper tariff policy accordingly has been recognized as relevant in this context. Designing new tariff structures allows the energy utilities to encourage competition, efficiency, economical use of the resources. By means of proper tariffs, it is possible to support customers' interests and, at the same time, to recover the cost of electricity in a reasonable time. At the end of the chain air pollution due to CO_2 power plants emissions could be reduced (peak shifting, power losses reduction) [McC03, PPB97]. Enel, a large international power utility, has recently completed an Italian project called the Telegestore project [BCR05, Rog07]. New smart meters recently adopted are able to measure and store load profiles of their mass-market LV customers in a flexible and effective way.

In recent years, problems in load profile data management have been arisen due to the technological improvement in electricity utility devices. A significant research effort has been focused on load profile classification, especially regarding clustering of medium-voltage customers and short-term load forecasting of anomalous days.

Customer classification puts the basis for properly designing tariff structures. The use of load pattern-based features has been identified as a key factor for classifying customers on the basis of their electrical consumption behavior. Classification allows utilities to promote collective tariffs rather than individual ones for each customer.

All the proposed techniques for load profile classification generally belong to pattern recognition and data mining approaches [FRVG05, CNP$^+$05, CNP06, NDJR06, ALB$^+$07, THD07, TSTK08]. Load profiles are usually represented as time sequences and the notions of proximity used for comparing them are typically based on the Euclidean distance. In the context of load profile clustering, the most used approaches refer to partitional clustering and hierarchical clustering (cf. Chapt. 2).

In this section, we present a clustering framework for electricity customer load profiles, which is supported by information on meta-data (e.g., customer type, meter type, day, contract, location) [GPT$^+$09a]. Enel supported this work by providing data about 30,000 LV load profiles of anonymous Italian customers. Handling such a large collection of customer load data has represented an important effort of our work. The proposed framework exploits a time series-based representation of load profiles by means of the DSA model (cf. Chapt. 9).

A major emphasis of our study is on the most typical class of electricity customers, i.e., private, residential domestic customers. Moreover, differently from other related work, each customer load profile was segmented with respect to the type of day, which enabled a characterization of the customers' profiles on a per day basis.

We performed experiments by varying the clustering algorithm and the distance measure. More precisely, we used the standard K-Means and the Euclidean distance as baseline method. However, we also resort to the DTW (Dynamic Time Warping) distance, which is widely known to provide a better way to compare time series. Moreover, we introduce a simple top-down partitional algorithm, named *TS-Part*. A major feature of TS-Part is that, unlike the K-Means algorithm, it does not require the user to specify a desired number of output clusters.

Experimental results have shown that the DTW supports higher-quality clustering than the Euclidean distance, in terms of both cluster separation and compactness. The best performance corresponded to setting TS-Part with the DTW, which resulted in more clusters than those obtained by using the Euclidean distance. However, we observed that most of the data tend to group together in a relatively small number of clusters. This scenario enables the identification of relevant aspects which allow for supporting the design of tariff policies.

10.2.1 Low-voltage Electricity Customer Data

As discussed in the previous section, the Enel utility is capable of measuring and storing customer load profiles effectively. Such an expertize of Enel is summarized in the Telegestore project [BCR05, Rog07].

The Enel Telegestore architecture is shown in Fig. 10.10. Communication between meters and concentrator is accomplished by a PLC (Power Line Carrier) channel, whereas the public GPRS/GSM Network is responsible for the communication between the concentrator and the central system. All energy related data are first collected from the smart meters by the concentrator. Then, such data are uploaded by the central system. The Enel Telegestore network devices consist of more than 31 millions of smart meters and more than 350,000 concentrators installed and remotely managed [Rog07].

Enel smart meter is able to record and store active and reactive load profiles for all the four energy quadrants. A load profile represents the shape of the customer consumption chronologically ordered. Given a sampling period time, a smart meter logs the consumption corresponding to the associated location in a circular buffer. The sampling period is programmable and ranges from 1 to 60 minutes; as default, this is set to 15 minutes which leads to store 38 days of load data, where each day is 96-sample long.

The smart meter also stores a flag register of "sample validity" in the stream of load data. This flag indicates a critical fault occurred during the sample measurement (e.g., a voltage interruption). In the experimental evaluation, we used this register to identify wrong samples and to correct each of them by using linear interpolation between the previous and the next valid sample.

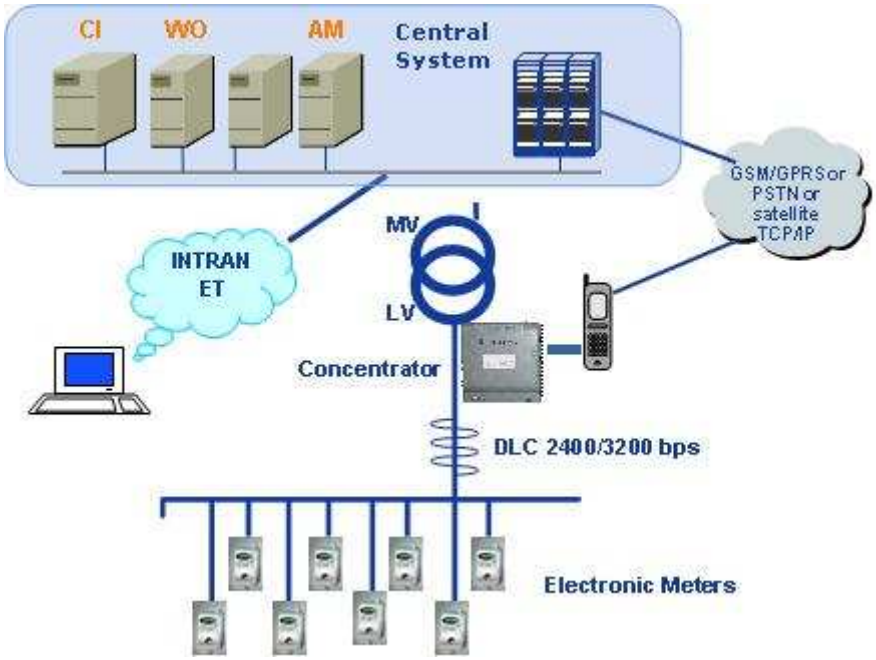


Fig. 10.10. The Enel Telegestore architecture

10.2.2 Clustering Load Profile Data

Algorithms

In this section, we describe the algorithms used for clustering load profiles. According to most of research works on clustering load profiles, we resort to the well-known paradigm of partitional relocation clustering, and exploit the K-Means algorithm (cf. Chapt. 2). We assume that the cluster centroids are computed as simple averages of the data (load profiles) in any specific cluster, since all the data have the same length in our context. Of course, this assumption does not hold in general, and more refined methods for computing cluster centroids in time series data might be used [GPTG09a].

We also propose a top-down partitional algorithm, named *TS-Part*. This algorithm follows a clustering strategy which is inspired to the AutoPart algorithm [Cha04]. A major feature of the AutoPart algorithm is that the number of output clusters is not required as a parameter, rather it is determined during the clustering task. This represents an advantage in many real application contexts, like ours, in which there is no a priori information which guides the user to properly set the number of output clusters.

The outline of TS-Part is reported in Alg. 10.1. TS-Part starts by considering the dataset D as a single cluster (Line 1), then two main steps are iteratively repeated until the convergence is reached (Lines 3-6). The first step

consists in finding the best split for each cluster in the current clustering $\mathcal{C}$ (Funct. 10.1). The second step recomputes the cluster centroids and reassigns all the data objects according to the current partition $\hat{\mathcal{C}}$; this step is accomplished like in the K-Means algorithm. The convergence of the algorithm is reached when the split procedure does not perform any split (Line 6).

Algorithm 10.1 TS-Part

Input: A set D of data objects
Output: A partition $\mathcal{C}$ of D
1: $\mathcal{C} \leftarrow D$
2: $\hat{\mathcal{C}} \leftarrow split(\mathcal{C})$ {Funct. 10.1}
3: **repeat**
4: $\quad \mathcal{C} \leftarrow relocation(\hat{\mathcal{C}})$
5: $\quad \hat{\mathcal{C}} \leftarrow split(\mathcal{C})$
6: **until** $\hat{\mathcal{C}} = \mathcal{C}$

In Funct. 10.1, the *quality* of a given clustering solution is computed as the difference between the inter-cluster distance and the intra-cluster distance (cf. Chapt. 2). The split operation hence depends on a threshold of minimum quality, which is initially set as the *quality* of the input clustering. Also, an *outlier* in a cluster is identified as an object whose distance from the associated centroid is maximum.

Cluster Validity

To assess the effectiveness of results provided by the clustering algorithms, we resorted to internal criteria to evaluate both compactness and separation of a clustering solution. In particular, we employed two of the most used criteria in load profile clustering, namely *Mean Index Adequacy* and *Clustering Dispersion Indicator* [CNP+05, CNP06, THD07, TSTK08]. Both criteria relies on information on the data to be clustered, the centroids of the clustering solution, and the number of desired clusters.

Let $D = \{X_1, \ldots, X_n\}$ be a dataset of time series and $\mathcal{C} = \{C_1, \ldots, C_K\}$ be a clustering solution for D. We denote with $\mathcal{V} = \{v_1, \ldots, v_K\}$ the set of centroids such that v_k is the centroid of the cluster $C_k, \forall k \in [1..K]$. We define the following distance measures:

$$\varpi(v_k, C_k) = \sqrt{\frac{1}{|C_k|} \sum_{X \in C_k} f(v_k, X)^2}$$

$$\vartheta(C_k) = \sqrt{\frac{1}{2n} \sum_{X \in D} \varpi(X, C_k)^2}$$

Function 10.1 split

Input: A set $\mathcal{C} = \{C_1, \ldots, C_K\}$ of clusters
Output: A set $\hat{\mathcal{C}}$ of K' clusters, where $K' \geq K$

$\hat{\mathcal{C}} \leftarrow \emptyset$
$q_0 \leftarrow quality(\mathcal{C})$
for all $C \in \mathcal{C}$ **do**
  $X^* \leftarrow outlier(C)$
  $C' \leftarrow C \setminus \{X^*\}, \quad q'_{max} \leftarrow quality(C')$
  $C'' \leftarrow \{X^*\}, \quad q''_{max} \leftarrow quality(C'')$
  $splittable \leftarrow false$
  for all $X \in C,\ X \neq X^*$ **do**
    $q' \leftarrow quality(C' \setminus \{X\})$
    $q'' \leftarrow quality(C'' \cup \{X\})$
    $gain' \leftarrow (q' - q'_{max})/q'_{max}$
    $gain'' \leftarrow (q'' - q''_{max})/q''_{max}$
    if $gain' > q_0 \ \vee \ gain'' > q_0$ **then**
      $q'_{max} \leftarrow q', \quad q''_{max} \leftarrow q''$
      $C' \leftarrow C' \setminus \{X\}, \quad C'' \leftarrow C'' \cup \{X\}$
      $splittable \leftarrow true$
    end if
  end for
  if *splittable* **then**
    $\hat{C} \leftarrow \hat{\mathcal{C}} \cup \{C'\} \cup \{C''\}$
  else
    $\hat{\mathcal{C}} \leftarrow \hat{\mathcal{C}} \cup \{C\}$
  end if
end for
return $\hat{\mathcal{C}}$

where $f(\cdot, \cdot)$ is a distance measure for comparing time series, e.g., Euclidean or DTW distance.

Based on the above formulas, the *Mean Index Adequacy* (MIA) and *Clustering Dispersion Indicator* (CDI) validity criteria are defined as follows:

$$MIA(\mathcal{C}) = \sqrt{\frac{1}{K} \sum_{k=1}^{K} \varpi(v_k, C_k)^2} \tag{10.1}$$

$$CDI(\mathcal{C}) = \frac{1}{\vartheta(\mathcal{V})} \sqrt{\frac{1}{K} \sum_{k=1}^{K} \vartheta(C_k)^2} \tag{10.2}$$

MIA measures the compactness of a clustering solution by averaging the distances between each object within a cluster and its centroid. CDI expresses the degree of cluster separation as directly proportional to the average of the intra-cluster distance between the objects within the same cluster and inversely proportional to the pair-wise distances between the cluster centroids.

For both criteria, lower values correspond to higher quality clustering solutions.

10.2.3 Experimental Evaluation

Data Description and Preparation

We were granted access to about 30,000 Enel Italian LV customer load profiles, measured during the period between the first week of February 2009 and the last week of March 2009. All the load profiles have been provided in anonymous form.

The load profile set was preliminarily partitioned according to meta-data associated to each individual customer. Such meta-data represents commercial and technical extra attributes that the Enel utility provided with each load profiles. Specifically, customer meta-data includes the following attributes:

- Meter type: specifies the power capacity and the number of phases (i.e, single-phase, multi-phase) of the meter associated to the customer;
- Contractual power: the maximum contractual power allowed to the customer;
- Contract date: the start date of the customer's contract;
- Commercial category: identifies the type of customer, including residential domestic, non-residential domestic, public lightning, etc.;
- Product category: identifies a particular (private or public) usage of the energy contract;
- Zone: geographical location of the customer.

According to the above information, we filtered in the available load profiles which correspond to the most common customer type, which is the "private", "residential domestic" customer. We also considered only the active energy type. The resulting 5,000 load profiles were segmented in order to extract *daily* profiles. Since each daily profile is comprised of 96 samples, we obtained 30 daily profiles of 96 samples from each customer profile. Moreover, daily profiles were further partitioned depending on the type of day; precisely, we distinguished "weekdays" profiles from "saturdays" profiles and "sundays/holidays" profiles. According to this classification, we found 132,041 "weekdays", 24,527 "saturdays", and 24,541 "sundays/holidays" daily profiles.

The Rialto Suite for Data Mining

Experiments and analysis described in this work were conducted using **Exeura Rialto**™ [Ria]. Rialto is a graphical environment for performing data mining and knowledge discovery tasks. Unlike other similar data mining tools, Rialto contains most of the functionalities required by one user-friendly tool that allows users to design, create, explore, analyze, and execute data

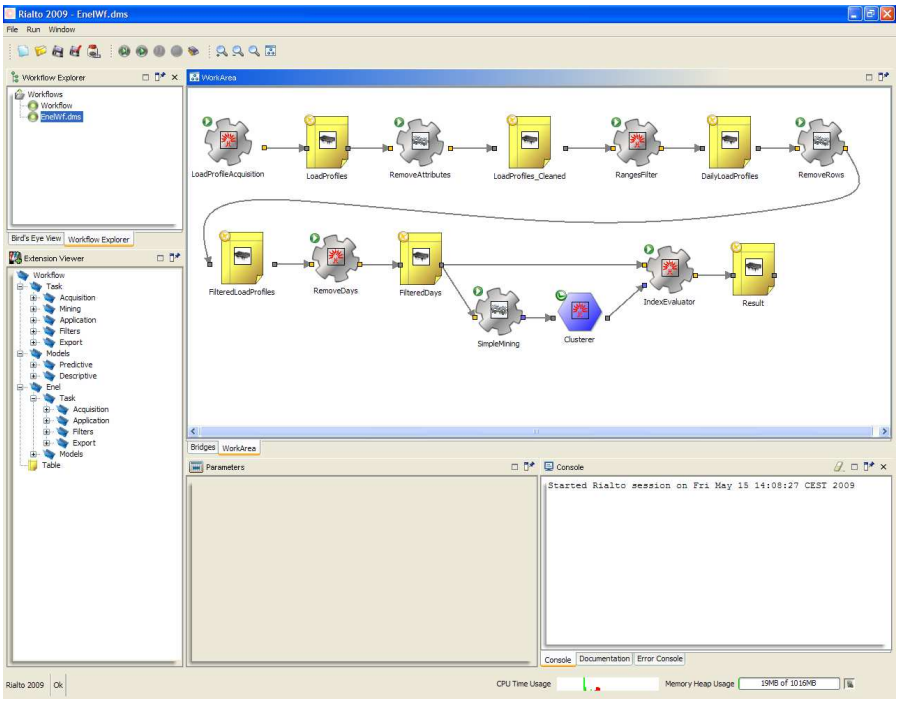


Fig. 10.11. A snapshot of the Exeura Rialto™ suite for data mining

mining tasks, as well as to deploy predictive and descriptive models into other tools, applications, and systems.

Rialto uses an intuitive, graphical interface for designing and developing workflows, i.e., sequences of functions and operations. Functions are represented by workflow nodes. A Rialto workflow is built using three types of nodes: tables, tasks, and models.

- Tables are used to store and manage data. It is conceptually similar to tables in a relational database.
- Tasks are operations used to gather new data, storing them in tables. They are also used to filter, transform, and apply data mining algorithms to the data stored in tables. Tasks can be executed sequentially or in parallel.
- Models are used to represent the results obtained by a mining task on a table.

A Java plug-in enables integration of domain-specific functionalities and development of tools that support the analytical frameworks for specific domains. Thanks to the possibility of extending the capabilities of Rialto, it was possible to generate a set of ad-hoc plug-ins for management of the data from the Enel legacy repositories. The implemented plug-ins concern acquisition, preprocessing, data analysis, statistics and visualization, data modeling, dis-

tance measures (i.e., Euclidean norm and DTW), and clustering algorithms (i.e., K-Means and TS-Part).

Preliminary Results

We present here main results from clustering experiments on the three types of daily load profile sets, namely "weekdays", "saturdays", and "sundays/holidays". For each of the three cases, we performed multiple runs of both clustering algorithms (i.e., K-Means and TS-Part) and finally averaged the quality results, in terms of MIA and CDI, obtained over the runs. Each algorithm was equipped with Euclidean distance or DTW as distance measure. For each setting, the number of clusters was determined by TS-Part and then used to set the parameter (i.e., initial value of the number of output clusters) for the K-Means.

clustering algorithm	*distance measure*	*# of clusters*	*MIA*	*CDI*
K-Means	Euclidean	10	9.775	0.682
TS-Part	Euclidean	10	9.103	0.914
K-Means	DTW	66	7.986	0.008
TS-Part	DTW	72	**5.514**	**0.004**

Table 10.2. Clustering load profile data: best (average) performance on Weekdays load profiles

clustering algorithm	*distance measure*	*# of clusters*	*MIA*	*CDI*
K-means	Euclidean	19	7.456	0.100
TS-part	Euclidean	19	6.520	0.124
K-means	DTW	31	12.942	0.010
TS-part	DTW	37	**10.310**	**0.009**

Table 10.3. Clustering load profile data: best (average) performance on Saturdays load profiles

Tables 10.2–10.4 summarize the best (average) performance of the clustering algorithms obtained on the three cases. Using the DTW as distance measure mostly enabled either clustering algorithm to produce higher quality clustering solutions with respect to the ones obtained by using the Euclidean distance. This always holds in terms of CDI for all the cases, and also in terms of MIA for the "weekdays" case (which corresponds to the largest set of daily load profiles). The better separation (CDI) obtained by using DTW reflects an increase in the number of clusters, which is explained by the fact that DTW

clustering algorithm	*distance measure*	*# of clusters*	*MIA*	*CDI*
K-means	Euclidean	17	9.964	0.109
TS-part	Euclidean	17	6.946	0.156
K-means	DTW	29	13.773	0.014
TS-part	DTW	32	**11.646**	**0.012**

Table 10.4. Clustering load profile data: best (average) performance on Sundays/holidays load profiles

is more sensitive than the Euclidean distance to time shifts and, consequently, it is capable of detecting more specific/descriptive clusters. The best results in each table correspond to the use of our TS-Part algorithm equipped with DTW. It should be noted that the better performance of TS-Part against K-Means concerns both MIA and CDI.

However, as we can see in Fig. 10.12 referring to TS-Part with DTW, most of the profile data were assigned to a relatively small number of clusters, whereas a significant part of clusters likely corresponds to untypical habits of certain residential domestic customers (e.g., customers spending most of their time in residences which are located in places different from those declared in the contract).

It should be emphasized that the presence of a few, large clusters attracts major attention from a perspective of tariff policy design; conversely, the many, small clusters will be probably discarded. We indeed observed that, in typical behaviors of residential domestic customers, most of the energy consumption is concentrated on the lunch and dinner hours.

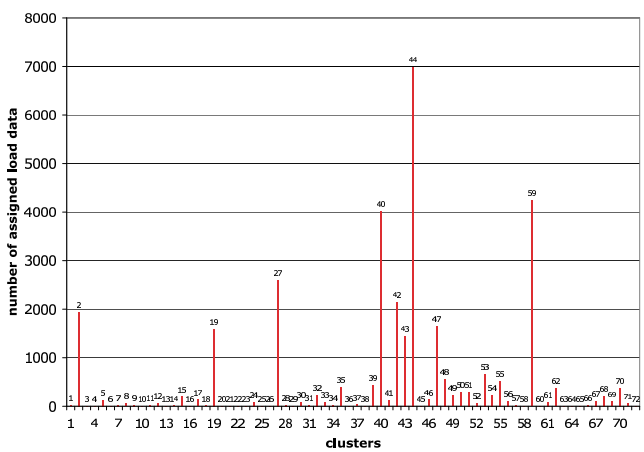


Fig. 10.12. Distribution of weekdays load profiles over clusters obtained by TS-Part with DTW

Part V

The Curse of Dimensionality in Data Clustering

– *Local Dimensionality Reduction* –

11

Projective Clustering: Background

Abstract Dimensionality reduction techniques that operate *locally* to each single cluster have been recognized as a valid and powerful solution to the problem of the curse of dimensionality in data clustering. Such techniques aim to discover one or more clustering solutions along with the subspaces associated to each cluster of the discovered clustering(s). The subspaces can be either *axis-aligned* or *arbitrarily-oriented*. According to the way how the local dimensionality is performed and the clustering solution(s) are built, major existing research works in this area may belong to either *subspace clustering* or *projective clustering* approaches. This chapter provides some background to the projective clustering problem, including a brief overview of the state-of-the-art.

11.1 Axis-aligned vs. Arbitrarily-oriented Subspaces

The typical issues arising from the curse of dimensionality in data clustering can be effectively overcome by performing a reduction of the dimensionality *locally* to the single cluster. This approach has been recognized as generally more effective than the classic global dimensionality reduction (cf. Chapt. 1).

The main goal of local dimensionality reduction techniques is to discover the cluster structure along with a set of *subspaces*, each one associated to a cluster of the discovered structure. The key idea underlined this approach is that, in a high dimensional representation (feature) space, the objects in a given cluster are highly similar to each other if (and only if) they are projected onto the subspace associated to the cluster they belong to.

A subspace assigned to any cluster can be *axis-aligned* (also known as *axis-parallel*) or *arbitrarily-oriented* (also known as *generalized*). The former refers to subspaces identified by hyperplanes parallel to the axes of the full representation space. They are represented by a subset of the features of the original feature space, and formally defined by orthogonal vectors $\mathbf{v} = [v_1, \ldots, v_d]$ such that $v_l \in \{0, 1\}$, $\forall l \in [1..d]$, and $\sum_{l=1}^{d} v_l = 1$ ($d < m$, where m is

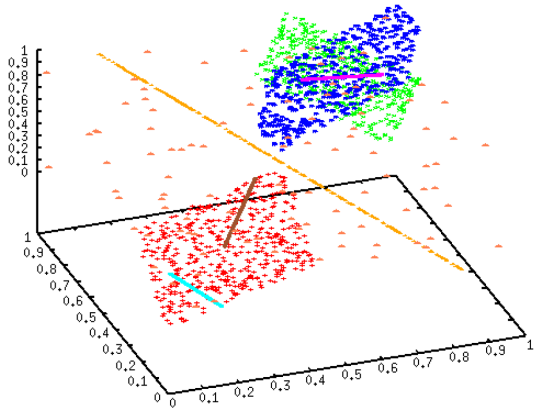


Fig. 11.1. Some projective clusters existing in arbitrarily-oriented subspaces

the dimensionality of the full representation space). Arbitrarily-oriented subspaces are identified by hyperplanes that may have a general orientation with respect to the axes of the full dimensional space. An example of clusters existing in arbitrarily-oriented subspaces is depicted in Fig. 11.1.[1] It should be noted that, given a full representation space Σ of dimensionality m, the number of all possible axis-aligned subspaces of Σ is $\mathcal{O}(2^m)$, whereas all possible arbitrarily-oriented subspaces of Σ are obviously an infinite number.

Several research works have recently addressed the problem of local dimensionality reduction considering arbitrarily-oriented subspaces [AY00, AM04, TXO05, GHPT05, BKKZ04, ABK+06a, ABK+06b, ABK+07b, ABK+07c, ABD+08]. However, the focus of this thesis is on axis-aligned subspaces.

11.2 Subspace Clustering vs. Projective Clustering

Local dimensionality reduction considering axis-aligned subspaces has been addressed by focusing on the *subspace clustering* [AGGR98] and *projective* (or *projected*) clustering [APW+99] problems. Although the two problems are strictly related to each other and the terms "subspace clustering" and "projective clustering" are not used in a unified way in the literature, there exist several differences between these problems.

The main goal of (axis-aligned) subspace clustering is to find all possible "interesting" subspaces of the original representation space, and, for each relevant subspace, discover the clusters existing in that subspace. As a major result, the output of any subspace clustering algorithm is a set of *subspace clusters*, i.e., clusters composed by a set of data objects and a set of features which identify the subspace. The output subspace clusters can be overlapping from

[1] Fig 11.1 is borrowed from [ABK+07b].

the point of view of both the associated subspaces and the object-to-cluster assignments (i.e., at a data clustering level). Existing subspace clustering algorithms [AGGR98, CFZ99, NGC01, KKK04, AKMS07, LLSW07] typically generate the subspaces to be analyzed in a *bottom-up* way. Indeed, they usually start from one-dimensional subspaces and consider, at each step, subspaces of dimensionality greater than that taken into account in the previous step. Subspace clustering algorithms differ to each other for the technique involved to generate subspaces, the strategies for pruning non interesting subspaces (e.g., *Apriori*-like strategies), and the approaches for discovering clusters given a subspace recognized as interesting [PHL04, KKZ09].

By contrast, projective clustering outputs only one cluster structure. A major peculiarity is that, like in the subspace clustering case, the output clusters are coupled with their corresponding subspaces (i.e., set of features). However, unlike subspace clustering, the clusters outputted by projective clustering may overlap only with respect to the associated subspaces.

11.3 Projective Clustering: Basic Definitions

The main goal of axis-aligned projective clustering is to discover a proper *projective clustering solution*, i.e., a set of *projective clusters*. A projective cluster is a pair composed by a subset of the input dataset D and a (axis-aligned) subspace of the original dimensional space in which data objects in D are originally represented.

Axis-aligned projective clustering approaches fall into two main categories. According to the first one, the subset of features that identifies the subspace at hand is not coupled with any further information. Thus, the feature-to-cluster assignment is considered as *equally weighted*, i.e., no information is provided to discriminate among the features of the subspace. A more refined model allows for *unequally weighted* feature-to-cluster assignment [DPGM04, DGM$^+$07, CJW08]. More precisely, a proper weight is assigned to each feature of the original full dimensional space, in order to somehow quantify the probability that such a feature belongs to the subspace at hand. In particular, the stronger the correlation of data objects in a cluster along a feature, the more the weight assigned to it.

Within this view, for the sake of generality, it is advisable to deal with projective clustering solutions that include both the equally and unequally weighted feature-to-cluster assignment models. Furthermore, for the same reason, hard as well as soft data object clustering should be taken into account (cf. Chapt. 2, Sect. 2.4). Thus, we hereinafter refer to the following general definition of projective clustering solution.

Definition 11.1 (projective clustering solution). *Let $D = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_n\}$ be a set of m-dimensional points (numerical vectors), where each $\boldsymbol{\omega} = (\omega_{i1}, \ldots, \omega_{im})$ represents a data object o_i, $\forall i \in [1..n]$. A* projective clustering solution $\mathcal{C}$ *defined over D is a triple $\langle \mathcal{L}, \Gamma, \Delta \rangle$:*

- $\mathcal{L} = \{\ell_1, \ldots, \ell_K\}$ *is a set of cluster labels which uniquely represent the* K *clusters*
- $\Gamma : \mathcal{L} \times D \rightarrow S_\Gamma$ *is a function which stores the probability that* $\boldsymbol{\omega}_i$ *belongs to the cluster labeled with* ℓ_k, $\forall k \in [1..K], i \in [1..n]$, *such that* $\sum_{k=1}^{K} \Gamma_{ki} = 1, \forall i \in [1..n]$, *where* Γ_{ki} *hereinafter refers to* $\Gamma(\ell_k, \boldsymbol{\omega}_i)$
- $\Delta : \mathcal{L} \times [1..m] \rightarrow [0, 1]$ *is a function which stores the probability that the* j*-th feature is a relevant dimension for the objects in the cluster labeled with* ℓ_k, $\forall k \in [1..K], j \in [1..m]$, *such that* $\sum_{j=1}^{m} \Delta_{kj} = 1, \forall k \in [1..K]$, *where* Δ_{kj} *hereinafter refers to* $\Delta(\ell_k, j)$

Note that Def. 11.1 comprises both the cases of soft and hard data object clustering, as well as the cases of feature-to-cluster assignments equally weighted and unequally weighted. More precisely, $\mathcal{C} = \langle \mathcal{L}, \Gamma, \Delta \rangle$ is a hard (resp., soft) projective clustering solution if $S_\Gamma = \{0, 1\}$ (resp., $S_\Gamma = [0, 1]$), whereas $\mathcal{C}$ has feature-to-cluster assignments equally weighted if, for any given cluster labeled with ℓ_k, $k \in [1..K]$, $\Delta_{kj} = \Delta_{kj'}, \forall j, j' \in [1..m]$.

11.4 Projective Clustering: State of the Art

Existing axis-aligned projective clustering algorithms can be classified in *bottom-up*, *top-down*, *soft*, and *hybrid* methods [KKZ09].

11.4.1 Bottom-up Methods

Bottom-up methods are based on two steps, i.e., finding subspaces recognized as "interesting", and assigning each data object to the most similar subspace. Thus, the projective cluster structure is computed by firstly searching, in a bottom-up way, for the subspaces to be associated to the discovered projective clusters.

The *Projected Clustering via Cluster Cores* (P3C) algorithm [MSE08] deals with numeric as well as categorical data and is designed to work in the case of projected clusters to be discovered on very few relevant dimensions. P3C is also able to compute overlapping projected clusters.

In [SZ04], the authors propose the *Support and Chernoff- Hoeffding bound-based Interesting Subspace Miner* (SCHISM) algorithm. SCHISM mines interesting subspaces rather than projective clusters, hence, it is not exactly a projective clustering algorithm, but solves a related problem: finding subspaces to look for clusters.

11.4.2 Top-down Methods

Top-down approaches aim to find the subspace to be associated to each cluster during the clustering stage, starting by taking into account the full dimensional space.

A top-down approach based on histograms is proposed by EPCH (*Efficient Projective Clustering by Histograms*) [KWC05], which identifies dense regions in each low-dimensional histogram. In [LXY00], the CLTree (*CLustering based on decision Trees*) algorithm assigns a common class label to all existing data points representing input objects and adds additional points uniformly distributed over the data space and labeled as a different class. Then, a decision tree is trained to separate the two classes.

Further approaches belong to hierarchical, partitional relocation, and density-based categories. Hierarchical algorithms are proposed in [YCN04, ABK+06a]. HARP (*a Hierarchical approach with Automatic Relevant dimension selection for Projected clustering* [YCN04] follows an AHC scheme with single link and requires two main parameters to control the cluster construction, which are the minimum number of selected dimensions and a threshold for selecting a dimension in a forming cluster. Unlike HARP, the *Hierarchical Subspace Clustering* (HiSC) algorithm [ABK+06a] produces a hierarchy of nested subspace clusters, i.e., a dendrogram storing relationships of lower dimensional subspace clusters that are embedded within higher-dimensional subspace clusters.

Partitional relocation methods [APW+99, YCN05] follows a classic relocation, alternating scheme (cf. Chapt. 2). PROCLUS (*PROjected CLUStering algorithm*) [APW+99] is a K-Medoids algorithm which makes a clustering initialization over the full dimensional space and, besides the number of desired clusters, requires a further parameter concerning the average dimensionality of cluster, which is not trivial to set. For this purpose, PROCLUS may fail in detecting clusters of very different sizes. Variants of PROCLUS comprises FINDIT (*a Fast and INtelligent subspace clustering algorithm using DImension voTing*) [WLKL04], which employs some heuristics to enhance efficiency and clustering accuracy, and SSPC (*SemiSupervised Projected Clustering*) [YCN05], which is able to further enhancing accuracy by using domain knowledge in the form of labeled objects and/or labeled attributes.

The *PreDeCon* algorithm proposed in [BKKK04] belongs to the class of density-based approaches. It exploits the density-based full dimensional clustering algorithm DBSCAN (cf. Chapt. 2) by using a specialized subspace distance measure that captures the subspace of each cluster.

11.4.3 Soft Methods

All the above methods provide clustering solutions which are hard at data clustering level and have feature-to-cluster assignments equally weighted. However, a recent corpus of study has focused on algorithms able to produce soft data clusterings [MSE08, CJW08], and/or clusterings having feature-to-cluster assignments unequally weighted [DGM+07, CJW08].

Locally Adaptive Clustering (LAC) [DGM+07] performs local feature selection in order to enable distance measures reflect local correlations of data. The soft feature selection procedure in LAC assigns weights to the features in

such a way that the stronger the correlation of data along a dimension, the more the weight assigned to it. The strength of this increase, which determines the incentive for clustering on more features, is controlled by a parameter $\tilde{h}$. However, at a data clustering level, LAC outputs hard clusters.

The very recent study proposed in [CJW08] focuses on a probabilistic modeling of projected clusters and proposes a *Fuzzy Projective Clustering* (FPC) algorithm, which can produce overlapping clusters, like P3C, and can also assign different weights to the subspace dimensions.

11.4.4 Hybrid Methods

Hybrid methods share some features with both actual projective clustering and subspace clustering methods. Indeed, they do not typically assign each data object to one cluster nor aim at finding all clusters in all interesting subspaces [KKZ09].

Most hybrid algorithms follows a density-based approach. *Density-based Optimal Projective Clustering* (DOC) [PJAM02] is a density-based algorithm that greedily discovers projected clusters. It can handle variable-size clusters and does not require the number of clusters as input parameter; however, it is sensitive to a different user-defined parameter required to control the cluster quality, and assumes that the projected clusters are hypercubes of same side-length over all dimensions. In [YM05], *MINECLUS* is proposed as an improvement to the efficiency of DOC based on an optimized adaptation of the frequent pattern tree growth method. The key idea is to model a data point (representing an input data object) as an itemset comprised of the dimensions in which the point is within a certain distance from a given pivot point.

DiSH (*Detecting Subspace cluster Hierarchies*) [ABK$^+$07a] follows a similar idea as PreDeCon but uses a hierarchical clustering model. DiSH exploits an algorithm that is inspired by the density-based hierarchical clustering algorithm OPTICS (cf. Chapt. 2).

FIRES (*FIlter REfinement Subspace clustering*) [KKRW05] computes one-dimensional clusters using any clustering technique provided in input by the user. These one-dimensional clusters are then merged by applying a "to cluster clusters" step. The clusters discovered by FIRES may overlap or not, but, usually, FIRES cannot produce all clusters in all interesting subspaces.

12

Projective Clustering Ensembles

Abstract Recent advances in clustering have been focused on *clustering ensembles* and *projective clustering* approaches, which distinctly aim to face typical issues in many clustering problems. However, such approaches are originally devised independently from each other. In this chapter, we address for the first time the *projective clustering ensembles* (PCE) problem, whose main goal is to derive a proper *projective consensus partition* from an ensemble of projective clustering solutions. We formalize PCE as an optimization problem which is designed to satisfy strong requirements on the independence on the specific clustering ensembles algorithm, ability to handle hard as well as soft data clustering, and different feature weightings. Specifically, we provide two formulations of PCE: as a two-objective optimization problem, in which the two objectives respectively account for the object-based and the feature-based representations of the solutions in the ensemble, and as a single-objective problem, in which the object-based and feature-based representation are embedded into a single function to measure the distance error between consensus partition and ensemble. Experiments have demonstrated the significance of the proposed methods for PCE, showing clear improvements in terms of accuracy of the output consensus partition.

12.1 Introduction

Research on clustering has traditionally assumed that, given a set of input data and a clustering problem for that data, *(i)* the problem at hand is addressed by a specific clustering method, which is usually equipped with a certain distance/similarity measure, and *(ii)* all the features (or dimensions) of the given data are considered during the clustering task.

The above assumptions are usually given for enabling a proposed approach to satisfy some special requirements for data clustering, such as simplicity, practical applicability, understandability of the results, and low computational cost. On the other hand, such assumptions may bring any clustering method to incur serious issues in both effectiveness and efficiency, especially when (1) the problem at hand is inherently multi-faceted as there is a number

of (differently relevant) aspects according to which a clustering task can be naturally performed, and/or (2) the input data is highly dimensional.

Issue 1 is related to the fact that a solution for the clustering problem is inevitably biased due to the peculiarities of the specific clustering algorithm being used; therefore, even though the chosen algorithm can be optimally tuned for a given input data or application domain, the structure of the discovered clusters will necessarily impact on the actual "usefulness" of the solution. Issue 2 is instead related to the so-called curse-of-dimensionality (cf. Chapt. 1), which breaks down the significance of the concept of proximity (thus, cluster) as the number of dimensions or features increases.

As discussed in the previous chapters, proper methodologies have been studied to distinctly address the above issues in clustering problems, orthogonally to the existing literature on clustering algorithms and data proximity measures. Clustering ensembles has recently emerged as a powerful tool to face issue 1 (cf. Chapt. 6), whereas a number of projective clustering techniques has been developed to overcome issue 2 (cf. Chapt. 11).

In this chapter, the problem of *projective clustering ensembles* (PCE) is addressed for the first time [GDT09]. The objective is to define methods for clustering ensembles that are able to deal with ensembles of projective clustering solutions and provide a projective consensus partition. In particular, we focus on ensembles composed by projective clustering solutions as defined in Def. 11.1.

The projective consensus partition to be discovered is computed as a solution of an optimization problem having one or more objective functions, which are defined by exploiting information available from the input ensemble. Moreover, since we are interested in developing general methods for PCE, the objective functions in the optimization problem should meet the following strong requirements: *(i)* to discard the original feature values of the input data; *(ii)* to be independent on the specific clustering algorithm and on any prior knowledge on the setup for ensemble generation; *(iii)* to handle hard as well as soft data clustering in a projective setting; *(iv)* to allow for feature-to-cluster assignments which are unequally weighted. Assigning feature to clusters with unequal weights enables solutions in which the subspaces associated to each cluster are expressed in terms of feature weights which are computed in such a way that the stronger the correlation of data along a dimension, the more the weight assigned to it.

Within this view, we propose two formulations of PCE in terms of optimization problem. More precisely, we propose a *two-objective* and a *single-objective* formulation of PCE: the first one involves two distinct objective functions which account for the data object clustering and feature-to-cluster assignment, respectively; the second formulation instead has a unique objective function which acts as an error criterion in the computation of any cluster (of a candidate clustering solution) by involving both the object-based representation and the feature-based representation of the cluster.

For each of the two proposed formulations of PCE, we develop well-founded heuristics, in which a *multi-objective evolutionary strategy* [Deb01] and an EM-like approach are employed. Experiments conducted on ten benchmark datasets have shown that both the proposed algorithms lead to more accurate consensus partitions, in terms of internal similarity with respect to reference classifications (i.e., external classifications and clustering ensembles) and in terms of intra-cluster error-rate.

We point out that, to the best of our knowledge, the PCE problem has not been investigated previously. Among the existing clustering ensemble methods in the literature (cf. Chapt. 6, Sect. 6.2), we think that WSBPA [DA09] is the only one that could be in principle compared to the approaches proposed in this chapter. However, WSBPA does not represent a valid solution for the PCE problem, since it does not satisfy any of the above requirements. Indeed, WSPA *(i)* requires to access the original features of the data objects, *(ii)* works only if the projective solutions are generated by running a specific projective clustering algorithm (i.e., LAC [DGM+07]), and *(iii)* does not deal with projective solutions that are soft at data clustering level.

12.2 Preliminaries

In this section, we provide the basic notions which shall be recalled during the explanation of the proposals of this chapter. Within this view, we hereinafter refer to a projective clustering solution as defined in Def. 11.1.

Definition 12.1 (projective ensemble). *Given a set D of data objects, a* projective ensemble *defined over D is a set $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$, where each $\mathcal{C}_h = \langle \mathcal{L}^{(h)}, \Gamma^{(h)}, \Delta^{(h)} \rangle$ is a projective clustering solution defined over D, $\forall h \in [1..H]$, and $\mathcal{L}^{(h)} \cap \mathcal{L}^{(h')} = \emptyset$, $\forall h, h' \in [1..H], h \neq h'$.*

Definition 12.2 (ensemble label set). *Let $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$ be a projective ensemble, where $\mathcal{C}_h = \langle \mathcal{L}^{(h)}, \Gamma^{(h)}, \Delta^{(h)} \rangle$, $\forall h \in [1..H]$. The* ensemble label set *of $\mathcal{E}$ is defined as $\mathbf{L} = \{\mathbf{l}_1, \ldots, \mathbf{l_H}\} = \bigcup_{h=1}^{H} \mathcal{L}^{(h)}$.*

Definition 12.3 (projective cluster representation). *Let $D = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_n\}$ be a set of m-dimensional data objects and $\mathcal{E}$ be a projective ensemble defined over D. The n-dimensional* object-based representation *and the m-dimensional* feature-based representation *for the cluster labeled with $\mathbf{l_h}$, $\forall \mathbf{h} \in [1..\mathbf{H}]$, are given by the vectors $\boldsymbol{\gamma}_\mathbf{h}$ and $\boldsymbol{\delta}_\mathbf{h}$, respectively, which are defined as follows:*

$$\boldsymbol{\gamma}_\mathbf{h} = (\gamma_{\mathbf{h}1}, \ldots, \gamma_{\mathbf{h}n}) = (\Gamma'_{k'1}, \ldots, \Gamma'_{k'n})$$

$$\boldsymbol{\delta}_\mathbf{h} = (\delta_{\mathbf{h}1}, \ldots, \delta_{\mathbf{h}m}) = (\Delta'_{k'1}, \ldots, \Delta'_{k'm})$$

where the Γ' and Δ' functions are involved in the solution $\mathcal{C}' \in \mathcal{E}$ such that $\mathcal{C}' = \langle \mathcal{L}', \Gamma', \Delta' \rangle$, $\mathcal{L}' = \{\ell'_1, \ldots, \ell'_{K'}\}$, $\mathbf{l_h} \in \mathcal{L}'$, and $k' \in [1..K']$ is the index such that $\ell'_{k'} = \mathbf{l_h}$.

12.3 Two-objective Projective Clustering Ensembles

A projective consensus partition $\mathcal{C}^* = \langle \mathcal{L}^*, \Gamma^*, \Delta^* \rangle$ derived from an ensemble $\mathcal{E}$ should meet two different kinds of requirements: the first one is related to the data object clustering of the solutions in $\mathcal{E}$, whereas the other one regards the feature-to-cluster assignment of the solutions in $\mathcal{E}$. For this purpose, the projective clustering ensembles problem can be naturally formulated as a two-objective optimization problem:

$$\mathcal{C}^* = \arg\min_{\hat{\mathcal{C}}} \Big[\Psi_o(\hat{\mathcal{C}}, \mathcal{E}, D),\ \Psi_f(\hat{\mathcal{C}}, \mathcal{E}, D) \Big] \tag{12.1}$$

where Ψ_o and Ψ_f are two optimization functions that account for the data clustering and the feature-to-cluster assignment of the projective clusterings in $\mathcal{E}$, respectively. Note that the only constraints for the above formulation are those required for guaranteeing that $\hat{\mathcal{C}}$ is a well-defined projective clustering solution (cf. Def. 11.1).

The functions Ψ_o and Ψ_f are defined using a *clustering-based* approach, which involves a comparison of the projective clustering solutions Formally, we have:

$$\Psi_o(\hat{\mathcal{C}}, \mathcal{E}, D) = \sum_{\mathcal{C} \in \mathcal{E}} \overline{\psi}_o(\mathcal{C}, \hat{\mathcal{C}}) \tag{12.2}$$

$$\Psi_f(\hat{\mathcal{C}}, \mathcal{E}, D) = \sum_{\mathcal{C} \in \mathcal{E}} \overline{\psi}_f(\mathcal{C}, \hat{\mathcal{C}}) \tag{12.3}$$

where $\overline{\psi}_o(\mathcal{C}_h, \mathcal{C}_{h'})$ (resp., $\overline{\psi}_f(\mathcal{C}_h, \mathcal{C}_{h'})$) is a function that measures the distance between the projective clustering solutions $\mathcal{C}_h = \langle \mathcal{L}^{(h)}, \Gamma^{(h)}, \Delta^{(h)} \rangle$ and $\mathcal{C}_{h'} = \langle \mathcal{L}^{(h')}, \Gamma^{(h')}, \Delta^{(h')} \rangle$ in terms of their corresponding object-based partitioning (resp., feature-to-cluster assignment):

$$\overline{\psi}_o(\mathcal{C}_h, \mathcal{C}_{h'}) = \frac{1}{2}\Big(\psi_o(\mathcal{C}_h, \mathcal{C}_{h'}) + \psi_o(\mathcal{C}_{h'}, \mathcal{C}_h) \Big) \tag{12.4}$$

$$\overline{\psi}_f(\mathcal{C}_h, \mathcal{C}_{h'}) = \frac{1}{2}\Big(\psi_f(\mathcal{C}_h, \mathcal{C}_{h'}) + \psi_f(\mathcal{C}_{h'}, \mathcal{C}_h) \Big) \tag{12.5}$$

where

$$\psi_o(\mathcal{C}_h, \mathcal{C}_{h'}) = \frac{1}{|\mathcal{L}^{(h)}|} \sum_{k=1}^{|\mathcal{L}^{(h)}|} \Big(1 - \max_{k' \in [1..|\mathcal{L}^{(h')}|]} J\big(\mathbf{a}_k^{(h)}, \mathbf{a}_{k'}^{(h')}\big) \Big)$$

$$\psi_f(\mathcal{C}_h, \mathcal{C}_{h'}) = \frac{1}{|\mathcal{L}^{(h)}|} \sum_{k=1}^{|\mathcal{L}^{(h)}|} \Big(1 - \max_{k' \in [1..|\mathcal{L}^{(h')}|]} J\big(\mathbf{b}_k^{(h)}, \mathbf{b}_{k'}^{(h')}\big) \Big)$$

with

$$\mathbf{a}_z^{(y)} = \big(\Gamma_{z1}^{(y)}, \ldots, \Gamma_{zn}^{(y)} \big)$$

$$\mathbf{b}_z^{(y)} = \big(\Delta_{z1}^{(y)}, \ldots, \Delta_{zn}^{(y)} \big)$$

and

$$J(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}\ \mathbf{v}^T}{\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - \mathbf{u}\ \mathbf{v}}$$

ranging within $[0, 1]$ and denoting the extended Jaccard similarity coefficient (also known as Tanimoto coefficient) between two any real-valued vectors $\mathbf{u}$ and $\mathbf{v}$ [JD88].

The MOEA-PCE Algorithm

The NP-hard problem P defined in (12.1) is a multi-objective optimization problem, in which the objectives are conflicting with each other; consequently, it is particularly hard to solve, since traditional optimization techniques do not apply. The conventional approach for solving this kind of problems consists in re-defining it as a single-objective problem, whose optimization function is computed as a weighted linear combination of the functions in the original problem. Unfortunately, this approach has several drawbacks, such as, e.g., mixing non-commensurable objectives, hard setting of the weight to assign to each function, prior knowledge of the application domain [Deb01]. A more refined approach that has been recognized as particularly appropriate in providing valid solutions for the problem at hand is given by the *Multi Objective Evolutionary Algorithms (MOEAs)* [Deb01]. This class of methods is able to solve a multi-objective problem maintaining the underlined multi-objective structure of the given problem, i.e., without combining the various objective functions into a single one.

Within this view, in order to provide a valuable heuristic for P, we resort to the MOEAs domain and define the proposed *MOEA-based Projective Clustering Ensembles (MOEA-PCE)* algorithm. In particular, we exploit the elitist MOEA *Nondominated Sorting Genetic Algorithm-II (NSGA-II)* [DPAM02], whose evolutionary strategy is based on the notion of *Pareto-ranking*.

Definition 12.4 (domination). *Let P be a multi-objective optimization problem of the form*

$$\left\{ x^* = \arg\min_{\hat{x}} [f_1(\hat{x}), \ldots, f_s(\hat{x})] \right\}$$

and x' and x'' two candidate solutions of P. x' dominates *x'' ($x' \prec x''$) if and only if*

$$f_i(x') \leq f_i(x''), \quad \forall i \in [1..s], \quad \textit{and}$$
$$\exists j \in [1..s] \ : \ f_j(x') < f_j(x'')$$

Definition 12.5 (Pareto-optimality). *Let P be a multi-objective optimization problem of the form*

$$\left\{ x^* = \arg\min_{\hat{x}} [f_1(\hat{x}), \ldots, f_s(\hat{x})] \right\}$$

and $\mathcal{S}$ a population of individuals for P, i.e., a set of candidate solutions of P. $\mathcal{S}_P^* \subseteq \mathcal{S}$ is a Pareto-optimal *solution set of P with respect to $\mathcal{S}$ if and only if*

$$x \not\prec x^*, \quad \forall x \in \mathcal{S}, \forall x^* \in \mathcal{S}_P^*$$

Definition 12.6 (Pareto-ranking). *Let P be a multi-objective optimization problem of the form*

$$\left\{ x^* = \arg\min_{\hat{x}} [f_1(\hat{x}), \ldots, f_s(\hat{x})] \right\}$$

and $\mathcal{S}$ a population of individuals for P. The Pareto-ranking *function $\eta : \mathcal{S} \rightarrow \mathbb{N}$ for P is defined as*

$$\eta(x) = \min\{r \in \mathbb{N}, r > 0 \ : \ x \in \mathcal{S}_{P,r}^*\}, \quad \forall x \in \mathcal{S}$$

where $\mathcal{S}_{P,z}^*$ is the Pareto-optimal solution set of P with respect to the population

$$\mathcal{S}_{P,z} = \{x' \in S \ : \ \eta(x') \geq z\}$$

The Pareto-ranking function η is exploited by NSGA-II algorithm for performing the classical MOEA steps of *selection*, *crossover*, and *mutation*. Such steps put also the basis for the proposed *MOEA-PCE algorithm*, whose outline is given in Alg. 12.1.

Algorithm 12.1 MOEA-PCE

Input: a projective ensemble $\mathcal{E}$ defined over a set D of data objects; the number K of clusters in the output projective consensus partitions; the population size t; the maximum number I of iterations

Output: a set $\mathcal{S}^*$ of projective consensus partitions

1: $\mathcal{S} \leftarrow populationRandomGen(\mathcal{E}, t, K)$
2: $it \leftarrow 1$
3: **repeat**
4: $\quad \eta \leftarrow computeParetoRanking(\mathcal{S})$ {cf. Def. 12.6}
5: $\quad \langle \mathcal{S}', \mathcal{S}'' \rangle \leftarrow \langle \check{\mathcal{S}}' \subset \mathcal{S}, \ \check{\mathcal{S}}'' \subset \mathcal{S} \rangle \ : \ |\check{\mathcal{S}}'| = |\mathcal{S}|/2, \ |\check{\mathcal{S}}''| = |\mathcal{S}|/2, \ \check{\mathcal{S}}' \cup \check{\mathcal{S}}'' = \mathcal{S}, \ \eta(x') \leq \eta(x''), \forall x' \in \check{\mathcal{S}}', x'' \in \check{\mathcal{S}}''$
6: $\quad \mathcal{S}'_{CM} \leftarrow crossoverAndMutation(\mathcal{S}')$
7: $\quad \mathcal{S} \leftarrow \mathcal{S}' \cup \mathcal{S}'_{CM}$
8: $\quad it \leftarrow it + 1$
9: **until** $it = I$
10: $\eta \leftarrow computeParetoRanking(\mathcal{S})$
11: $\mathcal{S}^* \leftarrow \{x' \in \mathcal{S} \ : \ \eta(x') \leq \eta(x''), \forall x'' \in \mathcal{S}, x'' \neq x'\}$

The algorithm starts by randomly generating the initial population $\mathcal{S}$ of t individuals (Line 1), and proceeds by performing the main loop until a maximum number I of iterations has been reached (Lines 3-9). At each iteration,

the Pareto-ranking function η, defined with respect to the current population $\mathcal{S}$, is computed according to Def. 12.6, where the problem denoted with P is the one reported in (12.1) (Line 4). The specific procedure used for computing η is the one described in [DPAM02]. The η values of each individual in $\mathcal{S}$ are then exploited for sorting $\mathcal{S}$ and partitioning it into two equal-sized subsets, i.e., $\mathcal{S}'$ and $\mathcal{S}''$, so that each individual in $\mathcal{S}'$ has a η value not greater than any other individual in $\mathcal{S}''$ (Line 5). The subset $\mathcal{S}'$ is involved into a crossover-and-mutation step, which is performed as described in [SD94] (Line 6). In particular, the mutation step consists in adding random Gaussian noise to each solution in $\mathcal{S}'$. The result of this step is the "offspring" set $\mathcal{S}'_{CM}$ of new individuals, which, together with $\mathcal{S}'$, forms the new population (Line 7). Finally, the Pareto-optimal solution set $\mathcal{S}^*$ (i.e., the set of output projective consensus partitions) is derived from the population $\mathcal{S}$ computed at the last iteration of the algorithm (Line 11).

Computational Aspects.

Let us now discuss the computational complexity of Alg. 12.1, given a set D of n m-dimensional data objects, a projective ensemble $\mathcal{E}$ of size H defined over D, the number K of clusters in the output projective consensus partitions, the size t of the population, and the number I of maximum iterations. Also, we assume that the total number of cluster labels in $\mathcal{E}$ (i.e., the size of the set $\mathbf{L}$, cf. Sect 12.1) is $\mathcal{O}(K\ H)$. The costs of the various steps of Alg. 12.1 are summarized as follows:

- the random initialization step (Line 1) is $\mathcal{O}(t\ K\ (n+m))$;
- the *computeParetoRanking* function (Line 4) comprises two steps: *(i)* computing the values of the functions Ψ_o (cf. (12.2)) and Ψ_f (cf. (12.3)) for each new individual in $\mathcal{S}$, which is $\mathcal{O}(t\ H\ K^2\ (n+m))$, and *(ii)* computing the η values for $\mathcal{S}$, which is performed in $\mathcal{O}(t^2)$, according to the procedure described in [DPAM02]. Therefore, since t is $\mathcal{O}(H)$, the total cost for the step at Line 4 is $\mathcal{O}(t\ H\ K^2\ (n+m))$;
- partitioning $\mathcal{S}$ into the subsets $\mathcal{S}'$ and $\mathcal{S}''$ (Line 5) is $\mathcal{O}(t\ \log t)$;
- the crossover & mutation operations (Line 6) are performed in $\mathcal{O}(t\ K\ (n+m))$;
- computing the output set $\mathcal{S}^*$ (Lines $10-11$) is $\mathcal{O}(t\ H\ K^2\ (n+m))$.

In conclusion, since the steps of the main loop (Lines 3-9) are repeated I times, we can state that Alg. 12.1 works in $\mathcal{O}(I\ t\ H\ K^2\ (n+m))$.

12.4 Single-objective Projective Clustering Ensembles

The two-objective projective clustering ensembles formulation may incur issues concerning the parameter setting and the interpretation of the convergence criterion. Within this view, we alternatively propose a different formulation which is computationally lower than MOEA-PCE in many practical

cases, and is based on a single objective function which applies to the whole information coming from the input ensemble $\mathcal{E}$:

$$\mathcal{C}^* = \arg\min_{\hat{\mathcal{C}}} Q(\hat{\mathcal{C}}, \mathcal{E}) \tag{12.6}$$

$$s.t.$$

$$\sum_{k=1}^{K} \hat{\Gamma}_{ki} = 1, \quad \forall i \in [1..n] \tag{12.7}$$

$$\sum_{j=1}^{m} \hat{\Delta}_{kj} = 1, \quad \forall k \in [1..K] \tag{12.8}$$

$$\hat{\Gamma}_{ki} \geq 0, \ \hat{\Delta}_{kj} \geq 0, \quad \forall k \in [1..K], i \in [1..n], j \in [1..m] \tag{12.9}$$

where

$$Q(\hat{\mathcal{C}}, \mathcal{E}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \hat{\Gamma}_{ki}^{\alpha} \sum_{\mathbf{h}=1}^{\mathbf{H}} \gamma_{\mathbf{h}i} \sum_{j=1}^{m} \left(\hat{\Delta}_{kj} - \delta_{\mathbf{h}j} \right)^2 \tag{12.10}$$

and $\alpha > 1$ is an integer that guarantees the nonlinearity of Q with respect to $\hat{\Gamma}_{ki}$, which is needed for ensuring that the values of $\hat{\Gamma}_{ki}$ range within $[0, 1]$ (instead of $\{0, 1\}$), for any solution of the problem P defined in (12.6)-(12.9).[1]

Let us now discuss the rationale underlying the definition of the optimization function Q. Essentially, Q measures the error of representing any cluster labeled with $\hat{\ell}_k$ of the candidate clustering solution $\hat{\mathcal{C}}$ by means of the $\hat{\Gamma}_{ki}$ (object-based representation) and $\hat{\Delta}_{kj}$ (feature-based representation) values; such errors are summed up over all the clusters of $\hat{\mathcal{C}}$. The error of representing a cluster labeled with $\hat{\ell}_k$ by means of $\hat{\Gamma}_{ki}$ and $\hat{\Delta}_{kj}$ is measured according to the information available from the ensemble $\mathcal{E}$; precisely, it is computed by considering, for each object $\boldsymbol{\omega}_i$ in D, the sum of the (weighted) squared Euclidean distances between the feature-based representations of cluster $\hat{\ell}_k$ (i.e., $\hat{\Delta}_{kj}$ values) and any cluster belonging to the solutions in $\mathcal{E}$ (i.e., $\delta_{\mathbf{h}j}$ values, for cluster labeled with $\mathbf{l}_\mathbf{h}$); the weight for each of these distances is computed as the probability that $\boldsymbol{\omega}_i$ belongs to both clusters $\hat{\ell}_k$ and $\mathbf{l}_\mathbf{h}$, i.e., the product of $\hat{\Gamma}_{ki}$ and $\gamma_{\mathbf{h}i}$. The final error for the cluster labeled with $\hat{\ell}_k$ is computed as the sum of such weighted distances over all the objects in D. Clearly, it is reasonable that a well-defined projective consensus partition should minimize the function Q, since its clusters would have a low (properly weighted) distance from each other cluster of the various solutions in the ensemble, in terms of both object-based and feature-based representations.

[1] Another alternative way to obtain $\hat{\Gamma}_{ki} \in [0, 1]$ is given by the introduction of properly-defined regularization terms (see, e.g., [LM99]).

Algorithm 12.2 EM-PCE

Input: a projective ensemble $\mathcal{E}$ defined over a set D of data objects; the number K of clusters in the output projective consensus partition;
Output: the projective consensus partition $\mathcal{C}^*$

1: $\mathcal{L}^* \leftarrow \{1, \ldots, K\}$
2: $\langle \Gamma^*, \Delta^* \rangle \leftarrow randomGen(\mathcal{E}, K)$
3: **repeat**
4: compute Γ^* according to (12.11)
5: compute Δ^* according to (12.12)
6: **until** *convergence*
7: $\mathcal{C}^* = \langle \mathcal{L}^*, \Gamma^*, \Delta^* \rangle$

The EM-PCE Algorithm

In order to provide a heuristic solution for the NP-hard problem defined in (12.6)-(12.9), we define a novel procedure that is inspired to the popular *Expectation Maximization (EM)* algorithm [DLR77].

The proposed algorithm, i.e., *EM-based Projective Clustering Ensembles (EM-PCE)* (Alg. 12.2), consists of two main EM-like steps, which are iteratively repeated until a convergence criterion is met. Such steps exploit the function Q (cf.(12.10)) and aim to find an optimal solution for $\hat{\Gamma}_{ki}$ (resp., $\hat{\Delta}_{kj}$) values, while maintaining fixed $\hat{\Delta}_{kj}$ (resp., $\hat{\Gamma}_{ki}$) values. The basic equations for the two steps are:

$$\Gamma^*_{ki} = \left[\sum_{k'=1}^{K} \left(\frac{\Lambda_{ki}}{\Lambda_{k'i}} \right)^{\frac{1}{\alpha-1}} \right]^{-1} \tag{12.11}$$

$$\Delta^*_{kj} = \frac{\Phi_{kj}}{\Xi_k} \tag{12.12}$$

where

$$\Lambda_{ki} = \sum_{\mathbf{h}=1}^{\mathbf{H}} \gamma_{\mathbf{h}i} \sum_{j=1}^{m} \left(\hat{\Delta}_{kj} - \delta_{\mathbf{h}j} \right)^2$$

$$\Xi_k = \sum_{i=1}^{n} \hat{\Gamma}^{\alpha}_{ki} \sum_{\mathbf{h}=1}^{\mathbf{H}} \gamma_{\mathbf{h}i}$$

$$\Phi_{kj} = \sum_{i=1}^{n} \hat{\Gamma}^{\alpha}_{ki} \sum_{\mathbf{h}=1}^{\mathbf{H}} \gamma_{\mathbf{h}i} \; \delta_{\mathbf{h}j}$$

Let us now derive the expressions reported in (12.11) and (12.12), i.e., the solutions for the problem P defined in (12.6)-(12.9). Since P has inequality constraints (cf. (12.9)), in order to find the optimal solution by means of the conventional Lagrange multipliers method, we consider the relaxed problem

P' obtained by temporarily discarding the inequality constraints from the constraint set of P.

We define the new (unconstrained) objective function Q_λ for P' as follows:

$$Q_\lambda(\hat{\mathcal{C}}, \mathcal{E}) = Q(\hat{\mathcal{C}}, \mathcal{E}) + \sum_{i=1}^{n} \lambda'_i \left(\sum_{k'=1}^{K} \hat{\Gamma}_{k'i} - 1 \right) + \sum_{k=1}^{K} \lambda''_k \left(\sum_{j'=1}^{m} \hat{\Delta}_{kj'} - 1 \right) \quad (12.13)$$

For a fixed assignment of $\hat{\Delta}_{kj}$, we compute the optimal Γ^*_{ki} by solving the following system of equations:

$$\frac{\partial\, Q_\lambda}{\partial\, \hat{\Gamma}_{ki}} = \alpha\, (\hat{\Gamma}_{ki})^{\alpha-1}\, \Lambda_{ki} + \lambda'_i = 0 \quad (12.14)$$

$$\frac{\partial\, Q_\lambda}{\partial\, \lambda'_i} = \sum_{k'=1}^{K} \hat{\Gamma}_{k'i} - 1 = 0 \quad (12.15)$$

Solving (12.14) with respect to $\hat{\Gamma}_{ki}$ and substituting such a solution in (12.15), we obtain:

$$\sum_{k'=1}^{K} \left(\frac{-\lambda'_i}{\alpha\, \Lambda_{k'i}} \right)^{\frac{1}{\alpha-1}} = 1 \quad (12.16)$$

Solving (12.16) with respect to λ'_i and substituting such a solution in (12.14), we obtain:

$$\alpha\, (\hat{\Gamma}_{ki})^{\alpha-1}\, \Lambda_{ki} - \left[\sum_{k'=1}^{K} \left(\frac{1}{\alpha\, \Lambda_{k'i}} \right)^{\frac{1}{\alpha-1}} \right]^{-(\alpha-1)} = 0 \quad (12.17)$$

Finally, solving (12.17) with respect to $\hat{\Gamma}_{ki}$, we obtain the optimal solution Γ^*_{ki} for P', whose expression is exactly equal to that reported in (12.11):

$$\Gamma^*_{ki} = \left[\sum_{k'=1}^{K} \left(\frac{\Lambda_{ki}}{\Lambda_{k'i}} \right)^{\frac{1}{\alpha-1}} \right]^{-1} \quad (12.18)$$

Analogously, for a fixed assignment of $\hat{\Gamma}_{ki}$, we compute the optimal Δ^*_{kj} by solving the following equations:

$$\frac{\partial\, Q_\lambda}{\partial\, \hat{\Delta}_{kj}} = \sum_{i=1}^{n} \hat{\Gamma}^{\alpha}_{ki} \sum_{\mathbf{h}=1}^{\mathbf{H}} 2\, \gamma_{\mathbf{h}i} \left(\hat{\Delta}_{kj} - \delta_{\mathbf{h}j} \right) + \lambda''_k = 0 \quad (12.19)$$

$$\frac{\partial\, Q_\lambda}{\partial\, \lambda''_k} = \sum_{j'=1}^{m} \hat{\Delta}_{kj'} - 1 = 0 \quad (12.20)$$

Equation (12.19) is equal to:

$$2\, \hat{\Delta}_{kj}\, \Xi_k - 2\, \Phi_{kj} + \lambda''_k = 0 \quad (12.21)$$

Thus, solving (12.21) with respect to $\hat{\Delta}_{kj}$ and substituting such a solution in (12.20), we obtain:

$$\sum_{j'=1}^{m} \left(\frac{2\ \Phi_{kj'} - \lambda''_k}{2\ \Xi_k} \right) = 1 \tag{12.22}$$

Solving (12.22) with respect to λ''_k and substituting such a solution in (12.21), we obtain:

$$2\ \hat{\Delta}_{kj}\ \Xi_k - 2\ \Phi_{kj} + \frac{2}{m} \left(\sum_{j'=1}^{m} \Phi_{kj'} - \Xi_k \right) = 0 \tag{12.23}$$

Finally, as it can easily be proved that $\sum_{j'=1}^{m} \Phi_{kj'}/\Xi_k = 1$, (12.23) can be solved with respect to $\hat{\Delta}_{kj}$ to obtain the optimal Δ^*_{kj} solution for P', whose expression is exactly equal to that reported in (12.11):

$$\Delta^*_{kj} = \frac{\Phi_{kj}}{\Xi_k} + \frac{1}{m} \left(\sum_{j'=1}^{m} \frac{\Phi_{kj'}}{\Xi_k} - 1 \right) = \frac{\Phi_{kj}}{\Xi_k} \tag{12.24}$$

In can be trivially noted that, according to the solutions for the problem P' that have been just derived (reported in (12.18) and (12.24)), it holds that $\Gamma^*_{ki} \geq 0$, $\Delta^*_{kj} \geq 0$, $\forall k \in [1..K], i \in [1..n], j \in [1..m]$. Therefore, such solutions satisfy the inequality constraints (reported in (12.9)) that were temporarily discarded in order to define the relaxed problem P'; thus, they represent the optimal solutions of the original problem P.

Proposition 12.7. *Algorithm 12.2 converges to a local optimum of the function Q defined in (12.10).*

Proof. According to the expression reported in (12.10), the value of function Q at the r-th iteration of Alg. 12.2 (for short, $Q^{(r)}$) can be expressed as a function of three terms:

$$Q^{(r)} = f(\mathbf{G}^{(r)}, \mathbf{D}^{(r)}, \mathcal{E})$$

where $\mathcal{E}$ is the input projective ensemble, and $\mathbf{G}^{(r)}$ and $\mathbf{D}^{(r)}$ are a $K \times n$ and $K \times m$ matrices, respectively, whose elements are defined as $\mathbf{G}^{(r)}_{ki} = \Gamma^{(r)}_{ki}$, $\mathbf{D}^{(r)}_{kj} = \Delta^{(r)}_{kj}$, $\forall k \in [1..K], i \in [1..n], j \in [1..m]$, with $\Gamma^{(r)}_{ki}$ and $\Delta^{(r)}_{kj}$ denoting the values Γ^*_{ki} and Δ^*_{kj} computed according to (12.11) and (12.12) at the r-th iteration of the algorithm, respectively.

According to how (12.11) has been derived, the first step of the main cycle of the algorithm (Line 4) computes the values in the matrix $\mathbf{G}$ at the $(r+1)$-th iteration, i.e., $\mathbf{G}^{(r+1)}$, in such a way that:

$$\mathbf{G}^{(r+1)} = \arg\min_{\check{\mathbf{G}}} f(\check{\mathbf{G}}, \mathbf{D}^{(r)}, \mathcal{E})$$

Thus, it holds that

$$f(\mathbf{G}^{(r+1)}, \mathbf{D}^{(r)}, \mathcal{E}) \leq f(\check{\mathbf{G}}, \mathbf{D}^{(r)}, \mathcal{E}),\ \forall\ \check{\mathbf{G}}$$

and, in particular:

$$f(\mathbf{G}^{(r+1)}, \mathbf{D}^{(r)}, \mathcal{E}) \leq f(\mathbf{G}^{(r)}, \mathbf{D}^{(r)}, \mathcal{E}) \tag{12.25}$$

Analogously, according to how (12.12) has been derived, the second step of the main cycle of the algorithm (Line 5) computes the values in the matrix $\mathbf{D}$ at the $(r+1)$-th iteration, i.e., $\mathbf{D}^{(r+1)}$, in such a way that:

$$\mathbf{D}^{(r+1)} = \arg\min_{\check{\mathbf{D}}} f(\mathbf{G}^{(r+1)}, \check{\mathbf{D}}, \mathcal{E})$$

Thus, it holds that

$$f(\mathbf{G}^{(r+1)}, \mathbf{D}^{(r+1)}, \mathcal{E}) \leq f(\mathbf{G}^{(r+1)}, \check{\mathbf{D}}, \mathcal{E}),\ \forall\ \check{\mathbf{D}}$$

and, in particular:

$$f(\mathbf{G}^{(r+1)}, \mathbf{D}^{(r+1)}, \mathcal{E}) \leq f(\mathbf{G}^{(r+1)}, \mathbf{D}^{(r)}, \mathcal{E}) \tag{12.26}$$

Due to the results reported in (12.25) and (12.26), it holds that

$$f(\mathbf{G}^{(r+1)}, \mathbf{D}^{(r+1)}, \mathcal{E}) \leq f(\mathbf{G}^{(r+1)}, \mathbf{D}^{(r)}, \mathcal{E}) \leq f(\mathbf{G}^{(r)}, \mathbf{D}^{(r)}, \mathcal{E})$$

Therefore, since $f(\mathbf{G}^{(r+1)}, \mathbf{D}^{(r+1)}, \mathcal{E}) = Q^{(r+1)}$ and $f(\mathbf{G}^{(r)}, \mathbf{D}^{(r)}, \mathcal{E}) = Q^{(r)}$, then:

$$Q^{(r+1)} \leq Q^{(r)} \tag{12.27}$$

The result in (12.27) proves that Alg. 12.2 performs a descendant gradient over the function Q, i.e., the value of Q at the $(r+1)$-th iteration of the algorithm is lower than or equal to the value of Q at the previous iteration. Furthermore, as the domain of definition of function Q is limited by the constraints reported in (12.7)-(12.9), the execution of the algorithm will necessarily terminate when a fixed point (i.e., a local minimum of Q) will be reached, i.e., when $Q^{(r^*)} = Q^{(r^*-1)}$ will hold at the r^*-th iteration. □

Computational Aspects

Let D be a set of n m-dimensional data objects, $\mathcal{E}$ be a projective ensemble of size H defined over D, and K a positive integer representing the number of clusters in the output projective consensus partition. Assuming that $\mathbf{H}$ is $\mathcal{O}(K\ H)$, it holds that both (12.11) and (12.12) are computed in $\mathcal{O}(H\ K^2\ n\ m)$; therefore, the computational complexity of Alg. 12.2 is $\mathcal{O}(I\ H\ K^2\ n\ m)$, where I is the number of iterations needed for the convergence of the algorithm.

12.5 Experimental Evaluation

We devised an experimental evaluation in order to assess the accuracy of the consensus partitions obtained by the proposed algorithms MOEA-PCE and EM-PCE. We recall that such algorithms are the first attempts to solve the projective clustering ensembles problem introduced in this chapter; thus, the comparison did not involve any other technique. In the following, we first discuss the evaluation methodology used in this work, which includes the selected datasets, the strategy used for generating the ensembles, the setup of the proposed algorithms and the measures to assess the quality of the consensus partitions. Then, we present the main experimental results obtained on the various datasets.

12.5.1 Evaluation Methodology

Datasets

We used eight benchmark datasets from the UCI Machine Learning Repository [ANml], namely Iris, Wine, Glass, Ecoli, Yeast, Segmentation, Abalone and Letter, and two time series datasets from the UCR Time Series Classification/Clustering Page [KXWRta], namely Tracedata and ControlChart. See Appendix A for more details.

Cluster Validity

For each dataset $D = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_n\}$, where $\boldsymbol{\omega}_i = (\omega_{i1}, \ldots, \omega_{im}), \forall i \in [1..n]$, accuracy of the results by the proposed algorithms, i.e., accuracy of the consensus partition $\check{\mathcal{C}} = \langle \check{\mathcal{L}}, \check{\Gamma}, \check{\Delta} \rangle$, $|\check{\mathcal{L}}| = \check{K}$, was evaluated in terms of:

1. *similarity with respect to the (hard) reference classification* $\widetilde{\mathcal{C}}$, which is defined as:

$$\widetilde{\mathcal{C}} = \langle \widetilde{\mathcal{L}}, \widetilde{\Gamma}, \widetilde{\Delta} \rangle$$

where $\widetilde{\mathcal{L}} = \{\widetilde{\ell}_1, \ldots, \widetilde{\ell}_{\widetilde{K}}\}$ and $\widetilde{\Gamma}$ are directly available from D, whereas $\widetilde{\Delta}$ is computed according to the following formula [DGM+07] ($\forall k \in [1..\widetilde{K}], j \in [1..m]$):

$$\widetilde{\Delta}_{kj} = \frac{\mathrm{e}^{-\Lambda_{kj}/\tilde{h}}}{\sum_{j'=1}^{m} \mathrm{e}^{-\Lambda_{kj'}/\tilde{h}}}$$

where

$$\Lambda_{kj} = \left(\sum_{i=1}^{n} \widetilde{\Gamma}_{ki} \right)^{-1} \sum_{i=1}^{n} \widetilde{\Gamma}_{ki} \left(\overline{v}_{kj} - \omega_{ij} \right)^2$$

$$\overline{v}_{kj} = \left(\sum_{i=1}^{n} \widetilde{\Gamma}_{ki} \right)^{-1} \sum_{i=1}^{n} \widetilde{\Gamma}_{ki} \; \omega_{ij}$$

also, parameter $\tilde{h}$ of LAC,[2] in our experiments, was set equal to 0.2. The evaluation between $\check{\mathcal{C}}$ and $\widetilde{\mathcal{C}}$ was performed according to both object- and feature-based representations, by using $1 - \overline{\psi}_o$ (cf. (12.4)) and $1 - \overline{\psi}_f$ (cf. (12.5)), respectively;

2. *similarity with respect to the solutions in the input ensemble* $\mathcal{E}$, according to $1 - \Psi_o(\check{\mathcal{C}}, \mathcal{E}, D)$ (cf. (12.2)), i.e., comparison in terms of object-based representation, and $1 - \Psi_f(\check{\mathcal{C}}, \mathcal{E}, D)$ (cf. (12.3)), i.e., comparison in terms of feature-based representation;
3. *error-rate* (ER) [DGM+07], which is an internal criterion and measures the intra-cluster compactness:

$$ER(\check{\mathcal{C}}) = \sum_{k=1}^{\check{K}} \sum_{j=1}^{m} \check{\Delta}_{kj} \left(\sum_{i=1}^{n} \check{\Gamma}_{ki} \right)^{-1} \sum_{i=1}^{n} \check{\Gamma}_{ki} \left(\overline{v}_{kj} - \omega_{ij} \right)^2$$

Ensemble Generation

For each set of experiments and dataset we generated 20 different ensembles; all the reported results were averaged over the results obtained on each such ensembles. Ensembles for each dataset were generated by running the LAC algorithm [DGM+07], where the diversity of the solutions was guaranteed by randomly choosing the initial centroids and varying the parameter $\tilde{h}$ in LAC. LAC yields projective clusterings that are hard at data clustering level and have feature-to-cluster assignments unequally weighted; consequently, in order to test the ability of the proposed algorithms to deal also with soft clustering solutions and with solutions having feature-to-cluster assignments equally weighted, we generated each ensemble $\mathcal{E}$ as a composition of four equal-sized subsets, namely $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$, and $\mathcal{E}_4$ such that:

- $\mathcal{E}_1$ contains solutions that are hard at data clustering level and have feature-to-cluster assignments unequally weighted, i.e., solutions obtained by standard LAC;
- $\mathcal{E}_2$ contains solutions that are hard at data clustering level and have feature-to-cluster assignments equally weighted. Starting from a LAC solution $\mathcal{C} = \langle \mathcal{L}, \Gamma, \Delta \rangle$ defined over a set of n m-dimensional objects, where $\mathcal{L} = \{\ell_1, \ldots, \ell_K\}$, we derive the corresponding projective clustering $\mathcal{C}'$, having feature-to-cluster assignments equally weighted, as follows:

$$\mathcal{C}' = \langle \mathcal{L}, \Gamma, \Delta' \rangle$$

where

$$\Delta'_{kj} = \lfloor \Delta_{kj} + 1/m \rfloor, \quad \forall k \in [1..K], j \in [1..m]$$

[2] This parameter controls the incentive for clustering on more features depending on the strength of the correlation of data along the features.

- $\mathcal{E}_3$ contains solutions that are soft at data clustering level and have feature-to-cluster assignments unequally weighted. Starting from a LAC solution $\mathcal{C} = \langle \mathcal{L}, \Gamma, \Delta \rangle$ defined over a set of n m-dimensional objects, where $\mathcal{L} = \{\ell_1, \ldots, \ell_K\}$, we derive the corresponding soft projective clustering $\mathcal{C}''$ as follows:
$$\mathcal{C}'' = \langle \mathcal{L}, \Gamma'', \Delta \rangle$$
where
$$\Gamma''_{ki} = \Pr(k|i), \quad \forall k \in [1..K], i \in [1..n]$$
and $\Pr(k|i)$ is the probability of the cluster labeled with ℓ_k given the observation of the object $\boldsymbol{\omega}_i$, which is computed as described in [DA09].
- $\mathcal{E}_4$ contains solutions that are soft at data clustering level and have feature-to-cluster assignments equally weighted, which are derived from the standard LAC solutions according to the methods employed for generating $\mathcal{E}_2$ and $\mathcal{E}_3$, respectively.

Setting of the Proposed Algorithms

We performed a tuning phase for properly setting the parameters of the proposed methods. We experimentally observed that our methods were scarcely influenced by any specific setting, which allowed us to easily detect setup values well-suited to each of the evaluation datasets. Precisely, in case of the MOEA-PCE algorithm, the population size (t) was set equal to 15% of the ensemble size and the number I of maximum iterations equal to 200; also, the random Gaussian noise needed for the mutation step was obtained by performing a *Monte Carlo* sampling on a Gaussian probability density function with a null mean value and variance equal to one. In case of the EM-PCE algorithm, parameter α of the objective function Q (cf. (12.10)) was set equal to 2.

12.5.2 Results

For each algorithm, dataset and ensemble, we performed 50 different runs and reported average results, and maximum (best) results with relative standard deviation. Note that for the MOEA-PCE algorithm, which may output one or more consensus partitions, we averaged the results over the whole output population.

Evaluation w.r.t. Reference Classification

Table 12.1 and Table 12.2 show the performance on the various datasets in terms of similarity with respect to the reference classifications, by considering the object-based representation and the feature-based representation, respectively.

Table 12.1. MOEA-PCE and EM-PCE: similarity results w.r.t. reference classification (object-based representation)

	ensemble	*MOEA-PCE*			*EM-PCE*		
dataset	*avg-max*	*avg*	*max-std*	*gain w.r.t. ens. (avg)*	*avg*	*max-std*	*gain w.r.t. ens. (avg)*
Iris	.632 .925	.919	.925 .015	+.287	.762	.767 .040	+.130
Wine	.738 .910	.913	.928 .105	+.175	.782	.840 .028	+.044
Glass	.565 .775	.683	.768 .046	+.118	.639	.644 .002	+.074
Ecoli	.421 .689	.603	.686 .054	+.182	.329	.419 .040	-.092
Yeast	.675 .750	.723	.745 .015	+.048	.638	.641 .001	-.037
Segmentation	.590 .821	.755	.835 .049	+.165	.653	.663 .004	+.063
Abalone	.509 .520	.518	.558 .043	+.009	.512	.542 .002	+.003
Letter	.522 .640	.597	.612 .031	+.075	.554	.562 .006	+.032
Tracedata	.772 .868	.862	.998 .059	+.090	.875	.935 .030	+.103
ControlChart	.681 .981	.895	.965 .049	+.214	.790	.806 .007	+.109

Table 12.2. MOEA-PCE and EM-PCE: similarity results w.r.t. reference classification (feature-based representation)

	ensemble	*MOEA-PCE*			*EM-PCE*		
dataset	*avg-max*	*avg*	*max-std*	*gain w.r.t. ens. (avg)*	*avg*	*max-std*	*gain w.r.t. ens. (avg)*
Iris	.662 .998	.988	1 .029	+.326	.845	.895 .043	+.183
Wine	.822 .989	.955	.997 .027	+.133	.869	.899 .080	+.047
Glass	.731 .891	.851	.900 .027	+.120	.817	.877 .041	+.086
Ecoli	.763 .879	.858	.884 .016	+.095	.903	.953 .052	+.140
Yeast	.720 .805	.790	.804 .009	+.070	.684	.690 .003	-.036
Segmentation	.618 .720	.729	.737 .049	+.111	.625	.632 .008	+.007
Abalone	.716 .754	.759	.849 .023	+.043	.726	.748 .013	+.010
Letter	.646 .693	.767	.818 .012	+.121	.780	.786 .007	+.134
Tracedata	.661 .818	.755	.811 .0.25	+.094	.753	.773 .021	+.092
ControlChart	.663 .894	.880	.910 .016	+.217	.734	.774 .022	+.071

In both cases, it can be noted how the performances of the proposed algorithms lead to an average similarity of the output consensus partition(s) that are generally comparable or far better than the average intra-ensemble similarity. According to the object-based representation (Table 12.1), the average improvements (gains) by MOEA-PCE and EM-PCE over all datasets are 13.6% and 4.3%, respectively, with peaks above 16% on five out of ten datasets by MOEA-PCE (up to 29% on Iris), and peaks above 10% on three datasets by EM-PCE (up to 13% on Iris). Analogously, according to the feature-based representation (Table 12.2), the average improvements by MOEA-PCE and EM-PCE over all datasets are 13.3% and 7.3%, respectively.

MOEA-PCE hence outperforms EM-PCE in nearly all datasets, especially according to the object-based representation; however, it seems that the best results achieved by EM-PCE deviate from the average less than the corresponding results by MOEA-PCE.

Evaluation in Terms of Similarity w.r.t. Ensemble

Using the ensemble for each dataset as a reference for comparison, Table 12.3 summarizes the (average) similarity results obtained by MOEA-PCE and EM-PCE.

It is interesting to observe that both the algorithms achieve an average similarity above 94% according to the object-based representation. Also, the algorithms perform well according to the feature-based representation, although MOEA-PCE tends to prevail EM-PCE (86.8% against 80.3%).

Table 12.3. MOEA-PCE and EM-PCE: average similarity results w.r.t. ensemble

	MOEA-PCE		*EM-PCE*	
dataset	*obj-based repres.*	*feat.-based repres.*	*obj-based repres.*	*feat.-based repres.*
Iris	.968	.843	.925	.722
Wine	.989	.876	.986	.847
Glass	.969	.933	.960	.725
Ecoli	.989	.928	.996	.914
Yeast	.962	.965	.968	.947
Segmentation	.939	.967	.932	.842
Abalone	.979	.977	.972	.971
Letter	.930	.721	.910	.677
Tracedata	.966	.830	.913	.750
ControlChart	.853	.638	.815	.634

Table 12.4. MOEA-PCE and EM-PCE: error rate results

	ref. class.	*ensemble*		*MOEA-PCE*					*EM-PCE*				
dataset		*avg*	*min*	*avg*	*min*	*std*	*gain w.r.t. ref. class.*	*gain w.r.t. avg ens.*	*avg*	*min*	*std*	*gain w.r.t. ref. class.*	*gain w.r.t. avg ens.*
Iris	.157	.166	.122	.131	.124	.004	+.026	+.035	.138	.138	0	+.019	+.028
Wine	.089	.111	.039	.041	.039	.001	+.048	+.070	.074	.068	.052	+.015	+.037
Glass	.029	.082	.009	.013	.009	.003	+.016	+.069	.050	.048	.010	-.021	+.032
Ecoli	.038	.056	.020	.033	.021	.007	+.005	+.023	.048	.040	.005	-.010	+.008
Yeast	.030	.049	.035	.039	.036	.002	-.009	+.010	.047	.041	.005	-.017	+.002
Segm.	.007	1.404	.007	.011	.007	.003	-.004	+1.393	.027	.013	.054	-.020	+1.377
Abalone	.032	.028	.022	.023	.017	.007	+.009	+.005	.013	.010	.005	+.019	+.015
Letter	1.632	.185	.098	.130	.109	.007	+1.502	+.055	.770	.764	.006	+.862	-.585
Tracedata	.068	.078	.045	.053	.045	.005	+.015	+.025	.082	.078	.020	-.014	-.004
ControlChart	5.152	2.697	.650	.807	.657	.080	+4.345	+1.890	0.932	0.854	.050	+4.220	+1.765

Evaluation in Terms of Error Rate

Table 12.4 compares the performance of MOEA-PCE and EM-PCE to both the reference classification and the ensemble, for each dataset, in terms of error rate. The gain values refer to differences between the error rate relative

to reference classification (resp., ensemble) and the error rate of the specific algorithm.

Similarly to the previously discussed evaluations based on an external criterion, this evaluation shows that MOEA-PCE outperforms the standard ensemble, obtaining an average improvement (gain) over all the datasets of +0.6 w.r.t. the reference classification and +0.358 w.r.t. the ensemble. EM-PCE also improves upon the error rate of the reference classification (+0.51) and of the ensemble (+0.27).

Conclusion

The main goal of KDD (*Knowledge Discovery in Databases*) process is to identify novel, valid, potentially useful, and ultimately understandable patterns in data. It comprises several steps, and *data mining* is the most representative one.

This thesis has focused on data mining and, in particular, on the task of *clustering*, whose main goal is, given a set of objects and a distance function between such objects, to partition such a set into a number of *compact* and *well-separated* clusters, where compactness and separation are measured according to the input distance measure. In particular, this thesis has addressed two main problems arising from clustering: the problem of *uncertainty* and the problem of *the curse of dimensionality*.

Uncertainty

Uncertainty in data clustering has been investigated in this thesis looking at two different aspects, i.e., uncertainty that can be inherently present in the representation of particular kind of data, and the so-called *clustering uncertainty*, which refers to uncertainty that typically affects the clustering output.

The specific model of uncertainty in data representation taken into account in this thesis is that exploited for representing the so-called *uncertain objects*, i.e, objects described by a multidimensional region of definition, which is a finite subset of the full dimensional space, and one or more probability distributions aimed at providing the probability that the exact location of the object coincides with a specific point of the region of definition. In this respect, two new algorithms for clustering uncertain objects has been proposed. The first one, called *UK-Medoids*, is a K-Medoids-based algorithm, mainly designed for overcoming some issues that typically affect the existing state-of-the-art partitional algorithms for clustering uncertain objects. The second algorithm proposed, i.e., *U-AHC*, is the first agglomerative hierarchical clustering algorithm for this kind of data. A major novelty of U-AHC is

the definition of a novel prototype link-based criterion for choosing the pair of clusters to be merged at each iteration of the standard AHC greedy scheme. Such a criterion exploits some notions from *Information Theory*, and has been proved to be particularly sound for the context of uncertain objects. A further contribution in this regard has been concerned the use of U-AHC algorithm in a real application context, i.e, it has been adapted to handle *microarray* biomedical data with *probe-level uncertainty*.

The other aspect related to uncertainty in data clustering addressed in this thesis has concerned the clustering level; indeed, due to its intrinsic ill-posed nature, clustering naturally outputs results that are unavoidably affected by uncertainty. In order to properly solve this problem, several *clustering ensembles* methods has been defined in the literature. Such methods aim to overcome clustering uncertainty by generating a set of clustering solutions, i.e., an *ensemble*, and determining the final clustering (i.e., the *consensus partition*) by properly exploiting the information available from the ensemble.

In this respect, this thesis has focused on the newly emerged problem of *weighted consensus clustering*, whose main goal is to discriminate among the solutions in the ensemble in order to recognize those that should in principle be taken in higher consideration than the remaining ones by any clustering ensembles algorithm. In particular, three schemes for weighting the solutions in the ensemble has been proposed based on the notion of *ensemble diversity*, i.e., *Single Weighting* (SW), *Group Weighting* (GW), and *Dendrogram Weighting* (DW). A major novelty of the proposed weighting schemes is their generality, because they are defined only exploiting information available from the ensemble while not requiring any particular clustering ensembles algorithm or consensus function to work. Accordingly, the proposed schemes are easily applicable to any existing clustering ensembles algorithm falling into one of the most popular categories of clustering ensembles algorithms, i.e., *instance-based*, *cluster-based*, and *hybrid*. In this regard, a further contribution has been regarded the development of three algorithms, i.e., *Weighted Instance-based Clustering Ensembles* (WICE), *Weighted Cluster-based Clustering Ensembles* (WCCE), and *Weighted Hybrid Clustering Ensembles* (WHCE), which allow for easily exploiting the information given by any proposed weighting scheme directly within any instance-based, cluster-based, or hybrid clustering ensembles algorithm.

The Curse of Dimensionality

The curse of dimensionality in data clustering is related to all the (accuracy and efficiency) issues which typically arise from applying classic clustering algorithms to data objects having large dimensionality. To overcome this problem, classic approaches resort to *global* and *local* dimensionality reduction techniques.

This thesis has focused on global dimensionality reduction by taking into account the scenario of *time series* data. In this respect, a new dimensionality reduction technique (i.e., representation model) for time series data has

been proposed, called *Derivative time series Segment Approximation* (DSA). The proposed model is essentially based on a simple idea that has not been previously involved into any of the existing state-of-the-art time series representation models; such an idea consists in performing a segmentation procedure on the derivative version of the original time series. This allows for capturing the main trends of the series and obtain high compression without significant loss of accuracy. Further contributions in this respect has been concerned the adaptation of DSA to some real application contexts. In particular, DSA has been exploited for managing *mass spectrometry* biomedical data, and low-voltage electricity customer *load profile* data.

Regarding local dimensionality reduction, the problem of *projective clustering* has been taken into account. The main goal of projective clustering is to discover clusters along with the corresponding *subspaces*, which have to be identified by respecting the following principle: data objects in the same cluster are highly similar to each other if and only if they are projected onto the subspace associated to that cluster.

In this regard, this thesis has introduced the novel problem of *Projective Clustering Ensembles* (PCE), which aims to give an unified view of the problems of clustering ensembles and projective clustering. In particular, the focus of PCE is on the development of techniques aimed at properly discovering a projective consensus partition from an ensemble composed by projective clustering solutions. More precisely, two specific formulation of PCE as an optimization problem have been provided, i.e., *two-objective* PCE and *single-objective* PCE. These formulations take into account the information available form the ensemble at a data clustering level and a feature level by defining two objective functions conflicting with each other (two-objective PCE) or combining these two different kinds of information into a single objective function (single-objective PCE). Furthermore, for each one of the proposed formulations, two heuristic algorithm have been defined, called *MOEA-PCE* and *EM-PCE*, respectively. MOEA-PCE resorts to the context of *Multi Objective Evolutionary Algorithms*, whereas EM-PCE exploits a scheme similar to that employed by the popular *Expectation Maximization* clustering algorithm.

Future Research

The research described in this thesis can be further extended in several directions.

Regarding uncertainty in data representation, two interesting problems which received so far few consideration in the literature can be addressed: the problems of *semi-supervised* clustering of uncertain objects and projective clustering of uncertain objects. The first one refers to clustering uncertain objects in a such a way that the entire process is guided by a set of constraints aimed at suggesting which objects should belong to the same cluster and which ones should not. The challenge here is due to the form of the constraints, which can be in turn probabilistic (i.e., uncertain) like the objects to

be clustered. Projective clustering of uncertain objects should aim to discover projective clusters whose subspaces are sound given the intrinsic probabilistic representation of the uncertain objects. As a result, such subspaces can be in turn considered as random variables, which can be described by pdfs derived by properly taking into account pdfs of the uncertain objects.

Research on clustering uncertainty and, in particular, on clustering ensembles may be extended as follows. The information-theoretic properties of the weight distributions outputted from the clustering weighting schemes SW, GW, and DW can be deeply analyzed in order to *(i)* auto-determine the user-defined parameter α involved in all the proposed schemes, and *(ii)* select a small number of clustering solutions to reduce the size of the ensemble without penalizing the accuracy of the consensus partition. Moreover, in addition to diversity and accuracy that are commonly involved for solving *clustering ensemble selection* problem, further structural ensemble properties may be exploited to develop innovative approaches to solve such a problem. A further objective is analyzing the theoretical properties of the hybrid approach, in order to give proper formulation(s) of this approach as an optimization problem (thus, not only limited to the graph-based formulation) and define proper heuristic solutions for the formulation(s) at hand.

The contribution of this thesis to local dimensionality reduction, i.e., the formulation of the novel projective clustering ensembles (PCE) problem, is actually at an early stage; thus, the improvements in this regard are manifold. A first goal to be accomplished is developing proper instance-based, cluster-based, and hybrid formulations of PCE (like for standard clustering ensembles), in order to define heuristics with lower complexity, both computational and implementation, than the formulations proposed in this thesis (i.e., two-objective PCE and single-objective PCE). A further goal is to provide a more detailed insight into the theoretical properties of two- and single-objective PCE, in order to improve accuracy by deriving a new formulation from the combination of the original ones. Moreover, since PCE is clearly related to the notion of uncertainty both in clustering output and in the subspaces associated to the various clusters, future research in this regard might concern the study of the probabilistic properties of PCE, particularly focusing on model-based clustering methods and techniques from the uncertain objects management context.

A

Appendix: Datasets Used in the Experiments

A.1 UCI Datasets

Table A.1 reports on the main characteristics of the benchmark datasets from UCI Machine Learning Repository [ANml] involved into the experiments of this thesis.

Table A.1. UCI benchmark datasets used in the experiments

dataset	*# of objects*	*# of attributes*	*# of classes*
Iris	150	4	3
Wine	178	13	3
Glass	214	10	6
Ecoli	327	7	5
Yeast	1,484	8	10
ISOLET	1,800	617	6
ImageSegmentation	2,310	19	7
Abalone	4,124	7	17
LetterRecognition	7,648	16	10

Iris contains measurements relating different iris plants. Wine refers to results of a chemical analysis on Italian wines derived from three different cultivars. In Glass, each glass instance is described by the values of its chemical components. Ecoli contains data on the Escherichia Coli bacterium, which are identified with values coming from different analysis techniques. Yeast objects describe the main features and the localization of various proteins. ISOLET contains objects representing letters of the alphabet spoken by certain subjects; we selected a subset of objects representing the letters A, B, C, D, E, and G. ImageSegmentation contains objects that were randomly drawn from a database of seven outdoor images; the images (3x3 regions) were hand-segmented to create a classification for each pixel. Abalone is about differ-

ent types of abalone shells. LetterRecognition contains character images corresponding to the capital letters in the English alphabet; we selected the instances of each letter from A to J.

A.2 Time Series Datasets

Six time series datasets (cf. Table A.2) were selected for carrying out the experiments of this thesis. All of these are publicly available from UCR Time Series Classification/Clustering Page [KXWRta].

Table A.2. Time Series datasets used in the experiments

dataset	*# of objects*	*# of attributes*	*# of classes*
GunX	200	150	2
Tracedata	200	275	4
ControlChart	600	60	6
CBF	300	128	3
Twopat	800	128	4
Mixed-BagShapes	160	1,614	9

GunX comes from the video surveillance domain. Tracedata simulates signals representing instrumentation failures. In CBF (Cylinder-Bell-Funnel), each class is characterized by a specific pattern, namely a plateau (C), an increasing ramp followed by a sharp decrease (B), a sharp increase followed by a decreasing ramp (F). ControlChart contains synthetically generated control charts which are classified into one of the following: normal, cyclic, increasing trend, decreasing trend, upward shift, and downward shift. In Twopat, two different patterns (upward step and downward step) are used to define the classes down-down, up-down, down-up, and up-up. Mixed-BagShapes contains time series derived from shapes belonging to nine classes (bone, cup, device, fork, glass, hand, pencil, rabbit and tool).

A.3 Microarray Datasets

Our experiments were performed on four large microarray datasets, each of which describing the expressions of thousands of genes in biological tissues, as shown in Table A.3.

Three datasets, namely Leukaemia, Neuroblastoma and Myelodysplastic are cancer tissue data of humans [oMHgi], while Mouse is about mouse tissues [EEml]. Leukaemia describes the transformation process of leukaemia stem cells initiated by MLL-AF9 fusion gene. Neuroblastoma contains expression-based screening results for neuroblastoma differentiation. In Myelodysplastic,

Table A.3. Microarray datasets used in the experiments

dataset	*# of objects (genes)*	*# of attributes (conditions)*
Leukaemia	22,690	21
Neuroblastoma	22,282	14
Myelodysplastic	22,277	25
Mouse	45,101	10

somatic chromosomal deletions in cancer are measured by means of an RNA-mediated interference (RNAi)-based approach to discovery of the $5q^-$ disease gene, which is a subtype of the myelodysplastic syndrome characterized by a defect in erythroid differentiation. Mouse contains a transcription profiling of mouse cochlea Reissner's membrane (RM). This is grown as explants and treated with dexamethasone and then subject to RNA extraction to investigate gene expressions.

A.4 Mass Spectrometry Datasets

We used mass spectrometry datasets publicly available from [fCRsp]. All these datasets contain SELDI-TOF spectra and were obtained using different clinical studies under different mass spectrometry platforms and experimental conditions. Table A.4 summarizes their main characteristics.

Table A.4. Mass spectrometry datasets used in the experiments

dataset	*# of objects*	*# of attributes*	*# of classes*
OvarianCancer	49	28,000	2
Cardiotoxicity	115	7,105	4
Pancreatic	181	6,771	2
Prostate	322	15,154	4

In OvarianCancer [PAH$^+$02] the spectra are derived from an analysis of serum samples of a female population belonging to two classes (ovarian cancer diseased and healthy). Also, OvarianCancer spectra were subject to a preliminary preprocessing phase specific for MS data [MCK$^+$05, WNP03]. Cardiotoxicity contains high resolution, binned spectra used in a toxiproteomic analysis of anthracycline-induced cardiotoxicity [PRH$^+$04]. Data are labeled according to four classes: definite negative (24), probable negative (43), probable positive (10), definite positive (34). Pancreatic comprises high resolution, binned spectra used in a study on premalignant pancreatic cancer detection [HPM$^+$03]. There are here two classes: control (101), pancreatic intraepithelial neoplasias (80). Prostate includes low resolution spectra used in a study on prostate cancer, which have already been provided with the baseline subtracted [POP$^+$02].

Data is assigned with four classes: cancer and PSA level $> 10\ ng/ml$ (43), cancer and PSA level within [4..10] ng/ml (26), benign and PSA level $> 4\ ng/ml$ (190), no evidence of disease and PSA level $< 1\ ng/ml$ (63).

B

Appendix: DDTW and DSA Derivative Estimation Models

Approximating the derivative of a given series plays an essential role in the DDTW method as well as in our DSA. We have described both the DDTW and DSA derivative estimation models in Chapt. 9, Sect. 9.2.1 ((9.1)-(9.2)). Here we present some experimental results which show a comparison of these two models concerning their *(i)* performance in approximating real derivatives of standard functions and *(ii)* impact on the performance of DSA and DDTW.

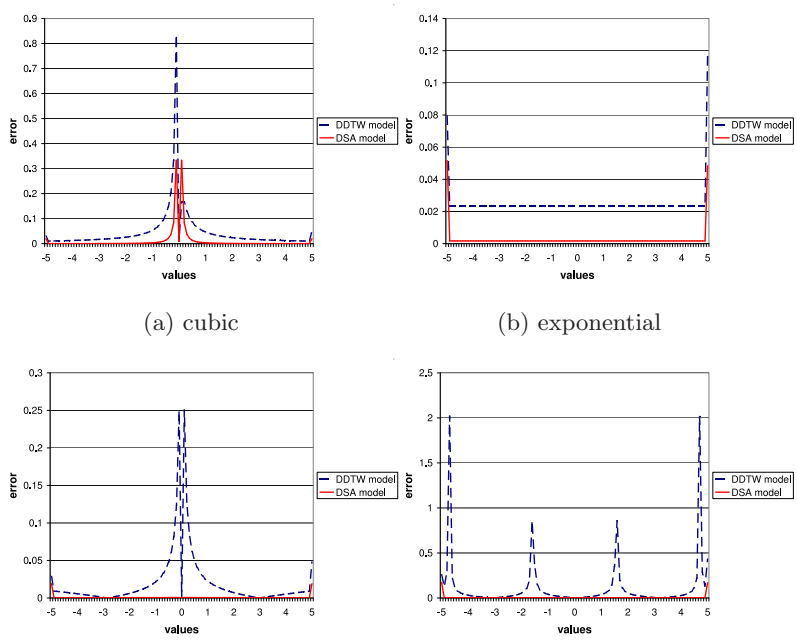
B.1 Evaluation of the Approximation of Real Derivative Functions

Let $f(x) : \Re \rightarrow \Re$ be a continuous function and $f'(x) : \Re \rightarrow \Re$ be its first derivative. Let $R = [x_1, \ldots, x_m] \in \Re^m$ denote a sequence of real values, over which we suppose to define a sequence X and the corresponding sequence X' of actual derivative values. Formally, let $X = [y_1, \ldots, y_m]$ and $X' = [y'_1, \ldots, y'_m]$, such that $y_j = f(x_j)$ and $y'_j = f'(x_j), \forall j \in [1..m]$. Given X, we also denote with $\dot{X} = [\dot{y}_1, \ldots, \dot{y}_m]$ the approximation derivative version of X which is obtained by a certain estimation model.

We compute the average approximation error of $\dot{X}$ (i.e., the estimated derivative sequence) with respect to X' (i.e., the actual derivative sequence) as follows:

$$e(\dot{X}, X') = \frac{1}{m} \sum_{j=1}^{m} \frac{|\dot{y}_j - y'_j|}{|y'_j|}$$

Figure B.1 shows a comparison between the DDTW and DSA derivative estimation models on four example functions, namely a cubic polynomial, an exponential function, a sine wave, and the Gaussian function. Table B.1 reports on the average approximation errors (in percentage) obtained by using the two models on the selected functions. It can be noted that the DSA derivative estimation model produces an average error of approximation which is always lower than the error by the DDTW model.



(a) cubic (b) exponential

(c) Gaussian (d) sin wave

Fig. B.1. Approximation errors on derivative estimation: DDTW model vs. DSA model

Table B.1. Average approximation errors on derivative estimation: DDTW model vs. DSA model. Each function is valued on 101 points over the range $[-5, +5]$.

function	*DDTW model*	*DSA model*
cubic	4.52%	1.12%
exponential	2.48%	0.26%
Gaussian	1.99%	0.08%
sine	11.48%	0.5%

B.2 Impact on the Performance of DSA and DDTW

In the main experimental sections we have presented clustering results obtained on the various data sets by using DSA and DDTW. Since both methods are characterized by a distinct model of derivative estimation, it was also interesting to gain an insight into the impact of this model on the performance of DSA and DDTW. For this purpose, we exchanged the respective models of derivative estimation in DSA and DDTW and then repeated the relative experimental evaluation in clustering frameworks.

Table B.2 shows the clustering results obtained by K-Means and UPGMA when DDTW was equipped with the DSA derivative estimation model, and compares these results with those previously reported in Table 9.2 and Table 9.3. The modified version of DDTW led to better performances than the original DDTW method in most cases, in particular Mixed-BagShapes (4% by UPGMA and 2% by K-Means), Twopat (3% by UPGMA and 1% by K-Means), and ControlChart (2%).

Analogously, Table B.3 reports on the clustering results when DSA was equipped with the DDTW derivative estimation model, and compares them with the original performances of DSA. Again, the DSA derivative estimation model prevailed against the DDTW one in most cases—OvarianCancer (5% by UPGMA and 4% by K-Means), CBF and GunX (4% by K-Means and 2% by K-Means), Mixed-BagShapes (3% by K-Means and 2% by UPGMA), and Twopat (2%).

Table B.2. DDTW-based clustering results by varying the derivative estimation model

clustering algorithm	derivative estimation	GunX	Tracedata	ControlChart	CBF	Twopat	Mixed-Bag Shapes	Ovarian Cancer
K-Means	DSA	*.90*	*1*	*.91*	*.95*	*.96*	*.78*	*.63*
	DDTW	.89	1	.89	.96	.95	.76	.62
UPGMA	DSA	*.72*	*.78*	*.56*	*.48*	*.67*	*.45*	*.64*
	DDTW	.72	.76	.54	.49	.64	.41	.63

Table B.3. DSA-based clustering results by varying the derivative estimation model

clustering algorithm	derivative estimation	GunX	Tracedata	ControlChart	CBF	Twopat	Mixed-Bag Shapes	Ovarian Cancer
K-Means	DSA	*.92*	*1*	*.90*	*.96*	*.97*	*.78*	*.75*
	DDTW	.90	1	.91	.92	.95	.75	.71
UPGMA	DSA	*.73*	*.82*	*.54*	*.60*	*.67*	*.51*	*.73*
	DDTW	.69	.80	.56	.62	.65	.49	.68

The above results led us to conclude that the DSA derivative estimation model can enhance the DDTW method in practice; conversely, the DDTW derivative estimation model does not bring any beneficial effect to (in general, it may negatively affect) the performance of the DSA method.

C

Appendix: Impact of Time Series Preprocessing on Similarity Detection

As we have discussed in Chapt. 9, Sect. 9.3.4, smoothing is performed prior to the mining tasks in order to handle noise in the raw data, regardless of the particular representation method or distance measure used. In our experimental evaluation, smoothing turned out to be useful for all the methods on every dataset—except for the OvarianCancer case. The intuition that skipping the smoothing stage would cause a decrease in performing similarity detection was supported by experimental evidence when we tried to directly classify the original (i.e., non-smoothed) data. Indeed, as shown for some prominent methods in Table C.1, the decrease would be significantly high in most datasets, with peaks of around 50% on ControlChart, CBF, and Twopat.

Another important remark is that the relative performances of most of the various methods (including our DSA) do not vary substantially whether or not smoothing is performed. This indicates that the representation model and similarity/distance measure play a more important role than the preprocessing operations in determining the best approach(es) to similarity detection in time series.

A special remark should also be made on the OvarianCancer dataset which is huge-dimensional and largely affected by noisy factors, like most of mass spectra datasets. On this dataset, DSA performed far better than DDTW, precisely +13% by K-Means, +10% by UPGMA, in terms of F1-Measure (cf. Tables 9.2–9.3).

Table C.2 summarizes the best preprocessing setups for DSA and the other methods on the various datasets, using the K-Means algorithm; we left the best setups for UPGMA out of the presentation, since they resulted fairly similar to those obtained by K-Means in most datasets. In the table, term MA (resp., EXP) stands for moving average (resp., exponential smoothing) and is followed by the value set for λ (resp., φ) and the number of iterations.

Table C.1. Time series clustering: K-Means performance reduction in case of no smoothing

	GunX	Tracedata	ControlChart	CBF	Twopat	Mixed-BagShapes
FTW	-21%	-16%	-41%	-29%	-23%	–
DTW	-16%	-2%	-46%	-48%	-53%	-4%
DDTW	-19%	–	-47%	-57%	-50%	–
DWT	-6%	-17%	-37%	-35%	-15%	–
SD	-5%	-19%	-45%	-46%	-47%	–
PLA	-14%	-1%	-47%	-46%	-35%	-3%
PAA	-7%	-2%	-46%	-46%	-30%	-5%
SAX	-22%	-2%	-43%	-47%	-25%	-4%
APCA	-18%	-5%	-48%	-42%	-46%	-4%
DSA	-19%	–	-48%	-54%	-51%	–

Table C.2. Summary of the preprocessing setups providing the best time series clustering results by K-Means

	GunX	Trace data	Control Chart	CBF	Twopat	Mixed-Bag Shapes	Ovarian Cancer
LCSS	MA λ=9 it=3	MA λ=5 it=3	EXP φ=0.3 it=1	No smooth.	EXP φ=0.1 it=4	No smooth.	No smooth.
EDR	No smooth.	EXP φ=0.3 it=5	EXP φ=0.7 it=1	EXP φ=0.7 it=1	No smooth.	EXP φ=0.1 it=3	No smooth.
ERP	EXP φ=0.1 it=5	EXP φ=0.1 it=1	EXP φ=0.7 it=3	EXP φ=0.1 it=5	MA λ=5 it=3	EXP φ=0.5 it=5	No smooth.
FTW	EXP φ=0.3 it=3	No smooth.	EXP φ=0.7 it=3	EXP φ=0.1 it=3	EXP φ=0.3 it=3	EXP φ=0.5 it=2	No smooth.
DTW	EXP φ=0.1 it=3	No smooth.	EXP φ=0.9 it=1	No smooth.	EXP φ=0.9 it=5	EXP φ=0.1 it=5	No smooth.
DDTW	EXP φ=0.9 it=3	EXP φ=0.1 it=1	EXP φ=0.3 it=5	EXP φ=0.5 it=5	EXP φ=0.1 it=1	EXP φ=0.1 it=1	No smooth.
DFT	EXP φ=0.2 it=4	EXP φ=0.1 it=3	MA λ=9 it=2	EXP φ=0.1 it=1	EXP φ=0.2 it=4	MA λ=5 it=1	No smooth.
DWT	EXP φ=0.1 it=2	EXP φ=0.6 it=2	EXP φ=0.3 it=1	EXP φ=0.1 it=3	EXP φ=0.6 it=3	EXP φ=0.1 it=2	No smooth.
CHEBY	EXP φ=0.1 it=3	EXP φ=0.9 it=5	EXP φ=0.7 it=5	EXP φ=0.5 it=1	MA λ=9 it=3	EXP φ=0.9 it=3	No smooth.
SD	EXP φ=0.7 it=5	MA λ=9 it=5	EXP φ=0.6 it=4	EXP φ=0.3 it=4	EXP φ=0.3 it=5	EXP φ=0.9 it=4	No smooth.
PLA	EXP φ=0.1 it=1	MA λ=5 it=1	EXP φ=0.5 it=1	EXP φ=0.5 it=1	EXP φ=0.3 it=1	EXP φ=0.4 it=1	No smooth.
PAA	EXP φ=0.1 it=1	EXP φ=0.5 it=1	EXP φ=0.7 it=3	EXP φ=0.7 it=1	EXP φ=0.1 it=1	EXP φ=0.1 it=1	No smooth.
SAX	EXP φ=0.3 it=3	EXP φ=1 it=1	EXP φ=0.4 it=3	EXP φ=0.5 it=1	EXP φ=0.1 it=1	EXP φ=0.5 it=3	No smooth.
APCA	EXP φ=0.1 it=3	EXP φ=0.7 it=5	EXP φ=0.7 it=3	No smooth.	EXP φ=0.5 it=1	EXP φ=0.1 it=3	No smooth.
DSA	EXP φ=0.9 it=3	EXP φ=0.1 it=1	EXP φ=0.2 it=2	EXP φ=0.4 it=3	EXP φ=0.2 it=2	EXP φ=0.1 it=2	No smooth.

Bibliography

[ABD+08] E. Achtert, C. Böhm, J. David, P. Kröger, and A. Zimek. Robust Clustering in Arbitrarily Oriented Subspaces. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, pages 763–774, 2008.

[ABK+06a] E. Achtert, C. Böhm, H. -P. Kriegel, P. Kröger, I. Mü ller-Gorman, and A. Zimek. Finding Hierarchies of Subspace Clusters. In *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 446–453, 2006.

[ABK+06b] E. Achtert, C. Böhm, H. -P. Kriegel, P. Kröger, and A. Zimek. Deriving Quantitative Models for Correlation Clusters. In *Proc. ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, pages 4–13, 2006.

[ABK+07a] E. Achtert, C. Böhm, H. -P. Kriegel, P. Kröger, I. Mü ller-Gorman, and A. Zimek. Detection and Visualization of Subspace Cluster Hierarchies. In *Proc. Int. Conf. on Database Systems for Advanced Applications (DASFAA)*, pages 152–163, 2007.

[ABK+07b] E. Achtert, C. Böhm, H. -P. Kriegel, P. Kröger, and A. Zimek. On Exploring Complex Relationships of Correlation Clusters. In *Proc. Int. Conf. on Scientific and Statistical Database Management (SSDBM)*, 2007.

[ABK+07c] E. Achtert, C. Böhm, H. -P. Kriegel, P. Kröger, and A. Zimek. Robust, Complete, and Efficient Correlation Clustering. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, 2007.

[ABKS99] M. Ankerst, M. M. Breunig, H. -P. Kriegel, and J. Sander. OPTICS: Ordering Points To Identify the Clustering Structure. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 49–60, 1999.

[AFS93] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient Similarity Search in Sequence Databases. In *Proc. Int. Conf. on Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, 1993.

[Agg07] C. C. Aggarwal. On Density Based Transforms for Uncertain Data Mining. In *Proc. IEEE Int. Conf. on Data Engineering (ICDE)*, pages 866–875, 2007.

[Agg09] C. C. Aggarwal. *Managing and Mining Uncertain Data*. Springer, 2009.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Appli-

cations. In *Proc. ACM SIGMOD Int. Conf. on Management Of Data*, pages 94–105, 1998.

[AJ56] B. P. Adhikari and D. D. Joshi. Distance Discrimination et Resume Exhaustif. *Publs. Inst. Statistics*, 5:57–74, 1956.

[AK03] H. Ayad and M. S. Kamel. Finding Natural Clusters Using Multi-Clusterer Combiner Based on Shared Nearest Neighbors. In *Proc. Int. Workshop on Multiple Classifier Systems (MCS)*, pages 166–175, 2003.

[AKG87] S. Abiteboul, P. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 34–48, 1987.

[AKMS07] I. Assent, R. Krieger, E. Muller, and T. Seidl. DUSC: Dimensionality Unbiased Subspace Clustering. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, 2007.

[ALB+07] D. Apetrei, I. Lungu, F. Batrinu, G. Chicco, R. Porumb, and P. Postolache. Load Pattern Classification and Profiling for A Large Supply Company. In *Proc. Int. Conf. on Electricity Distribution (CIRED)*, pages 1–4, 2007.

[AM03] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422:198–207, 2003.

[AM04] P. K. Agarwal and N. H. Mustafa. k-Means Projective Clustering. In *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 155–165, 2004.

[And73] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[ANml] A. Asuncion and D. J. Newman. UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/.

[APW+99] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast Algorithms for Projected Clustering. In *Proc. ACM SIGMOD Int. Conf. on Management Of Data*, pages 61–72, 1999.

[AS66] S. M. Ali and S. D. Silvey. A General Class of Coefficients of Divergence of One Distribution from Another. *Journal of the Royal Statistical Society*, 28(1):131–142, 1966.

[AY00] C. C. Aggarwal and P. S. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces. In *Proc. ACM SIGMOD Int. Conf. on Management Of Data*, pages 70–81, 2000.

[AY08] C. C. Aggarwal and P. S. Yu. Outlier Detection with Uncertain Data. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, pages 483–493, 2008.

[AY09] C. C. Aggarwal and P. S. Yu. A Survey of Uncertain Data Algorithms and Applications. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 21(5):609–623, 2009.

[Bal70] E. Balas. *Minimax and Duality for Linear and Nonlinear Mixed-Integer Programming*. North Holland, 1970.

[Bas89] M. Basseville. Distance Measures for Signal Processing and Pattern Recognition. *Signal Processing*, 18(4):349–369, 1989.

[BB99a] A. Baraldi and P. Blonda. A Survey of Fuzzy Clustering Algorithms for Pattern Recognition. I. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 29(6):778–785, 1999.

[BB99b] A. Baraldi and P. Blonda. A Survey of Fuzzy Clustering Algorithms for Pattern Recognition. II. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 29(6):786–801, 1999.

[BC94] D. J. Berndt and J. Clifford. Using Dynamic Time Warping To Find Patterns in Time Series. In *Proc. AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370, 1994.

[BC09] A. Benis and M. Courtine. Biomarker Discovery in Medical Genomics Data. In *Proc. Int. Conf. on Bioinformatics and Computational Biology (BIOCOMP)*, pages 265–274, 2009.

[BCR05] B. Botte, V. Cannatelli, and S. Rogai. The Telegestore project in ENEL's metering system. In *Proc. Int. Conf. on Electricity Distribution (CIRED)*, 2005.

[Bel61] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[Ber02] P. Berkhin. Survey Of Clustering Data Mining Techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[BF98] P. S. Bradley and U. M. Fayyad. Refining Initial Points for K-Means Clustering. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 91–99, 1998.

[BGG97] C. S. Burrus, R. A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, 1997.

[BGM+05] H. Bensmail, J. Golek, M. M. Moody, J. O. Semmes, and A. Haoudi. A novel approach for clustering proteomics data using Bayesian fast Fourier transform. *Bioinformatics*, 21(10):2210–2224, 2005.

[BGRS99] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When Is "Nearest Neighbor" Meaningful? In *Proc. Int. Conf. on Database Theory (ICDT)*, pages 217–235, 1999.

[Bha43] A. Bhattacharyya. On a Measure of Divergence Between Two Statistical Populations Defined by their Probability Distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–110, 1943.

[BK99] E. Bauer and R. Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36(1-2):105–139, 1999.

[BKKK04] C. Böhm, K. Kailing, H. -P. Kriegel, and P. Kröger. Density Connected Clustering with Local Subspace Preferences. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 27–34, 2004.

[BKKZ04] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing Clusters of Correlation Connected Objects. In *Proc. ACM SIGMOD Int. Conf. on Management Of Data*, pages 455–466, 2004.

[BMW+03] K. A. Baggerly, J. S. Morris, J. Wang, D. Gold, L. C. Xiao, and K. R. Coombes. A comprehensive approach to the analysis of matrix-assisted laser desorption/ionization-time of flight proteomics spectra from serum samples. *Proteomics*, 3:1667–1672, 2003.

[BO04] C. Boulis and M. Ostendorf. Combining Multiple Clustering Systems. In *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 63–74, 2004.

[BP92] J. C. Bezdek and S. K. Pal. *Fuzzy Models for Pattern Recognition*. IEEE Press, 1992.

[Bri90] E. H. Bristol. Swinging door trending: adaptive trending recording. In *Proc. ISA National Conf.*, pages 749–753, 1990.

[BSHW06] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with Uncertainty and Lineage. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 953–964, 2006.

[BZ04] J. Bi and T. Zhang. Support Vector Classification with Input Data Uncertainty. In *Proc. Int. Conf. on Neural Information Processing Systems (NIPS)*, pages 483–493, 2004.

[CBM07] K. R. Coombes, K. A. Baggerly, and J. S. Morris. *Pre-Processing Mass Spectrometry Data*, pages 79–99. Kluwer, 2007.

[CCKN06] M. Chau, R. Cheng, B. Kao, and J. Ng. Uncertain Data Mining: An Example in Clustering Location Data. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, pages 199–204, 2006.

[CENS06] R. Caruana, M. F. Elhawary, N. Nguyen, and C. Smith. Meta Clustering. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 107–118, 2006.

[CF99] K. Chan and A. Fu. Efficient Time Series Matching by Wavelets. In *Proc. Int. Conf. on Data Engineering (ICDE)*, pages 126–133, 1999.

[CFZ99] C. -H. Cheng, A. W. Fu, and Y. Zhang. Entropy-Based Subspace Clustering for Mining Numerical Data. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 84–93, 1999.

[Cha04] D. Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 112–124, 2004.

[Che52] H. Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[CJW08] L. Chen, Q. Jiang, and S. Wang. A Probability Model for Projective Clustering on High Dimensional Data. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 755–760, 2008.

[CKH07] C. K. Chui, B. Kao, and E. Hung. Mining Frequent Itemsets from Uncertain Data. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, pages 47–58, 2007.

[CKMP02] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. *ACM Transactions on Database Systems*, 27(2):188–228, 2002.

[CKP03] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating Probabilistic Queries Over Imprecise Data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 551–562, 2003.

[CLL06] V. Cantoni, L. Lombardi, and P. Lombardi. Challenges for Data Mining in Distributed Sensor Networks. In *Proc. Int. Conf. on Pattern Recognition (ICPR)*, pages 1000–1007, 2006.

[CN04a] Y. Cai and R. Ng. Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 599–610, 2004.

[CN04b] L. Chen and R. Ng. On The Marriage of L_p-norms and Edit Distance. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 792–803, 2004.

[CNP+05] G. Chicco, R. Napoli, F. Piglione, P. Postolache, M. Scutariu, and C. Toader. Emergent electricity customer classification. *IEEE Proceedings Generation, Transmission and Distribution*, 152(2):164–172, 2005.

[CNP06] G. Chicco, R. Napoli, and F. Piglione. Comparisons Among Clustering Techniques for Electricity Customer Classification. *IEEE Transactions on Power Systems*, 21(2):933–940, 2006.

[CÖO05] L. Chen, M. T. Özsu, and V. Oria. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 491–502, 2005.

[CT06] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 2 edition, 2006.

[CV07] M. Cannataro and P. Veltri. MS-Analyzer: preprocessing and data mining services for proteomics applications on the Grid. *Concurrency and Computation: Practice and Experience*, 19(15):2047–2066, 2007.

[DA09] C. Domeniconi and M. Al-Razgan. Weighted Cluster Ensembles: Methods and Analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(4), 2009.

[DCSL02] M. Dash, K. Choi, P. Scheuermann, and H. Liu. Feature Selection for Clustering - A Filter Solution. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 115–122, 2002.

[DE84] W. H. Day and H. Edelsbrunner. Efficient Algorithms for Agglomerative Hierarchical Clustering Methods. *Journal of Classification*, 1(1):7–24, 1984.

[Deb01] K. Deb. *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley & Sons, 2001.

[Def77] D. Defays. An Efficient Algorithm for a Complete Link Method. *Computer Journal*, 10(4):364–366, 1977.

[Dem67] A. P. Dempster. Upper and Lower Probabilities Induced by a Multivalued Mapping. *The Annals of Mathematical Statistics*, 38(2):325–339, 1967.

[Dep89] E. F. Deprettere, editor. *SVD and Signal Processing: Algorithms, Applications and Architectures*. North-Holland Publishing, 1989.

[DF03] S. Dudoit and J. Fridlyand. Bagging to Improve the Accuracy of a Clustering Procedure. *Bioinformatics*, 19(9):1090–1099, 2003.

[DGM+05] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-based approximate querying in sensor networks. *VLDB Journal*, 14(4):417–443, 2005.

[DGM+07] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos. Locally Adaptive Metrics for Clustering High Dimensional Data. *Data Mining and Knowledge Discovery*, 14(1):63–97, 2007.

[DGMH04] A. Deshpande, C. Guestrin, S. Madden, and J. Hellersteinand W. Hong. Model-Driven Data Acquisition in Sensor Networks. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 588–599, 2004.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[DK97] R. N. Davé and R. Krishnapuram. Robust Clustering Methods: A Unified View. *IEEE Transactions on Fuzzy Systems*, 5(2):270–293, 1997.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rdin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[DLY97] M. Dash, H. Liu, and J. Yao. Dimensionality Reduction of Unsupervised Data. In *Proc. Int. Conf. on Tools with Artificial Intelligence*, pages 532–539, 1997.

[Dom01] B. E. Dom. An Information-Theoretic External Cluster-Validity Measure. Technical Report RJ 10219, IBM T. J. Watson Research Center, May 2001.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[DPGM04] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma. Subspace Clustering of High Dimensional Data. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, 2004.

[Dra03] S. Draghici. *Data Analysis Tools for DNA Microarrays*. Chapman & Hall/CRC Mathematical Biology and Medicine Series, 2003.

[DS04] N. N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 864–875, 2004.

[DS05] N. N. Dalvi and D. Suciu. Answering Queries from Statistics and Probabilistic Views. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 805–816, 2005.

[DS07] N. N. Dalvi and D. Suciu. Management of Probabilistic Data: Foundations and Challenges. In *Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 1–12, 2007.

[DTS+08] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. *Proc. of the VLDB Endowment*, 1(2):1542–1552, 2008.

[Dun74] J. C. Dunn. A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact, Well-Separated Clusters. *Journal of Cybernetics*, 3(3):32–57, 1974.

[DWH01] E. Dimitriadou, A. Weingesse, and K. Hornik. Voting-Merging: An Ensemble Method for Clustering. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN)*, pages 217–224, 2001.

[EEml] European Bioinformatics Institute (EMBL-EBI). Microarray Data Resource Page, http://www.ebi.ac.uk/microarray-as/ae/browse.html.

[EKSX96] M. Ester, H. -P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.

[EKX95] M. Ester, H. -P. Kriegel, and X. Xu. A Database Interface for Clustering in Large Spatial Databases. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 94–99, 1995.

[FB03a] X. Z. Fern and C. E. Brodley. Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 186–193, 2003.

[FB03b] B. Fischer and J. M. Buhmann. Bagging for Path-Based Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(11):1411–1415, 2003.

[FB04] X. Z. Fern and C. E. Brodley. Solving Cluster Ensemble Problems by Bipartite Graph Partitioning. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 281–288, 2004.

[fCRsp] National Cancer Institute (NCI) Center for Cancer Research. Clinical Proteomics Program Databank Page (Proteomic Patterns), http://home.ccr.cancer.gov/ncifdaproteomics/ppatterns.asp.

[FGB02] A. Faradjian, J. Gehrke, and P. Bonnet. GADT: A Probability Space ADT for Representing and Querying the Physical World. In *Proc. IEEE Int. Conf. on Data Engineering (ICDE)*, pages 201–211, 2002.

[FJ02] A. L. N. Fred and A. K. Jain. Data Clustering using Evidence Accumulation. In *Proc. Int. Conf. on Pattern Recognition (ICPR)*, pages 276–280, 2002.

[FJ03] A. L. N. Fred and A. K. Jain. Robust Data Clustering. In *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 128–136, 2003.

[FL08] X. Z. Fern and W. Lin. Cluster Ensemble Selection. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, pages 787–797, 2008.

[Fod02] I. K. Fodor. A Survey of Dimension Reduction Techniques. Technical report, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 2002.

[FPS96] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 82–88, 1996.

[FPSU96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[Fre01] A. L. N. Fred. Finding Consistent Clusters in Data Partitions. In *Proc. Int. Workshop on Multiple Classifier Systems (MCS)*, pages 309–318, 2001.

[FRVG05] V. Figueiredo, F. Rodrigues, Z. Vale, and J. B. Gouveia. An electric energy consumer characterization framework based on data mining techniques. *IEEE Transactions on Power Systems*, 20(2):596–602, 2005.

[Fuh95] N. Fuhr. Probabilistic datalog—a logic for powerful retrieval methods. In *Proc. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 282–290, 1995.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[GDT09] F. Gullo, C. Domeniconi, and A. Tagarelli. Projective Clustering Ensembles. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*. TO APPEAR, 2009.

[GFdS+05] P. Geurts, M. Fillet, D. de Seny, M. A. Meuwis, M. Malaise, M. P. Merville, and L. Wehenkel. Proteomic mass spectra classification using decision tree based ensemble methods. *Bioinformatics*, 21(14):3138–3145, 2005.

[GGNZ06] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, editors. *Feature Extraction: Foundations and Applications*. Studies in Fuzziness and Soft Computing. Springer, 2006.

[GHPT05] A. Gionis, A. Hinneburg, S. Papadimitriou, and P. Tsaparas. Dimension Induced Clustering. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery in Data Mining*, pages 51–60, 2005.

[GM07] I. Gronau and S. Moran. Optimal Implementations of UPGMA and Other Common Clustering Algorithms. *Information Processing Letters*, 104(6):205–210, 2007.

[GMT07] A. Gionis, H. Mannila, and P. Tsaparas. Clustering Aggregation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 2007.

[GMW07] G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*. ASA-SIAM Series on Statistics and Applied Probability, 2007.

[GPT+07] F. Gullo, G. Ponti, A. Tagarelli, G. Tradigo, and P. Veltri. A Time Series Based Approach for Classifying Mass Spectrometry Data. In *Proc. IEEE Int. Symposium on Computer-Based Medical Systems (CBMS)*, pages 412–420, 2007.

[GPT08a] F. Gullo, G. Ponti, and A. Tagarelli. Clustering Uncertain Data via K-medoids. In *Proc. Int. Conf. on Scalable Uncertainty Management (SUM)*, pages 229–242, 2008.

[GPT+08b] F. Gullo, G. Ponti, A. Tagarelli, G. Tradigo, and P. Veltri. MSPtool: A Versatile Tool for Mass Spectrometry Data Preprocessing. In *Proc. IEEE Int. Symposium on Computer-Based Medical Systems (CBMS)*, pages 209–214, 2008.

[GPT+09a] F. Gullo, G. Ponti, A. Tagarelli, S. Iiritano, M. Ruffolo, and D. Labate. Low-voltage Electricity Customer Profiling based on Load Data Clustering. In *Proc. Int. Database Engineering and Applications Symposium (IDEAS)*, pages 330–333, 2009.

[GPT+09b] F. Gullo, G. Ponti, A. Tagarelli, G. Tradigo, and P. Veltri. Hierarchical Clustering of Microarray Data with Probe-level Uncertainty. In *Proc. IEEE Int. Symposium on Computer-Based Medical Systems (CBMS)*, 2009.

[GPT+09c] F. Gullo, G. Ponti, A. Tagarelli, G. Tradigo, and P. Veltri. MaSDA: A System for Analyzing Mass Spectrometry Data. *Computer Methods and Programs in Biomedicine (CMPB)*, 95(2):S12–S21, 2009.

[GPTG07] F. Gullo, G. Ponti, A. Tagarelli, and S. Greco. Accurate and fast similarity detection in time series. In *Italian Symposium on Advanced Database Systems (SEBD)*, pages 172–183, 2007.

[GPTG08] F. Gullo, G. Ponti, A. Tagarelli, and S. Greco. A Hierarchical Algorithm for Clustering Uncertain Data via an Information-Theoretic Approach. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 821–826, 2008.

[GPTG09a] F. Gullo, G. Ponti, A. Tagarelli, and S. Greco. A Time Series Representation Model for Accurate and Fast Similarity Detection. *Pattern Recognition*, 42(11):2998–3014, 2009.

[GPTG09b] F. Gullo, G. Ponti, A. Tagarelli, and S. Greco. Information-Theoretic Hierarchical Clustering of Uncertain Data. In *Proc. Italian Symposium on Advanced Database Systems (SEBD)*, pages 273–280, 2009.

[GRT06] S. Greco, M. Ruffolo, and A. Tagarelli. Effective and Efficient Similarity Search in Time Series. In *Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM)*, pages 808–809, 2006.

[GT06] T. Green and V. Tannen. Models for Incomplete and Probabilistic Information. *IEEE Data Engineering Bulletin*, 29(1):17–24, 2006.

[GTBC04] D. Greene, A. Tsymbal, N. Bolshakova, and P. Cunningham. Ensemble Clustering in Medical Diagnostics. In *Proc. IEEE Int. Symposium on Computer-Based Medical Systems (CBMS)*, pages 576–581, 2004.

[GTG09] F. Gullo, A. Tagarelli, and S. Greco. Diversity-Based Weighting Schemes for Clustering Ensembles. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, pages 437–448, 2009.

[GUP06] J. Galindo, A. Urrutia, and M. Piattini. *Fuzzy Databases: Modeling, Design, and Implementation*. Idea Group Publishing, 2006.

[GV03] G. L. Glish and R. W. Vachet. The basics of mass spectrometry in the twenty-first century. *Nature Reviews*, 2:140–150, 2003.

[HAK00] A. Hinneburg, C. C. Aggarwal, and D. A. Keim. What Is the Nearest Neighbor in High Dimensional Spaces? In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 506–515, 2000.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2001.

[HKT06] S. T. Hadjitodorov, L. I. Kuncheva, and L. P. Todorova. Moderate Diversity for Better Cluster Ensembles. *Information Fusion*, 7(3):264–275, 2006.

[HPM+03] S. R. Hingorani, E. F. Petricoin 3rd, A. Maitra, V. Rajapakse, C. King, M. A. Jacobetz, S. Ross, T. P. Conrads, T. D. Veenstra, B. A. Hitt, Y. Kawaguchi, D. Johann, L. A. Liotta, H. C. Crawford, M. E. Putt, T. Jacks, C. V. Wright, R. H. Hruban, A. M. Lowy, and D. A. Tuveson. Preinvasive and Invasive Ductal Pancreatic Cancer and its Early Detection in the Mouse. *Cancer Cell*, 6(4):437–450, 2003.

[HRC+05] A. -M. K. Hein, S. Richardson, H. C. Causton, G. K. Ambler, and P. J. Green. BGX: a fully Bayesian integrated approach to the analysis of Affymetrix GeneChip data. *Biostatistics*, 6:349–373, 2005.

[HY93] T. W. Hutchens and T. T. Yip. New desorption strategies for the mass spectrometric analysis of macromolecules. *Rapid Communications in Mass Spectrometry*, 7(7):576–580, 1993.

[ICS07] S. A. Imtiaz, M. A. A. Shoukat Choudhury, and S. L. Shah. Building Multivariate Model from Compressed Data. *Industrial Engineering Chemistry Research and Development*, 46(2):481–491, 2007.

[IL84] T. Imielinski and W. Lipski Jr. Incomplete Information in Relational Databases. *Journal of the ACM*, 31(4):761–791, 1984.

[J. 03] J. Sander and X. Qin and Z. Lu and N. Niu and A. Kovarsky. Automatic Extraction of Clusters from Hierarchical Clustering Representations. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, pages 75–87, 2003.

[JD88] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[Jef05] N. O. Jeffries. Algorithms for alignment of mass spectrometry proteomic data. *Bioinformatics*, 21(14):3066–3073, 2005.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A review. *ACM Computing Surveys*, 31(3), 1999.

[Jol02] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, 2002.

[JTZ04] D. Jiang, C. Tang, and A. Zhang. Cluster Analysis for Gene Expression Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(11):1370–1386, 2004.

[Kai67] T. Kailath. The Divergence and Bhattacharyya Distance Measures in Signal Selection. *IEEE Transactions on Communication Technology*, 15(1):52–60, 1967.

[KAKS97] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel Hypergraph Partitioning: Applications in VLSI Domain. In *Proc. Design Automation Conf. (DAC)*, pages 526–529, 1997.

[KAS98] K. V. Kanth, D. Agrawal, and A. Singh. Dimensionality Reduction for Similarity Searching in Dynamic Databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 166–176, 1998.

[KCHP01] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An Online Algorithm for Segmenting Time Series. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 289–296, 2001.

[KCPM01] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[KH88] M. Karas and F. Hillenkamp. Laser Desorption Ionization of Proteins with Molecular Masses Exceeding 10000 Daltons. *Analytical Chemistry*, 60:259–280, 1988.

[KH04] L. I. Kuncheva and S. T. Hadjitodorov. Using Diversity in Cluster Ensembles. In *Proc. IEEE Int. Conf. on Systems, Man and Cybernetics (SMC)*, volume 2, pages 1214–1219, 2004.

[KHT06] L. I. Kuncheva, S. T. Hadjitodorov, and L. P. Todorova. Experimental Comparison of Cluster Ensemble Methods. In *Proc. Int. Conf. on Information Fusion*, pages 1–7, 2006.

[KJ97] R. Kohavi and G. H. John. Wrappers for Feature Subset Selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[KJF97] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 289–300, 1997.

[KJNY01] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi. Low-Complexity Fuzzy Relational Clustering Algorithms for Web Mining. *IEEE Transactions on Fuzzy Systems*, 9(4):595–608, 2001.

[KJY99] R. Krishnapuram, A. Joshi, and L. Yi. A Fuzzy Relative of the k-Medoids Algorithm with Application to Web Document and Snippet Clustering. In *Proc. IEEE Int. Conf. on Fuzzy Systems (IEEE-FUZZ)*, 1999.

[KK93] R. Krishnapuram and K. Keller. A Possibilistic Approach to Clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993.

[KK98] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[KKK04] P. Kröger, H. -P. Kriegel, and K. Kailing. Density-Connected Subspace Clustering for High-Dimensional Data. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, 2004.

[KKPR05] H. -P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. Approximated Clustering of Distributed High-Dimensional Data. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, pages 432–441, 2005.

[KKRW05] H. -P. Kriegel, P. Kroger, M. Renz, and S. Wurst. A Generic Framework for Efficient Subspace Clustering of High-Dimensional Data. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 250–257, 2005.

[KKZ09] H. -P. Kriegel, P. Kröger, and A. Zimek. Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering. *ACM Trans. on Knowledge Discovery from Data (TKDD)*, 3(1):1–58, 2009.

[KL51] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[KLC+08] B. Kao, S. D. Lee, D. W. Cheung, W. -S. Ho, and K. F. Chan. Clustering Uncertain Data using Voronoi Diagrams. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 333–342, 2008.

[KP98] E. Keogh and M. Pazzani. An Enhanced Representation of Time Series which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 239–241, 1998.

[KP00] E. Keogh and M. Pazzani. Scaling up Dynamic Time Warping for Datamining Applications. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 285–289, 2000.

[KP01] E. Keogh and M. Pazzani. Derivative Dynamic Time Warping. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, 2001.

[KP05a] H. -P. Kriegel and M. Pfeifle. Density-Based Clustering of Uncertain Data. In *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 672–677, 2005.

[KP05b] H. -P. Kriegel and M. Pfeifle. Hierarchical Density-Based Clustering of Uncertain Data. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 689–692, 2005.

[KR87] L. Kaufmann and P. J. Rousseeuw. Clustering by Means of Medoids. In *Proc. Int. Conf. on Statistical Data Analysis based on the L_1 Norm*, pages 405–416, 1987.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[KS97] E. J. Keogh and P. Smyth. A Probabilistic Approach to Fast Pattern Matching in Time Series Databases. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining (KDD-97)*, pages 24–30, 1997.

[KSM00] Y. Kim, W. Street, and F. Menczer. Feature Selection for Unsupervised Learning via Evolutionary Search. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 365–369, 2000.

[Kul59] S. Kullback. *Information theory and statistics*. John Wiley & Sons, 1959.

[KWC05] E. Ka Ka Ng, A. Wai-Chee Fu, and R. Chi-Wing Wong. Projective Clustering by Histograms. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(3):369–383, 2005.

[KXWRta] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Page, http://www.cs.ucr.edu/∼eamonn/time_series_data/.

[LC03] E. L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer, 2003.

[LD08] T. Li and C. Ding. Weighted Consensus Clustering. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, pages 798–809, 2008.

[LDJ07] T. Li, C. Ding, and M. I. Jordan. Solving Consensus and Semi-supervised Clustering Problems Using Nonnegative Matrix Factorization. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 577–582, 2007.

[Lee92] S. K. Lee. An Extended Relational Database Model for Uncertain and Imprecise Information. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 211–220, 1992.

[LHY04] Y. Li, J. Han, and J. Yang. Clustering Moving Objects. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 617–622, 2004.

[Lia05] T. Warren Liao. Clustering of time series data–a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[LKLC03] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 2–11, 2003.

[LLAR07] X. Liu, K. K. Lin, B. Andersen, and M. Rattray. Including Probe-Level Uncertainty in Model-Based Gene Expression Clustering. *Bioinformatics*, 8:98, 2007.

[LLRS97] L. V. S. Lakshmanan, N. Leone, R. B. Ross, and V. S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. *ACM Transactions on Database Systems (TODS)*, 22(3):419–469, 1997.

[LLSW07] G. Liu, J. Li, K. Sim, and L. Wong. Distance Based Subspace Clustering with Flexible Dimension Partitioning. In *Proc. Int. Conf. on Data Engineering (ICDE)*, pages 1250–1254, 2007.

[LM99] R. -P. Li and M. Mukaidono. Gaussian clustering method based on maximum-fuzzy-entropy interpretation. *Fuzzy Sets and Systems*, 102(2):253–258, 1999.

[LMLR05] X. Liu, M. Milo, N. D. Lawrence, and M. Rattray. A Tractable Probabilistic Model for Affymetrix Probe-Level Analysis Across Multiple Chips. *Bioinformatics*, 21(18):3637–3644, 2005.

[LS74] L. A. Lusternik and V. J. Sobolev. *Elements of Functional Analysis*. Hindustan Publishing, 1974.

[LSS96] E. -P. Lim, J. Srivastava, and S. Shekhar. An Evidential Reasoning Approach to Attribute Value Conflict Resolution in Database Integration. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 8(5):707–723, 1996.

[LXY00] B. Liu, Y. Xia, and P. S. Yu. Clustering Through Decision Tree Construction. In *Proc. Int. Conf. on Information and Knowledge Management (CIKM)*, pages 20–29, 2000.

[Mac67] J. B. MacQueen. Some Methods for Classification and Analysis of MultiVariate Observations. In *Proc. Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[MBN02] L. C. Molina, L. Belanche, and Á. Nebot. Feature Selection Algorithms: A Survey and Experimental Evaluation. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 306–313, 2002.

[McC03] J. McConnach. Overview of draft standard for the quantification of CO2 emission credits. In *Proc. IEEE Power Engineering Society Summer Meeting*, pages 117–120, 2003.

[MCK+05] J. S. Morris, K. R. Coombes, J. Koomen, K. A. Baggerly, and R. Kobayashi. Feature Extraction and Quantification for Mass Spectrometry in Biomedical Applications Using the Mean Spectrum. *Bioinformatics*, 21(9):1764–1775, 2005.

[MEZ+05] S. Meleth, I. -E. Eltoum, L. Zhu, D. Oelschlager, C. Piyathilake, D. Chhieng, and W. E. Grizzle. Novel approaches to smoothing and comparing SELDI TOF spectra. *Cancer Informatics*, 1(1):78–85, 2005.

[MFNL03] M. Milo, A. Fazeli, M. Niranjan, and N. D. Lawrence. A Probabilistic Model for the Extraction of Expression Levels from Oligonucleotide Arrays. *Biochemical Society Transactions*, 31:1510–1512, 2003.

[MH03] J. C. Mason and D. Handscomb. *Chebyshev Polynomials*. Chapman & Hall, 2003.

[MIH08] S. Miyamoto, H. Ichihashi, and K. Honda. *Algorithms for Fuzzy Clustering*, volume 229 of *Studies in Fuzziness and Soft Computing*. Springer, 2008.

[MP00] G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics, 2000.

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. Hierarchical Clustering. In *Introduction to Information Retrieval*, pages 377–402. Cambridge University Press, 2008.

[MSE08] G. Moise, J. Sander, and M. Ester. Robust Projected Clustering. *Knowledge and Information Systems*, 14(3):273–298, 2008.

[MTP04] B. Minaei, A. Topchy, and W. Punch. Ensembles of Partitions via Data Resampling. In *Proc. Int. Conf. on Information Technology: Coding and Computing (ITCC)*, pages 188–192, 2004.

[Mur83] F. Murtagh. A Survey of Recent Advances in Hierarchical Clustering Algorithms. *Computer Journal*, 26(4):354–359, 1983.

[Mur85] F. Murtagh. *Multidimensional Clustering Algorithms*. Physica-Verlag, 1985.

[NC07] N. Nguyen and R. Caruana. Consensus Clustering. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 607–612, 2007.

[NDJR06] A. H. Nizar, Z. Y. Dong, M. Jalaluddin, and M. J. Raffles. Load Profiling Method in Detecting non-Technical Loss Activities in a Power Utility. In *Proc. IEEE Int. Power and Energy Conf. (PECon)*, pages 82–87, 2006.

[NGC01] H. S. Nagesh, S. Goil, and A. Choudhary. Adaptive Grids for Clustering Massive Data Sets. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, 2001.

[NH94] R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 144–155, 1994.

[NJW01] A. Y. Ng, M. I. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In *Proc. Int. Conf. on Neural Information Processing Systems (NIPS)*, pages 849–856, 2001.

[NKC+06] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip. Efficient Clustering of Uncertain Data. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 436–445, 2006.

[Ols95] C. F. Olson. Parallel Algorithms for Hierarchical Clustering. *Parallel Computing*, 21(8):1313–1325, 1995.

[oMHgi] Broad Institute of MIT and Harvard. Cancer Program Dataset Page, http://www.broad.mit.edu/cgi-bin/cancer/datasets.cgi.

[PAH+02] E. F. Petricoin 3rd, A. M. Ardekani, B. A. Hitt, P. Levine, V. A. Fusaro, and S. Steinberg. Use of Proteomic Patterns in Serum to Identify Ovarian Cancer. *Lancet*, 9306(359):572–577, 2002.

[Pao89] Y. -H. Pao. *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley, 1989.

[Ped90] W. Pedrycz. Fuzzy Sets in Pattern Recognition: Methodology and Methods. *Pattern Recognition*, 23(1-2):121–146, 1990.

[PH74] T. Pavlidis and S. L. Horowitz. Segmentation of Plane Curves. *IEEE Transactions on Computers*, 23(8):860–870, 1974.

[PHL04] L. Parsons, E. Haque, and H. Liu. Subspace Clustering for High Dimensional Data: A Review. *SIGKDD Explorations*, 6(1):90–105, 2004.

[PJ99] D. Pfoser and C. S. Jensen. Capturing the Uncertainty of Moving-Object Representations. In *Proc. Int. Symposium on Advances in Spatial Databases (SSD)*, pages 111–132, 1999.

[PJAM02] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proc. ACM SIGMOD Int. Conf. on Management Of Data*, pages 418–427, 2002.

[PKS+04] J. Prados, A. Kalousis, J. C. Sanchez, L. Allard, O. Carrette, and M. Hilario. Mining mass-spectra for diagnosis and biomarker discovery of cerebral accidents. *Proteomics*, 4(6):2320–2332, 2004.

[POP+02] E. F. Petricoin 3rd, D. K. Ornstein, C. P. Paweletz, A. Ardekani, P. S. Hackett, B. A. Hitt, A. Velassco, C. Trucco, L. Wiegand, K. Wood, C. B. Simone, P. J. Levine, W. M. Linehan, M. R. Emmert-Buck, S. M. Steinberg, E. C. Kohn, and L. A. Liotta. Serum Proteomic Patterns for Detection of Prostate Cancer. *Journal of the National Cancer Institute*, 20(94):1576–1578, 2002.

[PPB97] J. Parikh, J. P. Painuly, and K. Bhattacharya. Environmentally sound energy efficient strategies: a case study of the power sector in India. Technical report, Indira Gandhi Institute of Development Research, Working Paper No. 6, February, 1997.

[PPT+08] S. Pradervand, A. Paillusson, J. Thomas, J. Weber, P. Wirapati, O. Hagenbuchle, and K. Harshman. Affymetrix Whole-Transcript Human Gene 1.0 ST array is highly concordant with standard 3' expression arrays. *Biotechniques*, 44(6):759–762, 2008.

[PRH+04] E. F. Petricoin 3rd, V. Rajapaske, E. H. Herman, A. M. Arekani, S. Ross, D. Johann, A. Knapton, J. Zhang, B. A. Hitt, T. P. Conrads, T. D. Veenstra, L. A. Liotta, and F. D. Sistare. Toxicoproteomics: Serum Proteomic Pattern Diagnostics for Early Detection of Drug Induced Cardiac Toxicities and Cardioprotection. *Toxicologic Pathology*, 32 Suppl 1:122–130, 2004.

[PWJ+04] D. W. Powell, C. M. Weaver, J. L. Jennings, K. J. Mc Afee, Y. He, P. A. Weil, and A. J. Link. Cluster Analysis of Mass Spectrometry Data Reveals a Novel Component of SAGA. *Molecular and Cellular Biology*, 24(16):7249–7259, 2004.

[Ria] Rialto. Exeura S.r.l., http://www.exeura.com/rialto, 2009 edition. Easy Analytics, Ready for Decision Making.

[RJ93] L. Rabiner and B. -H. Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs, 1993.

[RJFD99] A. F. Ruckstuhl, M. P. Jacobson, R. W. Field, and J. A. Dodd. Baseline subtraction using robust local regression estimation. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 68:179–193, 1999.

[RM97] D. Rafiei and A. Mendelzon. Similarity-based Queries for Time Series Data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 13–25, 1997.

[RM98] D. Rafiei and A. Mendelzon. Efficient Retrieval of Similar Time Sequences Using DFT. In *Proc. Int. Conf. of Foundations of Data Organization (FODO)*, 1998.

[Rog07] S. Rogai. The Telegestore project: Progress and Results. In *Proc. IEEE Int. Symposium on Power Line Communications and Its Applications (ISPLC)*, 2007.

[S. 07] S. D. Lee and B. Kao and R. Cheng. Reducing UK-means to K-means. In *Proc. IEEE ICDM Workshops*, pages 483–488, 2007.

[Sad91] F. Sadri. Modeling Uncertainty in Databases. In *Proc. IEEE Int. Conf. on Data Engineering (ICDE)*, pages 122–131, 1991.

[SBHW06] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working Models for Uncertain Data. In *Proc. IEEE Int. Conf. on Data Engineering (ICDE)*, pages 7–18, 2006.

[Sch91] L. L. Scharf. The SVD and Reduced Rank Signal Processing. *Signal Processing*, 25(2):113–133, 1991.

[SD94] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[SEK04] M. Steinbach, L. Ertöz, and V. Kumar. The Challenges of Clustering High Dimensional Data. In L. T. Wille, editor, *New Directions in Statistical Physics : Econophysics, Bioinformatics, and Pattern Recognition*, pages 273–307. Springer, 2004.

[SG02] A. Strehl and J. Ghosh. Cluster Ensembles — A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research (JMLR)*, 3:583–617, 2002.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[SHR+03] D. J. Slotta, L. S. Heath, N. Ramakrishnan, R. Helm, and M. Potts. Clustering mass spectrometry data using order statistics. *Proteomics*, 3:1687–1691, 2003.

[Sib73] R. Sibson. SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method. *Computer Journal*, 16(1):30–34, 1973.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[SQL+03] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky. Automatic Extraction of Clusters from Hierarchical Clustering Representations. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, pages 75–87, 2003.

[SR62] R. R. Sokal and F. J. Rohlf. The Comparison of Dendrograms by Objective Methods. *Taxon*, 11:33–40, 1962.

[SS04] A. C. Sauve and T. P. Speed. Normalization, Baseline Correction and Alignment of High-Throughput Mass Spectrometry Data. In *Proc. Genomic Signal Processing and Statistics Conf.*, 2004.

[SSML06] D. P. De Souza, E. C. Saunders, M. J. McConville, and V. A. Liki. Progressive peak clustering in GC-MS Metabolomic experiments applied to Leishmania parasites. *Bioinformatics*, 22(11):1391–1396, 2006.

[Str88] G. Strang. *Linear Algebra and its Applications*. Brooks Cole, 1988.

[SYF05] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. FTW: Fast Similarity Search under the Time Warping Distance. In *Proc. ACM*

SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), pages 326–337, 2005.

[SZ04] K. Sequeira and M. Zaki. SCHISM: A New Approach for Interesting Subspace Mining. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 186–193, 2004.

[TA77] A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-posed Problems*. Winston & Sons, 1977.

[TBA+02] A. Tefferi, M. E. Bolander, S. M. Ansell, E. D. Wieben, and T. C. Spelsberg. Primer on medical genomics. part iii: Microarray experiments and data analysis. *Mayo Clinic Proceedings*, 77(9):927–940, 2002.

[TCS04] N. F. Thornhill, M. A. A. Shoukat Choudhury, and S. L. Shah. The impact of compression on data-driven process analyses. *Process Control*, 14(4):389–398, 2004.

[THD07] G. J. Tsekouras, N. D. Hatziargyriou, and E. N. Dialynas. Two-Stage Pattern Recognition of Load Curves for Classification and Electricity Customers. *IEEE Transactions on Power Systems*, 22(3):1120–1128, 2007.

[TJP03] A. P. Topchy, A. K. Jain, and W. Punch. Combining Multiple Weak Clusterings. In *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pages 331–338, 2003.

[TJP05] A. P. Topchy, A. K. Jain, and W. F. Punch. Clustering Ensembles: Models of Consensus and Weak Partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(12):1866–1881, 2005.

[TK99] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 1999.

[TSD+06] M. K. Titulaer, I. Siccama, L. J. Dekker, A. L. C. T. van Rijswijk, R. M. A. Heeren, P. A. Sillevis Smitt, and T. M. Luider. A database application for pre-processing, storage and comparison of mass spectra derived from patients and controls. *BMC Bioinformatics*, 7, 2006.

[TSTK08] G. J. Tsekouras, A. D. Salis, M. A. Tsaroucha, and I. S. Karanasiou. Load time-series classification based on pattern recognition methods. In Peng-Yeng Yin, editor, *Pattern Recognition Techniques, Technology and Applications*, pages 361–432. IN-TECH, 2008.

[TW81] J. Tind and L. A. Wolsey. An Elementary Survey of General Duality Theory in Mathematical Programming. *Mathematical Programming*, 21(3):241–261, 1981.

[TXC07] Y. Tao, X. Xiao, and R. Cheng. Range Search on Multidimensional Uncertain Data. *ACM Transactions on Database Systems (TODS)*, 32(3):15–62, 2007.

[TXO05] A. K. Tung, X. Xu, and B. C. Ooi. CURLER: Finding and Visualizing Nonlinear Correlation Clusters. In *Proc. ACM SIGMOD Int. Conf. on Management Of Data*, pages 467–478, 2005.

[UCA09] M. Ullman-Cullere, E. Clark, and S. Aronson. Implications of Genomics for Clinical Informatics. In *Encyclopedia of Database Systems*, pages 1400–1404. Springer US, 2009.

[van79] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 2 edition, 1979.

[VGK02] M. Vlachos, D. Gunopulos, and G. Kollios. Discovering Similar Multidimensional Trajectories. In *Proc. Int. Conf. on Data Engineering (ICDE)*, pages 673–684, 2002.

[Voo86] E. M. Voorhees. Implementing Agglomerative Hierarchical Clustering Algorithms for Use in Document Retrieval. *Information Processing and Management*, 22:465–476, 1986.

[VP07] J. Valente de Oliveira and W. Pedrycz. *Advances in Fuzzy Clustering and its Applications*. John Wiley & Sons, 2007.

[W. 70] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and their Applications. *Biometrika*, 57(1):97–109, April 1970.

[WAA00] Y. Wu, D. Agrawal, and A. Abbadi. A Comparison of DFT and DWT Based Similarity Search in Time-Series Databases. In *Proc. ACM Int. Conf. on Information and Knowledge Management (CIKM)*, pages 488–495, 2000.

[WAF+03] B. Wu, T. Abbott, D. Fishman, W. Mc Murray, G. Mor, K. Stone, D. Ward, K. Williams, and H. Zhao. Comparison of Statistical Methods for Classification of Ovarian Cancer Using Mass Spectrometry Data. *Bioinformatics*, 21(13):1636–1643, 2003.

[WCC05] J. W. H. Wong, G. Cagney, and H. M. Cartwright. SpecAlign - processing and alignment of mass spectra datasets. *Bioinformatics*, 21(9):2088–2090, 2005.

[WCD+05] B. Williams, S. Cornett, B. M. Dawant, A. Crecelius, B. Bodenheimer, and R. Caprioli. An algorithm for baseline correction of MALDI mass spectra. In *Proc. ACM Southeast Regional Conf.*, pages 137–142, 2005.

[Wid05] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *Proc. Int. Conf. of Data Systems Research (CIDR)*, pages 262–276, 2005.

[WKG04] W. E. Wallace, A. J. Kearsley, and C. M. Guttman. An Automated Peak Identification/Calibration Procedure for High-Dimensional Protein Measures From Mass Spectrometers. *Analytical Chemistry*, 76(9):2446–2452, 2004.

[WLKL04] K. -G. Woo, J. -H. Lee, M. -H. Kim, and Y. -J. Lee. FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[WNP03] M. Wagner, D. Naik, and A. Pothen. Protocols for Disease Classification from Mass Spectrometry Data. *Proteomics*, 3(9):1692–1698, 2003.

[WNP+04] M. Wagner, D. N. Naik, A. Pothen, S. Kasukurti, R. R. Devineni, B. L. Adam, O. J. Semmes, and G. L. Wright. Computational protein biomarker prediction: a case study for prostate cancer. *BMC Bioinformatics*, 5, 2004.

[WSB09] H. Wang, H. Shan, and A. Banerjee. Bayesian Cluster Ensembles. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, pages 209–220, 2009.

[WSCY99] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and Querying Databases that Track Mobile Units. *Distributed and Parallel Databases*, 7(3):257–387, 1999.

[WY05] W. Wang and J. Yang. Mining High-Dimensional Data. In *The Data Mining and Knowledge Discovery Handbook*, pages 793–799. Springer, 2005.

[WZ07] W. Wang and Y. Zhang. On Fuzzy Cluster Validity Indices. *Fuzzy Sets and Systems*, 158(19):2095–2117, 2007.

[XCCH02] F. Xiaodong, C. Changling, L. Changling, and S. Huihe. An Improved Process Data Compression Algorithm. In *Proc. Intelligent Control and Automation Conf.*, pages 2190–2193, 2002.

[XH07] L. Xiao and E. Hung. An Efficient Distance Calculation Method for Uncertain Objects. In *Proc. IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 10–17, 2007.

[Yan93] M. S. Yang. A Survey of Fuzzy Clustering. *Mathematical and Computer Modelling*, 18:1–16, 1993.

[YCN04] K. Y. Yip, D. W. Cheung, and M. K. Ng. HARP: A Practical Projected Clustering Algorithm. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(11):1387–1397, 2004.

[YCN05] K. Y. Yip, D. W. Cheung, and M. K. Ng. On Discovery of Extremely Low-Dimensional Clusters using Semi-Supervised Projected Clustering. In *Proc. Int. Conf. on Data Engineering (ICDE)*, pages 329–340, 2005.

[YF00] B. K. Yi and C. Faloutsos. Fast Time Sequence Indexing for Arbitrary L_p Norms. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 385–394, 2000.

[YM05] M. L. Yiu and N. Mamoulis. Iterative Projected Clustering by Subspace Mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(2):176–189, 2005.

[YMA+03] Y. Yasui, D. McLerran, B. L. Adam, M. Winget, M. Thornquist, and Z. Feng. An Operator-Independent Approach to Mass Spectral Peak Identification and Integration. *Journal of Biomedicine and Biotechnology*, 4:242–248, 2003.

[YY04] X. Yu and I. Yoo. Cluster Ensemble and its Applications in Gene Expression Analysis. In *Proc. Asia-Pacific Bioinformatics Conf. (APBC)*, pages 297–302, 2004.

[Zad65] L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8(3):338–353, 1965.

[ZAHB00] H. Zheng, S. S. Anand, J. G. Hughes, and N. D. Black. Methods for Clustering Mass Spectrometry Data in Drug Development. In *Proc. Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, 2000.

[ZLPZ08] W. Zhang, X. Lin, J. Pei, and Y. Zhang. Managing Uncertain Data: Probabilistic Approaches. In *Proc. Int. Conf. on Web-Age Information Management (WAIM)*, pages 405–412, 2008.

[ZTGG02] Y. Zeng, J. Tang, J. Garcia-Frias, and G. R. Gao. An Adaptive Meta-Clustering Approach: Combining the Information from Different Clustering Results. In *Proc. IEEE Computer Society Bioinformatics Conf. (CSB)*, pages 330–332, 2002.