To my family

Acknowledgements

This thesis is the result of hard work that lasted three years at the Institute of High Performance Computing and Networking of the National Research Council (ICAR - CNR) and at the Department of Electronics, Computer and Systems Sciences of the University of Calabria (DEIS - UNICAL). So proper thanks go to those two institutes. In particular I would like to thank my supervisors prof. Domenico Saccà and dr. Giuseppe Manco, ICAR Director prof. Domenico Talia and the coordinator of the PhD course prof. Luigi Palopoli.

I also thank my friends. The friends of a lifetime: Francesco Spina and Pierluigi Bastanzio. They have always been close to me. The distant friends Alessio Cozzolino, Michele Mastroianni, Cristian D'Andrea and Vincenzo Russo. The friends met at university: Giuseppe Sergi, Giovanni Salerno, Giuseppe Magurno, Marco Lavecchia, Giuseppe Iannello, Roberto Betrò, Sergio Giampietro (the famous 7 of London + Ettore, or 7 of Rome + Gimo), but also Annalisa Perrotta, Andrea Gentile, Alessandra Talarico and Loredana Iorfida.

Special thanks to my colleagues, without whom I could not fully carry out my research: Massimo Guarascio, Gianni Costa, Riccardo Ortale, Giuseppe Manco, Nicola Barbieri, Alfredo Cuzzocrea. I would like to say thanks to my new friends: Luca Ghionna, Fausto Pupo, Francesco Riccetti, Milena Guzzo, Umberto Scilinguo, Maria Ventura, Francesco Valentini, Francesco Santoro, Giuseppe Papuzzo, Giovanni Cannataro.

Last, but most important, I would like to thank my parents Francesco and Teodora, and my sister Tiziana that have always supported me even in moments of despair, and finally I would like to extend greetings to my sister Jacqueline and her family.

Contents

1

Introduction

1.1 Background and Motivations

An information society [17], such that we live in, is a society in which information has a great impact on daily life. We are entering a new phase of society [17]: we are moving from an industrial economy to a *knowledge economy* whereby wealth is created through the economic exploitation of understanding.

According to *Internet World Stats*[1] currently there are more than 1,966,514,816 internet users (June 30, 2010). These users generate a great quantity of data. Few examples are (year 2009) [2]:

- 90 trillion - The number of emails sent on the Internet in 2009.
- 247 billion - Average number of email messages per day.
- 234 million - The number of websites as of December 2009.
- 47 million - Added websites in 2009.
- 126 million - The number of blogs on the Internet.
- 350 million - People on Facebook.
- 4 billion - Photos hosted by Flickr (October 2009).
- 12.2 billion - Videos viewed per month on YouTube in the US (November 2009).

[1] http://www.internetworldstats.com/stats.htm

[2] Website stats from Netcraft (`http://news.netcraft.com/archives/2009/12/24/december_2009_web_server_survey.html`).

Email stats from Radicati Group (`http://www.radicati.com/?p=3237`). Malware stats from Symantec (`http://eval.symantec.com/mktginfo/enterprise/other_resources/b-symc_intelligence_quarterly_oct-dec_2009_20949850.en-us.pdf`) and McAfee (`http://www.mcafee.com/us/local_content/reports/7315rpt_threat_1009.pdf`). Online video stats from YouTube (`http://youtube-global.blogspot.com/2009/10/y000000000utube.html`).

Photo stats from Flickr (`http://blog.flickr.net/en/2009/10/12/4000000000/`) and Facebook (`http://www.facebook.com/press/info.php?statistics`). Social media stats from BlogPulse (`http://www.blogpulse.com/`) and Facebook

- 2.6 million - Amount of malicious code threats at the start of 2009 (viruses, trojans, etc.).

These numbers highlight two increasing needs:

1. Information management, which consists in the creation, distribution, diffusion, use, integration and manipulation of data.
2. Discovery of useful knowledge from this information.

In this scenario, new application contexts arise, whose solutions represent new forms of economic gain for industry and new challenges for researchers. Examples are bioinformatics, fraud detection, collaborative filtering, text analysis, spatial and temporal data analysis, and so on. Each one of these contexts, due to their own intrinsic difficulty, needs specific solutions aimed to effectively and efficiently resolve the related problems; very often solutions require a domain knowledge of the application context environment.

The goal of the thesis is to study some of these contexts, where knowledge discovery is needed, and to provide them formal and organic solutions.

1.2 Main Contributions

The first effort, of the thesis, is to provide a formalization for the process of knowledge discovery, through the definition of a formal language for data mining [67].

This need arises from the fact that, although in recent years data mining has become increasingly in research, yet to date it has no standard formalization, recognized by the scientific community. This means that, unlike the related problem of Query Answering [77], with its Relational Calculus [36], Relational Algebra [5] and SQL [14], data mining is associated with no formal language and no operational algebra. The reason is rather obvious: data mining tasks are incredibly heterogeneous (in the input data format, data structures, algorithms, etc), so it is very difficult to define basic operators that can support the birth of an algebra and that don't affect the complexity of the operations (dealing with large amounts of data, there are practical constraints on the complexity of the algorithms).

The proposed language comes from the Object Oriented SQL [69] and enriches its grammar with some operators that allow to express mining processes. The language is based on a formally defined algebra, which aims to unite the world of data and information with the world of concepts and knowledge.

After the definition of this inductive query language, the thesis tries to analyze some application contexts and tries to give them solutions that could be competitive with the current literature. The studied problems are three: classification in hostile environments, collaborative filtering and tree-based data mining.

Classification in hostile environments

Given a domain, whose elements can be grouped into known categories, *classification* is defined as the process of assigning domain samples, whose category (class)

is not known, to one of the categories of the domain. Classification consists of two phases:

1. Generation of mathematical models (classifiers) from a subset of the domain (called training set), whose elements exhibit the category they belong to.
2. Assignment to the categories of new domain elements whose classes are unknown, exploiting the mathematical models defined in the previous phase.

Classification in hostile environments has mainly three difficulties: low occurrence in the domain of cases of interest, low separability between the cases of interest and the rest of the samples, the presence of noise. The difficulty lies in the definition of mathematical models, since in this context, the statistical theory fails for lack of sufficient samples and their common characteristics, and for errors in the data.

The thesis proposes two solutions. One solution is a hierarchical model based on an associative classifier whose accuracy is improved by a series of probabilistic classifiers. The second solution is based on a maximum entropy model [20] whose features are determined by association rules discovered in the data.

Collaborative filtering

Recommender systems are applications born for that companies that offer a range of products/services to a user base. Their goal is to answer to this question: given a specific user, which are the products/services that she has not yet purchased, but for which she may show a strong interest?

Giving an answer to this question is not simple, and in many cases, the currently-in-use recommender systems answer by using collaborative filtering techniques [97]. These techniques are a family of mathematical models that are based on past purchases of the users without considering additional information. The basic assumption is a principle of conservation of the users' tastes: users who have shown similar preferences in the past, most likely will continue to show similar preferences in the future.

All data in this case can be seen as a big matrix M, where the u-th row represents the preferences of a user u, while the i-th column represents the set of preferences given to an item (product or service) i. The preferences can be quantified with a real or natural number. The biggest problem, associated with this data structure, is the sparsity of M: not all users give a preference to all items, indeed the total number of votes recorded in M is considerably smaller than the size of the matrix.

The thesis proposes a model for collaborative filtering based on latent factors. Its scheme is hierarchical and differs from the state-of-the-art approaches in the literature for its flexibility and for the reduction of imposed constraints. The algorithm is a coclustering (or biclustering) approach [25], which aims to minimize the mean square error of prediction procedure via an Expectation Maximization Model [22].

Tree-based data mining

In some context, data mining tasks need of domain knowledge in order to get good accuracies, hence, it's crucial to be able to manage this knowledge. Today, with the so

called Semantic Web and Internet of the Future, more and more of this knowledge is stored in a format become a de facto standard: XML (Extensible Markup Language).

In these years, researches proposed new formulations and formalization of XML mining. The difficulty lies in the fact that the XML format is tree-structured and not flat like the tuples in a database. It's essential to define what are the features on which to base the techniques of mining. Moreover, given this nature of tree-like structure, it becomes incredibly important to pay attention to the computational complexity of the defined algorithms (since the structure of an XML document can be very complex).

The thesis proposes a model for the mining of tree structured data: its goal is to show a hierarchical clustering that can be competitive with the current literature and can simplify the complexity of the data analysis. The model addresses the problem in several steps:

1. first defines the features to separately analyze; in order nodes, edges and paths.
2. for each feature launches mining tasks.
3. returns the results of the entire analysis.

1.3 Organization

The the thesis is structured as follows.

- Chapter 2 presents an overview of the aforesaid topics studied in this work;
- Chapter 3 presents a framework for a definition of a language for the knowledge discovery process;
- Chapter 4 presents the two proposed algorithms for the classification in imprecise environments;
- Chapter 5 proposes an hierarchical approach for collaborative filtering;
- Chapter 6 shows an algorithm for clustering of tree-base data mining (XML documents);
- Finally, Conclusions and Future Works are in chapter 7

2

Knowledge Base

In this chapter, a view of the state-of-art of the literature with regard to the aforesaid topics is presented.

2.1 Languages for data mining

The development of suitable data mining query languages has been lately investigated from different perspectives, with two main objectives [53]. On the one hand, focus is to provide an interface between data sources and data mining tasks. Under this perspective, a data mining query language is seen as a standard mean for specifying data sources, patterns of interest and properties characterizing them. On the other hand, a data mining query language is meant to support the design of specific procedural workflows, which integrate reasoning on the mining results and possibly define ad-hoc evaluation strategies and activations of the data mining tasks. The underlying idea here is to embody data mining query languages in a more general framework, where effective support to the whole knowledge discovery process is provided.

An influential foundation for data mining is the `3W Model`, originally introduced into [64] and subsequently refined by [26], that is reviewed next.

`3W Model` stands for Three Worlds for data mining: the D(ata)- world, the I(ntensional)-world, and the E(xtensional)-world. The `D-World` represents the raw data to be analyzed in terms of the basic entities of relational algebra, i.e. relational schemas and extensions. The attributes of such entities are associated with corresponding domains, that can be either categorical or numeric. Most activities, carried out in the preprocessing phase of a typical knowledge discovery application, can be modeled by means of specific operators of an extended relational algebra, that adds to the usual algebraic operators.

Objects in the `I-World` represent, instead, a particular class of data mining models, i.e. regions that can be defined as (sets of) conjunctions of linear inequality constraints on the attributes of the entities in the `D-World`. Starting from a set of basic regions, further regions can be formed via the definition of composition operators.

In the E-World, a region is simply represented as an enumeration of all the tuples belonging to that region. Relations in this world are obtained by combining the relations of the two worlds previously defined, so that the schema of the resulting relation is the union of the schemas of some relation in the D-World and some other relation in the I-World. Thus, the resulting 3W Model can be specified as a set of three worlds: the D-World (data world), the I-World (intensional world), and the E-World (extensional world). Entities in the three aforesaid worlds can be related via suitable inter-world operators. Precisely, a generic mining operator $regionize$ extracts regions in the I-World from data in the D-World. These regions can be iteratively refined by means of a mining loop from the I-World to the I-World. The population operator POP creates a relation in the E-World, starting from some regions in the I-World and some other relations in the D-World. Finally, composite objects of the E-World can be projected to the other two worlds via the operators π_{RDA} and π_A, that allow to return in the I-World and D-World, respectively, via a simple selection of the proper attributes (data or constraints) within the E-World relation.

The 3W Model is mightily interesting for many reasons. Foremost, it provides a view of data mining in algebraic terms: a knowledge discovery process is the application of a sequence of operators in order to transform a set of tables. Furthermore, it is also fascinating from a methodological point of view: the object representation of 3W Model entities and the implementation of a suitable set of operators are key elements in the design of a powerful tool for knowledge discovery. However, some major limitations affect the 3W Model. In the D-World there is no possibility to express complex relations (i.e. cyclic relation), because the nesting of this data model has a fixed depth. Furthermore, a more serious limitation lies in the I-World, where regions are expressed by linear inequality sets. This means that fundamental mining models are not expressible, since their representations require more complex mathematic structures (i.e. SVM and clustering results, time point series, surrounding regions and so forth). This thesis, in chapter 3, proposes an algebra, namely 2W Model, that avoids both the aforesaid limitations of the 3W Model. Indeed, it enables the description of complex objects and their properties and also supports the extraction all required patterns from raw data.

2.2 Classification in hostile environments

A wealth of approaches to learning classification models within imprecise domains exists in the literature, whose emphasis is mainly at addressing the issues related to class imbalance and different misclassification costs.

Cost-sensitive learning methods [38, 85] have been explored for accounting the issues related to rare classes and different misclassification costs. The idea is to bias the learning process towards rare classes by assigning an appropriately higher value to the recognition of the minority class(es) with respect to the identification of the majority class(es). The resulting classification model has hence broader decision regions associated to the minority class(es), would boundaries are suitably extended

via the specification of misclassification costs to cover more cases from the minority class(es), even if at the expense of an increased number of (misclassified) cases from the majority class(es). Nonetheless, the domain-specific information on the individual misclassification costs is seldom known or hardly quantifiable in an objective manner whenever related to domain experts' subjectiveness.

Various specific evaluation metrics have been also investigated for dealing with rare classes and different misclassification costs. The starting point here is that classification accuracy is not well-suited for imprecise domains, since it is strongly biased against rare classes and assumes equal misclassification costs. This has caused the widespread use of some alternative metrics in imprecise domains. ROC analysis is commonly used in machine learning for visualizing and evaluating the performance of classifiers. In particular, within an imprecise domain, the ROC space allows to decouple classifier performance from knowledge of both class and cost distributions. The overall performance can be summarized into a single figure, namely the Area Under the ROC Curve (AUC), which is not biased in favor of the majority classes. The approach in [89] proposes an elegant framework that combines ROC analysis, decision analysis and computational geometry for robust classification in imprecise domains. However, a disadvantage of the method is that it requires the apriori identification of some classifiers, whose ROC curves are dominating for certain operating conditions. This clearly involves the selection and exploitation of different induction schemes to learn as many classification models under various operating conditions as well as their experimental evaluation for the purpose of identifying those areas of the ROC space, in which the curve of one classifier dominates over the others. Such a preliminary process also impacts the time efficiency of building the ROC convex hull.

Sampling involves altering the original class distribution for the purpose of attenuating or removing rarity. There two basic forms of sampling. In particular, undersampling [73] aims at filtering cases from the majority classes, while retaining the initial population of the minority classes. Oversampling [60] is instead devoted to replicate examples from the minority classes. Both methods have disadvantages. Precisely, undersampling wastes potentially significant examples from the majority classes that may be useful to enforce class separation, thereby hindering the performance of the resulting classifier. Oversampling prevents from missing certain portions of the data space, in which a very small number of cases from the minority classes are located. This leads to the formation of the associated decision regions. Replication clearly involves augmenting the duration of the learning process. Also, since no new information is injected into the training data, oversampling is also susceptible to overfitting especially when data is noisy. In some circumstances, this could lead to the formation of classification rules that cover one replicated case.

Advanced sampling methods have also been considered. In particular, undersampling for majority classes is coupled in [29] with a special form of oversampling for the minority classes, that creates new synthetic cases from these latter classes. The technique is effective at inducing a stronger generalization for the minority classes which neatly contrasts to the specialization induced by pure replication. However, it is still susceptible to overfitting.

Progressive sampling [105] approximates the best class distribution for learning by iteratively adding to some initial training data a certain proportion of cases from the majority and minority classes, by using a geometric sampling schedule. The method is empirically proven to converge towards a nearly optimal class distribution for training. Nonetheless, it assumes the existence of costs for procuring additional training data and thus it is actually useful when procurement costs are known.

Cost-sensitive boosting [98] has been considered for addressing two major characteristics of imprecise environments, namely the rare classes and the different misclassification costs. Boosting is an iterative meta-technique for learning ensemble classifiers, that associates a weight with each training data. Weights determine the probability with which the corresponding training cases are adaptively sampled at each iteration for the purpose of forming a new dataset. The latter is used to learn a classifier through the application of some basic learning scheme. Cost-sensitive boosting lends to being used for improving the recognition of minority classes [49, 66, 30], since the latter are more error-prone w.r.t. to majority classes and, hence, their weights are suitably increased. While weight updating is uniform in pure boosting, i.e. no focus is paid on differentiating between correct and incorrect predictions of a certain kind, cost-sensitive boosting assigns varying weights to training cases on the basis of their classifications (e.g., in the two-class scenario, TP, FP, TN and FN). The weight updating process in [49] incorporates a misclassification cost adjustment function: the weights assigned to misclassified (resp. classified) cases from a minority class are more aggressively (resp. conservatively) increased (resp. decreased) with respect to the ones associated to misclassified (resp. classified) cases from majority classes. However, since no distinction is made between cases from a minority class that are incorrectly classified into a majority class and the viceversa, the approach in [49] may overly favor recall at the expense of a much lower precision. The latter limitation is avoided in [66] through a finer weight modification scheme.

A criticism to such approaches is that cost-sensitive boosting may incur into overfitting [103], by progressively increasing the weights for cases of the minority classes that are misclassified. For the purpose of avoiding overfitting and better catching the minority class, synthetic creation of cases of the minority class and boosting are combined in [30]. At each boosting iteration, a certain amount of artificial cases from the minority class are created. This allows to sample a higher number of cases from such a class, which ultimately enables the basic learning scheme to focus more on (i.e. to learn more general decision regions for) the minority class without modifying the weights of the training cases. However, it is not clear how to establish the appropriate amount of synthetic minority-class cases to generate. Besides the specific disadvantages of the enumerated methods, cost-sensitive boosting presents some general weaknesses when used for learning classification models in an imprecise domain. One such a weakness follows from the well-known inability of boosting at properly working in the presence of noise. Additionally, there is not general guarantee that it can improve the recognition of the rare class(es) since its performance is strictly dependent on the performance of the basic learning scheme. If the underlying scheme always achieves low recall or precision on the rare class(es) of an imprecise domains, the performance of boosting is also poor [65].

Finally, segmentation [103] is another major method for catching rare classes in imprecise domains. The underlying idea is to suitably divide the data space into disjoint regions, wherein globally rare classes tend to become less rare. Within each such a region there are two possibilities. The density of rarity is (much) higher w.r.t. the density in the whole training data. This clearly allows to focus on the rarities local to the region, which are also less affected by noise. Alternatively, the density of rarity is (much) lower than the corresponding density in the training data. In this circumstance, rarity becomes nearly unidentifiable in the specific region. Nonetheless, this is acceptable in practice, since most of the original class rarity is still captured within other regions.

2.3 Collaborative Filtering Approaches

In this section a brief discussion of the most used techniques for rating prediction in the collaborative filtering, namely *Baseline*, *Nearest Neighbors* and *Latent Factor* models, is provided.

Baseline models are basic techniques to compute a rating prediction and are considered a first step towards the rating personalization and user profiling. These first approaches to rating prediction are summarized in table 2.1, where μ denotes the overall mean of the ratings, $\overline{r}_i$ is the average rating given on the item i and, symmetrically, $\overline{r}_u$ is the average of the ratings given by the user u. Another simple and

Baseline	Personalization	Prediction $\hat{r}_i^u$
OverallMean	None	μ
ItemAvg	Item-oriented	$\overline{r}_i$
UserAvg	User-oriented	$\overline{r}_u$
WeightedCentering	Item&User oriented	$\alpha\,\overline{r}_i + (1-\alpha)\overline{r}_u$ $0 \leq \alpha \leq 1$

Table 2.1. Baseline Approaches

effective baseline approach has been proposed in [16], which describes a set of global effects that might influence user's ratings. For example, some users might tend to assign higher or lower ratings to items respect to their rating average (which is known as *User effect*), while some items tend to receive higher (or lower) rating values than others (*Item effect*).

Neighborhood based approaches based on explicit user feedback are the most commonly used techniques for generating suggestions and predictions. According to the item-based version [94], the predicted rating is computed by aggregating the ratings given by each user on the most similar items to the considered item. The underlying assumption is that the user might prefer items more similar to the ones she liked before, because they might belong to the same category or might share similar features. More formally:

$$\hat{r}_i^u = \frac{\sum_{j \in \mathcal{N}^K(i;u)} s_{ij} \cdot r_j^u}{\sum_{j \in \mathcal{N}^K(i;u)} s_{ij}}$$

where $\mathcal{N}^K(i;u)$ is the set of K items rated by the user u most similar to i, s_{ij} is the similarity coefficient between the item i and j, and r_j^u is the rating given by the user u to the item j. Similarity coefficients are compute on a global basis: two products are considered similar if they have received similar preference values from common users. Hence, this strategy fails in recognizing local item similarity within a same user community. Some alternative and more effective formulations of the neighborhood based approach have been proposed in [16, 70, 15]; the key idea is to determine the interpolation weights simultaneously according to a global optimization schema, which better reflects intra neighborhood relationships.

The assumption behind *Latent Factor models* is that the rating value can be expressed by considering a set of contributes which represent the interaction between a user and the target item on a set of features. Assuming that there are a set of K features which determine the user's preference on an item, the prediction can be generated as:

$$r_i^u = \sum_{z=1}^{K} U_{u,z}\, M_{z,i}$$

where $U_{u,z}$ is the response of the user u to the feature z and $M_{z,i}$ is the response on the same feature by item i. Several learning scheme have been proposed to overcome the sparsity of the original rating matrix and to produce accurate predictions. The learning phase relies either on a *gradient descent* error-minimization [52, 7] or a likelihood optimization procedure (based e.g., on Gibbs Sampling or Expectation Maximization). The peculiarity of a probabilistic model is the capability of estimating either the joint probability $P(r,u,i)$ (probability that a user u gives rating r to the item i) or the conditional probability $P(r|u,i)$ (given a user u and an item i, the probability of rating = r). The identification of $\hat{r}_i^u$ for a pair $\langle u,i\rangle$ can hence be computed as:

$$\hat{r}_i^u = E\left[r|u,i\right] = \sum_r r \cdot P(r|u,i)$$

where $E\left[.\right]$ is the expected value operator.

The *pLSA* (probabilistic latent semantic analysis, or *Aspect Model*) proposed by Hoffman in [58], is the reference probabilistic approach to CF. The underlying assumption is that the observed user preferences can be modeled as a mixture of user communities, and each user can be included into one or more groups[58]. Introducing a latent variable Z (ranging over K possible states) and assuming that user U and item I are conditionally independent given the state of Z, the probability of observing rating r for the pair (u,i) can be computed as:

$$P(r|u,i) = \sum_{z=1}^{K} P(r|i,Z=z)P(Z=z|u)$$

where $P(Z=z|u)$ represents the interest of u to topic z, and $P(r|i,Z=z)$ is the probability that a user belonging to pattern *z* gives rating *r* on the item *i*.

The *User Profile Model* extends this formulation by employing Dirichlet priors which provide a full generative model at the user level [78, 79]; a further adoption of the Latent Dirichlet Allocation approach [24] which includes a response variable, in this case the ratings on an item, has been proposed in [23].

Recently, novel probabilistic approaches [93, 2, 95] have been proposed to overcome the need for regularization and in order to prevent overfitting in matrix factorization methods. In particular, the *Probabilistic Matrix Factorization* [93] proposes a generative gaussian model for ratings, in the low-rank latent space of users and items. Extensions of this model include bayesian priors [92] and non-linear matrix factorization with gaussian processes [74].

Other works focus on combining preference data and content features [63, 3, 96] to produce more accurate recommendations and to address the cold-start problem. The underlying idea is to associate items and users with content-specific latent factors and thus to use this low-dimensional feature representations for regularization.

So far, co-clustering approaches exhibited limited predictive capability (clustering both items and users makes these approaches more prone to overfitting). In addition, the high computational burden make them unfeasible for realistic problems. [54] proposes simultaneous clustering of users and items based on an adaptation of the *Bregman coclustering*[11]: given an initial co-clustering assignment, the user-clusters (rows) and item-clusters (columns) are alternately optimized till convergence is achieved. A probabilistic approach to determine user-item memberships following a coclustering strategy has been discussed in [58]: the assumption behind the *Two-Sided Clustering Model* is that the rating value is independent of the user and item identities given their respective cluster memberships. The clustering approach is based on a standard EM likelihood maximization procedure.

Both these co-clustering approaches assume the existence of a unique partition over the item-set and the number of user-communities. Within these models, each user belongs to exactly a single user-community and each item belong to a single groups of item. By contrast, the *Flexible Mixture Model (FMM)* [62] extends the two-sided model by allowing an individual (either a user or an item) to be included in different clusters, with different degrees of membership. A novel approach to co-clustering have been proposed in [88];the resulting model, known as *Bi-LDA*, integrates Dirichlet priors and discovers simultaneous groups of users/items modeled via LDA.

2.4 Tree-based data mining

Analyzing tree-based data has become a very important research field, especially because a lot of information is recorded within XML document, that has become a frequently used format in the web. The main feature of the XML format is its structure, that allows to represent a lot of entities.

Hierarchical clustering has been widely adopted for grouping XML documents by structure [41, 46, 43, 47], because of the high quality of its results.

However, a major criticism to such proposals is that hierarchical clustering is impractical for processing large-scale databases of XML documents, being its time complexity is $O(N^2)$, where N represents the size of the available collection $\mathcal{D}$ of XML documents. This basic time requirement is further exacerbated in those approaches, such as [46], that compute similarity between XML trees or their structural summaries through some variant of the tree-edit distance [108]. The latter essentially involves computing the minimum-cost sequence of operations, necessary to transform one entity into the other. This is prohibitively expensive in the aforesaid setting, i.e., at least quadratic in the number of nodes within both entities [56].

As far as cluster summarization is concerned, the representative introduced in [41] actually catches all structural properties in a cluster of XML documents. However, it is computationally expensive, both in time and space. In particular, its time complexity is proportional to the product of the number of nodes in the representatives associated to the two least dissimilar clusters to be merged in the hierarchy.

A main limitation of [47] is the loose-grained similarity catched by the notion of s-graph, which is exploited to summarize clusters of XML documents. Indeed, two XML documents can share the same prototype s-graph and still have significant structural differences, such as in the hierarchical relationship between nodes. This has an undesirable effect, i.e., that structurally heterogeneous XML documents may be placed within a same cluster, with a consequent degrade of clustering quality.

No emphasis is paid in [46, 43] on providing an intelligible description of the discovered clusters.

XProj [39] is instead a partitioning method in which k clusters are formed around their representatives. These are collections of substructures, with a fixed number n of nodes, that frequently occur in the respective clusters and need be recomputed at each relocation of the XML documents.

A main shortcoming of *XProj* is the presence of many input parameters that require a careful tuning, namely the number k of clusters, the size n of the frequent substructures in a cluster representative and the minim frequency threshold for the substructures themselves. An improper setting of these parameters makes the discovery of clusters in the data problematic and too dependent on the characteristics of the available XML documents. Unfortunately, there is no general tuning for such parameters: a setting may ensure an acceptable performance over a dataset with certain structural properties and deliver very poor performances on other datasets with even small changes in their properties.

Also, in *XProj*, the notion of cluster representative is functional to some degree of cohesiveness of the intermediate clusters of XML documents. It is not meant for providing an understanding of their structural properties. This justifies two strong approximations: the inclusion in the representatives of structures only with a fixed size n (with a consequent loss of structural information from the corresponding clusters), that may also be unrepresentative (i.e., such that their edge sequences are subsequences of the edge representations of the XML documents in the clusters, although the structures themselves are not substructures of the XML documents).

3

Inductive Query Language

The thesis proposes a foundational model for the knowledge discovery process, called the `2W Model`, that enables progressive data-mining workflows and, thus, provides the underlying procedural semantics for data mining query languages. The `2W Model` follows from the `3W Model` [64, 26].

Knowledge discovery is a multi-step process, that involves data preprocessing, different pattern mining stages and pattern postprocessing. In the `2W Model` the essence of a knowledge discovery process is summarized as the interaction between two neatly divided worlds: the *data world* and the *model world*. More precisely, as shown in fig. 3.1, data pre-processing and model post-processing are viewed as world-specific operations. Instead, each intermediate pattern-mining step is considered as a suitable interaction relating entities in the two worlds.

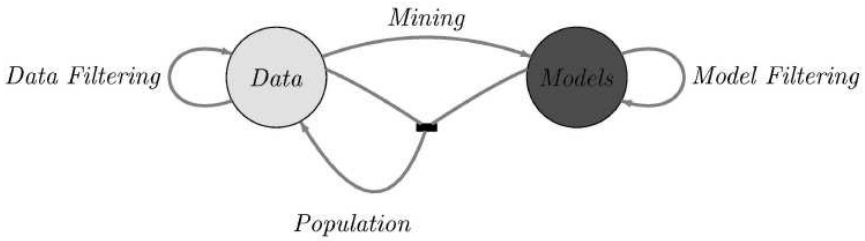


Fig. 3.1. The KDD process in the 2W Model

This allows to formalize any knowledge discovery process as an algebraic expression, that is essentially a composition of operators representing (pseudo-)elementary operations on the two worlds. There are three main kinds of operators for data and models:

- Filtering operators are self-injecting operations: indeed, they take a set of entities as input and produce a new set of entities. Within the figure, the *Data Filtering* and *Model Filtering* arrows denote such operations.
- Mining operators relate data entities to model entities. In practice, such operations correspond to the application of a data mining algorithm to a given data source. The result is a composite object, describing a pattern holding over such data sources.
- Population operators are meant to model a sort of dual operation w.r.t. mining functions. In general, a model is a specification of a set of properties holding in the data. Applying a model to a data source essentially means making such properties explicit in extensional form: e.g., by associating each trajectory in a table with the most likely target class according to the model, or by enumerating the frequent patterns appearing within the trajectory

For the definition of the contours of the two worlds and their operators, one has to concentrate on which entities (i.e. which patterns) are supported in the model world, how data entities relate to model entities, and how constraint solving takes place. The formalization of such aspects strictly depends on the nature of the underlying applicative domain and pursued objectives. The `2W Model` is a general model for the knowledge discovery process within any applicative setting.

3.1 The D-World

The D-World represents the entities to be analyzed, as well as their properties and mutual relationships. Raw data is organized in an object-relational format. The `D-World` can be viewed as a database $D = \{r_1(R_1), \ldots, r_n(R_n)\}$ of meaningful entities. The generic entity $r(R)$ is a relation with schema R. Formally, $R = \{A_1 : Dom(A_1), ..., A_h : Dom(A_h)\}, where A_1, \ldots, A_h$ correspond to descriptive attributes of the data within $r(R)$ and $Dom(A_1), \ldots, Dom(A_h)$ are their respective domains. Relation $r(R)$ is defined as $r(R) \subseteq Dom(A1) \times \ldots \times Dom(Ah)$. Attribute domains can be either primitive or object data types. Primitive types are assigned to simple features of the data and divide into categorical and numerical domains. Instead, object data types abstractly represent complex real-world entities as objects, equipped with application-dependant operations. Hereafter, the specification of relation schema is omitted and it is used the resulting simplified notation to indicate an entity of D. Furthermore, let denote by $t \in r$ a tuple of relation r and, also, exploit notation $t[A_i]$ to indicate the value of tuple t over a schema attribute A_i. So far, the description of the `D-World` is general enough to be employed within any applicative setting. Since this thesis aims at dealing with movement data, hereafter the `D-World` is assumed to be a repository of movement data. From this point of view, relation schemas involve object data types, modeling the addressed moving entities (such as points and regions).

Example 3.1. To elucidate, the reference relation *Trajectories* is introduced, that shall be used throughout the thesis to describe pedestrian and/or vehicle routes. Its schema

attribute comprises *ID* of type integer, `Type` which takes on the categorical values (i.e. vehicle and pedestrian), and *Trajectory* that is of an object data type, named *Moving_Point* [86]. The latter object actually models the notion of trajectory. More precisely, given a tuple $t \in Trajectories$, $t[Trajectory]$ represents a sequence of object locations evolving over time and, also, provides a set of basic operations, for manipulating route data as well as answering topological and distance queries. □

3.1.1 D-World Operators

Data in the `D-World` is manipulated via the usual (unary and binary) operators of traditional relational algebra, namely ρ, σ, π, $\cup$, $\cap$,
and $\times$. Also, aggregation functions (such as `SUM`, `COUNT`, `AVERAGE`, `MIN` and `MAX`) are allowed to operate on collections of domain elements. Notably, aggregates are not relational algebra operators. In principle, they are used as parameters of some suitable aggregate formation operator. In this formalization, express queries involving aggregates are expressed by means of suitably extended projection operators, in the spirit of the idea in [91], that allow the incorporation of aggregation functions into the algebra. Algebraic operators can be used to model suitable data preparation/ manipulation tasks.

Example 3.2. The composite operator below

$$\pi_{trajectory}\,(\sigma_{Type=''vehicle''}(Trajectories))$$

represents a trivial reduction of data size and dimensionality. □

More complex preparation/manipulation tasks can be expressed by incorporating the basic operations of the (domain-specific) object-relational entities in the corresponding algebraic formulation. This is shown below.

Example 3.3. A basic operation of the *Moving_Point* data type is *intersects*, which queries whether two trajectories encounter each other. Such a functionality can be exploited to filter from *Trajectories* and count all those vehicle routes that encounter, somewhere and at any given point in time, the route followed by a reference moving point (i.e. with a specified identifier). To this purpose, by means of the expression

$$T = \rho_{Route \leftarrow Trajectory}(\sigma_{ID=3}(Trajectories))$$

one obtains a new answer relation T consisting of the route followed by the moving point with $ID = 3$. Here, for convenience, the *Trajectory* attribute of T is renamed as *Route*. T can now be joined with the whole content of *Trajectories* to find and count the desired routes, i.e. those paths that intersect the one in T. This can be expressed as

$$\pi_{count(Trajectory)}(\sigma_{Trajectory.intersects(Route)}(Trajectories \times T))$$

where $\pi_{count(Trajectory)}(\cdot)$ is an extended projection operator [91] that returns the size of the column *Trajectory* if it appears in the input relation, 0 otherwise. Notice that, the aforesaid extended projection operator can come in two flavors, depending on whether or not the *Trajectory* column is viewed as a bag.

3.2 The M-World

Movement patterns concerning data entities, their properties and relationships are modeled as suitable decision regions in the model world, which provides an elegant and expressive framework for both exploratory analysis and reasoning. The `M-World` can be represented as a collection P of patterns, unveiled at the different stages of the knowledge discovery process. Each pattern p is associated with an object-relational schema R and represents a (possibly infinite) relation r over R. Intuitively, p represents a decision region over the schema R, so that a decidable operator $\vdash$ can be devised for bounding such a region.

Definition 3.4. *A pattern p is any (possibly infinite) set $\{t_1, \ldots, t_n\}$ of tuples in D such that, for each $r \in D$ and $t \in r$, the assertion $t \in p$ is decidable. The property $t \in p$ is denoted as $p \vdash t$.* □

Different types of movement patterns can be defined to populate the `M-World`, on the basis of the decision regions of interest. Hence, various definitions are possible for the $\vdash$ operator. To elaborate on the latter aspect, the semantics of the operator for some models of fig. 3.2 is exemplified in the next. In particular, the thesis focuses on the identification of moving objects within the *Trajectories* relation of Example 3.1, since this is generally a major target of investigation in movement data analysis.

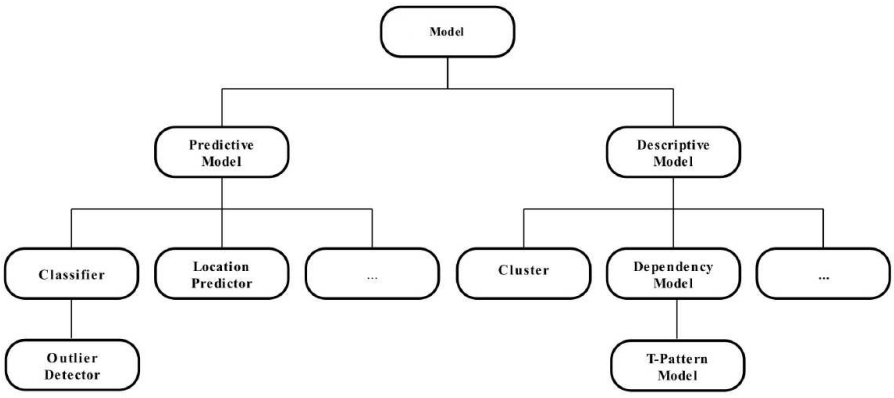


Fig. 3.2. A taxonomy of abstract data types in the M-World

Example 3.5. **SINGLE-CLASS CLASSIFIER**. Also, let the pattern $p^{(pedestrian)}$ be of the form $speed \leq 10Km/h$, where $speed$ is the instant speed of the moving object at hand. Informally, $p^{(pedestrian)}$ represents all those tuples exhibiting instant speed less than $10Km/h$, which, according to the common sense, can be labeled as "pedestrian". Each moving object $t \in Trajectories$ can be equivalently represented by its explicit route $(l_1, ts_1) \overset{speed_1}{\rightarrow} \ldots \overset{speed_{n-1}}{\rightarrow} (l_n, ts_n)$, i.e. a time-ordered

sequence of locations l_i with respective time stamps ts_i, where $speed_i$ is the instant speed of the transition between locations l_{i-1} and l_i. A generic moving object t is recognized by $p^{(pedestrian)}$, i.e., $p^{(pedestrian)} \vdash t$, if $speed_i \leq 10Km/h$ for all $i \in \{1, \ldots, n-1\}$. □

Example 3.6. **T-Pattern** Let p be a temporal annotated pattern [55] of the form $r_1 \overset{tc_1}{\rightarrow} \ldots \overset{tc_{n-1}}{\rightarrow} r_n)$ (where r_i is a spatial region and tc_i is a time constraint of the form $t_i^{min} \leq t \leq t_i^{max}$. A moving object $t \in Trajectories$ is recognized by p, i.e., $p \vdash t$, if t traverses all regions r_i in sequence, and the traversal time between r_i and r_{i+1} is within the time constraint tc_i. □

Starting from the basic patterns in P, it is possible to define composite patterns as suitable combinations of patterns from possibly different prototypes. The individual instance of a prototype enumerates only the raw data that results into a certain outcome when related to the pattern. However, in general, raw data can originate multiple outcomes. The notion of composite pattern is essentially an abstraction that allows to succinctly query multiple patterns of different prototypes for the recognized raw spatio-temporal data.

Definition 3.7. ***COMPOSITE PATTERN***. *A composite pattern p is defined as follows:*

- *$\mathsf{p} \equiv p$ with $p \in P$ is a composite pattern and the $\vdash$ operator is straightforwardly extended to p, since $\mathsf{p} \vdash t$ coincides with $p \vdash t$ in this basic definition;*
- *the disjunction $\mathsf{p}_1 \vee \mathsf{p}_2$ of two composite patterns p_1 and p_2 is still a composite pattern and the $\vdash$ operator is defined as $\mathsf{p}_1 \vee \mathsf{p}_2 \vdash t$ if and only if either $\mathsf{p}_1 \vdash t$ or $\mathsf{p}_2 \vdash t$;*
- *the conjunction $\mathsf{p}_1 \wedge \mathsf{p}_2$ of two composite patterns p_1 and p_2 is still a composite pattern and the $\vdash$ operator is defined as $\mathsf{p}_1 \wedge \mathsf{p}_2 \vdash t$ if and only if both $\mathsf{p}_1 \vdash t$ and $\mathsf{p}_2 \vdash t$;*
- *the negation $\neg\mathsf{p}$ of a composite pattern is still a composite pattern, where $\neg\mathsf{p} \vdash t$ if it is not the case that $\mathsf{p} \vdash t$.* □

In practice, a composite pattern is an abstraction for representing a new decision region that follows from the ones associated to the individual patterns.

Example 3.8. Let $p^{(pedestrian)}$ be the predictor defined in Example 3.5 , and p_1, p_2 two further temporal annotated patterns introduced in Example 3.6. The composite pattern $\mathsf{p} = p^{(pedestrian)} \wedge (p_1 \vee p_2)$ is a pattern characterizing all pedestrians traveling with an instant speed less than $10Km/h$ and traversing either p_1 or p_2. □

Definition 3.9. *Let p be a composite pattern. The structural operators $\lhd(\cdot)$, $\rhd(\cdot)$ and $\odot(\cdot)$ are defined as follows:*

- *The left and right operators on p, namely $\lhd(\mathsf{p})$ and $\rhd(\mathsf{p})$, return, respectively, the left and right component of p. Formally, if $\mathsf{p} = \mathsf{p}_1 \wedge \mathsf{p}_2$ or $\mathsf{p} = \mathsf{p}_1 \vee \mathsf{p}_2$ then $\lhd(\mathsf{p}) = \mathsf{p}_1$ and $\rhd(\mathsf{p}) = \mathsf{p}_2$; otherwise $\lhd(\mathsf{p}) = \rhd(\mathsf{p}) = \mathsf{p}$.*

- *The inside operator $\odot(\mathsf{p})$ on a composite pattern p extracts the positive component of p. Precisely, if $\mathsf{p} = \neg\mathsf{p}_1$, then $\odot(\mathsf{p}) = \mathsf{p}_1$; otherwise $\odot(\mathsf{p}) = \mathsf{p}$.* □

With an abuse of notation, in the following P is assumed to be extended to contain both singleton and composite patterns. The latter can be further classified into local or global patterns, depending on whether such patterns are capable of recognizing each tuple in the associated domain.

Definition 3.10. ***GLOBAL/LOCAL PATTERN***. *Let p be a (composite) pattern and R an object-relational schema. p is said a global pattern w.r.t. R if and only if, for each relation r over R and each $t \in r$, it holds that $p \vdash t$. Otherwise, p is said a local pattern.* □

Example 3.11. Let us consider the pattern $p^{(vehicle)}$, which recognizes tuples representing trajectories exhibiting an instant speed greater than $10Km/h$. The pattern $p^{(vehicle)} \vee p^{(pedestrian)}$ is a global pattern for $Trajectories$, as it recognizes each tuple in the relation.

As shown in fig. 3.2, a prototype **P** can be associated to a given pattern p, depending on whether the decision region characterizes the pattern as a predictive or descriptive spatio-temporal model. In practice, a prototype **P** enumerates a subset of P, such that some attributes $\varphi_1^{(\mathbf{P})}, \ldots, \varphi_n^{(\mathbf{P})}$ n can be associated to each component pattern p in **P**. Formally, a prototype associates each pattern with a relational schema, where some specific pattern properties can be specified.

Definition 3.12. *Let R be a relational schema. A pattern schema **P** on R is defined as $\mathbf{P} = \{S \times r | S \subseteq P, r \in R\}$. $P \in \mathbf{P}$ is defined a* `pattern instance`.

Example 3.13. Assume that $O = \{pedestrian, vehicle\}$ is a set of class labels denoting two alternative types of moving-object routes. Then, $\texttt{Classifier}^O$ is any subset of $P \times O$. In practice, each pair $\langle p, c\rangle \in \texttt{Classifier}^O$ represents a decision region denoted by p, whose tuples are associated to class c.

As a further abuse, $p \in P$ and $\langle p, \varphi_1, \ldots, \varphi_n\rangle \in P$ will be used interchangeably, where the former notation is a shorthand for the latter.

Definition 3.14. ***INTRA-WORLD OPERATORS***. *The usual definitions of the σ and π (relational) operators can be extended, and can be introduced the three (structural) operators, $\lhd(\cdot)$, $\rhd(\cdot)$ and $\odot(\cdot)$, over a pattern schema **P** as follows. Let P be a pattern instance of **P**. Then*

- $\sigma_E(P) = \{\langle p, t\rangle \in P | E(t)\}$ *where E is any boolean expression over the schema R;*
- $\pi_X(P) = \{\langle p, \pi_X(t)\rangle \mid \langle p, t\rangle \in 2P\}$ *where $X \subseteq R$.*
- $\gamma(P) = \{\langle \gamma(p), t\rangle \mid \langle p, t\rangle \in 2P\}$, *where $\gamma(\cdot)$ corresponds to either $\lhd(\cdot)$, $\rhd(\cdot)$ or $\odot(\cdot)$.* □

Example 3.15. Assume that prototype `T - pattern` is a pattern schema consisting of three attributes, namely Pattern, Length and Type, that associates each temporal annotated pattern in P with two characteristic features, respectively the length of the pattern in terms of consecutive spatial regions and the kind of routes that traverse it. Given the below instance P of **P**

Pattern	Length	Type
p_1	5	vehicle
p_2	2	pedestrian
p_3	3	mixed
p_4	4	vehicle
p_5	2	pedestrian
p_6	3	mixed
$p_2 \wedge p_5$	2	pedestrian
$p_3 \vee p_6$	3	mixed

the intra-world operators $\sigma_E(P)$ and $\pi_X(P)$ can be used to suitably manipulate P. For instance,

$$\sigma_{Length \geq 4 \wedge Type = ''vehicle''}(P)$$

selects those temporal-annotated patterns in P, i.e., p_1 and p_4, that consist of at least 4 spatial regions and are traversed only by vehicles. In the selected fragment of P, the $Type$ feature assumes a uniform value, thus becoming uninteresting. It can be filtered by means of the projection operator

$$\pi_{Pattern,Length}(\sigma_{Length \geq 4 \wedge Type = ''vehicle''}(P))$$

Finally, the generic structural operator $\gamma(P)$ can used to decompose the composite patterns in P into their constituents. Notice that P contains the composite patterns $p_2 \vee p_5$ and $p_3 \wedge p_6$. In general, depending on the required constituents, $\gamma(P)$ can be instantiated as either $\lhd(\cdot)$, $\rhd(\cdot)$ or $\odot(\cdot)$. □

Patterns are originated into the `M-World` from the raw data in the `D-World` via a mining operator. Such patterns are in turn used to inject new raw data into the `D-World`. Mining and populate operators formalize suitable interactions between raw data within the `D-World` and patterns in the `M-World`. Such interactions are the basic building-blocks in the definition of a knowledge discovery workflow.

3.2.1 Mining and Population Operators

The population of the `M-World` starting from the raw data in the `D-World` is performed through the mining operator k.

Definition 3.16. ***MINING OPERATOR****. The inter-world mining operator is defined as* $k : D^k \rightarrow !2^P$*.* k *represents a generic mining scheme, that receives a certain number of input relations and instantiates in the* `M-World` *an instance (i.e., multiple patterns) of a prototype (that is a pattern schema).* □

Example 3.17. Assume that $D = \{Trajectories\}$, where $Trajectories$ is the reference relation of Example 3.1. If k represents the T-pattern mining scheme in [55], $k(Trajectories)$ results into an instance $T-pattern$ of a prototype **T - pattern**. □

Once accomplished the forward population of the `M-World` with the required patterns, these can be employed in the opposite direction, i.e. to backwardly populate the `D-World` with further data. Interestingly, this does not involve the explicit representation of further (composite) objects as in the `E-World` of the `3W Model`. More simply, the raw data that falls within the decision region of a certain pattern is accumulated in the `D-World` as new data. The inter-world population operator $\bowtie$: $P \times D \rightarrow D$, is the basic step of the population process.

Definition 3.18. ***BASIC POPULATION OPERATOR****. Let* p *be a pattern in* P *and* r *a relation in* D *over an object-relational schema* R*. The basic population operator* $p \bowtie r$ *yields a new relation including each tuple* $t \in r$ *within the decision region of* p*. Formally,*

$$p \bowtie r \triangleq \{t \in r | p \vdash t\}$$

Clearly, the resulting relation $p \bowtie r$ *is still an instance of* R*.* □

Example 3.19. Let D and P be respectively the `D-World` and `M-World` of Example 3.17. If p is a temporal annotated pattern over the **T - pattern** prototype, the expression

$$Trajectories_1 \triangleq p \bowtie Trajectories$$

results into $D_1 = Trajectories, Trajectories_1$, where $Trajectories_1$ is a new relation including those moving entities from $Trajectories$, whose routes traverse the temporal annotated pattern p. In practice, D_1 represents the population of the original `D-World` with the new raw data $Trajectories_1$, whose schema is identical to the one of the $Trajectories$ relation.

The population operator can be straightforwardly generalized to deal with a composite pattern.

Definition 3.20. ***EXTENDED POPULATION OPERATOR****. Let* $\boldsymbol{p}$ *be a composite pattern and* r *a relation in* D *over an object-relational schema* R*. By abuse of notation, the extended population operator* $\bowtie$: $2^P \rightarrow D$ *is defined as follows*

$$p \bowtie r \triangleq t \in r | p \vdash r$$

Relation $p \bowtie r$ *is again an instance of* R*.* □

The above expression yields an enumeration of all the tuples in r that fall within the composite decision region **p**.

Example 3.21. Again, let D and P be respectively the `D-World` and `M-World` of Example 3.17. Moreover, assume that $\mathbf{p} = p_i \vee p_j$ is a composite pattern such that both p_i and p_j are two temporal annotated patterns from the Trajectories relation. The below expression

$$Trajectories_2 = p \bowtie Trajectories$$

populates the original `D-World` with a further $Trajectories_2$ relation, that enumerates all moving objects of $Trajectories$ that traverse either p_i or p_j. □

Besides the population operator, it is also defined a pattern identification operator, that is dual with respect to the former.

Definition 3.22. ***PATTERN IDENTIFICATION OPERATOR****. Assume that* ***P*** *is a prototype and* P *an instance of* ***P****. Let* S *be any subset of* P*. The pattern identification operator* $\diamond : 2^P \times D \rightarrow 2^P$ *is defined as follows:*

$$S \diamond r = \{p \in S | \exists t \in r : t \in p \bowtie r\}$$

□

In practice, the pattern identification operator queries a homogeneous pattern collection S for those models that recognize certain raw data.

Example 3.23. One may ask which are the temporal annotated patterns from pedestrian routes, that are also traversed by vehicles. To this purpose, let

$$Trajectories_1 = \pi_{Type=''pedestrian''}(Trajectories)$$

and

$$Trajectories_2 = \pi_{Type=''vehicle''}(Trajectories)$$

be the two required partitions of the $Trajectories$ relation. If $T - pattern$ is an instance of the **T - pattern** prototype, that consists of the temporal annotated patterns in $Trajectories_1$, the foregoing query can be expressed via the following expression

$$T - pattern \diamond Trajectories_2$$

□

As a final remark, this work emphasizes that pattern-identification and population operators can be suitably combined to express complex analytical queries.

Example 3.24. With respect to the setting of Example 3.23, one may further ask which are the individual vehicle trajectories that traverse the pedestrian T-patterns. This query can be expressed as

$$(T - pattern \diamond Trajectories_2) \bowtie Trajectory_2$$

□

3.2.2 Discussion

The 2W Model introduces several meaningful differences w.r.t. the 3W Model. Firstly, entities in the M-World can represent any required patterns, even if with a mathematically complex structure, whereas I-World models correspond to simple regions, expressible via linear inequalities on the data attributes. Secondly, in the 2W Model, k is not predefined and acts as a template to extract a model from the raw data. Thirdly, the composite E-World representation is avoided and the objects of the 3W Model that reside within such a world are directly mapped, in the 2W Model, to simple raw data in the D-World. In particular, by the definition of the population operator, the application of any model to the data in a relation of the D-World always produces a further relation within the D-World. This ensures that mining results can be progressively analyzed on a par with raw data via further manipulations, as exemplified next.

Example 3.25. **KDD WORKFLOWS IN THE 2W MODEL**.

Consider the case where one wishes to uncover the groups objects that move close to each other within a certain temporal annotated pattern [55]. In such a case, temporal annotated patterns are first extracted into the M-World via a specific mining operator k from the $Trajectories$ relation. This results into an instance $k(Trajectories)$, that groups all the unveiled patterns. The latter are then treated on a par with raw data, to the purpose of identifying the trajectories inside the required pattern, which is accomplished by means of the inter-world population operator. For instance, Moving clusters [84] are then discovered in the required pattern, by applying a second mining operator k_1 to the newly obtained raw data. In the 2W Model, the algebraic formulation of the aforesaid knowledge discovery workflow is

$$k_1 \left(p \bowtie Trajectories \right)$$

where $p \in k(Trajectories)$ is the pattern to investigate for moving clusters, that can be chosen by means of the structural operators $\lhd(k(Trajectories))$ and $\rhd(k(Trajectories))$. The above expression reveals the fundamental role of the population operator in the definition of a knowledge discovery workflow. Indeed, the operator enables the progressive and seamless discovery of further patterns in the raw data resulting at the end of a previous analytical process. □

Finally, the D-World operators contribute to the expressiveness of the 2W Model framework, by playing a twofold role. On the one hand, such operators can be used to represent preprocessing tasks, e.g. the reduction in size and/or dimensionality of the available data. On the other hand, they are useful for postprocessing purposes, such as in the act of filtering interesting patterns.

3.3 IMPLEMENTING THE 2W MODEL

The 2W Model is a natural foundation for the development of domain-specific data mining query languages. Within a specific applicative domain, this mainly involves

the definition of appropriate mining and population operators, as well as the specification of the basic object-relational entities. In this thesis, a data mining query language is proposed. It is designed to support knowledge discovery from movement data as an actual multi-step knowledge discovery process. The intuition consists in starting from the conventional SQL language, that provides basic mechanisms for interactively querying and manipulating the entities within the `D-World` (i.e. both original raw data and the outcome of population operators). These are extended in two major respects. Firstly, the introduction of a pattern definition statement, i.e. *CREATE MODEL*, for the specification of the required movement models, with which to populate the `M-World`. Secondly, the capability of supporting generic population operators, which ultimately allows the application of models in the `M-World` to raw data within the `D-World`. For this reason, the traditional join semantics inherent in the *SELECT-FROM-WHERE* statement is revised, so that raw data and unveiled patterns can be uniformly manipulated and joined for further analysis.

3.3.1 Model Definition

The *CREATE MODEL* statement implements the mining operator k of the `2W Model` and builds a particular model in the `M-World`.

Definition 3.26. ***MODEL DEFINITION STATEMENT***. *The syntax of the* CREATE MODEL *statement is reported below.*

```
CREATE MODEL <model_name> AS MINE <mining_algorithm>
FROM <<table>>
WHERE <mining_algorithm>.param_1 = val_1 ⊗ ... ⊗
      <mining_algorithm>.param_n = val_n
```
□

The above statement specifies a pattern-discovery task, via a call to some corresponding mining algorithm. In terms of the `2W Model` algebra, the definition creates a model object in the `M-World` named `<model_name>`, according to the procedural semantics $k(table)$, where the effect of the mining operator k is the application of the `<mining_algorithm>` to table. In this respect, an important difference with respect to the traditional SQL `CREATE` statement is that the latter guarantees closure by returning a table, so that further SQL statements can be issued over it. Instead, the `CREATE MODEL` statement results into a (singleton or composite) object, that is an instance of some corresponding abstract data type. Closure is enforced by the possibility of manipulating both raw data and pattern objects, as previously said.

The `CREATE MODEL` statement enables the development of data mining query languages, that meet users requirements in any given applicative setting. This chapter focuses on data analysis and assumes that, hereafter, `<mining_algorithm>` denotes any methods for discovering patterns. `<<table>>` denotes the primary data of the `D-World`, from which `<model_name>` is extracted, that can be in the form of either a materialized database table, a view, or a query. The `WHERE` clause allows

to properly specify the input parameters of the invoked <mining_algorithm>, that involve algorithm-specific parameters, search biases and thresholds for interestingness measures. Combinations of logical conditions on input parameters are expressed via any connectors $\otimes$ from traditional SQL grammar.

Notice that, depending on the <mining_algorithm>, background knowledge within some further table of the D-World can be directly taken into account to assist pattern discovery, by either specifying further tables in the FROM clause or exploiting the relational organization of the trajectory data. This is useful for several reasons, such as either enriching the data at hand, deriving good initial hypotheses with which to start the search for patterns, defining preference biases that prune the pattern search space, or providing a reference for the interpretation of the discovered patterns. Furthermore, since it is often difficult to define adequate statistical measures for subjective concepts like novelty, usefulness, and understandability, background knowledge can be also helpful in capturing such concepts more accurately.

Example 3.27. The following CREATE MODEL statement exemplifies the definition of a T-pattern mining task, which requires the availability in the D-World of the Trajectories table introduced in the previously section.

```
CREATE MODEL T_Patterns AS MINE Dynamic_TPattern_Mining
FROM Trajectories
WHERE Dynamic_TPattern_Mining.density = δ AND
      Dynamic_TPattern_Mining.snr = ϵ AND
      Dynamic_TPattern_Mining.tt = τ
```

The **T_Patterns** composite object, as easy to see, is instantiated by applying the Dynamic_TPattern_Mining algorithm to the basic Trajectory data.

Algorithm-specific parameters appear in the above syntax as features of the **T_Patterns** object and are suitably set in the WHERE clause. Here, the minimum density threshold (density), spatial neighborhood radius (snr) and temporal threshold (tt) are set to suitable values, respectively represented by δ, ϵ and τ . Further details on the T-pattern mining algorithm and the mentioned input parameters are provided in [55]. □

3.3.2 Data and Model Manipulation

The SELECT-FROM-WHERE statement can be used to accomplish several different tasks of the knowledge discovery process. The procedural semantics of the individual statement is defined as some suitable combinations of 2W Model operators. In the following, the thesis elucidates the SELECT-FROM-WHERE statement in the manipulation of raw data as well as in the definition and further analysis of patterns.

Raw Data Manipulation. Data manipulation and querying represents the simplest exploitation of the statement.

Example 3.28. Query $\mathbf{Q}_1$ defines a simple preprocessing of trajectory data, before it is used in any subsequent analytical task.

```
SELECT Trajectories.id, Trajectories.Trajectory
FROM Trajectories
WHERE Trajectories.type="vehicle".
```

Clearly, $\mathbf{Q}_1$ filters vehicle trajectories and projects them on attributes ID and $Trajectory$, deemed relevant for subsequent pattern discovery.

In terms of D-World operators, the procedural semantics of the above statement is

$$\pi_{ID,Trajectory}(\sigma_{type=''vehicle''}(Trajectories))$$

□

Hybrid Manipulation of Raw Data and Models. The manipulation of pattern objects in the M-World enables more advanced uses of the SELECT-FROM-WHERE statement. By suitably joining data within the D-World and models in the M-World, it is possible to find out the raw data that meets a particular constraint on some feature of a certain pattern.

Example 3.29. Consider the below query $\mathbf{Q}_2$, that asks for all trajectories within the database that intersect a T-pattern.

```
CREATE MODEL T_Patterns AS MINE Dynamic_TPattern_Mining
FROM Trajectories
WHERE Dynamic_TPattern_Mining.density = δ AND
      Dynamic_TPattern_Mining.snr = ϵ AND
      Dynamic_TPattern_Mining.tt = τ;
SELECT Trajectories.id
FROM Trajectories, T_Patterns
WHERE T_Patterns.contains(Trajectories.id)
```

Procedurally, the above statement involves two steps, i.e. the identification of the required T-patterns and the hybrid manipulation of the latter with raw trajectory data. In algebra:

$$\pi_{ID}(\sigma_{k(Trajectories).contains(Trajectories.ID)}(Trajectories))$$

□

Progressive Mining Tasks. The possibility of specifying suitable population operators allows multiple stages of analysis for the mining results.

Example 3.30. For instance, to diagnose the causes of mobility congestion, the user may wish to gain an insight into the collective movement of the vehicles in the answer to the last $\mathbf{Q}_2$ (example 3.29. In particular, she may focus on the ones returned by $\mathbf{Q}_2$.

```
CREATE MODEL T_Patterns AS MINE Dynamic_TPattern_Mining
FROM Trajectories
```

```
WHERE Dynamic_TPattern_Mining.density = δ AND
      Dynamic_TPattern_Mining.snr = ϵ AND
      Dynamic_TPattern_Mining.tt = τ;
CREATE MODEL Moving_Clusters AS MINE MC
FROM (SELECT *
 FROM Trajectories, T_Patterns
 WHERE T_Patterns.contains(Trajectories.id))
```

This pattern discovery task involves the execution of the moving cluster algorithm MC, that is one of the schemes in [84]. In algebra:

$$k_{MC}(\sigma_{k_{TPattern}(Trajectories).contains(Trajectories.ID)}(Trajectories))$$

□

4

Classification in hostile environments

Various types of classifiers have been proposed in the literature, that meet several different requirements in a wealth of distinct applicative settings, such as decision trees, rule-based classifiers, neural networks, bayesian classifiers, support vector machines and statistical classifiers [37]. In particular, rule learning is a method for inducing minimal rule-based concept descriptions, that can be used for classification. Rule-based classifiers are a mainstay of research in machine learning, because of various desirable properties such as, e.g., their expressiveness and intelligibility to humans as well as their efficiency and effectiveness in classification. Such classifiers have been empirically shown to be effective in processing (sparse) high-dimensional training data with categorical attributes [99] and are comparable in performance with other classification methods in several applicative domains [81]. Unfortunately, like most classification models, rule-based classifiers exhibit a poor classification performance in imprecise (multi-class) learning environments, which are challenging domains wherein cases and classes of primary interest for the learning task are rare. Besides, minority and majority classes can be hardly separable and the cost of misclassifying a case of a minority class as belonging to a predominant class is much higher than the cost of the dual error. Also, training data may be corrupted by noise, which further obstacles the identification of rarities.

Imprecise domains are often encountered in practical applications. Examples include fraud detection [50, 87], intrusion detection, manufacturing line monitoring [90], risk management, telecommunications management [48], medical diagnosis [29], text classification [103] and oil-spill detection in satellite images [72]. The peculiarities of such settings pose several challenging issues to traditional algorithms for learning rule-based classifiers, that essentially make the resulting models low sensitive to rarities.

Rarity is clearly the major obstacle. Rare classes corresponds to the well known *class imbalance* issue [60, 61], i.e. an evenly distribution of classes, such that majority classes overwhelm minority ones. Instead, rare cases are very small portions of the training data, that can be viewed as exceptional sub-concepts seldom occurring within predominant or rare classes. As it is pointed out in [103], rarity actually

prevents conventional algorithms for rule induction from finding and reliably generalizing the regularities within infrequent classes and exceptional cases.

Indeed, class imbalance generally leads to classification models tending to exhibit a high specificity (i.e. capability at recognizing majority classes), coupled with a low sensitivity (i.e. capability at recognizing minority classes).

Rare cases, instead, tend to materialize within the learnt classification models as *small disjuncts* [59], i.e. rules covering very few training cases [102]. Small disjuncts were empirically shown to be a major cause of poor predictive performance [104] and cannot be easily removed without adversely affecting the remaining classification rules.

The foregoing effects of rarity on rule learning are exacerbated by noise. On one hand, the latter may further skew class imbalance. On the other hand, it may also appear to the learner as nearly indistinguishable from rare cases.

Besides rarity and noise, different misclassification costs as well as low class separability also have a role in making conventional rule learning schemes inadequate within imprecise domains.

In the last decade, classification based on frequent patterns, also known as *associative classification*, has emerged as a powerful enhancement of conventional rule learning, based on converging research efforts in machine learning and data mining. Precisely, the basic intuition behind associative classification is to substitute conventional rule induction with an association-rule mining step. The resulting classification models, said associative classifiers, consist of class association rules, i.e. suitable association rules meeting some specific constraints. The antecedents of these rules are co-occurrent attribute values, that frequently appear across the training data, while their consequents are suitable values of the target class attribute. Associative classification is in principle better suited for unsupervised predictive modeling within imprecise learning settings: it retains the advantages of traditional rule learning and also tends to achieve a better performance for several reasons. Foremost, while rule induction dilutes rarity and produces overly biased rules, associative classification yields rules with an appropriate degree of generality/specificity, that summarize the whole training data. Also, the individual class association rules catch strong, i.e. frequently occurring, associations between (combinations of) data items and class labels. This is a robust mechanism with which to handle noise in data. Additionally, such associations reflect the inherent semantics of the training data and, thus, have a high discriminative power. The resulting associative classifiers are statistically significant and are hence deemed to properly generalize on unseen data [31]. Furthermore, frequent patterns represent a more expressive feature space, where the original training data is likelier to be linearly separable.

One limitation for associative classification, that is particularly relevant in imprecise learning settings, is borrowed from traditional rule learning. More specifically, the decision regions induced by a rule-based classifier and the true distribution of the classes in the space of data do not match. Indeed, classes form regions with irregular and interleaved shapes, whereas the induced decision regions are neatly separated by boundaries parallel to the features of the data space. As a consequence, those cases falling within and close to the boundary of a decision region may be misleadingly

predicted as belonging to the class associated with that decision region, even if the true class membership in the surroundings of the boundary is different. This is problematic in imprecise applicative domains, wherein the separability between classes is low, since these form true overlapping (or embedded) regions. In such cases, indeed, the true regions formed by rare classes may be (partly or completely) overlapped by the decision regions associated to the predominant classes and, thus, the recognition of previously unseen cases of the rare classes becomes a major concern.

in this thesis, two approaches are proposed that look at associative classification from two dual perspectives.

From the *global-to-local* point of view, associative rule learning yields a global (high level) classification model, whose class assignments are then refined locally to the individual classifier rules. In this regard, one approach essentially builds a hierarchical classification framework, that combines associative rule learning and probabilistic smoothing [34]. The underlying idea is to use the individual rules of an associative classifier to divide the original training data into as many segments, wherein it is likely that some globally rare cases/classes become less rare. The resulting segments are then used to build as many local probabilistic generative models, that better catch the forms of rarity local to their segments. These probabilistic generative models are then used to refine the predictions from the classifier rules. Two distinct schemes are proposed for tightly integrating associative classification and probabilistic smoothing, that decide the class of an unlabeled case by considering multiple class association rules as well as their corresponding probabilistic generative models.

From the *local-to-global* point of view, instead, associative rule learning provides local data features, that determine global assignments of class probabilities. Therein, in the second approach, the individual rules of an associative classifier are used as features. Given a data case, classification takes into account the predictions from all those rules that are local to the case (i.e. that cover the case). The relevance of a rule with respect to its targeted class determines the weight of the corresponding feature on the discrimination of that particular class. This enables the recognition of minority classes via those classification rules, that are highly representative of such classes (i.e. whose antecedent reflects item co-occurrences that are inherently characteristic of such classes). The maximum entropy framework is used to elegantly and seamlessly integrate associative classification with discriminative learning.

4.1 Preliminaries

Here is an introduction of the notation used throughout the manuscript and some basic notions. Let $\mathcal{D}$ be a relation storing the labeled training cases. Also, let the schema of $\mathcal{D}$ be a set $\mathcal{A} = \{A_1 : Dom(A_1), \ldots, A_n : Dom(A_n), L : \mathcal{L}\}$ of descriptive attributes. In particular, features $A_1, \ldots, A_n$ are defined over as many categorical or numeric domains, whereas the target class attribute L is a categorical feature. The generic labeled training case $t \in \mathcal{D}$ is a structured tuple, i.e. $t \in Dom(A_1) \times \ldots Dom(A_n) \times Dom(L)$. Each tuple t can also be equivalently

represented in a transactional form. Therein, assume that $\mathcal{M} = \{i_1, \ldots, i_m\}$ is a finite set of items denoting relationships between any attribute of $\mathcal{A}$ but L and a corresponding value. Precisely, the generic item i has the form $A = v$ where $A \in \mathcal{A} - L$. In the proposed formulation, $v \in Dom(A)$ if A is a categorical attribute. Otherwise, if A is a numeric attribute, v stands for the label of some suitable range of numeric values, whose center is closest in Euclidean distance to the original value of A (more details on the discretization of numeric attributes are provided within section 4.4).

Any unlabeled case I defined over $\mathcal{A}$ can be represented as some suitable subset of items in $\mathcal{M}$. Notice that there must be exactly one item in I for each attribute of the relational schema $\mathcal{A}$. This is concisely expressed by means of the $\subset$ operator, whose meaning is revised as follows $I \subset \mathcal{M} = \{i_{j_1}, \ldots, i_{j_n} |\, i_{j_h} \in \mathcal{M} \wedge attr(i_{j_h}) = A_h,\ \forall h = 1, \ldots, n\}$, where notation $attr(\cdot)$ indicates the attribute referred to by the individual items of I. Viewed from this perspective, a labeled case over $\mathcal{A}$ simply becomes an unlabeled case supplemented with its corresponding class label. Let $\mathcal{L}$ be a finite domain of class labels, the original dataset $\mathcal{D}$ can thus be equivalently expressed in transactional form over $\mathcal{M}$ as a collection $\mathcal{D} = \{t_1, \ldots, t_n\}$, in which the generic labeled case is represented as $t = I \cup \{class(t)\}$, where $I \subset \mathcal{M}$ and $class(t) \in \mathcal{L}$ denotes the class label of t.

A number of definitions recalled in throughout the manuscript are reported next.

Definition 4.1 (Class association rule). *A class association rule (CAR) $r : I \rightarrow c$ is a pattern whose implicative catches the association (i.e. the co-occurrence) in $\mathcal{D}$ of some subset of items $I \subset \mathcal{M}$ with a class label c from $\mathcal{L}$.* □

The notions of support, coverage and confidence are typically employed to define the interestingness of a rule r.

Definition 4.2 (Support of a class association rule). *Let $\mathcal{D}$ be a set of training cases. A training case $t \in \mathcal{D}$ is said to* support *rule $r : I \rightarrow c$ if it holds that $(I \cup \{c\}) \subseteq t$. The support count of r, denoted by $\sigma(r)$, is the overall number of training cases that support r, i.e., $\sigma(r) = |\{t \in \mathcal{D} | (I \cup \{c\}) \subseteq t\}|$. The support of r is instead the fraction of training cases supporting r, i.e., $supp(r) = \frac{\sigma(r)}{|\mathcal{D}|}$, where $|\mathcal{D}|$ indicates the cardinality of $\mathcal{D}$.* □

Support is useful to avoid spurious rules. Intuitively, rule antecedents with high support in the individual classes capture the inherent semantics of the underlying data, rather than being artifacts.

Definition 4.3 (Coverage of a class association rule). *Let $\mathcal{D}$ be a set of training cases. Rule $r : I \rightarrow c$ is said to* cover *a training case $t \in \mathcal{D}$ (and, dually, t is said to trigger or fire r) if the condition $I \subseteq (t - \{class(t)\})$ holds. The set of training cases covered by r is denoted by $\mathcal{D}_r = \{t \in \mathcal{D} | I \subseteq (t - \{class(t)\})\}$. Hence, the coverage of r can be defined as the fraction of cases in $\mathcal{D}$ that are covered by r, i.e. $coverage(r) = \frac{|\mathcal{D}_r|}{|\mathcal{D}|}$. Analogously, the aforesaid rule $r : I \rightarrow c$ is said to* cover *an unlabeled training case I' if it holds that $I \subseteq I'$.* □

Definition 4.4 (Confidence of a class association rule). *The confidence of a rule r, denoted by $conf(r)$, is the ratio of support to coverage, i.e. $conf(r) = \frac{supp(r)}{coverage(r)}$.* □

Confidence measures the predictive strength of a CAR.

Although the traditional support and confidence framework allows to effectively discover all the required class association rules, it still produces uninteresting rules when the class distribution is imbalanced. The point is that, in such cases, confidence is not a reliable measure of the interestingness of a rule, since it does not properly take into account the actual implicative strength of the rule, whose antecedent and consequent can be negatively correlated [8, 6]. To overcome such a limitation, it is possible to consider the degree of positive correlation between the antecedent and the consequent a rule.

Definition 4.5 (CAR correlation). *The correlation of a rule $r : I \rightarrow c$, denoted by $corr(r)$, measures the relationship between the antecedent I and the consequent c. Formally, it is defined as $corr(r) = \frac{P(I \cup c)}{P(I)P(c)}$, where $P(I \cup c)$ is the occurrence frequency $supp(I \cup c)$ of $I \cup c$ across a set $\mathcal{D}$ of training cases. Analogously, $P(I)$ and $P(c)$ correspond to the occurrence frequencies of I and c in $\mathcal{D}$. If $corr(r) < 1$, r is negatively correlated. Instead, $corr(r) = 1$ denotes absence of correlation (i.e. I and c co-occur by chance), whereas $corr(r) > 1$ represents positive correlation.* □

In highly imprecise learning settings, a class association rule r is interesting if it is positively correlated and also meets certain minimum requirements on its support and confidence. An associative classifier is a suitable disjunction of propositional `if-then` rules, that can be used for the classification of unlabeled cases.

Definition 4.6 (Associative classifier). *An associative classifier $\mathcal{C}$ approximates the (unknown) discrete-valued case labeling function behind $\mathcal{D}$. The learnt approximation is represented as a disjunction $\mathcal{C} = \{r_1 \vee \ldots \vee r_k\}$ of interesting class association rules extracted from $\mathcal{D}$.* □

An associative classifier $\mathcal{C}$ is used in section 4.2 to globally segment the whole training data, for the purpose of bringing to the surface those originally rare data, that becomes less rare within each resulting segment. Instead, in section 4.3, the CARs of $\mathcal{C}$ are viewed as properties local to each individual data case.

4.2 The *Global-to-Local* Supervised Learning Framework

Here is a discussion on a *global-to-local* approach aimed to learn a hierarchical framework from the training cases $\mathcal{D}$, that consists of two classification levels. At the higher level, an associative classifier is built such that its component CARs meet some requirements on the minimum support and confidence. For each CAR $r \in \mathcal{C}$, the lower level of the framework includes a local probabilistic generative model $P^{(r)}$

that allows to confirm or rectify r in the classification of an unlabeled case. The overall learning process is shown in algorithm. 1. Given a database $\mathcal{D}$ of training cases (defined over a set $\mathcal{M}$ of items and a set $\mathcal{L}$ of class labels), the algorithm begins (at line 1) by extracting a set $\mathcal{R}$ of class association rules from $\mathcal{D}$ via the MINECARS search strategy.

The rule set $\mathcal{R}$ is subsequently sorted (at line 2) according to the total order $\prec$, which is a refinement of the one in [75]. Precisely, given two rules $r_i, r_j \in \mathcal{R}$, r_i precedes r_j, which is denoted by $r_i \prec r_j$, if (*i*) the confidence of r_i is greater than that of r_j, or (*ii*) their confidences are the same, but the support of r_i is greater than that of r_j, or (*iii*) both confidences and supports are the same, but r_i is shorter than r_j.

The learning process proceeds (at line 3) to distil a classifier $\mathcal{C}$ by pruning $\mathcal{R}$, which generally includes a very large number of CARs, that may overfit the training cases. For this purpose, the overfitting avoidance strategy presented in [27] is exploited to reduce the complexity of the discovered CARs, while still improving their error rate. This is essentially accomplished via the removal of individual items and/or whole rules.

The resulting classifier $\mathcal{C}$ may leave some training cases uncovered. Therefore, a default rule $r_d : \emptyset \rightarrow c^*$ is appended to $\mathcal{C}$ (at line 5), such that its antecedent is empty and c^* is the majority class among the uncovered training cases.

As a remark, notice that, due to the total order $\prec$ enforced over $\mathcal{R}$, the associative classifier $\mathcal{C}$ is actually a decision list: each training case is classified by the first CAR in $\mathcal{C}$ that covers it. In other words, the CARs in $\mathcal{C}$ are mutually exclusive, i.e. a training case is covered by at most one rule of the classifier. As a consequence, the generic CAR $r : I \rightarrow c$ hereinafter covers the set of all those training cases that are not covered by any other CAR with higher precedence. More precisely, the definition of the coverage $\mathcal{D}_r$ of CAR r is refined into $\mathcal{D}_r = \{t \in \mathcal{D} | I \subseteq (t - \{class(t)\}) \wedge \nexists r' \in \mathcal{C} : r' \prec r, r' : I' \rightarrow c', I' \subseteq (t - \{class(t)\})\}$. Moreover, the addition to $\mathcal{C}$ (at line 5) of the default rule r_d ensures that $\mathcal{C}$ is also exhaustive, i.e. that every training case of $\mathcal{D}$ is covered by at least one CAR of $\mathcal{C}$.

Finally, for each CAR $r \in \mathcal{C}$ other than r_d, a local probabilistic model $\mathcal{P}^{(r)}$ is built (lines 7-9) over $\mathcal{D}_r$ to catch a better generalization of those globally rare cases/classes that become less rare within $\mathcal{D}_r$. This allows to refine the prediction from r with a local generative model that is better suited to deal with the local facets of rarity.

The MINECARS procedure is covered in subsection 4.2.1. The TRAINLOCALCLASSIFIER step is instead discussed in subsection 4.2.2, that also covers the classification of unlabeled cases (not reported in algorithm 1) in the context of two schemes for a tight integration between associative and local probabilistic classification.

4.2.1 Mining the Class Association Rules

MINECARS is an Apriori-based algorithm, adopted to mine positively-correlated CARs from the available training data $\mathcal{D}$. MINECARS combines into the basic Apriori algorithm [4] two individually effective mechanisms, namely multiple minimum

Algorithm 1 HierarchicalLearning($\mathcal{M}$,$\mathcal{D}$,$\mathcal{L}$,τ)

Input: a finite set $\mathcal{M}$ of boolean attributes;
a training dataset $\mathcal{D}$;
a set $\mathcal{L}$ of class labels in $\mathcal{D}$;
and a support threshold τ;
Output: An associative classifier $\mathcal{C} = \{r_1 \vee \ldots \vee r_k\}$ and a set of local classifier $\mathcal{P}_{r_i}$;
1: $\mathcal{R} \leftarrow$ MINECARS$(\mathcal{M}, \mathcal{D}, \tau)$;
2: $\mathcal{R} \leftarrow$ ORDER$(\mathcal{R})$;
3: $\mathcal{C} \leftarrow$ PRUNE$(\mathcal{R})$;
4: **if** there are cases in $\mathcal{D}$ that are not covered by any rule within $\mathcal{C}$ **then**
5: $\mathcal{C} \leftarrow \mathcal{C} \cup \{r_d\}$;
6: **end if**
7: **for** each rule $r \in \mathcal{C}$, such that $r \neq r_d$ **do**
8: $\mathcal{P}^{(r)} \leftarrow$ TRAINLOCALCLASSIFIER(r);
9: **end for**
10: RETURN $\mathcal{C}$ and $\mathcal{P}^{(r)}$ for each $r \in \mathcal{C}$

class support [76] and complement class support [8]. Although both designed to deal with rarity in data, to the best of the candidate's knowledge, the joint effectiveness of such mechanisms has not yet been exploited. Algorithm 3 sketches the scheme of MINECARS algorithm, which divides into frequent itemset discovery (lines 1- 18) and CAR generation (lines 19- 26).

Frequent itemset discovery starts (at line 3) with C_2, a set of candidate 2-itemsets, including an item and a class label. At the generic iteration, MINECARS builds L_k, a set of frequent k-itemsets, from L_{k-1}. Two steps are performed to this purpose. The *join step* (at line 14) involves joining L_{k-1} with itself to yield C'_k, a collection of candidate k-itemsets. Notice that this requires joining pairs of frequent $k-1$-itemsets with identical class labels. The well-known *Apriori* property, according to which an infrequent itemset cannot have frequent supersets, is then used (at line 15) to drop from C'_k those k-itemsets with at least one $k-1$-subset that is not in L_{k-1}. The *support counting step* (lines 5- 12) involves counting the occurrences of the surveyed candidate itemsets in C_k by scanning the training data $\mathcal{D}$. Those candidates whose support exceeds a class-specific threshold are considered to be frequent and retained within L_k. The level-wise search halts when no more frequent itemsets can be discovered.

Multiple minimum class support [76] is employed at line 13 to the purpose of automatically adjusting the global minimum support threshold τ provided by the user to minimum support threshold specific for each class. Essentially, the generic candidate itemset c is frequent if its support is over $\tau \cdot \sigma(class(c))$, the minimum support threshold for $class(c)$. Multiple minimum class support implements a first stage of focused pruning, that dynamically assigns a higher minimum support threshold to majority classes (which prevents from yielding several overfitting rules) and a lower minimum support threshold to minority classes (which enforces the generation of an appropriate number of rules).

Complement class support [8] is instead used in the CAR generation stage, to avoid the specification of a global minimum confidence threshold. In particular, a specific property of complement class support (shown in [8]) is exploited at line 22 to automatically identify a class-specific minimum confidence threshold. According to such a property, a rule $r : I \rightarrow c$ is such that I and c are positively correlated if and only if $conf(I \rightarrow c) > \frac{\sigma(c)}{|\mathcal{D}|}$, where $\sigma(c)$ is the overall number of occurrences of class c in $\mathcal{D}$. Therefore, the CARs whose confidence exceeds (at line 22) the minimum threshold corresponding to their targeted class are guaranteed to be positively correlated. Thus, both confidence and positive correlation between rule components can be verified without additional parameters or further correlation analysis.

The dynamic selection of a class-specific minimum confidence threshold acts essentially a second stage of focused pruning, that ensures the discovery of accurate rules targeting the rare classes and still avoids the generation of an overwhelming number of rules from the predominant classes.

4.2.2 Training Local Classifiers

To improve the classification performance both in the surroundings of decision boundaries as well as within the inner areas of decision regions (wherein classes other than the ones associated to the whole regions may influence the classification of nearby unlabeled cases), each CAR $r \in \mathcal{C}$ is associated with a local probabilistic generative model $\mathcal{P}^{(r)}$, trained over the regularities across the training cases local to $\mathcal{D}_r$. In principle, such regularities are likely to be more descriptive of those globally rare cases/classes that become less rare within $\mathcal{D}_r$. Hence, the individual $\mathcal{P}^{(r)}$ can be involved into the classification process for more accurately dealing with the corresponding forms of rarity.

In this thesis, it is adopted, as probabilistic generative model, the naïve Bayes classification model. It, naturally, allows to incorporate the effects of locality on classes and cases in terms of, respectively, class priors and item posteriors. To elucidate, an unlabeled case $I \subset \mathcal{M}$ is assigned by the generic generative model $\mathcal{P}^{(r)}$ to the class $c \in \mathcal{L}$ with highest posterior probability

$$\mathcal{P}^{(r)}(c|I) \triangleq p(c|I,r) = \frac{\prod_{i \in I} p(i|c,r)p(c|r)}{\sum_{\overline{c} \in \mathcal{L}} \prod_{i \in I} p(i|\overline{c},r)p(\overline{c}|r)}$$

Locality influences factors $p(c|r)$'s and $p(i|c,r)$'s, whose values are estimated by computing $p(c)$ and $p(i|c)$ over $\mathcal{D}_r$, and allows to better value rare cases/classes. Indeed, if a significant extent of some form of rarity falls within $\mathcal{D}_r$, the corresponding cases/classes are obviously less rare than in $\mathcal{D}$ and, hence, factors $p(c)$'s and $p(i|c)$'s are accordingly higher (w.r.t. their values in $\mathcal{D}$). Dually, $p(c)$'s and $p(i|c)$'s are sensibly lower, if the density of that form of rarity within $\mathcal{D}_r$ is much lower than in $\mathcal{D}$. However, this is acceptable, since most of that form of rarity is still captured within some other region(s). An inconvenient behind the adoption of naïve Bayes as the underlying model for local probabilistic classifiers is its performance degrade (e.g. accuracy loss) due to the attribute independence assumption. To alleviate such an issue, the weaker attribute independence assumption postulated in AODE [101] can be

Algorithm 3 The process for mining class association rules from data with rarity

Input: a finite set of boolean attributes $\mathcal{M}$;
a training dataset $\mathcal{D}$;
and a support threshold τ;
Output: a set $\mathcal{R}$ of class association rules;
/* Frequent itemset discovery */
$I \leftarrow \emptyset$, $k \leftarrow 2$;
Let $\mathcal{L}$ be the set of class labels in $\mathcal{D}$;
Let $C_2 \leftarrow \{c | c = \{l, i\}$ where $l \in \mathcal{L}, i \in \mathcal{M}\}$;
while $C_k \neq \emptyset$ **do**
  for each candidate itemset $c \in C_k$ **do**
    $supp(c) \leftarrow 0$;
  end for
  for $t \in \mathcal{D}$ **do**
    for $c \in C_k$ such that $c \subseteq t$ **do**
      $supp(c) \leftarrow supp(c) + \frac{1}{|\mathcal{D}|}$;
    end for
  end for
  $L_k \leftarrow \{c \in C_k | supp(c) > \tau \cdot supp(class(c))\}$;
  $C'_{k+1} \leftarrow \{c_i \cup c_j | c_i, c_j \in L_k \wedge class(c_i) = class(c_j) \wedge |c_i \cup c_j| = k + 1\}$;
  $C_{k+1} \leftarrow \{c \in C'_{k+1} | \forall c' \subset c$ such that $|c'| = k$ it holds that $c' \in L_k\}$;
  $k \leftarrow k + 1$;
end while
$\mathcal{I} \leftarrow \cup_k L_k$;
/* CAR generation */
$\mathcal{R} \leftarrow \emptyset$;
for each frequent itemset $I \in \mathcal{I}$ **do**
  create rule $r : I - class(I) \rightarrow class(I)$;
  if $conf(r) > \frac{\sigma(class(I))}{|\mathcal{D}|}$ **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$;
  end if
end for
RETURN $\mathcal{R}$;

plugged into the above formulation, that simply refines naïve Bayes by considering each attribute dependent upon at most n other attributes in addition to the class. This is more realistic in practical applications and is empirically shown in section 4.4 to yield a better performance.

Two alternative approaches for refining the predictions from the associative classifier $\mathcal{C}$ through the local probabilistic generative models $\mathcal{P}^{(r)}$'s are discussed next.

Local priors and local instance posteriors.

The idea is to reformulate a generative approach to classification which spans into local generative models. Given an unlabeled case I, let $\Omega = \{e_r | r \in \mathcal{C}\}$ a space of events related to the classification of I via an associative classifier $\mathcal{C}$. More precisely, the individual event e_r corresponds to the coverage of I through a corresponding

CAR $r \in \mathcal{C}$. The exclusiveness and exhaustiveness of the CARs in $\mathcal{C}$ imply, respectively, the mutual exclusiveness and the collective exhaustiveness of the events in Ω. Therefore, it is possible to employ the well-known law of total probability to define a joint probability distribution over unlabeled cases and class labels as shown below

$$p(c, I) = \sum_{e_r \in \Omega} p(c, I, r) = \sum_{e_r \in \Omega} p(c, I|r)p(r) = \sum_{e_r \in \Omega} \mathcal{P}^{(r)}(c|I)p(I|r)p(r)$$

The interpretation of the terms within the above formula is provided next. $p(I|r)$ represents the compatibility of I with the rule r. $p(I|r)$ is modeled as the relative number of items that I shares with r: intuitively, the number of (mis)matches represents the closeness of I to the region bounded by r. $\mathcal{P}^{(r)}(c|I)$ denotes the probability associated with c by the local naïve Bayes classifier $\mathcal{P}^{(r)}$ trained over $\mathcal{D}_r$. $p(r)$ indicates the support $supp(r)$ of CAR r and weights its contributions to $p(c, I)$ by the relative degree of rarity of its antecedent and consequent.

Finally, the probability of class c given the unlabeled case I can be formalized as the following generative model

$$p(c|I) = \frac{p(c, I)}{\sum_{\overline{c} \in \mathcal{L}} p(\overline{c}, I)}$$

Cumulative rule effect.

A stronger type of interaction between global and local effects can be injected into the classification process, if the predictions from a CAR r and unrelated local generative model $\mathcal{P}^{(r')}$ (with $r \neq r'$) are compared for selecting the most confident one. The overall approach sketched in algorithm 4. Precisely, the generic unlabeled case $I \subset \mathcal{M}$ is presented to the associative classifier $\mathcal{C}$ and the first CAR $r : I \rightarrow c$ (in the precedence order $\prec$ enforced over $\mathcal{C}$) is chosen (at line 1). If r does not cover I, it is skipped and the next rule is recursively taken into account (at line 20). Otherwise, r is used for prediction. However, its target class c is not directly assigned to I. Rather, the local probabilistic generative model $\mathcal{P}^{(r)}$ corresponding to r is exploited to produce a possibly more accurate prediction (at line 4). Some tests are performed to identify the more confident prediction (lines 9- 15). If both counterparts agree or one is deemed to be more reliable than the other one, the better prediction (in terms of class-membership probability distribution) is returned (lines 10 and 12). Otherwise, in the absence of strong evidence to reject the prediction from $\mathcal{P}^{(r)}$ (which is in principle preferable to r, being more representative of the local regularities that may come from globally rare cases/classes that fall within $\mathcal{D}_r$), r is skipped in favor of the next CAR $r' \in \mathcal{C}$ covering I (at line 14). To this point, if $\mathcal{P}^{(r')}$ predicts I more confidently than $\mathcal{P}^{(r)}$ (at line 5), the probability distribution from $\mathcal{P}^{(r')}$ replaces the current best distribution yielded by $\mathcal{P}^{(r)}$ (at line 6) and the choice of a better prediction is hence made between r' and $\mathcal{P}^{(r')}$. In the opposite case, the choice involves r' and the current best distribution $\mathcal{P}^{(r)}$. If no prediction is clearly eligible as the most confident throughout the search, the process halts when the default rule is met

and the current best distribution is returned (at line 17). Notice that the so far best class-membership probability distribution is remembered throughout the consecutive stages of the search process via the input arguments $p_1, \ldots, p_k$ (such arguments are individually set to 0 at the beginning of the search process). A key aspect of the overall search process is represented by the criteria adopted to choose the more confident prediction between the ones from a CAR r_h and a local probabilistic generative model $\mathcal{P}^{(r_i)}$. Accuracy is used as a discriminant between the alternatives. In particular, the accuracy $acc^{(c)}\left(\mathcal{P}^{(r_i)}\right)$ is the percent of cases in $\mathcal{D}^{(r)}$ correctly predicted by $\mathcal{P}^{(r_i)}$ as belonging to class c. The accuracy $acc^{(c)}(r_h)$ of a CAR r_h predicting class c is its confidence $conf\,(r_h)$. When comparing the accuracies of a CAR r_h and a local probabilistic generative model $\mathcal{P}^{(r_i)}$ there four possible outcomes.

1. $\mathcal{P}^{(r_i)}$ is clearly deemed more reliable than r_h (at line 9), if the weighted accuracy of the former, p^*, is greater than the accuracy of the latter.
2. r_h is preferred to $\mathcal{P}^{(r_i)}$ (at line 11) if the accuracy of the former is greater than or equal to the weighted accuracy of the latter and both agree anyhow.
3. r_h is preferred to $\mathcal{P}^{(r_i)}$ (again at line 11) if its accuracy is much greater than the weighted accuracy of $\mathcal{P}^{(r_i)}$. Therein, $\frac{p^*}{p} > p*$ is a prudential threshold, that represents the normalized weighted accuracy from $\mathcal{P}^{(r_i)}$. In practice, r_h is actually preferable to $\mathcal{P}^{(r_i)}$ iff its accuracy exceeds $\frac{p^*}{p}$.
4. There is no strong evidence (at line 16) to reject either r_h or $\mathcal{P}^{(r_i)}$ when the accuracy of r_h lies in the interval $(p*, \frac{p^*}{p})$. In such a case, r is skipped and the search proceeds to considering the next CAR in the associative classifier $\mathcal{C}$ that covers I (through the recursive call at line 14).

4.3 The *Local-to-Global* Supervised Learning Framework

Here is proposed a *local-to-global* learning framework, that uses suitable features local to a data case, for predicting the global conditional probability of classes given the case. Features are a sort of declarative mechanism for specifying aspects of data cases that are relevant, to some extent, towards classification into the individual classes. The relevance of a feature with respect to a certain class determines the weight of that feature on the discrimination of the particular class. Thus, the recognition of minority classes can be addressed by identifying specific features that are highly representative of such classes.

The starting point is the observation that the training data $\mathcal{D}$ generally provides partial information on the associations between data cases and corresponding class labels. The latter is especially true in imprecise domains because of rarity. This suggests that the conditional probability distribution of classes should minimize commitment, i.e. fit the evidence observable in $\mathcal{D}$ and still be as uniform as possible in the prediction of whatever is not observable in $\mathcal{D}$. Such a conditional probability distribution represents the most unbiased assignment of class probabilities complying

Algorithm 4 Prediction($\mathcal{C}$,I,$p_1, \ldots, p_k$)

Input: An associative classifier $\mathcal{C}$;
an unlabeled case $I \subset \mathcal{M}$;
Output: the class distribution for I;

```
select the first rule r : I' → c_h in sequence within C;
if r covers I (i.e. I' ⊆ I) then
    if |C| > 1 (i.e. r is not the default rule) then
        let p̄_i = P^(r)(c_i|I) · acc^(c_i)(P^(r)), ∀i = 1, ..., k;
        if max_i(p̄_i) > max_i(p_i) then
            let p_i = p̄_i, ∀i = 1, ..., k;
        end if
        let p* = max_i(p_i) and i* = argmax_i(p_i) and p = Σ_i p_i;
        if acc^(c_h)(r) < p* then
            RETURN the distribution (p_1/p, ..., p_k/p);
        else if i* = h or acc^(c_h)(r) > p*/p then
            RETURN the distribution (acc^(c_1)(r), ..., acc^(c_k)(r));
        else
            Prediction(C − {r},I,p_1, ..., p_k);
        end if
    else
        RETURN the distribution (p_1/p, ..., p_k/p);
    end if
else
    Prediction(C − {r},I,p_1, ..., p_k);
end if
```

with the observable evidence. Any other probabilistic assignment would be biased, i.e. would assume the availability of arbitrary information that is not present in $\mathcal{D}$.

To elucidate, consider an hypothetical four-class classification setting, where $\mathcal{L} = \{c_1, c_2, c_3, c_4\}$. Classes c_1 and c_2 are rare, whereas c_3 and c_4 are predominant. Assume that an exploratory analysis of $\mathcal{D}$ reveals that a certain itemset $I \subset \mathcal{M}$ appears within classes c_1 and c_2 with a frequency that amounts to, respectively, 50% and 30% of the overall number of its occurrences. I can be viewed as a data feature and the statistical observations concerning I can be stated as constraints for the conditional probability distribution in order for the latter to agree with the empirical evidence. When a newly arrived case I' is presented to the conditional probability distribution for classification, there are two possibilities. If I' includes I (i.e. $I \subseteq I'$), the conditional probability distribution provides the assignments $p(c_1|I') = 0.5$ and $p(c_2|I') = 0.3$. The remaining 0.2 of the probability mass is uniformly distributed between classes c_3 and c_4 (in the absence of any further specific information on this aspect), so that $p(c_3|I') = 0.1$ and $p(c_4|I') = 0.1$. Notably, I is inherently characteristic of the rare classes c_1 and c_2 and its adoption as a data feature allows for a proper discrimination of such classes. If instead I' does not contain I, the conditional probability distribution assumes (in the absence of any further evidence observable in $\mathcal{D}$) maximal ignorance and, hence, predicts each of the four possible classes with

uniform probability, i.e. $p(c_1|I') = p(c_2|I') = p(c_3|I') = p(c_4|I') = 0.25$. In this manner, the conditional probability distribution agrees with the observed evidence in $\mathcal{D}$ and still avoids assumptions on whatever is unknown.

The *local-to-global* approach relies on statistical modeling and learning to fit the evidence in $\mathcal{D}$. For this purpose, the training data is used to identify a set of features useful for classification. The individual features are then employed to specify as many constraints for the conditional probability distribution to learn. The generic constraint essentially forces the expected value that the conditional probability distribution assigns to some corresponding feature to be the same as the expected value of that feature empirically observed in $\mathcal{D}$. In general, the space of features can be potentially very large. In these cases, computing the optimal conditional probability distribution as a closed form solution that meets all the specified constraints is prohibitive. Maximum entropy model [19] provides an expressive and powerful mathematical framework for iteratively computing the required distribution. It is also used to elegantly and seamlessly integrate two established methods from the fields of machine learning and data mining. On the machine learning side, discriminative learning is used to directly compute the conditional probability distribution of the classes, given an unseen case. The main difference with respect to generative learning, that would instead model a joint probability distribution over classes and cases, is that discriminative learning allows to better fit the training data by carefully setting the distribution parameters. On the data mining side, associative classification provides the space of features to which the training and newly arrived data is mapped.

4.3.1 Modeling Data Evidence through CARs, Features and Constraints

In the proposed *local-to-global* learning framework, features are associated to the individual CARs of an associative classifier formed as described in subsection 4.2.1 (no further post-pruning is applied to these CARs). Therein, let $\mathcal{C}$ be an associative classifier. The space of features $\mathcal{F} = \{f_{r_i} | r_i \in \mathcal{C}\}$ is essentially a finite set of real-valued indicator functions f_{r_i}, each of which is associated to a corresponding CAR $r_i \in \mathcal{C}$. Assume that the generic r_i has the implicative structure $r_i : I' \rightarrow c'$, with $I' \subset \mathcal{M}$ and $c' \in \mathcal{L}$. Moreover, let $I \subset \mathcal{M}$ denote a data case and $c \in \mathcal{L}$ represent any class label. The generic feature f_{r_i} is defined as the following indicator function

$$f_{r_i:I' \rightarrow c'}(I, c) = \begin{cases} 1 \text{ if } I' \subseteq I \wedge c' = c \\ 0 \text{ otherwise} \end{cases}$$

The individual feature $f_{r_i:I' \rightarrow c'}$ is said to be *local* to I if $f_{r_i:I' \rightarrow c'}(I, c) > 0$. Training and newly arrived cases can hence be represented as suitable configurations of local features, which are useful for classification. Intuitively, the interpretation of CAR r_i is that I' can be viewed as a sort of contrast set [13] for the targeted class c', i.e. as a co-occurrence of items that is inherently characteristic of c', since its distribution in $\mathcal{D}$ is meaningfully associated with the class c'. Therefore, if r_i covers I (i.e. $I' \subseteq I$) and the class c considered for I coincides with the class c' targeted by r_i, c' is an eligible class for I. Therein, a measure of the suitability of c' for I is provided by the value of $f_{r_i:I' \rightarrow c'}$.

Actually, features are normalized so that their sum amounts to 1. Here, it is specified that the generic feature $f_{r_i:I'\to c'}(I,c)$ is normalized into the corresponding

$$f'_{r_i:I'\to c'}(I,c) = \frac{f_{r_i:I'\to c'}(I,c)}{\sum_{r_i} f_{r_i:I'\to c'}(I,c)}$$

and additionally highlight that, for simplicity, the original notation $f_{r_i:I'\to c'}(I,c)$ is still maintained in the ongoing discussion to mean $f'_{r_i:I'\to c'}(I,c)$.

Features are the basic building block for specifying constrains. These are necessary to make the required conditional probability distribution fit the observed evidence in $\mathcal{D}$. To elaborate, the empirical evidence relative to each feature f_{r_i} is summarized into $E_{\mathcal{D}}(f_{r_i})$, which is the expected value of f_{r_i} observed in $\mathcal{D}$. Its definition is

$$E_{\mathcal{D}}(f_{r_i}) = \sum_{t\in\mathcal{D}} p_{\mathcal{D}}(t) f_{r_i}(t - class(t), class(t))$$

where $p_{\mathcal{D}}$ is the observed occurrence frequency of t in $\mathcal{D}$, i.e. $p_{\mathcal{D}}(t) = \frac{1}{|\mathcal{D}|}$. Constraints force the required conditional probability distribution $\mathcal{P}$ to agree with the feature expectations observed in $\mathcal{D}$. In other words, for each feature f_{r_i}, a corresponding constraint is specified that equates the expected value that $\mathcal{P}$ assigns to f_{r_i} to the expected value $E_{\mathcal{D}}(f_{r_i})$ observed in $\mathcal{D}$. With respect to the generic feature f_{r_i}, the expected value of f_{r_i} due to $\mathcal{P}$ can be approximated as shown below

$$\begin{aligned} E(f_{r_i}) &= \sum_{I\subset\mathcal{M}, c\in\mathcal{L}} p(I,c) f_{r_i}(I,c) \\ &= \sum_{I\subset\mathcal{M}} p(I) \sum_{c\in\mathcal{L}} \mathcal{P}(c|I) f_{r_i}(I,c) \\ &\approx \sum_{I\subset\mathcal{M}} p_{\mathcal{D}}(I) \sum_{c\in\mathcal{L}} \mathcal{P}(c|I) f_{r_i}(I,c) \end{aligned}$$

where the (unknown) prior probability distribution of cases $p(\cdot)$ is approximated by the empirical distribution $p_{\mathcal{D}}(\cdot)$. The term $p_{\mathcal{D}}(I)$ approximates $p(I)$ by the occurrence frequency of I in $\mathcal{D}$.

Finally, restricting $\mathcal{P}$ to have the same feature expectations as the ones observed in $\mathcal{D}$ requires setting the following constraints

$$E(f_{r_i}) = E_{\mathcal{D}}(f_{r_i}) \text{ for each } f_{r_i} \in \mathcal{F}$$

The above restrictions exclude from further consideration all those conditional probability distributions, that do not accord with the observed feature expectations.

In principle, there are infinitely many conditional probability distributions consistent with the specified constraints. The maximum entropy principle suggests to choose the conditional probability distribution $\mathcal{P}$ that fits the constraints (i.e. agrees

with the observed evidence) and maximizes entropy for those cases that are not subject to the constraints. These latter cases are hence predicted to be members of the distinct classes with the most uniform probability distribution.

The mathematical derivation of the required conditional distribution $\mathcal{P}$ as well as the estimation of its parameters are beyond the scope of this manuscript. The interested reader is referred to [19] for an exhaustive coverage.

4.4 Evaluation

It was conducted a systematical experimental study devoted to understand whether the proposed hierarchical classification scheme exhibits improvement in classification performance with respect to established competitors. To this purpose, the comparative evaluation is carried out over some standard datasets. In particular, some datasets chosen from the UCI KDD repository [9], with high class imbalance, are used. Also, the approach is tested over the KDD99 intrusion detection dataset, named `kdd99`. The latter is a extremely unbalanced dataset, wherein low-frequency classes are characterized by noise. A further non-publicly available test dataset, `fraud`, is a real-life fraud detection dataset, with a very low class separability.

Experiments consists in comparisons against several established rule-based and associative classifiers. The selected rule-based competitors are Ripper [33] and PART [51], while the associative ones include CBA [75] and CMAR [106]. In particular, the implementations of CBA and CMAR in [32] were exploited. All tests are conducted on an Intel Titanium processor with 4Gb of memory and 2Ghz of clock speed.

All numeric attributes in the selected datasets are suitably discretized prior to the application of the devised schemes. The adopted discretization strategy partitions the values of each numeric attribute into natural clusters, via model-based clustering. The idea is to view the values of a numeric attribute as the result of a statistical generative process, which is modeled through a mixture of univariate Gaussian distributions. For each numeric attribute, the choice of the most appropriate number of clusters (i.e. distinct Gaussian distributions in the mixture model) is performed by letting such number range from 1 up to a certain maximum, which is fixed to 16 in the experimental setting. The discretization of each numeric attribute into any number of clusters in the aforesaid range is then assessed through 5-fold cross validation. More precisely, 4 folds of attribute values are used to estimate the values of the parameters in the hypothesized mixture model (i.e. the mean and standard deviation for each Gaussian distribution as well as the weights of the individual distributions) by means of the well-known EM algorithm [80]. The remaining fold is employed to evaluate discretization quality. This latter step involves employing the estimates of model parameters to compute the likelihood of the attribute values in the test fold. Eventually, the number of clusters chosen to partition the values of the generic numeric attribute is the one with maximum average likelihood on the test fold and, thus, the values of the attribute are replaced by the label of the cluster to which they belong with highest probability.

The execution of the selected classifiers is reiterated several times, under different parameter configurations and the result of the individual execution were averaged through leave-one-out method. For each classifier, the results corresponding to the best parameter configuration are reported.

Schemes simply require the specification of a global minimum support. Due to the adoption of minimum class support [75], such threshold is automatically adjusted to become a class specific threshold. In particular, the global support threshold is fixed to 20%, which is transparently adjusted to be, within the individual class in the data at hand, the 20% of the frequency of that class. The exploitation of complement class support [76] permits to avoid specifying a minimum confidence threshold.

The approaches are compared using accuracy, some meaningful ROC curves and the Area Under the Curve (AUC) relative to the minority class. Tables 4.1 and 4.2 display the results. Within the tables, competitors are numbered from (1) to (4). Precisely, (1) indicates Ripper, (2) corresponds to PART, while (3) and (4) stand for CBA and CMAR, respectively. Proposed schemes are instead numbered from (5) to (9). More specifically, (5) and (6) indicate naive Bayesian smoothing (respectively through local priors or cumulative effect), (7) and (8) are AODE smoothing (respectively, through local priors or cumulative effect). Finally, (9) represents the maximum entropy approach.

Table 4.1. Classification accuracy

Dataset	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
anneal	98.26	98.25	92.81	96.33	98.53	98.53	98.43	98.43	66.70
balance-scale	80.30	83.17	68.81	68.49	81.40	81.04	80.27	80.30	63.52
breast-cancer	71.45	69.41	69.20	67.67	70.34	70.34	72.30	72.30	76.02
horse-colic	85.10	84.37	81.62	83.96	82.56	82.56	83.20	83.20	85.21
credit-rating	85.16	84.45	81.74	83.76	80.48	80.48	85.90	85.90	85.32
german_credit	72.21	70.54	73.10	73.34	74.03	74.03	74.87	74.87	69.67
pima_diabetes	75.18	73.45	77.87	73.03	73.31	73.31	75.02	75.02	64.39
Glass	66.78	68.75	72.69	74.23	58.94	59.17	67.48	67.15	67.32
cleveland-14-heart-diseas	79.95	78.00	82.12	75.12	81.29	81.29	81.15	81.01	90.54
hungarian-14-heart-diseas	79.57	81.14	82.06	79.69	81.24	81.24	82.62	82.38	86.70
heart-statlog	78.70	77.33	82.59	84.19	80.41	80.41	78.96	78.96	88.70
hepatitis	78.13	79.80	79.89	81.08	81.22	81.22	81.10	81.10	80.63
ionosphere	89.16	90.83	87.89	89.74	82.85	82.85	88.30	88.30	75.21
labor	83.70	77.73	86.67	88.77	84.60	84.60	87.13	87.13	100.00
lymphography	76.31	76.37	81.18	89.59	78.38	78.38	78.00	78.08	88.54
sick	98.29	98.62	97.51	97.64	98.25	98.25	98.39	98.39	97.64
sonar	73.40	77.40	80.00	82.78	75.28	75.28	73.79	73.79	96.63
fraud	93.07	93.02	80.82	90.52	91.78	91.79	92.61	92.61	93.27
kdd99	96.61	96.98	94.65	94.63	95.98	95.98	96.65	96.65	92.34

The results clearly state that the combination of associative classification and probabilistic smoothing is at least as accurate as the seminal rule-based classifiers

Table 4.2. Area Under the Curve

Dataset	(1)	(2)	(5)	(6)	(7)	(8)	(9)
anneal	0.76	0.88	0.93	0.93	0.93	0.93	0.90
balance-scale	0.86	0.92	0.94	0.94	0.87	0.87	0.67
breast-cancer	0.60	0.59	0.67	0.67	0.69	0.69	0.75
horse-colic	0.83	0.86	0.85	0.85	0.88	0.88	0.95
credit-rating	0.87	0.88	0.88	0.88	0.93	0.93	0.94
german_credit	0.63	0.67	0.77	0.77	0.78	0.78	0.71
pima_diabetes	0.72	0.78	0.78	0.78	0.79	0.79	0.76
Glass	0.80	0.79	0.80	0.80	0.81	0.80	0.76
cleveland-14-heart-diseas	0.81	0.80	0.88	0.88	0.90	0.89	0.97
hungarian-14-heart-diseas	0.78	0.86	0.88	0.88	0.90	0.90	0.92
heart-statlog	0.80	0.78	0.86	0.86	0.81	0.81	0.96
hepatitis	0.62	0.78	0.80	0.80	0.84	0.84	0.99
ionosphere	0.89	0.89	0.90	0.90	0.90	0.90	0.97
labor	0.82	0.73	0.86	0.86	0.95	0.95	1.00
lymphography	0.40	0.64	0.56	0.56	0.98	0.89	0.94
sick	0.94	0.95	0.97	0.97	0.96	0.96	0.90
sonar	0.75	0.79	0.80	0.80	0.77	0.77	1.00
fraud	0.68	0.77	0.81	0.81	0.92	0.90	0.60
kdd99	0.98	0.99	0.99	0.99	0.99	0.99	0.92

chosen for the comparison. In many cases, however, (5) and (9) achieve improvements in accuracy, that are statistically significant according to the t-test. In addition, a deeper analysis reveals that the response versus the classes of interest is strongly improved. Such an improvement can be appreciated by looking at the details of the individual datasets. To elucidate, in fig. 4.4 the confusion matrices originated by (1) and (7) over the `german-credit` dataset are reported. Notice that the probabilistic smoothing recovers 39 tuples to the minority class, thus allowing to achieve higher precision.

Predicted ->	good	bad
good	607	93
bad	155	145

AODE local priors (9)

Predicted ->	good	bad
good	611	89
bad	194	106

Ripper (1)

Fig. 4.1. A comparison between the confusion matrices yielded by AODE local priors (7) and Ripper (1) over the `german-credit` dataset

A further analysis of the results obtained over the `fraud` and the `kdd99` datasets provides an in-depth into the effects of smoothing. Figure 4.4 shows the ROC curves relative to (1), (2), (5) and (7). There is an evident improvement in the underlying area with respect to the competitors (1) and (2), whose trends are plotted in red. Results with the `kdd99` dataset are even more surprising, and in particular with the

u2r class, as shown in figure 4, that represents the curves relative to the schemes (1), (2) and (7). The u2r class is made of 56 tuples (out of 150K), and still the probabilistic adjustment is capable of recovering some problematic cases.

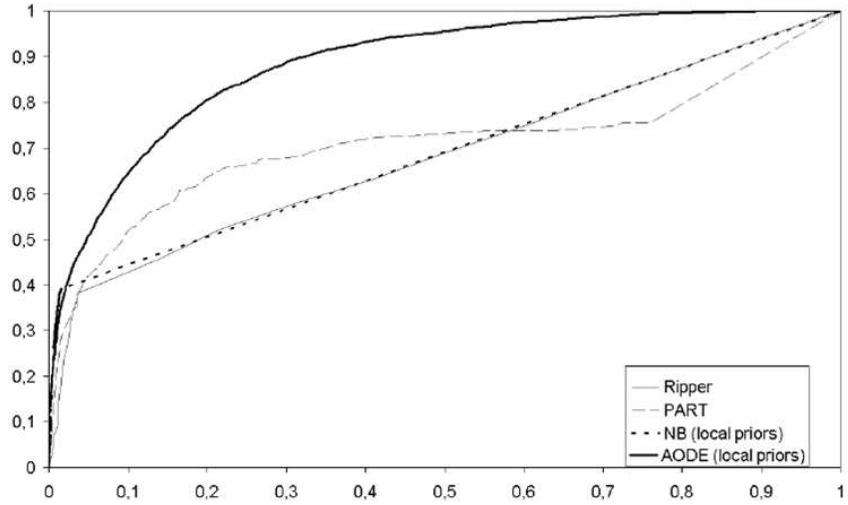


Fig. 4.2. ROC curve for the minority class of fraud

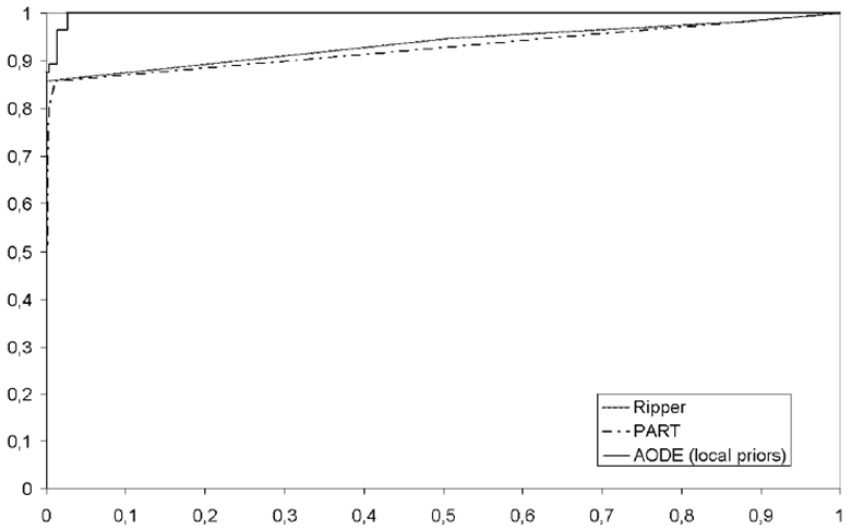


Fig. 4.3. ROC curve for the minority u2r class within the kdd99 dataset

Finally, the ability of the approaches at dealing with the classes is compared in table 4.2, which tabulates the average of the AUC values over the classes within the selected datasets. Overall, the devised schemes outperform the competitors by exhibiting a significantly improved performance (i.e. a considerable increase in the area under the ROC curve) across all classes within the distinct datasets and, in particular, with `hepatitis`, `lymphography` and `fraud`, where the improvement is over 10%.

5

Collaborative Filtering

Collaborative filtering (CF) data exhibit global patterns (i.e. tendencies of some products of being 'universally' appreciated) as well significative local patterns (i.e tendency of users belonging to a specific community to express similar preference indicators on the same items). Local preferences affect the performance of the *Recommender System* (RS) especially when the number of users and items grows, and their importance has been acknowledged by the current CF literature.

Typically, local patterns can be better detected by means of co-clustering approaches [54, 58, 62, 68, 83]. Unlike traditional CF techniques, which try to discover similarities between users or items using clustering techniques or matrix decomposition methods, co-clustering approaches aim to partition data into homogeneous blocks enforcing a simultaneous clustering on both the dimensions of the preference data. This highlights the mutual relationships between users and items: similar users are detected by taking into account their ratings on similar items, which in turn are identified considering the ratings assigned by similar users.

However, a main weakness of the current approaches to co-clustering is the static structure enforced by fixed row/column blocks where both users and items have to fit. For example, the movies "Titanic" and "Avatar", are typically associated with different categories: the former is about romance, whereas the latter can be considered an action, sci-fi movie. Assuming a global and unique partition on the item-set, it is expected to see the movies into different partitions. However, that structure would fail to recognize a group of users who are really into the movies of James Cameron (the director of both movies). Analogously, any method associating the two movies with the same partition would fail in catching the difference in genre.

The issue in the previous example is that different user groups can infer different interpretations of item categories. A more flexible structure, where item categories are conditioned by user categories, would better model such situation, by e.g., allowing "Titanic" and "Avatar" to be observed in the same item category within the "Cameron" group, and in different categories outside. Notice that traditional clustering approaches are not affected by this problem, as they only concentrate on local patterns in one dimension of the rating matrix. The drawback, however, is that they

ignore structural information in the other dimension, which by the converse can be exploited both for more accurate prediction and user profiling.

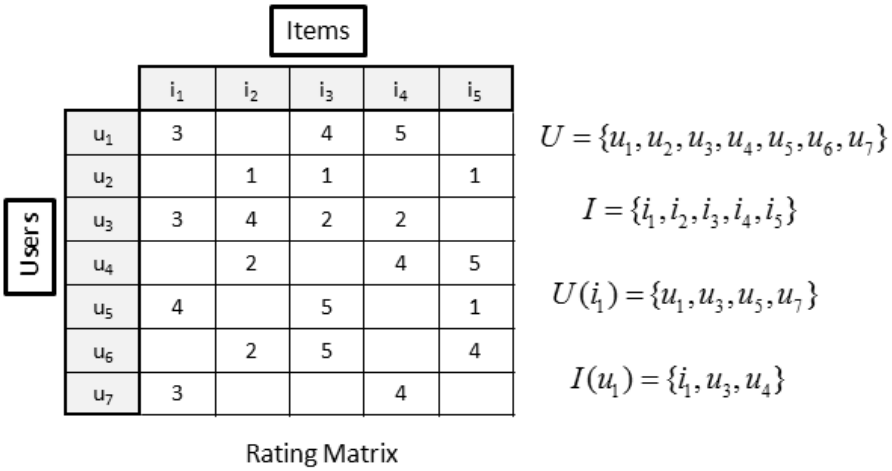
This thesis presents a probabilistic hierarchical approach which is able to discover both global and local trends in data, allowing different user communities to show different preference values on distinct groups of items. The proposed schema differs from the previously proposed coclustering approaches to CF data because it does not assume the existence of a unique partition on the item-set: each user community is characterized by having its own set of topics involving items and user preferences. Following a hierarchical clustering approach, initially, user communities, gathering together similar users, are determined. Then, for each user community the clustering phase produces a mixture of topics upon which the item set and the user preferences are accommodated into categories. Each item group is characterized by the *intracluster consistency property* with respect to the considered user community: each item and its neighbors, associated by having received common rating value in the context of the community, will belong to the same cluster with high probability.

The hierarchical coclustering model does not enforce any strong assumption on the membership of users and items improving the flexibility of the model itself. Each user participates to different user communities with a certain degree and, given a user community, each item may belong to different item-categories. As a result, the proposed model summarizes the advantages of a flexible probabilistic structure for user profiling and a competitive prediction accuracy on user ratings.

5.1 Notation

A RS consists of a set of M users $\mathcal{U} = \{u_1, \cdots, u_M\}$, which will be indicated for short as the *user-set*, a set of N items $\mathcal{I} = \{i_1, \cdots, i_N\}$, named *item-set*, and a collection of rating values expressing the preference of one user on a corresponding item. Such collection of preference indicators can be represented as a $M \times N$ rating matrix $\mathbf{R}$, where r_i^u is the rating given by the user u on the item i. Ratings can be integer values within a scale 1 (low interest) to V (strong interest). Even in the case of a very dynamic system, the rating matrix is typically characterized by an exceptional sparsity rate; if the rating for the pair $(user, item)$ is unknown it will be assumed $r_i^u = 0$.

Let $\mathcal{U}(i)$ the set of users who evaluated the item i, while $\mathcal{I}(u)$ will denote the set of all the items for which the user u has expressed her preference. An example of rating matrix with $M = 7$ users and $N = 5$ items is shown in Fig 5.1. The goal of a RS is to learn a preference function $p : \mathcal{U} \times \mathcal{I} \rightarrow \{1, \cdots, V\}$, which associates to each pair $(user, item)$ a rating value within the admissible range. Let $\hat{r}_i^u$ denote the predicted rating for the pair (u, i). Considering the case of users and products which have provided/received at least one preference value, several evaluation metrics have been proposed to quantify the quality of a prediction algorithm. Denoting by $\mathcal{T}$ a test-set collection of triples $(user, item, rating)$, one of the most referenced methods to measure the performance of a predictor is the Root Mean Squared Error, which emphasizes large errors:



	i_1	i_2	i_3	i_4	i_5
u_1	3		4	5	
u_2		1	1		1
u_3	3	4	2	2	
u_4		2		4	5
u_5	4		5		1
u_6		2	5		4
u_7	3			4	

Fig. 5.1. An example of rating matrix

$$RMSE = \sqrt{\frac{\sum_{(u,i) \in \mathcal{T}} \left(r_i^u - \hat{r}_i^u\right)^2}{|\mathcal{T}|}}$$

In a probabilistic settings, random variables R, I, and U are adopted, and denote a rating, an item and a user respectively. Then, $P(R = r)$ will denote the probability to observe a rating with value r, and analogously $P(U = u)$ will denote the probability that a given rating has been give by user u. With an abuse of notation, the random variable in the specification will be omitted. For example, $P(r, u, i)$ will denote the joint probability $P(R = r, U = u, I = i)$.

5.2 A Hierarchical Co-Clustering Approach for Modeling User Preferences

The starting point in the proposed approach is the observation that different communities can infer different evaluations of the same item. Specific groups of users tend to be co-related according to different subsets of features. However, though semantically-related, two users with (possibly several) differences in their item ratings would hardly be recognized as actually similar by any global model imposing a fixed structure for item categories. Individual user can be intended as a mixture of latent concepts, each of which being a suitable collection of characterizing features. Accordingly, two users are considered as actually similar if both represent at least a same concept. Viewed in this perspective, the identification of *local patterns*, i.e. of proper combinations of users and items, would lead to the discovery of natural clusters in the data, without incurring into the aforesaid difficulties. Consider fig. 5.2. In this toy example, there are 7 users clustered into two main communities. Community 1 is characterized by 3 main topics (with groups $d_{11} = \{i_1, i_2, i_3\}$,

$d_{12} = \{i_4, i_5, i_6, i_7\}$ and $d_{13} = \{i_8, i_9, i_{10}\}$), whereas community 2 includes 4 main topics (with groups $d_{21} = \{i_1, i_4, i_5\}$, $d_{22} = \{i_2, i_3, i_7\}$, $d_{23} = \{i_6, i_{10}\}$ and $d_{24} = \{i_8, i_9\}$). The novelty is that different communities group the same items differently. This introduces a topic hierarchy which in principle increases the semantic power of the overall model.

		i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	
Community 1	u_1	1		1	5		4	5		2	2	$d_1 = \{i_1, i_2, i_3\}$
	u_2	1	1			5	4	4	5	2	2	$d_2 = \{i_4, i_5, i_6, i_7\}$
	u_3	1	1	1	4	5			5	2		$d_3 = \{i_9, i_{10}\}$
	u_4		1	1		5	4	5	4		2	
Community 2	u_5	5		4	5	5	1	4	3		1	$d_1 = \{i_1, i_4, i_5\}$
	u_6		4	4	5	5	1	4	3	3	1	$d_2 = \{i_2, i_3, i_7\}$
	u_7	5	4		5		1	4	3	3		$d_3 = \{i_6, i_{10}\}$
												$d_4 = \{i_8, i_9\}$

Fig. 5.2. Example of Local Pattern in CF Data

The generative model for the proposed scheme is shown fig. in 5.3 and can be summarized as follows:

1. Select a user community c_k according to the probability distribution π_k;
2. select a user u with probability $P_k(u) = P(u|c_k)$ and an item i with probability $P_k(i) = P(i|c_k)$;
3. Choose a topic d_h with probability $P(d_h|i, c_k)P(d_h|u, c_k)$;
4. produce the rating r with probability $\phi_h(r) = P(r|d_h)$.

Formally, the probability of a triplet $\langle u, i, r\rangle$ is

$$P(u, i, r) = \sum_{k=1}^{K} \pi_k P_k(u) P_k(i) P(r|i, u, c_k) \tag{5.1}$$

where

$$P(r|i, u, c_k) = \sum_{h=1}^{H_k} \phi_h(r) P_k(d_h|i) P_k(d_h|u) \tag{5.2}$$

The latter correspond to a "local" probabilistic latent semantic analysis, provided that the user communities are known.

The idea, in the above formula, is learning latent communities from the data as well as a collection of characterizing concepts for each community. In particular, each rating can be seen as the outcome of a mixture of various concepts, where some

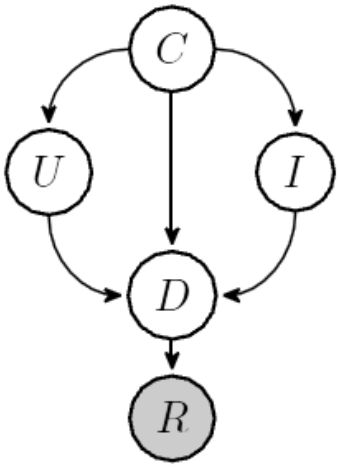


Fig. 5.3. Graphical Model for the Hierarchical Approach

concepts are more or less probable according to the cluster where the user fits. Hence, a data tuple can be thought as the outcome of the following generative model: firstly pick a distribution over latent clusters; next, choose the concepts associated and finally generate the individual values. Also, notice the role of the $\pi_k, k = 1, \ldots, K$ prior probabilities in the generative process. In practice, they model the assumption that observing a pair $\langle u, i \rangle$ is not totally random, but it is instead the result of the grouping of users into communities.

Due to the strong coupling between the user community latent variable c and the one corresponding to local patterns d, the exact inference for the model characterized by the joint probability in eq. 5.1, which would maximize both the user community cohesion and the local topic similarity, is difficult to solve analytically. Hence, an approximated solution is adopted, it is based on a hard clustering policy for user communities, such that the inference of the parameters can be performed efficiently without compromising the generative semantic and the flexibility of the model.

A hierarchical approach is devised to the estimation of the components involved into eq. 5.1. In practice, the proposed approach consists in a preliminary discovery structure, where user communities are detected. Next, for each user community, a topic model is investigated, and the most prominent topics are discovered and properly modeled.

The general scheme of the algorithm is shown in algorithm 5 and could be summarized as follows: given a rating matrix $\mathbf{R}$, discover k user communities; then, for each of those communities, according to an hard clustering approach, select from $\mathcal{U}$ a subset of users that belong to the considered community and generate a set of H_k topic models for their ratings.

The hierarchical model for users' ratings consists in a set of K user community models and for each of them a set of H_k topic models which represent local preference patterns for the member of the considered community. The user community

Algorithm 5 HierarchicalModel_build

Input: The sets $\mathcal{U} = \{u_1, \ldots, u_M\}$ and $\mathcal{I} = \{i_1, \ldots, i_N\}$ and the corresponding rating matrix $\mathbf{R}$;
Output: a set $\mathcal{C} = \{c_1, \ldots c_K\}$ of user community models and a subset $\mathcal{D}^k = \{d_1^{(k)}, \ldots, d_{H_k}^{(k)}\}$ for each user community k

1: $\mathcal{C} \leftarrow GenerateUserCommunities(\mathbf{R})$;
2: **for all** community model c_k, $k = 1, \ldots, K$ **do**
3: let $\mathcal{U}_k = \{u \in \mathcal{U} | p(c_k|u) \geq p(c_j|u), j = 1, \ldots, K\}$, and $\mathbf{R}_k$ the corresponding submatrix of $\mathbf{R}$;
4: $\mathcal{D}^k \leftarrow GenerateTopicModels(\mathbf{R}_k)$;
5: **end for**

level specifies the probabilities $\gamma_{uk} = P(c_k|u)$ with $k = 1, \ldots, K$, which measure how much the ratings given by the user u fit the preference behavior underlined by each of the communities.

The probability of observing the rating r for the pair (u, i) can be computed considering two schema, summarized in algorithm 6:

- *Hard-Clustering Prediction:*

$$P(r|i,u) = \sum_{h=1}^{H_k} \phi_h(r) P_k(d_h|i) P_k(d_h|u) \tag{5.3}$$

where

$$k = argmax_{j=1,\cdots,K}(\gamma_{uj}) \tag{5.4}$$

is the cluster that better represents the previously observed rating of the user u. This prediction rule relies exclusively on the information given by the topic model corresponding to the user's cluster; thus it might produce low quality predictions if the user's community is not identified with enough confidence.

- *Soft-Clustering Prediction:*

$$P(r|i,u) = \sum_k \gamma_{uk} \cdot \tilde{P}(r|i,u,c_k) \tag{5.5}$$

where the probabilities γ_{uk} act as mixture weights and the distribution over rating values corresponding to the community c_k is computed taking into account both global and local patterns:

$$\tilde{P}(r|i,u,c_k) = \begin{cases} P(r|i,u,c_k) & \text{if } u \in \mathcal{U}_k \\ P(r|i,c_k) & \text{otherwise} \end{cases} \tag{5.6}$$

Note that if $u \in \mathcal{U}_k$ then γ_{uk} is the dominant mixing weight and the distribution over ratings is refined by considering the corresponding set of topic models; in the opposite case the distribution over ratings can be estimated by considering the probability of observing each rating given an item within the considered community.

Algorithm 6 HierarchicalModel_computeRatingsProbability

Input: a pair $\langle u, i \rangle$
Output: a probability $P(R = r|u, i)$ for each rating value r

let $c = argmax_{j=1,\cdots,K} p(c_k|u)$
for all $r = 1$ to V **do**
  if *Hard-Clustering* **then**
    $P(R = r|u, i) = \sum_{h=1}^{H_k} \phi_h(r) P_k(d_h|i) P_k(d_h|u)$
  else
    for all community model c_k, $k = 1, \ldots, K$ **do**
      if $k = c$ **then**
        $prob \leftarrow \mathcal{D}^k.getRatingProbability(r, u, i)$
      else
        $prob \leftarrow c_k.getRatingProbability(r, i)$
      end if
      $P(R = r|u, i) \leftarrow P(R = r|u, i) + \gamma_{uk} \times prob$
    end for
  end if
end for

5.2.1 Modeling User Communities.

The discovery of the communities is accomplished essentially via a model-fitting procedure based on a maximum-likelihood estimation. In practice, the rating matrix $\mathbf{R}$ is modeled as a set of user vectors, where each vector is characterized by the preferences of the user. Formally, this means that the probability $p(r, i|u)$ for each triplet $\langle r, i, u \rangle$ can be modeled.

The corresponding probability of observing a user hence corresponds to the joint probability of observing all his ratings, that is

$$P(u|\Theta, \mathbf{R}) = \prod_{i=1}^{N} \prod_{r=1}^{V} \left(P(i|\Theta) \cdot P(r|i, \Theta) \right)^{\delta(u,i,r)}$$

where

$$\delta(u, i, r) = \begin{cases} 1 \text{ if } r_i^u = r \\ 0 \text{ otherwise} \end{cases}$$

This modeling allows us to adopt a maximum likelihood approach to the estimation of the Θ parameters characterizing the $P(i|\Theta)$ and $P(r|i, \Theta)$. For example, $P(i|\Theta)$ can be characterized via a bernoullian pdf parameterized by α_i, and $P(r|i, \Theta)$ as a multinomial (with factors σ_{ri}).

The component $P(r, u, i, c_k)$ and the posteriors γ_{uk} can be estimated by assuming the existence of a set of communities, where each community models specific user attitudes. In particular, the probability of observing a user is given by the mixture

$$P(u|\mathcal{C}) = \sum_{j=1}^{K} P(u|c_j) \pi_j$$

Where a single community j is characterized by the parameters α_{ij} and σ_{rij}.

Estimating the parameters by means of an EM procedure yields the following equations:

- **E-Step:**

$$\gamma_{uj} = P(c_j|u) = \frac{P(u|c_j) \cdot \pi_j}{\sum_{j'=1}^{K} P(u|c_{j'}) \cdot \pi_{j'}}$$

- **M-Step:**

$$\pi_j = \frac{\sum_{u=1}^{M} \gamma_{uj}}{M}$$

$$\alpha_{ij} = \frac{\sum_{u=1}^{M} \gamma_{uj} \sum_{r=1}^{V} \delta(u,i,r)}{\sum_{u=1}^{M} \gamma_{uj} \sum_{i'=1}^{N} \sum_{r=1}^{V} \delta(u,i',r)}$$

$$\sigma_{rij} = \frac{\sum_{u=1}^{M} \gamma_{uj} \cdot \delta(u,i,r)}{\sum_{u=1}^{M} \sum_{r'=1}^{V} \gamma_{uj} \cdot \delta(u,i,r')}$$

A further advantage of the above formalization is the possibility of exploiting the above model for prediction purposes as well as for for structure discovery. A prediction function in fact can be defined as

$$\hat{r}_i^u = E[R|u,i] = \sum_{r=1}^{V} r \cdot \sum_{k} \sigma_{rik} \cdot \gamma_{uk} \tag{5.7}$$

and used as a baseline for the special case described in step 10 of algorithm 6. The resulting baseline function is even competitive with state-of-the art approaches.

The above formalization also allows an alternative gaussian model

$$P(r|i,c_j) = N(v_u^r; \mu_{ij}, \sigma_{ij}) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left[-\frac{(v_u^r - \mu_{ij})^2}{2\sigma_{ij}^2}\right]$$

where v_u^r is the Z-score normalization of r with regards to user u:

$$v_u^r = \frac{r - \mu_u}{\sigma_u}$$

and the means and the variances are estimated as proposed in [57].

The rating prediction for the pair (u,i) can be hence computed as:

$$\hat{r}_i^u = \mu_u + \sigma_u \left(\sum_{k=1}^{K} \gamma_{uk} \cdot \mu_{ik} \right) \tag{5.8}$$

and the **M-Steps** can be rewritten as:

$$\mu_{ik} = \frac{\sum_{u}^{M} \sum_{r}^{V} \gamma_{uk} \cdot \delta(u,i,r) \cdot v_u^r}{\sum_{u}^{M} \sum_{r}^{V} \gamma_{uk} \delta(u,i,r)}$$

$$\sigma_{ik}^2 = \frac{\sum_{u=1}^{M} \sum_{r}^{V} \gamma_{uk} \cdot \delta(u,i,r)(v_u^r - \mu_{ik})^2}{\sum_{u=1}^{M} \sum_{r}^{V} \gamma_{uk} \delta(u,i,r)}$$

5.2.2 Local Community Patterns via Topic Analysis

The approach to the discovery of local community patterns is based again on a EM procedure which aims at maximizing the likelihood of the $\mathbf{R}_k = \{\langle r, u, i\rangle | p(c_k|u) \geq p(c_j|u), j = 1, \ldots, K\}$ rating matrix associated to a community model c_k. In practice, the expected log-likelihood is defined

$$\mathcal{Q}(\mathbf{R}_k; \mathbf{\Psi}) = \sum_{u}^{M}\sum_{i}^{N}\sum_{r}^{V}\sum_{h}^{H_k} \psi_k(h; r, i, u) \cdot \\ [\log \phi_h(r) + \log P_k(d_h|i) + \log P_k(d_h|u)]$$

where $\psi_k(h; r, i, u) = P(d_h|r, i, u, c_k)$. The EM algorithm can hence be defined in terms of the following formulas:

- **E-Step:**

$$\psi_k(h; r, i, u) = \frac{\phi_h(r)P_k(d_h|i)P_k(d_h|u)}{\sum_j \phi_j(r)P_k(d_j|i)P_k(d_j|u)}$$

- **M-Steps:**

$$P_k(d_h|i) = \frac{\sum_u^M \sum_r^V \psi_k(h; r, i, u)}{\sum_{h'} \sum_u^M \sum_r^V \psi_k(h'; r, i, u)}$$
$$P_k(d_h|u) = \frac{\sum_i^N \sum_r^V \psi_k(h; r, i, u)}{\sum_{h'} \sum_i^N \sum_r^V \psi_k(h'; r, i, u)}$$
$$P(r|d_h) = N(r; \mu_{d_h}, \sigma_{d_h})$$

where

$$\mu_{d_h} = \frac{\sum_u^M \sum_i^N \sum_r^V \psi_k(h; r, i, u)\delta(u, i, r) \cdot r}{\sum_u^M \sum_i^N \sum_r^V \psi_k(h; r, i, u)\delta(u, i, r)}$$
$$\sigma_{d_h} = \frac{\sum_u^M \sum_i^N \sum_r^V \psi_k(h; r, i, u)\delta(u, i, r) \cdot (r - \mu_{d_h})^2}{\sum_u^M \sum_r^V \sum_i^N \psi_k(h; r, i, u)\delta(u, i, r)}$$

5.2.3 Computational aspects

Once the parameters of the hierarchical model have been estimated, the on-line complexity for computing predictions scales with the number of user communities and corresponding topics, while the off-line phase requires more resources. In fact, the complexity of the learning phase is determined by the complexity of discovering user communities, which is linear with the number of observed ratings.

To avoid overfitting, which could deteriorate the predictive skills of the models on unobserved data an *Early Stopping* criterion is adopted: a fraction of the data has

been retained as held-out dataset and the models have been trained on the remaining part of the data until the accuracy on the held-out data begins to increase.

The estimation of the correct number of clusters is accomplished by resorting to a Cross-Validation approach based on a penalized Log-Likelihood principle, as described below. Given a set D of observations (in this case, the rating matrix $\mathbf{R}$ and its subsets $\mathbf{R}_k$), the objective is to find the model parameters Θ maximizing the probability $P(\Theta|D)$. In logarithmic terms,

$$\begin{aligned}\log(P(\Theta|D)) \propto &= \log P(D|\Theta) + \log P(\Theta) \\ &= \log(\mathcal{L}(\Theta|D)) + \log P(\Theta)\end{aligned}$$

The idea in the above formula is to counterbalance two opposing requirements: the fitting of the data and the complexity of the model. By modeling $P(\Theta)$ can be modeled as an exponential distribution w.r.t the size of Θ, so

$$\log(P(\Theta|D)) \approx \log(\mathcal{L}(\Theta|D)) - m \log n$$

where m is the size of Θ (i.e., the number of model parameters), and n is the size of D. The evaluation strategy hence consists in computing $\log(P(\Theta|D))$ for each possible Θ, and in choosing the model where it is maximal. In particular, the strategy can be summarized as follows:

1. fix the values K_{min} and K_{max};
2. choose the number C of cross-validation trials;
3. for each trial c:
   a) sample a subset D_{train} from D;
   b) for k ranging from K_{min} and K_{max}:
   c) compute $\log(P(\Theta_k|D_{train}))^c$;
4. for each K, average the values $\log(P(\Theta_k|D_{train}))^c$ over c;
5. choose the value k^* such that $\log(P(\Theta_{k^*}|D_{train}))^{avg}$ is maximal.

5.2.4 Discussion.

There are several major differences between the models described in section 2.3 and the above formalization. Considering pLSA, the hidden variable z there is used to discover similar trends in the rating behavior and encourages grouping users into user communities. The prediction relies solely on $P(r|i,z)$ and does not consider item hierarchies and, hence boosted predictions triggered by similar items. By contrast, the proposed hierarchical approach aims to discover local patterns for each user community. Also, there are two further components which boost the prediction accuracy of the underlying user community model. First, the multinomial prior π_j for each user community j, which helps in preventing overfitting by counterbalancing the contribute of each user u in γ_{uj}. The π_j component can be interpreted as a laplacian smoothing based on uniform Dirichlet priors. Clearly, explicit modeling of such priors via Bayesian estimation, in the style of [78], can be adopted. However, as discussed in the next section, the computational cost would leverage significantly. Also,

the α_{ij} component explicitly models the likelihood that item i has been rated within community j. The latter also is a major difference, at the user community level, with respect to the multinomial mixture and the User Rating Profile models, discussed in [79].

Also, notice that the co-clustering techniques discussed in the previous section, like the Flexible Mixture Model, assume the existence of a fixed partition both for user communities and for item categories. In the analyzed case instead, each user community is characterized by its own partition over the item-set with a flexible number of topics. In addition, co-clustering models only produce prediction on the basis of local contribution $P(r|c_k, d_h)$. By contrast, according to eq. 5.6, prediction of the proposed model benefits from both local and global information.

A final remark is concerned with the possibility of considering the proposed approach symmetrical. The proposed model starts with user communities and then generates topics. In theory a dual scheme could be viable as well, by first generating item categories and then specific user communities conditioned to item categories. However, duality only holds if the number of rows and columns of the rating matrix are of the same order of magnitude. In fact, the number of model parameters in an item-based mixture grows linearly to the number of users. If the number of items is significantly less than the number of users, this would cause the generation of few categories characterized by too many parameters (and as a consequence the resulting model would be prone to overfitting).

5.3 Evaluation

The effectiveness of the proposed approach is evaluated in three different respects:

- To measure the effectiveness of the EM algorithm adopted in the first stage in discovering communities fitting the training data. Since each community should be able to model a user's preferences, it is interesting to measure the prediction accuracy of eq. 5.7 and eq. 5.8, which exploit the community mixtures.
- To measure the overall prediction accuracy of the hierarchical approach, and to compare it to other well-known approaches in the literature.
- To inspect informative content of the structures discovered by the hierarchical approach proposed so far. Essentially, the objective is to inspect the communities and the relevant topics discovered, and to find empirical confirmations concerning the key ideas explained in the beginning of section 5.2.

Two popular benchmark datasets (Netflix and Movielens) are used for rating prediction and validation of the predictive performance of the proposed approach. In short, Netflix dataset contains over 100 million of ratings given by $480,189$ users on a set of $17,770$ movies, collected between October 1998 and December 2005. The Netflix Prize dataset has been the reference data for empirical comparisons of Collaborative Filtering algorithms during the last years, mainly for 3 reasons: (i) size of dataset and sparseness coefficient; (ii) availability of results from competitive algorithms; (iii) availability of a baseline score for the prediction error, achieved by a

	Netflix		MovieLens	
	Training Set	Test Set	Training Set	Test Set
Users	435,656	389,305	6,040	6,040
Items	2,961	2,961	3,706	3,308
Ratings	5,714,427	3,773,781	800,168	200,041
Avg ratings (user)	13.12	9.69	132,47	33,119
Avg ratings (item)	1929.90	1274.50	215.91	60.47
Sparseness Coeff	0,9956		0,9643	

Table 5.1. Summary of the Data used for validation.

real RS (the Netflix Cinematch algorithm) on the same dataset. A subsample of the above data was exploited. The data is divided into training and test set, where the latter contains ratings given by a subset of the users in the training set over the same set of items. Info about this dataset are summarized in tab. 5.1.

fig. 5.4 shows the empirical cumulative densities for both user and item ratings within the subsample adopted here. There are some major differences between the original Netflix dataset and the subsample used here. For example, it can be see from fig. 5.4(a) that over 60% of the users have less than 10 ratings and the average number of evaluations given by users is 13 (whereas the original dataset exhibits an average 200 ratings). In addition, figure 5.4(b) shows that over 50% of the items have received less than 200 ratings, with an average value of 1929. Again, the average ratings in the original dataset were 5000. In practice, the exploited subsample is more difficult than the original dataset. [1]

The MovieLens-1M[2] dataset consists of $1,000,209$ ratings given by $6,040$ users on approximately $3,706$ movies; each user in this dataset has at least 20 ratings. The original data is randomly partitioned into $4/5$ training and $1/5$ test set. Again, tab. 5.1 summarizes the values exhibited by the subsets. MovieLens has been a reference dataset for several CF algorithms.

5.3.1 Predictive Accuracy.

The proposed approach is compared with most algorithms mentioned in section 2.3. In particular, Regularized SVD [7], pLSA [57], FMM [62], Multinomial Mixture Model [79] and URP [78]. The summary of the results can be found in tab. 5.1(a) and tab. 5.1(b). The algorithms not listed here will be discussed separately.

In a first set of experiments, the achieved performance is evaluated by the User Community Models, considering both the Multinomial and the Gaussian version and performed a suite of experiments varying the number of user communities and compared the obtained RMSE values with the ones achieved by the Gaussian pLSA algorithm on the same data.

[1] This also explains the difference between the values declared in the original papers by the competitors and the values reproduced on the subsample.

[2] `http://www.grouplens.org/system/files/ml-data-10M100K.tar.gz`

Experiments on the three models were performed by retaining the 10% of the training (user,item,rating) triplets as held-out data; finally 10 attempts have been executed to determine the best initial configurations. Predictions for the User Community Models are generated according to eq. 5.6, because preliminary experiments have shown that it outperforms the Hard-Clustering prediction rule. Performance results of the two User Communities Models and pLSA are shown in Figures 5.5(a) and 5.5(b).

Considering NetFlix, the multinomial User Community approach and the pLSA do not produce a significant improvement over the Cinematch base, which is close to 0.95; for both these models the best RMSE values is achieved by considering 150 user communities. The average RMSE for the pLSA model is 0.9474 and only minor improvements on this result are observed varying the number of clusters. The gaussian User Community version outperforms both the multinomial model and pLSA, achieving the best RMSE value of 0.9280 when 30 user communities are employed. The learning phase corresponding to the best model takes about 30 minutes on a INTEL XEON E5520 at 2.27 Ghz, with an average of 6 iterations needed to reach convergence. It was not possible to extensively report on FMM and URP here, due essentially to the high computational resources needed by these models. Tab. 5.1(a) shows some partial results on FMM obtained with a limited number of user clusters. Surprisingly, the multinomial User Community model has a significant worsening on MovieLens. The z-score normalization, exploited in eq. 5.8, plays an important role in improving both the clustering and the predictive capabilities of the gaussian model respect to the multinomial version.

Switching to the hierarchical schema allows us to obtain more refined results. The Hierarchical approach has been evaluated by considering both the multinomial and the gaussian version on the first layer clustering (by restricting to a range of user communities from 10 to 100), and adopting the procedure for the dynamic estimation of the number of topics described in section 5.2.3. fig. 5.5(d) and fig. 5.5(c) show the performances achieved by the two version and the ones achieved by a natural competitor based on latent factors: the regularized SVD. In both the cases, the hierarchical approach produces a significant improvement over the first clustering layer, outperforming the SVD model. On Netflix, hierarchical approach produces RMSE values 0.9222 (multinomial model) and 0.9211 (gaussian model), while the best result achieved by the SVD model is 0.9275. This situation is also reflected in MovieLens where the Reg. SVD produced 0.9345. Again, it's a surprise to see that in this case the multinomial hierarchical approach (0.9274) outperforms the Gaussian hierarchical (0.9296). This result is even more surprising, especially because the multinomial user communities didn't perform very well in the first level. It seems that the adoption of specific item categories boosts the performance significantly.

Figure 5.5(e) compares all the probabilistic approaches to co-clustering on MovieLens data. Here there is a comparison of the proposed approaches with FMM, Bregman Co-clustering [54] and Block Mixture Model [83]. Again, the hierarchical approaches outperform the other co-clustering approaches. This gives evidence that conditioning item categories to user communities provides better structures. Finally,

fig. 5.5(f) shows the execution times of these co-clustering approaches. Here, 10 item categories and vary the number of user communities are employed.

A final validation qualitatively compares the proposed approach approach with some among the most popular and effective approaches for making recommendations. Thesis focused on single techniques rather than ensembles or combinations of multiple predictors. Also, models, that directly model temporal aspects, was not taken into account, such as the *Time Effect* normalization [16] or the *SVD withtemporal dynamics* [71]: in fact they exploit temporal information about the preference of the users in a given period in order to refine predictions in the same period, while in a typical setting a recommender system is asked to make suggestions for the future.

(a) NetFlix Data

Approaches	Best RMSE	Parameters
Overall Mean	1.0839	
User Avg	1.0368	
Item Avg	1.009	
Knn Simple	1.0066	$K = 15$
Scalable Coclustering	0.9862	$\#User\ Communities = 3$, $\#Item\ Categories = 5$
Weighted Centering	0.9707	$\alpha = 0.6$
Knn with Double Cen. Baseline	0.9637	$K = 20$
Flexible Mixture Model	0.9540	$\#User\ Communities = 10$, $\#Item\ Categories = 70$
Block Mixture model	0.9477	$\#User\ Communities = 30$, $\#Item\ Categories = 30$
PLSA	0.9474	$\#Latent\ factors = 100$
Knn with user effect baseline	0.9453	$K = 20$
Multinomial Mixture Model	0.9434	$\#Latent\ factors = 10$
User Communities Multinomial	**0.9391**	$\#Latent\ factors = 70$
Regularized SVD	0.9275	$\#Latent\ factors = 100$
User Communities Gaussian	**0.9274**	$\#Latent\ factors = 30$
KNN Relationship model	0.9258	$K = 20$
Hierarchical model Multinomial - Fixed	**0.9251**	$\#User\ Communities = 50$, $\#Item\ Categories = 100$
Hierarchical model Multinomial - Flexible	**0.9222**	$\#User\ Communities = 100$
Hierarchical Model Gaussian - Fixed	**0.9212**	$\#User\ Communities = 50$, $\#Item\ Categories = 100$
Hierarchical Model Gaussian - Flexible	**0.9211**	$\#User\ Communities = 30$

(b) MovieLens-1M Data

Approaches	Best RMSE	Parameters
Overall Mean	1.1150	
User Avg	1.0462	
URP	0.9869	$\#Latent\ factors = 10$
Item Avg	0.9862	
User Communities Multinomial	**0.9638**	$\#Latent\ factors = 4$
Multinomial Mix	0.9640	$\#Latent\ factors = 2$
Weighted Centering	0.961	$\alpha = 0.7$
URP - Boosted	0.9568	$\#Latent\ factors = 3$
PLSA	0.9468	$\#Latent\ factors = 2$
Regularized SVD	0.9345	$\#Latent\ factors = 8$
Block Mixture model	0.9467	$\#User\ Communities = 10$, $\#Item\ Categories = 7$
Scalable Coclustering	0.9416	$\#User\ Communities = 7$, $\#Item\ Categories = 5$
User Communities Gaussian	**0.9359**	$\#Latent\ factors = 2$
Flexible Mixture Model	0.9335	$\#User\ Communities = 10$, $\#Item\ Categories = 10$
Hierarchical Model Gaussian - Fixed	**0.9297**	$\#User\ Communities = 2$, $\#Item\ Categories = 2$
Hierarchical Model Gaussian - Flexible	**0.9296**	$\#User\ Communities = 2$
Hierarchical model Multinomial - Fixed	**0.9278**	$\#User\ Communities = 2$, $\#Item\ Categories = 3$
Hierarchical model Multinomial - Flexible	**0.9274**	$\#User\ Communities = 3$

TABLE II
COMPARATIVE ANALYSIS ON MOVIELENS1M

Table 5.2. Summary of the comparative analysis

Results on Netflix data show that the prediction accuracy achieved by proposed model is competitive to the ones achieved by other popular recent approaches, such as *PMF* [93], *Bi-LDA* [88] and *SVD++* [70]: the first one is reported to achieve on a

sample of 1M ratings of Netflix data an RMSE equals to 0.9253; the latter achieves 0.9333 on the overall Netflix dataset. As far as the *SVD++* is concerned, although it achieves an RMSE value 0.904 on the considered dataset, the problem with such an approach is that it takes advantage of implicit information contained in the test-set.

The proposed model seems to be also competitive with *fLDA* [3] and the *Regression-based latent factor models* [2], which integrate user/item features and on a $75\% - 25\%$ split of the MovieLens-1M achieve RMSE values of 0.9381 and 0.9258.

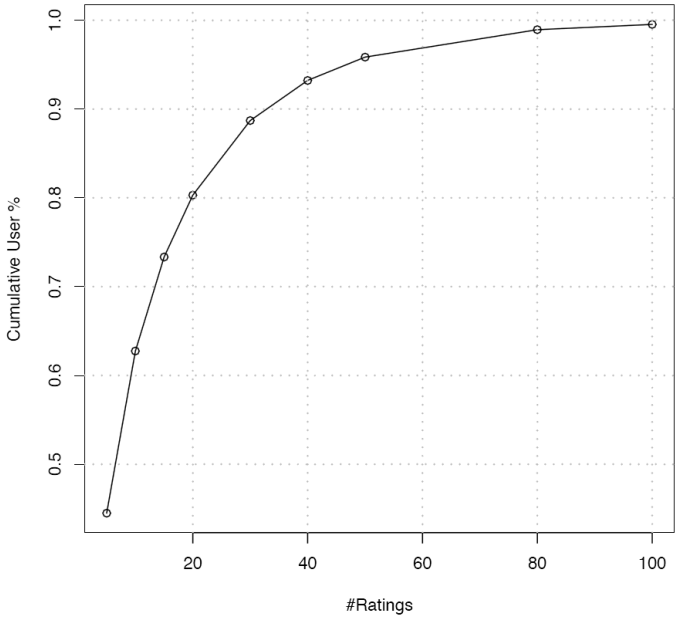
5.3.2 Structure Discovery

The hierarchical model can be used for classical pattern discovery tasks for CF, such as the identification of significant or the most appreciated items for each user community, as well a new kind of analysis, in which focus is on different topics and their impacts on the rating behavior of users within the same community. Table 5.3.2 shows a selection from the most significant items for 10 user communities and their topics. Only 5 communities are shown, and the 5 most relevant topics within them. An item i is considered significant with respect to a topic h within the community k if $P_k(d_h|i) > P_k(d'_h|i)\ \forall h' \neq h$. For each community, its prior probability (in square brackets) and the a-posteriori interpretation of its topics are registered.

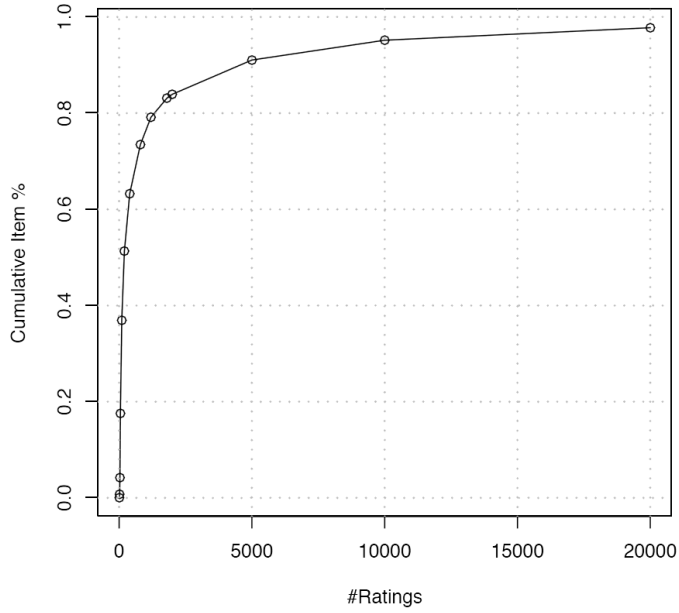
For instance, user community #2 is characterized by the topics: “Fantasy”, “Sci-Fi”, “Live-Music Performance” “Action” and “Drama”. It is worth noticing how the informative content in the hierarchy allows to better discriminate among topics and tendencies. By focusing on the first level only, the same community would exhibit a global attitude towards action movies (as “Gladiator”, “Die Hard” and “Terminator 2” are the most probable items here).

	Community 1 [0.09]	**Community 2** [0.05]	**Community 3** [0.17]	**Community 4** [0.06]	**Community 5** [0.04]
Topic 1	Curb Your- Enthusiasm The Office: Series 2 The Office Special Monty Python's Flying Circus	Star Wars: EpIV The Incredibles The Princess Bride Lord of the Rings: The Two Towers	Gladiator The Shield Star Wars: EpIV Saving- Private Ryan	The Best- of Friends Friends: S6 Gilmore Girls Friends: S5	It's a- Wonderful Life Star Wars: EpIV Ben-Hur Gone with- the Wind
Interpr.:	**Comedy**	**Fantasy**	**Action, war**	**Sitcom**	**Classic**
Topic 2	Bruce Springsteen: Anthology 1978-2000 Karajan: Mozart: Don Giovanni Music of the Heart Music for Montserrat	Doctor Who: Pyramids of Mars Doctor Who: The Ribos Operation Battlestar Galactica Last Exile	Knowing Me- Knowing You Shag Aladdin Side Out	The Life and- Times of Frida Kahlo Birth of the- Blues / Blue Skies Julius Caesar American Dream	Blue's Clues: Shapes and Colors Yu-Gi-Oh! Sesame Street Black Beauty
Interpr.:	**Music**	**Sci-Fi**	**Comedy**	**Documentary**	**For children**
Topic 3	Glengarry- Glen Ross JFK Bataan Changing Lanes	Harry Connick Jr.: Only You Donna Summer: Live Ben Harper: Live Mozart: Don Giovanni	The Secret- Lives of Dentists Proof of Life The Ice Storm Body Story	Reservoir Dogs Get Shorty The Naked Gun 8MM	Gone in- 60 Seconds Intolerable Cruelty Confidence The Naked Gun
Interpr.:	**Drama**	**Live performances**	**Drama**	**Crime**	**Crime**
Topic 4	Highlander The Recruit Ali Rambo: First Blood	Robin Hood: Prince of Thieves Proof of Life Mission Impossible II Vanilla Sky	Equilibrium Ladder 49 Bad Company Waking Life	Amelie Victor / Victoria Princess Mononoke Sophie's Choice	A Midsummer- Night's Dream Chances Are Fools Rush In Mighty Aphrodite
Interpr.:	**Action**	**Action, famous actors**	**Thriller**	**Romance**	**Comedy, Fantasy**
Topic 5	Men in Black Alien Resurrection Spider-Man 2 X-Men: Evolution	Love Story Coffee and Cigarettes A Walk in the Clouds Hannah and- Her Sisters	13 Going on 30 Planet of the Apes Men in Black Rosemary- and Thyme	All the Pretty Horses Romeo Must Die Great Expectations The Manchurian- Candidate	The Parallax View Waterworld Romeo Must Die Swimming- with Sharks
Interpr.:	**Action, Sci-Fi**	**Drama, romance**	**Fantasy, Comedy**	**Drama**	**Thriller**

Table 5.3. User communities and relevant topics

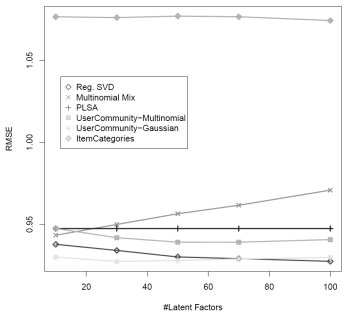


(a) User ratings

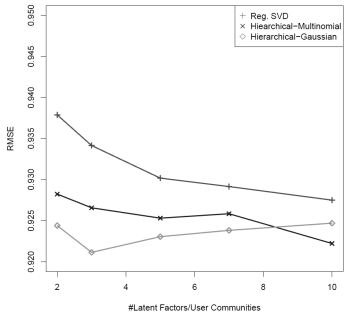


(b) Item ratings

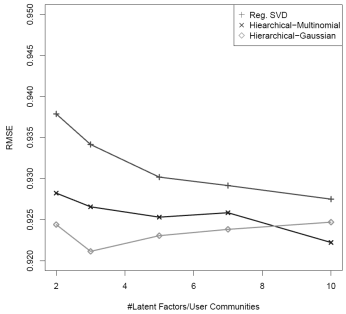
Fig. 5.4. ECDF for user and item ratings on NetFlix.



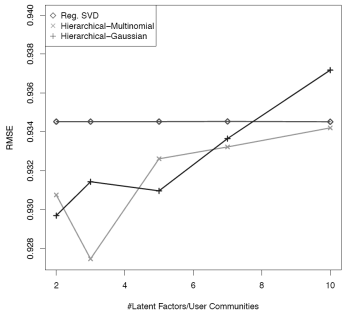
(a) Latent factors on Netflix



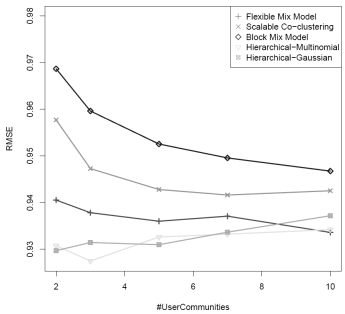
(b) Latent factors MovieLens



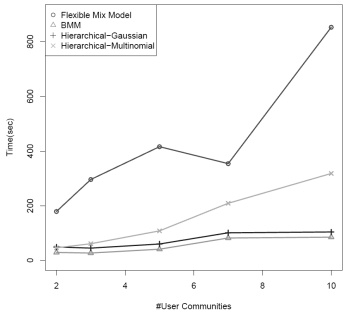
(c) Hierarchical approach on Netflix



(d) Hierarchical approach on Movielens



(e) Coclustering results on Movielens



(f) Execution times on Movielens

Fig. 5.5. Performance results.

6

Tree-based data mining

XML is a popular model for data representation, in which two main types of information coexist, i.e., pure content and structural information. The latter is a valuable support for information management, since it allows the definition of irregular structures explaining the nested content.

Conventional approaches to information handling are not meant for exploiting the structural information of XML data [42, 100], being either devoted to the management of highly structured data, such as relational databases, or too focused on the textual nature of the data, such as in the case of information retrieval techniques. From this perspective, XML data is a challenging research domain [39], that calls for the development of suitable methods for information handling.

The analysis of the structural information enables more effective processing of XML data, since it allows both to understand the spectrum of queries answerable by the available XML documents (i.e. the type of information as well as its organization) and to identify XML documents with similar structures as sources of the same type of contents.

In particular, clustering XML documents by their structural features is useful in several applicative contexts. For example, the detection of structural similarities among documents can help to recognize different sources providing the same kind of information [18]. Also, it can support the extraction of (schema or DTD) structures from a collection of XML documents [44, 82], by enabling the identification of more accurate structures from structurally-homogeneous subsets of the original collection. Yet, query processing can substantially benefit from the re-organization of the XML documents on the basis of their structure [41, 47].

The problem of clustering XML documents by structure has been extensively investigated, with the consequent development of several approaches, such as [39, 41, 46, 43, 47]. Catching structural resemblance between XML trees is a challenging research issue, that has an impact on both the effectiveness and efficiency of the clustering process. XML trees can share various forms of common structural components, ranging from simple node/edge and pairwise tags [56], to more complex substructures such as groups of siblings, paths (either root-to-node [56] or root-to-leaf [41]), as well as subtrees or even summaries [47, 46]. Therefore, if the addressed form

of structural patterns does not accord with the underlying properties of XML data, valuable relationships of structural resemblance between the XML documents can be missed, with a consequent degrade of clustering effectiveness. Moreover, judging differences only in terms of one type of structural components may not suffice to effectively separate the available XML documents. Therein, a careful investigation of the resulting clusters is likely to reveal an extent of intra-cluster inhomogeneity, that may be due to uncaught differences in the structures of the XML documents within the same clusters, ascribable to further unconsidered forms of structural patterns.

To address the aforesaid challenging issues, this thesis proposes a new hierarchical approach to clustering that considers various forms of structural patterns in the XML documents to progressively derive a hierarchy of nested clusters. The basic idea is that, at any level in the hierarchy, clusters are formed by grouping the XML documents by some form of structural patterns. Other forms of structural patterns are used to divide the resulting clusters into sub-clusters at the subsequent levels of the hierarchy, to highlight previously uncaught structural differences. In addition, the characterization of each cluster is accomplished by means of new summarization method, aimed at subsuming the structural properties within each cluster in terms of strongly representative substructures.

As a result, an experimental evaluation over both real-world and artificial XML data reveals that the quality of the attained results, in terms of effectiveness and cluster summarization, is on a par and even better than the quality of established competitors.

6.1 Preliminaries

The notation used throughout this chapter as well as some basic concepts are introduced. The structure of XML documents without references can be modeled in terms of *rooted ordered labeled trees*, that represent the hierarchical relationships among the document elements (i.e., nodes).

Definition 6.1. ***XML Tree****. An XML tree is a rooted, labeled, ordered tree, represented as a tuple* $\mathbf{t} = (r_{\mathbf{t}}, \mathbf{V_t}, \mathbf{E_t}, \lambda_{\mathbf{t}})$*, whose individual components have the following meaning.* $\mathbf{V_t} \subseteq \mathbb{N}$ *is a set of nodes and* $r_{\mathbf{t}} \in \mathbf{V_t}$ *is the root node of* $\mathbf{t}$*, i.e. the only node with no entering edges.* $\mathbf{E_t} \subseteq \mathbf{V_t} \times \mathbf{V_t}$ *is a set of edges, catching the parent-child relationships between nodes of* $\mathbf{t}$*. Finally,* $\lambda_t : \mathbf{V_t} \mapsto \Sigma$ *is a node labeling function and* Σ *is an alphabet of node tags (i.e., labels).* □

Let n_i and n_j be any two nodes from $\mathbf{V_t}$. If $(n_i, n_j) \in \mathbf{E_t}$ n_i is said to be the parent of n_j, which is instead a child of n_i. This type of parent-child structural relationship is represented as $n_i \prec n_j$. If there is a path of any positive length p from n_i to n_j, n_i is an ancestor of n_j, whereas n_j is a descendant of n_i. The ancestor-descendant structural relationship is indicated as $n_i \prec_p n_j$: clearly, if $p = 1$, the ancestor-descendant relationship reduces to the parent-child relationship. Nodes in $\mathbf{V_t}$ divide into two disjoint subsets: the set $\mathbf{L_t}$ of *leaves* and the set $\mathbf{V_t} - \mathbf{L_t}$ of

inner nodes. A leaf is a node with no children, whereas an inner node has at least one child. The children of each inner node are ordered [1]. No distinction is made between elements and tags of an original XML documents: both are mapped to nodes in the corresponding XML-tree representation.

Tree-like structures are also used to represent generic structural patterns occurring across a collection of XML trees (such as individual nodes, edges and root-to-leaf paths).

Definition 6.2. ***Substructure***. *Let* $\mathbf{t}$ *and* $\mathbf{s}$ *be two XML trees.* $\mathbf{s}$ *is said to be a substructure of* $\mathbf{t}$*, if there exists a total function* $\varphi : \mathbf{V_s} \rightarrow \mathbf{V_t}$*, that satisfies the following conditions for each* $n, n_i, n_j \in \mathbf{V_s}$*. Firstly,* $(n_i, n_j) \in \mathbf{E_s}$ *iff* $\varphi(n_i) \prec_p \varphi(n_j)$ *in* $\mathbf{t}$ *with* $p \geq 1$*. Secondly,* $\lambda_{\mathbf{s}}(n) = \lambda_{\mathbf{t}}[\varphi(n)]$. □

The mapping φ preserves node labels and hierarchical relationships. In this latter regard, depending on the value of p, two definitions of substructures can be distinguished. In the simplest case $p = 1$ and a substructure $\mathbf{s}$ is simply an *induced* tree pattern that matches a contiguous portion of $\mathbf{t}$, since φ maps the parent-child edges of $\mathbf{s}$ onto parent-child edges of t. This is indicated as $\mathbf{s} \sqsubseteq \mathbf{t}$. A refined definition follows when $p \geq 1$ [40, 107]. In such a case, $\mathbf{s}$ matches not necessarily contiguous portions of $\mathbf{t}$, since φ summarizes hierarchical relationships by mapping parent-child edges of $\mathbf{s}$ into either parent-child or ancestor-descendant edges of $\mathbf{t}$. This is denoted as $\mathbf{s} \subseteq \mathbf{t}$ and $\mathbf{s}$ is also said to be an *embedded* tree pattern of $\mathbf{t}$. The notion of embedded substructure (i.e., $\sqsubseteq$) is useful to catch structural patterns, that may be missed by exploiting the basic notion of induced substructure (i.e., $\subseteq$). Embedded substructures are exploited (in sec. 6.4.1) to mine representative substructures, that subsume a collection of XML trees with common (i.e., characteristic) structures intermixed with infrequent (i.e., unrepresentative) structures. Hereafter, the notions of substructure, (structural) component and tree pattern are used as synonyms.

Clustering by structure aims to divide a collection $\mathcal{D} = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$ of N XML trees in order to form a partition $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ of nonempty clusters $\mathcal{C}_i$, with $i = 1, \ldots, K$. The clustering process generally attempts to maximize the degree of structural resemblance exhibited by the XML trees in the same cluster and to minimize the extent of structural similarity between XML trees inside distinct clusters.

At the heart of the problem is the definition of structural resemblance. Ideally, the clustering process should take into account the most suitable forms of structural components for the specific applicative domain. Moreover, efficiency and scalability should not suffer from the number and the structural complexity of the chosen components, although more complex substructures could make clustering less efficient.

In order to accommodate the aforesaid requirements, a multi-stage clustering approach, that produces a hierarchy of nested clusters, is developed. The devised approach substantially differs from conventional hierarchical clustering in that a same basic partitioning algorithm is exploited at each stage i for further separating the individual clusters discovered at the previous stage $i-1$. Furthermore, cluster separation at stage i is performed with respect to some specific type of structural components, that has not been taken into account up to stage $i - 1$. Examples of structural components for the generic stage of clustering include:

1. The selection of one-node substructures modeling the individual nodes in the XML trees, i.e.,

$$\mathcal{T}^{(v)} \triangleq \{\mathbf{s}|\mathbf{V_s} = \{r_\mathbf{s}\}, \exists \mathbf{t} \in \mathcal{D}, \mathbf{s} \sqsubseteq \mathbf{t}\}$$

   In such a case, the generic substructure $\mathbf{s}$ consists only of its root $r_\mathbf{s}$, that matches some corresponding node of an XML tree $\mathbf{t}$ in $\mathcal{D}$.
2. The selection of one-edge substructures modeling the different parent-child edges in the XML trees, i.e.,

$$\mathcal{T}^{(e)} \triangleq \{\mathbf{s}|\mathbf{E_s} = \{(r_\mathbf{s}, n)\}, \exists \mathbf{t} \in \mathcal{D}, \mathbf{s} \sqsubseteq \mathbf{t}\}$$

   Here, the individual substructure $\mathbf{s}$ consists only of one edge $(r_\mathbf{s}, n)$, that matches some corresponding parent-child edge of an XML tree $\mathbf{t}$ in $\mathcal{D}$.
3. The selection of one-path substructures modeling the distinct root-to-node paths in the XML trees, i.e.,

$$\mathcal{T}^{(p)} \triangleq \{\mathbf{s}|\mathit{paths}^{(\mathbf{s})} = \{\langle r_\mathbf{s}, \ldots, n\rangle\}, \exists \mathbf{t} \in \mathcal{D}\mathbf{s} \sqsubseteq \mathbf{t}\}$$

   In the above, $\mathit{paths}^{\mathbf{s}}$ indicates the set of all all root-to-node paths of a substructure $\mathbf{s}$. Each $\mathbf{s}$ in $\mathcal{T}^{(p)}$ consists of only one such a path, matching some (root-to-node or root-to-leaf) path of an XML tree $\mathbf{t}$ in $\mathcal{D}$.

At any stage of clustering, the structural components enable the projection of the XML trees into a high-dimensional space, wherein the occurrence of the individual substructures within each XML tree is explicitly represented. More precisely, let $\mathcal{T}^{(i)}$ denote the collection of substructures selected at the generic stage i of clustering. The XML trees at this stage are modeled as transactions over a feature space $\mathcal{S}^{(i)} \triangleq \{\mathcal{F}_\mathbf{s}|\mathbf{s} \in \mathcal{T}^{(i)}\}$. Here, the generic feature $\mathcal{F}_\mathbf{s}$ is a boolean attribute, whose value indicates the presence/absence of the corresponding component $\mathbf{s}$ of $\mathcal{T}^{(i)}$ within the individual XML trees.

Assume that $\mathbf{x}^{(\mathbf{t})}$ is the high-dimensional representation of an XML tree $\mathbf{t}$. Also, let $\mathbf{x}^{(\mathbf{t})}[\mathcal{F}_\mathbf{s}]$ be the value assumed by $\mathcal{F}_\mathbf{s}$ in the context of $\mathbf{x}^{(\mathbf{t})}$. $\mathbf{x}^{(\mathbf{t})}[\mathcal{F}_\mathbf{s}]$ is true if $\mathbf{s} \sqsubseteq \mathbf{t}$, otherwise it is false. Hence, $\mathbf{x}^{(\mathbf{t})}$ can be modeled as a proper subset of $\mathcal{S}^{(i)}$, namely $\mathbf{x}^{(\mathbf{t})} \triangleq \{\mathcal{F}_\mathbf{s} \in \mathcal{S}^{(i)}|\mathbf{s} \sqsubseteq \mathbf{t}\}$, with the meaning that the features explicitly present in $\mathbf{x}^{(\mathbf{t})}$ take value true, whereas the others assume value false. In the resulting feature space $\mathcal{S}^{(i)}$, the cost for testing the presence of the selected components within the individual XML trees is independent of the structural complexity of the same components. However, the transactional representation over $\mathcal{S}^{(i)}$ involves the non-trivial discovery of meaningful clusters in large-scale databases of high-dimensional transactions.

From this perspective, the proposed approach reformulates the original problem of grouping $\mathcal{D}$ by structure as that of progressively projecting $\mathcal{D}$ into high-dimensional representations $\mathcal{D}^{(i)} = \{\mathbf{x}^{(\mathbf{t})} \subseteq \mathcal{S}^{(i)}|\mathbf{t} \in \mathcal{D}\}$ and then to exploit such

representations throughout the various stages of clustering. As it has been anticipated, this is accomplished by forming a hierarchy of nested clusters rooted at $\mathcal{D}^{(1)}$. In this hierarchy, the structural homogeneity of each parent cluster $\mathcal{C}^{(i-1)}$ resulting at the end of stage $i-1$ (over a feature space $\mathcal{S}^{(i-1)}$) is refined by separating $\mathcal{C}^{(i-1)}$ into an appropriate number k of child clusters $\mathcal{C}_1^{(i)}, \ldots, \mathcal{C}_k^{(i)}$ (with respect to another feature space $\mathcal{S}^{(i)}$) at the end of stage i.

6.2 Partitioning XML Trees

At a given stage i, finding clusters in the high-dimensional feature space $\mathcal{S}^{(i)}$ is a challenging issue for various reasons [28]. Foremost, transactions tend to form different clusters on distinct subsets of features, which penalizes the effectiveness of clustering and exacerbates its time requirements. Also, poor scalability in both the size and the dimensionality of the transactions is usually a major limitation. Yet, an underestimation (resp. overestimation) of the number K of child clusters for a parent cluster misses (resp. uncovers) actual (resp. artificial) groups in the XML data.

To best fit the peculiarities of the transactional setting at each stage of clustering, the XML trees are partitioned by means of the AT-DC algorithm [28]. The latter is an effective and parameter-free method for transactional clustering, that autonomously partitions each parent cluster into a natural number k of child clusters by isolating groups of transactions with meaningful and discriminatory co-occurrences of structural features. AT-DC was shown to outperform several established competitors. It achieves nearly-optimal results in terms of cluster homogeneity, compactness and separation and, also, linearly scales with respect to both the size and the dimensionality of the underlying transactions. These properties make AT-DC an ideal basic partitioning scheme to be used in the design of an overall multi-stage clustering process as discussed in the next section.

AT-DC implements a top-down divisive clustering approach. The algorithm initially maps the original set $\mathcal{D}$ of XML trees to a space of clustering features in $\mathcal{S}^{(i)}$. This yields the transactional representation $\mathcal{D}^{(i)}$. The algorithm hence starts with a partition $\mathcal{P}$ containing a single cluster corresponding to the whole transactional dataset. The core of the algorithm is a loop, where an attempt to generate a new cluster is performed by (*i*) choosing a candidate node (corresponding to a cluster with low quality); (*ii*) splitting the candidate cluster into two subclusters; and (*iii*) evaluating whether the splitting allows a new partition exhibits better quality than the original partition. If this is the case, the partition is updated, by replacing the candidate cluster with the new subclusters. Viceversa, subclusters are discarded and a new candidate cluster is considered for splitting.

The core of the AT-DC algorithm is the definition of local cluster quality: ideally, a cluster exhibits a good quality if it there is a core subset of relevant features. Thus, it is possible to measure the gain in feature strength with respect to the whole dataset, i.e.,

$$Quality(\mathcal{C}) \propto \sum_{\mathcal{F} \in \mathcal{S}^{(i)}} \left[\Pr(\mathcal{F}|\mathcal{C})^2 - \Pr(\mathcal{F}|\mathcal{D})^2 \right] \tag{6.1}$$

where $\Pr(\mathcal{F}|\mathcal{C})^2$ corresponds to the relative strength of $\mathcal{F}$ within $\mathcal{C}$. Features with a high occurrence frequency, compared to the occurrence frequency in the original dataset, define a subset of relevant features, as opposed to low-occurrence features, that are irrelevant for the purpose of clustering.

AT-DC is embedded in the main clustering procedure, as reported in fig. 7. The latter consists of m stages of clustering (line 1). The end user incorporates (at line 1) valuable domain knowledge and application semantics into the clustering process, by establishing the most appropriate set of structural features $\mathcal{S}^{(i)}$ for each stage as well as the overall number m of stages.

The generic stage i (lines 4-19) consists of two phases: cluster separation and summarization. Cluster separation exploits AT-DC to divide the individual clusters belonging to the current partition $\mathcal{P}$ with respect to the feature space $\mathcal{S}^{(i)}$ (lines 5 - 13). At the beginning, i.e. when $i = 1$, the current partition $\mathcal{P}$ includes a single cluster, which coincides with the whole dataset $\mathcal{D}$ of XML trees (line 3). The partition $\mathcal{P}^{(i)}$ resulting at the end of stage i (line 13) is itself a collection of partitions. More precisely, at the current stage i, each parent cluster $\mathcal{C}$ from $\mathcal{P}^{(i-1)}$ is divided into an appropriate number of child clusters (line 8), which together form the partition $\mathcal{R}$ of the aforesaid $\mathcal{C}$. At this point, each child cluster $\mathcal{C}'$ in $\mathcal{R}$ is associated (lines 9-11) with its siblings $\overline{\mathcal{C}'} \triangleq \mathcal{R} - \mathcal{C}$ (for the cluster summarization purpose) and $\mathcal{R}$ is then added to the ongoing $\mathcal{P}^{(i)}$.

Cluster summarization (lines 14-16) is applied to each cluster $\mathcal{C}$ from the obtained $\mathcal{P}^{(i)}$. It consists of a procedure, discussed in section 6.4, which associates $\mathcal{C}$ with a set $Rep(\mathcal{C})$ of representative substructures, that subsume the structural information within $\mathcal{C}$. $\mathcal{P}^{(i)}$ becomes (at line 17) the current partition $\mathcal{P}$ for the subsequent stage $i+1$. At this stage, AT-DC is re-applied to further divide every cluster $\mathcal{C} \in \mathcal{P}^{(i)}$ with respect to another set of structural features, i.e., $\mathcal{S}^{(i+1)}$.

The choice of a distinct feature space at each stage guarantees a progressively increasing degree of structural homogeneity, since the XML trees (corresponding to the transactions) within the generic cluster of $\mathcal{P}^{(i)}$ (that are already homogeneous according to the so far considered sets of features $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(i)}$) can still be separated by isolating groups of such XML trees, which are strongly discriminated by meaningful co-occurrences of the (previously unconsidered) structural patterns relative to $\mathcal{S}^{(i+1)}$. Obviously, this also implies a significant differentiation in the representatives of clusters at different stages. Indeed, at each distinct stage, representatives provide a summarization of the tree structures within the corresponding clusters in terms of (a combination of) the structural features considered a that particular stage. Hence, the representative of a subcluster highlights local patterns of structural homogeneity, that are not caught by the representative of the parent cluster.

Algorithm 7 Generate-Hierarchy($\mathcal{D}$)

Input: a set $\mathcal{D} = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$ of XML trees;
Output: a set $\cup_i \mathcal{P}^{(i)}$ of multiple cluster partitions;
1: let $\mathcal{S}^{(i)}$ be the set of features at stage i, with $i = 1, \ldots, m$;
2: let $i \leftarrow 1$;
3: let $\mathcal{P} \leftarrow \{\mathcal{D}\}$;
4: **while** $i \leq m$ **do**
5: **while** $\mathcal{P} \neq \emptyset$ **do**
6: let $\mathcal{C}$ be a cluster in $\mathcal{P}$;
7: $\mathcal{P} \leftarrow \mathcal{P} - \mathcal{C}$;
8: $\mathcal{R} \leftarrow$ *Generate-Clusters*$(\mathcal{C}, \mathcal{S}^{(i)})$;
9: **for each** $\mathcal{C}' \in \mathcal{R}$ **do**
10: let $\overline{\mathcal{C}'} \leftarrow \mathcal{R} - \{\mathcal{C}'\}$ be the set of siblings of $\mathcal{C}'$;
11: **end for**
12: $\mathcal{P}^{(i)} \leftarrow \mathcal{P}^{(i)} \cup \mathcal{R}$;
13: **end while**
14: **for each** $\mathcal{C} \in \mathcal{P}^{(i)}$ **do**
15: $Rep(\mathcal{C}) \leftarrow$ *MineRep*$(\mathcal{C}, \overline{\mathcal{C}}, \alpha)$;
16: **end for**
17: $\mathcal{P} \leftarrow \mathcal{P}^{(i)}$;
18: $i \leftarrow i + 1$;
19: **end while**
20: RETURN $\cup_i \mathcal{P}^{(i)}$;

6.3 The AT-DC Algorithm

In this section, is shown review of the fundamentals of the AT-DC algorithm and show how it can be exploited to develop a cluster hierarchy. A discussion on the convergence of AT-DC together with a comparative analysis of its empirical behavior against a wide variety of established competitors can be found in [28].

6.3.1 The Basic Partitioning Method

The key idea behind AT-DC is to develop a clustering procedure which resembles the general schema of a top-down decision-tree learning algorithm. It starts from an initial partition containing a single cluster (representing the whole dataset), and then iteratively tries to split a cluster within the partition into two subclusters. If subclusters guarantee a higher homogeneity in the partition than the original cluster, the latter is removed and the outcome of splitting is added to the partition. The approach is based on the capability of splitting clusters on the basis of their homogeneity. Let assume that a function $Quality(\mathcal{C})$ measures the degree of homogeneity of a cluster $\mathcal{C}$. In practice, clusters with high intra-homogeneity exhibit high values of $Quality$.

The general schema of the AT-DC algorithm is specified in alg. 8. AT-DC initially maps (lines 1- 2) the original set $\mathcal{D}$ of XML trees to a space of clustering features in $\mathcal{S}$. This yields the transactional representation $\mathcal{D}'$. The algorithm starts with a partition $\mathcal{P}$ containing a single cluster corresponding to the whole transactional dataset

Algorithm 8 Generate-Clusters($\mathcal{D}, \mathcal{S}$)

Input: A set $\mathcal{D} = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$ of XML trees;
a set of clustering features $\mathcal{S}$;
Output: A partition $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of clusters;
1: let $\mathbf{x}^{(\mathbf{t}_i)} \leftarrow \{\mathcal{F}_\mathbf{s} \in \mathcal{S} | \mathbf{s} \sqsubseteq \mathbf{t}_i\}$ for each $i = 1, \ldots, N$;
2: let $\mathcal{D}' \leftarrow \{\mathbf{x}^{(\mathbf{t}_i)} \subseteq \mathbf{S} | \mathbf{t}_i \in \mathcal{D}\}$;
3: let $\mathcal{P} \leftarrow \{\mathcal{D}'\}$;
4: **repeat**
5: Generate a new cluster $\mathcal{C}$ initially empty;
6: **for each** cluster $\mathcal{C}_i \in \mathcal{P}$ **do**
7: `Partition-Cluster`($\mathcal{C}_i, \mathcal{C}$);
8: $\mathcal{P}' \leftarrow \mathcal{P} \cup \{\mathcal{C}\}$;
9: **if** $Quality(\mathcal{P}) < Quality(\mathcal{P}')$ **then**
10: $\mathcal{P} \leftarrow \mathcal{P}'$;
11: `Stabilize-Clusters`($\mathcal{P}$);
12: **break**
13: **else**
14: Restore all $\mathbf{x}^{(\mathbf{t}_j)} \in \mathcal{C}$ into $\mathcal{C}_i$;
15: **end if**
16: **end for**
17: **until** no further cluster $\mathcal{C}$ can be generated

(line 3). The core of the algorithm is the body of the loop between lines 4-17. Within the loop, an attempt to generate a new cluster is performed by (*i*) choosing a candidate node (corresponding to a cluster with low quality) to split (line 6); (*ii*) splitting the candidate cluster into two subclusters (line 7); and (*iii*) evaluating whether the splitting allows a new partition exhibits better quality than the original partition (lines 8-15). If this is the case, the loop can be stopped (line 12) and the partition is updated, by replacing the candidate cluster with the new subclusters (line 10). Viceversa, subclusters are discarded and a new candidate cluster is considered for splitting.

The PARTITION-CLUSTER procedure at line 7 iteratively evaluates, for each transaction $\mathbf{x}^{(\mathbf{t})} \in \mathcal{C}_i \cup \mathcal{C}$, whether a membership reassignment improves the degree of local homogeneity of the two clusters. The contribution of $\mathbf{x}^{(\mathbf{t})}$ to the local quality is evaluated in two cases: both in the case that $\mathbf{x}^{(\mathbf{t})}$ is maintained in its original cluster of membership and in the case that $\mathbf{x}^{(\mathbf{t})}$ is moved to the other cluster. If moving $\mathbf{x}^{(\mathbf{t})}$ causes an improvement in the local quality, then the swap is accepted.

Differently from the PARTITION-CLUSTER procedure, where the improvement in quality is attempted locally to a cluster, the STABILIZE-CLUSTERS procedure tries to increase global quality, by finding, for each element, the most suitable cluster among the ones available in the partition. Precisely, the generation of a new cluster triggers the call to STABILIZE-CLUSTERS on line 11, which aims at further improving the overall quality by attempting relocations among the the clusters. Also, clusters at line 6 are considered in increasing order of quality. This guarantees that the effects of splitting are evaluated first on clusters with lower quality. Indeed, if a

cluster exhibits a lower degree of homogeneity, it is more eligible for producing an improvement in the overall quality, provided that it is properly split.

6.3.2 Cluster and Partition Qualities

AT-DC exploits two different quality measures, namely one for the local homogeneity within a cluster and the other for the global homogeneity of the partition. These measures are employed with opposed objectives. Indeed, as shown in alg. 8, it can be noticed that partition quality is used to establish whether the insertion of a new cluster is really convenient: in a sense, it is aimed at maintaining compactness. By the converse, cluster quality in procedure PARTITION-CLUSTER at line 7 of alg. 8 is aimed at the best localized splitting and, hence, at a good separation.

Let $\mathcal{S}_\mathcal{C}$ be the subset of features that appear within the generic cluster $\mathcal{C}$, i.e., $\mathcal{S}_\mathcal{C} \triangleq \{\mathcal{F} | \mathcal{F} \in \mathbf{x}^{(\mathbf{t})}, \mathbf{x}^{(\mathbf{t})} \in \mathcal{C}\}$. The quality of $\mathcal{C}$ is defined as the gain in feature strength with respect to the whole dataset, i.e.,

$$Quality(\mathcal{C}) = \Pr(\mathcal{C}) \sum_{\mathcal{F} \in \mathcal{S}_\mathcal{C}} \left[\Pr(\mathcal{F}|\mathcal{C})^2 - \Pr(\mathcal{F}|\mathcal{D})^2\right]$$

where $\Pr(\mathcal{F}|\mathcal{C})^2$ corresponds to the relative strength of $\mathcal{F}$ within $\mathcal{C}$, whereas $\Pr(\mathcal{C})$ represents the relative strength of $\mathcal{C}$. These two factors work in contraposition: singleton clusters exhibit strong features in a sparse region, whereas highly populated clusters exhibit weaker features in a dense region. The above formula finds an interpretation in terms of subspace clustering. Features exhibiting a high occurrence frequency compared to the occurrence frequency in the original dataset, define a subset of relevant features, as opposed to low-occurrence features which are indeed irrelevant for the purpose of clustering. Thus, clusters exhibit high quality whenever a subspace of relevant features occurs whose frequency is significantly higher than in the whole dataset.

The quality of a partition $\mathcal{P}$ is instead meant to measure both the homogeneity of clusters and their compactness. Viewed in this respect, partition quality is defined as the weighted sum of the qualities of the available clusters:

$$Quality(\mathcal{P}) = \sum_{\mathcal{C} \in \mathcal{P}} \Pr(\mathcal{C})\, Quality(\mathcal{C})$$

The formula finds an interpretation in terms of average increase in quality obtained by partitioning the dataset. Notice that the component $Quality(\mathcal{C})$ is already proportional to the contribution $\Pr(\mathcal{C})$. As a result, in the overall partition quality, the contribution of each cluster is weighted by $\Pr(\mathcal{C})^2$. This weighting has a major effect in the GENERATE-CLUSTERS procedure: splitting in extremely small clusters is penalized. Indeed, the generated clusters are added to the partition only if their contribution is really worth. Notice the different role of the contribution $\Pr(\mathcal{C})$ in the two quality measures: a strong penalization on singleton clusters would not allow a proper splitting in the PARTITION-CLUSTER procedure. In particular, splitting would suffer from the bottleneck of the initial reassignment, since in principle the

possible loss in cluster $\mathcal{C}_1$ would not be counterbalanced by a proper gain in $\mathcal{C}_2$. On the contrary, the (stable) result of the PARTITION-CLUSTER procedure has to be accepted only if it yields a significant change in the average cluster quality.

6.4 Cluster Summarization

The representative of a cluster of XML trees is modeled as a set of highly representative tree patterns, which provide an intelligible summarization of the most relevant structural properties in the cluster. Notice that, as mentioned before, a cluster is already characterized by a set of relevant features. However, features can be combined further, and they do not necessarily allow to distinguish among different clusters.

A set of tree patterns is actually viewed as the representative of a cluster of XML trees, if the following two conditions are satisfied. Firstly, each tree pattern must appear as a substructure of the XML trees in the cluster with an occurrence frequency that is much higher than the one with which it occurs throughout the whole collection of XML trees. Secondly, there must be a strong degree of correlation between the individual tree pattern and the cluster of XML trees, which guarantees that the former is an actual summarization of specific structural properties of the latter.

An XML tree pattern is essentially a substructure that catches some common structural properties of a collection of XML trees. The features in $\mathcal{S}^{(i)}$ are considered as the basic tree pattens within a cluster. However, tree patterns can be suitably combined into *composite* tree patterns. To avoid combinatory explosion, only two types of composite tree pattern are admitted, namely *parent-child* and *sibling* tree patterns.

Definition 6.3. ***Parent-child tree pattern***. *A parent-child tree pattern is an arrangement of two basic tree patterns, in which one of the two tree patterns is rooted at some leaf node of the other tree pattern. Let* $\mathbf{s}_i$ *and* $\mathbf{s}_j$ *be two generic tree patterns. Also, assume that* l *is some leaf node of* $\mathbf{s}_i$*. The operator* $\mathbf{s}_i \lhd_l \mathbf{s}_j$ *defines a new parent-child tree pattern* $\mathbf{s}$*, such that* $|\mathbf{V}_\mathbf{s}| = |\mathbf{V}_{\mathbf{s}_i}| + |\mathbf{V}_{\mathbf{s}_j}|$ *and* $|\mathbf{E}_\mathbf{s}| = |\mathbf{E}_{\mathbf{s}_i}| + |\mathbf{E}_{\mathbf{s}_j}| + 1$*, wherein the root* $r_{\mathbf{s}_j}$ *of* $\mathbf{s}_j$ *is a child of* l*. Formally,* $\mathbf{s}_i \lhd_l \mathbf{s}_j$ *defines a tree pattern* $\mathbf{s}$ *such that there exist two mappings* $\varphi_i : \mathbf{V}_{\mathbf{s}_i} \to \mathbf{V}_\mathbf{s}$ *and* $\varphi_j : \mathbf{V}_{\mathbf{s}_j} \to \mathbf{V}_\mathbf{s}$ *satisfying the following conditions:*

- $\varphi_i(r_{\mathbf{s}_i}) = r_\mathbf{s}$ *(i.e.* $r_{\mathbf{s}_i}$ *matches* $r_\mathbf{s}$*)*
- $\forall n \in \mathbf{V}_{\mathbf{s}_i}$ *and* $\forall n' \in \mathbf{V}_{\mathbf{s}_j}$*, it holds that* $\varphi_i(n) \neq \varphi_j(n')$*;*
- $\forall n \in \mathbf{V}_{\mathbf{s}_h}$*,* $\lambda_{\mathbf{s}_h}(n) = \lambda_\mathbf{s}(\varphi_h(n))$ *for each* $h \in \{i, j\}$*;*
- $\forall n, n' \in \mathbf{V}_{\mathbf{s}_h}$*, it holds that* $(n, n') \in \mathbf{E}_{\mathbf{s}_h}$ *iff* $(\varphi_h(n), \varphi_h(n')) \in \mathbf{E}_\mathbf{s}$ *for each* $h \in \{i, j\}$*;*
- $(\varphi_i(l), \varphi_j(r_{\mathbf{s}_j})) \in \mathbf{E}_\mathbf{s}$ *(i.e.,* $\varphi_i(l) \prec \varphi_j(r_{\mathbf{s}_j})$ *in* $\mathbf{s}$*);*

Given any two tree patterns $\mathbf{s}_i$ *and* $\mathbf{s}_j$*, the set of all possible parent-child tree patterns in which the root of* $\mathbf{s}_j$ *is a child of the individual leaves of* $\mathbf{s}_i$ *is denoted as*

$$\mathbf{s}_i \lhd \mathbf{s}_j \triangleq \bigcup_{l \in \mathbf{L}_{\mathbf{s}_i}} \{\mathbf{s}_i \lhd_l \mathbf{s}_j\}$$

where $\mathbf{L}_{\mathbf{s}_i}$ *represents the set of leaves of* $\mathbf{s}_i$. □

A parent-child tree pattern is a vertical arrangement of two component tree patterns. Instead, a sibling tree pattern follows from an horizontal arrangement of its components.

Definition 6.4. ***Sibling tree pattern***. *Given two tree patterns with a same label at their roots, a sibling tree pattern is a composite structure, whose root-to-leaf paths are the union of the root-to-leaf paths in the two component patterns. Let* $\mathbf{s}_i$ *and* $\mathbf{s}_j$ *be two tree patterns such that* $\lambda_{\mathbf{s}_i}(r_{\mathbf{s}_i}) = \lambda_{\mathbf{s}_j}(r_{\mathbf{s}_j})$. *The operator* $\mathbf{s}_i \wedge \mathbf{s}_j$ *defines a sibling tree pattern* $\mathbf{s}$, *such that there exist two mappings* $\varphi_i : \mathbf{V}_{\mathbf{s}_i} \rightarrow \mathbf{V}_{\mathbf{s}}$ *and* $\varphi_j : \mathbf{V}_{\mathbf{s}_j} \rightarrow \mathbf{V}_{\mathbf{s}}$ *satisfying the following conditions:*

- $\varphi_i(r_{\mathbf{s}_i}) = \varphi_j(r_{\mathbf{s}_j}) = r_{\mathbf{s}}$;
- $\forall n \in \mathbf{V}_{\mathbf{s}_h}$, $\lambda_{\mathbf{s}_h}(n) = \lambda_{\mathbf{s}}(\varphi_h(n))$ *for each* $h \in \{i, j\}$;
- $\forall n, n' \in \mathbf{V}_{\mathbf{s}_h}$, *it holds that* $(n, n') \in \mathbf{E}_{\mathbf{s}_h}$ *iff* $(\varphi_h(n), \varphi_h(n')) \in \mathbf{E}_{\mathbf{s}}$ *for each* $h \in \{i, j\}$. □

6.4.1 Mining Representative XML Tree Patterns

The MINEREP procedure, illustrated in fig. 9, is an Apriori-based technique to mine a representative for a cluster $\mathcal{C}$ of XML trees. The latter is a set of discriminatory substructures, obtained through progressive combinations of the elementary structures in $\mathcal{C}$.

MINEREP receives three input parameters, namely the cluster $\mathcal{C}$ to be summarized, the set $\overline{\mathcal{C}}$ of all siblings of $\mathcal{C}$ (as defined at line 10 of fig. 7) and a significance threshold α. The procedure starts (at line 3) by considering a space $\mathcal{S}_{\mathcal{C}}$ of features, whose occurrence frequency in cluster $\mathcal{C}$ is higher than in the whole partition $\mathcal{C} \cup \overline{\mathcal{C}}$. These features are inherently characteristic of $\mathcal{C}$ and, according to the definition of cluster quality in (6.1), are directly provided by AT-DC without having to be recomputed. It is worth to emphasize that focusing on such characteristic features from the beginning strongly prunes the search space into which to search for progressively more-complex candidates.

The elementary structures from the feature space $\mathcal{S}_{\mathcal{C}}$ are considered (at line 4) as candidate tree patterns. Each such a candidate $\mathbf{s}$ is associated (lines 5-7) with a bitmap $\mathcal{B}(\mathbf{s})$, that keeps trace of the transactions from $\mathcal{C} \cup \overline{\mathcal{C}}$ exhibiting the corresponding feature $\mathcal{F}_{\mathbf{s}}$. Notice that, in fig. 9, the generic bitmap $\mathcal{B}(\mathbf{s})$ is represented as a set for clarity. The interpretation is that the transactions explicitly present in $\mathcal{B}(\mathbf{s})$ exhibit $\mathcal{F}_{\mathbf{s}}$, whereas all others do not include $\mathcal{F}_{\mathbf{s}}$. Henceforth, the bitmaps associated to the elementary structures allow to obtain the bitmap of any combination of such structures, without scanning the transactions in $\mathcal{C} \cup \overline{\mathcal{C}}$ again. This is accomplished by intersecting the bitmaps associated to the combined substructures (line 4 of the CANDIDATE-GENERATION sub-procedure in fig. 10).

At the heart of MINEREP is a loop (lines 8-20), whose purpose is twofold: distilling representative tree patterns from $C^{(k)}$ and generating more-complex (parent-child or sibling) candidates through structural combinations of representative tree patterns. The loop is reiterated until no more candidates from which to distill representative tree patterns.

At any generic iteration k, MINEREP computes the occurrence frequencies $\Pr(\mathbf{s}|\mathcal{C})$ and $\Pr(\mathbf{s}|\overline{\mathcal{C}})$ of each candidate $\mathbf{s}$ from $C^{(k)}$ within, respectively, $\mathcal{C}$ and $\overline{\mathcal{C}}$. This is accomplished by looking at the transactions in the associated bitmap $\mathcal{B}(\mathbf{s})$. The latter, by construction (at line 4 of the CANDIDATE-GENERATION scheme in fig. 10), includes all those transactions in $\mathcal{C} \cup \overline{\mathcal{C}}$ that exhibit all and only the features in $\mathbf{s}$. However, at any iteration $k \geq 2$ (tested at line 10), such transactions cannot be directly exploited to compute the occurrence frequencies. This is due to the fact that $\mathbf{s}$ is not necessarily an embedded substructure of all the XML trees associated to the transactions in $\mathbf{s}$, because of the possibility that, in some of these trees, the features of $\mathbf{s}$ originate distinct structural combinations. Therefore, $\mathcal{B}(\mathbf{s})$ is inspected for the purpose of identifying (at line 11) the subset $\mathcal{D}(\mathbf{s})$ of those transactions, whose corresponding XML trees do not actually contain $\mathbf{s}$ as a substructure. $\mathcal{D}(\mathbf{s})$ is then removed from $\mathcal{B}(\mathbf{s})$ (at line 12), which enables the computation of the occurrence frequencies $\Pr(\mathbf{s}|\mathcal{C})$ and $\Pr(\mathbf{s}|\overline{\mathcal{C}})$ (at lines 14 and 15) from the bitmap.

As a matter of fact, the identification of $\mathcal{D}(\mathbf{s})$ is optimized. Precisely, the available XML trees are individually associated with revised s-graphs [47]. These are explicit representations of the parent-child and ancestor-descendant hierarchical relationships within the XML trees. Such representations are used to avoid the time-expensive test (at line 11) on the embedding of a substructure $\mathbf{s}$ in an XML tree $\mathbf{t}$, whenever all edges of $\mathbf{s}$ are not included in the revised s-graph associated to $\mathbf{t}$.

The representative tree patterns are distilled in $L^{(k)}$ (at line 17) from the set $C^{(k)}$ of candidates, by choosing the ones that satisfy the following two conditions. Firstly, the occurrence frequency of each representative tree pattern $\mathbf{s}$ must be higher in $\mathcal{C}$ than in the whole partition to which $\mathcal{C}$ belongs, namely $\Pr(\mathbf{s}|\mathcal{C}) > \Pr(\mathbf{s}|\overline{\mathcal{C}})$. Both $\Pr(\mathbf{s}|\mathcal{C})$ and $\Pr(\mathbf{s}|\overline{\mathcal{C}})$ are computed (respectively at lines 14 and 15) after the pruning phase. Secondly, there must be a strong degree of correlation between $\mathbf{s}$ and $\mathcal{C}$. This is useful to establish whether the occurrence of $\mathbf{s}$ in $\mathcal{C}$ is statistically relevant and, hence, structurally representative. Statistical hypothesis testing is used for this purpose, as it shall be discussed hereafter. The resulting set $L^{(k)}$ of representative tree patterns provides the basic structures for the generation of candidate tree patterns at the subsequent iteration $k+1$.

MINEREP halts when $C^{(k)}$ is empty and, hence, no more representative tree patterns can be discovered. In such a case MINEREP returns (line 22) all of the XML tree patterns, that were found in the previous steps to be strongly discriminatory of the structural properties of cluster $\mathcal{C}$. The candidate generation phase as well as the exploitation of statistical hypothesis testing for the identification of representative substructures are analyzed next.

Algorithm 9 MineRep($\mathcal{C}, \overline{\mathcal{C}}, \alpha$)

Input: a set $\mathcal{C} = \{\mathbf{x}^{(\mathbf{t}_1)}, \ldots, \mathbf{x}^{(\mathbf{t}_h)}\}$ of XML trees;
the set $\overline{\mathcal{C}}$ of siblings of $\mathcal{C}$ as defined at line 10 of fig. 7;
a significance threshold α;
Output: a set $\mathcal{R}$ of representative XML structures;

1: $\mathcal{R} \leftarrow \emptyset$;
2: $k \leftarrow 1$;
3: let $\mathcal{S}_\mathcal{C} \leftarrow \{\mathcal{F}_\mathbf{s} | \exists \mathbf{x}^{(\mathbf{t})} \in \mathcal{C}, \mathcal{F}_\mathbf{s} \in \mathbf{x}^{(\mathbf{t})}, \Pr(\mathcal{F}_\mathbf{s}|\mathcal{C}) > \Pr(\mathcal{F}_\mathbf{s}|\mathcal{C} \cup \overline{\mathcal{C}})\}$;
4: let $C^{(k)} \leftarrow \{\mathbf{s} | \mathcal{F}_\mathbf{s} \in \mathcal{S}_\mathcal{C}\}$;
5: **for each** $\mathbf{s} \in C^{(k)}$ **do**
6: let $\mathcal{B}(\mathbf{s}) \leftarrow \{\mathbf{x}^{(\mathbf{t})} \in \mathcal{C} \cup \overline{\mathcal{C}} | \mathcal{F}_\mathbf{s} \in \mathbf{x}^{(\mathbf{t})}\}$;
7: **end for**
8: **while** $(C^{(k)} \neq \emptyset)$ **do**
9: **for each** $\mathbf{s} \in C^{(k)}$ **do**
10: **if** $(k > 1)$ **then**
11: $\mathcal{D}(\mathbf{s}) \leftarrow \{\mathbf{x}^{(\mathbf{t})} \in \mathcal{B}(\mathbf{s}) | \mathbf{s} \not\subseteq \mathbf{t}\}$;
12: $\mathcal{B}(\mathbf{s}) \leftarrow \mathcal{B}(\mathbf{s}) - \mathcal{D}(\mathbf{s})$;
13: **end if**
14: $\Pr(\mathbf{s}|\mathcal{C}) \leftarrow \frac{|\{\mathbf{x}^{(\mathbf{t})} \in \mathcal{B}(\mathbf{s}) \cap \mathcal{C}\}|}{|\mathcal{C}|}$;
15: $\Pr(\mathbf{s}|\mathcal{C} \cup \overline{\mathcal{C}}) \leftarrow \frac{|\{\mathbf{x}^{(\mathbf{t})} \in \mathcal{B}(\mathbf{s})\}|}{|\mathcal{C} \cup \overline{\mathcal{C}}|}$;
16: **end for**
17: $L^{(k)} \leftarrow \{\mathbf{s} \in C^{(k)} | \Pr(\mathbf{s}|\mathcal{C}) > \Pr(\mathbf{s}|\mathcal{C} \cup \overline{\mathcal{C}}), \chi^2(\mathbf{s}, \mathcal{C}) > \tau_\alpha\}$;
18: $k \leftarrow k + 1$;
19: $C^{(k)} \leftarrow$ *Candidate-Generation*$(L^{(k-1)})$;
20: **end while**
21: $\mathcal{R} \leftarrow \cup_k L^{(k)}$;
22: **return** $\mathcal{R}$;

Candidate generation

The CANDIDATE-GENERATION sub-procedure is fed with the current set L of frequent and discriminative tree patterns. CANDIDATE-GENERATION is a tree-pattern growth strategy, that considers each pair of distinct tree patterns $\mathbf{s}$ and $\mathbf{s}'$ from L for combination into further candidate (parent-child or sibling) tree patterns. Precisely, the set T contains all possible parent-child tree patterns obtainable from $\mathbf{s}$ and $\mathbf{s}'$. The bitmap $\mathcal{B}$ (at line 4) is common to all of the parent-child tree patterns in T. The candidate generation strategy soon prunes T (lines 5-11): the height of each candidate $\mathbf{s}$ in T, denoted as *height*$(\mathbf{s})$, is tested (lines 6-10) against the maximum height H of the XML trees in cluster $\mathcal{C}$. If *height*$(\mathbf{s})$ does not to exceed H, $\mathbf{s}$ is left in T and it is associated with the bitmap $\mathcal{B}$ (at line 7). Otherwise, $\mathbf{s}$ is removed from T (at line 9). The distilled T is added to the ongoing set C of candidates (at line 12). At this point, CANDIDATE-GENERATION considers the sibling pattern obtainable from $\mathbf{s}_i$ and $\mathbf{s}_j$, if both are not tree-like representations of individual nodes (tested at line 13) and share a common root label (tested at line 14). The sibling pattern is retained (at lines 13-21) as a candidate within C if it passes the pruning condi-

Algorithm 10 Candidate-Generation(L)

Input: a set L of discriminative tree patterns;
Output: a set C of new (combined) candidate tree patterns;

```
 1: C ← ∅;
 2: for each s_i, s_j ∈ L do
 3:    T ← s_i ◁ s_j ∪ s_j ◁ s_i;
 4:    B ← B(s_i) ∩ B(s_j);
 5:    for each s ∈ T do
 6:       if (height(s) ≤ H) then
 7:          B(s) ← B;
 8:       else
 9:          T ← T − {s};
10:       end if
11:    end for
12:    C ← C ∪ T;
13:    if (|V_{s_i}| > 1) and (|V_{s_j}| > 1) then
14:       if (λ_{s_i}(r_{s_i}) = λ_{s_j}(r_{s_j})) then
15:          s ← s_i ∧ s_j;
16:          if (width(s) ≤ W) then
17:             B(s) ← B;
18:             C ← C ∪ {s};
19:          end if
20:       end if
21:    end if
22: end for
23: return C;
```

tion on its width (at line 16), that must not be greater than the maximum width W of the XML trees in the cluster. CANDIDATE-GENERATION halts by returning (at line 23) the resulting set C of new candidate tree patterns. The pseudo-code for the CANDIDATE-GENERATION procedure is illustrated in fig. 10.

Representativeness of Candidate Tree Patterns

As to the use of statistical hypothesis testing in MINEREP (at line 17), the non-parametric chi-square test is used to establish whether the representativeness of a candidate tree pattern $\mathbf{s}$ is statistically grounded. This involves a decision between two alternative hypotheses: a *null* hypothesis according to which the occurrence of $\mathbf{s}$ in $\mathcal{C}$ is a consequence of chance (and, thus, the representativeness of $\mathbf{s}$ must necessarily be considered as statistically groundless) and an *alternative* hypothesis, according to which the occurrence of $\mathbf{s}$ in $\mathcal{C}$ is statistically relevant (and, hence, $\mathbf{s}$ must be representative of some corresponding structural properties). To make a proper decision between the two alternative hypotheses, the following four statistics are considered (that can be efficiently computed from $\mathcal{B}(\mathbf{s})$ and $\mathcal{D}(\mathbf{s})$): $n_{\mathbf{s}\mathcal{C}}$, the number of XML trees in $\mathcal{C}$ of which $\mathbf{s}$ is an embedded substructure; $n_{\mathbf{s}\neg\mathcal{C}}$, the number of XML trees, within any cluster other than $\mathcal{C}$, of which $\mathbf{s}$ is an embedded substructure; $n_{\neg\mathbf{s}\mathcal{C}}$, the

number of XML trees within $\mathcal{C}$ of which $\mathbf{s}$ is not an embedded subtree; and $n_{\neg\mathbf{s}\neg\mathcal{C}}$,the number of XML trees within all clusters but $\mathcal{C}$ of which $\mathbf{s}$ is not an embedded subtree.

The above statistics enable the computation of two marginal totals, namely, the overall numbers $n_{\mathbf{s}}$ and $n_{\neg\mathbf{s}}$ of XML trees in $\mathcal{C} \cup \overline{\mathcal{C}}$ that contain, respectively, do not contain $\mathbf{s}$ as an embedded substructure. Marginal totals, in turn, allow to compute the expected values of the foregoing statistics, respectively denoted as $\overline{n}_{\mathbf{s}\mathcal{C}}$, $\overline{n}_{\mathbf{s}\neg\mathcal{C}}$, $\overline{n}_{\neg\mathbf{s}\mathcal{C}}$, $\overline{n}_{\neg\mathbf{s}\neg\mathcal{C}}$, that represent those values that would be expected if there was no meaningful correlation between $\mathbf{s}$ and $\mathcal{C}$, i.e., if $\mathbf{s}$ occurred in $\mathcal{C}$ by chance. Precisely, $\overline{n}_{\mathbf{s}\mathcal{C}} \triangleq \frac{n_{\mathbf{s}}|\mathcal{C}|}{|\mathcal{C}\cup\overline{\mathcal{C}}|}$, $\overline{n}_{\neg\mathbf{s}\mathcal{C}} \triangleq \frac{n_{\neg\mathbf{s}}|\mathcal{C}|}{|\mathcal{C}\cup\overline{\mathcal{C}}|}$, $\overline{n}_{\mathbf{s}\neg\mathcal{C}} \triangleq \frac{n_{\mathbf{s}}|\overline{\mathcal{C}}|}{|\mathcal{C}\cup\overline{\mathcal{C}}|}$, $\overline{n}_{\neg\mathbf{s}\neg\mathcal{C}} \triangleq \frac{n_{\neg\mathbf{s}}|\overline{\mathcal{C}}|}{|\mathcal{C}\cup\overline{\mathcal{C}}|}$. Given both the observed and expected statistics, it is possible to compute the value of the following test-statistic:

$$\chi^2(\mathbf{s},\mathcal{C}) = \sum_{\mathbf{s}'\in\{\mathbf{s},\neg\mathbf{s}\},\mathcal{C}'\in\{\mathcal{C},\neg\mathcal{C}\}} \frac{\left(n_{\mathbf{s}'\mathcal{C}'} - \overline{n}_{\mathbf{s}'\mathcal{C}'}\right)^2}{\overline{n}_{\mathbf{s}'\mathcal{C}'}}$$

The null hypothesis is rejected in favor of a statistically relevant occurrence of $\mathbf{s}$ in $\mathcal{C}$ if the difference between observed and expected statistics is high, i.e. if $\chi^2(\mathbf{s},\mathcal{C}) > \tau_\alpha$, where τ_α is the threshold for the chi-square distribution with one degree of freedom at a significance level α.

6.5 Evaluation

In this section, the behavior of the devised clustering approach is investigated through an empirical evaluation with three main objectives: the assessment of clustering quality, the evaluation of cluster-summarization and a performance comparison.

All experiments were conducted on a Windows machine, with an Intel Itanium processor, 2Gb of memory and 2Ghz of clock speed. Standard benchmark data sets were employed for a direct comparison among the competitors. Real-world data, named *Real*, encompasses the following collections.

- Astronomy, 217 documents extracted from an XML-based metadata repository, that describes an archive of publications owned by the *Astronomical Data Center* at NASA/GSFC.
- Forum, 264 documents concerning messages sent by users of a Web forum.
- News, 64 documents concerning press news from all over the world, daily collected by *PR Web*, a company providing free online press release distribution.
- Sigmod, 51 documents concerning issues of SIGMOD Record. Such documents were obtained from the XML version of the ACM SIGMOD Web site produced within the *Araneus* project [35].
- Wrapper, 53 documents representing wrapper programs for Web sites, obtained by means of the *Lixto* system [12].

The distribution of tags within the above documents is quite heterogeneous, due to the complexity of the DTDs associated with the classes, and to the semantic differences among the documents. It is worth noticing that the Sigmod dataset consists of

a larger collection of 988 documents, complying to three different DTDs (reported in `http://www.sigmod.org/record/xml/`). This larger collection can be exploited for testing the evaluating the generation of the cluster hierarchy.

Three further synthetic data sets were generated from as many collections of DTSs reported in [39]. The first synthesized data set, referred to as *Synth1*, comprises 1000 XML documents produced from a collection of 10 heterogeneous DTDs (illustrated in fig. 6 of [39]), that were individually used to generate 100 XML documents. These DTDs exhibit strong structural differences and, hence, most clustering algorithms can produce high-quality results.

A finer evaluation can be obtained by investigating the behavior of the compared algorithms on a collection of XML documents, that are very similar to one another from a structural point of view. To perform such a test, a second synthesized data set, referred to as *Synth2* and consisting of 3000 XML documents, was assembled from 3 homogeneous DTDs (illustrated in fig. 7 of [39]), individually used to generate 1000 XML documents. Experiments over *Synth2* clearly highlight the ability of the competitors at operating in extremely-challenging applicative-settings, wherein the XML documents share multiple forms of structural patterns.

In addition, the collection *Synth3* consisting of the synthesized documents in [41], is used. It exhibits a 30% degree of overlap. Again, this dataset allows us to compare the effectiveness of the current proposed approach to the previous approach proposed in [41].

The generation of artificial data was performed by means of the XML data generator described in [45]. The latter essentially accepts an input DTD and produces a set of conforming documents, on the basis of suitable statistical models governing the occurrences of elements marked by operators $*$, $?$, $|$, and $+$. The generation process was constrained as in [39]. Precisely, the maximum number of occurrences of a child node in the context of its parent node is fixed to 6. The actual number of repetitions is, hence, randomly chosen in the interval $[0, 6]$. The maximum depth of the synthetic XML trees was set to 7.

GENERATE-HIERARCHY effectiveness is evaluated in multiple steps. Let $\mathcal{P}^{(l)}$ be the partition of a data set $\mathcal{D}$ produced by GENERATE-HIERARCHY at level l of the resulting cluster hierarchy.

One interesting aspect is to fix l to 1, which reduces GENERATE-HIERARCHY to GENERATE-CLUSTERS, and assess the effectiveness of the basic partitioning scheme at separating $\mathcal{D}$ with respect to multiple forms $\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(m)}$ of structural patterns. This involves a comparison of clustering quality across the partitions $\mathcal{P}_i^{(1)}$, with $i = 1, \ldots, m$, where $\mathcal{P}_i^{(1)}$ is the outcome of GENERATE-CLUSTERS$(\mathcal{D}, \mathcal{S}^{(i)})$.

Clustering effectiveness is evaluated over each partition $\mathcal{P}_i^{(1)} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ according to external criteria, i.e. some pre-specified structure, that corresponds to a meaningful explanation of the data at hand. More precisely, the XML documents within the generic data set $\mathcal{D}$ can be grouped into structurally-homogeneous groups on the basis of the known conformity to some generating DTD. Thus, experiments aim to assess the effectiveness of GENERATE-HIERARCHY in recognizing

such groups. Effectiveness is measured in terms of average precision and recall [10] of partition $\mathcal{P}_i^{(1)}$, as described follows.

Let $\mathbf{C}_1, \ldots, \mathbf{C}_h$ be the true classes that actually partition $\mathcal{D}$, i.e., such that $\mathcal{D} = \cup_i \mathbf{C}_i$ and $\mathbf{C}_i \cap \mathbf{C}_j = \emptyset$ for each $i \neq j$. Also, let $\mathcal{P}_i^{(1)} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ be a partition of $\mathcal{D}$ into k clusters, where k is automatically fixed by GENERATE-CLUSTERS. $\mathcal{P}_i^{(1)}$ can be summarized into a contingency table m, where columns represent discovered clusters and rows represent true classes. Each entry m_{ij} indicates the number of transactions, related to XML documents in $\mathcal{D}$, that were associated with cluster $\mathcal{C}_j$, with $1 \leq j \leq k$, and actually belongs to class C_i, with $1 \leq i \leq h$. The table provides an immediate visual description of the degree of agreement between the results yielded by GENERATE-CLUSTERS and the actual class partition. The table also permits a quantitative evaluation of such a degree of correspondence, that can be finely caught through two traditional measures from the field of information retrieval, namely precision and recall. Intuitively, each cluster $\mathcal{C}_j$ corresponds to the class $\mathbf{C}_i$ that is best represented in $\mathcal{C}_j$, i.e., such that m_{ij} is maximal. For any cluster $\mathcal{C}_j$, the index $h(j)$ of the class with maximal m_{ij} is defined as $h(j) \triangleq max_i\, m_{ij}$. Precision $P(\mathcal{C}_j)$ and recall $R(\mathcal{C}_j)$ for cluster $\mathcal{C}_j$ are, hence, defined as follows:

$$P(\mathcal{C}_j) = \frac{|\{\mathbf{x^{(t)}} \in \mathcal{C}_j | \mathbf{t} \in \mathbf{C}_{h(j)}\}|}{|\mathcal{C}_j|},\ R(\mathcal{C}_j) = \frac{|\{\mathbf{x^{(t)}} \in \mathcal{C}_j | \mathbf{t} \in \mathbf{C}_{h(j)}\}|}{|\mathbf{C}_{h(j)}|}$$

Starting from the above measures, it is possible to define the average precision $P^{(1)}$ and recall $R^{(1)}$, respectively, as $PP^{(1)} = \frac{1}{|\mathcal{P}^{(1)}|} \sum_{\mathcal{C} \in \mathcal{P}^{(1)}} P(\mathcal{C})$, $R = \frac{1}{|\mathcal{P}^{(1)}|} \sum_{\mathcal{C} \in \mathcal{P}^{(1)}} R(\mathcal{C})$. The average precision and recall for all other partitions $\mathcal{P}^{(i)}$ with $i > 1$ are defined analogously.

Table 6.1 shows the results of clustering on such collections. Precision and recall are optimal, even for the collection *Synth2* of homogeneous documents.

Collection	N. of Docs	Classes	Clusters	Precision	Recall	Avg Γ	Time
Real	649	5	5	1	1	0.9558	20.48s
Synth1	1000	10	10	1	1	0.9455	13.32s
Synth2	3000	3	3	1	1	0.3833	7.5s
Synth3	1400	7	7	1	1	0.7875	2.68s
Synth4	800	8	10	1	0.8	0.7127	3.68s

Table 6.1. Evaluation of Separability and homogeneity

A second aspect to investigate is the effectiveness of the GENERATE-HIERARCHY procedure when all of the m forms of structural patterns are progressively considered (in some meaningful sequence) to progressively partition $\mathcal{D}$. To this purpose, a new synthetic dataset was produced *Synth4*, with 800 documents complying to the schema shown in fig. 6.1. Four separate classes of documents can be noted, each using a different subset of nodes (in particular, `DTD1` and `DTD2` share the nodes

A1, . . . , A5, whereas DTD3 and DTD4 share nodes A6, . . . , A10). The clustering process was applied by considering node, edge and path features, in sequence.

DTD1

<!ELEMENT A1 (A2 | A5)>
<!ELEMENT A2 (A3,A4,A5)>
<!ELEMENT A5 (A3 |(A4,A3))>
<!ELEMENT A4 (A5)>
<!ELEMENT A3 (#PCDATA)>

DTD2

<!ELEMENT A5 ((A2 ,A4)|(A2,A5))>
<!ELEMENT A2 ((A4 ,A5)|(A4,A1))>
<!ELEMENT A4 (A1|A3|(A1,A3)>
<!ELEMENT A1 (#PCDATA)>
<!ELEMENT A3 (#PCDATA)>

DTD3

<!ELEMENT A6 (A9,A10)>
<!ELEMENT A9 ((A10 ,A9)|(A7))>
<!ELEMENT A10 (A7,A8)>
<!ELEMENT A7 (#PCDATA)>
<!ELEMENT A8 (#PCDATA)>

DTD4

<!ELEMENT A10 (A9)>
<!ELEMENT A9 ((A6|(A7,A10))|(A7,A8,A9))>
<!ELEMENT A6 (#PCDATA)>
<!ELEMENT A7 (#PCDATA)>
<!ELEMENT A8 (#PCDATA)>

DTD5

<!ELEMENT A11 (A12 | A15)>
<!ELEMENT A12 (A13,A14,A15)>
<!ELEMENT A15 (A13 |(A14,A13))>
<!ELEMENT A14 (A15)>
<!ELEMENT A13 (#PCDATA)>

DTD6

<!ELEMENT A15 ((A12 ,A14)|(A12,A15))>
<!ELEMENT A12 ((A14 ,A15)|(A14,A11))>
<!ELEMENT A14 (A11|A13|(A11,A13)>
<!ELEMENT A11 (#PCDATA)>
<!ELEMENT A13 (#PCDATA)>

DTD7

<!ELEMENT A16 (A19,A20)>
<!ELEMENT A19 ((A20 ,A19)|(A17))>
<!ELEMENT A20 (A17,A18)>
<!ELEMENT A17 (#PCDATA)>
<!ELEMENT A18 (#PCDATA)>

DTD8

<!ELEMENT A20 (A19)>
<!ELEMENT A19 ((A17,A20)|(A17,A18,A19))>
<!ELEMENT A17 (#PCDATA)>
<!ELEMENT A18 (#PCDATA)>

Fig. 6.1. DTDs for the *Synth3* dataset

The DTDs capture substantial similarities and differences. In particular, all dataset exhibit different paths (but they can share some edges). Further, the documents in DTD4 can further split, since they can exhibit trees with paths ending in the node A6. Also, node frequencies in DTD4 can substantially differ, thus differentiating this DTD from the others even at a node level. This situation is fully captured by the clustering algorithm, as shown in fig. 6.2. DTD4 is kept separate from DTD3 at a node level, and further split in two subclusters at the edge level, according to whether edges contain or not edge (A9, A6). Also, the trees containing such an edge can be further split according on whether they contain the path from A10 to A6 or not. Notice that, by contrary, DTD8 does not behave similarly, since there's no such a node like A6 capable of differentiating the trees in the class.

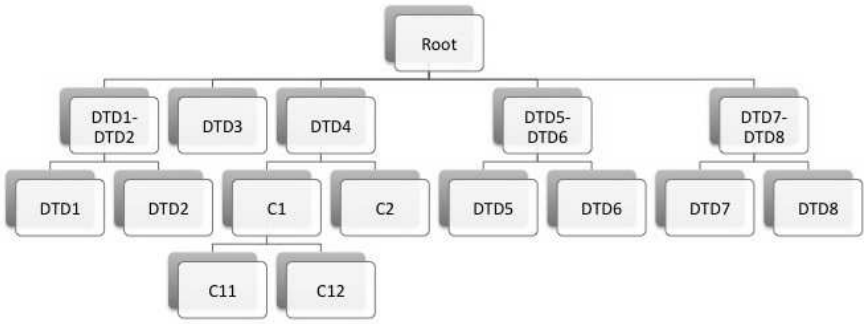


Fig. 6.2. Cluster hierarchy for the *Synth3* dataset

The evaluation of the multi-stage clustering is further confirmed by experimenting on Sigmod. As already mentioned, this dataset consists of documents

complying with three different DTDs. In particular, the distribution of the documents is unbalanced, since one of the DTDs, named `IndexTermsPage` (see `http://www.sigmod.org/record/xml/` for details), contains 920 documents. Figure 6.3 shows that GENERATE-HIERARCHY separates all documents complying to different DTDs and further splits the documents in the class related to `IndexTermsPage`, according to whether or not these documents contain the optional elements described in the DTD (`categoryAndSubjectDescriptorsTuple`, `category`, `content` and `term`). In particular, the separation of such a class leads to two subclasses $\mathtt{C}_1$ and $\mathtt{C}_2$, that can be described by two DTDs, both subsumed by `IndexTermsPage`. $\mathtt{C}_1$ is a subclass of 437 documents, in which the optional elements of `IndexTermsPage` are absent. Instead, the remaining 483 documents in which such optional elements occur are assembled into subclass $\mathtt{C}_2$, which finely catches the overall degree of structural resemblance induced by these additional pieces of structures.

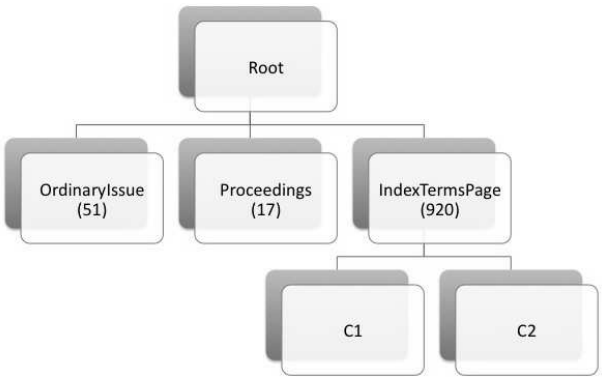


Fig. 6.3. Cluster hierarchy for the *Synth3* dataset

The evaluation of the accuracy of cluster summarization is inspired to an idea originally proposed in [39] for a different purpose, i.e., measuring the structural homogeneity of a set of intermediate clusters obtained while partitioning a collection of XML documents. Let $\mathbf{t}$ be an XML tree and $\mathcal{R}$ a set of substructures. The representativeness $\gamma(\mathcal{R}, \mathbf{t})$ of $\mathcal{R}$ with respect to $\mathbf{t}$ is the fraction of nodes in $\mathbf{t}$ matched by the embedded substructures of $\mathcal{R}$:

$$\gamma(\mathcal{R}, \mathbf{t}) \triangleq \frac{|\cup_{\mathbf{s}\in\mathcal{R},\mathbf{s}\subseteq\mathbf{t}} \{n \in \mathbf{V} | \mathbf{V_s} \mapsto \mathbf{V} \subseteq \mathbf{V_t}\}|}{|\mathbf{V_t}|}$$

where, $\mathbf{V_t}$ and $\mathbf{V_s}$ are the sets of nodes of, respectively, the XML tree $\mathbf{t}$ and the generic substructure $\mathbf{s}$. $\mathbf{V}$ is instead the subset of the nodes in $\mathbf{t}$ matched by the nodes of $\mathbf{s}$ (which is the meaning of notation $\mathbf{V_s} \mapsto \mathbf{V}$). Representativeness can be easily generalized to clusters. The representativeness $\Gamma\,[Rep(\mathcal{C})]$ of the representative $Rep(\mathcal{C})$ with respect to a cluster $\mathcal{C}$ can be defined as the average representativeness of the documents in the cluster.

Cluster representativeness finds an interpretation in terms of intra-cluster structural homogeneity. Indeed, $\Gamma\left[Rep(\mathcal{C})\right]$ is low if cluster $\mathcal{C}$ includes structurally heterogeneous XML trees and, hence, MINEREP (fig. 9) is unable to extract frequent-occurring and statistically-representative substructures. On the contrary, $\Gamma\left[Rep(\mathcal{C})\right]$ is high if $\mathcal{C}$ contains structurally homogeneous XML trees, whose nodes are matched in a high percentage by the variety of embedded substructures extracted by MINEREP. Table 6.1 shows the average Γ value exhibited in each experiment. *Synth1* exhibits the maximum value of Γ among the synthetic datasets, and by contrast *Synth2* exhibits the lowest. Indeed, documents in *Synth2* share several features, and hence the representative pattern trees only cover small fragments that really discriminate among clusters.

Scalability and Comparison

In order to evaluate the scalability of the algorithm, were used the DTDs for *Synth1* and produced respectively 100, 1000, 10.000 and 100.000 documents with 2, 4, and 8 clusters. The results are shown in fig. 6.4. The algorithm is linear both in the number of documents and in the number of clusters. This is mainly inherited by the intrinsic scalability of the core AT-DC algorithm, which is used for clustering. The only possible issue of the algorithm is the computation of the representative when the documents are complex and the intra-similarity is high. This is shown in table 6.1, where the times are higher than all the others on synthesized data. Clearly, a levelwise approach to the computation of a representative might suffer from the high density of the substructures in the documents.

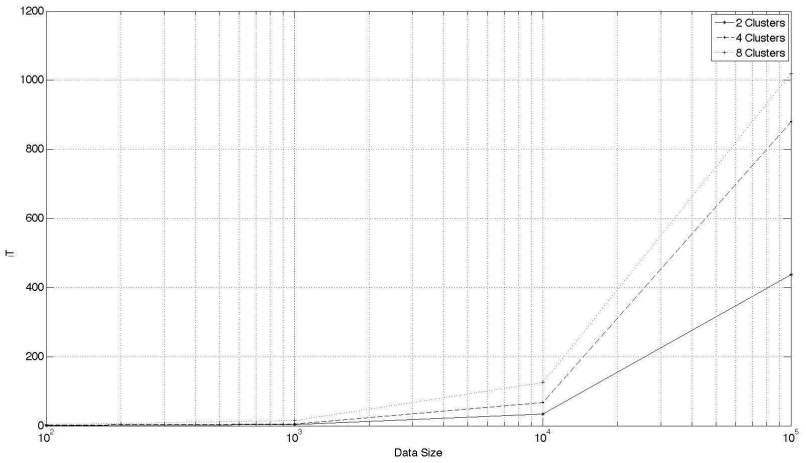


Fig. 6.4. Performance results for datasets of different size

At the end of this intensive empirical evaluation, the proposed approach can be compared against a state-of-the-art competitor, namely the XProj approach [39]. Unfortunately, a direct comparative evaluation is not possible since, it was not possible to obtain the latter framework from its authors. Notwithstanding, the exploitation of the same data sets used in [39], namely *Sigmod*, *Synth1* and *Synth2*, still enables an indirect and robust comparison. By looking at the performance of XProj reported in [39], it can be seen that the result of the proposed clustering approach attains the same quality. However, two strong advantages of the proposed approach are: the development of a hierarchy of nested clusters, that explain multiple forms of structural relationships in the data, that remain uncaught with XProj; the summarization of a cluster of XML documents, which provides an intelligible subsumption of its structural properties, which is not offered by XProj. Also, notice that the scalability of the proposed approach is much higher than the one of XProj. Finally, the proposed approach is fully-automatic, parameter-free, whereas the optimal performance of XProj on each dataset requires a complex parameter-setting process.

7

Conclusions and FutureWork

This thesis has proposed formal solutions to complex problems born from needs of the information society. Below are presented in detail the obtained results.

7.1 Inductive Query Language

For the sake of formality, the thesis first proposed a language for formulating analytical statements with which to progressively mine and query data. The procedural semantics of the language is founded on the `2W Model` algebraic framework, that allows to accommodate and combine disparate mining tasks into a multi-step knowledge discovery process.

There are some challenging issues, that are worth further research. Foremost, the identification of a compact `2W Model` algebra, consisting of a fixed, minimal set of operators. Analogously to the case of the `3W Model` framework, this is useful in two respects, i.e. the possibility of expressing the required patterns via suitable combinations of such basic operators, rather than relying on an arbitrary number of task-oriented mining operators, and the development of a solid theoretical background concerning expressiveness and complexity results.

The development of strategies for optimizing processing plans is also an interesting task, because it would increase the overall performance of a possible engine that implements this language.

7.2 Classification in hostile environments

In this area the thesis proposed two novel schemes for a tight integration between rule-learning and probabilistic learning [21], aimed to improve the classification performance on the classes of interest in imprecise environments. A massive evaluation revealed that the resulting learning framework is competitive and often superior in classification performance w.r.t. established rule-based competitors. There are many viable lines of research that may improve performance on rare classes even further.

The ongoing research efforts are geared towards the definition of a discriminative model, based on the maximum entropy principle, that explicitly enables the specification of the features (i.e. the CAR antecedents) that either concur to or prevent from the individual class memberships.

Also, it's possible to pursue the improvement of the accuracy of the local probabilistic generative models through the analysis of ROC curves. The point is that the classification threshold typically used in the proposed framework assigns a class label when the associated probability is higher than 0.5. However, the latter may not necessarily be the best threshold, especially if the bias introduced by the CAR associated with the probabilistic classifier is considered. In general, lower thresholds produce improvements in recall, by contemporarily degrading precision as a side effect. However, as suggested by fig.4.4 where a better threshold value can be obtained in correspondence to the (0.8,0.01) pair (corresponding to the threshold 0.2), by automatically choosing the best class-specific threshold, probabilistic smoothing can still allow to remove some locality effects within the CAR and maintain high precision.

7.3 Collaborative filtering

In this field, the thesis proposed a probabilistic model for the discovery of both global and local patterns from users' preference data. Experimental evaluation showed that both the User Community model (for the discovery of global patterns) and the hierarchical topic detection model (for the discovery of local patterns) exhibits prediction capabilities comparable to state-of-the art approaches. Also, the proposed approach exhibits high flexibility in discovering structural patterns capable of providing suitable interpretations of the users' preference data.

The proposed approach is suitable for further investigations in several respects. Foremost, the proposed strategy can be combined with temporal information in order to better model user changes in preferences. Also, the proposed approach allows suitable integration of prior modeling for the "cold-start" issues. Finally, more in general, the "local patterns" approach can be extended to other approaches based on ensembles.

7.4 Tree-Structured data mining

In this thesis a new approach to clustering of XML documents was proposed, that produces a hierarchy of nested clusters. Along the paths from the root to the leafs of the hierarchy, the approach progressively separates the XML data by looking at the occurrence of different types of structural patterns in their structures. Also, each cluster in the hierarchy is subsumed, through a novel summarization method, by a set of representative substructures, that provide an understanding of the structural properties considered in the cluster. A comparative evaluation proved that the devised approach is on a par and even better than established competitors in terms of effectiveness, scalability and cluster summarization.

In contrast to previous works, the devised approach does not rely on fixed reference structures (such as summaries and s-graphs) to partition XML documents. Rather, it takes into account various (user-supplied) forms of structural patterns in the XML documents to guide the clustering process.

Multi-stage clustering is also a parameter-free method, that only requires setting a significance threshold for testing the statistical representativeness of the substructures during cluster summarization. No tuning is required: usual settings correspond to well-known values, e.g., 0.05, or 0.01.

Finally, multi-stage clustering efficiently processes large-scale databases of XML documents and provides an intelligible understanding of the structural properties of the clusters in the hierarchy.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.
2. Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28, 2009.
3. Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 91–100, 2010.
4. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of Int. Conf. on Very Large Data Bases*, pages 487 – 499, 1994.
5. Alfred V. Aho and Jeffrey D. Ullman. Universality of data retrieval languages. In *Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '79, pages 110–119, New York, NY, USA, 1979. ACM.
6. M.-L. Antonie and O. R. Zaïane. An associative classifier based on positive and negative rules. In *Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 64–69, 2004.
7. Paterek Arkadiusz. Improving regularized singular value decomposition for collaborative filtering. In *Proc. of KDD Cup Workshop at SIGKDD'07*, pages 39–42, 2007.
8. B. Arunasalam and S. Chawla. CCCS: A top-down association classifier for imbalanced class distribution. In *Proc. of ACM KDD*, pages 517–522, 2006.
9. A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.
10. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
11. Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S. Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 509–514, 2004.
12. R. Baumgartener, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. In *Procs. VLDB'01 Conf.*, pages 119 – 128, 2001.
13. S. D. Bay and M. J. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5:213–246, 2001.
14. Alan Beaulieu. *Learning SQL*. Mary E. Treseler, 2009.
15. Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD '07: Proceedings*

of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 95–104, 2007.
16. Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 43–52, 2007.
17. James R. Beniger. *The control revolution: technological and economic origins of the information society*. Harvard University Press, Cambridge, MA, USA, 1986.
18. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.
19. A. L. Berger, V. J. Della Pietra, and S. A. Della Pietra. A maximum entropy approach to natural language processing. *Journal of Artificial Intelligence Research*, 22:39–71, 1996.
20. Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22:39–71, March 1996.
21. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
22. Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
23. David M. Blei and Jon D. Mcauliffe. Supervised topic models. In *NIPS '08: Neural Information Processing Systems Advances in Neural Information Processing Systems 21*, 2008.
24. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.
25. Stanislav Busygin, Oleg Prokopyev, and Panos M. Pardalos. Biclustering in data mining. *Comput. Oper. Res.*, 35:2964–2987, September 2008.
26. Toon Calders, Laks V. S. Lakshmanan, and Raymond T. Ng. Expressive power of an algebra for data mining. *ACM Trans. Database Syst*, 31:1169–1214, 2006.
27. E. Cesario, F. Folino, A. Locane, G. Manco, and R. Ortale. Boosting text segmentation via progressive classification. *Knowledge and Information Systems*, 15(3):285–320, 2008.
28. E. Cesario, G. Manco, and R. Ortale. Top-down parameter-free clustering of high-dimensional categorical data. *IEEE TKDE*, 19(12):1607 – 1624, 2007.
29. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
30. N.V. Chawla, A. Lazarevic, L.O. Hall, and K. Bowyer. Smoteboost: Improving prediction of minority class in boosting. In *Proc. of Principles of Knowledge Discovery in Databases*, pages 107–119, 2003.
31. H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proc. of Int. Conf. on Data Engineering*, pages 716–725, 2007.
32. F. Coenen. LUCS KDD implementations of CBA and CMAR, 2004.
33. W. W. Cohen. Fast effective rule induction. In *Proc. of Conf. on Machine Learning*, pages 115–123, 1995.
34. G. Costa, M. Guarascio, G. Manco, R. Ortale, and E. Ritacco. Rule learning with probabilistic smoothing. In *Proc. of Int. Conf. on Data Warehousing and Knowledge Discovery*, 2009. To appear.
35. V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Procs. VLDB'01 Conf.*, pages 109–118, 2001.
36. Christopher J. Date. *An Introduction to Database Systems*. Addison Wesley, 2004.

37. R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley & Sons, 2001.
38. C. Elkan. The foundations of cost-sensitive learning. In *Proc of Int. Joint Conf. on Artificial Intelligence*, pages 973–978, 2001.
39. C. C. Aggarwal et al. Xproj: A framework for projected structural clustering of xml documents. In *Procs. SIGKDD'07 Conf.*, pages 46 – 55, 2007.
40. C. Wang et al. Efficient pattern-growth methods for frequent tree pattern mining. In *Procs. PAKDD'04 Conf.*, pages 441 – 451, 2004.
41. G. Costa et al. A tree-based approach to clustering xml documents by structure. In *Procs. PKDD'04 Conf.*, pages 137 – 148, 2004.
42. K. Abe et al. Efficient substructure discovery from large semi-structured data. In *Procs. SDM'02 Conf.*, pages 158 – 174, 2002.
43. M. L. Lee et al. Xclust: Clustering xml schemas for effective integration. In *Procs. CIKM'02 Conf.*, pages 292 – 299, 2002.
44. M. N. Garofalakis et al. Xtract: A system for extracting document type descriptors from xml documents. In *Procs. SIGMOD'00 Conf.*, pages 165 – 176, 2000.
45. S. Flesca et al. Fast detection of xml structural similarity. *IEEE TKDE*, 17(2):160 – 175, 2005.
46. T. Dalamagas et al. A methodology for clustering xml documents by structure. *Information Systems*, 31(3):187 – 228, 2006.
47. W. Lian et al. An efficient and scalable algorithm for clustering xml documents by structure. *IEEE TKDE*, 16(1):82 – 96, 2004.
48. K. Ezawa, M. Singh, and S.W. Norton. Learning goal oriented bayesian networks for telecommunications risk management. In *Proc. of Int. Conf. on Machine Learning*, pages 139–147, 1996.
49. W. Fan, S.J. Stolfo, J. Zhang, and P.K. Chan. Adacost: Misclassification cost-sensitive boosting. In *Proc. of Int. Conf. on Machine Learning*, pages 97–105, 1999.
50. R.E. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 3(1):291–316, 1997.
51. E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proc. of Int. Conf. on Machine Learning*, pages 144–151, 1998.
52. S. Funk. Netflix update: Try this at home, 2006.
53. F. Giannotti B. Kujpers A. Raffaeta G. Manco, M. Baglioni and C. Renso. Querying and reasoning for spatio-temporal data mining. In *In GIANNOTTI F. and PEDRESCHI D., editors, Mobility, Data Mining, and Privacy: Geographic Knoweledge Discovery*. Springer-Verlag, 2008.
54. Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 625–628, 2005.
55. Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007.
56. S. Helmer. Measuring the structural similarity of semistructured documents using entropy. In *Procs. VLDB'07 Conf.*, pages 1022 – 1032, 2007.
57. Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, 2003.
58. Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *IJCAI'99: Proceedings of the 16th international joint conference on Artificial intelligence*, pages 688–693, 1999.

59. R.C. Holte, L. Acker, and B. Porter. Concept learning and the problem of small disjuncts. In *Proc. of Int. Conf. on Artificial Intelligence*, pages 813–818, 1989.
60. N. Japkowicz. The class imbalance problem: Significance and strategies. In *Proc. of Int. Conf. on Artificial Intelligence*, pages 111–117, 2000.
61. N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002.
62. Rong Jin, Luo Si, and Chengxiang Zhai. A study of mixture models for collaborative filtering. *Inf. Retr.*, 9(3):357–382, 2006.
63. Xin Jin, Yanzan Zhou, and Bamshad Mobasher. A maximum entropy web recommendation system: combining collaborative and content features. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 612–617, 2005.
64. Theodore Johnson, Laks V.S. Lakshmanan, and Raymond T. Ng. The 3w model and algebra for unified data mining. pages 21–32. Morgan Kaufmann, 2000.
65. M.V. Joshi, R.C. Agarwal, and V. Kumar. Predicting rare classes: Can boosting make any weak learner strong? In *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 297–306, 2002.
66. M.V. Joshi, V. Kumar, and R.C. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Proc. of IEEE Int. Conf. on Data Mining*, pages 257–264, 2001.
67. Mehmed Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley and Sons, 2003.
68. Mohammad Khoshneshin and W. Nick Street. Incremental collaborative filtering via evolutionary co-clustering. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 325–328, 2010.
69. Michael Kifer, Won Kim, and Yehoshua Sagiv. Querying object-oriented databases. In *ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*, pages 393–402. ACM New York, 1992.
70. Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.
71. Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, 2009.
72. M. Kubat, R.C. Holte, S. Matwin, R. Kohavi, and F. Provost. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2):192–215, 1998.
73. M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proc of Int. Conf. on Machine Learning*, pages 179–186, 1997.
74. Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with gaussian processes. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 601–608, 2009.
75. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of ACM SIGKDD Int. Conf. on Kwnoledge Discovery and Data Mining*, pages 80–86, 1998.
76. B. Liu, Y. Ma, and C.K. Wong. Improving an association rule based classifier. In *Proc. of Principles of Data Mining and Knowledge Discovery*, pages 504–509, 2000.
77. David Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.
78. Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In *In NIPS*17*, 2003.

79. Benjamin Marlin and Benjamin Marlin. Collaborative filtering: A machine learning perspective. Technical report, Department of Computer Science, University of Toronto, 2004.
80. G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.
81. T.M.. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
82. S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In *Procs. SIGMOD'98 Conf.*, pages 295 – 3006, 1998.
83. Barbieri Nicola, Guarascio Massimo, and Manco Giuseppe. A block mixture model for pattern discovery in preference data. In *TFDOM '10 (to appear)*, 2010.
84. N. Mamoulis P. Kalnis and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *In C. B. Medeiros, M. J. Egenhofer, and E. Bertino, editors, SSTD, volume 3633 of Lecture Notes in Computer Science, pages 364381*. Springer, 2005.
85. M. Pazzani, C. Merz, P. Murphy, T. Hume, and C. Brunk. Reducing misclassification costs. In *Proc. of Int. Conf. on Machine Learning*, pages 217–225, 1994.
86. Nikos Pelekis and Yannis Theodoridis. Boosting location-based services with a moving object database engine. In *Proc. of Int. ACM SIGMOD/PODS Workshop on Data Engineering for Wireless and Mobile Access*, 2006.
87. C. Phua, D. Alahakoon, and V. Lee. Minority report in fraud detection: Classification of skewed data. *ACM SIGKDD Explorations Newsletter*, Special issue on learning from imbalanced datasets:50–59, 2004.
88. Ian Porteous, Evgeniy Bart, and Max Welling. Multi-hdp: a non parametric bayesian model for tensor factorization. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pages 1487–1490. AAAI Press, 2008.
89. F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.
90. P. Riddle, R. Segal, and O. Etzioni. Representation design and brute-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence*, 8(1):125–147, 1994.
91. R. Hull S. Abiteboul and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
92. Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 880–887, 2008.
93. Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, pages 1257–1264. 2008.
94. Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.
95. Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658, 2008.
96. David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 111–120, 2009.
97. Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.
98. K.M. Ting. A comparative study of cost-sensitive boosting algorithms. In *Proc. of Int. Conf. on Machine Learning*, pages 983–990, 2000.
99. J. Wang and G. Karypis. HARMONY: Efficiently mining the best rules for classification. In *Proc. of SIAM Int. Conf. on Data Mining*, pages 205–216, 2005.
100. K. Wang and H. Liu. Discovering typical structures of documents: A road map approach. In *Procs. SIGIR'98 Conf.*, pages 146 – 154, 1998.

101. G. Webb, J. Boughton, and Z. Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.
102. G.M. Weiss. Learning with rare cases and small disjuncts. In *Proc. of Int. Conf. on Machine Learning*, pages 558–565, 2000.
103. G.M. Weiss. Mining with rarity: A unifying framework. *ACM SIGKDD Explorations*, 6(1):7–19, 2004.
104. G.M. Weiss and H. Hirsh. A quantitative study of small disjuncts. In *Proc. of Nat. Conf. on Artificial Intelligence*, pages 675–670, 2000.
105. G.M. Weiss and F. Provost. Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.
106. X. Xin and J. Han. CPAR: Classification based on predictive association rules. In *Proc. of SIAM Int. Conf. on Data Mining*, pages 331–335, 2003.
107. M. J. Zaki. Efficiently mining frequent trees in a forest: Algorithms and applications. *IEEE TKDE*, 17(8):1021 – 1035, 2005.
108. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245 – 1262, 1989.

List of Figures

List of Tables