Data Min Knowl Disc (2017) 31:424–464
DOI 10.1007/s10618-016-0480-z

Survey on using constraints in data mining

Valerio Grossi[1] · **Andrea Romei**[1] · **Franco Turini**[1]

Received: 9 July 2014 / Accepted: 28 September 2016 / Published online: 22 October 2016
© The Author(s) 2016

Abstract This paper provides an overview of the current state-of-the-art on using constraints in knowledge discovery and data mining. The use of constraints in a data mining task requires specific definition and satisfaction tools during knowledge extraction. This survey proposes three groups of studies based on classification, clustering and pattern mining, whether the constraints are on the data, the models or the measures, respectively. We consider the distinctions between hard and soft constraint satisfaction, and between the knowledge extraction phases where constraints are considered. In addition to discussing how constraints can be used in data mining, we show how constraint-based languages can be used throughout the data mining process.

Keywords Data mining · Constraints · Background knowledge

1 Introduction

This paper is the first comprehensive survey of the current state of the art in data mining and constraint solving. The survey is designed to help researchers in the field

Responsible editor: Kristian Kersting.

✉ Valerio Grossi
vgrossi@di.unipi.it

Andrea Romei
andrearomei74@gmail.com

Franco Turini
Franco.Turini@unipi.it
http://pages.di.unipi.it/turini/

[1] Department of Computer Science, University of Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy

of constraint solving to find opportunities for applying their methods to data mining problems, whereas designers of data mining methods can redesign them in order to exploit constraints in novel ways to improve the efficiency and efficacy of the mining process.

Knowledge extraction includes steps such *pre-processing*, *mining* and *reasoning* on the extracted knowledge. By using knowledge discovery processes, where data mining is the fundamental step, analysts obtain important and strategic information for their business and decision support systems. Knowledge discovery transforms the role of data profoundly, changing the interaction with traditional databases. It does this by encouraging the main DBMS producers to integrate data mining functions into their products, or enriching well-established mining approaches with the use of constraints that exploit the background knowledge.

The integration of constraints in data mining has rapidly emerged as a challenging topic for the research community. Many ad-hoc extensions of mining algorithms that use constraints for improving the quality of their results have been proposed for pattern mining, classification and clustering models. The use of constraints requires a way to define and satisfy them during the knowledge extraction process. This is crucial for the quality of the extracted data mining models as well as for the scalability of the entire process. On the one hand, an analyst can define the knowledge extraction phase where a constraint must be satisfied. On the other, an optimizer is required to understand where a constraint must be satisfied inside the process flow, in an automatic way. Mining algorithms must also be rewritten to directly satisfy constraints during the model extraction.

Expressing and reasoning on background knowledge through the definition and the integration of constraints allows the user to specify additional information on input data as well as requirements and expected properties of data mining models in output in a declarative way. When constraints are fully integrated into the data mining algorithms, for example as additional search space pruning mechanisms, this raises efficiency considerably. The extraction of association rules typically leads to a large quantity of useless rules. An approach that extracts the rules by specifying the analysts needs can speed up both the domain experts evaluation of the extracted rules and the extraction algorithm itself.

Data mining extracts synthetic models from datasets, i.e. collections of records characterizing data with respect to several dimensions. Background knowledge on the domain of interest, especially when represented by constraints, maybe useful in data mining for:

- *Filtering* and *organizing* the dataset before applying data mining methods.
- *Modifying* the data mining algorithms in order to improve them either by making the search more efficient or by focusing the search itself.
- *Reasoning* on the results of the mining step in order to refine them and present a more refined view of the extracted mod els.

In this survey, the use of constraints is considered with respect to four dimensions:

1. The kind of mining **task** they are used for, specifically *classification*, *clustering* and *pattern mining*.

2. Which **objects** the constraints are applied to, specifically *data*, *models* and *measures*.
3. The **type** of constraints: *hard* and *soft*.
4. The **phases** of the knowledge extraction, in which the constraints are used, namely *pre*, *mining* and *post*.

Finally, there are two perspectives on the merging of constraints and mining algorithms. The most common is the embedding of constraints into the algorithms. However a complementary view requires studying how constraint programming languages, or declarative programming in general can be used to define mining algorithms and the mining process.

Paper Organization The paper is organized as follows. Section 2 introduces some basic concepts on how to classify constraints in data mining. Sections 3, 4, and 5 deal with data mining tasks, and within each section we address the objects the constraints are dealing with. Section 3 describes classification and how data constraints and model constraints are used when making classifications. Section 4 deals with clustering as well as how constraints are used for data, models, and measures. Section 5 considers pattern mining, data constraints, measure constraints, and model constraints. In Sect. 6 we focus on how a declarative approach can be used to perform data mining. Section 6.1 discusses how constraint programming can be directly used to compute data mining. Section 6.2 discusses how another declarative paradigm, i.e. inductive databases, can be used for data mining tasks. In this context, two examples are proposed. Section 6.2.1 deals with extensions to SQL, while Sects. 6.2.2 and 6.2.3 deal with extensions to XML and include a case study.

2 Constraints in data mining: basic concepts

There are several works in the literature on constraints in data mining tasks. Currently, each mining task has its own way to classify constraints. A full view that combines for example mining tasks and the objects on which constraints are defined, is still lacking. One of the aims of this paper is thus to provide a general overview on where a constraint can be categorized. This section therefore provides a complete description of the dimensions on which constraints can be characterized. This view is based on the main dimensions that each kind of constraint impacts on in its specific mining context.

Constraints can be on *data*, on the mining *model*, and on *measures*.

– *Constraints on data (or items)* these involve either the item per se or attributes of the item. This class of constraints defines how the data are analyzed by mining algorithms. They can be specified at the item/instance level, e.g. *must* and *cannot-links* in a clustering problem (e.g. see Sect. 4), or at the feature level, e.g. *consider only the items with a higher price than a given threshold* for pattern mining (see Sect. 5).
– *Constraints on the mining model* these define specific requirements that an extracted model should satisfy. This kind of constraint does not involve background knowledge directly, but requires the complete knowledge of the characteristics

required by the output model. For example, they include the extraction of association rules with a specific set of items in the body and in the head, or the discovery of clusters with a minimum number of elements.
– *Constraints on measures* measures, e.g. entropy for classification, support and confidence for frequent itemsets and distance for clustering, play an important role in data mining, since they are related to the quality of the model extracted. This class of constraints specifies a requirement that the computation of a measure should respect. This involves both knowledge about data and about the characteristics of a model. For example, if we consider clustering people as moving objects, the trajectory implementing the shortest distance cannot cross a wall, since people typically walk around an obstacle, or we can constrain a classifier to provide a minimum level of accuracy (e.g. see Sect. 3).

This survey is organized according to the *data mining task* and *kind of object* the constraints are applied to. The other two dimensions for the categorization of the use of constraints are the type (*hard* vs. *soft*), and the mining phase (*pre-processing*, *mining*, *post-processing*) in which they are used (see Table 1).

The use of constraints does not necessarily guarantee the best solution, which in fact may entail relaxing constraints. This leads to the need to classify constraints as either hard, or soft (i.e. relaxable).

– *Hard constraint* a constraint is *hard* if a model that violates it is unacceptable. A hard-constrained algorithm halts when there is no state that satisfies all the constraints, and it returns no results (Okabe and Yamada 2012). This situation is common when a large set of constraints are provided as input.
– *Soft constraint* a constraint is *soft* if, even though a model that satisfies all the constraints is preferable, a solution that does not satisfy the constraints is acceptable and especially when no other (or better) solution is available (Bistarelli et al. 1997). Some constraints work well for the required solution, while others do not, and in some contexts where a result is needed, it is important to select a set of useful constraints that should be considered as hard, while others can be treated as soft (Davidson et al. 2006).

This dimension is strictly related to the actual definition of a constraint and it should not be perceived as a rigid categorization. As we shall see in the following sections, there are some constraints that are conceived as hard but they may be redefined as soft (by means of weights) based on the problem and the properties required by the final solution. For example, a set of must/cannot-links can include a weight/cost associated with each constraint, and then an algorithm can choose to minimize the cost by violating some constraints.

Since a data mining task, or more generally a knowledge extraction process, is based on several iterated phases, constraints can also be characterized according to when a knowledge extraction process satisfies the set of constraints defined by the user. The *pre-processing* includes data cleaning, normalization, transformation, feature extraction and selection, and its aim is to produce a set of data for the subsequent processing/mining step. Pyle (1999) presents basic approaches for data pre-processing. *Processing* is the core step where the knowledge extraction is performed. This is the mining phase where a model is extracted.

Table 1 Main dimensions of the different classes of constraints

	Classification	Clustering	Pattern
Data	*phase*: pre, mining	*phase*: mining	*phase*: pre, mining
	type: hard	*type*: hard, soft	*type*: hard
Model	*phase*: mining, post	*phase*: mining	*phase*: mining
	type: soft	*type*: soft, hard	*type*: hard, soft
Measure	*phase*: mining, post	*phase*: mining	*phase*: mining, post
	type: hard, soft	*type*: hard	*type*: hard

Finally, *post-processing* is required to verify if the models extracted by the data mining algorithms are valid and useful. If a model does not meet the desired standards, it is necessary to re-run the process and carefully change some parameters of the pre-processing and mining steps. A detailed explanation of mining approaches and the related post-processing measures and techniques is provided in the following sections.

Techniques for constraint-driven mining can be roughly characterized on the basis of the knowledge extraction phase in which they are satisfied:

- *Pre-processing constraints* are satisfied during the pre-processing phase. They enable a restriction of the source data to the instances that can only generate patterns that satisfy them.
- *Processing/Mining constraints* are directly integrated into the mining algorithm used to extract the model. The constraint satisfaction in this case is embedded directly into the mining algorithms, leading to a reduction of the search space.
- *Post-processing constraints* are satisfied either for filtering out patterns generated by the mining algorithm, and for highlighting only the relevant results given a measure of interest provided by the user.

The knowledge extraction phase is the last dimension that we introduce. This phase gives a complete picture of the constraints for data mining, and in the following sections we explicitly describe the phase where the constraint is satisfied.

Table 1 summarizes the main characteristics related to the different dimensions of constraints proposed in this section. The two main dimensions are the mining task and the kind of object where a constraint is applied. In addition, for each of the pairs, the phase and the type of constraints are presented.

Starting with the classification, we can state that data constraints can be applied only in the pre-processing phase to build the training set, and in the mining phase, while the model is being extracted. The post-processing phase rarely involves the training data since it only considers features of the extracted classifier. In a classification task, data constraints are considered to be hard, since they are used to define and build the training set. Data constraints for clustering are only applied during the mining phase. As we shall see in Sect. 4.1, they are satisfied when all the items are grouped into clusters. As in the classification, data constraints for pattern mining are satisfied in the pre-processing phase, for example by establishing hierarchies, which may also be used in the mining phase.

Model constraints are typically satisfied in the mining phase. In the case of classification, these kinds of constraints are soft. As we will see in Sect. 3.2, model constraints are defined on performance measures, and in many cases a hard satisfaction of the constraints could imply empty solutions (i.e. none of the classifiers are extracted).

Constraints on measures are related to the mining phase, and also to the post-processing phase, given that mining and post-processing steps can be repeated until the required result is reached. The measure constraints related to clustering are hard, since they are part of the formulation of the problem. For example, consider the clustering of trajectories, where a set of constraints defines the one-way streets. Relaxing these constraints can imply solutions not allowed in reality, since a user is not permitted to go in the wrong direction.

3 Using constraints in classification

Classification is one of the most popular approaches in data mining. The aim is to predict the behavior of new elements, i.e. the *label* or *class*, given a set of past and already classified instances. The process of classifying new data begins from a set of classified elements (*training set*), and tries to extract some regularities from them. *Decision trees* or *rule-based learners* are examples of eager approaches. In this category, most of the computing resources are focused on extracting a model, but once a model has been built, classifying a new object is a relatively fast process. By contrast, lazy learners, such as *nearest-neighbor classifiers*, do not require an explicit model building phase. However classifying a test example can be very expensive, since the element to classify must be compared with all the elements in the training set. Using constraints in classification thus defines requirements that data elements and the output model must satisfy.

Constraints in classification task can be defined on data, both at the level of instances and classes, on models and measures. Data constraints are typically hard, and are used in the pre-processing and mining phases. Model and measure constraints are used in the mining and post-mining phases and can be either hard or soft (i.e. relaxable), under specific circumstances.

We now introduce a way of defining constraints in classification tasks, based on data attributes including class label values (Sect. 3.1), and model/measure selection via constraints (Sect. 3.2).

3.1 Classification: data constraints

The literature in this field has explored the relations among instances and classes, and among different classes themselves. This is principally due to the fact that a classifier is extracted from a training set. The training set is explicitly conceived by the analyst, and given a set of requirements of a classification task, a training set is defined by employing data constraints directly in the pre-processing step. Some constraints such as those that explicitly assign a property to certain attributes, e.g. "*attribute Name is secret for employees whose Salary is greater than* $1000" or classify the association between different attributes, also depending on some conditions, for instance, the

association between names, role and salaries, are applied directly when the training set is created (Dawson et al. 1999). Thus, as shown in Table 1, the easiest way to use data constraints in classification is during the pre-processing step when the training set is built.

Other kinds of constraints that are satisfied in the mining phase are the pairwise constraints. They are mainly used for improving supervised (and unsupervised) classification. In classification (as in clustering, see Sect. 4.1) a pairwise constraint between two pairs of examples indicates whether or not they belong to the same class.

Many single-label classifiers have been extended to manage pairwise constraints, such as Logistic regression, k-Nearest Neighbors (k-NN), and Support Vector Machine (SVM).[1]

Nguyen and Caruana (2008), Zhang and Yan (2007), Lange et al. (2005) show how pairwise constraints can be used for integrating labeled data. In the cited works, the authors propose a supervised learning algorithm that integrates labeled data with pairwise constraints. Generally, pairwise constraints in classification are useful when there is a lack of training data. They are used to infer decision boundaries exploiting the "space" defined by the examples that are either constrained to be together or are not. Pairwise constraints can be applied in a real application where video objects are classified (Yan et al 2006). Also in this case, a classification framework directly models the decision boundary with labeled data as well as additional pairwise constraints, but without explicitly estimating the underlying data distribution. Another interesting approach is proposed in Druck et al. (2008), where there is a lack of labeled instances. In this case, the knowledge base is a set of labeled features, and the authors propose a method for training probabilistic models with labeled features (constrained by domain knowledge) from unlabeled instances. Labeled features are employed directly to constrain the model predictions on unlabeled instances. Tsochantaridis et al (2005) proposes a constrained SVM approach: the authors consider cases where the prediction is a structured object or consists of multiple dependent constrained variables. Brunner et al. (2012) extends an SVM framework for handling large pairwise classification problems. This work changes the perspective since its aim is pairwise classification, i.e. predicting whether the examples *a*,*b* of a pair *(a, b)* belong to the same class or to different classes. Label constraints also exploit monotonic relationships. The latter describe relationships between class labels and the values of attributes expressed on ordinal or numeric scales. Monotonicity constraints reflect a semantic correlation that can be used to discover inconsistent objects by increasing the accuracy of the final model (Blaszczynski et al. 2012, 2010; Duivesteijn and Feelders 2008).

Data constraints are applied in multi-label classification, i.e. where the learning task requires a model that predicts a subset of relevant class labels for a new instance. This case allows one instance to have more than one label simultaneously. This enables the user to define several constraints on labels (i.e. not only on pairs of examples), or to attribute values during the training set creation.

[1] The basic idea of support vector machine is to represent the decision boundary using a subset of the training examples, known as support vectors. Given a set of hyperplanes, the classifier selects one hyperplane for representing its decision boundary, based on how well they are expected to perform on the example to classify.

The multi-label problem can be modeled by a *subset constraint*, which given a set of relevant labels L^+ (and irrelevant L^-), specifies that if a label λ_i is relevant for a given instance x then λ_j has to be relevant, i.e. $\lambda_i \in L^+ \rightarrow \lambda_j \in L^+$. Similarly, *exclusion constraints* can be created by specifying that two labels cannot be relevant or irrelevant at the same time $((\lambda_i \in L^+ \rightarrow \lambda_j \in L^-) \wedge (\lambda_i \in L^- \rightarrow \lambda_j \in L^+))$.

Subset and *exclusion* constraints are quite similar to instance-level constraints that have been explored in semi-supervised or constraint-based clustering (see Sect. 4). They define relations between different labels (known groups of instances), whereas the constraints for semi-supervised clustering are defined between instances (e.g. this pair of instances must (or not) belong to the same cluster).

Subset and exclusion constraints are processed and satisfied in the mining phase. Har-Peled et al. (2002) introduces a constrained classification task, where each example is labeled with a set of constraints that relate multiple classes. Each constraint specifies the relative order of two classes and the goal is to learn a classifier consistent with these constraints. As reported in Park and Fürnkranz (2008), in many applications explicit constraints among the labels can be easily discovered. For example, in the context of hierarchical classification, the presence of one label in the hierarchy often also implies the presence of all its ancestors.

In many domains, data constraints can be easily available from background knowledge. As reported below, a simple hierarchy defines a set of subset and exclusion constraints. However, in many applications domain knowledge is not readily available, and an algorithm can automatically discover label constraints from the data.

In a technical report (Park and Fürnkranz 2008), the authors propose discovering label constraints via an association rule mining approach, where each item set is derived from a training sample x_i consisting of a set of relevant labels L_i^+ (mined by applying multi-label classification). Such association rules take the form $\lambda_{i1} \ldots \lambda_{ib} \rightarrow \lambda_j$ with b labels in the antecedent and only one label in the consequent. A similar method is proposed in Gu et al. (2010) where a decomposition technique is used to divide a multi-label problem into binary class sub-problems. A rank list of labels is generated by combining the probabilistic outputs of each binary SVM classifier. In this context, label constraint rules are learned by minimizing the ranking loss. Multi-label classification is also solved with the use of decision trees (Vens et al 2008), in this case the constraints are defined over a hierarchy of classes.

All the constraints introduced above are typically considered as hard: they filter features of the training set as well as relations among the classes.

3.2 Constraints on classifiers or measures: model optimization via constraints

These kinds of constraints are requirements on models and/or on measures. We collect the constraints on models and those on measures in the same group since they are strictly connected. We start by first introducing the constraints on measures, and then outlining the constraints on models. This is useful in order to understand that a measure such *entropy* or the *distance* in a k-NN classifier, influences the final models. Some constraints can also be defined both on measures and models. There are several standard metrics for testing the quality of a classifier. These metrics include

error count, *accuracy*, *true-positive* (tp) and *false-positive* (fp) rates from which we can derive *precision*, *recall* or *sensitivity*, *F-measure*, and *specificity*. As discussed in Fawcett et al. (2006), tp-rate and fp-rate have some attractive properties, such as being insensitive to changes in class distribution.

Yan and Goebel (2006), Vanderlooy et al. (2009) deal with the design of a classifier under constrained performance requirements. In particular, Vanderlooy et al. (2009) enables the user to define a desired classifier performance. The Vanderlooy et al. (2009) provides a complete analysis of when a classifier is constrained to a desired level of precision, F-measure and generally to tp/fp-rate related performance measures. The model learned is adjusted to achieve the desired performance, abstaining from classifying the most ambiguous example in order to guarantee the required level of performance. This is an example when a set of constraints defined on measures also clearly influences the learned model implicitly. Similarly in Yan and Goebel (2006), an ensemble of neural networks is constrained by a given tp or fp-rate to ensure that the classification error for the most important class is within a desired limit. The process of constructing the final ensemble consists in iteratively training the individual neural network on different samples of the training set and considering different sets of features.

Recalling Table 1, we can state that the constraints on measures are typically satisfied during the mining phase, when the classifier is built. They can also be satisfied during the post-processing step, e.g. re-mining/re-arranging iteratively a classifier until the desired level of accuracy is reached. These kinds of constraints are hard. In some cases in order to avoid empty models, some constraints, for example on tp/fp-rate, can be relaxed. Thus, constraints on the models that are introduced at the end of this section can also be considered as soft.

In this context, constraints are also useful to address the problem of the average tp-rate maximization under average fp-rate constraints in online settings (Bernstein et al. 2010), or can be employed in dimensionality reduction approaches aimed at extracting lower-dimensional features relevant for classification tasks. In Costa et al. (2005), this is obtained by modifying the Laplacian approach to manifold learning through the introduction of class-dependent constraints. The correlation between the two classifiers (or more) can also be represented in terms of their respective errors, and thus an ensemble of classifiers can also be constrained by defining requirements among classifiers (Niyogi et al 2000; Yan and Goebel 2006; Rigollet and Tong 2011a, b).

Since k-NN methods are based on a distance measure, Lucey and Ashraf (2013) introduces a k-NN approach that includes spatial constraints, and demonstrates empirically that they are essential to a good generalization performance by showing how constraints for distance measures enforce the accuracy of a classifier.

Garofalakis (2003) proposes an approach where constraints are defined both on model and measures. It enables users to specify constraints on either the size (i.e. number of nodes), or the number of misclassified records of the target classifier, and employs these constraints to efficiently construct the "best possible" decision tree. The authors propose a branch-and-bound algorithm in order to make the constraints part of the building phase of classifiers, and to prune early tree nodes that cannot satisfy the constraints.

Finally, Nijssen and Fromont (2007, 2010), Niyogi et al (2000) are examples of constraints defined only on models. These works propose different kinds of constraints, related to the form of a decision tree, e.g. internal nodes should not have pure class distributions or rules regarding the class distribution. In particular, Nijssen and Fromont (2010) defines a framework for determining which model constraints can be pushed into the pattern mining process, by proposing an optimal classifier model. More precisely, Nijssen and Fromont (2010) shows how several categories of constraints defined for frequent itemset mining, e.g *monotonic*, *anti-monotonic* and *convertible* (available in Sect. 5), can be applied in decision tree induction. It highlights the connection between constraints in pattern mining and constraints in decision tree extraction, by developing a general framework for categorizing and managing decision tree mining constraints.

A comparison of SVM classifiers with constraints can be found in Zaidan and Eisner (2008), Druck et al. (2008), Small et al. (2011). In this case, a SVM classifier is enriched by adding additional constraints from a domain knowledge. Small et al. (2011) emphasizes the direct encoding of expert beliefs in the form of weight constraints, and also exploits ranked labeled features.

4 Constrained clustering models

Clustering is the process of partitioning a set of data objects into subsets without any supervisory information such as data labels. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. Clustering can lead to the discovery of previously unknown groups within the data. Examples of data objects include database records, graph nodes, a set of features describing individuals or images. Cluster analysis is used in a wide range of applications such as: business intelligence, image pattern recognition, web analysis, biology (Witten et al. 2011; Tan et al. 2006; Han and Kamber 2012). Well known examples of constraint-based clustering include the detection of road lanes and trip routines from GPS data (Wagstaff et al 2001), or the navigation aids of Sony Aibo Robot (Davidson and Ravi 2005b).

The literature proposes several ways to compute and represent a cluster. The partition method is based on prototypes and is one of the most widely studied and applied approaches. In this case, every cluster is selected and represented by a prototype called a centroid (e.g. *K-means* and *K-medoid*). Prototype-based techniques tend to consider the region only based on a distance value from a center. This approach typically provides clusters with globular shapes. Density-based approaches also work with non-globular regions and are designed for discovering dense areas surrounded by areas with a low density typically formed by noise or outliers. Hierarchical-clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Finally, spectral clustering solves clustering problems when the available information is encoded as a graph. This section introduces the main techniques related to clustering with constraints, and explores the use of data constraints considering the above types of clustering techniques (see Sect. 4.1.1 for partitional and density-based methods and Sect. 4.1.2 for the other methods). The following subsections highlight two main aspects in this field by analyzing the main contributions that define constraints on data and then on models and measures.

As we will see, the dimensions chosen for categorizing the use of constraints in clustering are definable both on data, and on models and measures, which are generally hard, with the possibility of relaxing them under very specific circumstances.

4.1 Clustering with data constraints

This class of constraints includes clustering methods that incorporate pairwise constraints on clustering membership or the study of specific distance measures that produce desirable output. In the literature, constraints for clustering related to data are also referred to as instance-level constraints, which have the form of *must* and *cannot-link* constraints. As shown in Table 1, instance-level constraints are generally satisfied during the mining phase. Although constraints are typically defined as hard, in some cases they can be changed into soft constraints. Some constraints can be relaxed to prevent the discovery of an empty cluster, or because more generally, soft constraints increase the expressivity by allowing users to specify their confidence (or uncertainty) in a constraint. For example, PC-KMeans (Basu et al. 2004a) permits some constraints to be violated by performing a soft constraint satisfaction. Law et al. (2004, 2005) propose a clustering algorithm in the presence of soft constraints, and similarly to Lu and Leen (2007) show how the use of soft constraints can improve expressivity, also by comparing different scenarios where both soft and hard constraints are considered.

Well-established approaches on data constraints for clustering problems have focused on the introduction of instance-level constraints (Wagstaff et al 2001; Wagstaff and Cardie 2000). In this case a domain expert specifies that a pair of instances must be in the same cluster (or in different clusters). *Must-link* constraints enforce two instances to be placed in the same cluster, while *cannot-link* constraints enforce two instances to be in different clusters. Several properties are related to instance-level constraints (Davidson and Ravi 2006). Must-link constraints are symmetric, reflexive and transitive. Transitive constraints enable a system to infer additional must-link constraints.

As shown in Davidson and Ravi (2005b, 2009, 2007), inferring must-links is quite easy. Let G_M be the must-link graph for data set X, with a node for each $x_i \in X$ and an edge between nodes i and j for each must-link $c_=(i, j)$ in C. Let CC_1 and CC_2 be two connected components in G_M. If a constraint $c_=(x, y)$ exists, where $x \in CC_1$ and $y \in CC_2$, then $c_=(a, b)$ can be inferred for all $a \in CC_1, b \in CC_2$.

Cannot-links are not transitive property, thus inferring cannot-link constraints is more complicated than must-link ones. If points x_i and x_j are known to belong to different clusters and the same situation is valid for points x_j and x_k, we cannot infer anything about the relation between x_i and x_k. Additional cannot-link constraints can be inferred by combining must and cannot-link. Let G_M be the must-link graph for data set X, with a node for each $x_i \in X$ and an edge between nodes i and j for each $c_=(i, j)$ in C. Let CC_1 and CC_2 be two connected components in G_M. If a cannot-link constraint $c_{\neq}(x, y)$ exists, where $x \in CC_1$ and $y \in CC_2$, then $c_{\neq}(a, b)$ can be inferred for all $a \in CC_1, b \in CC_2$. Chapter 1 in Basu et al. (2008) reports a detailed definition of the properties related to must and cannot-link.

Another class of constraints can define clustering spatial properties. The δ-constraint requires that the distance between any pair of points in two different clusters must be at least δ, while ϵ-constraint enforces that each instance x in a cluster must have another instance y in the same cluster such that the distance between x and y is at most ϵ. A complete discussion on complexity features related to the use of these kinds of constraints can be found in Davidson and Ravi (2005b); Davidson et al. (2005a). Instance-level approaches are widely studied in the literature and represent a starting point for studying constrained clustering (Wang et al 2014; Davidson and Ravi 2006).

We complete this section by introducing how the different types of constraints on data defined above are actually used in the various clustering approaches.

4.1.1 Partitional and density-based approaches

Several algorithms are available in the literature for these kinds of methods. COP-KMeans (Wagstaff et al 2001) represents a popular approach for the integration of instance-level constraints in K-means. The algorithm takes as input the data, the number of clusters required (as K-means), a set of must-links, and a set of cannot-links. Unlike K-means, by performing a hard constraint satisfaction COP-KMeans ensures that when the clustering is updated none of the specified constraints are violated. Generally, there is no guarantee that the use of must and cannot-links will ensure optimal clustering. In some cases, constraints may be redundant, e.g. a constraint that does not affect the search solution space, and/or can cause conflicts and introduce inconsistencies in the final result. For this reason, Davidson et al. (2006), Wagstaff et al. (2006) introduce the concepts of informativeness and coherence. The authors define informativeness as the amount of information in the constraint set that the algorithm cannot determine on its own. Informativeness is determined by the clustering algorithm objective function (bias) and search preference. On the other hand, given a distance matrix, the coherence measures the amount of agreement within the constraints themselves. Given these two measures, an algorithm can relax some constraints and detect dependencies among them.

Okabe and Yamada (2012) proposes a modification of COP-KMeans by introducing an ensemble approach for managing constraint priorities in order to provide a model even though all the constraints cannot be satisfied. PC-KMeans (Basu et al. 2004a) permits some constraints to be violated by performing a soft constraint satisfaction. In this case, each must and cannot-link has a weight w associated with it. By setting the value w, the algorithm looks for a trade-off between minimizing the total distance between points and cluster centroids, and the cost of violating the constraints.

Klein et al. (2002) extends the use of instance-level constraints by introducing a method for the propagation of these kinds of constraints by showing and exploiting their spatial implication. The basic observation is that must and cannot-links suggest *space-level* generalization. For example, if we consider a must-link between points a and b, the constraint suggests that a and b must be in the same cluster, but probably also the points near to a and b should be in the same cluster. The same spatial inference can be done for cannot-links.

In Bade and Nürnberger (2006, 2008), Vu et al. (2010), relative constraints are investigated under the term must-link-before constraints with a focus on text clustering.

Liu et al. (2011) defines a constraint among three data instances, two of which form the closest pair. Given three instances a, b, c, it uses $ab\|c$ to denote that (a, b) is the closest pair. This is equivalent to two relative comparisons defined on a, b, c: $d(a, b) < d(a, c)$ and $d(a, b) < d(b, c)$.

Other approaches do not require the satisfaction of constraints but they perform metric learning from constraints or adopt a hybrid solution including both constraint satisfaction and metric learning (Basu et al. 2004b). The set of constraints is mapped onto distance measures (Bar-Hillel et al. 2003). The basic idea of these approaches is to weigh the features differently, based on their importance for the distance computation based on available constraints (Bilenko et al. 2004; Xing et al 2002). Xing et al (2002) parametrizes the Euclidean distance using a symmetric positive-definite matrix, minimizing at the same time the distance between must-linked instances and maximizing the distance between cannot-linked instances. A probabilistic model based on Hidden Markov Random Fields (HMRF) can be used for incorporating constraints into prototype-based clustering. In Basu et al. (2004b) the objective function is derived from the posterior energy of the HMRF framework, and the authors propose an EM-based clustering algorithm (HMRF-KMeans) for finding the (local) minimum of this objective function. The distance measure is estimated during clustering to validate the user-specified constraints as well as to incorporate data variance. MPCK-Means (Bilenko et al. 2004) learns an individual metric for each cluster, allowing violations by imposing penalties. All the partitional methods based on instance-level constraints have been reported under data constraints, since they define relations among data. The Bilenko et al. (2004) use both must-link and cannot-link constraints between pairs of instances, and the same constraints are also used to influence the distance metric, as we will show in Sect. 4.2.

Unlike partitional approaches, density-based clustering is interesting for applications where the data instances can be grouped into clusters with different sizes or shapes, and the number of clusters is not known beforehand.

C-DBSCAN An et al. (2007), Ruiz et al. (2010) extends DBSCAN by supporting instance-level constraints. It uses a k-d Tree (Bentley 1975) to divide the data space into dense partitions and enforces cannot-link constraints within each tree leaf, thus producing "local clusters". It then merges adjacent local clusters, by enforcing the must-link constraints. Finally, it merges adjacent clusters hierarchically, while enforcing the remaining cannot-link constraints. Wang et al. (2012) proposes an approach (DBRS+) where a density-based cluster is performed in the presence of both obstacles and facilitators.[2]

4.1.2 Spectral and hierarchical clustering

Spectral clustering was originally proposed to address clustering problems where all available information is encoded in the Laplacian graph (von Luxburg 2007). With respect to K-means, spectral clustering has a deterministic polynomial-time solution,

[2] The Euclidean distance, i.e. straight-line distance is an ineffective measure in the presence of obstacles and facilitators. An obstacle is a physical object that obstructs reachability among the individuals. On the contrary, a facilitator is a physical object that enhances reachability among people.

arbitrarily modeling shaped clusters, and is equivalent to certain graph cut problems.

Several works on spectral clustering can be found in the literature, and can be differentiated based in terms of how they enforce the constraints. For example Lu et al. (2008), Wang et al (2009) directly manipulate the Laplacian graph (or equivalently, the affinity matrix) according to the given constraints. Unconstrained spectral clustering is then applied on the modified Laplacian graph; on the other hand (Li et al. 2009; Coleman et al. 2008; Li et al. 2009) use constraints to restrict the feasible solution space.

Wang and Davidson (2010), Wang et al (2014) propose a general framework for an embedded solution that incorporates large amounts of both hard and soft constraints both to manipulate the graph and to restrict feasible solutions. Wang et al (2014) proposes a comparison between label propagation and pairwise constraints (i.e. must and cannot-links) in a constrained spectral clustering approach. Finally, clustering based on constrained hierarchies is widely employed where the use of such a knowledge base already exists and provides a readily available source of constraints. Several preliminary works in this field adopt instance-level constraints for hierarchical clustering. In particular, Davidson and Ravi (2009, 2007) demonstrate how instance-level can be also used for hierarchical clustering.

The limitation introduced by only using instance-level constraints is highlighted in Kestler et al. (2006). The paper also proposes a method to limit the scope of the constraints. In the same line, Gilpin and Davidson (2011) adapt instance-level constraints to their problem, enabling the definition of constraints that can be specified at more than two levels. Furthermore, Gilpin and Davidson (2011) demonstrated how constrained hierarchical clustering can be solved by integrating a SAT problem in polynomial time. The algorithm uses SAT as an oracle to decide whether or not a pair of instances can be grouped into the same cluster.

Bade and Nürnberger (2008) proposes a specific type of constraint specifically designed for the hierarchical clustering problem: must-link-before (MLB) is a ternary relationship that specifies that two points should be joined together before they are both joined to a specified third point in the resulting hierarchy.

4.2 Clustering with model and measure constraints

The literature has widely explored how instance-level constraints can be employed in different and heterogeneous clustering applications. The must- and cannot-links proposed in the previous section can also be considered constraints on models or measures since they influence the measure by forcing some instance to stay together or not.

This section overviews recent studies on constraints on clustering models and/or on distance measures. Model and measure constraints specify properties or requirements regarding the generated model, or define specific constraints on how to compute the distance between the data elements. For example, an analyst wants to avoid empty clusters or to define a measure for managing a specific situation, e.g. a measure that

considers walls, or obstacles when clustering people that are moving in a square as reported in Han and Kamber (2012).

The K-means algorithm is a basic approach for forcing clustering models to have specific properties. In Bradley (2000), Demiriz et al. (2008), the authors avoid empty clusters by adding k constraints to the clustering problem, by requiring cluster h to contain at least τ_h points. The solution proposed is equivalent to a minimum cost flow linear network optimization problem (Bertsekas 1991).

Another approach to a constrained model can be found in Banerjee and Ghosh (2008), Banerjee and Ghosh (2006). In this case the constraint requires the clusters obtained to have a comparable size. The proposed method has three steps: *(1)* sampling, *(2)* clustering of the sampled set and *(3)* populating and refining the clusters while satisfying the balancing constraints.

Other methods for constraining the clustering approach to find balanced clusters can be found in Strehl and Ghosh (2003) which proposes graph partition techniques or hierarchical approaches which encourage balanced results while progressively merging or splitting clusters (Zhong and Ghosh 2003).

As reported in Xing et al (2002), in several applications it is desirable to provide a more systematic way for users to specify the concept of "similar". The distance function is typically implemented in a look-up conversion matrix, allowing individual distance values for any pair of records, e.g. by separating instances that should be linked, because they are still too far away from each other. The use of an adjusted function for constraint enforcement implies that constraints may be violated. Constraint violation may also be associated with some penalty value. For example, instead of the binary statement that a constraint is or is not violated, a partial satisfaction of a constraint can be considered (Law et al. 2005).

Many papers focus on metric learning driven by constraints. Distance measure learning and clustering with constraints in K-means were both considered in Bilenko et al. (2004), and the result was extended to a hidden Markov random field formulation in Basu et al. (2004b).

In Schultz and Joachims (2003), an SVM-like approach is employed to learn a weighted distance function from relative comparisons. The method learns a weighted Euclidean distance from constraints by solving a convex optimization problem that is similar to SVMs to find the maximum margin weight vector. Kumar and Kummamuru (2008) proposed learning an SVaD (Kummamuru et al. 2004) measure from relative comparisons. Relative comparisons were first employed in Schultz and Joachims (2003) to learn distance measures using SVMs. The results on relative comparisons can be used to solve clustering problems with relative constraints (since each relative constraint is equivalent to two relative comparisons).

Recalling Table 1, these kinds of constraints are satisfied during the mining phase, when clustering is performed. Both constraints on measures and models are generally hard. While constraints on measures can be used to model rigid characteristics for computing the distances between instances, the constraints on models can be changed into soft ones if empty models are discovered.

5 Pattern mining

In pattern discovery the objective is to discover all the patterns of interest. Thus, the use of constraints is important both to speed up the analysis and to obtain a precise output. The basic methods of pattern mining are *frequent itemsets mining* (FIM), *association rule mining* (ARM) and *sequential pattern mining* (SPM). See Zhang and Zhang (2002), Han et al. (2007), Shankar (2009) for past surveys on ARM, and Mabroukeh and Ezeife (2010), Chand et al. (2012a) for surveys on SPM.

In constrained pattern mining the aim is to introduce a set of constraints that patterns should satisfy. On the one hand, the analyst can use a conjunction of constraints to specify the properties of the patterns of interest. On the other, data mining systems should be able to exploit such constraints to speed up the knowledge extraction.

After introducing the relevant concepts on constrained pattern mining (Sect. 5.1), we survey three groups of studies: *data*, *patterns* and *interest measures*. Data constraints specify which subset of items should (or should not) be present in the patterns, possibly including measures associated with each item. They include multi-level constraints and weight constraints (Sect. 5.2). Pattern constraints are expressed on the whole pattern. This class includes the use of regular expressions as background knowledge and the relaxation of constraints (Sect. 5.3). Finally, constraints based on interesting measures include time and recency constraints and those based on aggregates (Sect. 5.4).

In conclusion, with reference to the dimensions in Table 1, constraints are used in the mining phase with reference to data, models and measures, whereas in the pre-processing phase they are also used on data and in the post-mining phase on measures. In all cases, the type of constraint is generally hard, with relaxation possible only in very specific cases.

5.1 Constrained pattern mining

Several classes of constraints and their properties have been identified in the literature, including those for pruning the data and those for pruning the search space. The first class includes succinct and anti-monotonic constraints (see Chapter 7 in Han and Kamber 2012). Here, we concentrate on the second class of constraints, including the anti-monotonic, monotonic, succinct, convertible, and loose anti-monotone properties:

- *Anti-monotonic*: if a predicate holds for a data set S, then it holds for any subset of S (Agrawal et al. 1993). An interesting example of an anti-monotone constraint is the Apriori property. For example, suppose we have access to data on web pages accessed by users on an internet website. Each item of the mined pattern is a visited page. Each page has a fixed numeric attribute, pr, representing its pagerank value. Given an itemset S, the constraint $max(S.pr) \leq 0.6$ is anti-monotone, where $max(S.pr)$ is the maximum value of the pagerank among the items in S.
- *Monotonic*: if a predicate holds for a data set S, it also holds for any superset of S (Bucilă et al. 2003). Recalling the web page access for anti-monotonic constraints, given an itemset S, the constraint $max(S.pr) > 0.6$ is monotone, where $max(S.pr)$ is the maximum value of the pagerank among the items in S.

– *Succinct*: the decision concerning the satisfaction of a predicate can be determined on the basis of the individual elements in the itemset (Lakshmanan et al. 1999). Continuing the example of the web page access, the two kinds of constraints previously defined are both succinct, since a single page of S can be used as a comparator for the constraint. Conversely, the $avg(S.pr) > 0.5$ constraint is not succinct, where $avg(S.pr)$ stands for the average value of the pagerank among the pages in S.

There are other interesting properties for constraints in pattern mining, such as the class of *convertible anti-monotone* and *convertible monotone* (Pei and Han 2000). This class expresses properties that define an item-ordering relation, and the class of *loose anti-monotone* constraints (Bonchi and Lucchese 2007), which are valid for at least one of the subsets of a pattern. See Bonchi and Lucchese (2007) and Pei et al (2007) for a characterization of the classes of constraints in FIM and SP, respectively.

5.2 Data constraints for pattern mining

Data constraints are introduced to discover the patterns that include or do not include specific items. They can also be used for generalizing items when such items are too specific and do not have sufficient support. As shown in Table 1, these kinds of constraints are satisfied during the pre-processing or mining phase, and are hard.

There are two ways in which the background knowledge can be used to express data constraints. First, by means of a *concept hierarchy*, i.e. a multi-level organization of the various entities or concepts defined in a particular domain. For example, in web mining, a concept hierarchy has the form of a taxonomy describing the "is-a" relationship among the pages visited by users, e.g. "narrative book shopping" is a kind of "book shopping", which, in turn, is a kind of "shopping". Second, *weighted pattern mining* emerges when considering the different semantic significance of the items, in which greater importance in the mining process is given to certain items.

Multi-level constraints Mining on concept hierarchies is also called *constraint-based multilevel DM*, and patterns extracted with the aid of a taxonomy are also called *multi-level*. They are useful since patterns mined at the bottom level of a hierarchy are too specific or have insufficient support.

Multi-level constraints are satisfied during the pre-processing phase, generalizing items at the bottom level to higher levels of the hierarchy before applying the mining algorithm. Basically, each transaction t, is replaced with its "extended transaction", which contains all the items in t along with their ancestors (Srikant and Agrawal 1995). Methods to integrate multi-level constraints into mining algorithms are introduced in Han and Fu (1999), in which frequent itemsets are generated one level at a time in the hierarchy. Srikant et al. (1997) and Han et al. (1999) can be seen as the first attempts to integrate multilevel mining directly into the Apriori, employing anti-monotonic properties existing across levels. More recent work on generalized rule mining includes (Sriphaew and Theeramunkong 2002) on exploiting the lattice of generalized itemsets, and Wu et al (2011), using efficient data structures to retrieve item generalizations.

Baralis et al. (2012) exploits schema constraints and the opportunistic confidence constraints to remove uninteresting rules. Schema constraints are expressed by a set of data constraints on the structure of the pattern of interest. An opportunistic confidence constraint compares each generalized rule with a set of similar ones. Similar rules are then pruned at each level of the hierarchy.

The approach in Srikant and Agrawal (1996) is one of the first to deal with multi-dimensional sequential patterns, allowing items across all levels of a taxonomy. Pinto et al. (2001) considers a pair to be a pattern in which the first element is a set of dimensions over the data (e.g. location and customer group), and the second element is a sequence (e.g. item purchased). Plantevit et al. (2010) extends this concept to several dimensions combined over time and w.r.t. different levels of granularity.

Ontologies are a natural extension of taxonomies since they are based on relationships among concepts, where a concept is a class of entities within a domain. Ontologies can guide the mining process towards more interesting patterns, thus association rule mining has attracted a large number of researchers (Mansingh et al 2011). All works in this area share the idea of using ontologies to guide the extraction of multi-level constraint-based association rules during pre-processing (Bellandi et al. 2007), mining (Antunes 2009), or post-processing (Mansingh et al 2011; Marinica and Guillet 2010a).

Weight constraints The weight of an item is a non-negative real number measuring the importance of the item in the transaction database. The weighted support of a pattern is defined as the product of the patterns support by the weight of the pattern. For example, in extracting sequences from web pages for a recommendation system, the importance of each page can be considered differently, e.g. according to the resulting values of certain algorithms such as *pagerank* measuring the impact of the page. The idea is that although an item may violate the minimum support constraint, it might still be considered in the resulting patterns due to its high weight.

Yao et al (2004) calls this problem *utility mining*. The authors define two types of utilities for items: *transaction utility* and *external utility*. A transaction utility is the numerical value assigned to an item in a transaction, e.g. the quantity of the item purchased in that transaction. The external utility is stored in a utility table and is independent of each transaction, e.g. the additional profit on a certain item if sold.

Weighted and utility pattern mining are extensively proposed in FIM and ARM, in the discussion of a new tree structure that is robust to database modifications (Ahmed et al. 2012); in forcing the weight constraint into pattern growth algorithms (Yun and Leggett 2005; Tseng et al. 2013; Yun et al 2012), or into level-wise methods (Wang et al 2000; Tao and Murtagh 2003; Li et al. 2008); and in suggesting approximated weighted frequent pattern mining, as a fault tolerant factor (Yun and Ryu 2011). As far as the SPM problem is concerned, (Chang 2011) focuses on finding time-interval weighted sequential patterns; and Yun (2008), Yun and Ryu (2010) look at integrating weight constraints in the prefix projection growth method.

Weight constraints represent another example where constraints defined on data clearly influence the entire model. Considering the concept of utility, weight constraints also influence a measure related to pattern mining.

5.3 Pattern-model constraints

Pattern constraints are related to the form or structure of the entire pattern, as well as to the relations among items. In the case of association rules, for example we might want to extract the following rules: $visit(X, sport) \wedge visit(X, shopping) \Rightarrow buy(X, tshirt)$ which means that when the user X visits a sports page and a shopping page together, then he/she buys a t-shirt. Similar considerations hold for sequential patterns. For example, we might want to find patterns that first include visiting a sports page, then a shopping page, and finally a finance page (Pei et al 2007). Here, we are searching for useful meta-rules to specify the syntactic form of the patterns (Fu and Han 1995). These constraints can be specified using either high-level user interfaces or declarative data mining query languages.

Here, we briefly review the regular expressions (RE) in sequential pattern mining. These expressions are based on the typical RE operators, such as disjunction and Kleene closure, to constrain the set of items. Then, we deal with the relaxation of constraints.

There are several algorithms supporting RE constraints. SPIRIT (Garofalakis 1999) is based on an evolution of the GSP algorithm. RE-Hackle represents RE by means of a tree structure (Capelle et al. 2003). Prefix-growth extends the prefix-span approach with several kinds of constraints, including RE (Pei et al 2007). SMA (Sequence Mining Automata) is based on an adaptation of Petri Nets (Trasarti et al. 2008). These kinds of constraints are related to the form of the pattern to discover, and they are satisfied during the mining phase. Pattern constraints are typically hard since they are driven by the aims of the analysts.

Relaxation of constraints In Sect. 5.1, we showed that monotonic and anti-monotonic properties can be used to prune the search space. Even if a constraint does not satisfy any of these properties, the notion of constraint relaxation enables it to be *approximate* to a more tractable constraint (Garofalakis 1999). For example, let X be an itemset, $freq(X)$ its support, $count(X)$ its cardinality and $area(X)$ the product of its support and its cardinality. The minimal area constraint of X, $area(X) \geq 6$, is neither monotone, nor anti-monotone. However, $area(X)$ can be relaxed to the conjunction of the constraints $freq(X) \geq 2$ and $count(X) \geq 2$ which are anti-monotone and monotone, respectively (Soulet and Crémilleux 2009).

The concept of relaxation is a classic example of a soft constraint in pattern mining. In Antunes and Oliveira (2004), Bistarelli and Bonchi (2007) constraints are no longer rigid Boolean functions and the proposed frameworks can measure the level of interest of a pattern. The user can choose the level of relaxation allowed. In the field of frequent itemsets, Soulet and Crémilleux (2009) uses relaxations to reduce the search space during the pre-processing step. Wang et al (2005) focuses on relaxing aggregate constraints in itemsets. Soft temporal constraints in the GSP algorithm are studied in Fiot et al. (2009).

5.4 Constraints on interestingness measures

Interestingness constraints specify thresholds on the statistical measures of a pattern. Here, we briefly consider three kinds of interestingness measures: time constraints in sequences, "recency, frequency and monetary" (RFM) constraints, and finally aggregate constraints. See Zhang et al (2009) for a review on the various interestingness measures in ARM. As for the classification task, these constraints involve measures on the extracted model, thus they can be satisfied both during the mining and the post-processing phases.

Several performing frameworks have been proposed with different properties and characteristics of constraints, such as those discussed above. Constraint-driven frameworks are designed to push several kinds of constraints deep into the mining process (Han et al. 1999; Pei and Han 2000) in order to get the maximum gain (Jeudy and Boulicaut 2002; Bonchi et al. 2005; Bonchi and Lucchese 2007; Bucilă et al. 2003; Baralis et al. 2012). A preliminary approach to propose a general framework for constrained frequent itemsets mining is shown in Bonchi and Lucchese (2007). This consists in the design of *ExAMinerGEN*, a general Apriori-like algorithm, which exploits all the possible kinds of constraints previously discussed. The authors report a data reduction algorithm, named *ExAnte* (Bonchi et al. 2005; Bonchi and Lucchese 2007), which exploits the conjunction of monotone and anti-monotone properties. The idea is that a transaction that does not satisfy the given monotonic constraint can be removed from the input dataset, since it will never contribute to the support count. As a consequence, this data filtering implicitly reduces the support of a large amount of itemsets that do not satisfy the monotonic constraint. This results in a reduced number of candidate itemsets generated during the mining algorithm. In addition, for the same anti-monotonic property, infrequent singleton items can be deleted from all transactions in the input database, thus reducing the possibility that a transaction will satisfy a monotonic constraint. Within this iteration, two different kinds of data-reductions cooperate to reduce the search space and the input database. Monotonicity and anti-monotonicity have also been studied in Bucilă et al. (2003), in exploring the search space from the top and from the bottom of the lattice.

Pei et al (2007) is the first systematic study on constraint-based sequential pattern mining. It considers monotonic and succinctness properties in pushing seven different kinds of constraints deep into pattern growth algorithms. It includes constraints on the presence of certain data (*item constraint*); on the number of occurrences of items or the number of transactions (*length constraint*); on a particular subset of patterns (*super-pattern constraint*); on aggregate functions of items (*aggregate constraint*); on constraints expressed as regular expressions over the set of items (*RE constraint*); on time stamp differences between the first and the last itemsets in a sequential pattern (*duration constraint*); and, finally, on the timestamp differences between two adjacent itemsets in the sequential pattern (*gap constraint*).

Recent work on exploiting the preferences within constraint-based data mining can be found in Soulet et al. (2011). The authors propose a recursive pattern mining framework where the mined patterns are recursively redefined until a few relevant patterns are obtained. Another important contribution to constraint-based mining is the concept of flexibility where a set of Boolean basic-flexible constraints for pattern mining

leads to the definition of new more complicated constraints (Soulet and Crémilleux 2005). Boulicaut and Jeudy (2010) proposes a good starting point for reasoning on disjunctions and conjunctions of monotonic and anti-monotonic primitive constraints.

Cerf et al. (2009) considers the most general class of constraints, e.g. piece-wise (anti-) monotonicity, that can be efficiently processed during typical exhaustive enumerations. The authors introduce an algorithm called a Data-Peeler, which extracts all closed n-sets satisfying given piecewise (anti-) monotonic constraints from an n-ary relation. Cerf et al. (2013) extends the previous work considering a fault-tolerant approach also discovering relevant patterns in the presence of noisy data, similarly to Besson et al. (2006) where concepts of monotonicity and anti-monotonicity are extended in order to discover fault-tolerant patterns.

Finally, languages for constraint-based pattern mining have been proposed for both itemsets (Guns et al. 2011) and pattern sets (Guns et al. 2013). Other languages for pattern mining have been proposed following the idea of deductive databases (Bonchi et al. 2009). Guns et al. (2011), Guns et al. (2013) approaches are based on the concept of mathematical programming. Bonchi et al. (2009) follows the idea of declarative languages for data mining. General purpose constraint-based systems for constraints programming are introduced in Sect. 6.1.

Time constraints When dealing with sequences, the user can choose not only the minimum support, but also the time gaps and window size (Srikant and Agrawal 1996; Pei et al 2007; Masseglia et al. 2009). The time gaps enable itemsets to be constrained in a pattern that is neither too close, nor too far w.r.t. time. This happens by constraining the time-stamp difference between two adjacent itemsets to fit in a given range (*gap constraint*). The window size states that the time-stamp difference between the first and the last itemset in a sequential pattern must be longer or shorter than a given threshold (duration constraint). For example, for monthly shopping items that are needed within one month, but more than one week, we can specify the range $(7, 30)$. We can also filter out patterns where the last set of items was purchased at most 10 days after the first one.

The aim of the approaches in the literature is to unify these time constraints (or their variants) into one algorithm. They are based on an extension of the SPADE algorithm (Zaki 2000), the pattern-growth methodology (Lin et al. 2005, 2008; Hirate and Yamana 2006; Antunes and Oliveira 2003), and the GSP algorithm combined with a dataset filtering optimization (Masseglia et al. 2009). As modifications of existing algorithms, all these studies underline the importance of integrating time constraints in either the mining or the earlier stage of the process.

Recency, frequency and monetary constraints The RFM is a model used to predict the behavior of a customer on the basis of historical data, analyzing how often and recently a customer purchases as well as how much he/she spends (Bult and Wansbeek 1995). The RFM includes three main measures (Wei et al 2010). *Recency* is the length of the elapsed period since the last purchase. *Frequency* is the number of purchases within a specified time period. *Monetary* is the amount of money spent in this specified time period.

The concept of RFM was recently included in constrained pattern mining, in order to integrate these three kinds of constraints into the mining process. Recency is specified by ensuring that the last transaction in the mined patterns takes place after a given time threshold w.r.t. the start data of the database. For example, a high value of recency in a supermarket database means that a customer tends to repeat a purchase in the given interval. The monetary constraint is specified by a monetary minimum support, ensuring that the total value of the discovered pattern must be greater than that threshold. Finally, the *frequency of a pattern* is the percentage of transactions satisfying both the recency and the monetary constraints.

Recent works include a modification of the PrefixSpan algorithm by reducing the size and the number of database projections (Chand et al. 2012b; Hu and Kao 2011), an extension of the GSP algorithm (Chen et al. 2009), and the definition of a new pattern growth-based algorithm (Hu and Yen 2010).

Aggregate constraints An aggregate constraint is defined on an aggregation of items in a pattern, where the aggregate function can be sum, avg, max, min, etc. For example, for all association rules where the average price of all the items in the antecedent is over a given threshold. These kinds of constraints cannot be directly handled with existing properties such as monotonicity.

Pei et al. (2004) develops the notion of a convertible constraint, such as $avg(X) \geq 25$, pushing it deep inside the FP(Frequent Pattern)-growth algorithm for frequent itemset mining. The basic idea consists in using an appropriate order of frequent items (e.g. in descending order for the average) in such a way that the ordered list of frequent items satisfies anti-monotonic or monotonic properties. Pei et al (2007) shows that a similar optimization is applicable for sequential pattern mining. Here, the prefix-growth is extended to handle aggregate constraints by establishing a value-ascending ordering over the set of items. Unpromising items are pruned in the first scan of the projected database. Again based on an arrangement of the items, Leung et al. (2010) proposes a tree-based algorithm to mine *uncertain data* under user-specified aggregate constraints. Finally, Chen (2008) reports a prefix-growth based algorithm for aggregate constraints.

6 Declarative approaches to data mining

In this section we consider a complementary approach: instead of incorporating constraints into imperative algorithms, we discuss declarative approaches to express data mining tasks. So far we have discussed how constraints can be embodied into algorithms that are expressed in an imperative way, as summarized in Table 1. Now we consider a more radical approach, i.e. a declarative style in expressing constraints which is also used to express the mining process. In this context all our previous remarks regarding the nature of constraints still hold. However, defining the mining process moves from an algorithmic viewpoint to a declarative one. The formalisms we consider are declarative in nature, so the mining process is described by means of declarative statements and the results are computed by finding a way to satisfy the declarations. One of these approaches relies on constraints programming, which

consists in extending the basic features of constraint solving by specifying the data mining task. A second approach exploits inductive databases, by proposing extensions to declarative query languages again in order to support the specification of mining tasks. Section 6.2 describes how declarative approaches are used to address the problem of specifying the mining process.

This final section also provides a full overview of the main technologies for expressing constraints in a mining task and more generally all the activities regarding the reasoning behind data mining models and the background knowledge. We introduce several aspects of constraining a model, of reasoning about a generalization of knowledge according to parametrized constraints, and of defining and querying background knowledge. On the one hand, we show how existing constraining languages can be employed for solving mining problems, and we highlight the pros and cons of constraint programming with respect to "traditional" imperative programming. We also discuss how the inductive database perspective introduces a set of application technologies in which the data, the mining models and the background knowledge are represented, retrieved, constrained and manipulated in a uniform way.

6.1 Advances in constraint programming for data mining

Many data mining problems can be formalized as combinatorial problems in a declarative way. As shown in the previous sections, tasks such as the discovery of patterns, or finding clusters of similar examples in data, often require constraints to be satisfied and require solutions that are optimal with respect to a given scoring function. In this section, we show how a clustering problem can be modeled using constraint programming techniques, which enables us to incorporate additional constraints (De Raedt et al. 2010).

Constraint programming is a general declarative methodology for solving constraint satisfaction problems. Constraint programs specify what the problem is, rather than outlining how the solution should be computed. It does not focus on a particular application. The user specifies a problem in terms of constraints, and the solver is responsible for finding solutions. The problems that constrain programming systems focus on are constraint satisfaction problems.

A *Constraint Satisfaction Problem* (CSP) $P = (V, D, C)$ is defined by: a finite set of variables V, an initial domain D, which maps every variable $v \in V$ to a set of possible values $D(v)$, and a finite set of constraints C.

Several properties need to be verified by a solver in order to provide a solution:

1. a variable $x \in V$ is called fixed if $|D(x)| = 1$;
2. a domain D is fixed if all its variables are fixed, $\forall x \in V : |D(x)| = 1$;
3. a domain D' is said to be stronger than a domain D if $D'(x) \subseteq D(x)$ for all $x \in V$;
4. a domain is false if an $x \in V$ exists such that $D(x) = \emptyset$;
5. a constraint $C(x_1, \ldots, x_k) \in C$ is an arbitrary Boolean function on variables $\{x_1, \ldots, x_k\} \subseteq V$.

A solution to a CSP is a stronger fixed domain D' than the initial domain D that satisfies all the constraints. This implies that $\forall\ C(x_1, \ldots, x_k) \in C : C(D'(x_1), \ldots, D'(x_k)) = true$.

Constraint programming systems provide declarative modeling languages in which many types of constraints can be expressed and combined. They often support a much wider range of constraints than more specialized systems such as satisfiability (SAT) and integer linear programming (ILP) solvers.

As reported in Guns et al. (2013a), there is limited support for formalizing a data mining task and capturing a problem specification in a declarative way. If we consider traditional algorithmic approaches, developing and implementing the algorithms is quite intensive with only a limited re-use of software. MiningZinc (Guns et al. 2013a) is a language for modeling constraint-based mining problems. It is based on the Zinc Marriott et al. (2008) modeling language for combinatorial problems. It proposes a high-level and natural modeling of mining tasks including user-defined constraints, and it is also solver-independent. It can be considered as an attempt to bring the well-established modeling techniques from the constraint programming community to the field of data mining. The system is currently validated and applied to item-set mining problems (Guns et al. 2013a, b).

While the application of CP approaches to frequent item-set mining has been widely studied in the literature (Guns et al. 2011, 2013), CP models are important for other mining methods such as clustering (Grossi et al. 2015). We thus propose a CP formalization of the *K-medoids* algorithm. The only difference between the K-medoid and the most famous K-means is that while the K-means represents the centroid as the average of all points in the cluster, the K-medoids considers the most representative current data point in the cluster. To find clusters that provide an optimal solution considering a Euclidean metric measure for a general value of k, e.g. minimizing the Sum of Squared Error (SSE), is an NP-Hard problem (Hansen and Aloise 2009).

Recalling (Babaki et al. 2014), given a set of n points, and a k value, the aim is to find a data partition in k clusters that minimizes a function distance based on SSE where for each point, the error is the distance to the nearest cluster. We thus define a binary variable x_{il} indicating whether or not entity i belongs to cluster l. We also introduce a Boolean variable m_{il} to select the specific entity i as a medoid of cluster l. For each cluster l, SSE is then defined as follows:

$$SSE_l = \sum_{i=1}^{n} x_{il} \left(\sum_{h=1}^{n} m_{hl} \left(d\left(i, h\right) \right)^2 \right) \qquad \forall\, 1 \leq l \leq k \tag{1}$$

where $d(i, h)$ is the distance between points i and h.

The formalization of K-medoids via constraints is simpler than the one for K-means since it does not require an iterative updating of the distances given a new centroid, and everything can be easily computed given the distance matrix. Due to the nature of CP solvers, with respect to classical imperative programming, CP implementations do not require an initialization step where a set of initial centroids are randomly selected, and do not require the iteration to be explicitly defined. The CP solver satisfies the problem by selecting a data partitioning that minimizes SSE, and by providing information on

Fig. 1 An example of the MiniZinc constraint for discovering clusters with a minimum number m of elements

```
constraint forall (c in Clusters)(
        sum (p in Points) (
                bool2int(assigned[p] == c)
                        ) >= m
);
```

the optimality of the solution provided.

$$\sum_{l=1}^{k} x_{ij} = 1 \qquad \forall\, 1 \leq i \leq n$$
$$\sum_{i=1}^{n} m_{il} = 1 \qquad \forall\, 1 \leq l \leq k \tag{2}$$

The additional constraints (2) are required to guarantee that each entity will be assigned to exactly one cluster, and that at the same time each cluster will be represented by a real medoid. Finally, given Eq. (1), the objective function can be defined as: $min\ \sum_{l=1}^{k} SSE_l$. Dao et al. (2015) represents another recent example of clustering formalization via CP. The work also takes must and cannot-links in input, and proposes a filtering algorithm to reduce both the domain of the objective variable and also those of the decision variables.

CP guarantees a high level of expressivity, since adding new constraints to the required output is quite easy and natural. Figure 1 proposes an example of a new constraint coded in MiniZinc language.[3] The constraint requires clusters with a minimum number of elements. For methods such as centroid-based clustering (see Sect. 4), given a distance measure to minimize and/or a set of further constraints, CP methods also guarantee an optimal solution. Unfortunately, CP does not scale well since the space search for the optimal clustering partition can be huge. A complete framework for constrained clustering is defined in Dao et al. (2013). Similarly to the proposed approach presented above, the framework in Dao et al. (2015) includes different objective criteria, e.g. including SSE and the extension to maximal cluster diameter, and different data constraints starting from the must and cannot-links presented in Sect. 4.1.

Table 2 presents a brief summary of the main features of constraint programming (CP) and traditional imperative programming (IP). With respect to CP-methods, IP approaches guarantee a high level of scalability and the possibility to manage complex structures. IP methods can also be used when a solution is required even though there is not a best one, or the search space for the optimal one is huge. For example, existing IP solutions to the clustering problem, including the K-medoid, trade the optimality of the solution with shorter run times. A key problem in doing this is that not only is the solution not always optimal, but the method does not provide any indication of

[3] MiniZinc is a constraint-modeling language. It is sufficiently high-level to express most constraint problems easily, but low-level enough to be mapped onto existing solvers easily and consistently (Nethercote et al. 2007). The MiningZinc language (Guns et al. 2013b) cited in this section is an extension of MiniZinc for data mining.

Table 2 CP versus IP

	Pros	Cons
CP	Optimal solution	Scalability
	Expressivity	Feasibility
IP	Find a solution	Expressivity
	Feasibility (managing complex structure)	No guarantees on optimality

how close it is to the optimum, nor any guaranteed bound of the error introduced by the heuristics.

On the other hand, CP approaches propose a more natural way of expressing and adding new constrains to the model, and provide guarantees regarding the optimality of the solution discovered. Since CP solvers explore the whole space search, they have scalability problems, and are useless when there is a medium/large quantity of data. In any case, the CP approach can be used in all those applications where only small data sets are available and an optimal/accurate solution is required, considering the runtime performance as secondary.

6.2 Advances in inductive databases

As reported in Dzeroski et al. (2010), inductive databases and constraint-based data mining are at the intersection of data mining and database research. Solving complex mining tasks requires high expressiveness both in terms of the definition of mining methods and in the exploitation of the background knowledge. Constraint programming provides a good basis for high expressiveness, but it does not exploit the background knowledge.

Let us consider an example. Suppose that a set of records consists of both the textual and citation information of a group of scientific articles. The goal is to perform a co-citation analysis using a clustering approach, e.g. by first calculating the similarity between pairs of reference papers using co-citation counts, and then by extracting clusters of papers.

- A *data constraint* can be used to limit the search to those articles containing some specific term, e.g. data mining, in the abstract section of a paper.
- A *model constraint* can be used to force the algorithm to discover clusters with a minimum number of elements.

We can now enrich our data with an ontology on the authors. This consists in information on the affiliation of the researcher in terms of department, university, city and country. The way the distance measure is computed can be constrained by this knowledge domain. Given a set of constraints, the distance measure can be "trained" to satisfy the given set of constraints. For example, the similarity function can be adapted to consider higher levels of researcher affiliations, involving only macro areas, e.g. considering the "European Community", which can only be defined in the ontology. This

constraint, expressed in the background knowledge, *must* be pushed into the mining algorithm (e.g. K-means).

Although CP provides functionality for filtering or propagating constraints, the CP modeling languages do not have advanced constructs/tools for formalizing these constraints, since they are based on the facts represented in the ontology. As shown in De Raedt (2002), Inductive Databases (IDBs) are one way to introduce and express constraints in data mining applications. IDBs not only store data but also mining models by introducing declarative querying possibilities.

Below, we review the IDBs and then give an example of how constraints can be represented and encapsulated in a mining process.

Inductive Databases IDBs are general-purpose databases in which the data, mining models, constraints and domain knowledge are represented, retrieved, and manipulated in a uniform way (Imielinski and Mannila 1996; Boulicaut and Masson 2005). Embedding the data mining in a uniform framework facilitates the exploitation of induction and deduction in solving problems. In fact, both pre-processing, and especially post- processing, require some deductive reasoning either on the data to be fed to the inductive data mining step, or on the synthetic models produced by the mining step. For example, pre-processing may require the verification of certain domain rules on the data in order to filter them. On the other hand, forms of meta-reasoning on the induced models can be useful in many cases, for example to compare two models extracted from two different sets of homogeneous data.

A critical aspect is the choice of the most suitable formalism to represent models, data sources, background knowledge, as well as the queries needed to apply to them. Given that data mining can be considered as a form of querying, it is natural to embed it in declarative database approaches, such as the following.

- A *logic-based Query Language (QL)* uses a logic formalism for developing KDD applications, and a deductive DBMS for storing both models and raw data. The focus is generally on datalog-based or Prolog-based languages. Their high expressiveness also makes them useful to represent background knowledge (Giannotti 2000).
- An *SQL-based QL* includes the development of a mining tool integrated with a relational DBMS. According to the closure principle,[4] both the source data and the induced models are represented as database relations. Queries are specified in an SQL-like language (Meo et al. 1998; Masson et al. 2004).
- An *XML-based QL* uses XML in order to represent the models, the data sources and the background knowledge by extending a language such as XQuery with mining primitives (Romei and Turini 2010).

The idea is to view knowledge discovery as an extension of a querying process. We focus on SQL- and XML-based QL since they have attracted the greatest research interest - see Dzeroski et al. (2010) and Romei and Turini (2011) for a full overview of the state-of-the art (and comparative study) of IDBs.

[4] Inductive databases extend the closure principle to the knowledge discovery field. The principle simply states that the output of a query for knowledge extraction can be the input of another query of a compatible type (Imielinski and Mannila 1996).

6.2.1 SQL-based languages for data mining

The idea of integrating knowledge discovery into a relational DBMS has been addressed in many papers (Bernhardt et al. 2001; Ceri et al. 1998; Fu et al. 1996; Imielinski and Virmani 1999; Morzy and Zakrzewicz 1997; Baralis et al. 2005; Bonchi et al. 2009; Richter et al. 2008; Blockeel et al. 2012), most of which focus on the ability to satisfy the closure property, a critical aspect in inductive databases.

The trend in SQL-based languages is twofold. On the one hand, past experiences in query optimizations have shown that it is very difficult to extend the relational environments to handle powerful optimizations and data structures (Sarawagi et al 2000). In order to overcome these limitations, commercial and industrial database vendors are taking an alternative approach. Microsoft OLE DB for DM (Bernhardt et al. 2001) achieves a closer integration and a better interoperability of the mining task by combining predictive and descriptive mining. On the academic side, ATLaS (Law et al. 2004) handles data streams. ATLaS obtains a good degree of scalability in the implementation of mining algorithms via a tightly-coupled approach. In addition, it overcomes the lack of extensibility of the other SQL-based approaches by extending SQL with user-defined aggregates.

While association rules have been extensively studied in SQL-based approaches (Ceri et al. 1998; Imielinski and Virmani 1999; Baralis et al. 2005; Bonchi et al. 2009), other kinds of mining tasks, such as classification or clustering, have received little attention, mainly due to the difficulty in efficiently representing mining models via relational tables. Researchers have tried to overcome this limitation by presenting relational-based views of decision trees (Bentayeb and Darmont 2002; Fromont et al. 2006). In the inductive query language discussed in Richter et al. (2008), the goal is to provide rudimentary support for many different mining operations, including discretization, feature selection, frequent patterns, clustering, and classification.

Finally, Blockeel et al. (2012), Blockeel et al. (2008a, b) propose integrating data mining into relational database systems without extending the query language. These relations are called *virtual mining views*. They focus more on the semantics of learned models rather than their structure, thus they handle conceptual models such as decision trees and clusters in a general way. More specifically, they integrate data mining models into relational database systems without extending the query language. They extend the database schema with new tables that contain descriptive or predictive models. The cited works show how several types of patterns and models on the data, such as itemsets, association rules and decision trees, can be represented and queried using a unifying framework.

6.2.2 XML-based languages for data mining

An important aspect when dealing with IDBs is how to make all the heterogeneous patterns, sources of data and other KDD objects coexist in a single framework. A recent solution is the exploitation of XML as a flexible and extensible instrument for IDBs (Meo and Psaila 2006; Romei et al 2006; Euler et al. 2006; Baralis et al. 2007; Romei and Turini 2010).

In Meo and Psaila (2006) XML is used for a semi-structured data model designed for KDD, called XDM. It explicitly represents via XML *(i)* the data item, i.e. a container of data and patterns and *(ii)* a statement, i.e. a description of an operator application. The KDD process is represented as a set of relationships between data items and statements. RapidMiner (Euler et al. 2006) is an open-source environment for KDD and machine learning in which experiments are described via XML files created with a graphical user interface. While the graphical user interface supports interactive design, the underlying XML representation enables automated applications after the prototyping phase. Thus, KDD processes are modeled as trees. The perspective suggested by XDM and RapidMiner is also taken in XML-based KDDML (Romei et al 2006). Mining XML data is used in an instrumental way in Baralis et al. (2007) to construct summarized representations of XML data. The authors propose extracting association rules from XML databases as the basis for a pattern based representation of XML datasets.

Finally, XQuake (Romei and Turini 2010) is a new contribution to mining XML data. In contrast with similar works, XQuake uses native XML databases to store both models and data, while an extension of the XQuery language is used to express the KDD process. Extensions of the original study have been proposed to show its expressiveness in representing the KDD process as an XQuery program (Romei and Turini 2011), in integrating several kinds of constraints into the mining process (Romei and Turini 2010), and in supporting domain constraints via OWL ontologies (Grossi and Romei 2012).

To clarify these concepts, we complete this section with an example, taken from Grossi and Romei (2012), which integrates constraints expressed on the domain knowledge into a knowledge extraction process. We try to answer three different questions: *how to represent the background knowledge*, and *how and when to satisfy the constraints on the background knowledge*. Such questions deal with the way in which the background knowledge is represented, stored and then queried. In Sect. 6.2.3, we examine how these aspects have been addressed in the literature.

6.2.3 Case study: using XQuake for the complete mining process

XQuake is a system for programming knowledge extraction over native XML databases, in the spirit of inductive databases. First, it satisfies the closure principle, since both data and mining models are coded as XML structures. It also represents the KDD process in a declarative way, by means of an XQuery program extended with mining primitives, and finally it satisfies constraints via XQuery predicates.[5] XQuake supports knowledge extraction by providing a unified view for querying and constraining data and models—see Romei and Turini (2010) for a detailed description.

Even when the set of constraints is known, providing both a language for formalizing them and a system that can efficiently process their properties is challenging. For example, the extraction of association rules implies the discovery of a large quantity of useless rules. The aim is to enrich our data with semantic information enabling the

[5] The system automatically captures the properties of such constraints (e.g. monotonicity and anti-monotonicity), to be used directly during the extraction of the mining model.

```
<MBA>
 <store @id="id00193">
        <purchase @date="05/01/2012">
               <item @quantity="2" @price="3.5">vodka lemon</item>
               <item @quantity="1" @price="1.5">bread</item>
               ...
        </purchase>
          ...
 </store>
   ...
</MBA>
```

Fig. 2 XML fragment shows data on the items purchased in a supermarket; taken from Grossi and Romei (2012)

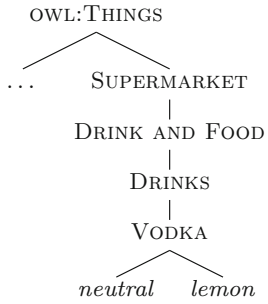
definition of constraints based on this background knowledge, and then to provide a fully integrated system that can process data and constraints (Marinica and Guillet 2010b; Hwang and Gu 2014). Background knowledge enriches knowledge extraction by providing a semantic level to the data under analysis, and by enabling the user to define domain constraints. The use of constraints is related to knowledge extraction at different levels, since it defines the problem in a declarative way, and forces the analyst to model mining problems by specifying desirable properties of the extracted knowledge.

Representing Background Knowledge There are several alternatives for representing the background knowledge. Knowing how to logically and physically organize and represent our knowledge domain is very important. This can be represented by a simple hierarchy, a set of relational tables, or as an ontology. In our view, an ontological model provides several data features that do not change over time.

Figure 2 shows an XML fragment representing "raw data" on the items purchased in a supermarket. It simply reports the list and the quantity of items bought together by the user on a specific day.

Figure 3 shows the semantic information related to a feature of a specific item. In this example, data are enriched with background knowledge represented by an OWL document containing the properties related to each item and their hierarchical organization. Also note the separation between the raw data and domain specification. On the one hand we have an OWL-coded file representing the domain knowledge, for example an ontology. On the other, we have actual data for example stored in a DB or in text file. Thus, adding or modifying a new concept in the ontology does not imply any change to the database transactions (Bellandi et al. 2008; Marinica and Guillet 2010b).

Querying Background Knowledge A means for querying the background information is needed in order to exploit the facts stored in the domain in a knowledge extraction process. The choice of which language to query the background knowledge is clearly



attribute	value
hasColour	*neutral*
hasAlcoholic	*high*
hasFlavour	*lemon*
hasBrand	*Best*
hasSize	*70 cl*
hasPrice	*EUR 7.56*
hasPromotion	*Christmas*

Fig. 3 A fragment of the hierarchical structure of `items.owl` (*left*). Data properties for Vodka Lemon (*right*)

```
declare function local:hasRecurrence($item, $class, $prop)
                 as xs:boolean {
         let $owl := owldoc("items.owl")
         return sw:hasSuperclass($owl, $item, $class) and
                  sw:hasProperty($owl, $item, $prop, "hasRecurrency")
 };
```

Fig. 4 XQuery function

related to the way we represent it. For example, let us extract the following association rules:

($i_1 \in$ EasterDrink) and ($i_2 \in$ AnyItem) and ... and ($i_n \in$ AnyItem) $\Rightarrow$ ($i_{n+1} \in$ ChristmasCake) [supp] [conf]

where `EasterDrink` (resp. `ChristmasCake`) is the class of items that are drinks (resp. cakes) with an Easter (resp. Christmas) promotion, and `AnyItem` is the entire set of distinct items. Grossi and Romei (2012) proposes to express constraints through XQuery expressions supported by a built-in library. An example of such an XQuery function is reported in Fig. 4.

The function returns true if a frequent item `$item()` belongs to the superclass `$class` and has data property `$prop` in the ontology of Figure 3. Calling such a function makes it possible to apply the above constraint to several concepts of the hierarchy - see Grossi and Romei (2012) for details.

Background Knowledge in the KDD process Knowledge extraction from data is a complex process including a chain of steps such as pre-processing, mining and reasoning on the extracted knowledge. The introduction of constraints requires a way of satisfying them during the knowledge extraction. This is crucial both for the quality of the extracted data mining model, and for the scalability of the entire process. On the one hand an analyst can define the phase where the constraint must be satisfied, explicitly. On the other, an optimizer is required to understand where a constraint must be satisfied inside the process flow, in an automatic way.

We are now interested in extracting association rules with, in the consequent, at least one item belonging to the `drink` concept. Such a predicate is clearly monotone, since if it holds for a consequent C, it also holds for any superset of C. The monotonicity property can be automatically recognized during the query compilation (Romei and Turini 2011), and can be used to prune the search space during the invocation of the mining algorithm (Bonchi and Lucchese 2007).

7 Conclusions

The use of constraints is an important and challenging task for the data mining community. Current approaches need to be radically redesigned in order to define and satisfy constraints throughout the whole knowledge extraction process, also with the support of domain knowledge.

We have proposed a framework to categorize the approaches present in the literature that offers the possibility of looking at the use of constraints from different angles. The dimensions that we have chosen are the objects on which the constraints are defined (*data*, *models*, *measures*), the mining task the constraints are used for (*classification*, *clustering*, *pattern mining*), the phase of the mining process in which the constraints are used (*pre-processing*, *mining*, *post-mining*), and the nature of the constraints (*hard*, *soft*).

Most of the approaches proposed in the literature deal with constraints and their use in a restricted view. We hope that this survey will enable researchers to see which approaches have some common strategies, so that a cross fertilization may spring out.

The categorization discussed above has been instantiated with proposals from the literature where the use of constraint satisfaction is embedded into data mining implemented in an algorithmic way. The second part of the paper considers another, more radical approach, where constraints are also used to describe the mining process. We thus move from an approach in which the algorithms that implement the mining process use constraint satisfaction as a tool to improve their performance, to an approach in which the whole mining process is implemented as the search for a solution to a problem described in a declarative way.

This second approach still poses performance problems, however we believe that it has a theoretical appeal that may be of interest to and bring together the communities of constraint programming and deductive and inductive databases.

Acknowledgements This work was supported by the European Commission under the project "*Inductive Constraint Programming (ICON)*" contract number FP7-284715, and by a Grant for "*Big Data Social Mining*" of the University of Pisa. We warmly thank the anonymous referees for their very valuable suggestions.

References

Agrawal R, Srikant R, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international conference on management of data, pp 207–216

Ahmed CF, Tanbeer SK, Jeong BS, Lee YK, Choi HJ (2012) Single-pass incremental and interactive mining for weighted frequent patterns. Expert Syst Appl 39(9):7976–7994

An A, Stefanowski J, Ramanna S, Butz CJ, Pedrycz W, Wang G (eds) (2007) Rough sets, fuzzy sets, data mining and granular computing. In: Proceedings of the 11th international conference, RSFDGrC 2007, Toronto, Canada, May 14–16, 2007, (Lecture Notes in Computer Science), vol 4482. Springer

Antunes C (2009) Pattern mining over star schemas in the Onto4AR framework. In: Proceedings of the IEEE international conference on data mining (ICDM) workshops, pp 453–458

Antunes C, Oliveira AL (2003) Sequence mining in categorical domains: Incorporating constraints. In: Proceedings of the 3th international conference on machine learning and data mining in pattern recognition (MLDM), pp 239–251

Antunes C, Oliveira A (2004) Constraint relaxations for discovering unknown sequential patterns. In: Proceedings of the third international workshop on knowledge discovery in inductive databases (KDID), pp 11–32

Babaki B, Guns T, Nijssen S (2014) Constrained clustering using column generation. In: Simonis H (ed) Integration of AI and OR techniques in constraint programming: proceedings of the 11th international conference, CPAIOR 2014, Cork, Ireland, May 19–23, 2014. Lecture Notes in Computer Science, vol 8451, pp. 438–454. Springer. doi:10.1007/978-3-319-07046-9_31

Bade K, Nürnberger A (2006) Personalized hierarchical clustering. In: IEEE/ACM international conference on web intelligence (WIC), pp 181–187

Bade K, Nürnberger A (2008) Creating a cluster hierarchy under constraints of a partially known hierarchy. In: Proceedings of the SIAM international conference on data mining (SDM), pp 13–24

Banerjee A, Ghosh J (2006) Scalable clustering algorithms with balancing constraints. Data Min Knowl Discov 13(3):365–395

Banerjee A, Ghosh J (2008) Clustering with balancing constraints. Constrained clustering: advances in algorithms, theory, and applications. Chapman and Hall/CRC, Boca Raton, pp 171–200

Baralis E, Garza P, Quintarelli E, Tanca L (2007) Answering XML queries by means of data summaries. ACM Trans Inf Syst J 25(3):10–16

Baralis E, Cagliero L, Cerquitelli T, Garza P (2012) Generalized association rule mining with constraints. Inf Sci 194:68–84

Baralis E, Cerquitelli T, Chiusano S (2005) Index support for frequent itemset mining in a relational DBMS. In: Proceedings of the 21st international conference on data engineering (ICDE), pp 754–765

Bar-Hillel A, Hertz T, Shental N, Weinshall D (2003) Learning distance functions using equivalence relations. In: Proceedings of the twentieth international conference on machine learning (ICML), pp 11–18

Basu S, Davidson I, Wagstaff KL (2008) Constrained clustering: advances in algorithms, theory, and applications. Chapman and Hall/CRC, Boca Raton

Basu S, Banerjee A, Mooney RJ (2004a) Active semi-supervision for pairwise constrained clustering. In: Proceedings of the Fourth SIAM international conference on data mining (SDM)

Basu S, Bilenko M, Mooney RJ (2004b) A probabilistic framework for semi-supervised clustering. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 59–68

Bellandi A, Furletti B, Grossi V, Romei A (2007) Ontology-driven association rules extraction: a case study. In: Proceedings of the international workshop on contexts and ontologies: representation and reasoning (C&O:RR), pp 1–10

Bellandi A, Furletti B, Grossi V, Romei A (2008) Ontological support for association rule mining. In: Proceedings of the 26th IASTED international conference on artificial intelligence and applications (AIA), AIA '08. ACTA Press, Anaheim, pp 110–115. http://dl.acm.org/citation.cfm?id=1712759.1712781

Bentayeb F, Darmont J (2002) Decision tree modeling with relational views. In: Proceedings of the 13th international symposium on foundations of intelligent systems (ISMIS), pp 423–431

Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517

Bernhardt J, Chaudhuri S, Fayyad U, Netz A (2001) Integrating data mining with SQL databases: OLE DB for data mining. In: Proceedings of the 17th international conference on data engineering (ICDE), pp 379–387

Bernstein A, Mannor S, Shimkin N (2010) Online classification with specificity constraints. In: Proceedings of the 24th annual conference on neural information processing systems (NIPS), pp 190–198

Bertsekas DP (1991) Linear network optimization: algorithms and codes. MIT Press Cambridge. http://opac.inria.fr/record=b1089011

Besson J, Pensa RG, Robardet C, Boulicaut JF (2006) Knowledge discovery in inductive databases: 4th international workshop, KDID 2005, Porto, Portugal, October 3, 2005, Revised selected and invited papers, chap. Constraint-based mining of fault-tolerant patterns from boolean data. Springer, Berlin Heidelberg, pp 55–71. doi:10.1007/11733492_4

Bilenko M, Basu S, Mooney RJ (2004) Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the twenty-first international conference on machine learning (ICML), ICML '04. ACM, New York, pp. 11–18. doi:10.1145/1015330.1015360

Bistarelli S, Montanari U, Rossi F (1997) Semiring-based constraint satisfaction and optimization. J ACM 44(2):201–236. doi:10.1145/256303.256306

Bistarelli S, Bonchi F (2007) Soft constraint based pattern mining. Data Knowl Eng 62(1):118–137

Blaszczynski J, Deng W, Hu F, Slowinski R, Szelag M, Wang G (2012) On different ways of handling inconsistencies in ordinal classification with monotonicity constraints. In: Greco S, Bouchon-Meunier B, Coletti G, Fedrizzi M, Matarazzo B, Yager RR (eds) Advances on computational intelligence: 14th international conference on information processing and management of uncertainty in knowledge-based systems, IPMU 2012, Catania, Italy, July 9–13, 2012. Proceedings, Part I, communications in computer and information science, vol 297. Springer, pp 300–309. doi:10.1007/978-3-642-31709-5_31

Blaszczynski J, Slowinski R, Szelag M (2010) Probabilistic rough set approaches to ordinal classification with monotonicity constraints. In: Computational intelligence for knowledge-based systems design, 13th international conference on information processing and management of uncertainty, IPMU 2010, pp 99–108

Blockeel H, Calders T, Fromont É, Goethals B, Prado A, Robardet C (2012) An inductive database system based on virtual mining views. Data Min Knowl Discov 24(1):247–287

Blockeel H, Calders T, Fromont É, Goethals B, Prado A (2008a) Mining views: database views for data mining. In: Alonso G, Blakeley JA, Chen ALP (eds) Proceedings of the 24th international conference on data engineering, ICDE 2008, April 7–12, 2008, Cancún, México. IEEE computer society, pp 1608–1611. doi:10.1109/ICDE.2008.4497633

Blockeel H, Calders T, Fromont É, Goethals B, Prado A, Robardet C (2008b) An inductive database prototype based on virtual mining views. In: Li Y, Liu B, Sarawagi S (eds) Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, Las Vegas, NV, August 24–27, 2008. ACM, pp 1061–1064. doi:10.1145/1401890.1402019

Bonchi F, Giannotti F, Mazzanti A, Pedreschi D (2005) ExAnte: a preprocessing method for frequent-pattern mining. IEEE Intell Syst 20(3):25–31

Bonchi F, Giannotti F, Lucchese C, Orlando S, Perego R, Trasarti R (2009) A constraint-based querying system for exploratory pattern discovery. Inf Syst 34(1):3–27

Bonchi F, Lucchese C (2007) Extending the state-of-the-art of constraint-based pattern discovery. Data Knowl Eng 60(2):377–399

Boulicaut J, Jeudy B (2010) Constraint-based data mining. In: Maimon O, Rokach L (eds) Data mining and knowledge discovery handbook, 2nd edn. Springer, New York. doi:10.1007/978-0-387-09823-4_17

Boulicaut JF, Masson C (2005) Data mining query languages. In: Maimom O, Rokach L (eds) The data mining and knowledge discovery handbook. Springer, New York, pp 715–727

Bradley PS, Bennett KP, Demiriz A (2000) Constrained k-means clustering. In: Technical report, MSR-TR-2000-65, Microsoft Research

Brunner C, Fischer A, Luig K, Thies T (2012) Pairwise support vector machines and their application to large scale problems. J Mach Learn Res 13(1): 2279–2292. http://dl.acm.org/citation.cfm?id=2503308.2503316

Bucilă C, Gehrke J, Kifer D, White W (2003) DualMiner: a dual-pruning algorithm for itemsets with constraints. Data Min Knowl Discov 7(3):241–272

Bult JR, Wansbeek TJ (1995) Optimal selection for direct mail. Market Sci 14(4):378–394

Capelle M, Masson C, Boulicaut J (2003) Mining frequent sequential patterns under regular expressions: a highly adaptive strategy for pushing constraints. In: Proceedings of the third SIAM international conference on data mining (SDM), pp 316–320

Cerf L, Besson J, Robardet C, Boulicaut J (2009) Closed patterns meet n-ary relations. ACM Trans Knowl Discov Data (TKDD). doi:10.1145/1497577.1497580

Cerf L, Besson J, Nguyen K, Boulicaut J (2013) Closed and noise-tolerant patterns in n-ary relations. Data Min Knowl Discov 26(3):574–619. doi:10.1007/s10618-012-0284-8

Ceri S, Meo R, Psaila G (1998) An extension to SQL for mining association rules. Data Min Knowl Discov 2(2):195–224. doi:10.1023/A:1009774406717
Chand C, Thakkar A, Ganatra A (2012a) Sequential pattern mining: survey and current research challenges. Int J Soft Comput Eng (IJSCE) 2(1):2231–2307
Chand C, Thakkar A, Ganatra A (2012b) Target oriented sequential pattern mining using recency and monetary constraints. Int J Comput Appl 45(10):12–18
Chang JH (2011) Mining weighted sequential patterns in a sequence database with a time-interval weight. Knowl Based Syst 24(1):1–9
Chen E, Cao H, Li Q, Qian T (2008) Efficient strategies for tough aggregate constraint-based sequential pattern mining. Inf Sci 178(6):1498–1518
Chen YL, Kuo MH, yi Wu S, Tang K (2009) Discovering recency, frequency, and monetary (RFM) sequential patterns from customers' purchasing data. Electron Commer Res Appl 8(5):241–251
Coleman T, Saunderson J, Wirth A (2008) Spectral clustering with inconsistent advice. In: Proceedings of the twenty-fifth international conference on machine learning (ICML), pp 152–159
Costa JA, Iii AOH (2005) Classification constrained dimensionality reduction. In: Proceedings of the IEEE international conference on acoustics, speech, and signal processing (ICASSP), pp 1077–1080
Dao TBH, Duong KC, Vrain C (2013) A declarative framework for constrained clustering. In: Blockeel H, Kersting K, Nijssen S, Zelezn F (eds) ECML/PKDD (3), Lecture Notes in Computer Science, vol 8190. Springer, pp 419–434. doi:10.1007/978-3-642-40994-3
Dao TBH, Duong KC, Vrain C (2015) Constrained minimum sum of squares clustering by constraint programming. In: Proceedings of the 21st international conference on principles and practice of constraint programming (CP 2015). Cork, Ireland, pp 557–573. https://hal.archives-ouvertes.fr/hal-01168193
Davidson I, Ravi SS (2005a) Agglomerative hierarchical clustering with constraints: theoretical and empirical results. In: Knowledge discovery in databases: PKDD 2005, 9th European conference on principles and practice of knowledge discovery in databases (PKDD), pp 59–70
Davidson I, Ravi SS (2005b) Clustering with constraints: feasibility issues and the k-means algorithm. In: Kargupta H, et al. (eds) Proceedings of the 2005 SIAM international conference on data mining, pp 138–149. doi:10.1137/1.9781611972757.13
Davidson I, Ravi SS (2006) Identifying and generating easy sets of constraints for clustering. In: Proceedings of the twenty-first national conference on artificial intelligence and the eighteenth innovative applications of artificial intelligence conference (AAAI), pp 336–341
Davidson I, Ravi SS (2007) The complexity of non-hierarchical clustering with instance and cluster level constraints. Data Min Knowl Discov 14(1):25–61
Davidson I, Ravi SS (2009) Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. Data Min Knowl Discov 18(2):257–282
Davidson I, Wagstaff K, Basu S (2006) Measuring constraint-set utility for partitional clustering algorithms. In: Knowledge discovery in databases: PKDD 2006, 10th European conference on principles and practice of knowledge discovery in databases (PKDD), pp 115–126
Dawson S, di Vimercati SDC, Samarati P (1999) Specification and enforcement of classification and inference constraints. In: IEEE symposium on security and privacy, pp 181–195
De Raedt L, Guns T, Nijssen S (2010) Constraint programming for data mining and machine learning. In: Fox M, Poole D (eds) Proceedings of the twenty-fourth AAAI conference on artificial intelligence, AAAI 2010, Atlanta, July 11–15, 2010. AAAI Press, pp 1671–1675. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1837
De Raedt L (2002) A perspective on inductive databases. SIGKDD Explor 4(2):69–77. doi:10.1145/772862.772871
Demiriz A, Bennett KP, Bradley PS (2008) Using assignment constraints to avoid empty clusters in k-means clustering. Constrained clustering: advances in algorithms, theory, and applications. Chapman and Hall/CRC, Boca Raton, pp 201–220
Druck G, Mann GS, McCallum A (2008) Learning from labeled features using generalized expectation criteria. In: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval (SIGIR), pp 595–602
Duivesteijn W, Feelders A (2008) Nearest neighbour classification with monotonicity constraints. Mach Learn Knowl Discov Databases Eur Conf ECML/PKDD 2008:301–316
Dzeroski S, Goethals B, Panov P (2010) Inductive databases and constraint-based data mining. Springer, New York

Euler T, Klinkenberg R, Mierswa I, Scholz M, Wurst M (2006) YALE: rapid prototyping for complex data mining tasks. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 935–940

Fawcett T (2006) An introduction to roc analysis. Pattern Recognit Lett 27(8):861–874

Fiot C, Laurent A, Teisseire M (2009) Softening the blow of frequent sequence analysis: soft constraints and temporal accuracy. Int J Web Eng Technol 5(1):24–47

Fromont É, Blockeel H, Struyf J (2006) Integrating decision tree learning into inductive databases. In: Proceedings of the 5th international workshop on knowledge discovery in inductive databases (KDID), pp 81–96

Fu Y, Han J (1995) Meta-rule-guided mining of association rules in relational databases. In: Proceedings of the post-conference workshops on integration of knowledge discovery in databases with deductive and object-oriented databases (KDOOD/TDOOD), pp 39–46

Fu Y, Han J, Koperski K, Wang W, Zaiane O (1996) DMQL: a data mining query language for relational databases. In: Proceedings of the first workshop on research issues in data mining and knowledge discovery (DMKD), pp 122–133

Garofalakis MN, Rastogi R, Shim K (1999) SPIRIT: Sequential pattern mining with regular expression constraints. In: Proceedings of 25th international conference on very large data bases (VLDB), pp 223–234

Garofalakis MN, Hyun D, Rastogi R, Shim K (2003) Building decision trees with constraints. Data Min Knowl Discov 7(2):187–214

Giannotti F, Nanni M, Pedreschi D (2000) Logic-based knowledge discovery in databases. In: Proceedings of tenth European–Japanese conference on information modelling and knowledge bases (EJC), pp 279–283

Gilpin S, Davidson I (2011) Incorporating SAT solvers into hierarchical clustering algorithms: an efficient and flexible approach. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 1136–1144

Grossi V, Monreale A, Nanni M, Pedreschi D, Turini F (2015) Software engineering and formal methods: SEFM 2015 collocated workshops: ATSE, HOFM, MoKMaSD, and VERY*SCART, York, UK, September 7–8, 2015. Revised selected papers, chap. clustering formulation using constraint optimization. Springer, Berlin, Heidelberg, pp 93–107. doi:10.1007/978-3-662-49224-6_9

Grossi V, Romei A (2012) XQuake as a constraint-based mining language. In: Proceedings of the ECAI 2012 workshop on combining constraint solving with mining and learning (CoCoMile), pp 90–91

Gu W, Chen B, Hu J (2010) Combining binary-svm and pairwise label constraints for multi-label classification. In: Proceedings of the IEEE international conference on systems, man and cybernetics (SMC), pp 4176–4181

Guns T, Nijssen S, De Raedt L (2011) Itemset mining: a constraint programming perspective. Artif Intell 175(12–13):1951–1983

Guns T, Nijssen S, De Raedt L (2013) k-Pattern set mining under constraints. IEEE Trans Knowl Data Eng 25(2):402–418

Guns T, Dries A, Tack G, Nijssen S, De Raedt L (2013a) Miningzinc: a modeling language for constraint-based mining. In: Rossi F (ed) IJCAI 2013, proceedings of the 23rd international joint conference on artificial intelligence, Beijing, China, August 3–9, 2013. IJCAI/AAAI. http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6947

Guns T, Dries A, Tack G, Nijssen S, De Raedt L (2013b) The miningzinc framework for constraint-based itemset mining. In: Ding W, Washio T, Xiong H, Karypis G, Thuraisingham BM, Cook DJ, Wu X (eds) 13th IEEE international conference on data mining workshops, ICDM workshops, TX, December 7–10, 2013. IEEE computer society, pp 1081–1084. doi:10.1109/ICDMW.2013.38

Han J, Lakshmanan LVS, Ng RT (1999) Constraint-based multidimensional data mining. IEEE Comput 32(8):46–50

Han J, Cheng H, Xin D, Yan X (2007) Frequent pattern mining: current status and future directions. Data Min Knowl Discov 15(1):55–86

Han J, Fu Y (1999) Mining multiple-level association rules in large databases. IEEE Trans Knowl Data Eng 11(5):798–805

Han J, Kamber M (2012) Data mining: concepts and techniques, 3rd edn. Morgan Kaufmann, San Francisco

Hansen P, Aloise D (2009) A survey on exact methods for minimum sum-of-squares clustering, pp 1–2. http://www.math.iit.edu/Buck65files/msscStLouis.pdf

Har-Peled S, Roth D, Zimak D (2002) Constraint classification: a new approach to multiclass classification. In: Proceedings of the 13th international conference algorithmic learning theory (ALT), pp 365–379
Hirate Y, Yamana H (2006) Generalized sequential pattern mining with item intervals. J Comput 1(3):51–60
Hu YH, Kao YH (2011) Mining sequential patterns with consideration to recency, frequency, and monetary. In: Proceedings of the Pacific Asia conference on information systems (PACIS), pp 78–91
Hu YH, Yen TW (2010) Considering RFM-values of frequent patterns in transactional databases. In: Proceedings of the 2th international conference on software engineering and data mining (SEDM), pp 422–427
Hwang JH, Gu MS (2014) Ontology based service frequent pattern mining. Future Inf Technol 309:809–814. doi:10.1007/978-3-642-55038-6-123
Imielinski T, Mannila H (1996) A database perspective on knowledge discovery. Commun ACM 39(11):58–64
Imielinski T, Virmani A (1999) MSQL: a query language for database mining. Data Min Knowl Discov 2(4):373–408
Jeudy B, Boulicaut JF (2002) Optimization of association rule mining queries. Intell Data Anal 6(4):341–357
Kestler H, Kraus J, Palm G, Schwenker F (2006) On the effects of constraints in semi-supervised hierarchical clustering. In: Schwenker F, Marinai S (eds) Artificial neural networks in pattern recognition, vol 4087., Lecture notes in computer scienceSpringer, Berlin, heidelberg, pp 57–66
Klein D, Kamvar SD, Manning CD (2002) From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. In: Proceedings of the nineteenth international conference on machine learning, ICML '02. Morgan Kaufmann Publishers Inc., San Francisco, CA, pp 307–314. http://dl.acm.org/citation.cfm?id=645531.655989
Kumar N, Kummamuru K (2008) Semisupervised clustering with metric learning using relative comparisons. IEEE Trans Knowl Data Eng 20(4):496–503
Kummamuru K, Krishnapuram R, Agrawal R (2004) Learning spatially variant dissimilarity (svad) measures. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 611–616
Lakshmanan LVS, Ng R, Han J, Pang A (1999) Optimization of constrained frequent set queries with 2-variable constraints. ACM SIGMOD Rec 28(2):157–168. doi:10.1145/304181.304196
Lange TCMH, Anil L, Jain K, Buhmann JM (2005) Learning with constrained and unlabeled data. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 731–738
Law MHC, Topchy AP, Jain AK (2005) Model-based clustering with probabilistic constraints. In: Kargupta et al., pp 641–645. doi:10.1137/1.9781611972757.77
Law MHC, Topchy A, Jain AK (2004) Structural, syntactic, and statistical pattern recognition: joint IAPR international workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18–20, 2004. Proceedings, chap. Clustering with soft and group constraints. Springer, Berlin, Heidelberg, pp 662–670. doi:10.1007/978-3-540-27868-9_72
Law Y, Wang H, Zaniolo C (2004) Query languages and data models for database sequences and data streams. In: Proceedings of the 30th international conference on very large data bases (VLDB), pp 492–503
Leung CKS, Hao B, Brajczuk DA (2010) Mining uncertain data for frequent itemsets that satisfy aggregate constraints. In: Proceedings of the 2010 ACM symposium on applied computing (SAC), pp 1034–1038
Li YC, Yeh JS, Chang CC (2008) Isolated items discarding strategy for discovering high utility itemsets. Data Knowl Eng 64(1):198–217
Li Z, Liu J, Tang X (2009) Constrained clustering via spectral regularization. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 421–428
Lin MY, Hsueh SC, Chang CW (2008) Fast discovery of sequential patterns in large databases using effective time-indexing. Inf Sci 178(22):4228–4245
Lin MY, Lee SY (2005) Efficient mining of sequential patterns with time constraints by delimited pattern growth. Knowl Inf Syst 7(4):499–514
Liu EY, Zhang Z, Wang W (2011) Clustering with relative constraints. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 947–955
Lu Z, Carreira-Perpiñán MÁ (2008) Constrained spectral clustering through affinity propagation. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 1–8
Lucey S, Ashraf AB (2013) Nearest neighbor classifier generalization through spatially constrained filters. Pattern Recognit 46(1):325–331. doi:10.1016/j.patcog.2012.06.009

Lu Z, Leen TK (2007) Penalized probabilistic clustering. Neural Comput 19(6):1528–1567. doi:10.1162/neco.2007.19.6.1528

Mabroukeh NR, Ezeife CI (2010) A taxonomy of sequential pattern mining algorithms. ACM Comput Surv 43(1):1–3

Mansingh G, Osei-Bryson KM, Reichgelt H (2011) Using ontologies to facilitate post-processing of association rules by domain experts. Inf Sci 181(3):419–434

Marinica C, Guillet F (2010a) Knowledge-based interactive postmining of association rules using ontologies. IEEE Trans Knowl Data Eng 22(6):784–797

Marinica C, Guillet F (2010) Knowledge-based interactive postmining of association rules using ontologies. IEEE Trans Knowl Data Eng 22(6):784–797. doi:10.1109/TKDE.2010.29

Marriott K, Nethercote N, Rafeh R, Stuckey PJ, de la Banda MG, Wallace M (2008) The design of the zinc modelling language. Constraints 13(3):229–267. doi:10.1007/s10601-008-9041-4

Masseglia F, Poncelet P, Teisseire M (2009) Efficient mining of sequential patterns with time constraints: reducing the combinations. Expert Syst Appl 36(2):2677–2690

Masson C, Robardet C, Boulicaut J (2004) Optimizing subset queries: a step towards sql-based inductive databases for itemsets. In: Haddad H, Omicini A, Wainwright RL, Liebrock LM (eds) Proceedings of the 2004 ACM symposium on applied computing (SAC), Nicosia, Cyprus, March 14–17, 2004. ACM, pp 535–539. doi:10.1145/967900.968013

Meo R, Psaila G, Ceri S (1998) An extension to SQL for mining association rules. Data Min Knowl Disc 2(2):195–224. doi:10.1023/A:1009774406717

Meo R, Psaila G (2006) An XML-based database for knowledge discovery. In: Proceedings of the 10th international conference on extending database technology (EDBT), pp 814–828

Morzy T, Zakrzewicz M (1997) SQL-like language for database mining. In: Proceedings of the first east-European symposium on advances in databases and information systems (ADBIS), pp 331–317

Nethercote N, Stuckey PJ, Becket R, Brand S, Duck GJ, Tack G (2007) Minizinc: towards a standard CP modelling language. In: Proceedings of the 13th international conference on principles and practice of constraint programming, CP'07. Springer, Berlin, Heidelberg, pp 529–543. http://dl.acm.org/citation.cfm?id=1771668.1771709

Nguyen N, Caruana R (2008) Improving classification with pairwise constraints: a margin-based approach. In: Daelemans W, Goethals B, Morik K (eds) ECML/PKDD (2), Lecture Notes in Computer Science, vol 5212. Springer, pp 113–124. http://dblp.uni-trier.de/db/conf/pkdd/pkdd2008-2.html#NguyenC08

Nijssen S, Fromont E (2007) Mining optimal decision trees from itemset lattices. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 530–539

Nijssen S, Fromont E (2010) Optimal constraint-based decision tree induction from itemset lattices. Data Min Knowl Disc 21(1):9–51. doi:10.1007/s10618-010-0174-x

Niyogi P, Pierrot JB, Siohan O (2000) Multiple classifiers by constrained minimization. In: Proceedings of the acoustics, speech, and signal processing, 2000. On IEEE international conference, vol 06, ICASSP '00. IEEE Computer Society, Washington, DC, pp 3462–3465. doi:10.1109/ICASSP.2000.860146

Okabe M, Yamada S (2012) Clustering by learning constraints priorities. In: Proceedings of the 12th international conference on data mining (ICDM), pp 1050–1055

Park SH, Fürnkranz J (2008) Multi-label classification with label constraints. In: Technical report, knowledge engineering group, TU Darmstadt

Pei J, Han J, Lakshmanan LVS (2004) Pushing convertible constraints in frequent itemset mining. Data Min Knowl Disc 8(3):227–252

Pei J, Han J, Wang W (2007) Constraint-based sequential pattern mining: the pattern growth methods. Inf Sci 28(2):133–160

Pei J, Han J (2000) Can we push more constraints into frequent pattern mining? In: Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 350–354

Pinto H, Han J, Pei J, Wang K, Chen Q, Dayal U (2001) Multi-dimensional sequential pattern mining. In: Proceedings of the 2001 ACM CIKM international conference on information and knowledge management (CIKM), pp 81–88

Plantevit M, Laurent A, Laurent D, Teisseire M, Choong YW (2010) Mining multidimensional and multi-level sequential patterns. Trans Knowl Discov Data 4(1):4

Pyle D (1999) Data preparation for data mining. Morgan Kaufmann Publishers Inc., San francisco

Richter L, Wicker J, Kessler K, Kramer S (2008) An inductive database and query language in the relational model. In: Proceedings of the 11th international conference on extending database technology (EDBT), pp 740–744

Rigollet P, Tong X (2011a) Neyman-pearson classification, convexity and stochastic constraints. J Mach Learn Res 12:2831–2855

Rigollet P, Tong X (2011b) Neyman-pearson classification under a strict constraint. Proc Track J Mach Learn Res 19:595–614

Romei A, Ruggieri S, Turini F (2006) KDDML: a middleware language and system for knowledge discovery in databases. Data Knowl Eng 57(2):179–220. doi:10.1016/j.datak.2005.04.007

Romei A, Turini F (2011) Programming the KDD process using XQuery. In: Proceedings of the international conference on knowledge discovery and information retrieval (KDIR), pp 131–139

Romei A, Turini F (2010) XML data mining. Softw Pract Exp 40(2):101–130. doi:10.1002/spe.944

Romei A, Turini F (2011) Inductive database languages: requirements and examples. Knowl Inf Syst 26(3):351–384

Ruiz C, Spiliopoulou M, Ruiz EM (2010) Density-based semi-supervised clustering. Data Min Knowl Disc 21(3):345–370

Sarawagi S, Thomas S, Agrawal R (2000) Integrating association rule mining with relational database systems: alternatives and implications. Data Min Knowl Disc 4(2/3):89–125

Schultz M, Joachims T (2003) Learning a distance metric from relative comparisons. In: Thrun S, Saul LK, Schölkopf B (eds) Proceeding of advances in neural information processing systems (NIPS), December 8–13, 2003, Vancouver and Whistler, British Columbia. MIT Press, pp 41–48. http://papers.nips.cc/paper/2366-learning-a-distance-metric-from-relative-comparisons

Shankar S (2009) Utility sentient frequent itemset mining and association rule mining: a literature survey and comparative study. Int J Soft Comput Appl 4:81–95

Small K, Wallace BC, Brodley CE, Trikalinos TA (2011) The constrained weight space SVM: learning with ranked features. In: Proceedings of the 28th international conference on machine learning (ICML), pp 865–872

Soulet A, Crémilleux B (2005) Optimizing constraint-based mining by automatically relaxing constraints. In: Proceedings of the 5th IEEE international conference on data mining (ICDM), 27–30 November 2005, Houston. IEEE Computer Society, pp 777–780. doi:10.1109/ICDM.2005.112

Soulet A, Crémilleux B (2009) Mining constraint-based patterns using automatic relaxation. Intell Data Anal 13(1):109–133

Soulet A, Crémilleux B, Plantevit M (2011) Summarizing contrasts by recursive pattern mining. In: Spiliopoulou M, Wang H, Cook DJ, Pei J, Wang W, Zaïane OR, Wu X (eds) Data mining workshops (ICDMW), 2011 IEEE 11th international conference on, Vancouver, December 11, 2011. IEEE Computer Society, pp 1155–1162. doi:10.1109/ICDMW.2011.161

Srikant R, Agrawal R (1995) Mining generalized association rules. In: Proceedings of the 21th conference on very large data bases (VLDB), pp 407–419

Srikant R, Agrawal R (1996) Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the 5th international conference on extending database technology (EDBT), pp 3–17

Srikant R, Vu Q, Agrawal R (1997) Mining association rules with item constraints. In: Proceedings of the third international conference on knowledge discovery and data mining (KDD), pp 67–73

Sriphaew K, Theeramunkong T (2002) A new method for finding generalized frequent itemsets in generalized association rule mining. In: Proceedings of the 7th IEEE symposium on computers and communications (ISCC), pp 1040–1045

Strehl A, Ghosh J (2003) Relationship-based clustering and visualization for high-dimensional data mining. INFORMS J Comput 15(2):208–230

Tan PN, Steinbach M, Kumar V (2006) Introduction to data mining. Addison Wesley, Boston

Tao F, Murtagh F (2003) Weighted association rule mining using weighted support and significance framework. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 661–666

Trasarti R, Bonchi F, Goethals B (2008) Sequence mining automata: a new technique for mining frequent sequences under regular expressions. In: Proceedings of the 8th IEEE international conference on data mining (ICDM), pp 1061–1066

Tseng VS, Shie BE, Wu CW, Yu PS (2013) Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans Knowl Data Eng 25(8):1772–1786. doi:10.1109/TKDE.2012.59

Tsochantaridis I, Joachims T, Hofmann T, Altun Y (2005) Large margin methods for structured and interdependent output variables. J Mach Learn Res 6:1453–1484

Vanderlooy S, Sprinkhuizen-Kuyper IG, Smirnov EN, van den Herik HJ (2009) The roc isometrics approach to construct reliable classifiers. Intell Data Anal 13(1):3–37. http://dl.acm.org/citation.cfm?id=1551758.1551760

Vens C, Struyf J, Schietgat L, Dzeroski S, Blockeel H (2008) Decision trees for hierarchical multi-label classification. Mach Learn 73(2):185–214

von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416. doi:10.1007/s11222-007-9033-z

Vu V, Labroche N, Bouchon-Meunier B (2010) An efficient active constraint selection algorithm for clustering. In: 20th international conference on pattern recognition, ICPR 2010, Istanbul, Turkey, 23–26 August 2010. IEEE Computer Society, pp 2969–2972. doi:10.1109/ICPR.2010.727

Wagstaff K, Basu S, Davidson I (2006) When is constrained clustering beneficial, and why? In: Proceedings, the twenty-first national conference on artificial intelligence and the eighteenth innovative applications of artificial intelligence conference (AAAI)

Wagstaff K, Cardie C (2000) Clustering with instance-level constraints. In: Proceedings of the seventeenth national conference on artificial intelligence and twelfth conference on Innovative applications of artificial intelligence (AAAI/IAAI), pp 1103–1110

Wagstaff K, Cardie C, Rogers S, Schrödl S (2001) Constrained k-means clustering with background knowledge. In: Proceedings of the eighteenth international conference on machine learning, ICML '01. Morgan Kaufmann Publishers Inc., San Francisco, pp 577–584. http://dl.acm.org/citation.cfm?id=645530.655669

Wang K, Jiang Y, Yu JX, Dong G, Han J (2005) Divide-and-approximate: a novel constraint push strategy for iceberg cube mining. IEEE Trans Knowl Data Eng 17(3):354–368

Wang X, Rostoker C, Hamilton HJ (2012) A density-based spatial clustering for physical constraints. J Intell Inf Syst 38(1):269–297

Wang X, Qian B, Davidson I (2014) On constrained spectral clustering and its applications. Data Min Knowl Disc 28(1):1–30. doi:10.1007/s10618-012-0291-9

Wang X, Davidson I (2010) Flexible constrained spectral clustering. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 563–572

Wang F, Ding CHQ, Li T (2009) Integrated kl (k-means—laplacian) clustering: a new clustering approach by combining attribute data and pairwise relations. In: Proceedings of the SIAM international conference on data mining (SDM), pp 38–48

Wang W, Yang J, Yu PS (2000) Efficient mining of weighted association rules (WAR). In: Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining (KDD), pp 270–274

Wei JT, Lin SY, Wu HH (2010) A review of the application of rfm model. Afr J Bus Manag 4(19):4199–4206

Witten IH, Frank E, Hall M (2011) Data mining, pratical machine learning tools and techiniques, 3rd edn. Morgan Kaufmann, San Francisco

Wu CM, Huang YF (2011) Generalized association rule mining using an efficient data structure. Expert Syst Appl 38(6):7277–7290

Xing EP, Ng AY, Jordan MI, Russell SJ (2002) Distance metric learning with application to clustering with side-information. In: Advances in neural information processing systems (NIPS), pp 505–512

Xing EP, Ng AY, Jordan MI, Russell S (2002) Distance metric learning, with application to clustering with side-information. Advances in neural information processing systems 15. MIT Press, Cambridge

Yan R, Zhang J, Yang J, Hauptmann AG (2006) A discriminative learning framework with pairwise constraints for video object classification. IEEE Trans Pattern Anal Mach Intell 28(4):578–593. doi:10.1109/TPAMI.2006.65

Yan W, Goebel KF (2004) Designing classifier ensembles with constrained performance requirements. In: Proceedings of the SPIE defense security symposium, multisensor multisource information fusion: architectures, algorithms, and applications (2004), pp 78-87

Yao H, Hamilton HJ, Butz CJ (2004) A foundational approach to mining itemset utilities from databases. In: Proceedings of the fourth SIAM international conference on data mining (SDM), pp 482–486

Yun U (2008) A new framework for detecting weighted sequential patterns in large sequence databases. Knowl Based Syst 21(2):110–122

Yun U, Shin H, Ryu KH, Yoon E (2012) An efficient mining algorithm for maximal weighted frequent patterns in transactional databases. Knowl Based Syst 33:53–64

Yun U, Leggett JJ (2005) WFIM: weighted frequent itemset mining with a weight range and a minimum weight. In: Kargupta et al., pp 636–640. doi:10.1137/1.9781611972757.76

Yun U, Ryu KH (2010) Discovering important sequential patterns with length-decreasing weighted support constraints. Int J Inf Technol Decis Mak 9(4):575–599

Yun U, Ryu KH (2011) Approximate weighted frequent pattern mining with/without noisy environments. Knowl Based Syst 24(1):73–82

Zaidan O, Eisner J (2008) Modeling annotators: a generative approach to learning from annotator rationales. In: Proceedings of the conference on empirical methods in natural language processing (EMNLP), pp 31–40

Zaki MJ (2000) Sequence mining in categorical domains: incorporating constraints. In: Proceedings of the 9th international conference on information and knowledge management (CIKM), pp 422–429

Zhang J, Yan R (2007) On the value of pairwise constraints in classification and consistency. In: Proceedings of the 24th international conference on machine learning, ICML '07. ACM, New York, pp 1111–1118. doi:10.1145/1273496.1273636

Zhang C, Zhang S (2002) Association rule mining, models and algorithms, lecture notes in computer science. Springer, New York

Zhang Y, Zhang L, Nie G, Shi Y (2009) A survey of interestingness measures for association rules. In: Proceedings of the second international conference on business intelligence and financial engineering, (BIFE), pp 460–463

Zhong S, Ghosh J (2003) Scalable, balanced model-based clustering. In: Proceedings of the third SIAM international conference on data mining (SDM), San Francisco, pp 71–82