Chapter 17
Supervised Pattern Mining and Applications to Classification

Albrecht Zimmermann and Siegfried Nijssen

Abstract In this chapter we describe the use of patterns in the analysis of supervised data. We survey the different settings for finding patterns as well as sets of patterns. The pattern mining settings are categorized according to whether they include class labels as attributes in the data or whether they partition the data based on these labels. The pattern set mining settings are categorized along several dimensions, including whether they perform iterative mining or post-processing, operate globally or locally, and whether they use patterns directly or indirectly for prediction.

Keywords Rules · Classification · Subgroup discovery· Prediction · Pattern sets

1 Introduction

Although early constrained pattern mining in the form of frequent itemset mining (*FIM*) focused on an *unsupervised* setting, a natural extension is to apply these techniques in a supervised context as well. In the supervised context, one attribute (or sometimes a small set of attributes) is considered to be special, and we are only interested in finding relationships between this attribute and the other attributes. Whereas this limits the patterns that will be found, it makes the analysis more targeted and in many cases more useful. Consider for instance the context of customer defection (churn), where one wishes to find relationships between the loyalty of customers and other characteristics of the customers; or consider applications in cheminformatics, where one wishes to find relationships between molecular structures and their activity: in all these cases, a targeted analysis with respect to the indicated target attribute is likely to produce the most valuable results.

A. Zimmermann (✉)
INSA Lyon, LIRIS CNRS UMR 5205, Bâtiment Blaise Pascal,
69621 Villeurbanne CEDEX, France
e-mail: albrecht.zimmermann@insa-lyon.fr

S. Nijssen
KU Leuven, Celestijnenlaan 200 A, 3001 Leuven, Belgium

Universiteit Leiden, Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands
e-mail: siegfried.nijssen@cs.kuleuven.be

C. C. Aggarwal, J. Han (eds.), *Frequent Pattern Mining,*
DOI 10.1007/978-3-319-07821-2_17, © Springer International Publishing Switzerland 2014

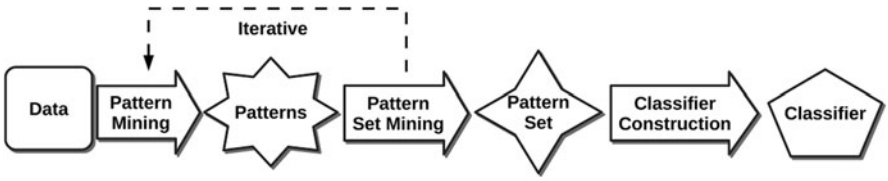

Fig. 17.1 The process of classifier construction via supervised pattern mining

In this chapter, we will provide an overview of pattern mining techniques that can be used in such a supervised context. The patterns found by these techniques can often be interpreted as *rules*: the conditions of the rule identify examples for which a certain property in the target attribute holds. The techniques are hence related to Machine Learning: many traditional Machine Learning algorithms are rule-based as well. A natural question is how to link these two fields to each other, in particular given that the focus of both areas is complementary: most traditional machine learning techniques deal with the large search space of potential rules by adopting heuristics; *pattern mining* methods, on the other hand, offer more efficient methods for traversing a search space exhaustively, promising to find better rules than those found by traditional rule learners. We will address this as well.

The earliest techniques that integrated both areas mirrored the FIM techniques closely, using support and confidence to constrain itemsets and rules, and support's anti-monotonicity to prune the search space. In addition to new challenges, supervised pattern mining also offers new opportunities, however, since the supervision allows to use additional quality measures and prune based on the properties of constraints based on these measures. By now, the field has developed far from its origins, encompassing other representations, incorporating approaches and quality measures developed in the context of Machine Learning, and paying much attention to pattern set mining.

The latter topic is not limited to supervised pattern mining but is of particular importance there: when constructing classifiers, rule lists or sets, but also decision trees, or non-symbolic classifiers, redundancy among or irrelevance of patterns is often detrimental to the classifier's performance.

We have given a unifying perspective on pattern-based classification in the past [9] in which we focused on two dimensions. The first concerned pattern set mining, specifically whether techniques performed *post-processing*, selecting some patterns out of the result set of a single pattern mining step, or whether they iterated pattern mining. The second dimension focused on whether they let the pattern mining and selection process be *guided by a particular model* or not. While these distinctions still stand, in our opinion, we have decided to structure this chapter differently, discussing each of the three steps shown in Fig. 17.1 separately: pattern mining, pattern set mining, and finally classifier construction, and surveying the different, sometimes numerous, options available.

2 Supervised Pattern Mining

The majority of texts in this book deal with different unsupervised pattern mining settings. We will quickly repeat the relevant definitions here to clarify which setting we discuss:

Definition 2.1 *Given a data language $\mathcal{L}_\mathcal{D}$ which describes the syntax of potential transactions in the data, a transactional data set $\mathcal{D} \subseteq \mathcal{L}_\mathcal{D}$ is of the form $\mathcal{D} = \{d_1, \ldots, d_n\}, d_i \in \mathcal{L}_\mathcal{D}$. Given a pattern language $\mathcal{L}_\pi$, we define a function match : $\mathcal{L}_\pi \times \mathcal{L}_\mathcal{D} \mapsto \{0, 1\}$, which decides whether a pattern occurs in a transaction or not. The set of transactions from a data set $\mathcal{D}$ matched by a pattern π are referred to as its cover: $cov_\mathcal{D}(\pi) = \{d \in \mathcal{D} \mid match(\pi, d) = 1\}$, and the size of the cover is referred to as π's (absolute) support: $supp_\mathcal{D}(\pi) = |cov_\mathcal{D}(\pi)|$.*

The easiest instantiation of this definition is the case of itemset databases: given a set of items $\mathcal{I}$, $\mathcal{L}_\pi = \mathcal{L}_\mathcal{D} = 2^\mathcal{I}$, and $match(\pi, d) = 1 \Leftrightarrow \pi \subseteq d$. For other types of data, such as for instance graph data or sequential data, alternative definitions for $\mathcal{L}_\mathcal{D}$, $\mathcal{L}_\pi$ and *match* can be used, and most ideas presented in the rest of this paper can be applied immediately for these alternative definitions.

The biggest difference between unsupervised and supervised pattern mining is the presence of a variable of interest. This variable is often the class variable that can take on one out of several nominal *class labels*.

Definition 2.2 *Given a data language $\mathcal{L}_\mathcal{D}$, and a set of class labels $\mathcal{C} = \{C_1, \ldots, C_k\}$, a labeled data set $\mathcal{D}_C$ is of the form $\mathcal{D}_C = \{(d_1, c_1), \ldots, (d_n, c_n)\}$, $d_i \in \mathcal{L}_\mathcal{D}, c_i \in \mathcal{C}$.*

The most common setting is that of *classification*, in which the task is to learn a mechanism to predict the class label for unseen data based on rules or patterns. Alternatively, the target for prediction can also be numerical, requiring a *regression* model.

However, another popular setting is that of *subgroup discovery*, which can be generalized to *exceptional model mining* when the target attribute is not a single categorical attribute [24].

Instead of prediction, the goal in this setting is the *characterization* of subsets of the data, i.e. subgroups. The mined rules are therefore not means to the end of prediction but the end themselves, and users are expected to inspect them to gain a deeper understanding of the data. In other words, classification is concerned with *outcomes* on future data, subgroup discovery with *descriptions* of current data.

As a result of this, the quality criteria and heuristics used are sometimes different. However, many of the techniques used are also shared, and for reasons of clarity of presentation, we will mainly focus on classification in this chapter, and make differences to the other settings explicit when appropriate.

2.1 Explicit Class Labels

A first, straight-forward interpretation considers class labels just additional items in the transactional data, i.e. $\mathcal{I}' = \mathcal{I} \cup \mathcal{C}$, and imposes a *syntactical* constraint on itemsets being mined from it: each itemset has to include exactly one of those class-label items. This has the tremendous advantage that existing techniques for FIM can be used directly, e.g. Apriori [2], Eclat [40], or FPGrowh [20].

The typical FIM mining approach identifies interesting itemsets by using a *minimum support* threshold that itemsets' support has to exceed, and chooses relevant rules by using a *minimum confidence* threshold. Since specializations of patterns, e.g. extensions of an itemset with additional items, will have less than or equal support as the pattern itself, the search space can be pruned, allowing for exhaustive enumeration.

This can be adapted by using class labels explicitly as items. It allows to treat settings with more than two classes in a straightforward way:

- For all class labels C:
  1. Mine all itemsets including C that exceed the minimum support threshold
  2. Retain all association rules $r \rightarrow C$ that exceed the minimum confidence threshold

The resulting association rules are referred to as *class association rules* (*cars*) and are restricted to having only the class item as their *right-hand side*. Their quality is usually evaluated using *confidence* as in the case of general association rules:

Definition 2.3 *Given a set of items $\mathcal{I}$ and a set of class labels $\mathcal{C}$, a class association rule is of the form $r \rightarrow c, r \subseteq \mathcal{I}, c \in \mathcal{C}$. r is called its left-hand side (LHS), antecedent, or rule body, c its right-hand side, consequent, or rule head. Its confidence is defined as $conf(r \rightarrow c) = \frac{supp_{\mathcal{D}}(r \cup c)}{supp_{\mathcal{D}}(r)}$.*

Prominent examples of classification learners that build upon class association rules are the CBA [27] and CMAR [25] algorithms. The Harmony algorithm, introduced by [38], also takes this view of class labels, as does the ART technique [17]. As a direct application of FIM techniques, these methods are somewhat limited by typically using only a single minimum support and confidence threshold, which might be inappropriate in the case of skewed class distributions. They can, however, benefit from all developments in FIM research, such as better rule quality measures (replacing confidence), and the development of more efficient algorithms.

2.2 Classes as Data Subsets

A second interpretation of different *classes* in the data is to consider each class a separate data set and a whole database the union of those subsets:

Definition 2.4 *Given a labeled data set $\mathcal{D}_C$, and a set of class labels $\mathcal{C}$, the subsets $\forall C_i \in \mathcal{C} : \mathcal{D}^i = \{(d, C_i) \in \mathcal{D}_C\}$ are called classes.*

Each of these classes can be treated like a distinct data set—the ARC-BC algorithm by [3], for instance, mines *car*s from each class separately, using a single relative support threshold that is used as a constraint on each class in turn. Using this interpretation also opens up several new possibilities.

The first, and potentially most important one, is that this opens up the supervised pattern mining setting to all possible pattern languages: whether itemsets, sequences, trees, or graph-structured data and patterns, the techniques that we describe in this section are applicable to all of them.

Second, there are new ways of using significance and quality measures.

Multiple Support Thresholds There is the possibility of using support thresholds. The XRules classifier [41], for instance, uses a separate minimum support threshold for each class. It is also a first example of supervised pattern mining in a different pattern domain than itemsets, producing predictive rules the rule body of which consists of tree fragments, called *structural rules* in the work.

Instead of minimum support constraints, it is also natural to use maximum support constraints: a rule which is specific for one class should after all not cover many examples in other classes than the class it is predicting. The technique introduced by [22], for instance, exploits this observation by finding patterns that are frequent within one class, but infrequent in the other. It exploits a relationship with version space theory from machine learning.

The CCCS classifier [4] even relies only on a maximum support constraint and removes the minimum support constraint entirely. It is argued that infrequent patterns in a class can be found by enumerating small subsets of transactions in this class.

The problem that remains in each of these cases is a similar one as for single support thresholds: how to set the parameters. A pattern that occurs in 50 % of one class, and 15 % of the other, could be considered a valuable predictive pattern, as might be a pattern that occurs in 80 % of the first and 30 % of the second. Support constraints that accommodate both patterns, however, e.g. $supp_{\min} = 0.5$, $supp_{\max} = 0.3$ would allow results of questionable usefulness.

To address this, the Fitcare classifier proposed by [10] takes this idea further and uses a much larger parameter set: given k classes, each class is mined separately, parametrized by a minimum support constraint and $k - 1$ maximum support constraints on all other classes. To make this manageable, the support constraints are dynamically adjusted during mining.

Statistical Measures A popular alternative approach is the use of constraints on measures specifically designed for supervised data. These measures typically serve as a replacement for confidence in selecting relevant predictive patterns; the underlying patterns are still found using a minimum support threshold on the complete data.

As a straightforward example, consider the *accuracy* measure:

Definition 2.5 *Given two classes $\mathcal{D}^+$, $\mathcal{D}^-$, pattern r. The accuracy of r is defined as* $acc(r) = \frac{supp_{\mathcal{D}^+}(r)+(|\mathcal{D}^-|-supp_{\mathcal{D}^-}(r))}{|\mathcal{D}|}$.

In general, most measures for evaluating the predictive power of a rule can be expressed as functions from the values in the *contingency table*:

	C_+	$\neg C_+$	
r	$p = supp_{\mathcal{D}^+}(r)$	n	$supp_{\mathcal{D}}(r)$
$\neg r$	$\lvert\mathcal{D}^+\rvert - supp_{\mathcal{D}^+}(r)$	$N - n$	$supp_{\mathcal{D}}(\neg r) = \lvert\mathcal{D}\rvert - supp_{\mathcal{D}}(r)$
	$P = \lvert\mathcal{D}^+\rvert$	$N = \lvert\mathcal{D}\rvert - \lvert\mathcal{D}^+\rvert$	$\lvert\mathcal{D}\rvert$

An arrangement in a contingency table invites the use of well-established measures such as *Information Gain* or χ^2 to mine *correlating* [29], *contrast* [6], or *discriminating* patterns [11]. Similarly, the *growth rate* can be used to mine *emerging patterns* [14, 26, 37]. It divides the support in one class by the support in the other one.

A measure that is often used in *subgroup discovery* is *Weighted Relative Accuracy* [23]:

Definition 2.6 *Given a rule $r \rightarrow C_+$, its Weighted Relative Accuracy is defined as* $WRAcc(r \rightarrow C_+) = \frac{supp_{\mathcal{D}}(r)}{\lvert\mathcal{D}\rvert}\left(\frac{supp_{\mathcal{D}^+}(r)}{supp_{\mathcal{D}}(r)} - \frac{\lvert\mathcal{D}^+\rvert}{\lvert\mathcal{D}\rvert}\right)$.

It is instructive to compare *accuracy* and *WRAcc* to gain a better understanding of the conceptual differences between classification and subgroup discovery.

Since the final goal is to find rules with good predictive accuracy, accuracy treats covering one negative instance less as equal to covering one positive instance more. Consider a data set consisting of 60 instances in $\mathcal{D}^+$, and 40 in $\mathcal{D}^-$, and a rule covering 40 positive and 15 negative instances. Its accuracy is 0.65, and rules that covered 5 positive instances more, or 5 negative instances less, would both achieve a (better) accuracy of 0.7. In the case of WRAcc, the situation is different: the original rule would have a score of 0.07 and while covering 5 negative instances less improves it to 1.0, covering 5 positive instances more yields a smaller improvement (to 0.09).

Since subgroup discovery aims to characterize *differences*, this behavior makes perfect sense: the positive class is overrepresented in the entire data and coverage of this class has to increase more strongly to be interesting. Given a heavily skewed data set (e.g. $\lvert\mathcal{D}^+\rvert = 0.9\lvert\mathcal{D}\rvert$), a rule predicting all transactions to belong to the majority class might be acceptable for a classifier but would be unattractive for subgroup discovery.

WRAcc also includes a normalizing factor that weights a rule's score by its *effect size* but this is in fact *not* particular to subgroup discovery. When it comes to normalization, the difference between classification and subgroup discovery measures lies in the motivation: classification wants assurance that mined rules will work on unseen data, subgroup discovery wants rules to be representative of the data they have been mined from.

In combination with a minimum support constraint, *WRAcc* can be used in a class association rule miner instead of confidence [21]. This idea can be generalized to other subgroup discovery measures (and the measures listed above), replacing the confidence measure in class association rule miners by numerous other functions as proposed by [5]. CMAR, for instance, filters *car*s using a χ^2 minimum threshold in addition to the minimum confidence threshold.

That the differences between different types of supervised patterns mainly come down to a change in quality function has been shown in detail by [32], the authors of which coined the term "supervised descriptive rule discovery" for such approaches

and has been leveraged by [44] to use one type of mining technique to address different tasks: classification, subgroup discovery, and conceptual clustering.

Eliminating Minimum Support The above settings essentially apply statistical measures in addition to minimum support. The minimum support parameter remains a parameter that needs to be set. Several approaches have successfully eliminated this parameter.

The main observation is that thresholds on quality measures can be translated into support thresholds; hence, if a support threshold is not given, it is possible to automatically determine an additional support threshold for use in a pattern mining algorithm.

Returning to the accuracy measure, we can set a minimum threshold on it: $acc(r) \geq \theta_{acc}$. This can be transformed into $p + (N - n) \geq \theta_{acc} \cdot |\mathcal{D}|$, and further into $p \geq \theta_{acc} \cdot |\mathcal{D}| - N + n \geq \theta_{acc} \cdot |\mathcal{D}| - N$. So we derive support constraints based on the threshold on the quality measure itself [31].

For measures that are convex, which includes the ones mentioned above but also many others, a similar argument is possible: convex functions take their maxima at extreme points, i.e. points with $p = 0$ or $n = 0$. Thus, based on a threshold on the minimal acceptable values for a statistical scoring function, thresholds on a pattern's p and n can be derived and enforced during mining. This makes it effective to use the quality measure to prune *during* rule mining [6, 7, 11, 12, 15, 19, 29, 31, 35, 39, 43–45].

Thus far we have discussed approaches that use thresholds and exhaustively mine all patterns that satisfy the thresholds. An even easier and often more effective approach is to perform top-k mining instead. In top-k mining, one is interested in finding only those patterns which have the k highest scores; the only parameter that needs to be specified is k. This has been leveraged by [7, 11, 12, 15, 35, 42, 45]. The nature of this mining process means that the threshold(s) increase during mining, pruning more and more candidate patterns as the search progresses. To achieve a quick increase of the threshold, it can be useful to perform a *best-first* search during which it is always the rule with the highest upper bound that is specialized.

2.3 Numerical Target Values

As opposed to the setting discussed in the preceding sections, in which each transaction is labeled with one out of a set of *discrete* labels, a numerical variable of interest can have potentially infinitely many values. As a result of this, each transaction in the data may have a different target value, and learning a predictor for particular values, or partitioning the data into subsets consisting of transactions with the same target value, are strategies that are unlikely to be successful. Nevertheless, there exist a number of techniques for *discretizing* numerical values, in which case the problem can be reduced to the classification setting.

Alternatively, one can either attempt to mine patterns that partition the data into transactions that have approximately the same numerical value, or those that can be used as elements of a regression function that outputs a numerical result based on their appearance in a transaction. An interestingness measure that can be used in the former case is *interclass variance*:

Definition 2.7 *Given a data set with numerical labels of the form* $\mathcal{D}_Y = \{(d_1, y_1), \ldots, (d_n, y_n)\}$, $y_i \in \mathrm{R}$, *pattern* π, *the average* y *in a subset* $\mathcal{D}_\subseteq$ *of that data set is:*

$$avg(\mathcal{D}_\subseteq) = \frac{\sum_{(d_i,y_i)\in\mathcal{D}_\subseteq} y_i}{|\mathcal{D}_\subseteq|}$$

The interclass variance of π *is defined as:*

$$var(\pi) = |cov(\pi)|\,(avg(cov(\pi)) - avg(\mathcal{D}_Y))^2 \\ +|\mathcal{D}_Y \setminus cov(\pi)|\,(avg(\mathcal{D}_Y \setminus cov(\pi)) - avg(\mathcal{D}_Y))^2$$

Interclass variance is convex, which means that thresholds on its value can be translated into thresholds on support values, and thresholded or top-k mining used in the same manner as for discrete target values.

In the latter case, works such as [13, 33, 34] have chosen linear regression functions that weight the contributions of individual patterns. Based on these weights, the authors define a quality function for individual patterns, and derive upper bounds that they use to perform top-k mining for component patterns of the regression model.

3 Supervised Pattern Set Mining

The result of a supervised pattern mining operation, as so often in pattern mining settings, is typically a very large set of redundant and contradictory patterns. Even when mining only the top-k patterns, many of those will cover (almost) the same instances. As we mentioned in the introduction, when constructing classifiers, redundant patterns or patterns that are irrelevant in the presence of others can be undesirable. If the classifier takes the form of an unordered rule set, for instance, which we will describe in Sect. 4, certain rules could strongly boost each other, far in excess of their actual relevance and usefulness.

Hence many techniques in the literature include a mechanism for mining or selecting a subset of the result set. Where the techniques for supervised pattern mining intended to improve on Machine Learning techniques, replacing heuristics with exhaustive search, the methods for supervised *pattern set mining* are strongly inspired by Machine Learning techniques. In particular, both sequential (covering/re-weighting) or *separate-and-conquer*, and decision tree like *divide-and-conquer* techniques can be found time and again in works on supervised pattern mining.

There are two wide-spread approaches to pattern set mining. One is *post-processing*:

1. Mine a set of supervised patterns satisfying certain constraints
2. Select some patterns out of this set following certain criteria

and *iterative pattern are derived set mining*:

1. Mine a (set of) supervised pattern(s) satisfying certain constraints
2. Modify the constraints or data
3. Return to 1.

The main argument in favor of the post-processing approach is its efficiency. It allows to run a pattern mining algorithm only once and hence avoids the possibly time consuming repeated execution of pattern mining algorithms. The main arguments in favor of iterative mining algorithms are their potentially higher accuracy and their potential to use parameter-free pattern mining algorithms; in many of these algorithms, it is not necessary to define a minimum support threshold in advance.

Both separate-and-conquer and divide-and-conquer techniques have been used within either of these categories.

Most of these techniques can be understood in terms of the partition that a set of patterns induces on the data. We therefore first need to introduce the concept of *equivalence relations* and *partitions*:

Definition 3.1 *An equivalence relation on $\mathcal{D}$ is a binary relation $\sim$ such that for all $d_1, d_2, d_3 \in \mathcal{D}$, the relation is:*

1. *Reflexive:* $d_1 \sim d_1$.
2. *Symmetric:* $d_1 \sim d_2 \Rightarrow d_2 \sim d_1$.
3. *Transitive:* $d_1 \sim d_2 \wedge d_2 \sim d_3 \Rightarrow d_1 \sim d_3$.

The equivalence relation partitions $\mathcal{D}$ into disjunct subsets called equivalence classes or blocks. The equivalence class of an element $d \in \mathcal{D}$ is given as $[d] = \{d' \in \mathcal{D} \mid d \sim d'\}$. The set of blocks is called partition or quotient set, and is denoted by $\mathcal{D}/\sim$.

Intuitively, transactions are in an equivalence class if they can not be distinguished from each other. We can use patterns to create a new database, in which each transaction is described by a list of patterns present in it. We consider two transactions equivalent in this new representation if they are described using the same lists of patterns.

More formally, an individual pattern r induces an equivalence relation $\forall d_1, d_2 \in \mathcal{D}, d_1 \sim_r d_2 \Leftrightarrow match(r, d_1) = match(r, d_2)$, and so does a set of patterns $\mathbb{P}$: $\forall d_1, d_2 \in \mathcal{D}, d_1 \sim_{\mathbb{P}} d_2 \Leftrightarrow (\forall r \in \mathbb{P} : match(r, d_1) = match(r, d_2))$.

In fact, the partitioning of a data set into classes that we defined in Definition 2.3 is induced by an equivalence relation based on the class labels.

In a supervised setting, it are derived is important to distinguish blocks which are *pure* and which are not pure. A block is pure if all examples in it have the same class label. Within a supervised setting it is important that the partition induced by a set of patterns contains mostly pure blocks: if two examples with different class labels contain exactly the same set of patterns, it will be impossible for a deterministic algorithm to predict both correctly.

Most pattern set mining techniques can be summarized in the following manner:

1. Mine or evaluate a (set of) pattern(s), possibly only on parts of the data
2. Based on result of 1, modify the partition, for instance by removing one block or several blocks, or by partitioning them further
3. Return to 1, unless a stopping criterion is met

The differences lie mainly in the blocks on which patterns are evaluated, and in the choice of blocks that are modified.

3.1 Local Evaluation, Local Modification

The first, and largest, class of techniques evaluates or mines patterns *locally*, i.e. only on some of the blocks of a partition, and then also modifies only some of those blocks, typically only those blocks from which the patterns have been mined. This includes in particular those techniques that draw more or less directly on machine learning forebears.

Separate-and-Conquer *Sequential* "local-local" techniques owe much to the sequential covering paradigm of early rule learners. They start from the full database and iteratively remove examples from the dataset, as follows:

1. Find the best rule on the currently remaining data
2. Remove all data covered by that rule
3. Return to 1.

This approach falls squarely into the "local-local" category. Each pattern splits the data that it has been *mined on* into two blocks (the local modification) and its successor pattern is only mined on *one* of these, the uncovered one (the local evaluation). Several early algorithms have used this approach for post-processing, for instance CBA, ARC-BC, and CMAR, whose authors refer to it as *database coverage*.

Separate-and-conquer can be applied both in the post-processing setting and in the iterative mining setting.

Post-processing can be done in two ways: (1) considering the *complete* set of previously mined patterns in each iteration of the sequential covering algorithm, or (2) fixing the order in which patterns are considered and only search for the best rule among those rules that have not been considered in the order yet. The latter means that (a) each pattern is only considered once—if it is rejected, it will never be evaluated again, and (b) the decision which patterns are "best" given certain data is effectively made before pattern set mining. In return, however, the complexity of the learning algorithm is lower.

The algorithms mentioned above (CBA, ARC-BC, CMAR) proceed by fixed order. CMAR differs from the other algorithms in removing data instances only after they have been covered by *several* rules, guided by a user-supplied parameter. CorClass also uses sequential covering with a fixed order as post-processing. Another variation was proposed by [1] in the context of string classification; here, the rules are processed

in order of confidence, but only those instances are removed which are classified correctly by the rule under consideration.

The ART algorithm [17] learns several best *car*s, splits their respective coverage off, and re-iterates on the uncovered data. DDPMine by [12] is perhaps the algorithm that stays truest to the original sequential covering idea: it mines a highest-scoring pattern, removes all covered instances from the data, and recurs.

Divide-and-Conquer Techniques The second type of "local-local" techniques takes its cues from *decision tree induction*:

1. Find the best splitting criterion on a subset of the data
2. Split the data into two blocks corresponding to covered and uncovered instances
3. Recur on the new blocks

A potential advantage of this type of technique is that all mistakes by one pattern can be corrected by other patterns, since all data are reused in later instances to derive additional patterns. In addition, patterns that might not appear interesting on the whole data might become relevant as soon as parts of the data are removed.

This technique is most commonly used in an iterative mining setting, in which the best pattern is searched for using a branch-and-bound top-1 pattern mining algorithm. Examples are Tree^2, proposed by [7], and M^{b}T [15].

A post-processing approach can also be used. For instance, [18] developed a setting in which δ-free patterns are first mined, and then combined for use as tests in a decision tree.

3.2 Global Evaluation, Global Modification

Alternatively, patterns can be mined or evaluated on the *entire* data set, and *all* blocks in the partition are modified. While this means that mining (or selecting) patterns is done using the maximal amount of information, this usually has to be paid for by increased computational complexity, as in each iteration the complete data needs to be traversed. Additionally, the semantics of patterns' relationships are less easy to understand than in the case of "local-local" approaches.

Such techniques necessarily proceed sequentially, either post-processing or mining patterns one after another. The Picker* algorithm by [8] performs post-processing in this manner, picking the pattern that creates the most balanced partition, and splitting all blocks accordingly. It proceeds according to the first option for post-processing described above, considering all promising patterns. The fCork [35] technique uses a measure based on *correspondences*:

Definition 3.2 *Given an equivalence relation $\sim_{\mathbb{P}}$ on a labeled data set $\mathcal{D}_C = \mathcal{D}^+ \cup \mathcal{D}^-$, the number of correspondences in this partition is calculated as* $occ(\mathbb{P}) = \sum_{[d] \in \mathcal{D}_C/\sim_{\mathbb{P}}} |[d] \cap \mathcal{D}^+| \cdot |[d] \cap \mathcal{D}^-|$.

and uses this measure both to post-process mined patterns, and to iteratively mine patterns that reduce correspondences the most. This criterion, as well as that used

by the Picker* algorithm, is *sub-modular*, allowing to give a bound on the quality of greedy approximation to the optimal solution.

There are other "global-global" techniques that differ in that they do not manipulate the data explicitly: the technique introduced by [11] post-processes patterns by rewarding them for class correlation on the full data and penalizing overlap on data already covered by selected patterns. The Krimp technique, described by [36], also falls into this category since it evaluates for each pattern how much it adds to the overall, i.e. global, compression of the data, post-processing a fixed order on patterns.

Instead of removing examples, a reasonable alternative is to attach a weight to examples and modify the weights based on the current composition of a rule set, as in the following generic approach:

1. Find the best rule on the current weighted data
2. Modify the weights of the examples in the data
3. Return to 1.

A reason to give a lower weight to an example may for instance be that we already have many rules that predict this example correctly, and we would like to focus on finding rules for examples that are predicted incorrectly.

This setting performs global evaluation as each new pattern is evaluated on the complete dataset and in principle the weights of all examples can be modified.

Examples of approaches within this setting were proposed by [13, 33, 34, 44]; they can be used either in iterative mining or in post-processing. In the first work, transaction weights are adjusted directly in a subgroup discovery setting. Since subgroup discovery is more concerned with mining good descriptions of statistically different subgroups than with accurate prediction, the removal of covered instances is undesirable. The other works, comprising the GPLS and GBOOST algorithms, and a Bayesian linear regression technique, derive the transaction weights indirectly from weights for patterns involved in a linear classification or regression function and mine patterns iteratively. Since pin point prediction of a numerical value is difficult, reweighting instances based on the current performance of the function is superior to removing instances.

The upshot of these techniques is that the increased computational complexity pays off in a pattern set of smaller cardinality than for "local-local" approaches, typically of comparable or even better quality.

3.3 Local Evaluation, Global Modification

Given the faster running times yet larger pattern sets of "local-local" approaches, and the more expensive operation yet smaller, high-quality sets of "global-global" techniques, the development of "local-global" algorithms should be obvious:

1. Find a best pattern on a subset of the data

2. Based on *all* patterns, manipulate the *entire* data
3. Recur on the new blocks

Notwithstanding this statement, the ReMine algorithm proposed by [45] so far is the only one to proceed in this way to iteratively mine supervised patterns.

3.4 Data Instance-Based Selection

In addition to the partition-based techniques, there is another paradigm, which selects patterns based on individual instances. The Harmony algorithm retains for each training instance the highest-confidence rule, as does CCCS, whereas the technique described by [28], called Large Bayes (LB), selects patterns based on the instances whose labels are to be predicted. This is similar to DeEP, described by [26], and LAC, proposed by [37], which only *generate* patterns that match the instances to be predicted by projecting the data on the items contained in the unlabeled instance.

4 Classifier Construction

After supervised patterns have been mined, and suitable subsets have been selected, the remaining question is how to employ them for predictive purposes. The solutions that have been found fall into two main categories: (1) direct use of patterns as rules to predict the label of an unseen class—the techniques following this paradigm borrow heavily from rule learning approaches in machine learning, or (2) indirect use of patterns in a model; here patterns are typically treated as *features* that are used in well-established machine learning methods.

4.1 Direct Classification

There are two main methods in rule learning when it comes to making predictions. In decision lists, rules are ordered according to some criterion and the first rule that matches the unseen instance makes the prediction. For such classifiers to work requires rules with high accuracy that at the same time do not *overfit* the training data. This means that certain approaches to optimizing quality measures will work better than others: given that maximizing information gain or χ^2 trades off correlation with effect size, maximizing confidence or WRAcc will be more suitable for such classifiers. CBA follows this first approach, ordering the rule list by confidence (descending), support (descending) and length (ascending), as does LAC, ordering by information gain (descending).

The second method consists of various voting mechanisms that collect all rules that match the unseen instance and has each class "gather votes" from them. This

approach places less importance on the prediction of *individual* rules and is related to the *ensembles* idea from machine learning: if predictors' errors are uncorrelated, using several of them should remove many non-systematic errors.

A straightforward method consists of *majority voting*, in which the predicted class label is that predicted by the majority of rules. Alternatively, rules' votes can be weighted, by their accuracy, strength, or support in a given class, for instance, and the class with the strongest vote is predicted. Many pattern-based classifiers use this scheme: CMAR performs weighted voting, discounting rules' vote by their deviation from their potentially maximal χ^2-score, whereas FitCare simply adds up rules relative support per class, as does ARC-BC. CAEP sums up patterns' growth rate multiplied by their relative support in a class, and DeEP takes the proportion of instances in a class that contain any of the voting patterns as the weight of the vote for that class. Harmony includes three voting options: either the *highest-confidence* rule, or *all*, or the top-k rules vote for a particular class, similar to XRules, which also uses different rule strength measures.

CTC has used different options: the decision list, majority vote, and two weighted voting strategies, as has CorClass.

The analogy with machine learning is exploited most in the gBoost algorithm [34]. In gBoost, an analogy is observed between *weak learners* and patterns. This analogy is exploited by modifying the LPBoost boosting algorithm, developed in the machine learning literature, to iteratively search for patterns instead of weak learners. It can be shown that under certain conditions this algorithm finds optimal linear classification and regression models, where patterns are used as features in the linear models. The boosting algorithm operates by iteratively modifying the weights of examples based on the outcome of a linear program.

A particular feature of some sets of rules is that they represent decision trees. Essentially, every path from the root of a decision tree to a leaf of a tree can be seen as a rule that predicts the label of that leaf. All the rules cover disjoint parts of the data. It is hence not surprising that patterns can also be used to represent paths in decision trees. This observation was exploited in the DL8 approach by [30], which showed that by post-processing a set of patterns found under constraints, a decision tree can be constructed that is optimal under certain conditions. The approach differs from Tree^2 (see below) in that each pattern represents a path in the tree, while in Tree^2 each pattern represents a node.

4.2 Indirect Classification

Indirect classification comes in several flavors. First, there are the techniques that partition the data, sort unseen instances into a certain block, and use the majority label of the block's instances in the training data to make the prediction, like decision trees. The Tree^2 and M^{b}T build this kind of classifier. Other machine learning formalisms can also be adopted to work with supervised patterns—the LB algorithm uses a Naïve Bayes-like formulation to derive predictions from the support of patterns in different

classes—different classes have different products of probabilities and the class with the highest probability is predicted.

This is somewhat similar to the Krimp algorithm: in this technique, *coding tables* are created for each class separately, and an unseen instance's label is predicted based on the coding table that compresses it best.

These approaches are arguably still limited by what the pattern themselves can do, although the upshot is that their models are somewhat more understandable. The alternative is to mine patterns as features for use in sophisticated machine learning techniques that can add modeling and generalization capabilities that are missing from symbolic patterns themselves. This is the second big group of techniques: the technique proposed by [22] belongs to it, as does DDPMine, the method introduced by [12], Picker*, fCork, and ReMine.

5 Summary

In this chapter, we have given a high level overview of supervised pattern mining and its application to prediction, specifically classification. We have abstracted from the pattern languages used and structured the chapter along the three main steps involved in building a classifier from class-labeled data: supervised pattern mining, supervised pattern set mining, and classifier construction.

Regarding the first step, we have laid out that many techniques view different classes as separate subsets of the data and evaluate patterns' co-occurrence with one of these subsets. In our opinion, this view clarifies that different quality measures will lead to similar semantical information of patterns, and that different mining approaches can be taken to find patterns that score highly with any of these measures.

Regarding the second step, we have pointed out the similarities to approaches that have been pioneered in machine learning in the context of rule learning, decision tree induction, and instance-based learning. We have interpreted the former two approaches in terms of partitions to show the similarities of existing techniques, and also identified two types of approaches that always manipulate the entire data. Although some pattern set mining techniques, in particular iterative ones, make certain demands on the pattern mining step, most of them can still be combined relatively freely with different pattern mining techniques.

Finally, when it comes to classifier building, we have made the distinction between direct and indirect classification, with the former paralleling rule-based classification in machine learning, and the latter comprising quite a few approaches that mine patterns as *features* for use in propositional learners. As a comparison of references shows, different classifiers also do not track closely with particular pattern or pattern set mining approaches.

In general, in surveying the field we find that many solutions to the three phases have been developed, most of which can be mixed-and-matched rather freely. The field is larger than the algorithms we have mentioned here yet many techniques are arguably variations of the approaches that we have contrasted.

References

1. C. C. Aggarwal. On effective classification of strings with wavelets. In *KDD*, pages 163–172. ACM, 2002.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996. ISBN 0-262-56097-6.
3. M.-L. Antonie and O. R. Zaïane. Text document categorization by term association. In *ICDM*, pages 19–26. IEEE Computer Society, 2002.
4. B. Arunasalam and S. Chawla. CCCS: a top-down associative classifier for imbalanced class distributions. In T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, editors, *KDD*, pages 517–522. ACM, 2006.
5. M. Atzmüller and F. Puppe. SD-Map-a fast algorithm for exhaustive subgroup discovery. In [16], pages 6–17. ISBN 3-540-45374-1.
6. S. D. Bay and M. J. Pazzani. Detecting group differences: Mining constrast sets. *Data Mining and Knowledge Discovery*, 5 (3): 213–246, 2001.
7. B. Bringmann and A. Zimmermann. Tree^2-Decision trees for tree structured data. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 46–58. Springer, 2005.
8. B. Bringmann and A. Zimmermann. One in a million: picking the right patterns. *Knowledge and Information Systems*, 18 (1): 61–81, 2009.
9. B. Bringmann, S. Nijssen, and A. Zimmermann. Pattern-based classification: A unifying perspective. In A. Knobbe and J. Fürnkranz, editors, *From Local Patterns to Global Models: Proceedings of the ECML/PKDD-09 Workshop (LeGo-09)*, pages 36–50, 2009.
10. L. Cerf, D. Gay, N. Selmaoui-Folcher, B. Crémilleux, and J.-F. Boulicaut. Parameter-free classification in multi-class imbalanced data sets. *Data Knowl. Eng.*, 87: 109–129, 2013.
11. H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proceedings of the 23rd International Conference on Data Engineering*, pages 716–725. IEEE, 2007.
12. H. Cheng, X. Yan, J. Han, and P. S. Yu. Direct discriminative pattern mining for effective classification. In *Proceedings of the 24th International Conference on Data Engineering*, pages 169–178. IEEE, 2008.
13. S. Chiappa, H. Saigo, and K. Tsuda. A Bayesian approach to graphy regression with relevant subgraph selection. In *SDM*, pages 295–304. SIAM, 2009.
14. G. Dong, X. Zhang, L. Wong, and J. Li. Caep: Classification by aggregating emerging patterns. In S. Arikawa and K. Furukawa, editors, *Discovery Science*, volume 1721 of *Lecture Notes in Computer Science*, pages 30–42. Springer, 1999. ISBN 3-540-66713-X.
15. W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. S. Yu, and O. Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In Y. Li, B. Liu, and S. Sarawagi, editors, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 230–238. ACM, 2008. ISBN 978-1-60558-193-4.
16. J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors. *Knowledge Discovery in Databases: PKDD 2006,10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Berlin, Germany, September 18–22, 2006, Proceedings*, 2006. Springer. ISBN 3-540-45374-1.
17. F. B. Galiano, J. C. Cubero, D. Sánchez, and J.-M. Serrano. Art: A hybrid classification model. *Machine Learning*, 54 (1): 67–92, 2004.
18. D. Gay, N. Selmaoui, and J.-F. Boulicaut. Pattern-based decision tree construction. In *ICDIM*, pages 291–296. IEEE, 2007.
19. H. Grosskreutz, S. Rüping, and S. Wrobel. Tight optimistic estimates for fast subgroup discovery. In W. Daelemans, B. Goethals, and K. Morik, editors, *ECML/PKDD (1)*, volume 5211 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2008. ISBN 978-3-540-87478-2.

20. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *SIGMOD Conference*, pages 1–12. ACM, 2000. ISBN 1-58113-218-2.
21. B. Kavsek and N. Lavrac. Apriori-SD: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, 20 (7): 543–583, 2006.
22. S. Kramer and L. De Raedt. Feature construction with version spaces for biochemical applications. In C. E. Brodley and A. P. Danyluk, editors, *ICML*, pages 258–265. Morgan Kaufmann, 2001. ISBN 1-55860-778-1.
23. N. Lavrač, B. Kavsek, P. A. Flach, and L. Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5: 153–188, 2004.
24. D. Leman, A. Feelders, and A. J. Knobbe. Exceptional model mining. In *ECML/PKDD (2)*, pages 1–16, 2008.
25. W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 369–376, San José, California, USA, Nov. 2001. IEEE Computer Society.
26. J. Li, G. Dong, K. Ramamohanarao, and L. Wong. A new instance-based lazy discovery and classification system. *Machine Learning*, 54 (2): 99–124, 2004.
27. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 80–86, New York City, New York, USA, Aug. 1998. AAAI Press.
28. D. Meretakis and B. Wüthrich. Extending naïve bayes classifiers using long itemsets. In U. M. Fayyad, S. Chaudhuri, and D. Madigan, editors, *KDD*, pages 165–174. ACM, 1999. ISBN 1-58113-143-7.
29. S. Morishita and J. Sese. Traversing itemset lattices with statistical metric pruning. In *Proceedings of the Nineteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 226–236, Dallas, Texas, USA, May 2000. ACM.
30. S. Nijssen and É. Fromont. Optimal constraint-based decision tree induction from itemset lattices. *Data Min. Knowl. Discov.*, 21 (1): 9–51, 2010.
31. S. Nijssen and J. N. Kok. Multi-class correlated pattern mining. In F. Bonchi and J.-F. Boulicaut, editors, *KDID*, volume 3933 of *Lecture Notes in Computer Science*, pages 165–187. Springer, 2005. ISBN 3-540-33292-8.
32. P. K. Novak, N. Lavrac, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10: 377–403, 2009.
33. H. Saigo, N. Krämer, and K. Tsuda. Partial least squares regression for graph mining. In Y. Li, B. Liu, and S. Sarawagi, editors, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 230-238. ACM, 2008., pages 578–586. ISBN 978-1-60558-193-4.
34. H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. gboost:a mathematical programming approach to graph classification and regression. *Machine Learning*, 75 (1): 69–89, 2009.
35. M. Thoma, H. Cheng, A. Gretton, J. Han, H.-P. Kriegel, A. J. Smola, L. Song, P. S. Yu, X. Yan, and K. M. Borgwardt. Discriminative frequent subgraph mining with optimality guarantees. *Statistical Analysis and Data Mining*, 3 (5): 302–318, 2010.
36. M. van Leeuwen, J. Vreeken, and A. Siebes. Compression picks item sets that matter. In [16], pages 585–592. ISBN 3-540-45374-1.
37. A. Veloso, W. M. Jr., and M. J. Zaki. Lazy associative classification. In *ICDM*, pages 645–654. IEEE Computer Society, 2006.
38. J. Wang and G. Karypis. Harmony: Efficiently mining the best rules for classification. In *SDM*, 2005.
39. G. I. Webb. Opus: An efficient admissible algorithm for unordered search. *J. Artif. Intell. Res. (JAIR)*, 3: 431–465, 1995.

40. M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12 (3): 372–390, 2000.
41. M. J. Zaki and C. C. Aggarwal. XRules: an effective structural classifier for XML data. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings http://www.nakedcapitalism.com/of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 316–325, Washington, DC, USA, Aug. 2003. ACM.
42. A. Zimmermann and B. Bringmann. Ctc-correlating tree patterns for classification. In J. Han, B. W. Wah, V. Raghavan, X. Wu, and R. Rastogi, editors, *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 833–836, Houston, Texas, USA, Nov. 2005. IEEE.
43. A. Zimmermann and L. De Raedt. Corclass: Correlated association rule mining for classification. In E. Suzuki and S. Arikawa, editors, *Proceedings of the 7th International Conference on Discovery Science*, pages 60–72, Padova, Italy, Oct. 2004. Springer.
44. A. Zimmermann and L. De Raedt. Cluster-grouping: from subgroup discovery to clustering. *Machine Learning*, 77 (1): 125–159, 2009.
45. A. Zimmermann, B. Bringmann, and U. Rückert. Fast, effective molecular feature mining by local optimization. In J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, editors, *ECML/PKDD (3)*, volume 6323 of *Lecture Notes in Computer Science*, pages 563–578. Springer, 2010. ISBN 978-3-642-15938-1.