Incorporating Exceptions: Efficient Mining of ϵ-Relevant Subgroup Patterns

Florian Lemmerich and Martin Atzmueller

University of Würzburg,
Department of Computer Science VI
Am Hubland, 97074 Würzburg, Germany
{lemmerich, atzmueller}@informatik.uni-wuerzburg.de

Abstract. Subgroup discovery is a prominent method for discovering local patterns. However, often a set of very similar (and overlapping) subgroup patterns is retrieved, potentially decreasing its interestingness and suppressing other relevant patterns. This paper presents an approach for the efficient mining of relevant subgroup patterns by allowing exceptions to the strict relevancy criteria. We provide an evaluation of the presented approach using representative data sets and show the impact and benefit of the proposed approach.

1 Introduction

Subgroup discovery is a flexible data mining method for discovering local patterns that can be utilized for global modeling in the context of exploratory data analysis, description, characterization and classification tasks. However, since subgroup discovery is a non-covering approach, in real-world applications often a set of very similar (and overlapping) subgroups is mined. Then, in a typical k-best scenario, a set of very similar subgroup patterns is returned: This can cause a decreased interestingness of the set of patterns, but also a suppression of vital information since further potentially interesting and more diverse subgroup patterns are not presented to the user.

In this paper, we present an efficient method for discovering ϵ-relevant subgroup patterns. This helps to increase the interestingness of the retrieved set of subgroups by handling the problem of overlapping subgroup descriptions that are irrelevant to each other; these non-interesting patterns can then be suppressed. We introduce the concept of ϵ-relevancy in detail, and we propose an efficient algorithm, that is, the *BSD*-algorithm based on efficient bitsets for the discovery of ϵ-relevant subgroup patterns. A detailed evaluation of the proposed methods applying representative (real-world) data sets demonstrates the impact and benefit of the presented approach.

The rest of the paper is structured as follows: Section 2 summarizes the basics of subgroup discovery, and (ir-)relevant patterns. After that, Section 3 presents the approach for the efficient mining of ϵ-relevant subgroup patterns: We first provide the theoretical foundations of the presented technique, show how the relevant patterns can be determined, introduce the efficient *BSD*-algorithm and finally discuss related work. An evaluation of the approach is given in Section 4. Finally, Section 5 concludes with a summary and interesting directions for future research.

2 Preliminaries

In the following, we first introduce the necessary notions concerning the used knowledge representation, before we introduce subgroup mining, subgroup discovery, and the relation between relevant subgroup patterns and closed (item-)sets.

2.1 Subgroup Mining and Subgroup Discovery

The main application areas of subgroup mining are exploration and descriptive induction to obtain an overview of the relations between a (dependent) target variable and a set of explaining (independent) variables. Then, the goal is to uncover properties of the selected target population of individuals featuring the given target property of interest. Therefore, not necessarily complete relations but also partial relations, i.e., (small) subgroups with "interesting" characteristics can be sufficient. Specifically, these interesting subgroups should have the most unusual (distributional) characteristics with respect to the concept of interest given by the target variable [1]. Subgroup discovery is the core data mining step of the subgroup mining process, e.g., [2], that is usually implemented using automatic (algorithmic) and interactive techniques, e.g., visualization methods.

A subgroup discovery task mainly relies on the following four main properties: the target variable, the subgroup description language, the quality function, and the discovery strategy. Since the search space is exponential concerning all the possible selectors of a subgroup description efficient discovery methods are necessary.

For some basic notation, let Ω_A denote the set of all attributes. For each attribute $a \in \Omega_A$ a range $dom(a)$ of values is defined. Let DB be the database (dataset) containing all available cases (instances). A case $c \in DB$ is given by the n-tuple $c = ((a_1 = v_1), \ldots, (a_n = v_n))$ of $n = |\Omega_A|$ attribute values, $v_i \in dom(a_i)$ for each a_i.

The subgroup description language specifies the individuals belonging to the subgroup. For a commonly applied single-relational propositional language a subgroup description can be defined as follows:

Definition 1 (Subgroup Description). *A subgroup description $sd(s) = \{e_1, \ldots, e_n\}$ of the subgroup s is defined by the conjunction of a set of selection expressions (selectors). The individual selectors $e_i = (a_i, V_i)$ are selections on domains of attributes, $a_i \in \Omega_A, V_i \subseteq dom(a_i)$. We define Ω_E as the set of all selection expressions and Ω_{sd} as the set of all possible subgroup descriptions.*

A subgroup s described by $sd(s)$ is given by all cases $c \in DB$ covered by the subgroup description $sd(s)$. A subgroup s' is called a *refinement* of s, if $sd(s) \subset sd(s')$.

A quality function measures the interestingness of the subgroup and is used to rank these. Typical quality criteria include the difference in the distribution of the target variable concerning the subgroup and the general population, and the subgroup size.

Definition 2 (Quality Function). *Given a particular target variable $t \in \Omega_E$, a quality function $q : \Omega_{sd} \times \Omega_E \rightarrow R$ is used in order to evaluate a subgroup description $sd \in \Omega_{sd}$, and to rank the discovered subgroups during search.*

For binary target variables, examples for quality functions are given by

$$q_{WRACC} = \frac{n}{N} \cdot (p - p_0)\,, \quad q_{PS} = n \cdot (p - p_0)\,, \quad q_{LIFT} = \frac{p}{p_0}\,, n \geq \mathcal{T}$$

where p is the relative frequency of the target variable in the subgroup, p_0 is the relative frequency of the target variable in the total population, $N = |DB|$ is the size of the total population, n denotes the size of the subgroup, and $\mathcal{T}$ specifies a minimal size constraint for the subgroup.

As discussed by Lavrac et al. [3] q_{WRACC} (weighted relative accuracy) trades off the increase in the target share p vs. the generality (n) of the subgroup. The Piatetsky-Shapiro quality function q_{PS} (e.g., [4]) is a variation of q_{WRACC} without considering the size of the total population. Finally, the quality function q_{LIFT} focuses on the decrease/increase of the target share.

2.2 Relevant Subgroup Patterns

Let T be the set of positive examples, i.e., the cases for which the target concept is $true$, and F be the set of negative examples. Given a subgroup description $sd(s)$, we denote the set of positive examples, which fulfill sd and in which the target concept is true as $TP(s) = T \cap s$. Analogously, we denote as $FP = F \cap s$ the set of cases, which fulfill sd and in which the target concept is $false$.

As proposed in [5] in the context of closed sets for labeled data, we consider a subgroup s as relevant with respect to another subgroup description s', if it covers all positive examples of s', but no negative examples not covered by s' as well. Formally s' is irrelevant with respect to s, iff

$$TP(s') \subseteq TP(s) \text{ and } FP(s) \subseteq FP(s').$$

3 Subgroup Mining of Relevant Patterns

In the following section we present strategies for the efficient mining of relevant subgroup patterns. We first introduce the concept of ϵ-relevant subgroups and discuss basic strategies for the filtering of subgroups. Next, we discuss a method for fast and effective mining of the set of the best k relevant subgroups.

3.1 ϵ-relevancy

While the concept of strict irrelevance provides good results in many scenarios, it lacks applicability in other real world scenarios. This is partially due to the fact, that even after the filtering of irrelevant subgroups, there remain subgroups patterns, that are considered as irrelevant by domain specialists: For example, these could cover *almost* the same cases as another subgroup, and are therefore perceived as irrelevant.

So, since the difference with respect to the covered cases might be caused by noisy data, these subgroup patterns are expected to be treated in a way similar to the filtered subgroup patterns. However, the technical definition of irrelevance does not cover these. To overcome these problems we propose a generalized definition for the irrelevance of subgroup patterns shown below.

Definition 3. *We consider a subgroup s as $\epsilon_{tp}/\epsilon_{fp}$ irrelevant with respect to a subgroup s' if and only if there exists $E_{tp} \subseteq DB, E_{fp} \subseteq DB$, such that:*

$$TP(s) \subseteq TP(s') \cup E_{tp}, |E_{tp}| \leq \ \epsilon_{tp} \textit{ and}$$
$$FP(s') \subseteq FP(s) \cup E_{fp}, |E_{fp}| \leq \ \epsilon_{fp}$$

As a simplification we use a single variable $\epsilon = \epsilon_{tp} + \epsilon_{fp}$ to specify the boundary for irrelevance:

Definition 4. *We consider a subgroup s as ϵ-irrelevant with respect to a subgroup s' if and only if there exist $E_{tp} \subseteq DB, E_{fp} \subseteq DB$, such that:*

$$TP(s) \subseteq TP(s') \cup E_{tp}, |E_{tp}| \leq \ \epsilon_{tp} \textit{ and}$$
$$FP(s') \subseteq FP(s) \cup E_{fp}, |E_{fp}| \leq \ \epsilon_{fp} \textit{ and}$$
$$\epsilon_{tp} + \epsilon_{fp} \leq \epsilon$$

A subgroup s is ϵ-irrelevant with respect to a set R of subgroups, if R contains any subgroup $s' \in R$, such that s is irrelevant with respect to s'. By this definition a subgroup s is ϵ-irrelevant to another subgroup s', if there exist a set of ϵ cases (*exceptions*) in the database, such that s is irrelevant (according to the traditional definition [5]) with respect to s', when these cases are excluded from the database.

It is easy to see, that this definition includes the existing definition of irrelevant subgroups [5] as a special case, using $\epsilon = 0$. Furthermore, if $S_i(s)$ is the set of subgroups, that are i-irrelevant with respect to a subgroup s, then $S_0(s) \subseteq S_1(s) \subseteq \ldots \subseteq S_k(s)$ holds for any s, as i-relevancy implies $i+1$ relevancy by definition.

	A	B	C	D	E	Target
1	-	+	+	+	+	+
2	+	+	+	+	+	+
3	+	+	+	+	+	+
4	-	-	+	-	-	+
5	-	-	+	-	-	+
6	+	+	+	+	+	-
7	-	+	+	+	-	-
8	-	+	+	-	-	-
9	-	-	+	-	-	-
10	-	-	-	-	-	-

Fig. 1. A small example dataset. It shows 10 cases, the coverage of the subgroups A, B, C, D and the value of the target concept (class label)

We illustrate the idea of ϵ-irrelevancy using the small sample dataset shown in figure 1: Using the traditional definition, the subgroups A and B both are relevant with respect to each other, as B indeed covers more cases labeled as false, but also one single case more, that is labeled positive. However, B is considered 1-irrelevant with respect to

A, as it gets irrelevant, if case 1 is excluded from the database. On the other hand, all positive examples covered by B are also covered by C, which also covers some more positive examples. Since C covers one negative case more than B, B is considered as relevant with respect to C in the traditional sense. By excluding case 9 from the database B gets irrelevant and is thus considered 1-irrelevant with respect to C.

Equi-relevant subgroups and potential problems Using 0-relevancy, if two subgroups are mutually irrelevant to each other, then both subgroups cover exactly the same cases. This does not occur, if we use the notion of generalized ϵ-relevancy. Using the example dataset from figure 1, consider the two subgroups A and D. Using the traditional irrelevance definition both subgroups are relevant with respect to each other. However, when excluding case 7, A gets irrelevant to D and when excluding case 1 D gets irrelevant to A. Thus A is 1-irrelevant to D and D is 1-irrelevant to A. We call this type of relationship *equi-relevant*. Please note, that this equi-relevancy is not an equivalence relation, i.e., it is not transitive. E.g., in the sample dataset the subgroups E is equi-relevant with respect to subgroup D and D is equi-relevant with respect to subgroup B, but E is *not* equi-relevant with respect to B. Therefore, we propose the concept of a *chain of equi-relevant patterns*:

Definition 5. *A chain of equi-relevant patterns is a set C of subgroups, such that for each pair of subgroups s_x and s_y in C, there exists a sequence of subgroups $(s_1, \ldots, s_n), s_i \in C$, such that s_i is equi-relevant to s_{i+1}, s_x is equi-relevant to s_1 and s_y is equi-relevant to s_n.*

If all subgroups in the chain are ϵ-irrelevant to a subgroup s_h, then we call s_h a *head of the chain C*. Please note, that there may be more than one head in a chain, but there is not necessarily a head in the chain.

Chains of equi-relevant subgroups can be used to solve a problem during the mining of ϵ-relevant subgroup patterns: If we completely discard one of two equi-relevant subgroups s_1 and s_2, which cover different cases, then further subgroups found later in the mining process may be irrelevant to the discarded subgroup, but not to the one left in the result set. Thus, both subgroups should be arranged in one chain in the result set, but should be saved separately. The mining process then results not in a set of subgroups, but in a set of chains of equi-relevant patterns. For the result presentation representative subgroups can be chosen for each of these chains. We can consider the following selection criteria for determining these representatives:

- Head of Chain: All subgroups in the chain are ϵ-irrelevant to the representing subgroup.
- Quality: The representing subgroup has the best quality in the chain. Please note, that the subgroup quality should only differ slightly using reasonably small ϵ values.
- Description Length: Since short subgroup descriptions, i.e., descriptions containing only few selectors, are in general better to interpret by the user, subgroups with a short description should be preferred.
- User Defined Ordering: Some influencing factors could be marked as more relevant to the user, e.g., by giving an ordering of the considered attributes.

In practice, the value ϵ should be rather small, so chains should be of limited length.

3.2 Basic Mining Strategies

In the following section we first outline two (naive) approaches for determining the k best relevant subgroups. These can easily complement standard subgroup mining algorithms for determining the relevant patterns. Section 3.3 considers advanced methods and proposes the *BSD*-algorithm, that is based on efficient bitset-based techniques.

Naive Subgroup Filtering 1: Postprocessing The simplest filtering approach requires the application of a standard (exhaustive) discovery algorithm, e.g., the SD-Map* [6] or the DpSubgroup algorithm [7], without incorporating adaptations. Then, any irrelevant subgroup patterns can be removed during a separate postprocessing step.

While this procedure allows for a relatively fast mining process, the size of the result set is rather unpredictable, as the number of irrelevant, filtered subgroups is dataset dependent. As shown in the evaluation in Section 4, it can vary from few single subgroups to the vast majority of the result set. Thus, in order to find the set of the best k relevant subgroups, a very large set of subgroups must be retrieved in the search-algorithm, thus strongly limiting the pruning possibilities of the applied methods. However, even then there is no guarantee, that enough relevant subgroups have been discovered in order to mine a result set containing the k best and relevant subgroups.

Naive Subgroup Filtering 2: During Search To avoid a potentially massive reduction of the size of the result set during postprocessing, a relevancy-check can be included into the subgroup discovery algorithm. Thus, whenever a subgroup s is added to the set S of the best k subgroups, it is checked, if s is irrelevant to any subgroup $s' \in S$. If this is not the case, then the subgroup s can be added to the set S. Afterwards for any subgroup $s'' \in S$ it is checked, if s'' is covered by s and thus can be removed. Please notice, that by doing so more than one subgroup can be removed from the set, while only one is added.

However, in contrast to the postprocessing approach the resulting reduction of the size of the result set is very limited in practice, so potential problems can be avoided by only a slight increase of the size of the result set in most cases. This assessment is also confirmed by our evaluation (see section 4).

The naive approach requires one pass over the database for each such check. Thus, the time needed to check for irrelevant subgroups can easily exceed the time needed for the regular subgroup search by far.

First improvements can be made by reducing the amount of required checks: In order to store the result set of subgroups we use a linked list, which is sorted by the subgroups quality. If a subgroup s is irrelevant in respect to another subgroup s', then $q(s) \leq q(s')$. When adding a subgroup to the result set, we can use this relationship and test only subgroups $\hat{s}$ with $q(\hat{s}) \geq q(s)$, if they cover s and identify s as an irrelevant subgroup. On the other hand, if s is relevant and added to the result set, then only subgroups $\bar{s}$ with $q(\bar{s}) \leq q(s)$ can be irrelevant to s and need to be tested.

3.3 BSD: A Bitset based Subgroup Discovery Algorithm

Checking each possible coverage of subgroups in a separate database pass is obviously a very time consuming task. To speed up this process we propose a new vertical subgroup mining algorithm, which is tailored to the task of finding relevant subgroups. It utilizes a vertical, *bitset based* representation of the needed information. Bitsets are vectors of set and unset bits, which are implemented very time and memory efficient in most programming languages. In Java, for example, even a boolean primitive needs 4 bytes of memory space, while one value inside a bitset just needs one single bit. In addition, logical operations, e.g. logical or/and, are implemented very efficiently for bitsets in most common programming languages.

The bitset based subgroup discovery algorithm BSD for exhaustive mining of relevant sets of subgroups uses a branch-and-bound strategy, where a conditioned search space is mined recursively, similar to the SD-Map* [6] or the DpSubgroup algorithm, e.g., [7]. In the initialization phase we construct a bitset for each selector involved in the subgroup discovery. The length of each bitset is given by the number of cases (instances) in the database. The ordering of the cases in the database is the same for all selectors, so the i-th bit in each selector describes the i-th instance in the database. Additionally, a bitset is constructed for the target concept in the same way. This bitset has set bits on all positions, where the respective case is labeled as true. The construction of these bitsets can be accomplished in one single pass through the database. The rest of the algorithm can operate on the generated data structures and does not need further database passes. The memory consumption of the bitsets for n instances in the database and m selectors is then given by $n \cdot (m + 1)$ bits.

After the initialization, the main recursive step is called with no conditioned selectors, a current bitset, in which all bits are set and all selectors possibly relevant for the search. The main step is shown in Algorithm 1. It consists of two phases: During phase one (lines 1 to 20) all possible refinements using the possibly relevant selectors sel_{rel} are considered. For each of these selectors the subgroup description, that consists of the current conditioned selectors sel_{cond} and this new selector s_{curr}, is evaluated. Therefore, the bitset representation $c_{current}$ of all cases, which fulfill this new description, is computed by performing a logical AND operation on the current bitset of all cases and the bitset of the new selector (line 3). In doing so, the i-th bit in the resulting bitset is set, if all conditioned selectors and the new selector apply to the respective instance. Thus, the cardinality (i.e., the number of ones) of this bitset determines the number of cases n described by these selectors. If this cardinality exceeds the required minimum support for the result, the number tp of these cases with a positive target concept is computed by performing another logical AND with the bitset of positives target concepts (line 6) and then again counting the set bits.

By using the values n and tp an optimistic estimate can be computed, e.g., [1, 7]. If this optimistic estimate of the subgroup quality indicates, that there may be refinements of the current subgroup with the new selector, that have a sufficient quality to be included in the overall result, then this selector is added to the list of relevant selectors for the next level of search (line 9). Additionally it is tested, if the quality of the current subgroup description (consisting of the conditioned selectors and the new selector) is high enough to be added to the result set.

Algorithm 1 function bsd

```
Require:
    $sel_{cond}$: List of conditioned selectors,
    $sel_{rel}$: List of potentially relevant selectors,
    $c_{cond}$: Bitset of instances that fulfill each $s_{curr} \in sel_{cond}$,
    *depth*: Current search depth,
    *result*: The result set
Ensure:
    *result* as a set of the best relevant subgroups

 1: $newSel_{rel}$:= new List()
 2: for all Selector $s_{curr}$ in $sel_{rel}$ do
 3:     $c_{current}$= $c_{cond}$ AND ($s_{curr}$.bitset)
 4:     $n$ = $c_{current}$.cardinality()
 5:     if $n$ > MinTP then
 6:         $c_{currentPos}$= $c_{current}$ AND (positives)
 7:         $tp$ = $c_{currentPos}$.cardinality()
 8:         if fulfillsMinSupport ($tp$)
            AND optEstimateHighEnough (*result*, $tp$, $n$) then
 9:             $newSel_{rel}$.add ($s_{curr}$)
10:             if qualityHighEnough (*result*, $tp$, $n$) then
11:                 *r* = checkRel (*result*, $c_{current}$, $c_{currentPos}$)
12:                 if *r* then
13:                     *sg* = createSubgroup ($sel_{cond}$, $s_{curr}$)
14:                     *result*.add (*sg*, $c_{current}$, $c_{currentPos}$)
15:                     *result*.checkRelevancies(*sg*)
16:                 end if
17:             end if
18:         end if
19:     end if
20: end for
21: sort ($sel_{rel}$)
22: if *depth* < MAXDEPTH then
23:     for all Selector $relSel$: $sel_{rel}$ do
24:         if optEstimateHighEnough ($relSel$) then
25:             $newSel_{rel}$.remove ($relSel$)
26:             $sel_{cond}$.add ($relSel$)
27:             $c_{new}$= getCurrentBitSetFor ($relSel$)
28:             bsd ($sel_{cond}$, $newSel_{rel}$, *depth* + 1, $c_{new}$, *result*)
29:             $newSel_{rel}$.add($relSel$)
30:             $sel_{cond}$.remove ($relSel$)
31:         end if
32:     end for
33: end if
```

In this case a relevancy check is performed (line 11). If the subgroup is considered as relevant, it is saved in the result set, together with the bitset representations of the positive cases and all cases of the subgroup (line 14). Additionally, if any subgroup already contained in the result set is irrelevant to the new subgroup, then this subgroup is removed from the result (line 15).

Using the stored bitset representations, it can be efficiently checked, if a subgroup s is irrelevant with respect to another subgroup s': A representation of all cases with a positive target concept, that are contained in s, but not in s' can be computed by performing a logical AND-NOT between the positives bitset of s and s'. Analogously, a bitset reflecting all cases labeled as negative, that are contained in s' and not in s can be computed by another AND-NOT operation between the respective bitsets of negative cases. If both bitsets resulting from these operations do not contain any set bits, then the s is irrelevant.

At the start of phase two, the list of relevant selectors is sorted by their computed optimistic estimate (line 21). By doing so, more promising paths in the search space are evaluated first, so more branches of the search tree can be pruned earlier. Afterwards for each relevant selector this selector is added to the list of conditioned selectors and the main procedure of the algorithm is called recursively using the respective bitset and the conditioned selectors.

3.4 Related Work

The issue of relevancy is a very important topic in machine learning and data mining. For example, it has been introduced for feature selection in the machine learning context by Koller and Sahami [8]. Lavrac and Gamberger [9] apply relevancy filtering for feature extraction/filtering. This approach was later extended by Garriga et al. [5], covering closed itemsets and putting relevancy in context with supervised learning techniques. In contrast to these techniques, the presented approach is able to incorporate some ’fuzziness’ concerning the relevancy definition, and so larger parts of the pattern space can be filtered. This is especially useful concerning exceptions and noise in the data, which cannot be handled by the existing approaches.

For the subgroup discovery task several efficient approaches for exhaustive mining have been proposed, e.g., SD-Map [10] and SD-Map* [6]. Similarly, Grosskreutz et al. [7] introduced the DpSubgroup algorithm that incorporates pruning using tight optimistic estimates. While the general structure of the DpSubgroup algorithm is somewhat similar to the BSD algorithm, the underlying data representation is completely different. BSD algorithm uses a vertical data structure utilizing bitsets, while SD-Map, SD-Map* and DpSubgroup use FP-Trees.

Using vertical data representations for fast data mining algorithms was proposed by Zaki [11]. Burdick et al. [12] used a bitset (bitmap) representation for the mining of maximal frequent itemsets. They also utilize a method for bitset compression, that could be utilized in the future to further adapting the BSD algorithm for very large datasets.

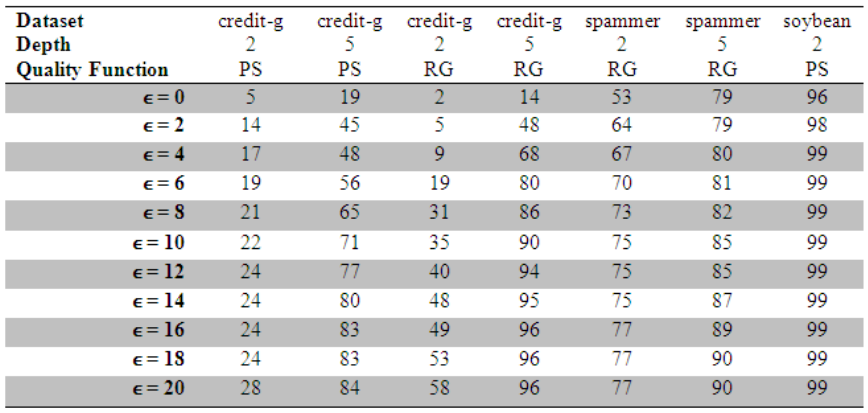
To the best of the authors’ knowledge, up to now no exhaustive specialized algorithm for subgroup discovery has been proposed, that especially focuses on the mining of relevant subgroups.

4 Evaluation

In the following, we present several empirical observations and evaluations with respect to the described concepts and methods using datasets from the UCI-repository [13], real world and synthetic evaluation data.

Evaluation datasets For the evaluation we used a variety of datasets. From the UCI-repository we took the datasets “mushroom”, “soybean” and “credit-g”. Furthermore, the approach was tested with a larger real world dataset. This dataset describes spammers in a social network tagging system and consists of more than 17.000 cases and 25 attributes. Additionally we used synthetic evaluation data generated from a bayesian network, that consists of 10.000 cases and 15 attributes.

Influence of different ϵ values In our first experiments we evaluated, how different values of ϵ influence the number of filtered subgroups. To accomplish this, we performed an exhaustive subgroup search algorithm to find the 100 best subgroups for the target variable. From the results we filtered and counted the ϵ-irrelevant subgroups. In case of equi-relevant subgroups we counted only one to the irrelevant subgroups. For the tests we considered different datasets, different maximum search depths and two different quality functions, i.e., the relative gain and the Piatetsky-Shapiro quality function. The results are depicted in Figure 2.

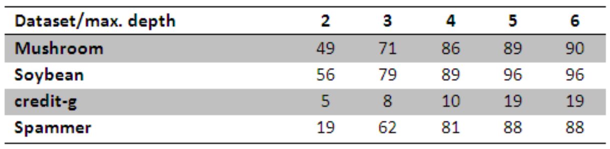


Dataset	credit-g	credit-g	credit-g	credit-g	spammer	spammer	soybean
Depth	2	5	2	5	2	5	2
Quality Function	PS	PS	RG	RG	RG	RG	PS
$\epsilon = 0$	5	19	2	14	53	79	96
$\epsilon = 2$	14	45	5	48	64	79	98
$\epsilon = 4$	17	48	9	68	67	80	99
$\epsilon = 6$	19	56	19	80	70	81	99
$\epsilon = 8$	21	65	31	86	73	82	99
$\epsilon = 10$	22	71	35	90	75	85	99
$\epsilon = 12$	24	77	40	94	75	85	99
$\epsilon = 14$	24	80	48	95	75	87	99
$\epsilon = 16$	24	83	49	96	77	89	99
$\epsilon = 18$	24	83	53	96	77	90	99
$\epsilon = 20$	28	84	58	96	77	90	99

Fig. 2. Number of ϵ-irrelevant subgroups after an exhaustive search for the 100 best subgroups, using different ϵ

These results indicate, that an increase of ϵ leads to a significant increase in the number of irrelevant subgroups in all datasets. The increase is stronger for the relative gain quality functions and for higher maximum search depths, as the subgroups in the result set tend to be smaller in these cases. This increases the likelihood of ϵ-irrelevance using a constant value of ϵ. However, which value of ϵ is useful in an application is strongly dependent on the quality of the underlying data and should be considered carefully together with domain specialists.

Post-processing Next, we evaluate the post-processing approach with respect to the task of finding relevant subgroup patterns. To estimate the applicability of this method, we investigated, how many subgroups remained in the result set after the filtering process. For a more general result, we used 0-relevancy in this evaluation. The results are shown in Figure 3.



Dataset/max. depth	2	3	4	5	6
Mushroom	49	71	86	89	90
Soybean	56	79	89	96	96
credit-g	5	8	10	19	19
Spammer	19	62	81	88	88

Fig. 3. Number of 0-irrelevant subgroups after an exhaustive search for the 100 best subgroups, using different datasets and maximum depth boundaries. As quality function the Piatetsky-Shapiro quality functions was applied.

It is easy to see, that the number of irrelevant subgroup patterns in the result set is difficult to estimate. So, in order to guarantee a minimum size of the result set after the filtering process a huge number of saved subgroup descriptions would be needed. This indicates, that for many problems filtering irrelevant subgroups in a post processing step is not practical. In comparison, we performed the same tests with filtering steps during the search whenever the subgroup changed, and in none of the performed tests the (theoretically possible) phenomenon occurred, that the algorithm ends up with less than the required subgroup size. These results stress the importance of a subgroup discovery method, that can seamlessly integrate the filtering of irrelevant subgroups during the search process.

BSD Algorithm In the last part of our evaluation we analyze the runtime of the presented novel subgroup discovery algorithm, i.e., the Bitset based subgroup discovery algorithm (BSD algorithm). We compared the runtimes of the state-of-the art exhaustive subgroup discovery algorithm DpSubgroup with and without filtering of irrelevant subgroups with the BSD algorithm. The evaluation was done using different datasets and depth boundaries. As quality functions the Piatetsky-Shapiro function and relative gain were used. The results are shown in Figures 4, 5, 6 and 7.

The results indicate, that the naive filtering approach incorporated in the DpSubgroup algorithm has a strong increasing effect on the runtimes. The BSD algorithm performed all tasks significantly faster than the DpSubgroup algorithm when filtering out irrelevant subgroups. In several cases it is more than an order of magnitude faster.

	BSD w/o rel. check	BSD w/ rel. check	DpS w/o rel. Check	DpS w/ rel. check
2	46	391	188	1203
3	46	579	171	1750
4	47	656	157	2296
5	47	688	187	2625
6	47	765	157	2531

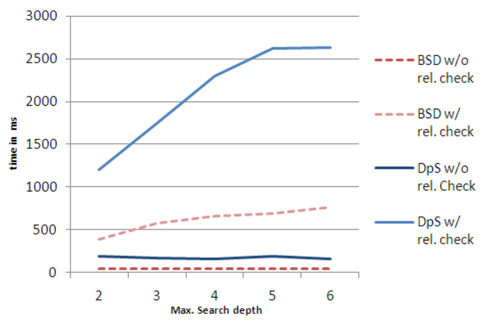


Fig. 4. Runtime evaluation using credit-g dataset.

	BSD w/o rel. check	BSD w/ rel. check	DpS w/o rel. Check	DpS w/ rel. check
2	187	1422	2344	12874
3	219	2234	3343	17812
4	1172	4828	6484	19227
5	5171	9578	7577	19446
6	10281	15577	7734	19705

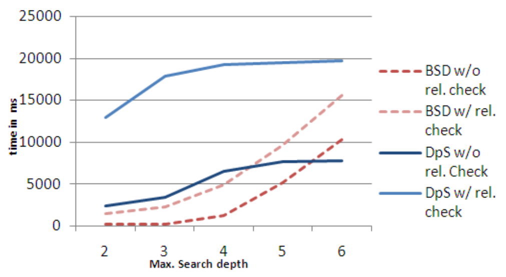


Fig. 5. Runtime evaluation using a synthetic dataset.

In several tests, especially if the maximum-depth is bounded to a low search depth, the BSD algorithm outperforms the DpSubgroup algorithm without relevancy checks. Besides the well integrated tests for irrelevance we explain that by the fact, that the BSD algorithm needs only one database pass compared to the two passes needed by DpSubgroup and the lower construction costs of the used data structures. On the other hand, for large maximum depth boundaries, the runtime of the BSD algorithm shows a sharp increase in some datasets, e.g., the synthetic dataset, in comparison to the Dp-Subgroup algorithm. This can be explained by the elaborated pruning strategies, that can be performed using the more complex data structures, i.e., FP-trees, used in the DpSubgroup algorithm. However, for most practical problems a maximum depth of 5 or 6 can be considered as sufficient, since the search depth only limits the number of describing selectors of a subgroup, and subgroups with more selectors are often very difficult to interpret by humans. Furthermore, only few additional subgroups can be found at larger depths. Therefore, we regard the BSD algorithm as the currently best choice when mining for relevant subgroups and propose to consider it also for subgroup discovery without relevancy checks, for low and medium maximum depths.

	BSD w/o rel. check	BSD w/ rel. check	DpS w/o rel. Check	DpS w/ rel. check
2	31	406	141	2609
3	31	562	110	6640
4	32	609	109	12217
5	47	749	93	17138
6	16	1000	109	19575

	BSD w/o rel. check	BSD w/ rel. check	DpS w/o rel. Check	DpS w/ rel. check
2	242	1453	1140	214433
3	250	2140	1171	855317
4	265	2297	1219	2256945
5	265	2234	1266	4561555
6	266	2266	1281	7452757

Fig. 6. Runtime evaluation using the soybean dataset (on the left) and the mushroom dataset (on the right).

	BSD w/o rel. check	BSD w/ rel. check	DpS w/o rel. Check	DpS w/ rel. check
2	500	2437	2500	232590
3	500	4703	2594	910248
4	515	7296	2750	2067787
5	516	9109	2812	3204517
6	516	9922	2812	3820749

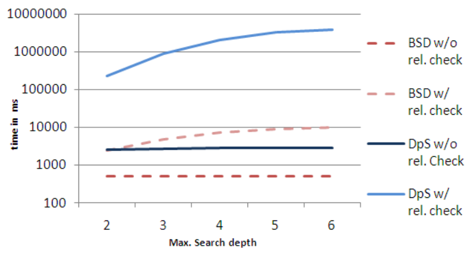


Fig. 7. Runtime evaluation using the real world dataset from the spam characterization domain. Please notice, that the diagram uses a logarithmic scale on the y-axis. DpSubgroup performs up to 2 orders of magnitude slower, if a relevancy check is performed.

5 Conclusions

In this paper, we have presented a method for discovering ϵ-relevant subgroup patterns, in order to handle the problem of overlapping subgroup descriptions that are irrelevant to each other and to make the set of the presented patterns more interesting to the user. We have introduced the concept of ϵ-relevancy, that incorporates exceptions into the relevancy framework. Additionally, we have proposed a new efficient and effective algorithm for the discovery of (relevant) subgroup patterns. The evaluation demonstrates the impact and benefit of the presented approach.

For future work, we plan to integrate more background knowledge for improving the semantic features of the approach. Furthermore, bitset compression techniques for further advancing the scalability of the algorithm, and suitable visualization techniques for inspecting the set of relevant and the (suppressed) set of irrelevant patterns are additional interesting research directions.

Acknowledgements

This work has been partially supported by the German Research Council (DFG) under grant Pu 129/8-2.

References

1. Wrobel, S.: An Algorithm for Multi-Relational Discovery of Subgroups. In: Proc. 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-97), Berlin, Springer Verlag (1997) 78–87
2. Atzmueller, M., Puppe, F., Buscher, H.P.: Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery. In: Proc. 19th Intl. Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland (2005) 647–652
3. Lavrac, N., Kavsek, B., Flach, P., Todorovski, L.: Subgroup Discovery with CN2-SD. Journal of Machine Learning Research **5** (2004) 153–188
4. Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI Press (1996) 249–271
5. Garriga, G.C., Kralj, P., Lavrač, N.: Closed sets for labeled data. J. Mach. Learn. Res. **9** (2008) 559–580
6. Atzmueller, M., Lemmerich, F.: Fast subgroup discovery for continuous target concepts. In: Proc. 18th International Symposium on Methodologies for Intelligent Systems (ISMIS 2009), Springer Verlag (2009) accepted paper
7. Grosskreutz, H., Rüping, S., Shaabani, N., Wrobel, S.: Optimistic estimate pruning strategies for fast exhaustive subgroup discovery. Technical report, Fraunhofer Institute IAIS, http://publica. fraunhofer.de/eprints/urn:nbn:de:0011-n-723406.pdf (2008)
8. Koller, D., Sahami, M.: Toward optimal feature selection, Morgan Kaufmann (1996) 284–292
9. Lavrac, N., Gamberger, D.: Relevancy in Constraint-based Subgroup Discovery. In Jean-Francois Boulicaut, Luc de Raedt, H.M., ed.: Constraint-based Mining and Inductive Databases. Volume 3848 of LNCS. Springer Verlag, Berlin (2006)
10. Atzmueller, M., Puppe, F.: SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery. In: Proc. 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006). Number 4213 in LNAI, Berlin, Springer Verlag (2006) 6–17
11. Zaki, M.J.: Efficient enumeration of frequent sequences. In: CIKM '98: Proceedings of the seventh international conference on Information and knowledge management, New York, NY, USA, ACM (1998) 68–75
12. Burdick, D., Calimlim, M., Gehrke, J.: Mafia: A maximal frequent itemset algorithm for transactional databases. In: In ICDE. (2001) 443–452
13. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/∼mlearn/mlrepository.html (1998)