ROCsearch — An ROC-guided Search Strategy for Subgroup Discovery*

Marvin Meeng† Wouter Duivesteijn‡ Arno Knobbe§

Abstract

Subgroup Discovery (SD) aims to find coherent, easy-to-interpret subsets of the dataset at hand, where something exceptional is going on. Since the resulting subgroups are defined in terms of conditions on attributes of the dataset, this data mining task is ideally suited to be used by non-expert analysts. The typical SD approach uses a heuristic beam search, involving parameters that strongly influence the outcome. Unfortunately, these parameters are often hard to set properly for someone who is not a data mining expert; correct settings depend on properties of the dataset, and on the resulting search landscape. To remove this potential obstacle for casual SD users, we introduce ROCsearch, a new ROC-based beam search variant for Subgroup Discovery. On each search level of the beam search, ROCsearch analyzes the intermediate results in ROC space to automatically determine a sensible search width for the next search level. Thus, beam search parameter setting is taken out of the domain expert's hands, lowering the threshold for using Subgroup Discovery. Also, ROCsearch automatically adapts its search behavior to the properties and resulting search landscape of the dataset at hand. Aside form these advantages, we also show that ROCsearch is an order of magnitude more efficient than traditional beam search, while its results are equivalent and on large datasets even better than traditional beam search results.

Keywords: Subgroup Discovery, Beam Search, ROC Analysis

1 Introduction

Subgroup Discovery (SD) [1, 2] aims to find coherent subsets of the dataset at hand, where something exceptional is going on with respect to a designated target column. Consider for instance a dataset concerning medical records of people, in which the target indicates whether a person has developed lung cancer. Then, Subgroup Discovery should report the group of smokers, as well as the group of people who have worked with asbestos, since these groups tend to have an unusually high incidence of lung cancer. SD might alternatively be employed to find groups with an unusually *low* incidence of lung cancer, which could indicate previously unknown risk-reducing factors for this disease.

Since SD aims at finding subsets of a dataset, the search space is potentially exponentially large in the number of records. There are two main schools of thought in the SD community on how to overcome this computational problem. On the one hand, when we disallow numeric attributes in the data, and restrict the type of exceptionality that we search for, then anti-monotonicity constraints can be used to prune away significant parts of the search space. In practice, this makes the SD algorithm tractable for reasonably-sized datasets. On the other hand, many quantities that we can measure in real life come in numeric form: temperatures, distances, a smartphone's battery life, et cetera. The SD framework has the potential to be much more powerful when such attributes can be used without any pre-analysis discretization. To allow for that, and still keep the process tractable, we need to turn to heuristic search; usually a form of *beam search* is selected.

Beam search is a level-wise search strategy, in which the candidate space is traversed in a general-to-specific manner. On the first level, relatively general subgroups are considered, of which the most promising w (for *search width*) are selected as the *beam*. On the second level, we only consider subgroups that are specializations (i.e.: coherent subsets) of the subgroups in the beam, and again, the w most promising subgroups are selected to form the beam for the next level. By limiting the number of levels and controlling the search width, the end-user can control the amount of time invested in the search. Meanwhile, the beam search strategy finds the middle ground between a completely greedy method and exhaustive search; on the one hand, selecting only the w most promising subgroups on each level keeps the process focused, while on the other hand, considering w alternatives decreases the likelihood that the process ends up in a local optimum.

Performance of the beam search algorithm for SD is strongly dependent on the chosen search width. Setting the width too low will lead to missing interesting results. Setting the width too high will lead to consideration of uninteresting results. Unfortunately, in practice, the search width is nontrivial to set. A proper value strongly depends on many properties of the dataset at hand (column types and cardinalities, the number of columns

*This research is supported in part by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis".

†LIACS, Leiden University, m.meeng@liacs.leidenuniv.nl

‡Fakultät für Informatik, LS VIII, Technische Universität Dortmund, wouter.duivesteijn@tu-dortmund.de

§LIACS, Leiden University, a.j.knobbe@liacs.leidenuniv.nl

and rows, the number of underlying concepts present in the data), and the type of subgroup a domain expert is interested in (for instance, in the medical domain, small but very precise subgroups are usually preferred, while statistics for newspaper headlines should concern as many people as possible). Therefore, we introduce ROCsearch, a new search strategy for SD, marrying concepts from beam search and ROC analysis [3, 4], in which the value for this sensitive parameter w is automatically learned from the data and tuned by the amount of underlying concepts encountered in the data.

The subgroups under consideration for the beam in ROCsearch, are evaluated by mapping them in *ROC space*, detailing their *False Positive Rate* and *True Positive Rate*. Within this set, the subgroups that are optimal in some sense lie on the *ROC convex hull* [5]. The subgroups that lie on the ROC convex hull form the beam for the next level. Thus, on each search level, w is automatically determined. Also, selecting the subgroups that lie on the ROC convex hull neatly mirrors the fact that the most common quality measures for Subgroup Discovery are convex in ROC space.

ROCsearch will automatically adjust the search width of the beam search process to the peculiarities of the dataset at hand. Hence, ROCsearch takes away the responsibility to set this parameter from the domain expert, who is usually ill-equipped to perform this task, yet is punished with bad results when setting the parameter wrongly. Thus, ROCsearch makes Subgroup Discovery substantially more user-friendly. Moreover, ROCsearch is an order of magnitude more efficient than traditional beam search, while the results that it finds are equivalent on small datasets and even better on large datasets, compared to the results found with traditional beam search.

2 Related Work and Preliminaries

Throughout this work we assume a dataset Ω, consisting of N *records* of the form $x = (a_1, \ldots, a_m, \ell)$. We allow the *attributes* $\{a_1, \ldots, a_m\}$ to come from an unrestricted domain $\mathcal{A}$. The *target*, ℓ, must have a type that can be handled as the target in ROC analysis. In practice, we consider either binary targets, or nominal targets that can be treated naturally with a one-versus-all strategy.

2.1 Subgroup Discovery Subgroup Discovery (SD) [1, 2] is a data mining framework that seeks subsets (satisfying certain user-specified constraints) of the dataset where something exceptional is going on. In this process, we explore a large search space, guided by a user-defined quality measure that expresses the type of exceptionality we seek. Typically, *subgroups* are found by a level-wise search through attribute space [6]. We define constraints on single attributes, and define corresponding subgroups as those records satisfying the constraint. The *quality measure*, which can be any function assigning a numeric value to a subset of the data, determines how exceptional the constructed subgroups are, after which (a selection of) the subgroups are *refined* by adding more constraints on attributes to their definition. When this process ends, the top-k subgroups with the highest quality are returned to the user. Though the quality measure is an unrestricted parameter of the SD framework, practically all sensible quality measures are convex in the False Positive Rate and True Positive Rate [7]; a subgroup is better if it contains more positives and less negatives.

Subgroup Discovery is a supervised technique; there is a *target concept* defined on one or several designated attributes that will henceforth be known as *targets*. The quality measure is usually defined in such a way that subgroups are found that display distributional differences in these targets.

The most traditional form of SD [1, 2] concerns the case where there is only one target, and all attributes and the target are compelled to be nominal. One particular value of the target is singled out as the desired value; records having that particular value are the *positives* in the dataset, and the other records are the *negatives* (hence, a multi-class target is handled with a one-versus-all approach). The goal of SD in this form is to find subgroups that contain as many of the positives and as few of the negatives as possible. By exploiting a property akin to the anti-monotonicity property that is well-known from frequent itemset mining, the search space of nominal-attributes-only SD can be explored exhaustively [8].

Since a nominal-attributes-only setting is quite restrictive, recent focus has been on incorporating *numeric attributes*. These commonly have a very high cardinality, which makes considering all subgroups intractable. A recent inroad towards exhaustive SD on numeric attributes has been made by Grosskreutz [9]. It imposes two restrictions on the SD process: one on the type of quality measure allowed in SD, and one on the type of conditions one is allowed to impose on the numeric attributes. Under these restrictions, another property akin to anti-monotonicity can be formulated, which again allows exhaustive search space traversal.

One of the key strengths of the SD framework is that the quality measure is an *unrestricted* parameter of the algorithm. Hence, a domain expert is allowed to specify precisely what he is interested in finding in the dataset at hand. Ideally, we would want to impose no restrictions on the type of exceptionalities that can be found with SD. Hence, ideally, we want our

SD methods to be able to handle numeric attributes natively, and allow for a generic quality measure. For this setting, no tractable exhaustive algorithm exists. Instead, we usually turn to a *beam search* approach [1]. In such an approach, the user is asked to set a few parameters, bounding the level-wise search through attribute space that is usual in SD. Two of these parameters are especially noteworthy: w, the *search width*, and d, the *search depth*. Recall that on the first search level, candidate subgroups are built by imposing one condition on one attribute; all such subgroups are considered. Following search levels are bounded by w. From the first-level subgroups, w subgroups are selected as the *beam* for the second level. Candidate subgroups for the next level are generated by refining *only the subgroups from the beam*. Again, w subgroups are gathered to form the beam for the next level. The search ends when all d^{th} level candidates are considered.

To get an extensive overview of SD, many survey papers are available, including [10]. It may also be noteworthy that SD is closely related to Contrast Set Mining and Emerging Pattern Mining, as explored by Kralj Novak et al. [11].

2.1.1 SD and Clustering One might be tempted to view SD as a close relative of clustering. After all, a *single* discovered subgroup can be seen as a single cluster. However, this line of reasoning overlooks the substantial differences between *the overall task* of clustering, and the overall SD task. For instance, a clustering algorithm typically divides a majority of the records into (potentially overlapping) groups. On the other hand, the aim of SD is to highlight coherent *exceptional* groups. Hence, records representing the norm in the dataset are typically not included in any high-quality subgroup. Therefore, we consider SD and clustering more as distant cousins.

2.2 ROC Analysis ROC analysis has a rich history in the field of signal processing [3]. Originally it was used to visualize the tradeoff between hit rates and false alarm rates of classifiers, the value of which for algorithm comparison was recognized in the eighties [4]. In its most traditional form, ROC analysis concerns a learning problem with a binary response variable. Whether a classifier or a local pattern mining technique is involved, ROC analysis interprets the results in terms of a tradeoff between the *True Positive Rate* (TPR) and the *False Positive Rate* (FPR).

When the value of ROC analysis for algorithm comparison was recognized, the machine learning community adopted it, initially in order to compare classifiers. Extensive papers have been written on incorporating ROC analysis into machine learning and data mining [12, 13]. ROC analysis has found its way into the data mining community beyond classifiers, particularly in local pattern mining methods such as SD [7].

The ROC methodology analyzes found subgroups in *ROC space*: the two-dimensional unit square with FPR on the horizontal axis and TPR on the vertical axis. The *perfect subgroup*, containing all the positives and no negatives, is found in the top left corner; its coordinates in ROC space are (0,1). The *empty subgroup* is found in the bottom left corner with coordinates (0,0), and the *full subgroup* containing the whole dataset is found in the top right corner with coordinates (1,1). Given the domain at hand, we may desire to assign different costs to the two kinds of imperfections we can have in a subgroup: false positives and false negatives.

Definition (Optimal Subgroup). Suppose a subgroup set $\mathcal{S}$, and a cost assignment to false positives and false negatives. A subgroup $S \in \mathcal{S}$ is called *optimal* within $\mathcal{S}$ for this cost assignment, if its total cost equals the minimum total cost for any subgroup in $\mathcal{S}$.

Hence, a subgroup S is optimal within a subgroup set $\mathcal{S}$ for *some* cost assignment if and only if it lies on the convex hull of $\mathcal{S}$ in ROC space [5].

Recently, ROC analysis has drawn some criticism [14], particularly concerning the usage of the Area Under the ROC Curve (AUC) as a measure of classifier performance or subgroup set quality. The main discussion point was that AUC as a metric for classifier performance seemingly depends on the classifier at hand, which would be very damaging to the practical value of AUC. This led to an exchange of papers (for a recap, see [15]) which helped the community in finetuning how it should interpret certain concepts that occur in ROC analysis. Eventually, a property of the ROC curve that was different than the one discussed in the original paper [14], was shown to be related to expected loss in a classifier, reinstating the role that ROC analysis plays as an objective metric to compare classifiers.

3 Paper Outlook

The *main contribution* of this paper is ROCsearch, a new search strategy for Subgroup Discovery. The search strategy should cater for datasets with numeric attributes, and an arbitrary concept of interest to be sought after. Hence, for the reasons outlined in the previous section, the current state-of-the-art algorithm for this setting is heuristic beam search. What sets ROCsearch apart, is that it automatically determines appropriate values for the hard-to-manually-set search width parameter, which comes with two direct benefits. On the one hand, it fulfills a task which would normally

be the responsibility of the user of the SD algorithm, but which lies in the domain of the data mining expert. This removes a potential obstacle to use the SD algorithm for people who are not data mining experts. On the other hand, the autoconfiguring beam search strategy adapts itself to the peculiarities of the dataset at hand.

A desirable side-effect, although by no means guaranteed by the ROC-based approach, is that the convex hull tends to consist of relatively few candidates, such that in practice, the algorithm will have runtimes comparable to that of regular beam search with an unusually low value of w. We will investigate this computational aspect of the new approach in the experimental section (5), where we compare ROCsearch with a normal-size beam search, to appreciate any speed-advantages, and with an unusually narrow beam search, to verify whether a modest traditional approach might not have produced comparable results.

4 The ROCsearch Algorithm and Other Beam Selection Strategies

Before discussing the ROCsearch algorithm, we would like to devote some words to a philosophical stance regarding SD. The subgroups we consider are formed by (conjunctions of) conditions of the form 'attribute operator value'. Descriptions are equal if and only if they contain the exact same set of conditions. Conditions are equal if and only if the attribute, operator, and value are equal. When a description evaluates to 'true' for a record in the dataset, that record is said to be a member of the subgroup selected by that description. It is important to stress the difference between subgroup intensions and subgroup extensions. It may very well be the case that different descriptions (intensions) select the exact same records (extension) from the dataset. To the domain expert, both descriptions are potentially useful. Considering the lung cancer example again, even when the two descriptions 'smokes=true' and 'retired=true' select the exact same people, they may both be equally valuable to the domain expert, as something new can be learned from each. Therefore, SD should strive to find all interesting descriptions, even when their extensions are identical. This principle will hereafter be referred to as *(Frege's) Sinn principle*, after the distinction between sense and reference introduced by the German mathematician Frege, in his work 'Über Sinn und Bedeutung' [16].

In the following subsections, we will introduce the new ROCsearch algorithm, and outline existing beam selection strategies for SD.

4.1 ROCsearch

ROCsearch is a variant of the generic beam search strategy outlined in Section 2.1. It is a level-wise search, where candidate subgroups are built by refining subgroups that are selected as the beam on the previous level. What sets ROCsearch apart from existing beam search strategies, is the way subgroups are selected to form the beam. The beam in ROCsearch will be called the *ROC-beam* from here on, and we refer by any of the terms *subgroup-ROC-point*, *ROC-point* and *point* to the point (FPR,TPR) in ROC space corresponding to a subgroup.

As hinted in Section 1, for the beam we select those subgroups that lie on the convex hull in ROC space. For our purposes, this is the shortest polyline that connects the lower left corner with the upper right corner, such that all the points lie on or below the polyline. Note that the convex hull (CH) includes the two (implicit) subgroups: the empty subgroup, and the full subgroup (with coordinates $(0, 0)$ and $(1, 1)$ in ROC space, respectively).

At each search level, a new CH is initiated with these two points, after which candidate subgroups are tentatively added to the convex hull. When a subgroup is added, one of three situations occurs. If the point falls below the CH, it is simply ignored. If the point falls exactly on the CH, the subgroup is added to the ROC-beam and the CH does not require an update, as it does not change form. If the point falls above the CH, the CH will have to be updated, and some of the existing subgroup-ROC-points may now fall below the CH. As a consequence, the subgroup(s) corresponding to these points are removed from the ROC-beam. The resulting (often fairly small) subset of the candidates, is then further processed as usual in beam search. Note that although the empty subgroup $(0, 0)$ and the entire dataset $(1, 1)$ are always part of the CH, they are obviously ignored for the ROC-beam.

Observe that the CH created by the algorithm is not required to be *strictly* convex. 'Strictly convex' in this context means that points that lie on the convex hull, but not on one of its corners, would not end up in the ROC-beam. As these subgroups are optimal as defined in Section 2.2, we decide to add such subgroups to the ROC-beam, and thus compute a non-strict convex hull. We allow multiple subgroups with the same (FPR,TPR) to be included in the beam, in compliance with the Sinn principle: such subgroups may have very distinct descriptions, and allowing just one into the ROC-beam would be arbitrary and needlessly restrictive.

Special treatment is necessary for subgroups that fall on the TPR-axis. This involves all points of the form (0,TPR), where TPR ranges from 0 to 1. For such subgroups, any refinement would not remove any FPs, and, as the subgroup can only become smaller, its TPR can only decrease. Therefore, it makes no sense to

use these subgroups as seeds for the next search level; using any convex (hence sensible) quality measure, any refinement of such a subgroup would score equal to or worse than its parents. For the purpose of constructing the CH, the point that lies highest on the TPR-axis is maintained until the end of a level, as it defines the form of the CH. Subgroups that fall on the line TPR=1 do not receive special treatment. After all, their refinements can have a severely reduced FPR; there is room for improvement by refinement.

4.1.1 Online Construction of the Convex Hull

A number of CH construction and maintenance algorithms exist, with varying complexity properties. Here, we describe a modified version of Melkman's linear-time algorithm [17] for simple polylines, which is widely regarded the best CH algorithm. It is fast, elegant, and can do online construction of the CH, and therefore fits our application well.

Where algorithms like Graham Scan [18, 19] require pre-processing to achieve linear time complexity, Melkman's algorithm does not, and it processes vertices (the ROC-points) sequentially. This properties makes it really suitable as implementation for our ROC-beam. The main SD-algorithm creates subgroups one by one, and for these subgroups the corresponding ROC-point is determined. Using Melkman's algorithm, a check is performed that decides whether the subgroup is added to the ROC-beam or not. When added, the CH is modified as needed, potentially removing subgroups from the ROC-beam in the process. By the nature of our update process, we can guarantee that the CH we construct will always remain a simple polyline, meaning it has no intersections. This is required by the Melkman algorithm, allowing its linear-time complexity. Melkman requires a lower hull, which is formed by the line segment from $(0, 0)$ to $(1, 1)$. The upper hull is formed from incoming points corresponding to new subgroups, but no subgroups having TPR $<$ FPR are ever added. This ensures that the polyline remains simple. Whenever a point is added to the upper hull, the hull as a whole is updated, 'radiating' outward from the inserted point, removing others when needed, and so avoiding the introduction of intersections. The modified version of the algorithm updates only the upper hull, and exploits knowledge of the ROC space, the orientation of the hull therein, and its extreme points.

The online property of Melkman's algorithm is strongly required for our application, since it is applied on each level of the ROCsearch algorithm for determining the convex hull of the set of candidate subgroups in ROC space. This set can grow extremely large (which is well-known as the *pattern explosion* problem). If, as opposed to online, our CH algorithm would need to post-process all candidate subgroups, the entire set needs to be stored. We can relieve ROCsearch from this unnecessary — for large datasets even prohibitive — memory burden by selecting an online CH algorithm, and therefore we choose Melkman's.

4.2 Top-k Beam Search The traditional beam search approach to SD [1] is a more straightforward implementation of the generic beam search approach as outlined in Section 2.1. Beforehand, the search width w needs to be fixed. Selected as beam for the next search level are those w subgroups considered on the current level, that have the highest value for the quality measure. Whether these w subgroups share information between them is not considered at all; the best w are simply selected.

4.3 Cover-Based Beam Selection Dedicated to trading in a bit of subgroup quality for more diversity in the result set, Van Leeuwen and Knobbe [20] introduce the task of *Non-Redundant Generalised Subgroup Discovery*. They discuss a number of techniques to reduce redundancy, using two beam selection schemes that are relevant in the current context.

The *description-based beam selection* strategy orders all candidates by descending quality, and considers them one by one until search width w is reached. For each considered subgroup, if its quality and all but one conditions are equal to any subgroup in the beam, it is discarded, otherwise it is included in the beam. We think that this strategy is directly conflicting with the Sinn principle as outlined at the beginning of Section 4. Any two conjunctions that differ in just one condition can both provide new insights, as these differing conditions need not be known to correlate to each other, or the target. No experiments have been performed using this strategy.

The other strategy, *cover-based beam selection* (CBBS), aims at reducing subgroup set redundancy by focusing on their extensions. A score based on multiplicative weighted sequential covering [2] is used to weigh the quality of each subgroup, aiming to minimize the overlap between the selected subgroups. Again, all candidates are ordered by descending quality. Now, until search width w is reached, the score for each candidate is computed using statistics of how often its records are also covered by subgroups in the beam. The candidate with the highest score, or the least amount of overlap, is added to the beam, and the selection procedure starts anew.

Despite the different motivations behind CBBS (redundancy reduction) and our approach (simplifying

the application of SD for non-experts and gaining efficiency), we have included CBBS in our experiments.

5 Experiments

To evaluate ROCsearch, we compare the performance of four beam selection strategies for SD. The first is *Wide Beam*, a regular top-k strategy, with a fixed search width of 100, which is generally considered to lead to a generous exploration of the search space. The second is ROCsearch, the subject of this paper, which automatically selects appropriate search widths. The third is *Narrow Beam*, a top-k strategy that matches the search widths from the ROCsearch experiments. The fourth is *CBBS*, the cover-based beam selection strategy, with again a fixed search width of 100. All discussed search strategies have been implemented in the generic Subgroup Discovery package Cortana, available from `http://datamining.liacs.nl/cortana.html`.

Because of the strongly differing goals outlined in Section 2.1.1, it is nontrivial to define an evaluation criterion that does justice to the results of both SD and clustering. Therefore it would be unfair to compare SD algorithms with clustering algorithms. Hence, we compare our new ROCsearch algorithm merely against other beam selection strategies: Wide Beam and Narrow Beam stem from [1], and CBBS is taken from [20].

The performance of the beam selection strategies is gauged on eight datasets, all taken from the UCI repository [21]. This sample features a mixture of datasets having only binary/nominal or only numeric attributes, as well as datasets mixing attribute types. For datasets having a binary target, the 'positive' value is used as target value. For the nominal target of the Covertype data, the majority class "Lodgepole Pine" is used. The YPMSD (full name: YearPredictionMSD) dataset has a numeric target corresponding to the year of recording of a song. We binarize this target by putting a threshold at the turn of the millennium; records with 'Year $\geq$ 2000' get the positive label, and all other records get the negative label. Characteristics of the used datasets, including statistics on the available attributes per type, can be found in Table 1.

We record several statistics for each beam selection strategy on each dataset, and report them per search level. Notice that beam selection strategies only start to have an effect beyond search level 1. To observe selection strategy behavior, we report $\left|\mathcal{B}_p^n\right|$, the size of the beam when moving from level p to level n (i.e., the search width: the number of seeds to be refined at the next level). To gauge the efficiency of the selection strategies, we observe $|\mathcal{C}|$, the number of candidates considered during the search, and #s, the number of seconds needed to complete the search.

Table 1: Dataset characteristics.

Dataset	N	m	bin	nom	num
Ω_1 = Adult	48842	14	1	7	6
Ω_2 = Covertype	581012	54	44	0	10
Ω_3 = Credit-a	690	15	3	6	6
Ω_4 = Ionosphere	351	34	2	0	32
Ω_5 = Mushroom	8124	22	6	16	0
Ω_6 = Tic-tac-toe	958	9	0	9	0
Ω_7 = Wisconsin	699	9	0	0	9
Ω_8 = YPMSD	515345	90	0	0	90

Finally, to assess the quality of the selection strategies, we observe AUC, the area under the ROC convex hull of the resulting subgroup set. Since SD typically has a convex quality measure, and hence strives to find subgroups with as many positives and as few negatives as possible, in effect it tries to reach the ROC heaven of $(0, 1)$. The AUC measures how close to this ideal a resulting subgroup set has managed to come, incorporating contributions of all subgroups that are optimal as defined in Section 2.2. Since all beam search strategies for Subgroup Discovery aim for a maximal AUC, we think evaluating performance of the strategies by the AUC is fair.

The resulting statistics of the beam selection strategies on the datasets can be found in Table 2. Some entries for CBBS are blank; experiments were aborted after a day had passed without algorithm termination.

The rationale behind considering both the Wide and Narrow Beam strategies, is that the Wide Beam is more akin to a setting one would see in real life. However, inspecting the results, we noticed that ROCsearch typically selects a much more modest search width than is used for Wide Beam. To explore whether results of a similar quality to ROCsearch could also be attained by a top-k beam selection strategy, we feed the top-k algorithm an amount of to-be-refined seeds comparable to the ROCsearch algorithm. Hence, the Narrow Beam strategy is defined, taking for search level d a search width of $\max_{i\in\{1,\ldots,d-1\}} \left|\mathcal{B}_i^{i+1}\right|$, where the $\mathcal{B}$-values are taken from the ROCsearch experiment. Note that this might actually favor Narrow Beam a bit, as it might receive a slightly larger number of input seeds at a lower level than ROCsearch.

5.1 Discussion Let us focus first on the relative performance of Wide Beam and ROCsearch, as detailed in the first two rows for each dataset in Table 2. The search width for Wide Beam was manually set to 100 beforehand, which is a typical setting when no infor-

Table 2: Candidate set size, used time in seconds, AUC, and search width for the beam search algorithms

Ω	Algorithm		**Search level 2**				**Search level 3**				**Search level 4**			
		$\lvert\mathcal{B}_1^2\rvert$	$\lvert\mathcal{C}\rvert$	#s	AUC	$\lvert\mathcal{B}_2^3\rvert$	$\lvert\mathcal{C}\rvert$	#s	AUC	$\lvert\mathcal{B}_3^4\rvert$	$\lvert\mathcal{C}\rvert$	#s	AUC	$\lvert\mathcal{B}_4^5\rvert$
Ω_1	Wide Beam	100	13858	4	0.8417758	100	27723	9	0.8468678	100	41158	13	0.8480668	100
	ROCsearch	6	962	<1	0.8395724	8	2057	1	0.8461147	15	4006	1	0.8474961	18
	Narrow Beam	6	995	<1	0.8400000	8	2417	1	0.8457916	15	6409	2	0.8463876	18
	CBBS	100	13848	7734	0.7981340	100	27879	23906	0.8076945	100	41679	39889	0.8074181	100
Ω_2	Wide Beam	100	22368	81	0.7920484	100	44432	159	0.7976004	100	66212	234	0.7990936	100
	ROCsearch	12	2893	14	0.7897330	37	11013	55	**0.7979165**	92	31403	157	**0.8039704**	177
	Narrow Beam	12	2854	11	0.7917895	37	16589	59	0.7972163	92	60928	216	0.7990936	177
	CBBS	100	-	-	-	100	-	-	-	100	-	-	-	100
Ω_3	Wide Beam	100	10719	1	0.9146971	100	22146	1	0.9159643	100	33455	1	0.9161429	100
	ROCsearch	3	464	<1	0.9142038	5	1036	<1	0.9154455	12	2405	<1	0.9156241	14
	Narrow Beam	3	471	<1	0.9142038	5	1268	<1	0.9153137	12	4241	<1	0.9154158	14
	CBBS	100	10729	168	0.8784753	100	22250	428	0.8787347	100	33717	683	0.8787347	100
Ω_4	Wide Beam	100	44704	1	0.9312522	100	89560	3	0.9574780	100	134492	4	0.9625573	100
	ROCsearch	13	2242	<1	0.9194004	13	8086	1	0.9500705	9	12132	1	0.9625397	5
	Narrow Beam	13	6224	<1	0.9165608	13	12064	1	0.9361552	13	17898	1	0.9389242	13
	CBBS	100	44994	909	0.9174603	100	89926	1880	0.9354850	100	134894	2845	0.9373545	100
Ω_5	Wide Beam	100	6107	<1	*0.9201521*	100	13875	1	*0.9205715*	100	21543	2	*0.9205715*	100
	ROCsearch	2	317	<1	*0.9201521*	4	666	<1	*0.9205715*	8	1267	<1	*0.9205715*	3
	Narrow Beam	2	310	<1	*0.9201521*	4	836	<1	*0.9205715*	8	2142	<1	*0.9205715*	8
	CBBS	100	6107	441	0.9163071	100	14307	1468	0.9163071	100	21996	2464	0.9163071	100
Ω_6	Wide Beam	100	702	<1	*0.6624774*	100	3002	<1	*0.6736980*	100	5102	<1	*0.6736980*	100
	ROCsearch	1	52	<1	*0.6624774*	8	236	<1	*0.6736980*	16	572	<1	*0.6736980*	168
	Narrow Beam	1	52	<1	*0.6624774*	8	411	<1	*0.6736980*	16	1131	<1	*0.6736980*	168
	CBBS	100	702	8	*0.6624774*	100	3002	19	*0.6736980*	100	5102	24	*0.6736980*	100
Ω_7	Wide Beam	100	4610	<1	0.9834478	100	14920	1	0.9846165	100	25352	1	0.9848702	100
	ROCsearch	5	540	<1	0.9795883	7	1254	<1	0.9810016	9	2182	<1	0.9813052	12
	Narrow Beam	5	600	<1	0.9733190	7	1526	<1	0.9740528	9	2878	<1	0.9746236	12
	CBBS	100	4610	150	0.9635299	100	14126	463	0.9616862	100	23592	764	0.9647711	100
Ω_8	Wide Beam	100	127260	826	0.6414609	100	253260	1587	0.6431013	100	379260	2447	0.6436127	100
	ROCsearch	7	10080	85	0.6412821	16	30240	178	**0.6431076**	22	57960	259	**0.6440413**	22
	Narrow Beam	7	10080	81	0.6412433	16	41580	264	0.6429721	22	84420	522	0.6434536	22
	CBBS	100	-	-	-	100	-	-	-	100	-	-	-	100

mation is available on dataset peculiarities; setting w to 100 usually allows for a generous exploration of the search space, during which most of the underlying concepts in a dataset that one reasonably could expect to capture, should be considered.

Typically, the autoconfiguring ROC-beam chooses a substantially lower search width. Averaging over the eight datasets, the width of the ROC-beam is 6.125 from level 1 to 2, 12.25 from level 2 to 3, 22.875 from level 3 to 4, and 52.375 from level 4 to 5. From the $|\mathcal{C}|$ and #s columns, we see that this translates to considering a number of candidates that is an order of magnitude smaller, and a substantially shorter runtime (though this is not always measurable, since the Wide Beam algorithm can be fairly quick on small datasets as well). While being an order of magnitude more efficient, the ROCsearch strategy manages to maintain the quality of the resulting subgroup set: the AUC is usually only slightly decreased, and occasionally exactly equal. *For the two largest datasets, on the third and fourth search level, AUC is even* increased *for* ROCsearch *when compared to Wide Beam* (bold values in Table 2).

An interesting observation is that typically, the autoconfiguring ROC-beam gets wider for higher search levels. However, this relation is not always monotone: for both the Ionosphere (Ω_4) and the Mushroom (Ω_5) datasets, the ROC-beam width decreases around search level 4. Apparently, the ROC-beam is closing in on capturing the target at hand so well, that fewer Subgroups are needed to express the underlying concepts.

When feeding back the search width w, learned in the ROCsearch experiment, into a regular top-k strategy (the Narrow Beam), we observe that the efficiency performance between these two strategies (as measured in number of candidates and seconds) is equivalent. In terms of AUC performance, ROCsearch outperforms Narrow Beam fifteen times, they tie seven times, and Narrow Beam outperforms ROCsearch two times. On closer inspection, we see that on the two datasets featuring no numeric attributes (Ω_5 and Ω_6), AUC is exactly equal for Wide Beam, ROCsearch, and Narrow Beam. As highlighted in Section 2.1, the beam search strategy is particularly prudent when numeric attributes are available. The other three cases where ROCsearch does not outperform Narrow Beam all occur at search level 2 (for Ω_1, Ω_2, and Ω_3). Deeper searches on those same datasets all have ROCsearch outperforming Narrow Beam. Hence, when the search is not too shallow, and when mining on datasets featuring numeric attributes, ROCsearch outperforms Narrow Beam, while being equally efficient. Also, the Narrow Beam width was extracted from ROCsearch experiments; it remains nontrivial to come up with this setting out of the blue.

Finally, we observe that CBBS did not fare well in comparison with any of the other strategies. It was only marginally better than a Narrow Beam experiment once, but in all other cases the AUC was worse, sometimes by a few percent. Furthermore, the runtime of the experiments was also extremely bad in comparison to the other settings. In its defence, it should be noted that CBBS has redundancy reduction as its main ambition rather than search space reduction or performance; it is not designed to perform well here.

6 Conclusions

In order to benefit from the full expressive power available in the general Subgroup Discovery (SD) framework, native treatment of numeric attributes should be available, and any quality measure should be allowed that is convex in ROC space. This precludes exhaustive algorithms; usually a beam search strategy is selected instead. Setting the involved search width parameter is typically nontrivial but very influential: setting the width too low will lead to performance loss, while setting the width too high will lead to inefficiency. A correct setting depends on intricate properties of the dataset at hand, and is usually a matter of trial and error even for data mining experts. It is unrealistic to expect a domain expert to come up with a proper setting. But setting the parameter suboptimally will punish the domain expert with either bad performance or inefficiency.

We introduce ROCsearch, a beam search variant for Subgroup Discovery that configures its search width automatically, tuning the parameter to the peculiarities of the dataset at hand. Breaking into the traditional beam selection strategy, ROCsearch selects those subgroups that lie on the convex hull in ROC space as the beam for the next level. These subgroups do not necessarily maximize the SD quality measure, but they are optimal in another sense: each subgroup on the convex hull has minimal cost for a cost assignment to false positives and false negatives. Moreover, the number of subgroups on the convex hull is automatically determined, hence does not have to be set beforehand.

Experiments on eight diverse UCI datasets show that ROCsearch is an order of magnitude more efficient than the typical Wide Beam strategy, which is configured to explore the search space generously. Despite considering substantially fewer candidate subgroups, we find that ROCsearch finds subgroup sets of equivalent quality to the subgroup sets found with Wide Beam. On the largest two datasets and the two deepest considered search depths, ROCsearch even finds better subgroup sets.

When configuring the fixed-width beam selection strategy with the same widths as used by ROCSEARCH, we find that this Narrow Beam strategy is equivalently efficient to ROCSEARCH. For datasets with only nominal attributes, Narrow Beam performs equivalently to both ROCSEARCH and Wide Beam. For datasets involving numeric attributes, Narrow Beam can occasionally outperform ROCSEARCH on a shallow search depth. However, when the search is not too shallow, and numeric attributes are available in the dataset, ROCSEARCH always outperforms Narrow Beam while being equally efficient. Also, choosing a proper beam width for Narrow Beam without running ROCSEARCH first, remains a nontrivial problem.

The fourth considered beam selection strategy, Cover-Based Beam Selection, compared badly to the other three strategies. Its performance is typically worse, and the runtime efficiency was significantly worse. Since CBBS has redundancy reduction as main goal rather than performance or efficiency, this was to be expected.

All datasets are taken from the UCI repository, with two of them containing more than half a million records. Hence, these datasets are approaching the magnitude of real-life datasets. Promisingly, these are the two datasets on which ROCSEARCH even manages to outperform the generous Wide Beam selection strategy. We expect ROCSEARCH to perform particularly well in real-life settings.

In future work, we would like to investigate whether results can be improved by softening the demands of ROCSEARCH. Currently, only those subgroups that lie on the ROC convex hull are selected for the beam. Subgroups that fall way below this hull, and subgroups that lie close to it, are both unceremoniously discarded. Although we have seen that this radical choice leads to good performance, we may consider treating subgroups that lie close enough to the convex hull differently.

References

[1] W. KLÖSGEN, *Subgroup Discovery*, in: W. Klösgen, J. M. Zytkow (eds.), Handbook of Data Mining and Knowledge Discovery, pp. 354–361, Oxford University Press, Oxford, 2002.

[2] N. LAVRAČ, B. KAVŠEK, P. FLACH, L. TODOROVSKI, *Subgroup Discovery with CN2-SD*, Journal of Machine Learning Research 5, pp. 153–188, 2004.

[3] J. P. EGAN, *Signal Detection Theory and ROC Analysis*, Series in Cognition and Perception, Academic Press, New York, 1975.

[4] K. A. SPACKMAN, *Signal Detection Theory: Valuable Tools for Evaluating Inductive Learning*, Proc. International Workshop on Machine Learning, pp. 160–163, 1989.

[5] C.B. BARBER, D.P. DOBKIN, H. HUHDANPAA, *The quickhull algorithm for convex hulls*, ACM Transactions on Mathematical Software 22 (4), pp. 469–483, 1996.

[6] H. MANNILA, H. TOIVONEN, *Levelwise search and borders of theories in knowledge discovery*, Data Mining and Knowledge Discovery 1 (3), pp. 241–258, 1997.

[7] J. FÜRNKRANZ, P. A. FLACH, *ROC 'n' Rule Learning — Towards a Better Understanding of Covering Algorithms*, Machine Learning 58 (1), pp. 39–77, 2005.

[8] M. ATZMUELLER, F. PUPPE, *SD-Map – A Fast Algorithm for Exhaustive Subgroup Discovery*, Proc. PKDD, pp. 6–17, 2006.

[9] H. GROSSKREUTZ, S. RÜPING, *On Subgroup Discovery in Numerical Domains*, Data Mining and Knowledge Discovery 19 (2), pp. 210–226, 2009.

[10] F. HERRERA, C. J. CARMONA, P. GONZÁLEZ, M. J. DEL JESUS, *An Overview on Subgroup Discovery: Foundations and Applications*, Knowledge and Information Systems 29 (3), pp. 495–525, 2011.

[11] P. KRALJ NOVAK, N. LAVRAČ, G. I. WEBB, *Supervised Descriptive Rule Discovery: a Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining*, Journal of Machine Learning Research 10, pp. 377–403, 2009.

[12] A. P. BRADLEY, *The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms*, Pattern Recognition 30, pp. 1145–1159, 1997.

[13] F. J. PROVOST, T. FAWCETT, *Robust Classification for Imprecise Environments*, Machine Learning 42 (3), pp. 203–231, 2001.

[14] D. J. HAND, *Measuring Classifier Performance: a Coherent Alternative to the Area Under the ROC Curve*, Machine Learning 77 (1), pp. 103–123, 2009.

[15] J. HERNÁNDEZ-ORALLO, P. FLACH, C. FERRI, *ROC Curves in Cost Space*, Machine Learning 93 (1), pp. 71–91, 2013.

[16] G. FREGE, *Über Sinn und Bedeutung*, Zeitschrift für Philosophie und philosophische Kritik (100), pp. 25–50, 1892.

[17] A. A. MELKMAN, *On-line Construction of the Convex Hull of a Simple Polyline*, Information Processing Letters 25, pp. 11–12, 1987.

[18] R. L. GRAHAM, *An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set.*, Information Processing Letters 1, pp. 132–133, 1972.

[19] A. M. ANDREW, *Another Efficient Algorithm for Convex Hulls in Two Dimensions*, Information Processing Letters 9, pp. 216–219, 1979.

[20] M. VAN LEEUWEN, A. KNOBBE, *Non-Redundant Subgroup Discovery in Large and Complex Data*, Proc. ECML/PKDD, 2011.

[21] K. BACHE, M. LICHMAN, *UCI Machine Learning Repository*, 2013.