ROCsearch in a Wider Context — A ROC-Guided Search Strategy for Subgroup Discovery and Beyond

Wouter Duivesteijn[1], Marvin Meeng[2], and Arno Knobbe[2]

[1] Fakultät Informatik, LS VIII, Technische Universität Dortmund, Germany*
wouter.duivesteijn@tu-dortmund.de
[2] LIACS, Leiden University, the Netherlands
{m.meeng, a.j.knobbe}@liacs.leidenuniv.nl

Abstract. ROCsearch is a ROC-based beam search variant, initially developed for Subgroup Discovery (SD). In ordinary beam search, on each search level, a fixed number of best-scoring candidates are selected to generate candidates for the next search level. This fixed number, the beam width, is typically hard to set, and its setting strongly influences the outcome of the mining process. In ROCsearch, however, on each search level, the beam width is set dynamically by analyzing the intermediate results in ROC space. Thus, setting the beam width parameter 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. In Subgroup Discovery, ROCsearch has been shown to be 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. However, ROCsearch has not been investigated beyond SD, while it should be readily applicable in machine learning and data mining outside of the SD subfield. As work in progress, we propose this wider outlook for ROCsearch.

1 Introduction

Beam search [1] 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 candidates are considered, of which the most promising w (for *search width*) are selected as the *beam*. On the second level, we only consider candidates that are specializations of the candidates in the beam, and again, the w most promising candidates 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. The beam search strategy finds middle

* This research is supported in part by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project C1.

ground between a completely greedy method and exhaustive search; on the one hand, selecting only the w most promising candidates 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 a beam search algorithm 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 too many uninteresting results. Unfortunately, in practice, the search width is nontrivial to set. A proper value strongly depends on many properties of the dataset (column types and cardinalities, the number of columns and rows, the number of underlying concepts present in the data) and the data mining problem at hand. Recently, we introduced the new beam search variant ROCsearch [2] in the context of Subgroup Discovery (SD) [3, 4]. ROCsearch marries concepts from beam search and ROC analysis [5, 6], 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.

ROCsearch automatically adjusts 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 heuristic data mining substantially more user-friendly. Moreover, on the Subgroup Discovery task, 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. With this work in progress report, we hope to invite discussion on where else ROCsearch can be employed.

2 Preliminaries: ROC Analysis

Throughout this work we assume a dataset in a supervised setting; one attribute of the data is the designated *target*, which 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. One particular value for the target is singled out as the desired value: records having that particular value are the *positives*, and the other records are the *negatives*.

ROC analysis has a rich history in the field of signal processing [5]. 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 [6]. In its most traditional form, ROC analysis concerns a learning problem with a binary response variable. Results of a learning task are interpreted 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 [7, 8].

The ROC methodology analyzes results in *ROC space*: the two-dimensional unit square with FPR on the horizontal axis and TPR on the vertical axis. The *perfect result*, containing all the positives and no negatives, is found in the top left corner; its coordinates in ROC space are (0,1). The *empty result* is found in the bottom left corner with coordinates (0,0), and the *full result* 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 result: false positives and false negatives.

Definition 1 (Optimal Result). *Suppose a result set $\mathcal{R}$, and a cost assignment to false positives and false negatives. A result $R \in \mathcal{R}$ is called* optimal *within $\mathcal{R}$ for this cost assignment, if its total cost equals the minimum total cost for any result in $\mathcal{R}$.*

Hence, a result R is optimal within a result set $\mathcal{R}$ for *some* cost assignment if and only if it lies on the convex hull of $\mathcal{R}$ in ROC space [9].

3 The ROCsearch Algorithm

ROCsearch is a level-wise search, where candidates are built by refining candidates that are selected as the beam on the previous level. What sets ROCsearch apart from existing beam search strategies, is the way candidates 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 *ROC-point* and *point* to the point (FPR,TPR) in ROC space corresponding to a candidate.

As hinted in Section 1, for the beam we select those candidates 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) results: the empty result, and the full result (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 candidates are tentatively added to the convex hull. When a candidate 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 candidate 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 ROC-points may now fall below the CH. As a consequence, the candidate(-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 result $(0, 0)$ and the entire result $(1, 1)$ 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 results are optimal as defined in Section 2, we decide to add such candidates to the ROC-beam, and thus compute a non-strict convex hull.

3.1 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 [10] for simple polylines, which is widely regarded the best CH algorithm. It is fast, elegant, and constructs the CH in an online manner. This online property 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 candidates in ROC space. Depending on the learning task at hand, this set can grow extremely large. 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 memory burden by selecting an online CH algorithm, and therefore we choose Melkman's.

Creating candidates one by one, we can determine the corresponding ROC-point. Using Melkman's algorithm, a check is performed that decides whether the candidate is added to the ROC-beam or not. When added, the CH is modified as needed, potentially removing candidates 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 candidates, but no candidates 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.

3.2 Performance

Compared to the traditional beam search setting, ROCsearch has proven beneficial in experiments with the Subgroup Discovery task on eight UCI datasets [2, Section 5]: apart from autoconfiguring a hard-to-set parameter, the subgroup sets found with ROCsearch are of equivalent quality as those found with traditional beam search, but the computational expense of ROCsearch is an order of magnitude lower.

4 Related Local Pattern Mining Methods

In the Local Pattern Mining subfield of data mining, using a convex hull or pareto front as a selection mechanism to determine a set of relevant patterns is a recurring idea. In predictive rule learning, the idea of reducing a set of candidate rules by only considering those on the convex hull was explored in the ROCCER algorithm [11]; ROCsearch can be seen as an exploration of the ROCCER concept as a driving mechanism for search instead of merely a selection mechanism. In skypattern mining [12], a set of measures on which a pattern is to be evaluated is input to the problem. The goal then becomes finding patterns that are not dominated by others on the full set of measures. This idea was instantiated by Van Leeuwen and Ukkonen [13] in work devoted to finding good subgroup *sets* that trade off quality and diversity; algorithms were designed to find a skyline of subgroup sets in this two-dimensional evaluation space. Neither of these works explicitly use the skyline/convex hull/pareto front as a leading device for the search process. Mampaey et al. [14] do consider only those subgroup refinements that lie on the convex hull in ROC space for advancement of the search process, but this work requires the employed quality measure to be convex, while ROCsearch does not impose such restrictions.

5 Beyond Local Pattern Mining

ROC analysis is well understood in Local Pattern Mining, which is why the benefit of ROCsearch for Subgroup Discovery is immediately intuitively clear. However, we think that the applicability of the ROC convex hull/skylines/pareto fronts as a device to bound a heuristic search process is not necessarily limited to the subfield of Local Pattern Mining.

Since ROC analysis is well understood in classification as well, this subfield of machine learning could also benefit from ROC-guided search. A similar strategy has been explored in the SAYU algorithm [15]. In this work, classification performance is improved by combining ILP rule induction with Bayesian network learning. Instead of the traditional two-step process, where classification rules are learned first and an encompassing classifier is learned afterwards, SAYU interleaves the two steps: the encompassing classifier is constructed while rules are still being learned. Throughout the process, a Bayesian network structure is maintained that holds the encompassing classifier model; experiments are performed with the Naive Bayes and Tree Augmented Naive Bayes structures. Candidate rules are evaluated based on by how much they improve the classifier, which is gauged in precision-recall space. Since this space is equivalent to ROC space, a ROC-guided heuristic search is at work here.

Another field where the ideas for ROCsearch can provide inspiration is that of *Feature Subset Selection* [16]. This task deals with selecting from the initial set of features a relatively small subset for which subsequent modeling is both more feasible computationally and possibly more accurate. One of the obvious approaches to FSS, especially when subsequent modeling over the feature subset

comes in the form of building classifiers, is known as a *wrapper approach*. In such an approach, candidate feature subsets are tested by means of a classifier, and the search for optimal subsets is guided by how well the classifier performs on the candidate. Now, when no prior assignment of misclassification costs is given, it makes sense to judge subsets/classifiers on the basis of their FPR/TPR. Hence, each candidate subset is associated with a point in ROC-space. A naive implementation of this approach would consider all 2^n subsets of the total n features, and finally report the few feature subsets that form the convex hull in ROC-space, or possibly all subsets of at most $k < n$ features. Clearly, this would be unattractive from a computational perspective. One could argue, however, that for any of the resulting feature sets, the subset is composed of sub-subsets that are also fairly good, and therefore close to the final convex hull. A promising algorithm would therefore generate all subsets of $k = 1$ features (in other words, n individual features), and determine the convex hull of these n points. Then, only the subsets on the CH will be tentatively extended with any of the features, to form 2-subsets. The algorithm would thus proceed until k-subsets, essentially mimicking the ROCsearch process for SD, where the role of the subgroups is now taken by the feature sets. The algorithm would report the few subsets on the CH, and it is hoped that these subsets are very similar to those produced by the naive, exhaustive approach.

In the tasks discussed so far, the evaluation space can be naturally decomposed into the two dimensions that make up ROC space. For other tasks, this space may be higher-dimensional; consider for instance classification with a nominal target having $v > 2$ distinct values, which cannot be handled naturally with a one-versus-all strategy — classifiers for such multi-way nominal targets can be analyzed in $v(v-1)$-dimensional ROC space [17]. A convex-hull-driven search strategy such as ROCsearch cannot be directly ported to such a setting without loss of performance (in terms of either result quality or runtime), since a higher-dimensional convex hull is computationally expensive. ROCsearch can however be made applicable on higher-dimensional output spaces by using random projection techniques. Both volume-preserving random projections [18] and high-dimensional randomly-projected simplices [19] have been studied, and the derived bounds may teach us what can be expected when a heuristic search with a high-dimensional output space is guided by a convex hull strategy on a randomly projected subspace. A similar random projection approach has been developed for one-class classification [20], though no convex hull guided search was involved; exploring this strategy should be beneficial for Subgroup Discovery with multi-dimensional responses [21], or multi-label classification [22]. One could also imagine improving on the traditional hyperparameter grid search for Support Vector Machines [23], by combining random projections with convex-hull-driven search.

6 Conclusions

Previously, we have introduced 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. Apart from the beneficial autoconfiguration, ROCsearch has obtained results with equivalent quality an order of magnitude faster when compared to traditional beam search on the Subgroup Discovery task.

Since the concepts underlying ROCsearch should be applicable more widely than just on Subgroup Discovery, we discuss the algorithm and its potential further deployment in this paper. A related strategy has been employed in a rule-based naive Bayes classification setting, and application on other simple classifiers should be straightforward. We propose a Feature Subset Selection scheme that uses a search process driven by ROC analysis, just like ROCsearch. When the evaluation space of a learning task is higher-than-two-dimensional, convex-hull-driven search strategies can be applied by employing random projections, thus counteracting the otherwise prohibitive computational cost of computing the high-dimensional convex hull. We think that such a strategy has the potential to improve the focus of any learning task where currently some form of heuristic search is employed; the automatic parameter tuning provided by a convex-hull-driven strategy begs to be exploited throughout our field.

References

1. R. Bisiani, *Beam Search*, in: S. Shapiro (ed.), Encyclopedia of Artificial Intelligence, John Wiley and Sons, pp. 56–58, 1987.
2. M. Meeng, W. Duivesteijn, A. Knobbe, *ROCsearch — An ROC-guided Search Strategy for Subgroup Discovery*, Proc SDM, pp. 704–712, 2014.
3. 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.
4. N. Lavrač, B. Kavšek, P. Flach, L. Todorovski, *Subgroup Discovery with CN2-SD*, Journal of Machine Learning Research 5, pp. 153–188, 2004.
5. J. P. Egan, *Signal Detection Theory and ROC Analysis*, Series in Cognition and Perception, Academic Press, New York, 1975.
6. K. A. Spackman, *Signal Detection Theory: Valuable Tools for Evaluating Inductive Learning*, Proc. International Workshop on Machine Learning, pp. 160–163, 1989.
7. 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.
8. F. J. Provost, T. Fawcett, *Robust Classification for Imprecise Environments*, Machine Learning 42 (3), pp. 203–231, 2001.

9. 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.
10. A. A. Melkman, *On-line Construction of the Convex Hull of a Simple Polyline*, Information Processing Letters 25, pp. 11–12, 1987.
11. R. C. Prati, P. A. Flach, ROCCER: An Algorithm for Rule Learning Based on ROC Analysis, Proc. IJCAI, pp. 823–828, 2005.
12. A. Soulet, C. Raïssi, M. Plantevit, B. Crémilleux, *Mining Dominant Patterns in the Sky*, Proc. ICDM, pp. 655–664, 2011.
13. M. van Leeuwen, A. Ukkonen, *Discovering Skylines of Subgroup Sets*, Proc. ECML/PKDD (3), pp. 272–287, 2013.
14. M. Mampaey, S. Nijssen, A. Feelders, A. J. Knobbe, *Efficient Algorithms for Finding Richer Subgroup Descriptions in Numeric and Nominal Data*, Proc. ICDM, pp. 499–508, 2012.
15. J. Davis, E. S. Burnside, I. de Castro Dutra, D. Page, V. Santos Costa, *An Integrated Approach to Learning Bayesian Networks of Rules*, Proc. ECML, pp. 84–95, 2005.
16. I. Guyon, A. Elisseeff, *An Introduction to Variable and Feature Selection*, Journal of Machine Learning Research 3, pp. 1157–1182, 2003.
17. A. Srinivasan, *Note on the Location of Optimal Classifiers in n-dimensional ROC-space*, technical report PRG-TR-2-99, Oxford University Computing Laboratory, 1999.
18. A. Magen, A. Zouzias, *Near Optimal Dimensionality Reductions That Preserve Volumes*, APPROX-RANDOM, pp. 523–534, 2008.
19. D. L. Donoho, J. Tanner, *Neighborliness of Randomly Projected Simplices in High Dimensions*, PNAS 102 (27), pp. 9452–9457, 2005.
20. P. Casale, O. Pujol, P. Radeva, *Approximate Convex Hulls Family for One-Class Classification*, MCS, pp. 106–115, 2011.
21. L. Umek, B. Zupan, *Subgroup Discovery in Data Sets with Multi-Dimensional Reponses*, Proc. IDA, pp. 533–549, 2011.
22. G. Tsoumakas, I. Katakis, I. P. Vlahavas, *Mining Multi-Label Data*, Data Mining and Knowledge Discovery Handbook, Springer, pp. 667–685, 2010.
23. C.-C. Chang, C.-J. Lin, *LIBSVM: a Library for Support Vector Machines*, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`