Lavesson, Niklas; Davidsson, Paul.
"Generic Methods for Multi-criteria Evaluation"
SIAM International Conference on Data Mining 2008, Atlanta, Georgia, USA

Generic Methods for Multi-criteria Evaluation

Niklas Lavesson * Paul Davidsson †

Abstract

When evaluating data mining algorithms that are applied to solve real-world problems there are often several, conflicting criteria that need to be considered. We investigate the concept of generic multi-criteria (MC) classifier and algorithm evaluation and perform a comparison of existing methods. This comparison makes explicit some of the important characteristics of MC analysis and focuses on finding out which method is most suitable for further development. Generic MC methods can be described as frameworks for combining evaluation metrics and are generic in the sense that the metrics used are not dictated by the method; the choice of metric is instead dependent on the problem at hand. We discuss some scenarios that benefit from the application of generic MC methods and synthesize what we believe are attractive properties from the reviewed methods into a new method called the candidate evaluation function (CEF). Finally, we present a case study in which we apply CEF to trade-off several criteria when solving a real-world problem.

1 Introduction.

We consider the problem for which a learning algorithm, given a set of training instances, generates a classifier that can be used to predict the class of new instances. The most frequently used criteria by which to assess classifier performance, and indirectly the performance of the applied learning algorithm, is classification accuracy. The most common metric for accuracy is the ratio between correctly classified instances and the total number of classified instances. However, serious concerns have been raised against the use of this metric as the only estimate for generalization performance [7]. Additionally, when applying a learning algorithm to solve a real-world problem there are often other important factors that need to be considered. For instance, it might be crucial for a particular application that the training time and classification time of the employed algorithm are lower than some desired thresholds. For another application it might be essential that the classifier representation is easily interpreted by humans. When selecting algorithms for real-world applications, one usually has to be aware of, and properly balance, trade-offs between different criteria. One of the most common practices today is to focus on just one criterion, i.e., using only one evaluation metric (e.g., accuracy) or to design a custom multi-criteria evaluation scheme for the particular application. Some of the drawbacks of the first method have already been presented and, regarding the second method, we argue that a customizable, generic multi-criteria method would be more effective and suitable for most applications since it simplifies comparison and puts the focus on which criteria to assess for a particular application, rather than on how to combine and trade-off criteria.

Next we will explain the concept of multi-criteria evaluation. In Section 2 we then present three existing generic multi-criteria methods. An analysis and comparison of these methods is carried out in Section 3. Building on the most useful parts of each method we then present a new generic multi-criteria method in Section 4. In the next section we present an interesting case study in which we apply the new method. We finish with conclusions and pointers toward future work.

1.1 Multi-Criteria Evaluation.

It is argued that some performance metrics are more appropriate than others for different domains [3]. This would imply that, analogous to the no-free-lunch theorems for supervised learning [9], which stipulate that no algorithm is superior on all problems, there is no single metric that is always superior for performance assessment. One conclusion to draw from this is that one has to investigate which *criteria* are relevant for a particular problem, and which *metrics* should be used to assess these criteria. We argue that it is important to evaluate on the basis of all criteria that are relevant for a particular problem. This should be accomplished by combining criteria, in a systematic way, into one measure instead of comparing different criteria in an ad-hoc manner.

We have distinguished between three types of candidates for evaluation: algorithms, algorithm configurations, and classifiers [5]. Algorithm evaluation refers to the overall evaluation of an algorithm, covering potentially all, or a selection, of its parameter settings, for

*Department of Software and Systems Engineering, Blekinge Institute of Technology, Box 520, SE-372 25 Ronneby, Niklas.Lavesson@bth.se.

†Department of Software and Systems Engineering, Blekinge Institute of Technology, Box 520, SE-372 25 Ronneby, Paul.Davidsson@bth.se.

a particular problem. An algorithm configuration evaluation, on the other hand, refers to the evaluation of one particular parameter setting for an algorithm. A classifier evaluation is an evaluation of one classifier on a specific data set.

A criterion can usually be assessed using a variety of metrics, e.g., the complexity of an induced tree can be assessed using a metric that simply counts the nodes, but a more generic complexity metric might be based on the number of bits needed to represent the classifier. In this study, we use the term multi-criteria (MC) evaluation for multi-metric evaluation. However, it should be pointed out that even though an MC evaluation involves a number of metrics, each addressing different criteria, some metrics could in fact address the same criterion. Besides selecting which criteria and metrics to use, it is also important to combine them in a suitable way and to balance trade-offs between them according to the application at hand. As we will see later on, there is occasionally conflicts between different criteria and one has to select an algorithm that balances the trade-offs in an acceptable manner. For instance, the most accurate classifier might take too long to generate.

Thus, there are a number of motivations for using an MC approach instead of the traditional single-criterion (SC) approach. However, MC evaluation is more complex than SC evaluation due to the computation and integration of several metrics. Additionally, it is often hard to compare different MC solutions since they might be hard coded for a particular application or described using different and sometimes ambiguous terminology. We therefore believe it is important to work toward a standardized and generic MC method for evaluation.

2 Generic Multi-Criteria Methods.

Generic MC methods can be described as frameworks which contain rules for how to combine, weigh or in other ways balance multiple evaluation metrics. They are generic in the sense that arbitrary metrics can be integrated. Thus, the actual metrics are not predetermined, i.e., dictated by the design of the method. The choice of metrics could then be based on the problem at hand. We will soon review three existing generic MC methods. However, since many studies use different terminology, we will first discuss some of the fundamental elements and concepts of MC evaluation.

The output of an MC evaluation method can be either single- or multi-valued. A method based on single-valued output converts the multiple results into a single value by adding, dividing or multiplying the metric values. A multi-valued result, on the other hand, is often visualized with a graph [4]. However, we will focus only on methods with single-valued output.

The included metrics can be used to assess algorithm-level qualities like training time and memory consumption, or classifier-level qualities like complexity and classification accuracy. In order to simplify discussion, we define metrics as functions with a particular range (e.g., 0 to 1) and type (e.g., discrete or continuous). The input is usually a candidate algorithm, algorithm configuration or classifier and a data set. The output is referred to as the metric score or value for a particular candidate. Metrics can be explicitly weighted to balance trade-offs. Regardless of the existence of an explicit weight, a metric always has an implicit weight. For example, if a metric outputs a percentage, this output could be given in the range of 0 to 1 (implicit weight of 1) or from 0 to 100 (implicit weight of 100). A common procedure to minimize the impact of such implicit weights is to normalize the metric so that all metrics share the same range.

We now give some definitions that will be used to present and compare the existing generic MC methods. We define I as an index set over the set of included metrics, M, and let c represent a candidate from a set of candidates, C, while D represents a data set. Each metric, m_i, can be associated with an explicit weight, w_i, a lower metric bound, b_i^l, and an upper metric bound, b_i^h.

2.1 The Efficiency Method.

One of the early studies that investigated the applicability of MC methods for classifier and algorithm evaluation [6] focused on the fact that most candidates feature both positive and negative properties. Quite intuitively, if a positive property is assessed using some metric the value should be as high as possible and if a negative property is assessed the metric value should be as low as possible. The study introduced a generic MC method, called efficiency, which is based on Data Envelopment Analysis (DEA). With this approach, the efficiency of a candidate is defined as:

$$\text{efficiency}(c, D) = \frac{\sum_{j \in J} w_j m_j(c, D)}{\sum_{k \in K} w_k m_k(c, D)} \tag{2.1}$$

using J as an index set over the positive metrics in M and K as an index set over the negative metrics in M. What really separates this method from the other is that the weight for each metric is selected automatically by an optimization algorithm with the goal of maximizing the efficiency score. The idea is that the method should find the most efficient candidate(s) out of a set of candidates by optimizing the efficiency of all candidates so that at least one candidate has an

efficiency score equal to 1 and no candidate has a higher score. The candidates that achieve an efficiency score equal to 1 together form the efficiency frontier.

2.2 The SIM Method. The Simple and Intuitive Measure (SIM) [8] is based on the notion of distance between an evaluated candidate and the optimal candidate (for which all metric values are optimal). This distance, denoted the SIM score, is defined as:

$$\text{SIM}(c, D) = \prod_I |m_i(c, D) - o_i| \tag{2.2}$$

where o_i is the optimal value for m_i. The SIM method does not allow explicitly setting the weights of metrics in order to balance trade-offs, because it is argued that these compromises are often not quantifiable. Instead, a slightly different approach is used, in which a bound, specifically defined for a particular application, is set for each metric. Adding the concept of bounds the original SIM can be refined to:

$$\text{BSIM}(c, D) = \begin{cases} \text{SIM} & \forall i(m_i(c, D) \in \left[b_i^l, b_i^h\right]) \\ \infty & \text{otherwise} \end{cases} \tag{2.3}$$

The SIM score of a candidate indicates the dissimilarity to the optimal candidate and the BSIM score indicates the dissimilarity to the ideal candidate. Whereas SIM and BSIM are mainly intended for candidate ranking, the Normalized Bounded SIM (NBSIM) score is used for the purpose of evaluating a single candidate:

$$\text{NBSIM}(c, D) = \frac{\text{BSIM}}{\prod_i \left|b_i^h - b_i^l\right|} \tag{2.4}$$

The NBSIM score is used to assess the quality of a candidate by representing the dissimilarity to the optimum as the proportion of its BSIM score to the SIM score of the minimally compliant candidate, i.e., the candidate for which all metrics yield their worst possible value. It is argued that the SIM method is adequate for application settings where user preferences are preferably defined as bounds for a set of equally important metrics.

2.3 The Measure-Based Method. Andersson et al. [1] introduced a generic MC framework called measure-based evaluation. The main idea is to define a measure function (MF) for a particular problem. The purpose is to get an explicit distinction between problem formulation, i.e., specifying the measure function and problem solving, i.e., finding a candidate maximizing the measure function. The measure function assigns a value for each combination of problem and candidate describing how well the candidate solves the problem. Thus, as soon as a non-trivial measure function is defined (which is basically any function that does not give the same output for all cases) some candidates will perform better than others. As previously mentioned the measure function should capture, in an algorithm independent way, a suitable algorithm behavior with regard to a particular application. Andersson et al. [1] presents a measure function example which focuses on common heuristics or inductive biases of popular algorithms. For instance, the ID3 tree inducing algorithm tries to find the smallest tree (simplicity) that is consistent with the training data (accuracy). The example MF combines three properties: accuracy on the training data, similarity and simplicity (the inverse of complexity). The similarity property indicates to what extent instances that are close to each other geometrically get assigned the same class label. Measure-based evaluation is very generic in the sense that few restrictions are enforced with regard to weighting and integration. The provided example function can be generalized into a generic method, that works for arbitrary metrics, as follows:

$$\text{MF}(c, D) = \sum_I w_i m_i(c, D) \tag{2.5}$$

3 Analysis.

Some concerns should be raised about efficiency and DEA since they could prove to be detrimental to the solution of some evaluation problems. Firstly, there is no point in applying DEA to optimize the weights if the purpose is to evaluate one single candidate, since the objective of DEA optimization is to rank candidates. Secondly, since DEA automatically adjusts the metric weights to optimize the efficiency score it is not possible for the user to specify the weights to reflect the importance of different metrics. Thus, it imposes restrictions which makes it impossible to manually balance trade-offs.

The term efficiency may also be misleading sometimes since the top ranked candidate according to DEA might not be the most efficient for the application at hand. Intuitively,the most efficient candidate would be the candidate with the most attractive metric scores when taking important trade-offs into consideration. However, the candidate ranked highest by DEA merely represents a candidate for which the optimization algorithm was able to adjust the weights in such a way so that it scored the highest efficiency score. It is not hard to come up with a scenario which involves the ranking of candidates using two metrics (one of them being much

more important to the application at hand than the other). A candidate which scores low for the more important metric and high for the other metric might still be considered the most efficient if its automatically selected weights yield the highest efficiency score. We also argue that the partitioning of metrics into positive and negative is a little bit arbitrary. For instance, accuracy could be a positive metric (as it is) or a negative metric if transformed to error rate (1-accuracy). Further more, the efficiency score is not defined for problems without any negative metrics (division by zero). Thus, it seems better to use the summation approach as done in measure-based evaluation to avoid this problem.

Considering the SIM method, it assumes that if a candidate is within bounds all criteria have equal importance but still claims that the compromise between different criteria is application-specific. We agree with this later claim and argue that SIM is overly restrictive for a generic MC method, which should include the possibility to specify weights. If a particular application depended on an equal metric importance idea this could easily be achieved by setting all explicit weights to the same value. It is also argued in the paper about SIM that the user may not be able to select explicit weights with such detail as required by DEA [8]. This argument seems somewhat confusing since the explicit weights are tuned automatically by an optimization algorithm for the DEA method.

One of the problems that we identified with the SIM method has to do with the dissimilarity measure, i.e., the difference between the optimal candidate and the evaluated candidate. The most relevant candidate for the solution of a particular problem might not always be the (optimal) candidate for which all metrics output the maximum scores, but rather it could be the candidate with the best balanced metric results. We argue that the most valuable component of the SIM method is the bounding feature, followed by normalization. Bounding lets SIM make sure that real-world application constraints are enforced by assigning a score of ∞ for a candidate of which at least one metric is out of bounds.

MF evaluation is the most generic of the three reviewed methods. In general it does not enforce much restrictions on weighting or integration of metrics. As an example, the measure function instance provided in the study of measure-based evaluation [1] is defined as the sum of accuracy and similarity subtracted by complexity, using different explicitly specified weights for each metric. Rather than focusing on how to integrate arbitrary metrics, the MF study essentially focuses on the presentation of the concept of measure-based evaluation, i.e., how to separate problem formulation and solution by establishing a measure function. Since normalization is not applied a metric with a high maximal value could totally dominate other metrics. There is no discussion about acceptable metric ranges either, which would mean that a measure function will output results as usual even though one metric might have been out of bounds, considering the application at hand.

In conclusion, we have identified a number of attractive properties of possible multi-criteria evaluation methods. First of all, it is important that an arbitrary number of metrics can be integrated and that the method itself does not dictate which metrics should be used. Secondly, one should be able to specify explicit weights for each metric to make it possible to properly represent trade-offs. Finally, one should be able to specify the acceptable range for each metric, pertaining to the application at hand.

4 A New Generic MC Method.

We now present a new generic MC method, called the candidate evaluation function (CEF), based on a synthesis of the reviewed methods. Starting from MF evaluation, which is the most generic method, we employ explicit weighting and simple addition of metrics. We have included concepts from SIM, namely bounding and normalization. The concept of bounding enables us to check if application domain constraints are violated. The use of normalization makes sure all metrics have the same range, which is a prerequisite for meaningful weighting. We let the total result be zero if at least one metric violates a bound. We extend the bounding concept to give each metric not just a lower but also an upper bound. The bounds effectively generate an acceptable range. Metrics are normalized so that $m_i(c, D) \in [0, 1]$ where 1 is the best value. If a metric value is higher than the upper bound ($m_i(c, D) > b_i^h$) then it is set to 1. We define CEF as:

(4.6)
$$\mathrm{CEF}(c, D) = \begin{cases} \sum_i w_i m_i(c, D) & \forall i (m_i(c, D) \in \left[b_i^l, 1\right]) \\ 0 & \text{otherwise,} \end{cases}$$

in which $\sum_{i \in I} w_i = 1$ assures that $\mathrm{CEF}(c, D) \in [0, 1]$.

5 Case study.

We now present a study investigating how data mining can be applied to prevent spyware [2]. The occurrence of spyware in applications available over the Internet has become very common. Ideally, countermeasures should not only remove unwanted software, but rather prevent spyware from ever entering computers. The presented study tries to take advantage of the fact that the vendors of spyware-hosting applications try to pose their software as legitimate. There are typically two

Algorithm	Accuracy	TPR	FPR	TET
AdaBoostM1	73.82	0.72	0.24	0.00
DecisionStump	68.82	0.54	0.16	0.00
HyperPipes	76.47	0.91	0.38	0.07
IBk	77.94	0.71	0.15	0.13
J48	73.24	0.72	0.26	0.00
JRip	71.18	0.71	0.29	0.00
KStar	59.71	0.96	0.77	9.20
NaiveBayes	79.41	0.91	0.32	0.11
NaiveBayesNominal	87.94	0.88	0.12	0.00
PART	72.65	0.72	0.26	0.00
RandomForest	75.29	0.79	0.28	0.00
RBFNetwork	77.35	0.75	0.21	0.17
Ridor	67.65	0.63	0.28	0.00
SMO	83.53	0.78	0.11	0.00
VotedPerceptron	81.47	0.85	0.22	0.02

Table 1: Original experimental results: accuracy, true positives rate (TPR), false positives rate (FPR), and testing time (TET) for 15 algorithms. All results are presented with the mean of 10 runs of holdout using a 66% training set / 34% test set randomized split.

Algorithm	TPR	FPR	EXP	TET	CEF
AdaBoostM1	0.72	0.76	0	1.00	0.67
DecisionStump	0.54	0.84	1	1.00	0.00
HyperPipes	0.91	0.62	0	1.00	0.00
IBk	0.71	0.85	0	1.00	0.72
J48	0.72	0.74	1	1.00	0.81
JRip	0.71	0.71	1	1.00	0.79
KStar	0.96	0.23	0	0.00	0.00
NaiveBayes	0.91	0.68	0	1.00	0.00
NaiveBayesNominal	0.88	0.88	0	1.00	0.77
PART	0.72	0.74	1	1.00	0.81
RandomForest	0.79	0.72	0	1.00	0.68
RBFNetwork	0.75	0.79	0	1.00	0.71
Ridor	0.63	0.72	1	1.00	0.00
SMO	0.78	0.89	0	1.00	0.75
VotedPerceptron	0.85	0.78	0	1.00	0.72

Table 2: Multi-criteria evaluation results. Each metric has been normalized (the range is from 0 to 1 and higher values are better). Five algorithms get a CEF score of 0 since they are out of bounds according to one or more criteria.

objectives that counteract in that they want users to download applications that covertly install spyware but without any legal consequences. A common solution is to mention in the End User License Agreement (EULA) that spyware will indeed be installed but to give this information in a way that is hard to understand. The study addresses the spyware problem by mining EULAs of both legitimate (good) and spyware-hosting (bad) software looking for patterns in order to determine if it is possible to detect from the EULA whether the associated software hosts spyware or not. The conducted experiment involved evaluating 15 learning algorithms on a data set containing 50 good and 50 bad EULAs. The results, which can be viewed in Table 1, were promising and so a tool for spyware prevention was suggested. This tool could operate between the operating system and any application to be installed, by classifying the EULA as soon as it appears during installation and providing the user with recommendations about whether to install or not. The tool could also visualize which parts of the EULA that contributed to its classification. We need to make sure that the tool is accurate since it essentially should be able to detect a very high quantity of bad software.

5.1 CEF Evaluation. Analyzing the requirements it is clear that we need metrics for accuracy (of classifying both good and bad applications), time, and explainability. Mapping the requirements to the available experiment data and making informed choices (based on a limited set of data) about bounds and explicit weighting, we suggest the following setup. We let the true positives rate (TPR) and false positives rate (FPR) from the original experiment reflect the correctness of the classification of good software and bad software, respectively. Furthermore, we establish a very simple explainability metric (EXP) which assigns 0 to non human-understandable classifiers and 1 to classifiers that can be interpreted (e.g., rules or trees). Finally, we use the original testing time data (the number of seconds elapsed during the classification phase of the evaluation) as a metric representing the classification time (TET) of the tool. We present a simple instantiation of the CEF measure in Table 3. For this case study we have decided the lower and upper bound, as well as the explicit weight, for each metric after informal discussions with domain experts. We have selected acceptable lower bounds for TPR and FPR considering the small amount of data and we have adopted a weight scheme that simply ranks the metrics according to importance in the following sequence: FPR (most important), TPR, EXP/TET.

The results of the 15 algorithms from the original EULA experiment (see Table 1) have been normalized and can be viewed in Table 2. Although, the weights and bounds were selected for the case study after short informal discussions, the results clearly indicate that the new method can be customized for a particular problem so that the rank of the candidates is dependent of the assessment of all criteria relevant for a particular application. CEF also invalidates candidates if some

Metric	Type	Range	Lower Bound	Upper Bound	Weight
TPR	Cont.	[0 . . . 1]	0.70	1.0	2/7
FPR	Cont.	[0 . . . 1]	0.70	1.0	3/7
EXP	Disc.	0, 1	0	1	1/7
TET	Cont.	[0 . . .	0.89	1.0	1/7

Table 3: A simple instantiation of the generic multi-criteria method CEF with four metrics (three continuous, one discrete). The original range is given, whereas the upper and lower bounds have been normalized. The lower bounds can be explained as follows: TPR must be higher than 70%, FPR must be lower than 30%, and testing time must not take longer than 1 second ($1.0/9.20 \approx 0.89$). The explicit weights sum up to 1.

metric is out of bounds. Comparing with the original results, and especially with the frequently used accuracy metric, the ranking is very different. For instance, when focusing only on accuracy, the best candidates are SMO and NaiveBayesNominal. However, for the tool investigated in our case study, CEF ranks J48 and PART at the top since they combine acceptable true and false positives rates with high explainability, i.e., making it possible to visualize and interpret the classifiers.

6 Conclusions and Future Work.

We investigate the concept of generic multi-criteria (MC) evaluation and perform a comparison of existing methods. We synthesize features from these methods into a new MC method, denoted the candidate evaluation function (CEF), and present a case study in which we apply it and show how to balance critical trade-offs according to the application at hand. The results show that CEF indeed provides a way to customize the evaluation so that several criteria can be balanced and assessed. It also provides a way to check that the requirements of an application are honored. There are several directions for future work. Firstly, we are investigating the concept of measure-based algorithms [1] in which a measure function is defined for a particular problem and used as an inductive bias to reformulate the learning problem as an optimization problem. We have generalized this idea to exchange the performance element of existing algorithms like back-propagated neural networks and rule learners with CEF. Secondly, we intend to develop a prototype of the spyware prevention tool and analyze performance on a larger data set in order to find the most suitable candidate algorithm(s) to include in the tool. Thirdly, we are working on a theoretical study of multi-criteria metrics which suggests the use of software engineering concepts like quality attributes and quality metrics to simplify taking the step from application requirements to a working multi-criteria evaluation.

Acknowledgments

The authors would like to thank Johan Holmgren at Blekinge Institute of Technology for fruitful discussions about the candidate evaluation function (CEF).

References

[1] A. Andersson, P. Davidsson, and J. Lindén. Measure-based Classifier Performance Evaluation. *Pattern Recognition Letters*, 20(11-13):1165–1173, 1999.

[2] M. Boldt, A. Jacobsson, N. Lavesson, and P. Davidsson. Automated Spyware Detection Using End User License Agreements. In *2nd International Conference on Information Security and Assurance*, Busan, Korea, 2008.

[3] R. Caruana and A. Niculescu-Mizil. An Empirical Comparison of Supervised Learning Algorithms. In *23rd International Conference on Machine Learning*, pages 161–168, 2006.

[4] A. Freitas. A Critical Review of Multi-Objective Optimization in Data Mining: A Position Paper. *ACM SIGKDD Explorations Newsletter*, 6(2):77–86, 2004.

[5] N. Lavesson and P. Davidsson. Quantifying the Impact of Learning Algorithm Parameter Tuning. In *Proceedings of the 21st AAAI National Conference on Artificial Intelligence*, pages 395–400. AAAI Press, 2006.

[6] G. Nakhaeizadeh and A. Schnabl. Development of Multi-Criteria Metrics for Evaluation of Data Mining Algorithms. In *3rd International Conference on Knowledge Discovery and Data Mining*, pages 37–42, Newport Beach, CA, USA, 1997.

[7] F. Provost, T. Fawcett, and R. Kohavi. The Case Against Accuracy Estimation for Comparing Induction Algorithms. In *15th International Conference on Machine Learning*, pages 445–453, Madison, WI, USA, 1998. Morgan Kaufmann.

[8] C. Soares, J. Costa, and P. Brazdil. A simple and intuitive measure for multi-criteria evaluation of classification algorithms. In *ECML2000 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pages 87–96, Barcelona, Spain, 2000. Springer Science.

[9] D. H. Wolpert. Off-training-set Error and A Priori Distinctions between Learning Algorithms. Technical Report SFI-TR 95-01-003, Sante Fe Institute, Santa Fe, NM, USA, 1995.