Optimizing Accuracy and Size of Decision Trees

Tea Tušar

Department of Intelligent Systems
Jožef Stefan Institute
Jamova 39, SI-1000 Ljubljana, Slovenia
tea.tusar@ijs.si

Abstract

This paper presents the problem of finding parameter settings of algorithms for building decision trees that yield optimal trees—accurate and small. The problem is tackled using DEMO algorithm, an evolutionary algorithm for multiobjective optimization that uses differential evolution to explore the decision space. The results of the experiments on six datasets show that DEMO is capable of efficiently solving this problem, offering the users a wide choice of near-optimal decision trees with different accuracies and sizes in a reasonable time.

1 Introduction

Domain experts often use machine learning algorithms for finding theories that would explain their data. However, they are usually unfamiliar with these algorithms and do not know how to set their parameters in order to produce the desired results. Moreover, they rarely know beforehand exactly what kind of theory they are looking for. Our approach can help them by exploring the parameter space of the learning algorithms while searching for theories with highest prediction accuracy and lowest complexity. The resulting set of the best found theories gives the users the possibility of comparing the theories among themselves and provides an additional insight into the data. All this helps the users to choose the theory that best suits their needs.

In the following, we limit our discussion of machine learning theories to decision trees for classification, where the best trees (or theories) are regarded as those that are accurate and small. In Section 2 we define the mentioned optimization problem, review the related work and show how the DEMO algorithm can be employed to solve it[1]. Section 3 presents the experiments and their results. The paper concludes with a summary and discussion of future work possibilities in Section 4.

[1]The source of the DEMO algorithm an be downloaded from `http://dis.ijs.si/tea/research.htm`.

2 Accuracy and size as two conflicting objectives

2.1 Optimization problem

The tackled optimization problem consists of finding the parameter settings of machine learning algorithms in order to construct accurate and small trees for a given domain. Accuracy is estimated using 10-fold cross validation, while the size of the tree is equal to the number of all nodes in the tree. Accuracy must be maximized and size minimized. We deal with this problem for the special case of decision trees induced by the C4.5 algorithm [7], or more precisely, its Java implementation in the Weka environment [12] called *J48*.

When building J48 trees, several parameters need to be set (see Table 1). The large amount of possible parameter settings calls for a heuristic method for solving this problem.

2.2 Related work

Kohavi and John [3] searched for parameter settings of C4.5 decision trees that would result in optimal performance on a particular dataset. They considered four parameters: M, C and S with the same meaning as described in Table 1, and G—a binary parameter that determined if the splitting criterion would be information gain or gain ratio. The optimization objective was 'optimal performance' of the tree, i.e., the accuracy measured using 10-fold cross validation. The problem was tackled as a discrete optimization problem and the best-first search was chosen to explore the parameter space. Experiments on 33 datasets showed that best-first search found better parameter settings than the default on nine domains, while on one dataset, default parameter values yielded better trees than the heuristic search.

Similar experiments were performed by Mladenić [4], who searched for the optimal setting of the m-value in m-estimate postpruning of decision trees [2]. The sole optimization objective was accuracy estimated with cross validation. She explored the deci-

Name	Possible values	Default value	Description
M – *number of instances*	$1, 2, \ldots$	2	Minimum number of instances in leaves (higher values result in smaller trees).
U – *unpruned trees*	yes/no	no	Use unpruned tree (the default value 'no' means that the tree is pruned).
C – *confidence factor*	$[10^{-7}, 0.5]$	0.25	Confidence factor used in postpruning (smaller values incur more pruning).
S – *subtree raising*	yes/no	yes	Whether to consider the subtree raising operation in postpruning.
B – *use binary splits*	yes/no	no	Whether to use binary splits on nominal attributes when building the tree.

Table 1: Parameters for building J48 trees.

sion space using different deterministic and stochastic algorithms. The tests on eight datasets showed that the problem is rather simple and that all tested algorithms achieved comparable results.

Both mentioned approaches optimized only the accuracy of the decision trees. To our best knowledge, no work has been done on searching for parameter settings of decision tree building algorithms that would consider accuracy and size of the trees as two optimization objectives.

Bohanec and Bratko [1] searched for good tradeoffs between accuracy and size of decision trees in a different way. They presented the OPT algorithm which explores the space of all trees that can be derived from a complete ID3 tree [6] by pruning, using dynamic programming. The result is an optimal sequence of pruned trees, decreasing in size, such that each tree has the highest accuracy among all possible pruned trees of the same size. While OPT works perfectly for its purpose, it has two serious drawbacks if it was to be applied to serve our needs. The first difficulty is its time complexity, which is quadratic with respect to the number of leaves of the original ID3 tree. The second disadvantage is that the accuracy of the trees is measured on the dataset that was used for constructing these trees by simply counting the additional classification errors made with pruning. If a separate test set would be used for estimating the accuracy, the time needed for building such trees would increase considerably.

2.3 Optimization with DEMO

DEMO [8, 9] is an evolutionary algorithm for multiobjective optimization that uses differential evolution (DE) instead of genetic algorithms (GAs) for exploring the decision space and outperforms state-of-the-art GA-based algorithms on several multiobjective benchmark problems [10].

Since we want to help the users of machine learning algorithms to find good trees without having to search for the right parameter settings manually, we must be careful not to demand from them to set the parameters of DEMO instead. This is why we chose a single parameter setting of DEMO for all the experiments performed on this problem. This setting was in no way fitted to the domains used and should therefore be appropriate for any classification domain.

We used the Lamarckian repair procedure to round the values of the variable M to the first nearest integer and the Baldwinian repair procedure for the binary variable C. The other parameters of DEMO were set as follows:

- population size = 20,
- number of generations = 25,
- DE selection scheme = DE/rand/1/bin,
- scaling factor $F = 0.6$,
- probability in binomial crossover $CR = 0.6$,
- environmental selection procedure = strength Pareto approach (as in SPEA2).

3 Experiments and results

In the experiments, we compare trees found by DEMO to trees found by random search of the decision space and the *default tree*—the tree that is constructed using the default parameters of J48. Because datasets can be very large and consequently the time to build a single J48 tree very long, we limit the number of generated trees by DEMO and random search to 500 each. While this was often not enough for DEMO to converge, we had to persist in the low number of evaluations to provide the final solutions in a reasonable time. Out of all trees, only the nondominated ones are presented in the results. Because DEMO and random search are stochastic algorithms, they were run 10 times on each dataset.

All algorithms were run on six datasets. The first *(EDM)* refers to process parameter selection in electrical discharge machining [11], while the other five *(dermatology*, *nursery*, *splice*, *vehicle* and *vowel)* were obtained from the UCI repository [5].

Figure 1 shows the default tree and the trees found in the best run of DEMO and random search (according to the I_{H} indicator). The comparison between the trees found by the two heuristic methods and the default tree shows that the default tree is often larger and less accurate than the other trees. On the dermatology, nursery, splice and vehicle datasets DEMO always finds trees that weakly dominate the default tree, while this happens in at least half of the runs on

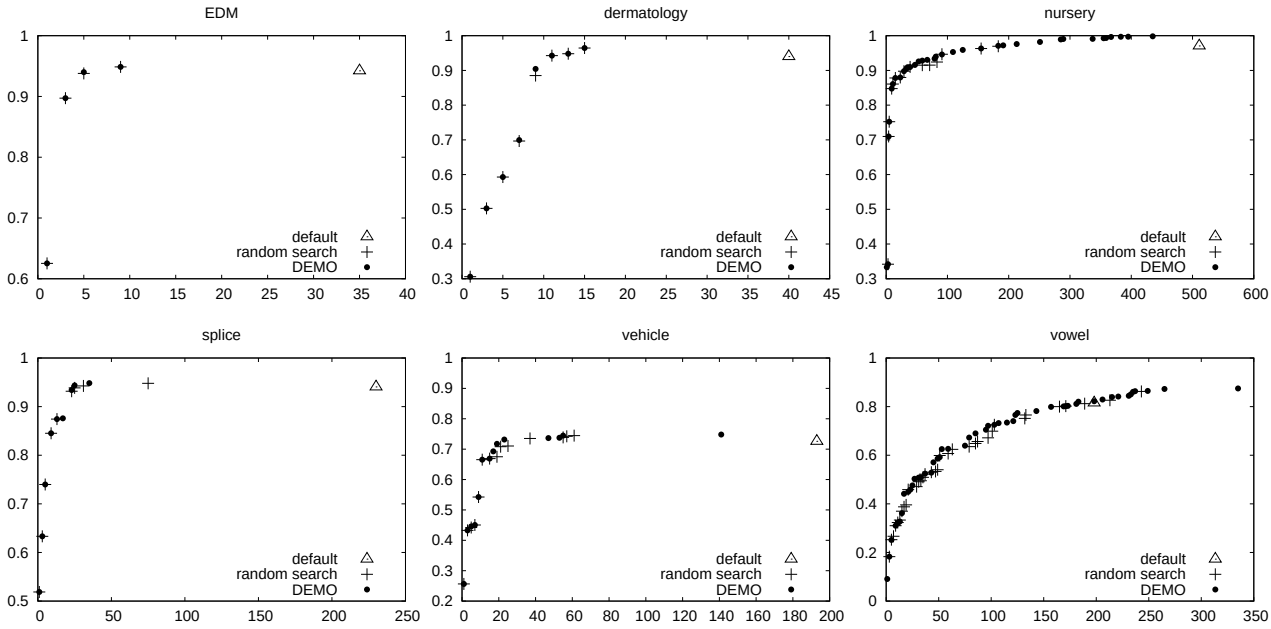

Figure 1: Objective values of trees found by the algorithms on the six chosen datasets. The size of trees is placed on the abscissa, while classification accuracy is represented on the ordinate.

the EDM and vowel datasets. In some of the runs, even trees found by random search dominate the default tree. This proves that it is indeed important for the users of the J48 algorithm to try some other parameter setting beside the default one when building a decision tree model. Additional significance tests have shown that while DEMO outperforms random search with regard to some performance indicator on the five UCI datasets (see [9] for more details on these experiments), there is no significant difference between the algorithms on the EDM dataset.

In a single run of DEMO and random search, 500 parameter settings were inspected by each algorithm. Since all experiments were repeated 10 times, the algorithms jointly built 10000 decision trees for each dataset. This gives us the possibility to look at the decision space of the optimization problem and see if there exist specific parameter settings that induce good decision trees. To this end we gathered all 10000 decision trees for each dataset and denoted which of them are dominated and which are not. Since the decision space is five-dimensional, it cannot be simply presented here. Therefore, we only show its projection on the two-dimensional space, defined by the parameters M and C, where the parameter U is set to *no* (only trees subject to postpruning are considered).

Inspecting the plots of dominated and nondominated trees in Figure 2 we can easily see that the parameter M has a big influence on the size and accuracy of the trees, while the effect of the parameter C seems to be much smaller. This causes the vertical 'stripes' of nondominated trees. When M is large, the trees are subject to heavy prepruning, which leaves little or no room for postpruning. This is why the 'stripes' are so well defined for large values of M. When M is smaller, the quality of trees depends also on the parameter C. Note that the quality of trees is always influenced also by the other three parameters (U, S and B), whose values are not shown in these plots.

The plots indicate that nondominated trees can be found at almost any point in the decision space and that their location depends very much on the dataset. Consequently, no general rule for predicting the optimal parameter values can be found. This gives additional evidence that searching for parameters of decision tree building algorithms has to be performed for every dataset separately.

4 Conclusion

The proposed real-world optimization problem consists of finding the parameter settings of a machine learning algorithm that result in accurate and small decision trees. The results of the performed experiments on six real domains showed that DEMO is capable of efficiently solving this problem, offering the users a wide choice of near-optimal decision trees with different accuracies and sizes to choose from. Such an approach for searching accurate and simple theories could be extended to other machine learning algorithms.

Another direction for future work is to use DEMO

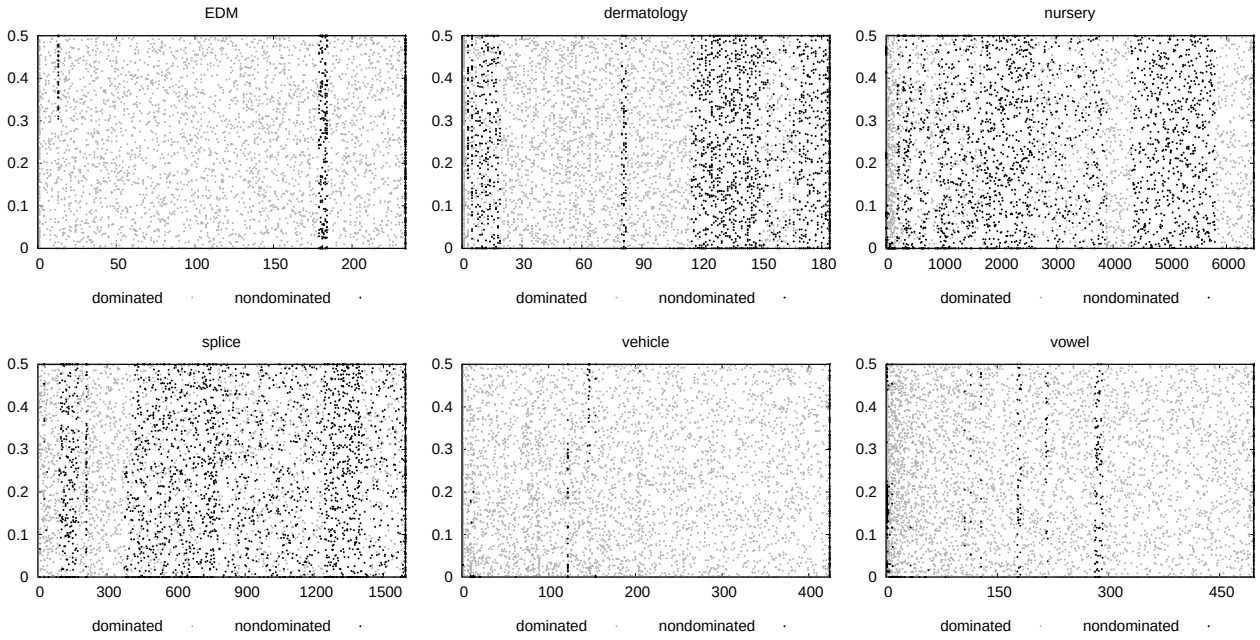

Figure 2: Dominated (grey) and nondominated (black) trees found by DEMO and random search for various values of parameters M (on the abscissa) and C (on the ordinate), while U is set to *no*.

for optimizing more than two objectives for the considered problem. In the case of decision trees, for example, the users might want to optimize also some other objective beside accuracy and size, such as the 'degree of interestingness' of the induced models, estimated in terms of presence or absence of some attributes in the models.

References

[1] M. Bohanec and I. Bratko. Trading accuracy for simplicity in decision trees. *Machine Learning*, 15(3):223–250, 1994.

[2] B. Cestnik and I. Bratko. On estimating probabilities in tree pruning. In *Proceedings of the European working session on learning on Machine Learning (EWSL '91)*, pages 138–150, 1991.

[3] R. Kohavi and G. H. John. Automatic parameter selection by minimizing estimated error. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pages 304–312, 1995.

[4] D. Mladenić. Domain-tailored machine learning. Master's thesis, University of Ljubljana, Faculty of Electrical Engineering and Computer Science, 1995.

[5] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

[6] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[7] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[8] T. Robič and B. Filipič. DEMO: Differential evolution for multiobjective optimization. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 520–533, 2005.

[9] T. Tušar. Design of an algorithm for multiobjective optimization with differential evolution. Master's thesis, University of Ljubljana, Faculty of Electrical Engineering and Computer Science, 2007.

[10] T. Tušar and B. Filipič. Differential evolution versus genetic algorithms in multiobjective optimization. In *Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*, pages 257–271, 2007.

[11] J. Valentinčič and M. Junkar. Detection of the eroding surface in the EDM process based on the current signal in the gap. *The International Journal of Advanced Manufacturing Technology*, 28(3-4):294–301, 2006.

[12] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.