ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-717

Multi-objective learning of hybrid classifiers

Rok Piltaver[1,2] and **Mitja Luštrek**[1] and **Jernej Zupančič**[1,3] and **Sašo Džeroski**[1] and **Matjaž Gams**[1,2]

Abstract. We propose a multi-objective machine learning approach guaranteed to find the Pareto optimal set of hybrid classification models consisting of comprehensible and incomprehensible sub-models. The algorithm run-times are below 1 s for typical applications despite the exponential worst-case time complexity. The user chooses the model with the best comprehensibility-accuracy trade-off from the Pareto front which enables a well informed decision or repeats finding new Pareto fronts with modified seeds. For a classification trees as the comprehensible seed, the hybrids include single black-box model, invoked in hybrid leaves. The comprehensibility of such hybrid classifiers is measured with the proportion of examples classified by the regular leaves. We propose one simple and one computationally efficient algorithm for finding the Pareto optimal hybrid trees, starting from an initial classification tree and a black-box classifier. We evaluate the proposed algorithms empirically, comparing them to the baseline solution set, showing that they often provide valuable improvements. Furthermore, we show that the efficient algorithm outperforms the NSGA-II algorithm in terms of quality of the result set and efficiency (for this optimisation problem). Finally we show that the algorithm returns hybrid classifiers that reflect the expert's knowledge on activity recognition problem well.

1 INTRODUCTION

In many real-life domains, a large part of expert knowledge can be represented in a comprehensible way, but there is usually also a part of the knowledge that is complex and difficult to formulize. The models should thus consist of comprehensible parts containing knowledge comprehensible to a human and incomprehensible parts enabling improved accuracy. This paper deals with the task of finding such hybrid models with the best trade-off between two conflicting objectives: accuracy and comprehensibility. The preferred approach to learning a model while considering multiple objectives is multi-objective learning [1] that returns the Pareto optimal set from which the user selects a single solution. The set contains pairwise incomparable solutions that are better than any other solution not belonging in the set i.e. non-dominated solutions.

While several existing machine learning (ML) algorithms already use multiple objectives to find the best models, the goal here is somewhat different: to extract knowledge similar to the knowledge of human experts from domain examples. Our algorithm takes an initial comprehensible classifier, a black-box (BB) classifier, and a set of training data as input. The result of the algorithm is a set of hybrid classifiers consisting of comprehensible and BB parts, similarly to how expert knowledge is structured.

[1] Jožef Stefan Institute, Ljubljana, Slovenia, email: {rok.piltaver, mitja.lustrek, jernej.zupancic, saso.dzeroski, matjaz.gams}@ijs.si
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[3] University of Ljubljana, Faculty of Mathematics and Physics, Ljubljana, Slovenia

As a motivating example we will consider activity recognition using acceleration data. The goal is to recognize the activity of a person wearing an accelerometer. An accurate black-box classifier can be constructed using high-quality laboratory data. However, since we were about to enter the EvAAL live activity recognition competition (http://evaal.aaloa.org/), we wanted a classifier that we could trust to perform correctly in a situation substantially different from the one in the laboratory. Since a completely understandable classifier performed poorly, a hybrid approach was called for.

The following sections present related work, our algorithm MOLHC (Multi-Objective Learning of Hybrid Classifiers), theoretical analyses, practical experiments and discussion.

2 RELATED WORK

The related work for this paper comes from two areas: increasing the comprehensibility of models constructed with ML, both in a single- and multi-objective way, and constructing hybrid models that combine elements of multiple model types.

Some established ML algorithms already use criteria other than accuracy in the learning process. For example, the RIPPER rule induction algorithm [2] ensures the compactness of the rule set by using the minimum description length principle. Girosi [3] included comprehensibility criteria in the regularization of artificial neural networks. Jin [4] and Gorzałczany and Rudziński [5] did likewise in the construction of fuzzy rules with evolutionary optimization methods. Common to these approaches is that they combine the accuracy and comprehensibility as a weighted sum in a single objective function, and output a single model.

One of the earliest examples of multi-objective ML is the work by Kottathra and Attikiouzel [6], who formulated the training of an artificial neural network as a bi-objective minimization problem with the mean square error and the number of neurons as conflicting objectives. They tackled the problem with a branch-and-bound algorithm, while later work mostly used evolutionary optimization methods. Examples include Ishibuchi et al. [7] and Pulkkinen [8], who constructed fuzzy rules; Jin et al. [9], who constructed artificial neural networks; Markowska-Kaczmar and Mularczyk [10], who extracted rules from neural networks; Tušar [11], who constructed classification trees; and Clark and Everson [12], who constructed relevance vector machines.

In the area of hybrid ML models, the best-known example are model trees, which have linear functions in the leaves [13]. The resulting models are accurate and compact. NBTrees place Naive Bayes classifiers in the leaves of classification trees [14]. This retains the comprehensibility of both model types and often improves the accuracy. The VFDTc algorithm for mining data streams similarly combines Naive Bayes with Hoeffding trees, improving the accuracy compared to regular Hoeffding trees [15].

Hybrid SVM-based classification trees [16] use a SVM classifier to classify examples close to the class boundary, and a classification tree for the examples farther away. This results in a significant speedup over SVM alone, without any compromise in accuracy.

Our research builds upon existing work on multi-objective ML, but introduces important distinctions. First, most related work defines the comprehensibility as the complexity of the classifier. In contrast, we define the comprehensibility as the fraction of examples classified by the comprehensible part of our hybrid classifiers. Such hybrid classifiers are ideally suited to searching for trade-offs between the accuracy and comprehensibility, because the comprehensibility can be dialled to anywhere between none and complete. Second, we use a novel algorithm that is guaranteed to find the complete Pareto optimal set, which is an improvement over the commonly used stochastic optimization algorithms. Third, our use of hybrid trees is motivated not only by the classification accuracy of models or the speed of learning them, but also by the desire to build models mimicking a human expert's knowledge: with some comprehensible parts (represented by a tree), and others incomprehensible. Finally, we are interested in an interactive learning process where the human expert provides some initial knowledge (tree) to build upon and possibly re-uses some of the proposed alternative models to further direct the learning process.

3 THE MOLHC ALGORITHM

The inputs to the algorithm are: an initial classification tree, a BB classifier and a set of examples used to evaluate the performance of the initial tree and the BB classifier in each subspace of the attribute space. The initial tree should be one that a human expert considers as comprehensible frame for knowledge representation, and can be constructed manually or by ML. Subspaces are defined by the initial classification tree: a subspace is composed of the examples belonging to a single leaf of the classification tree.

The goal of the multi-objective learning algorithm is to find a set of hybrid trees (derived from the initial classification tree and the BB classifier) which is Pareto optimal with respect to the two objectives: classification accuracy and comprehensibility. The algorithm constructs hybrid trees by replacing one or several leaves in the initial classification tree with the provided BB classifier. Some subspaces of the attribute space are thus classified by the initial classification tree and others by the BB classifier.

When classifying a new example, the algorithm first checks in which leaf (subspace) the example belongs. If it belongs in a leaf marked as regular, the example is classified as the majority class of that leaf – as in a regular classification tree. If it belongs in a leaf marked as hybrid, it is classified with the BB classifier.

The **accuracy** a_t of a hybrid tree t is defined as the ratio between the number of correctly classified examples and the number of all examples N used to evaluate the accuracy of the hybrid tree (Equation 1). The number of correctly classified examples is added over all the leaves in the hybrid tree: for a leaf j, the number of correctly classified examples is denoted $N_{j,t}$ if j is marked as regular, and $N_{j,bb}$ if j is marked as hybrid.

$$a_t = (\sum_{j\ marked\ regular} N_{j,t} + \sum_{j\ marked\ hybrid} N_{j,bb}) / N \quad (1)$$

The **comprehensibility** c_t of a hybrid tree t is defined as the ratio between the number of examples that are classified by the regular leaves and the number of all examples used to evaluate the comprehensibility of the hybrid tree (Equation 2).

$$c_t = (\sum_{j\ marked\ regular} N_j) / N \quad (2)$$

By definition, the comprehensibility of the initial classification tree is 1 and the comprehensibility of the BB classifier is 0, while the comprehensibility of hybrid trees are between 0 and 1. In reality, very large classification trees are not particularly comprehensible, so it makes sense to limit their size, as we do in Section 4.1 below.

To **compare hybrid trees** when dealing with multiple objectives, the Pareto dominance relation ($\preccurlyeq$) is used [17]. Equation 3 defines the relation where the goal is to maximize the accuracy and comprehensibility.

$$x \preccurlyeq y \Leftrightarrow (a_x > a_y \wedge c_x \geq c_y) \vee (a_x \geq a_y \wedge c_x > c_y) \quad (3)$$

A solution x is said to be non-dominated in a set of solutions S if no solution that dominates x exist in S (Equation 4).

$$x \in non\text{-}dom(S) \Leftrightarrow \nexists y \in S: y \preccurlyeq x \quad (4)$$

A set of solutions $P' = non\text{-}dom(P)$ is a non-dominated set among the set of solutions P if it is composed of the solutions that are not dominated by any member of the set P. The **Pareto set** (or globally Pareto-optimal set) is the non-dominated set of the entire solution space [17]. The goal of our algorithm is to find the Pareto set of hybrid trees. The user of the algorithm only needs to select one or more of the hybrid trees from the Pareto set P' and does not need to consider hybrid trees outside P', as it is guaranteed that all of them are worse than at least one of the hybrid trees from P'.

3.1 Naive implementation

In order to find the Pareto set of hybrid trees, the algorithm needs to search the entire search space of 2^n hybrid trees. The search space can be represented as the Cartesian product $\{0,1\}\times\ldots\times\{0,1\} = \{0,1\}^n$ where n is the number of leaves considered for replacing with the BB classifier: the leaves where the BB classifier is more accurate than the initial classification tree. The values 0 and 1 in each component of the product denote the type of corresponding leaf: 0 for regular and 1 for hybrid leaves. The initial classification tree is represented as $S_0 = (0,0,\ldots,0)$, while the BB classifier is represented as $S_{2\hat{}n} = (1,1,\ldots,1)$. By considering all possible replacements of leaves in the initial tree, the replacements of all the subtrees are implicitly considered as well because replacing all the leaves of a subtree is equivalent to replacing the root of the subtree for a BB - if an example belongs to the root of the subtree, it must belong to one of its leaves.

The algorithm starts by splitting the set of labelled examples into subspaces of the attribute space corresponding to the leaves of the initial classification tree. Afterwards, it computes the number of examples N_j belonging to each subspace/leaf j. It also computes the number of examples correctly classified by the initial classification tree $N_{j,t}$ and the BB classifier $N_{j,bb}$ in each leaf j. Finally, it computes the relative difference in accuracy $\delta_{j,a}$ (Equation 5) and comprehensibility $\delta_{j,c}$ (Equation 6) introduced by replacing the leaf j for a BB leaf. The relative differences are used to compute quality of hybrid trees using dynamic programing as described below.

$$\delta_{j,a} = (N_{j,bb} - N_{j,t}) / N \quad (5)$$

$$\delta_{j,c} = -N_j / N \quad (6)$$

A hybrid tree S_i can be transformed into a new hybrid tree S_j by replacing a leaf l marked as regular in S_i for a BB leaf in S_j, thus obtaining $S_j = (s_{j,1}, s_{j,2}, \ldots, s_{j,n})$, where $s_{j,l} = 1$ and $s_{j,k} = s_{i,k}$ for each $k \neq l$. The accuracy and comprehensibility (a_j, c_j) of the new hybrid tree S_j are computed from the accuracy and comprehensibility (a_i, c_i) of the original hybrid tree S_i using Equation 7.

$$(a_j, c_j) = (a_i + \delta_{l,a}, c_i + \delta_{l,c}) \quad (7)$$

The relative difference in accuracy and comprehensibility (Equation 7) between the hybrid trees S_i and S_j, depends solely on the relative differences in accuracy $\delta_{l,a}$ and comprehensibility $\delta_{l,c}$ of the leaf l, and not on any other property of the hybrid tree S_i: from Equations 1 and 5, it follows that $\delta_{l,a} = a_i - a_j$, and from Equations 2 and 6 that $\delta_{l,c} = c_i - c_j$.

The algorithm proceeds with an exhaustive search to find and evaluate all the hybrid trees. Each iteration of the exhaustive search algorithm (Algorithm 1) finds a set of hybrid trees that have one more hybrid leaf than the set found in the previous iteration. Two sets are used: *Proc* consisting of processed hybrid trees and *UProc* consisting of unprocessed hybrid trees. The algorithm begins with an empty set *Proc* and the set *UProc* containing only the initial classification tree S_0, i.e., with the call find({},{S_0}). The algorithm processes the set of unprocessed hybrid trees *UProc* as follows. For each hybrid tree S_i belonging to *UProc* it generates a set of new hybrid trees by replacing each regular leaf l in turn with a hybrid leaf. Each new hybrid tree is added to a temporary set *UProc'* and its comprehensibility and accuracy are computed using Equation 7. After that, the hybrid tree S_i is considered processed and is added to the set *Proc*. If at least one new hybrid tree has been generated (i.e., *UProc'* is not empty), the algorithm runs a new iteration using the temporary set *UProc'* as the new set of unprocessed hybrid trees *UProc*.

Algorithm 1. MOLHC, naïve implementation

```
find(Proc, UProc) {
repeat {
  UProc' = {};
  for (each S_i in UProc) {
    for (each l: s_i,l = 0) {
      S_j = S_i; s_j,l = 1;
      a_j = a_i + δ_l,a; c_j = c_i + δ_l,c;
      if (S_j not in UProc') {UProc'.add(S_j);}
    } Proc.add(S_i);
  } UProc = UProc';
} until (UProc' ≠ {});
return Proc;}
```

3.2 Efficient implementation

The naïve implementation searches the entire search space of hybrid trees, therefore it faces a combinatorial explosion, making its run-time unacceptable even for initial classification trees with as few as 15 leaves considered for marking as hybrid. Therefore we propose two optimizations: avoid generating dominated hybrid trees and avoid generating a hybrid tree more than once.

The key to **avoid generating dominated hybrid trees** is the fact that only non-dominated regular leaves should be replaced with hybrid leaves, since replacing a dominated leaf produces a dominated hybrid tree. Furthermore, no non-dominated hybrid tree can be generated from a dominated hybrid tree by replacing a subset of its leaves with hybrid leaves (except the ones that can also be generated from non-dominated trees). This enables an efficient implementation of the algorithm that is correct in the sense that it finds the complete Pareto optimal set of hybrid trees, even though it does not examine the entire search space. The proof is available in the Supplementary Material (SM) [18], Section 2.2, page 8.

Note that leaves (subspaces) can be compared using the Pareto dominance relation similar as trees: the two objectives are the relative differences in accuracy $\delta_{j,a}$ and comprehensibility $\delta_{j,c}$ (instead of the accuracy and comprehensibility themselves), and both need to be maximized.

The algorithm 1 needs to be changed so that the loop `for (each l: s_i,l = 0)` is replaced with `for (each l: l ∈ non-dom{l: s_i,l = 0})`, hence only non-dominated regular leaves get replaced with hybrid leaves. Therefore the algorithm needs to maintain the set of non-dominated leaves in each hybrid tree: it computes a set of leaves D_l that are directly dominated by the leaf l for each leaf l in the initialization phase. Directly dominated leaves D_l are the ones that are in a transitive reduction of the Pareto dominance relation on the set of all leaves in the initial classification tree with the leaf l. When generating a new hybrid tree S_j from an existing hybrid tree S_i by replacing a regular leaf l with a hybrid leaf, the set of non-dominated leaves L_j in the new tree S_j is computed using dynamic programming (Equation 8) from the set of non-dominated leaves L_i in the original tree S_i. This approach is considerably more efficient than examining all the leaves of the hybrid tree S_j for dominance.

$$L_j = \textit{non-dom}(L_i \cup D_l \setminus l) \tag{8}$$

The new set L_j includes all the leaves in L_i except the leaf l. However, since l is removed, the set L_j should be expanded with the set D_l consisting of leaves directly dominated by l.

If there are two incomparable leaves l_1 and l_2, the naive algorithm would generate a hybrid tree that has both leaves replaced with hybrid leaves twice: (1) by replacing l_1 first and l_2 in a later iteration, and (2) by replacing l_2 first and l_1 in a later iteration. In order to **prevent this duplication**, the leaves are enumerated and replacing only a leaf l in a hybrid tree S_i that has a larger number than any already replaced leaf in S_i is allowed. In order for this second improvement of the algorithm not to interfere with the first one (avoiding dominated hybrid trees), the enumeration must be such that any leaf x that dominates another leaf y has a smaller index than y, i.e., the enumeration must correspond to the non-dominated sorting [17].

Despite the fact that the efficient implementation limits the search space, it can still generate some dominated hybrid trees, therefore non-dominated hybrid trees have to be selected from the set of all generated hybrid trees before returning the result of the algorithm (proof available in the SM [18], Section 2.3, page 14).

4 EXPERIMENTAL RESULTS

4.1 Time complexity and run-times

The naïve and efficient implementations both have exponential time complexities $O(2^n)$ in the worst case, where n is the number of leaves in the initial classification tree considered for replacing with hybrid leaves (proof in SM [18], Section 2.1, page 5). Since the problem is obviously at least as difficult as the NP-hard 0/1 knapsack problem, no algorithm that solves the problem in polynomial time is expected to exist. However, the efficient implementation has considerably lower run-time as shown in Fig. 1: the measurements can be approximated with 2^{2n} and $2^{0.4n}$. Each measurements in Fig. 1 represents an average run-time over 100 randomly generated trees with relative differences in accuracy and comprehensibility of leaves similar to the typical initial trees (10 trees for run-times over 5 min were used). The average run-time of the naïve implementation of the algorithm in Java™ for a tree with 18 leaves is around 18 minutes, (3.2 min for 17 leaves) while it is

only 0.8 ms for the efficient implementation (< 0.5 s for 40 leaves) on a 3 GHz Intel® Core™ 2 Duo computer.

Since the goal is to find classifiers comprehensible to humans, only small initial trees are expected as an input to the algorithm. Since most people are not capable of maintaining more than 5 to 7 conditions in their short-term memory, only binary trees with approximately 32 (or at most 128) leaves at most are reasonable. Therefore one can expect that the efficient implementation of the algorithm can find the Pareto set within seconds.

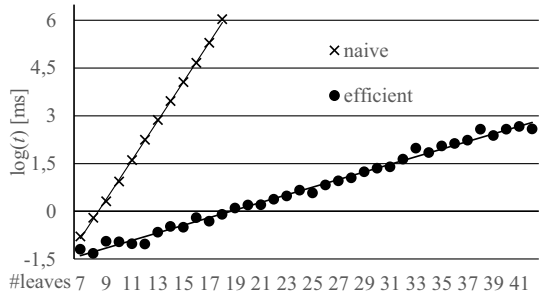

Figure 1. Run-times of the naive and the efficient algorithm

4.2 Measuring the success of MOLHC

To evaluate the MOLHC algorithm, we tested it on several datasets from the UCI repository. The results were compared with standard single-objective machine learning algorithms to assess the gain of MOLHC and with a state-of-the-art multi-objective optimization algorithm NSGA-II [19] in terms of execution time. Finally, an interactive experiment with a human expert was performed on dataset in the activity recognition domain.

We selected the testing datasets from the set of 94 classification datasets from the UCI repository [20] available in ARFF format at the Weka webpage [21]. Among the 49 datasets with more than 300 instances we chose the 23 datasets where the BB classifier achieved at least 10 % better accuracy than the tree with approximately 20 leaves. Finally 40 experiments with the MOLHC algorithm were conducted: one with a small tree (~20 leaves) for each of the 23 datasets and another with a big tree (~40 leaves) if the dataset allowed building such tree.

The following algorithms were used to build the BB classifiers: SVM, kNN, aNN, logistic regression and Naive Bayes, all implemented in Weka [22] and used with the default algorithm parameters. Among them, the classifier with the highest classification accuracy computed using 10-fold cross-validation was chosen as the BB classifier for each dataset. The two initial classification trees were obtained using the J48 algorithm, which is the Weka implementation of the C4.5 algorithm, using the pruning parameters to obtain trees with ~20 and ~40 leaves.

The results of multi-objective algorithms are usually compared using the hypervolume metric [23], which measures the volume of the dominated objective space between a reference point and the attainment surface – the envelope marking all the solutions which are sure to be dominated by the set of solutions returned by the algorithm. Since we have two objectives, the hypervolume is in fact the area under the attainment surface (the broken line in Fig. 2). The reference point was set to (–0.2, –0.2) following the typically used rule-of-thumb advocating a space that is a little bit larger than the actual objective space.

The results of the MOLHC are compared with the set of baseline solutions (Table 2) consisting of the initial classification tree and the BB classifier, and with the set of hybrid trees found by the NSGA-II algorithm (Table 4). The gain of the MOLHC is expressed as the difference in terms of hypervolume.

Two additional important measures of success are the average differences in the accuracy e_a and comprehensibility e_c of the hybrid trees in the solution set, estimated on the training dataset on one hand, and on the separate test dataset on the other hand. This is important since the user needs to select hybrid trees from the solution set based on the Pareto front (an image of the Pareto set in the objective space, e.g. Fig. 2). The solutions are positioned on the Pareto front based on the performance of the learned hybrid trees on the training dataset. If these are different from the results on the test set, the user is misled and may not choose the appropriate hybrid tree. The differences are defined as the average absolute error and are computed using Equations 9 and 10, where S is the set of solutions returned by the algorithm with cardinality $|S|$ and a'_j and c'_j are the classification accuracy and the comprehensibility of the classifier j computed on the test set, respectively.

$$e_a = \sum_{j \in S} |a_j - a'_j| \,/\, |S| \quad (9)$$

$$e_c = \sum_{j \in S} |c_j - c'_j| \,/\, |S| \quad (10)$$

4.3 Comparison with the baseline

Fig. 2 shows the predicted and validated (on a separate test set) results of the MOLHC and the baselines (the initial tree and the BB classifier) as fronts in the objective space. The predicted attainment surface is shown for the hybrid trees obtained by the MOLHC.

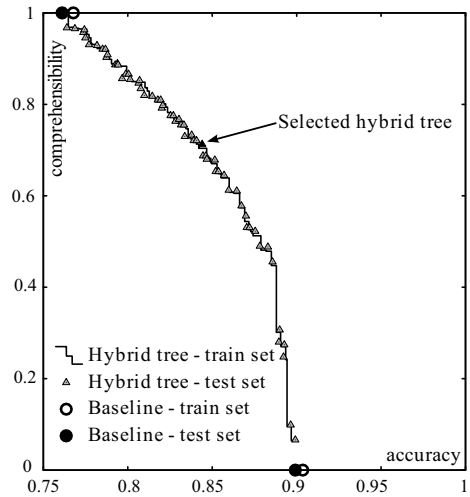

Figure 2. Pareto fronts in the objective space for the EvAAL domain

Table 1 shows the data about seven domains: three with the best improvements, one with average improvement and three with the worst improvements out of the 40 experiments.

Table 2 shows the comparison of the MOLHC and the baseline for the seven domains. Two columns show the hypervolume of the Pareto sets of the hybrid trees produced by MOLHC and the baselines, validated on the test set consisting of one third of all the instances in a dataset. An experiment was also performed using 10-fold cross validation, obtaining comparable results for the datasets with enough instances and exhibiting high deviations of the results for the small datasets. The column with bold font shows the differences in hypervolume between MOLHC and baseline. The last two columns show the average absolute errors of the predicted accuracy e_a and comprehensibility e_c.

Table 1. General data about the 3 best, an average, and 3 worst domains

Dataset	#inst. ◊	#cls. □	#atr. ○	BB accuracy	Tree accuracy
mfeat-pixel	2000	10	241	100	42,72
letter	20000	26	17	100	39,66
vowel	990	11	14	100	44,55
mfeat-zernike	2000	10	48	89,06	83,81
cylinder-bands	540	2	38	90,00	66,66
balance-scale	625	3	5	92,82	88,04
flags	194	8	30	100	86,20

◊ number of instances □ number of classes ○ number of attributes

Table 2. MOLHC evaluation and comparison with baseline

Dataset	MOLHC hypervol.	Baseline hypervol.	Hypervol. difference	e_a	e_c
mfeat-pixel	1,138	0,833	**0,305**	0,024	0,010
letter	1,108	0,815	**0,293**	0,005	0,005
vowel	1,003	0,772	**0,231**	0,019	0,018
mfeat-zernike	1,154	1,031	**0,123**	0,016	0,015
cylinder-bands	1,028	1,002	**0,026**	0,006	0,014
balance-scale	1,229	1,201	**0,029**	0,030	0,019
flags	0,914	0,896	**0,017**	0,023	0,07

Table 3 shows the relative improvements of hypervolume obtained with MOLHC compared to baseline in percentages for all the 40 experiments: the average improvement is 11.17 %, the best improvement is 36.6 % and the worst 1.9 %.

Table 3. The improvement of MOLHC vs. baseline measured as the relative increase of hypervolume in % for the 40 experiments

36.6	26.5	13.7	11.9	10.5	7.6	6.9	5.6	3.8	2.7
36.0	24.8	12.6	11.7	9.7	7.4	6.8	5.1	3.5	2.6
29.9	18.6	12.2	10.8	8.9	7.0	6.6	4.9	2.8	2.4
27.3	16.3	12.0	10.6	8.2	6.9	5.9	4.6	2.7	**1.9**

4.4 Comparison with the NSGA-II algorithm

The efficient implementation of MOLHC was compared to the state-of-the-art stochastic multi-objective optimization algorithm NSGA-II. The task of both algorithms was to search the space of hybrid trees for (approximation of) the Pareto set: trees with accuracy and comprehensibility as the two objectives. The DEAP framework (Distributed Evolutionary Algorithms in Python) [24] implementation of NSGA-II was used. The parameters of the NSGA-II algorithm were set as follows: two point crossover was used as the crossover technique, and bit flip as the mutation operator with the probability of a mutation equal to 1 / *number of leaves* for each bit. The stopping criterion was defined by setting the NSGA-II execution time limit to 10, 50 and 100 times longer than the average run-time over 10 runs of the MOLHC. Three sizes of population were set to: 6.3 × *number of leaves* – 15 (medium population), half that number (small) and twice that number (big). The formula was based on a linear approximation of the number of hybrid trees in the Pareto set in relation to the number of leaves considered to be marked as hybrid. The number of generations was not limited explicitly; however it depended on the run-time limit and the size of the population.

Table 4 shows the average relative increase of hypervolume expressed in % between the proposed efficient algorithm and the average over five runs of the NSGA-II algorithm averaged over six domains. The table includes data for two sizes of the initial decision trees, the three run-time multipliers, and the three sizes of the population used by NSGA-II. The results show that the MOLHC outperforms NSGA-II when searching for the trees of comprehensible size. However, NSGA-II might be preferred for large initial trees with more than 100 leaves (considered incomprehensible in this paper) in order to obtain an approximation of the Pareto set since MOLHC run-times reach the limits of practicality for such tree sizes.

Table 4. Relative difference in % of hypervolume - MOLHC vs. NSGA-II

Run-time /Population size	**Small tree** 12.8 leaves Big	Med.	Small	**Big tree** 22 leaves Big	Med.	Small
t(MOLHC) × 10	3.8	1.8	1.4	5.5	4.9	2.7
t(MOLHC) × 50	0.2	0.1	0.2	2.8	0.6	0.4
t(MOLHC) × 100	0.02	0.03	0.2	1.5	0.3	0.1

4.5 Domain expert validation

Besides quantitative evaluation of MOLHC on the UCI repository datasets, we also performed an experiment which involved interaction with a domain expert for the task of activity recognition described in the introduction. The data (from which a classifier that won at the EvAAL competition was built), contain examples of 10 activities (class) that have to be recognized from the movement of 9 persons. There are 48.000 instances with 61 attributes computed from the data measured by an accelerometer placed on person's chest. The best BB classifier (random forest) achieved an accuracy of 90.6 on this data (Fig. 2).

In the first iteration, the activity recognition expert found the initial classification tree (67.9 accuracy, 8 leaves) constructed using all the 61 attributes as well as the resulting hybrid trees poorly comprehensible, so he selected a subset of 12 attributes to build an initial tree (67.4 accuracy, 8 leaves) for the second iteration. From the resulting Pareto set the expert chose the hybrid tree in which 3 leaves - containing mostly instances of 3 classes that are difficult to distinguish using simple rules - were merged into a single BB classifier. The hybrid tree contained overall 6 leaves, accuracy was 80.2 and comprehensibility 0.689. In the third iteration (Pareto front in Fig. 2) a larger initial tree (76.1 accuracy, 12 leaves) was used. Finally the expert choose a hybrid tree that achieved (84.1, 0.721) and contained 7 regular and 3 BB leaves - one replacing a subtree with 3 leaves. The expert judged this hybrid tree accurate enough, all the regular leaves comprehensible, and corresponding classification rules correct, as well as confirmed that accurately classifying instances belonging to the BB leaves is impossible using simple rules. An additional accelerometers was suggested in order to further improve the comprehensibility and accuracy.

5 DISCUSSION AND CONCLUSION

In this paper, we introduce MOLHC – a novel algorithm for multi-objective learning aiming to find hybrid classifiers that resemble expert knowledge. The approach relies on hybrid trees that incorporate regular leaves and BB classifier. A human provides an initial classification tree which he considers a proper frame for the representation of the knowledge about the domain – with interpretable attributes, reasonable size, and consistent with his knowledge.

MOLHC provides a list of non-dominated hybrid trees according to the two criteria: accuracy and comprehensibility,

enabling the human to finally select the hybrid tree that best suits his needs. A comprehensibility measure of hybrid trees is introduced based on the number of examples that fall in the regular leaves of the hybrid tree. This is different from most other approaches that equate comprehensibility with tree size.

An important advantage of the MOLHC over the usual stochastic approaches to multi-objective learning is that it is guaranteed to find the complete Pareto set of solutions - a theoretical proof is provided. At the same time, the MOLHC is reasonably efficient: the average run-time on the test datasets is approximately $2^{0.4n}$ where n is the number of leaves in the initial classification tree, an improvement over the 2^{2n} complexity of exhaustive search. For small trees with up to 50 leaves (trees need to be small to be understandable), the Pareto set can be found in seconds.

This allows for interactive data-mining with a human in the loop, where the system quickly proposes a set of models based on the user's initial input, but with improved classification accuracy. The models, based on the human input, preserve the structure of the expert knowledge, which often consists of comprehensible and incomprehensible parts. The overall process can be iterative, with outputs from one phase being inputs for the next phases.

We showed that MOLHC performs well on 23 datasets from the UCI repository. We also tested it on the activity recognition domain in cooperation with a human expert, who confirmed that the algorithm produces models that correspond well to his knowledge. Compared to the NSGA-II algorithm, the MOLHC achieved better results in all experiments – it found all the solutions in the Pareto set and spent less run-time.

MOLHC algorithm also has some limitations. First, it finds hybrid classifiers with improved accuracy if the provided BB classifiers achieves better accuracy than the initial tree at least in some parts of the domain (instances belonging to a leave or subtree of the initial tree). Furthermore, the size of the Pareto set containing hybrid trees depends on the number of leaves considered to be replaced with a hybrid leaf: in the experiments it varies from 4 to 1500 (average 140). Second, the dataset must be reasonably large, so that there are enough examples in each leaf of the initial classification tree to reliably decide whether to replace it with the BB classifier. The reliability of estimating the accuracy and comprehensibility of hybrid trees on the test set also suffers if the dataset is too small.

In the future, we plan to extend our approach in several directions. First, we will improve the reliability of the decisions to replace a leaf with the BB classifier. Second, we will test the approach on different classifiers, such as classification rules and expert-crafted trees, and examine using multiple types of BB classifiers in a single hybrid tree. Third, extensive tests are planned in various domains, both in an automated fashion and in interaction with human experts.

REFERENCES

[1] Jin, Y.: Multi-Objective Machine Learning. Studies in Computational Intelligence, vol. 16. Springer, Heidelberg (2006)

[2] Cohen., W.W.: Fast Effective Rule Induction. In: 12th International Conference on Machine Learning, pp. 115–123. Morgan Kaufmann, San Francisco (1995)

[3] Girosi, F., Jones, M., Poggio, T.: Regularization Theory and Neural Networks Architectures. Neural Computation 7, 219–269 (1995)

[4] Jin, Y.: Fuzzy Modelling of High-Dimensional Systems: Complexity Reduction and Interpretability Improvement. IEEE Transactions on Fuzzy Systems 8(2), 212–221 (2000)

[5] Gorzałczany, M.B., Rudziński, F.: Accuracy vs. Interpretability of Fuzzy Rule-Based Classifiers - An Evolutionary Approach. In: 2012 International Conference on Swarm and Evolutionary Computation, pp. 222–230. Springer, Heidelberg (2012)

[6] Kottathra, K., Attikiouzel, Y.: A Novel Multicriteria Optimization Algorithm for the Structure Determination of Multilayer Feedforward Neural Networks. Journal of Network and Computer Applications 19(2), 135–147 (1996)

[7] Ishibuchi, H., Nakashima, T., Murata, T.: Three-Objective Genetics-Based Machine Learning for Linguistic Rule Extraction. Information Sciences 136, 109–133 (2001)

[8] Pulkkinen, P.: Multiobjective Genetic Fuzzy System for Obtaining Compact and Accurate Fuzzy Classifiers with Transparent Fuzzy Partitions. In: International Conference on Machine Learning and Applications, pp. 89–94. IEEE Press, New York (2009)

[9] Jin, Y., Sendhoff, B., Körner, E.: Simultaneous Generation of Accurate and Interpretable Neural Network Classifiers. In: Yaochu Jin (ed.) Multi-Objective Machine Learning. Studies in Computational Intelligence, vol. 16, pp. 291–312. Springer, Heidelberg (2006)

[10] Markowska-Kaczmar, U., Mularczyk, K.: GA-based Pareto Optimization for Rule Extraction from Neural Networks. In: Yaochu Jin (ed.) Multi-Objective Machine Learning. Studies in Computational Intelligence, vol. 16, pp. 291–312. Springer, Heidelberg (2006)

[11] Tušar, T.: Optimizing accuracy and size of decision trees. In: Sixteenth International Electrotechnical and Computer Science Conference - ERK 2007, vol. B, pp. 81–84. Slovenian Section IEEE, Ljubljana (2007)

[12] Clark, A. R. J., Everson, R. M.: Multi-objective learning of Relevance Vector Machine classifiers with multi-resolution kernels. Pattern Recognition 45(9), 3535–3543 (2012)

[13] Quinlan, J.R.: Learning with Continuous Classes. In: 5th Australian Joint Conference on Artificial Intelligence, pp. 343–348. Morgan Kaufmann, San Francisco (1992)

[14] Kohavi, R.: Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In: 2nd International Conference on Knowledge Discovery and Data Mining, pp. 202–207. AAAI Press, Menlo Park (1996)

[15] Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 523–528. ACM, New York (2003)

[16] Kumar, M.A., Gopal, M.: A Hybrid SVM Based Decision Tree. Pattern Recognition 43(12), 3977–3987 (2010)

[17] Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Hoboken (2009)

[18] Supplementary Material: http://molrc.webs.com/

[19] Deb, k.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE transactions on evolutionary computation vol. 6, no. 2, (2002)

[20] Frank, A., Asuncion, A.: UCI Machine Learning Repository, http://archive.ics.uci.edu/ml

[21] Weka: Collections of Datasets, http://www.cs.waikato.ac.nz/ml/weka/datasets.html

[22] Witten, I.H., Frank, E. Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, Third Edition. Morgan Kaufmann, San Francisco (2011)

[23] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation, 7(2), 117–132 (2003)

[24] Distributed Evolutionary Algorithms in Python (DEAP), https://code.google.com/p/deap/