Understanding evolutionary induction of decision trees: A multi-tree repository approach

Krzysztof Jurczuk
Bialystok University of Technology
Poland
k.jurczuk@pb.edu.pl

Marcin Czajkowski
Bialystok University of Technology
Poland
m.czajkowski@pb.edu.pl

Marek Kretowski
Bialystok University of Technology
Poland
m.kretowski@pb.edu.pl

ABSTRACT

This paper concerns the evolutionary induction of decision trees (DT)s. Noting that many DTs or their parts reappear during the evolution, a multi-tree concept is introduced. A multi-tree integrates all trees with the same test in the root. The differences between trees (below the root) are saved as variants, while the same DT parts are shared. All multi-trees are stored in an external repository that keeps the entire search history. We show that such a repository can achieve not only the goal of duplicated calculation prevention, but also better understanding of the evolutionary DT induction. The road to finding the best individual is easily retrieved, exploitation vs. exploration can be addressed, or similarities between individuals can be measured.

CCS CONCEPTS

• **Computing methodologies → Supervised learning**; **Classification and regression trees**.

KEYWORDS

evolutionary algorithms, data mining, decision trees, fitness reuse

ACM Reference Format:
Krzysztof Jurczuk, Marcin Czajkowski, and Marek Kretowski. 2021. Understanding evolutionary induction of decision trees: A multi-tree repository approach. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459422

1 INTRODUCTION

Decision trees (DTs) are popular techniques in the field of eXplainable Artificial Intelligence. Traditionally, DTs are induced using a top-down greedy search that may lead to sub-optimal solutions. One of the alternatives is an evolutionary induction [1]. It searches for the tree structure and tests globally, which results in less complex DTs with at least comparable prediction performance. However, such an approach is much more computationally complex. The fitness calculations can become severe computational bottleneck, especially for big data [3].

In this paper, we investigate the global DTs induction that follows the typical evolutionary algorithm schema with an unstructured, fixed size population and a generational selection. Individuals in the population are represented and processed in their actual form as binary classification trees with univariate tests in the internal nodes [3]. In each generation, genetic operators (mutation, crossover) differentiate DTs and force the global search. A fitness function is based on a simple weight formula that minimizes the reclassification error and the tree complexity at the same time.

The selection (with elitism), genetic operators and convergence lead to many similarities between individuals in successive populations [2, 4]. Both the mutation and crossover modify more probably only small parts of DTs. The selection replicates better solutions and generates DTs copies. Based on these observations, we challenge to create a special structure (called multi-tree) not only to remember and follow the similarities and differences but also to share the same DT parts and limit redundant calculations.

2 MULTI-TREE REPOSITORY APPROACH

A multi-tree can be perceived as a special structure to group (represent) all DTs with the same test in the root node. The root's test is crucial as all other tests need to be considered in the context of preceding tests. Figure 1 illustrates the integration of various trees into one multi-tree. All three DTs start with the same test, while the differences occur at lower levels of DTs. The root node and other matched tree parts are shared, while the differences are stored as variants locally in the nodes where they occur.

A DT variant can be interpreted as a unique DT (unique structure and tests). Variants are globally numbered inside each multi-tree. If in any node below the root a difference occurs, a new variant is created and noted in multi-tree nodes. Each multi-tree node contains the vector of nodes (tests/leaves) that are represented by it with the corresponding variant IDs (see Figure 2). During the evolution, one DT variant can appear many times.

The repository of multi-trees is built incrementally during the evolution. When the evolution stops, it keeps the entire search history (see Figure 3). Each time when a new individual (DT) appears in the population, it is archived in the repository. First, the repository is searched for a multi-tree with the same test in the root node. If it is found, the new DT is integrated (added) into it. The differences below the root level are searched, and if needed, a subsequent variant is created, while the same DT parts are shared.

If any multi-tree with the same test in the root node is not found, a new multi-tree is created. The initial multi-tree contains only one variant. Multiple statistics are gathered in each multi-tree, e.g., the total number of included trees and variants or the point in the evolution when each DT was generated (iteration and individual

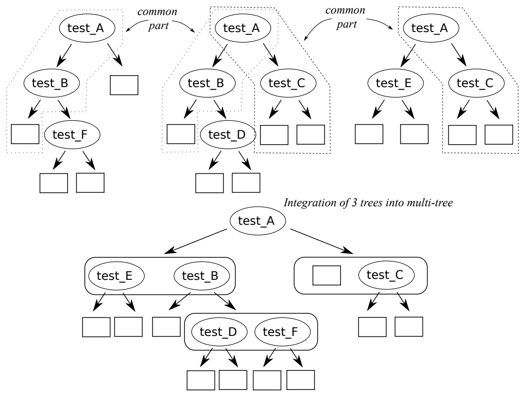

Figure 1: The concept of a multi-tree.

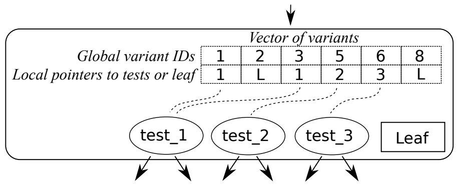
Vector of variants
Global variant IDs 1 2 3 5 6 8
Local pointers to tests or leaf 1 L 1 2 3 L
test_1
test_2
test_3
Leaf

Figure 2: The multi-tree node representing 4 different nodes.

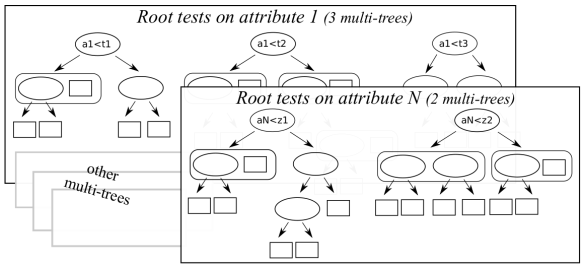
Root tests on attribute 1 (3 multi-trees)
a1<t1
a1<t2
a1<t3
Root tests on attribute N (2 multi-trees)
aN<z1
aN<z2
other multi-trees

Figure 3: The repository of multi-trees.

index inside the population). Moreover, in each multi-tree node, the number of training objects that fall into the node is stored.

3 EXPERIMENTS AND CONCLUSIONS

Preliminary validation was performed on an artificial dataset called *chess3x3* (a classification problem with 2 classes, 2 continuous valued attributes and objects arranged on a 3×3 chessboard) with 1 000 000 objects [3]. A default setup was used with the population size = 64 individuals and the maximal iteration number = 1 000.

The number of individuals (DTs) generated in one run is 64 000. They are represented by only 3 844 multi-trees archived in the repository. The total number of variants (unique DTs) is 30 549. It shows us that for less than 50% of DTs, the fitness calculations are really needed. Moreover, the number of multi-trees indicates how many times does the test in the root node actually needs to be performed for training data (3 844 instead of 64 000 times).

Figure 4 presents the capacity of multi-trees gathered in the repository in a typical run. We see that the number of multi-trees

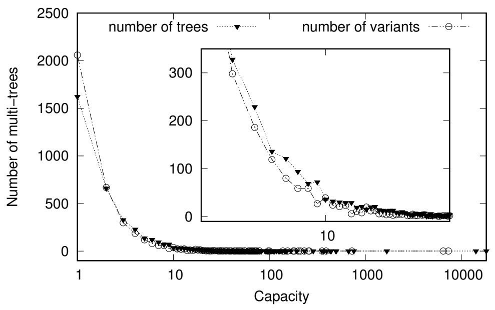

Figure 4: Capacity of multi-trees archived in the repository.

with a single variant equals approximately 2 000, and it decreases exponentially to 1 when the capacity grows. The three most capacious multi-trees include 725, 6 481 and 7 334 variants. Interestingly, the best individual is included in the 'biggest' multi-tree (with 7 334 variants). It is found in the 935-th iteration as a 6262-th variant. The detailed time analysis of successively generated variants (not shown here) allows us to follow the whole path to obtain the best DT. Another application of the repository may be the analysis of exploration and exploitation strength based on the number of multi-trees vs. their capacity (size). More multi-trees indicates more exploration, bigger multi-trees suggest more exploitation.

The similarities between variants are shared inside a multi-tree. Thus, we can easily measure the fitness calculations redundancy by finding the minimal and maximal number of tests (on training data) to be performed. They are calculated based on the number of training objects in each tree or multi-tree node, correspondingly. The maximal number is when for all individuals during the whole evolution all training objects are passed through each tree - from the root till leaves. The minimal number is when only not previously occurring evaluations are performed. For 1 000 000 objects of *chess3x3*, the maximum number of evaluations equals almost 200 trillion, and the redundancy is about 90%! So, the upper bound for the acceleration is $\approx 10\times$ just by eliminating the redundant calculations. In a recent study [2], it was shown that even when a relatively small number of last DTs is archived and a few similarity levels are considered, a substantial acceleration may be provided.

In conclusion, we show that the application of multi-tree concept may reveal the inside of the evolutionary DT induction. Moreover, if it would be directly used during the evolution, the memory complexity would increase but a large speedup would be provided.

Acknowledgments. This work was supported by Bialystok University of Technology under the Grant WZ/WI-IIT/3/2020 founded by Ministry of Science and Higher Education

REFERENCES

[1] Rodrigo C Barros et al. 2012. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on SMC, Part C* 42, 3 (2012), 291–312.
[2] Krzysztof Jurczuk, Marcin Czajkowski, and Marek Kretowski. 2021. Fitness evaluation reuse for accelerating GPU-based evolutionary induction of decision trees. *Int. J. High Perform. Comput. Appl.* 35, 1 (2021), 20–32.
[3] Marek Kretowski. 2019. *Evolutionary Decision Trees in Large-Scale Data Mining.* Springer.
[4] Yang Lou, Shiu Yin Yuen, and Guanrong Chen. 2021. Non-revisiting stochastic search revisited: Results, perspectives, and future directions. *Swarm and Evolutionary Computation* 61 (2021), 100828.