Building Boosted Classification Tree Ensemble with Genetic Programming

Sašo Karakatič
University of Maribor
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, SI-2000 Maribor, Slovenia
saso.karakatic@um.si

Vili Podgorelec
University of Maribor
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, SI-2000 Maribor, Slovenia
vili.podgorelec@um.si

ABSTRACT

Adaptive boosting (AdaBoost) is a method for building classification ensemble, which combines multiple classifiers built in an iterative process of reweighting instances. This method proves to be a very effective classification method, therefore it was the major part of our evolutionary inspired classification algorithm.

In this paper, we introduce the Genetic Programming with AdaBoost (GPAB) which combines the induction of classification trees with genetic programming (GP) and AdaBoost for multiple class problems. Our method GPAB builds the ensemble of classification trees and uses AdaBoost through the evolution to weight instances and individual trees.

To evaluate the potential of the proposed evolutionary method, we made an experiment where we compared the GPAB with Random Forest and AdaBoost on several standard UCI classification benchmarks. The results show that GPAB improves classification accuracy in comparison to other two classifiers.

CCS CONCEPTS

• **Computing methodologies → Supervised learning by classification**; **Genetic programming**; *Boosting*; Classification and regression trees;

KEYWORDS

Classification; Ensemble learning; Genetic programming; Machine learning; Decision trees

ACM Reference Format:
Sašo Karakatič and Vili Podgorelec. 2018. Building Boosted Classification Tree Ensemble with Genetic Programming. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3205651.3205774

1 INTRODUCTION

Boosting for classification problems combines the results of sequentially trained classifiers that improve one another in the final decision. AdaBoost [5] is one among numerous different boosting methods, which represent the current *state-of-the-art* approach to traditional classification problems as was evident from recent papers and data science competitions [2, 3].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5764-7/18/07.
https://doi.org/10.1145/3205651.3205774

On the other hand, genetic programming (GP) is an evolutionary method that builds programs with the process that mimics the evolution and natural selection. In our case, the programs are the decision trees for classification, where nodes represent the rules and leaves represent final decisions on the class of instances [4]. Genetic programming for inducing decision trees for classification has already been researched extensively [1, 7, 10] and several improvements to the basic algorithm have been made, however, the combination with the AdaBoost for classification has still not been introduced or thoroughly researched. Our main contribution with this paper is a novel algorithm called Genetic Programming with AdaBoost (GPAB) that uses already proven state of the art boosting implementation AdaBoost and applies it in the evolution of decision trees which results in better accuracy when compared to Random Forest and AdaBoost ensemble classification algorithms.

2 CLASSIFICATION WITH GENETIC PROGRAMMING AND ADABOOST

Our proposed GPAB algorithm uses standard evolutionary loop for creating classification decision trees [10], with following improvements. After each generation, the algorithm test if *boosting interval* has been reached and if boosting starts. The boosting interval is the number of generations that have to evolve before the boost is applied. Boosting is done with regular steps of AdaBoost.M1 and after every boosting the algorithm continues with the same classification tree population as before and tries to improve already evolved trees. This can work because individuals are reweighted so that the goals and effectively the fitness function changes, allowing trees to evolve further and continue in a new direction.

Boosting algorithm incrementally raises and lowers the significance of instances in the dataset after every boosting interval. This is done with prescribing weights to individuals — higher for more important and lower to less important ones. Reweighted individuals form the new dataset that is used in the new evolution generations in learning another classification model.

Then the evolution, that builds new classification models starts. After each boosting iteration, the classification model is executed, built model is evaluated, its error function ϵ_t is calculated (weighted sum of misclassified instances), and it is given the weight $\alpha_t = \ln{(1 - \epsilon_t / \epsilon_t\ )}$. The evolved classification model is added to ensemble along with its weight α. Next is the process of updating the weights of instances, where the new weight is $w_{i,t} = w_{i,t-1} * (1 - \epsilon_t / \epsilon_t\ )$ and is then normalized so that the sum of all weights equals 1. In the end, after generation limit is reached, evolution is

Algorithm 1 Pseudo code for GPAB.

```
Input: generation limit as $generationLimit$
Input: boosting interval as $boostingInterval$
Input: classification instances as $instances$
Output: evolved and boosted classification trees as $ensemble$
1: population = randomly generated classification trees
2: for all instance in $instances$ do
3:     instance.weight = $\frac{1}{instances.size}$
4: end for
5: generation = 1
6: while generation <= $generationLimit$ do
7:     for all classifier in population do
8:         classifier.fitness = evaluate(classifier)
9:     end for
10:    selection = binaryTournament(population)
11:    newPopulation = crossover(selection)
12:    newPopulation = mutation(newPopulation)
13:    if generation % $boostingInterval$ == 0 then
14:        bestClassifier = getBestByFitness(population)
15:        $\epsilon$ = 1− bestClassifier.accuracyOnTrainSet
16:        bestClassifier.weight = log $\frac{1-\epsilon}{\epsilon}$
17:        ensemble.add(bestClassifier)
18:        newWeight = $\frac{1-\epsilon}{\epsilon}$
19:        for all instance in $instances$ do
20:            predictedClass = bestClassifier.classify(instance)
21:            if predictedClass != instance.actualClass then
22:                instance.weight = instance.weight $*$ newWeight
23:            end if
24:        end for
25:        $instances$ = normalizeWeights($instances$)
26:    end if
27:    generation++
28: end while
29: return ensemble
```

stopped and the ensemble of trees is returned as the final solution. Final classification decision is then calculated as the weighted sum of all classifiers in an ensemble, where the weight of each classifier is α, and the decision with the largest sum is chosen as the final decision. Algorithm 1 shows GPAB pseudo code of whole evolution with boosting process, where *instance* is one classification example.

3 EXPERIMENTAL RESULTS

We conducted an experiment on 11 standard classification basic benchmark datasets from UCI repository to evaluate the performance of the proposed GPAB method. Results were compared to two traditional ensemble classification methods: Random Forest [8] and AdaBoost.M1 [6]. Results are based on 5 folds with 10 repeated runs on each fold for GPAB. GPAB settings are as follows: 100 generations with 250 classification trees with 5 elite member, binary tournament selection, standard random subtree crossover with 100% probability and random mutation of subtrees with 10% probability. Fitness is a minimization function $f = (1 - accuracy) + t/(n \cdot 10)$ and is based on [9], where n is the number of instances and t is the number of nodes in the tree.

GPAB has the boosting interval set to 10 generations interval, so it always produces 10 classifiers for the final ensemble. All of the settings of Random Forest and AdaBoost.M1 were left at default

Table 1: Classification accuracy results of GPAB.

Dataset	GPAB	AdaBoost	Random Forest
autos	0.6976	0.8195	**0.8341**
balance-scale	**0.8832**	0.7712	0.8144
breast-cancer	**0.7866**	0.7342	0.7063
breast-w	**0.9828**	0.9528	0.9685
car	0.8594	0.9201	**0.9387**
credit-a	**0.8826**	0.8391	0.8435
diabetes	**0.7943**	0.7279	0.7566
heart-c	**0.8811**	0.7391	0.8184
iris	**1.0000**	0.9533	0.9467
vehicle	0.6761	**0.7412**	0.7399
Average	**0.8444**	0.8198	0.8367
Mean Rank	**5.49**	3.53	4.51

Weka values. This does not present any issue because the aim of this experiment was to test the validity of GPAB method, and not to find the best possible classification results for every dataset.

The results in the Table 1 shown that GPAB achieved the best average accuracy over all datasets along with the highest rank for the accuracy results and such is a competitive alternative to Random Forest and AdaBoost. Of course, the experiment was done on a small and limited set of benchmarks, so we should still be careful in premature conclusions. Nonetheless, the results of GPAB show it to have a high potential and could be used in the future research endeavors and applied to various classification problems.

ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0057).

REFERENCES

[1] Rodrigo Coelho Barros, Marcio Porto Basgalupp, ACPLF De Carvalho, and Alex Alves Freitas. 2012. A survey of evolutionary algorithms for decision-tree induction. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42, 3 (2012), 291–312.
[2] Leo Breiman. 1996. *Bias, Variance, and Arcing Classifiers.* Technical Report. Statistics Department, University of California at Berkeley.
[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, New York, NY, USA, 785–794.
[4] P.G. Espejo, S. Ventura, and F. Herrera. 2010. A Survey on the Application of Genetic Programming to Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40, 2 (March 2010), 121–144.
[5] Yoav Freund and Robert E. Schapire. 1996. Experiments with a New Boosting Algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning (ICML'96).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 148–156.
[6] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics* 28, 2 (2000), 337–407.
[7] Muhammad Iqbal, Bing Xue, Harith Al-Sahaf, and Mengjie Zhang. 2017. Cross-domain reuse of extracted knowledge in genetic programming for image classification. *IEEE Transactions on Evolutionary Computation* 21, 4 (2017), 569–587.
[8] Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.
[9] V. Podgorelec, S. Karakatič, R. C. Barros, and M. P. Basgalupp. 2015. Evolving balanced decision trees with a multi-population genetic algorithm. In *2015 IEEE Congress on Evolutionary Computation (CEC).* IEEE, Piscataway, NJ, 54–61.
[10] Vili Podgorelec, Matej Šprogar, and Sandi Pohorec. 2013. Evolutionary design of decision trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 3, 2 (2013), 63–82.