Ensemble Genetic Programming

Nuno M. Rodrigues(✉), João E. Batista, and Sara Silva

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Lisbon, Portugal
{nmrodrigues,jebatista,sara}@fc.ul.pt

Abstract. Ensemble learning is a powerful paradigm that has been used in the top state-of-the-art machine learning methods like Random Forests and XGBoost. Inspired by the success of such methods, we have developed a new Genetic Programming method called Ensemble GP. The evolutionary cycle of Ensemble GP follows the same steps as other Genetic Programming systems, but with differences in the population structure, fitness evaluation and genetic operators. We have tested this method on eight binary classification problems, achieving results significantly better than standard GP, with much smaller models. Although other methods like M3GP and XGBoost were the best overall, Ensemble GP was able to achieve exceptionally good generalization results on a particularly hard problem where none of the other methods was able to succeed.

Keywords: Genetic Programming · Ensemble learning · Binary classification · Machine Learning

1 Introduction

Genetic Programming (GP) [25] is one of the most proficient Machine Learning (ML) methods. It is capable of addressing multiple tasks such as classification and regression, using a variety of techniques from the most classical [17] to the most recent, like the geometric semantic approaches [28] and the cluster-based multiclass classification [22].

Ensemble learning [9] is a powerful ML paradigm where multiple models are induced and their predicted outputs are combined in order to obtain predictions that are more accurate than the individual ones. Some of the most successful ML methods are based on ensemble learning, like Random Forests (RF) [4] and XGBoost (XG) [7]. On the other hand, their performance may vary substantially depending on the setting of some crucial parameters, like the number of trees and their maximum depth, which in turn depend on the properties of each dataset.

Inspired by the success of such methods, and motivated by the need to automatically find the right settings for these parameters, we have developed a new GP method called Ensemble GP (eGP). The evolutionary cycle of eGP follows the same steps as other GP systems, but with differences in the population structure, fitness evaluation and genetic operators. In particular, the population

T. Hu et al. (Eds.): EuroGP 2020, LNCS 12101, pp. 151–166, 2020.
https://doi.org/10.1007/978-3-030-44094-7_10

is composed of two subpopulations, trees and forests, where each subpopulation uses its own fitness function and genetic operators. The approach can be described as co-evolutionary, cooperative and compositional, and involves subsampling of both observations and features.

The rest of the paper is organized as follows. Section 2 describes related work, while Sect. 3 provides the details of the eGP method. Section 4 specifies the experimental setup, and Sects. 5 and 6 report and discuss the results obtained. Finally, Sect. 7 contains the conclusions and future work.

2 Related Work

Evolutionary computation and other bio-inspired methods have been linked to ensemble learning from early on (see [11] and references therein). An obvious way to build ensembles is to combine different individuals of a population, whether they are GP individuals (*e.g.* [31]) or other types, like neural networks (review in [15]). Many other types of ensembles have been built using evolutionary and other bio-inspired methods, like ensembles of clustering algorithms [8], Decision Trees [5], Support Vector Machines [1], or a mix of different types [10]. Diversity is important among ensemble members, and multiobjective evolutionary approaches have been often used to address this issue (*e.g.* [2,6,24] and references therein).

A multitude of publications focus on single specific aspects of ensemble learning, like selecting and combining the members of the ensemble (*e.g.* [10] and references therein), or evolving the functions that combine the different members (*e.g.* [10,16,19]). Others focus on building complete ensembles from scratch, but even if we limit ourselves to the ones that use GP exclusively (*e.g.* [3,13,29]), we find a large diversity of designs, goals and scales of application. A systematic review of this extremely vast and diverse literature is much needed in both evolutionary and ensemble learning communities.

3 Ensemble GP

Now, we describe the method we call ensemble GP (eGP) with all the variants we implemented and tested. The evolutionary cycle of eGP follows the same steps as other GP systems, but with differences in the population structure, fitness evaluation and genetic operators. In particular, the population is composed of two subpopulations, where each subpopulation uses its own fitness function and genetic operators. The approach can be described as co-evolutionary, cooperative and compositional, and involves subsampling of both observations and features. Algorithm 1 describes the main steps of eGP.

Before describing the details regarding the population, fitness and genetic operators of eGP, we briefly describe a GP system called M3GP [22] (Multidimensional Multiclass GP with Multidimensional Populations), not only because it is one of the baselines in our experiments, but also because some elements of eGP are highly inspired in M3GP.

Algorithm 1. eGP

```
procedure EGP(Dataset(D_s), n_t, n_f)
    Split D_s into training, testing and sub samples Φ
    T_list ← Generate Trees(Φ, n_t)
    F_list ← Generate Forests(n_f)
    while generation(g) < max generations do
        T_parents ← Selection(T_list)
        F_parents ← Selection(F_list)
        T_offspring ← Breeding(T_parents)
        F_offspring ← Breeding(F_parents)
        F_list ← Prune(F_offspring)          ▷ Prune only the best forest
        g++
    end while
end procedure
```

3.1 M3GP

In terms of representation of the solutions, the main difference between M3GP and standard tree-based GP is the number of trees that are part of the same individual. While a standard GP individual is a single tree, a M3GP individual may be composed of several trees, called dimensions. Originally developed for performing multiclass classification [14,22], M3GP evolves each individual as a set of hyperfeatures, each one represented by a different tree/dimension. After remapping the input data into this new multidimensional feature space, it calculates the accuracy by forming clusters based on the data labels and classifying each observation as the class of the closest centroid according to the Mahalanobis distance. M3GP has also been used for evolving hyperfeatures for regression [23] and for classification in other GP systems [18].

Starting with only one tree/dimension per individual, M3GP uses standard subtree crossover and mutation between individuals, and three other operators designed for removing a tree/dimension from an individual, adding a randomly created tree/dimension to an individual, and swapping trees/dimensions between individuals. Additionally, a pruning operator is applied to the best individual of each generation, removing the trees/dimensions that do not improve its accuracy.

3.2 eGP Population Structure

The population is composed of two types of individuals: trees and forests. A tree is not the output model, but only a part of it. The output model is a forest, built as an ensemble of trees. Each tree may be part of many different forests, and some trees may be part of none.

Trees have the same structure as those used in standard GP, but instead of having access to all the observations and features of the training dataset, each individual only sees a subset of observations, and in many cases also a subset of features. Different variants of eGP use different sampling options: (1) 60% of all observations, all features included; (2) between one and all observations,

one to all features included, these numbers being randomly chosen before each sampling. In both options, the sampling is done uniformly without replacement, and repeated whenever a new subset of training data is required for allocating to a new tree.

Forests have the same structure as the M3GP individuals, with each dimension being a tree from the subpopulation of trees.

3.3 eGP Fitness Functions

The subpopulation of trees uses a standard fitness function based on the error between expected and predicted outputs, like the Root Mean Squared Error (RMSE). In classification problems, the class labels are interpreted as the numeric expected outputs. The fitness of each tree is calculated using only the subset of observations allowed for this tree.

The subpopulation of forests uses a fitness function based on the accuracy obtained on all the observations of the training set. Each forest gathers, for each observation, a vote (on a class) from each of the trees that compose its ensemble. This vote is obtained by adopting the class label that is closer to the predicted output. The votes from the different trees of the ensemble can be combined by normal majority voting or by weighted voting.

In normal voting, for each observation the class that receives more votes wins, and ties are solved by randomly choosing one of the classes. In weighted voting (Algorithm 2), for each observation a certainty value is calculated for each class prediction of each tree, based on the vector of predicted values by all the trees of the ensemble (1). The sum of certainty values for each class is then calculated, and divided by the sum of certainty values for both classes. The class with highest results is chosen as the prediction.

We chose to use L2 normalization (2) for consistency with the cosine similarity used for the eCrossover (described next), which also uses L2. Other normalization methods were considered. Min-Max was discarded due to its inability for dealing with outliers; Z-Score was discarded because the resulting array was not contained in the $[0, 1]$ range.

$$certainty = 1 - l_2(X), X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{1}$$

$$l_2\ normalization = \sqrt{\sum_{k=1}^{n} |x_k|^2} \tag{2}$$

3.4 eGP Genetic Operators

The trees and forests of eGP use different genetic operators. Trees use what can be described as protected versions of the standard subtree crossover and mutation, designated here as eCrossover and eMutation, respectively. The protection

Algorithm 2. Weighted Voting

```
procedure WEIGHTED_VOTING(predictions, certainties)
    votes ← [ ]
    for row in predictions do
        zeros, ones ← 0
        for col in certainties do
            if predictions[row][col] == 1
                ones+ = certainties[row][col]
            else
                zeros+ = certainties[row][col]
        end for
        votes.append(0 if zeros/(zeros + ones) ≥ ones/(zeros + ones) else 1)
    end for
end procedure
```

is needed when parent trees are not allowed to see all the features due to feature sampling (see Sect. 3.2). In this case, the offspring must inherit feature restrictions from their parents, otherwise after a number of generations all the trees will be using all the features. Without feature sampling, these operators behave the same as the standard ones.

eMutation simply has to ensure that the new subtree created to replace a random branch of the parent is restricted to the same subset of features as the parent. eCrossover must guarantee that each swapped branch is also restricted to the subset of features inherited by the receiving offspring. Each of the two offspring inherits from one of its two parents. Instead of relying on a careful choice of compatible couples and branches to swap, eCrossover relies on a repair procedure that replaces features on the received branches whenever these features are not allowed by the inherited restrictions (Algorithm 3). Each illegal feature is compared to all the legal ones on the complete training set, using the cosine similarity measure (3). The chosen replacement is the most similar feature to the one that was removed. Unlike the euclidean distance, the cosine similarity can compare and recognize two vectors of similar meaning even if they have very different magnitudes.

$$S(X,Y) = \frac{\sum_{k=1}^{n} |x_k||y_k|}{\sqrt{\sum_{k=1}^{n} |x_k|^2}\sqrt{\sum_{k=1}^{n} |y_k|^2}} \tag{3}$$

Regarding the subpopulation of forests, it uses the same genetic operators as M3GP, namely two mutation operators to add and remove trees from the ensemble, and one crossover operator to swap trees between different ensembles.

4 Experimental Setup

This section describes our experimental setup for the eGP methods, comparing them against two baselines, standard GP and M3GP, and two state-of-the-art

Algorithm 3. eCrossover

procedure SUBTREE CROSSOVER($parent_1, parent_2$)
  $cp_1, cp_2 \leftarrow$ *choose crossover points* ▷ crossover point 1 and 2
  $refact\ tree(parent_1, parent_2, cp_1, cp_2, bag_1, bag_2, Data_{Training})$
end procedure

procedure REFACT TREE($parent_1, parent_2, cp_1, cp_2, bag_1, bag_2, Data_{Training}$)
  $parent_1, parent_2 \leftarrow swap\ branches(cp_1, cp_2)$
  $fix\ terminals(p_1, bag_1, bag_2, Data_{Training})$
  $fix\ terminals(p_2, bag_2, bag_1, Data_{Training})$
end procedure

classifiers, Random Forests (RF) and XGBoost (XG). Six different variants of eGP were tested, and the results were analysed in terms of training and test accuracy, number of trees and number of nodes of the final solutions. When comparing accuracy, statistical significance is determined using the non-parametric Kruskal-Wallis test at $p < 0.01$. Next, we describe all the 10 methods tested, their main parameter settings, and the eight datasets used for obtaining the reported results.

4.1 Methods

Table 1 contains the acronyms and descriptions of all the methods used, and will serve as a memory aid for the remainder of this paper. The six eGP variants are eGP-N and eGP-W (normal and weighted voting with sampling of features and observations); eGP-N5 and eGP-W5 (same as previous but with populations of 500 trees and 500 forests, instead of 250 each); eGPn and eGPw (same as eGP-N and eGP-W but without feature sampling).

Table 1. Acronyms and descriptions of the methods

GP	Standard Genetic Programming
M3GP	Multidimensional Multiclass GP with Multidimensional Populations
eGP-N	Ensemble GP, feature sampling, normal voting
eGP-W	Ensemble GP, feature sampling, weighted voting
eGP-N5	Ensemble GP, feature sampling, normal voting, larger population
eGP-W5	Ensemble GP, feature sampling, weighted voting, larger population
eGPn	Ensemble GP, no feature sampling, normal voting
eGPw	Ensemble GP, no feature sampling, weighted voting
RF	Random Forests
XG	XGBoost

Table 2. Main parameter settings

Runs	30
Generations	100
Population size	GP/M3GP = 500, eGP = {250 + 250, 500 + 500}
Function set	$\{+, -, \times, /, \log, \sqrt{}\}$ (protected)
Fitness	GP = RMSE, M3GP/eGP = Accuracy
Selection	Tournament size 5 (GP/M3GP = Double Tournament)
Crossover/Mutation	GP = 0.95/0.05, M3GP/eGP = 0.5/0.5
Number of estimators	{50, 100, 150, 200}
Maximum depth	{2, 4, 6, 8}
Impurity measure	RF = {Gini, Entropy}

4.2 Parameters

Table 2 summarizes the main parameters used in the GP-based methods and in the RF and XG methods. Each experiment is performed 30 times, with each run using a different partition of the dataset in 70% training and 30% test. The GP-based methods run for 100 generations. GP and M3GP use populations of 500 individuals, while eGP initializes each subpopulation with 250 (or 500) individuals, for a total of 500 (or 1000) trees + forests. Trees are initialized using Ramped Half-and-Half, as suggested by Koza [17], while forests are initialized in a similar fashion to M3GP, with only one tree per forest [22]. The arithmetic operators of the function set are protected in the following way: when dividing a value by zero, we return the numerator; when trying to square root or logarithm a negative number, we return the number untouched. Therefore, the protection is to ignore the presence of the operator whenever it raises an exception. No constants are used. The fitness guiding the evolution is the RMSE in GP, and the accuracy in M3GP and eGP. In order to obtain the accuracy from GP, the predicted outputs are transformed into the closest numeric class labels. Selection for breeding is made with Double Tournament [21] in GP and M3GP, and regular tournament in eGP, size 5. Regarding genetic operators, the crossover/mutation probabilities are 0.95/0.05 for GP, and 0.5/0.5 for both M3GP and eGP. This means choosing between crossover and mutation with equal probability, but for M3GP and eGP forests the specific type of crossover or mutation must then be chosen, also with equal probability. Elitism guarantees that the best parent is copied into the new population.

Regarding RF and XG (last three rows of the table), both were 10-fold cross-validated for number of estimators and maximum depth, and RF was also cross-validated for the impurity criterion.

4.3 Datasets

Table 3 describes the main characteristics of the datasets used in our experiments. We have selected eight problems from various domains, all being binary classification tasks, with a different number of features and observations.

Table 3. Number of features, observations and negative/positive ratio on each dataset.

Datasets	BCW	BRAZIL	GAMETES	HEART	IONO	PARKS	PPI	SONAR
Features	11	8	1000	13	33	23	3	61
Observations	683	4872	1600	270	351	195	31320	208
Neg/Pos Ratio	35/65	42/58	50/50	45/55	65/35	75/25	52/48	46/54

BCW, HEART, IONO, PARKS. *Breast Cancer Wisconsin, Heart Disease, Ionosphere* and *Parkinsons* are datasets included in the UCI ML repository [20].

BRAZIL. *Brazil* is a dataset for detecting burned areas in satellite imagery, containing the radiance values of a set of pixels from a Landsat 8 OLI image over Brazil, and corrected unburned/burned labels [26].

GAMETES. *GAMETES_Epistasis_2-Way_1000atts_0.4H_EDM-1_EDM-1_1* is a simulated Genome-Wide Association Studies (GWAS) dataset generated using the GAMETES tool [18], available in OpenML [12].

PPI. *GRID/HPRD-unbal-HS* is a dataset built from a Protein-Protein Interaction benchmark of the human species [30], containing the Resnik_{Max} semantic similarity measure between each pair of proteins on three different semantic aspects [27].

SONAR. *sonar.all-data* is a dataset for binary classification of sonar returns, available in Kaggle [32].

5 Results

Figures 1, 2, 3, 4, 5, 6, 7 and 8 show boxplots of the training and test accuracy obtained by all the methods on all the problems. For each problem there are two whiskered boxes, the left one for training and the right one for test. On the BRAZIL problem, five outliers were removed for visualization purposes, two on training (90.97% and 67.92%, both for GP) and three on test (90.70% and 68.95% for GP, 77.29% for eGP-N5).

Between the two baselines, as expected M3GP is better than standard GP, achieving significantly better training accuracy on all eight problems, and also significantly better test accuracy on five of them (BRAZIL, IONO, PARKS, PPI and SONAR). In fact, in all pairwise comparisons with the other methods in all the problems, standard GP is significantly worse in 96% of the cases on training, and 46% on test. The only exception where it performs significantly better is on the HEART problem, against RF on the test data.

Regarding the two proposed methods eGP-N and eGP-W, a comparison between them reveals that the weighted voting (eGP-W) does not seem to improve performance over the normal voting (eGP-N), as the weighted voting resulted in one significantly worse training accuracy in the PARKS problem

(and another borderline worse in IONO), all other results being equal to the ones of normal voting. Also between eGP-N5 and eGP-W5 the weighted voting resulted in one significantly worse training accuracy in the IONO problem, all other results showing no significant differences.

Increasing the population size from 250 to 500 proved to be only marginally beneficial, more to weighted than to normal voting. eGP-N5 achieved significantly better results than eGP-N on four problems (GAMETES, IONO, PARKS and SONAR) on training, and none on test, all other results being statistically equal. eGP-W5 was significantly better than eGP-W on five problems (BCW, HEART, IONO, PARKS and SONAR) on training, and on one problem (IONO) on test, all other results equal.

Regarding the eGP methods without feature sampling (eGPn and eGPw), in several cases they revealed to be significantly better than their feature sampling counterparts (eGP-N and eGP-W), more often on training but also on two test cases, on problems IONO and PARKS. Even when compared to the 500 individual counterparts, they were often better on training and never worse on test. The weighted voting did not improve or worsen the obtained accuracy.

When comparing the eGP methods with the M3GP baseline, we realize that on training accuracy M3GP is better than all eGP methods on four problems (GAMETES, HEART, PARKS and SONAR), worse than all eGP methods on two problems (BCW and PPI), and on the remaining problems it is better or equal to most eGP methods, except one case where it is worse (than eGPw, on BRAZIL). On test accuracy M3GP is better than all eGP methods on four problems (IONO, PARKS, PPI and SONAR), statistically the same as all eGP methods on three problems (BCW, GAMETES, HEART), and on the remaining problem M3GP is better than all eGP feature sampling methods and statistically the same as eGPn and eGPw.

When comparing the eGP methods with the state-of-the-art RF and XG, on training both are significantly better than practically all eGP methods on all problems (except SONAR, where RF is significantly worse than all except eGP-N and eGP-W). On test accuracy, on two problems (BCW and GAMETES) there are few significant differences (XG is better than eGP-N and eGP-W), on two other problems (IONO, PARKS) both RF and XG are better than all eGP methods, and on the remaining problems RF is either the same (BRAZIL and PPI), worse (HEART) or better (SONAR) in most cases, while XG is better in all except a few cases (eGPn and eGPw on BRAZIL, eGP-N on HEART, with no significant differences).

6 Discussion

In order to better understand how each of the 10 methods scored relatively to each other, we have counted how many significantly better results each one obtained among all $72 + 72 = 144$ (training + test) pairwise comparisons on all problems. Table 4 shows the counting (totals are the sum of all problems) and ranks the methods according to the test totals (training + test in case of tie).

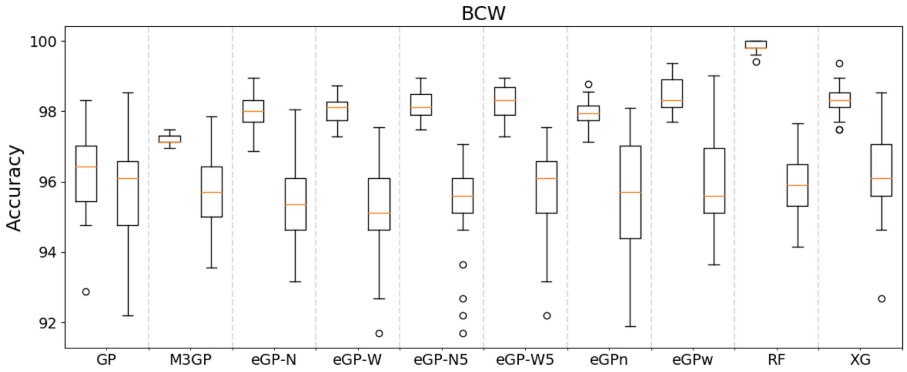

Fig. 1. Boxplot for the training (left) and test (right) accuracy of each method in the BCW dataset.

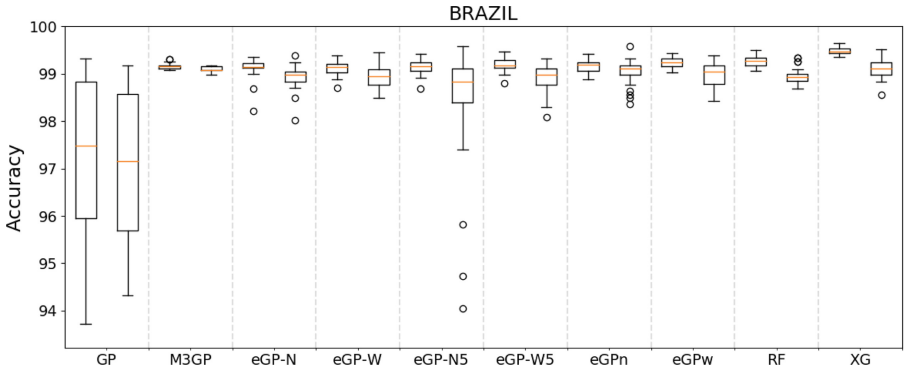

Fig. 2. Boxplot for the training (left) and test (right) accuracy of each method in the BRAZIL dataset. Outliers removed for visualization purposes: on training, 90.97% and 67.92%, both for GP; on test, 90.70% and 68.95% for GP, and 77.29% for eGP-N5.

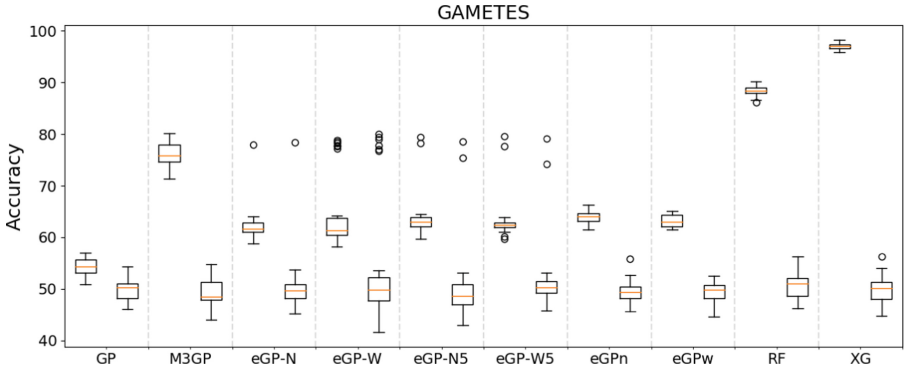

Fig. 3. Boxplot for the training (left) and test (right) accuracy of each method in the GAMETES dataset.

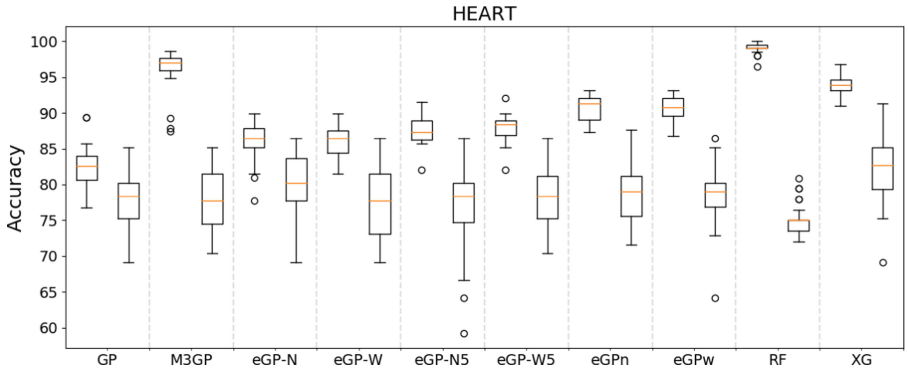

Fig. 4. Boxplot for the training (left) and test (right) accuracy of each method in the HEART dataset.

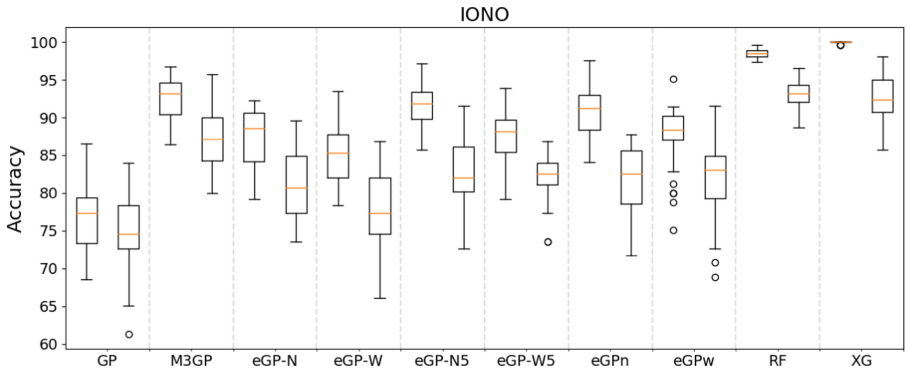

Fig. 5. Boxplot for the training (left) and test (right) accuracy of each method in the IONO dataset.

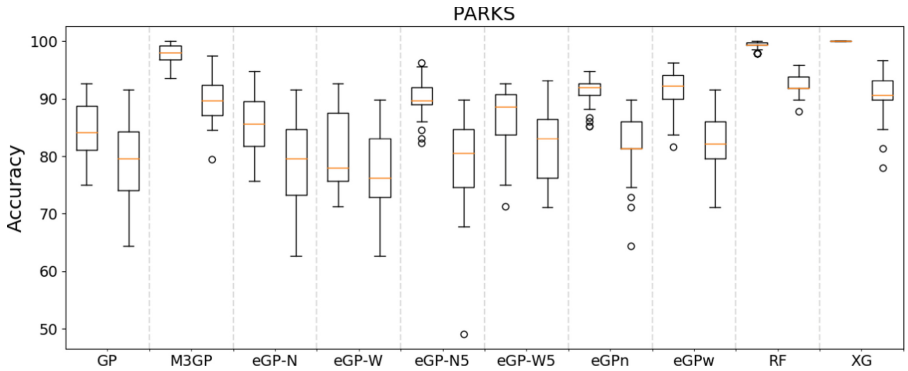

Fig. 6. Boxplot for the training (left) and test (right) accuracy of each method in the PARKS dataset.

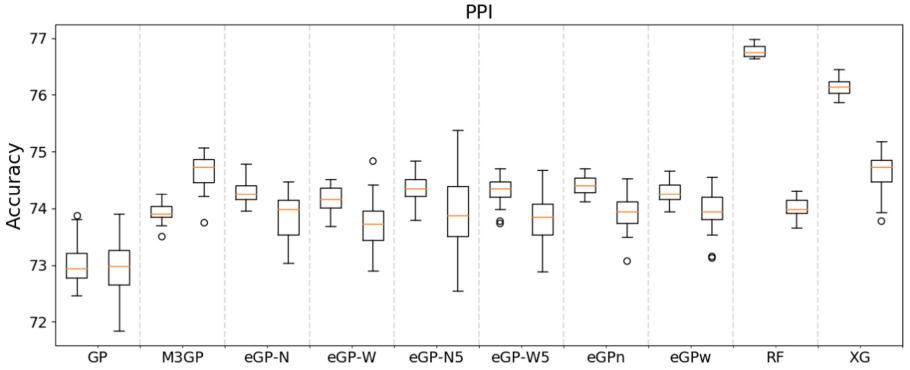

Fig. 7. Boxplot for the training (left) and test (right) accuracy of each method in the PPI dataset.

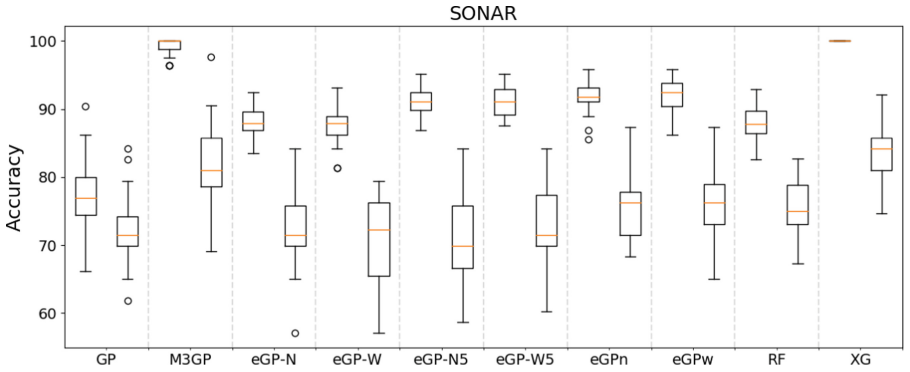

Fig. 8. Boxplot for the training (left) and test (right) accuracy of each method in the SONAR dataset.

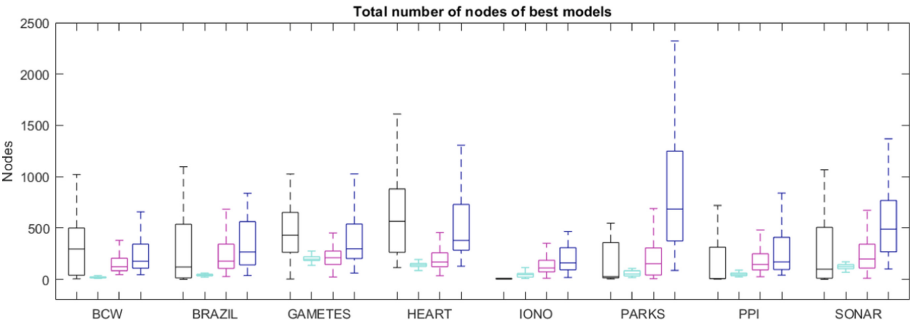

Fig. 9. Number of nodes of final models. For each problem, the four boxes are: GP (black), M3GP (cyan), eGP-N + eGP-W + eGP-N5 + eGP-W5 all together (magenta), and eGPn + eGPw together (blue). All outliers removed for visualization purposes. (Color figure online)

Table 4. Counting of how many significantly better results each method obtained among all pairwise comparisons. The totals are the sum for all problems. Order of the problems: BCW, BRAZIL, GAMETES, HEART, IONO, PARKS, PPI, SONAR.

Method	Training	Test
XG	$3+9+9+7+9+9+8+9=\mathbf{63}$	$2+6+0+8+8+7+8+8=\mathbf{47}$
M3GP	$1+1+7+8+5+7+1+8=\mathbf{38}$	$0+6+0+1+7+7+8+8=\mathbf{37}$
RF	$9+6+8+9+8+8+9+1=\mathbf{58}$	$0+1+0+0+8+8+2+4=\mathbf{23}$
eGPw	$5+5+3+5+1+4+2+4=\mathbf{29}$	$0+1+0+1+2+1+1+0=\mathbf{6}$
eGPn	$2+1+5+5+5+4+3+4=\mathbf{29}$	$0+1+0+1+2+0+1+0=\mathbf{5}$
eGP-N5	$2+1+2+2+5+3+3+4=\mathbf{22}$	$0+1+0+1+2+0+1+0=\mathbf{5}$
eGP-W5	$4+1+1+3+2+1+2+4=\mathbf{18}$	$0+1+0+1+2+0+1+0=\mathbf{5}$
eGP-N	$2+1+1+1+2+1+2+1=\mathbf{11}$	$0+1+0+1+1+0+1+0=\mathbf{4}$
eGP-W	$2+1+1+1+1+0+2+1=\mathbf{9}$	$0+1+0+0+0+0+1+0=\mathbf{2}$
GP	$0+0+0+0+0+0+0+0=\mathbf{0}$	$0+0+0+1+0+0+0+0=\mathbf{1}$

These numbers confirm what had already been observed in the boxplots: (1) the eGP methods, although better than standard GP, were not able to outperform M3GP or the state-of-the-art RF and XG, (2) the eGP variants without feature sampling (eGPn and eGPw) are better than the other eGP methods, and (3) normal voting is generally better than weighted voting.

Not being an ensemble method, it is noteworthy how well M3GP scored, better than RF and all other methods except XG. It is also important to emphasize that the only methods where the running parameters were tuned by cross-validation were RF and XG (see Sect. 4.2). Therefore, we have no doubt regarding the superiority of M3GP over RF, and raise the question of whether it could surpass XG had its parameters also been tuned.

Regarding the ranking of the eGP methods, it is possible that feature sampling is not necessary for a GP ensemble, due to the feature selection that most GP trees naturally do. We must also consider that the feature replacement performed by eCrossover may have highly destructive effects on the fitness of the offspring. Another thing to consider is the possible inadequacy of our certainty measure to weight the voting of the ensembles.

Although the results of the eGP methods seem disappointing, they are no doubt a viable alternative to standard GP, not only in terms of fitness but also in terms of the size of the evolved models. Figure 9 shows the total number of nodes of the best models found by the GP-based methods, grouped in four sets: (1) GP only (black); (2) M3GP only (cyan); (3) eGP methods with feature sampling (eGP-N + eGP-W + eGP-N5 + eGP-W5 all together, magenta); (4) eGP methods without feature sampling (eGPn + eGPw together, blue), results per problem.

The variants with feature sampling exhibit values with much less dispersion than the ones produced by GP (except on the IONO problem), and significantly

lower on three problems (BCW, GAMETES, HEART). This result becomes even more important when we recall that GP used Double Tournament for bloat control (see Sect. 4.2) and is composed of a single tree, while eGP did not use any bloat control and is composed of an ensemble of trees. M3GP produced the smallest solutions of all GP-based methods, however it also used Double Tournament. Regarding the number of trees that form the evolved ensembles (not shown), the eGP methods revealed a remarkable consistency among the different problems, with different runs always using between 2($\pm$1) and 13($\pm$2) trees on the best forest. This is in sharp contrast to the number of dimensions used by M3GP, with some problems using as few as 1–4 (BCW) and others using as many as 11–24 (SONAR), 13–30 (HEART) and 20–36 (GAMETES).

The GAMETES problem posed the largest difficulties to all the methods, but special attention must be given to the results obtained by some of the eGP methods, precisely the ones that scored worse in general: eGP-N, eGP-W, eGP-N5, eGP-W5. Looking back at Fig. 3, we observe a large amount of outliers of much higher accuracy than normal. On the test data, these are by far the best results achieved, similar to the ones reported in [18], and only the four mentioned eGP methods were able to achieve them. Although out of the scope of this paper, these methods were indeed the only ones able to find, among the 1000 features of this problem, the right combinations that allowed such a big "jump" in accuracy. Therefore, they deserve more investigation, despite their apparent modest performance.

7 Conclusions and Future Work

We have developed a new GP method called Ensemble GP (eGP) and tested it on eight binary classification problems from various domains, with a different number of features and observations. Different variants of eGP were compared to standard GP and M3GP baselines, and to the Random Forests and XGBoost state-of-the-art methods. The results show that eGP consistently evolves smaller and more accurate models than standard GP. M3GP and XGBoost were the best methods overall, but on a particularly hard problem the eGP variants were able to reach exceptionally good generalization results, way above all the other methods.

As future work, we will investigate ways to improve eGP in different fronts, making it more competitive with M3GP and XGBoost while maintaining the characteristics that granted its current success. For example, bloat control and some parameter tuning are two elements that other methods are benefiting from, and that we will incorporate also in eGP. Different voting schemes may also prove beneficial, as well as alternative ways to sample features and observations. Additionally, we will also work towards extending eGP in order to give it the ability to address also regression problems and multiclass classification problems.

Acknowledgement. This work was partially supported by FCT through funding of LASIGE Research Unit UIDB/00408/2020 and projects PTDC/CCI-INF/29168/2017, PTDC/CCI-CIF/29877/2017, DSAIPA/DS/0022/2018, PTDC/ASP-PLA/28726/2017 and PTDC/CTA-AMB/30056/2017.

References

1. Okayama,de Araújo Padilha, C.A., Barone, D.A.C., Neto, A.D.D.: A multi-level approach using genetic algorithms in an ensemble of least squares support vector machines. Knowl.-Based Syst. **106**, 85–95 (2016). https://doi.org/10.1016/j.knosys.2016.05.033
2. Bhowan, U., Johnston, M., Zhang, M., Yao, X.: Evolving diverse ensembles using genetic programming for classification with unbalanced data. IEEE Trans. Evol. Comput. **17**(3), 368–386 (2013). https://doi.org/10.1109/TEVC.2012.2199119
3. Brameier, M., Banzhaf, W.: Evolving teams of predictors with linear genetic programming. Genet. Program Evolvable Mach. **2**(4), 381–407 (2001). https://doi.org/10.1023/A:1012978805372
4. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
5. Cantu-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. IEEE Trans. Evol. Comput. **7**(1), 54–68 (2003)
6. Chandra, A., Yao, X.: Ensemble learning using multi-objective evolutionary algorithms. J. Math. Model. Algorithms **5**(4), 417–445 (2006). https://doi.org/10.1007/s10852-005-9020-3
7. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. ArXiv abs/1603.02754 (2016)
8. Coelho, A.L.V., Fernandes, E., Faceli, K.: Multi-objective design of hierarchical consensus functions for clustering ensembles via genetic programming. Decis. Support Syst. **51**(4), 794–809 (2011). https://doi.org/10.1016/j.dss.2011.01.014
9. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45014-9_1
10. Escalante, H.J., Acosta-Mendoza, N., Morales-Reyes, A., Gago-Alonso, A.: Genetic programming of heterogeneous ensembles for classification. In: Ruiz-Shulcloper, J., Sanniti di Baja, G. (eds.) CIARP 2013. LNCS, vol. 8258, pp. 9–16. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41822-8_2
11. Gagné, C., Sebag, M., Schoenauer, M., Tomassini, M.: Ensemble learning for free with evolutionary algorithms? In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007), pp. 1782–1789. ACM, New York (2007). https://doi.org/10.1145/1276958.1277317
12. Gijsbers, P.: Gametes_epistasis_2-way_1000atts_0.4h_edm-1_edm-1_1 (2017). https://www.openml.org/d/40645
13. Iba, H.: Bagging, boosting, and bloating in genetic programming. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation (GECCO 1999), vol. 2, pp. 1053–1060. Morgan Kaufmann Publishers Inc., San Francisco (1999). http://dl.acm.org/citation.cfm?id=2934046.2934063
14. Ingalalli, V., Silva, S., Castelli, M., Vanneschi, L.: A multi-dimensional genetic programming approach for multi-class classification problems. In: Nicolau, M., et al. (eds.) EuroGP 2014. LNCS, vol. 8599, pp. 48–60. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44303-3_5

15. Islam, M.M., Yao, X.: Evolving artificial neural network ensembles. IEEE Comput. Intell. Mag. **3**, 31–42 (2008)
16. Johansson, U., Lofstrom, T., Konig, R., Niklasson, L.: Building neural network ensembles using genetic programming. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 1260–1265, July 2006. https://doi.org/10.1109/IJCNN.2006.246836
17. Koza, J.R.: Genetic Programming (1992)
18. La Cava, W., Silva, S., Vanneschi, L., Spector, L., Moore, J.: Genetic programming representations for multi-dimensional feature learning in biomedical classification. In: Squillero, G., Sim, K. (eds.) EvoApplications 2017. LNCS, vol. 10199, pp. 158–173. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55849-3_11
19. Langdon, W.B., Buxton, B.F.: Genetic programming for combining classifiers. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 66–73. Morgan Kaufmann (2001)
20. Lichman, M.: UCI Machine Learning Repository (2013). https://archive.ics.uci.edu/ml/index.php
21. Luke, S., Panait, L.: Fighting bloat with nonparametric parsimony pressure. In: Guervós, J.J.M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., Fernández-Villacañas, J.-L. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 411–421. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45712-7_40
22. Muñoz, L., Silva, S., Trujillo, L.: M3GP – multiclass classification with GP. In: Machado, P., et al. (eds.) EuroGP 2015. LNCS, vol. 9025, pp. 78–91. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16501-1_7
23. Muñoz, L., Trujillo, L., Silva, S., Castelli, M., Vanneschi, L.: Evolving multidimensional transformations for symbolic regression with M3GP. Memetic Comput. **11**(2), 111–126 (2018). https://doi.org/10.1007/s12293-018-0274-5
24. de Oliveira, D.F., Canuto, A.M.P., de Souto, M.C.P.: Use of multi-objective genetic algorithms to investigate the diversity/accuracy dilemma in heterogeneous ensembles. In: 2009 International Joint Conference on Neural Networks, pp. 2339–2346 (2009)
25. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd., Essex (2008)
26. Silva, S., Vanneschi, L., Cabral, A.I., Vasconcelos, M.J.: A semi-supervised genetic programming method for dealing with noisy labels and hidden overfitting. Swarm Evol. Comput. **39**, 323–338 (2018). https://doi.org/10.1016/j.swevo.2017.11.003
27. Sousa, R.T., Silva, S., Pesquita, C.: Evolving knowledge graph similarity for supervised learning in complex biomedical domains. BMC Bioinform. **21**, 6 (2020). https://doi.org/10.1186/s12859-019-3296-1
28. Vanneschi, L.: An introduction to geometric semantic genetic programming. In: Schütze, O., Trujillo, L., Legrand, P., Maldonado, Y. (eds.) NEO 2015. SCI, vol. 663, pp. 3–42. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-44003-3_1
29. Veeramachaneni, K., Arnaldo, I., Derby, O., O'Reilly, U.M.: FlexGP. J. Grid Comput. **13**, 391–407 (2015)
30. Yu, J., Guo, M., Needham, C.J., Huang, Y., Cai, L., Westhead, D.R.: Simple sequence-based kernels do not predict protein-protein interactions. Bioinformatics **26**(20), 2610–2614 (2010). https://doi.org/10.1093/bioinformatics/btq483
31. Zhang, B., Joung, J.G.: Enhancing robustness of genetic programming at the species level. In: Genetic Programming Conference (GP 1997), pp. 336–342. Morgan Kaufmann (1997)
32. Zhang, S.: sonar.all-data (2018). https://www.kaggle.com/ypzhangsam/sonarall data