2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Ensemble Bayesian Model Averaging in Genetic Programming

Alexandros Agapitos, Michael O'Neill, Anthony Brabazon

***Abstract*—This paper considers the general problem of function estimation via Genetic Programming (GP). Data analysts typically select a model from a population of models, and then proceed as if the selected model had generated the data. This approach ignores the uncertainty in model selection, leading to over-confident inferences and lack of generalisation.**

We adopt a coherent method for accounting for this uncertainty through a weighted averaging of all models competing in a population of GP. It is a principled statistical method for post-processing a population of programs into an ensemble, which is based on Bayesian Model Averaging (BMA).

Under two different formulations of BMA, the predictive probability density function (PDF) of a response variable is a weighted average of PDFs centered around the individual predictions of component models that take the form of either standalone programs or ensembles of programs. The weights are equal to the posterior probabilities of the models generating the predictions, and reflect the models' skill on the training dataset.

The method was applied to a number of synthetic symbolic regression problems, and results demonstrate that it generalises better than standard methods for model selection, as well as methods for ensemble construction in GP.

I. Introduction

In the general function estimation problem, one is given a set of training examples $\{(x_i, y_i)\}$, $i = \{1, \ldots, N\}$, where $y \in \mathbb{R}$ is the response variable and $\mathbf{x} \in \mathbb{R}^d$ is a vector of explanatory variables. The goal is to find a function $F^*(\mathbf{x})$ that maps $\mathbf{x}$ to y, such that over the joint distribution $P(\mathbf{x}, y)$ the expected value of some specified loss function $L(y, F(\mathbf{x}))$ is minimised:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} \mathbb{E}_{x,y}[L(y, F(\mathbf{x}))] \quad (1)$$

The application of Genetic Programming (GP) [1] to a regression problem proceeds with the evolution of a population of candidate solutions using typically the squared error loss $L(y, F(\mathbf{x})) = (y - F(\mathbf{x}))^2$, and returns a single model [1] at the end of the evolutionary run. In the end result of this training-error-guided learning process there exists an inherent source of uncertainty as to which model to select in light of a finite set of training data, since the ultimate objective is that of generalisation and not that of merely training-error minimisation. That is, we are aiming to minimise the in-sample error as much as possible and at the same time ensure that this error estimate can act as a proxy for the non-ascertainable out-of-sample error. This uncertainty is usually addressed via model selection that is based on the error assessment of competing programs on an independent validation dataset in order to select the one with the smallest such error.

Ensembles of learning systems have stimulated a great research interest in the Machine Learning (ML) community thanks to their ability to enhance prediction performance over single models. Evidence suggests that combining the output of multiple accurate models (base models), each able to specialise on different parts of the input space, the ensemble can improve generalisation. Very importantly, it is often advantageous to use under-regularised base models, which are models that can potentially overfit the training data [2], [3].

The formation of ensembles arises naturally in GP that maintains a population of solutions, and several scholars have developed methods for combining individual members of a population [4]. As [5] points out, the key to successful ensemble methods is to construct accurate base models which are uncorrelated as far as making errors on the training set of examples is concerned. In GP, diversity was previously enforced via the use of explicit objectives like *negative correlation learning* and *pairwise failure crediting* in a Pareto-based multi-criterion fitness function [6], island architectures [7], and cooperating co-evolutionary approaches [8].

This paper applies a statistical post-processing method, Bayesian Model Averaging (BMA) to construct ensembles of genetic programs drawn from the population of the final generation. The version of BMA, first proposed by Raftery [9], pools across various models while meaningfully incorporating *a priori* uncertainty about the "best" model. It generates ensemble predictions similar to a weighted average of component model predictions, where the weight assigned to each model is calibrated according to its performance in some training set of examples. BMA was applied to diverse problem domains, and was shown to improve the generalisation performance of the resulting ensemble as pointed out in [10] and the references found therein.

We study two main ways in which GP ensembles can be constructed using BMA. First, Bayesian inference can be used to generate a weighted average of the programs in the population, integrating out uncertainty as to which model is correct is the sense of being the data generating model. Second, BMA can be modified to produce a weighted average of different-sized ensembles constructed from a population of programs, this time integrating out uncertainty about *ensemble selection* (the process of choosing which base models from a population to include in the final ensemble) and ensemble size [11], [12].

The work presented in this paper differs from previous

Alexandros Agapitos and Michael O'Neill are with the Complex and Adaptive Systems Laboratory, School of Computer Science and Informatics, University College Dublin, Ireland. Anthony Brabazon is with the School Of Business, University College Dublin, Ireland. (email: {alexandros.agapitos, m.oneill, anthony.brabazon}@ucd.ie).

[1] When referring to GP, the terms model and program refer to the same entity and will be used interchangeably.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

work in three major aspects. First, the majority of papers in the literature study the effect of fitness function, selection/replacement, and output combination methods in a setting where they are collectively rewarded according to the performance of the resultant ensemble in some training data. This allows the model to specialise on certain parts of the input space and at the same time tune the combination of their outputs in order to minimise the in-sample error of the ensemble. In contrast, ensemble BMA follows two independent stages in the learning process. In the first stage the models are trained with standard GP. To maintain a diverse population, a form of fitness sharing [13] is used to encourage the formation of different species. At the second stage, the competing models of the final generation are linearly combined in a Bayes optimal way. There is no interaction between the first and second stages, and the basic assumption is that no particular model can fully encapsulate the data generating model. In addition, an important aspect of the BMA ensemble method presented in [9] is that the component models need not share covariates and/or functional forms, and this makes it particularly appealing to GP.

Second, in previous work, the number of models in an ensemble are often predefined, and typically the evolutionary run returns a single ensemble of programs. In the second thread of our experiments, we use BMA to address the uncertainty arising both in ensemble size and selection, by returning a weighted average of different-sized ensembles.

Third, in contrast to previously developed ensemble methods, the model combination that arises from BMA possesses a range of theoretical optimality properties [14].

The rest of the paper will proceed as follows. Section II briefly reviews previous work on ensemble learning with GP. Section III describes BMA and shows how the parameters of this model can be estimated using the Expectation-Maximisation algorithm. Section IV adapts the BMA model to GP and details the experiment setup. Section V analyses the empirical results, and finally Section VI draws conclusions and suggests future research directions.

II. Ensemble learning with GP

Much work is based on the ensemble learning methods of *bagging* [15] and *boosting* [16]. GP was combined for the first time with bagging and boosting in [17]. GP bagging was applied in [18] to evolve temperature forecasts. In [19] boosting-like heuristics were employed to deal with training sets that do not fit into memory. In [20] they sampled the training set in a bagging-like fashion in the context of parallel cellular GP.

The work in [3] shows how the decomposition of the out-of-sample error into *bias* and *variance* terms can lead to the use of ensemble learning for minimising the error due to variance. Considering the contribution of bias and variance to the total error, Bagging is employed to reduce the error due to variance, while the bias is kept low provided that program-size is relatively unconstrained. Results on two symbolic regression problems were very encouraging, highlighting at the importance of using model averaging to minimise variance.

An ensemble learning approach that combines features from the systems of *adaptive mixtures of local experts* [21] and *stacked generalisation* [22] is presented in the studies of [23], [24]. The main characteristics of the new approach is (a) the use of different classification models (i.e., neural networks, decision trees), (b) the employment of different subsets to train different models, (c) the use of *receiver operator characteristics* curve to calculate fitness, and (d) the use of GP to learn expressions that combine the pre-trained models. Empirical results on a high-dimensional drug discovery dataset showed that the GP-ensemble performed better out-of-sample that the individual base classifiers. GP was also used to combine heterogeneous, pre-trained classifiers in [25].

The work of [26] investigated the effectiveness of various methods for combining the outputs of multiple predictors trained using linear GP. These are namely, *averaging*, *weighting-by-error*, *coevolution-of-weights*, *majority-voting*, *weighted-voting*, *winner-takes-all*, and *weight-optimisation*. These ensemble methods are applied to several prediction problems, and are found to significantly improve both the in-sample and out-of-sample performances. Cooperative coevolution for constructing ensembles was also studied in [8].

The purpose of the study presented in [7] is the formation of diverse ensembles of classifiers. A new GP method based on *N-version programming* is used to quantify semantic diversity, and defines an optimal ensemble as the ensemble that has independent misclassification occurrences among its base classifiers. Results showed improved out-of-sample performance.

The study of [27] uses majority voting to combine classifiers trained with GP. The ensembles are tested on large-scale datasets. Results showed that GP is very effective at learning base classifiers represented as non-linear discriminant functions that are thresholded at the value of zero. The out-of-sample performance of GP-ensembles was found to outperform that of ensembles built upon base classifiers that were trained with a decision tree learning algorithm, and logistic regression.

In the work of [28], ensembles of GP models are constructed by means of *bagging* with a particular emphasis on base-model-diversity preservation for increasing the classification accuracy. Diversity is quantified based on the syntactic structure of an expression-tree. A series of experiments performed on common gene expression datasets demonstrates the superior generalisation performance of the proposed method as opposed to other conventional approaches.

In [29] the authors present a method for building an ensemble of classifiers for cancer microarray data. The proposed method learns diverse base classifiers through the use of *k-means* clustering, and employs a feature selection scheme to obtain discriminant features from each cluster.

The work of [30] investigated whether it is possible to discard some of the GP models of an ensemble in order to increase generalisation. Ensemble pruning identified the most similar classifiers and removed them. For this purpose, three diversity measures are compared to each other. The first two measures are the pairwise syntactic distance between

expression-trees. The third measure is the *k statistic* that quantifies the behavioural similarity (in terms of program outputs) between two programs. The experimental results showed that ensembles can be substantially pruned without increasing misclassification errors. Up to 30% of pruning, out-of-sample classification accuracy was found to increase. Ensemble *pruning* was also investigated in [31], [32].

The work of [33] presents a framework for evolving teams of programs without the need to pre-specify the number of cooperating individuals. To do so, each individual evolves a mapping to a distribution of outcomes that, following clustering, establishes the parameterisation of a Gaussian local membership function. This gives individuals the opportunity to represent subsets of tasks, where the overall task is that of classification under the supervised learning domain.

Finally, the work of [6] presented a Pareto-based multi-objective GP approach to evolve accurate and diverse ensembles of classifiers with good performance in class-imbalance problems.

III. Bayesian Model Averaging

Using a single model for predictions ignores uncertainty about model correctness of a model trained on a finite amount of data. Bayesian Model Averaging [9], [10] (BMA) overcomes this problem by conditioning not on a single "best" model but on the entire set of competing models considered during model selection.

In the general function estimation problem, described in the introductory section, we have a quantity of interest $y \in \mathbb{R}$ to forecast on the basis of training data $D = \{(x_i, y_i)\}_1^N$ using K trained models $M_1, \ldots, M_K$. The law of total probability dictates that the forecast probability density function (PDF), $p(y)$, is given by:

$$p(y) = \sum_{k=1}^{K} p(y|M_k)p(M_k|D) \tag{2}$$

where $p(y|M_k)$ is the forecast PDF based on model M_k alone, and $p(M_k|x)$ is the posterior probability of model M_k being correct given the training data, and can be calculated using the Bayesian theorem:

$$p(M_k|D) = \frac{p(D|M_k)p(M_k)}{\sum_{k=1}^{K} p(D|M_k)p(M_k)} \tag{3}$$

where $p(D|M_k)$ denotes the likelihood (reflects how well model M_k fits the training data), and $p(M_k)$ is the prior probability distribution for model M_k. The posterior model probabilities add up to one, so that $\sum_{k=1}^{K} p(M_k|D) = 1$, and they can be viewed as weights. Thus, the BMA PDF is a weighted average of the PDFs given the K individual models, weighted by their posterior model probabilities.

Now, given a GP population of K models evolved on a set of training examples, the basic idea is that for any given prediction based on input $\mathbf{x}$ there is a "best" model but we do not know what it is, and this uncertainty is quantified by BMA. Each deterministic prediction f_k generated by model k is associated with a conditional PDF $g_k(y|f_k)$, which can be interpreted as the conditional PDF of y conditional of f_k, given that f_k is the best prediction generated amongst the population members. The BMA predictive model then takes the following form:

$$p(y|f_1, \ldots, f_k) = \sum_{k=1}^{K} w_k g_k(y|f_k) \tag{4}$$

where w_k is the posterior probability of prediction k being the best one, and is based on model k's accuracy in the training data. The w_k's add up to 1. We will assume that each conditional PDF $g_k(y|f_k)$ is approximated by a Gaussian distribution centered at f_k, so that it is a Gaussian PDF with mean f_k and program-specific standard deviation σ_k. We denote this situation by:

$$y|f_k \sim N(y, f_k, \sigma_k^2) = \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{(y-f_k)^2}{2\sigma_k^2}} \tag{5}$$

In that case the BMA predictive mean is the conditional expectation of y given the forecasts:

$$\begin{aligned} \mathbb{E}[y|f_1, \ldots, f_k] &= \sum_{k=1}^{K} \mathbb{E}[y|f_k]p(f_k|D) \\ &= \sum_{k=1}^{K} w_k f_k \end{aligned} \tag{6}$$

A. Parameter Estimation by Maximum Likelihood and the EM Algorithm

We now consider how to estimate the ensemble BMA model (Equation 4) parameters w_k and σ_k for $k = \{1, \ldots, K\}$, on the basis of training data $D = \{(x_i, y_i)\}_1^N$.We denote the set of BMA model parameters to be estimated by θ. We denote a training example by subscript i, so f_{ki} denotes the kth prediction in the ensemble for training example i, and y_i denotes the corresponding observation.

We estimate θ by maximum likelihood from the training data. The likelihood function is defined as the probability of the training data given θ, viewed as a function of θ. Here we are maximising the log-likelihood function, which for model 4 is:

$$l(\theta) = \sum_{i=1}^{N} log\left(\sum_{k=1}^{K} w_k g_k(y_i|f_{ki})\right) \tag{7}$$

The log-likelihood function cannot be maximised analytically, but [9] suggest using the *Expectation-Maximisation* (EM) algorithm [34] (page 272). The BMA model of Equation 4 is a finite *mixture model*. We introduce unobserved latent variables z_{ki}, which represent the posterior probability for model k for observation y_i. $z_{ki} = 1$ if ensemble member k is the best predictor for y_i, and $z_{ki} = 0$ otherwise. In

reality, the estimates of z_{ki} are not necessarily integers, even though the true values are 0 and 1.

The EM algorithm is iterative, and alternates between two steps, The E (Expectation) and M (Maximisation). It starts with an initial guess, $\theta^{(0)}$, for the parameter vector θ. In the E step, we perform a soft assignment of each observation to each model: the current estimates of the parameters are used to assign responsibilities according to the relative density of the training examples under each model k. For the normal BMA model given by 4 and 5, the E step is:

$$\hat{z}_{ki}^{(j)} = \frac{g(y_i|f_{ki}, \sigma_k^{(j-1)})}{\sum_{m=1}^{K} g(y_i|f_{mi}, \sigma_m^{(j-1)})} \tag{8}$$

where the superscript j refers to the jth iteration of the EM algorithm, and $g(y_i|f_{ki}, \sigma_k^{(j-1)})$ is the normal density with mean f_{ki} and standard deviation $\sigma_k^{(j-1)}$ evaluated at y_i. The M step performs a weighted maximum-likelihood estimation of w_k and σ_k for $k = \{1, \ldots, K\}$, using as weights the current estimates z_{ki}, i.e. $\hat{z}_{ki}^{(j)}$. This is as follows:

$$\begin{aligned} w_k^{(j)} &= \frac{1}{N}\sum_{i=1}^{N} \hat{z}_{ki}^{(j)} \\ \sigma_k^{2(j)} &= \frac{\sum_{i=1}^{N} \hat{z}_{ki}^{(j)}(y_i - f_{ki})^2}{\sum_{i=1}^{N} \hat{z}_{ki}^{(j)}} \end{aligned}$$

where N is the number of training examples. The E and M steps are then iterated until the improvement in the log-likelihood is no larger than some pre-defined tolerance which in our experiments is set to 0.01. Although the log-likelihood is guaranteed to increase at each iteration of the algorithm, convergence is only guaranteed to a local maximum of the log-likelihood function. Convergence to the global maximum is not assured, and the end result may be sensitive to initial conditions. In future research we will explore these convergence issues more extensively. In the experiments that follow, we initialise $w_k = 1/K \ \forall \ k \in \{1, \ldots, K\}$. Each σ_k^2 is set equal to the overall variance of the output of model k on the training examples, that is $\sum_{i=1}^{N}(y_i - \bar{y}_k)^2/N$, and $\bar{y}_k$ is the mean of outputs of model k that is equal to $\sum_{i=1}^{N} f_{ki}/N$.

IV. Methods

A. Proposed methods

Ensemble BMA is studied at two distinct levels. First, at the level of individual programs that reside in the final population, referred to as $EBMA_{base}$. Second, at the level of ensembles that can be formed from programs that reside in the final population, referred to as $EBMA_{ensemble}$. In both cases, the process starts by evolving a population of M programs for G generations using GP. To maintain a diverse population, a variation of the fitness sharing mechanism presented in [13] is used to encourage the formation of different species. Fitness sharing accomplishes speciation by degrading the raw fitness of an individual according to the presence of similar individuals in a population.

Here, fitness sharing is realised via a sharing function K_λ that is based on the Euclidean distance between the *semantics* vectors of two individual programs s_1 and s_2, and takes the form of a logistic kernel as follows:

$$\begin{aligned} K_\lambda(s_{f_1}, s_{f_2}) &= D\left(\frac{||s_{f_1} - s_{f_2}||}{\lambda}\right) \\ D(t) &= \frac{1}{e^t + 2 + e^{-t}} \end{aligned}$$

where s_i is the semantics vector for program i, $||\cdot||$ denotes the Euclidean norm, and λ is the radius of the kernel, which is set to 0.1 in the experiments. Given an individual f and a set of N training examples $D = \{(x_i, y_i)\}_1^N$ with input vectors $x \in \mathbb{R}^d$, we define the semantics vector $s_f = [f(x_1), \ldots, f(x_N)]$, where $f(x_i)$ is the output of program f on training example i.

Given a loss function that takes the form of Root Mean Square Error (RMSE) and a population of M programs, the shared error of an individual m (to be minimised) is defined as:

$$SE_{f_m} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f_m(x_i))^2} \cdot \left(1 + \sum_{j=1, j\neq m}^{M} K_\lambda(s_{f_m}, s_{f_j})\right) \tag{9}$$

Once the evolutionary run reaches the final generation, the population is sorted in ascending training RMSE values, and it is pruned to an *archive* of size $B < M$ of the fittest programs, where a program i is accepted to the archive if $RMSE_i \leq 2.0 \cdot RMSE_{fittest}$ ($RMSE_{fittest}$ is the minimum error of an individual in the final generation). The method of $EBMA_{base}$ uses all individual programs that reside in the archive. Thus in BMA model 4 we set $K = B$.

The method of $EBMA_{ensemble}$ uses a diverse set of ensembles that are generated from the archive. There is a single modification required to turn BMA model 4 into $EBMA_{ensemble}$. This is as follows:

$$p(y|e_1, \ldots, e_k) = \sum_{k=1}^{K} w_k g_k(y|e_k) \tag{10}$$

where e_k is the prediction of an element of the set E of ensembles of different size and different base-models. The output of an ensemble e_k is formed by a simple averaging of outputs of individual programs in the ensemble. In the case where the output of a program is undefined or infinite, the program output is excluded from the average.

Although the space of potential model ensembles that can be constructed from the archive is very large, here we are resorting to a variation of the method introduced in [35]. The programs of the archive are clustered using the *k-means* algorithm [34] (page 509). The resulting clusters correspond to different species. Clustering is based on the Euclidean distances between the semantics vectors of programs. The fittest individual (according to the training error) of each cluster is selected, and those individuals are then used to

TABLE I. GP SYSTEM SETUP

EA	elitist (p=1), generational, expression-tree representation
Function set	+, -, *, exp(x), power(x,y), log(x), sqrt(x), sin(x)
Terminal set	explanatory vars, 10 random consts $\in \{-1.0, \ldots, 1.0\}$
No. of generations	101
Population size	500
Tournament size	7
Tree creation	ramped half-and-half (depths of 2 to 5)
Max. tree depth	10
Subtree crossover	20% (90% inner-nodes, 10% leaf-nodes)
Subtree mutation	50% (max. depth of subtree: uniform randomly in [1, 4])
Point mutation	30% (prob. of a node to be mutated: 10% or 30% or 40%)
Fitness function	Equation 9

form an ensemble of programs. In order to generate a diverse collection of ensembles, we run the k-means algorithm with $k = \{2, 3, \ldots, 36\}$, and for each different value of k we generate a different ensemble of k programs. This allows the creation of $K = 35$ different-sized ensembles that will be averaged using BMA model 10.

B. Baseline methods

We compare the generalisation performance (as measured on an independent test set) of the ensemble methods proposed above with a number of models resulting from validation-based model selection, as well as other linear combination schemes for constructing ensembles. These are presented bellow:

1) **Elitist**. This returns the best-of-run individual (according to training RMSE).
2) **Validation-base**. A list of best-of-generation programs is initialised in the beginning of the evolutionary run. The best individual (according to training RMSE) of each generation is added to list. At the end of the evolutionary run, each member of the list is tested against an independent validation set. The individual with the smallest validation RMSE is designated as the output of the run. We adopt the common practice in ML, and divide the learning data into two disjoint sets for training (66%), and validation (33%).
3) **OLS-base**. Ordinary Least Squares (OLS) regression is used to fit a linear model that combines the outputs of the B individual programs populating the archive generated from the final generation.
4) **OLS-ensemble**. OLS regression is used to fit a linear model that combines the outputs of the 35 ensembles that result from the clustering method described in the previous section.

C. Experiment setup

Table I presents the experiment setup of the GP system. To demonstrate the effect of the two methods of ensemble BMA on model generalisation we selected a suit of synthetic symbolic regression problems taken from [36], and generated reliable noise-free data for training, validation and testing. For the testing data, we draw a further distinction between interpolation and extrapolation. The details of the sampling procedures used to create the aforementioned disjoint datasets are summarised in Table II.

V. RESULTS ANALYSIS

We conducted 50 independent GP runs for each problem. Generalisation performance is measured on the test set. Table III summarises the results. In each case of interpolation and extrapolation we count the fraction of solutions that showed "pathological" behaviour. Columns 2 and 8 is the percentage of solutions producing infinite or undefined RMSE for interpolation and extrapolation respectively. Columns 3 and 9 present the percentage of solutions for which the RMSE is infinite, undefined or excessively large. The threshold for pathologically high error is chosen to be 100 (it corresponds to an MSE of 10^4). Solutions producing these undefined, infinite or large errors on test data are dangerously erroneous, and a high percentage of them is indicative of overfitting.

We observe that all methods but $EBMA_{ensemble}$ and $EBMA_{base}$ exhibit pathologies on the test data for the majority of problems. This becomes more pronounced for the case of extrapolation. There is a difference in the type of pathologies between the standalone models produced by elitist and Validation-base methods and OLS methods, with the former suffering primarily by infinite on undefined RMSE, and the latter suffering only from large RMSEs. We suggest that this is due to the non-regularised version of OLS. This allows the mixing coefficients a to get excessively large values, which then contribute to large predictions generated by the ensemble especially in the case of extrapolation. A potential solution to this problem is to fit the OLS model with two extra constraints: $\sum_{k=1}^{K} a_k = 1$ and $a_k \geq 0$, leading to a formulation of a quadratic programming problem. One can argue however that undefined or infinite RMSE is generally more pathological than large RMSE, so in that sense the OLS-based ensembles exhibited a more graceful degradation by extrapolation than the standalone models. Finally, we note that validation-based model selection allowed for a reduction in the number of pathologies for the case of interpolation as opposed to the best-of-run individual, however its effect diminished for the case of extrapolation. Overall, results suggest that the BMA-based ensembles consistently exhibited no pathologies on test data.

The minimum, median [2] and interquartile range of the RMSE over 50 runs are reported in columns 4, 5, 6 and 10, 11, 12 for interpolation and extrapolation cases respectively. For example, for Elitist solutions in problem F_1, we observe that 62% of best-of-run solutions (31 out of 50) have an infinite or undefined or excessively large error pathology on the extrapolation test data. If these 31 solutions are excluded from the sample then the median of the remaining 19 solutions will be 0.068 and the interquartile range will be 0.071. This is an argument for using ensembles of solutions, where a model's pathological output is excluded from the ensemble average, and for being very cautious in using best-of-run solutions with or without model selection. If only best-of-run solutions are sought for, then all runs where the best program produces a pathology are lost. This corresponds to a significant waste of 62% of the computational effort in the aforementioned example.

[2] We preferred to report the median over the mean because it is more robust to outliers.

TABLE II. SYMBOLIC REGRESSION PROBLEMS WITH THE RESPECTIVE DATA SAMPLING RANGES FOR TRAINING, VALIDATION AND TEST DATASETS. NOTATION X=RAND(A,B) MEANS THAT THE X VARIABLE IS SAMPLED UNIFORM RANDOMLY FROM THE INTERVAL [A, B]. NOTATION $x_1 = (a_1 : c_1 : b_1), x_2 = (a_2 : c_2 : b_2)$ DETERMINES A UNIFORM MESH WITH STEP LENGTH (c_1, c_2) ON AN INTERVAL $[a_1, b_1] \times [a_2, b_2]$.

	Problem	**Training data**	**Validation data**	**Test data**	
				Interpolation	**Extrapolation**
F_1	$f(x_1, x_2) = \frac{e^{-(x_1-1)^2}}{1.2+(x_2-2.5)^2}$	100 points $x_,x_2$ =rand(0.3, 4.0)	50 points x_1, x_2=rand(0.3, 4.0)	1,369 points x_1, x_2=(0.3 : 0.1 : 4.0)	797 points x_1, x_2=(-0.2 : 0.02 : 0.3) x_1, x_2=(4.0 : 0.02 : 4.2)
F_2	$f(x) =$ $e^{-x}x^3cos(x)sin(x)(cos(x)sin^2x - 1)$	200 points x=rand(0.05, 10.0)	100 points x=rand(0.05, 10.0)	996 points x=(0.05 : 0.01 : 10.0)	525 points x=(-0.5 : 0.002 : 0.05) x=(10.0 : 0.002 : 10.5)
F_3	$f(x_1, x_2, x_3, x_4, x_5) = \frac{10}{5+\sum_{i=1}^{5}(x_i-3)^2}$	512 points x_1, x_2, x_3, x_4, x_5 =rand(0.05, 6.05)	256 points x_1, x_2, x_3, x_4, x_5 =rand(0.05, 6.05)	5,000 points x_1, x_2, x_3, x_4, x_5 =rand(0.05, 6.05)	5,000 points x_1, x_2, x_3, x_4, x_5 =rand(-0.25, 0.05) x_1, x_2, x_3, x_4, x_5 =rand(6.05, 6.35)
F_4	$f(x_1, x_2, x_3) = 30\frac{(x_1-1)(x_3-1)}{{x_2}^2(x_1-10)}$	300 points x_1, x_3 =rand(0.05, 2.0) x_2=rand(1.0, 2.0)	150 points x_1, x_3=rand(0.05, 2.0) x_2=rand(1.0, 2.0)	1,960 points x_1, x_3=(0.05 : 0.15 : 2.0) $x_2 = (1.0 : 0.1 : 2.0)$	1,326 points x_1, x_3=(-0.05 : 0.01 : 0.05) $x_2 = (0.95 : 0.01 : 0.05)$ x_1, x_3=(2.0 : 0.01 : 2.1) $x_2 = (2.0 : 0.01 : 2.05)$
F_6	$f(x_1, x_2) =$ $(x_1 - 3)(x_2 - 3) +$ $2sin((x_1 - 4)(x_2 - 4))$	300 points x_1, x_2 =rand(0.05, 6.05)	150 points x_1, x_2=rand(0.05, 6.05)	1,000 points x_1, x_2=rand(0.05, 6.05)	1,000 points x_1, x_2=rand(-0.25, 0.05) x_1, x_2=(6.05 : 0.05 : 6.35)
F_7	$f(x_1, x_2) = \frac{(x_1-3)^4+(x_2-3)^3-(x_2-3)}{(x_2-2)^4+10}$	50 points x_1, x_2 =rand(0.05, 6.05)	25 points x_1, x_2=rand(0.05, 6.05)	3,721 points x_1, x_2=(0.05 : 0.1 : 6.05)	1,861 points x_1, x_2=(-0.25 : 0.01 : 0.05) x_1, x_2=(6.05 : 0.01 : 6.35)
F_8	$f(x_1, x_2) =$ $x_1x_2 + sin((x_1 - 1)(x_2 - 1))$	20 points x_1, x_2 =rand(-3.0, 3.0)	10 points x_1, x_2=rand(-3.0, 3.0)	90,000 points x_1, x_2=(-3.0 : 0.02 : 3.0)	20,402 points x_1, x_2=(-4.0 : 0.01 : -3.0) x_1, x_2=(3.0 : 0.01 : 4.0)
F_{10}	$f(x_1, x_2) = \frac{8}{2+{x_1}^2+{x_2}^2}$	20 points x_1, x_2 =rand(-3.0, 3.0)	10 points x_1, x_2=rand(-3.0, 3.0)	90,000 points x_1, x_2=(-3.0 : 0.02 : 3.0)	20,402 points x_1, x_2=(-4.0 : 0.01 : -3.0) x_1, x_2=(3.0 : 0.01 : 4.0)
F_{12}	$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7,$ $x_8, x_9, x_{10}) =$ $x_1 + x_2 + x_3 * x_4 + x_5 * x_6$ $+x_1 * x_7 * x_9 + x_3 * x_6 * x_{10}$	50 points $x_1, \ldots, x_{10}$ =rand(-1.0, 1.0)	25 points $x_1, \ldots, x_{10}$ =rand(-1.0, 1.0)	30,000 points $x_1, \ldots, x_{10}$ =rand(-1.0, 1.0)	30,000 points $x_1, \ldots, x_{10}$ =rand(-1.5, -1.0) $x_1, \ldots, x_{10}$ =rand(1.0, 1.5)

We performed a pair-wise Wilcoxon-Mann-Whitney rank sum test to assess the statistical significance in the difference of the median RMSE obtained between $EBMA_{ensemble}$ and the rest of the methods. In order to obtain equal sample sizes, we truncated all infinite, undefined, and large values of RMSE (RMSE ≥ 100) at the value of 100. The significance level is set to 95%, and the*p*-values of the tests are given in columns 7 and 13 for interpolation and extrapolation respectively. For the case of interpolation, statistically significance differences were obtained in 5 our of 9 problems between $EBMA_{ensemble}$ and the rest of the non-BMA methods, with the former outperforming the latter. For the case of extrapolation, statistically significant differences were obtained in 7 out 9 problems, with $EBMA_{ensemble}$ performing better than every other non-BMA method. Overall, no statistically significant differences were found between the performance of the two EBMA methods either for interpolation or extrapolation.

VI. CONCLUDING REMARKS AND FUTURE WORK

Standard GP practice typically selects the best model from a population of competing models and then proceeds as if the selected model had generated the data. This approach ignores the uncertainty in model selection and may lead to bad generalisation. The use of a validation-based model selection, while beneficial in the majority of cases, is not a panacea since this independent dataset may itself suffer from sampling bias and not be representative of the true data generating PDF.

One explanation for the success of ensemble methods is that as opposed to a single model, ensembles address the uncertainty about the correctness of a single model via some kind of model combination. This paper presented the application of Bayesian Model Averaging to perform optimally weighted averages of standalone programs or ensembles of programs at the end of the evolutionary run. Fitness sharing was used to encourage the evolution of semantically-diverse individuals. Results are encouraging, showing that the proposed method outperformed standard methods of model selection and ensemble construction in terms of out-of-sample performance. Very importantly, both EBMA models showed no pathological behaviour when extrapolated outside the training range. The fact that no significance difference was found between the two versions of EBMA surely warrants further investigation.

There exist various directions for extending this research:

1) Comparison of performance between EBMA and other post-processing methods like boosting, bagging, and stacked generalisation.
2) Investigation of the impact of different initial values in the maximisation of the log-likelihood function

TABLE III. MINIMUM, MEDIAN AND INTERQUARTILE RANGE (IQR) FOR RMSE CALCULATED ON 50 INDEPENDENT RUNS. P-VALUES FOR PAIR-WISE WILCOXON-MANN-WHITNEY TESTS ABOUT THE SIGNIFICANCE IN DIFFERENCES OF MEDIAN RMSE BETWEEN $EBMA_{ensemble}$ AND THE REST OF THE METHODS. BOLD FACE INDICATES STATISTICAL SIGNIFICANCE IN ALL DIFFERENCES BETWEEN THE MEDIAN OF $EBMA_{ensemble}$ METHOD AND THE REST OF NON-BMA METHODS, WITH THE FORMER OUTPERFORMING THE REST.

	Method	**Interpolation**						**Extrapolation**					
		Pathologies		**Min**	**Median**	**IQR**	**p-value**	**Pathologies**		**Min**	**Median**	**IQR**	**p-value**
		RMSE $= \infty$, NaN	RMSE ≥ 100					RMSE $= \infty$, NaN	RMSE ≥ 100				
	Column no:	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
F_1	Elitist	2.0%	2.0%	0.019	0.052	0.025	0.0	62.0%	62.0%	0.019	0.068	0.071	0.0
	Validation-b	2.0%	2.0%	0.019	0.052	0.028	0.0	54.0%	54.0%	0.019	0.068	0.190	0.0
	OLS-b	0.0%	0.0%	0.004	0.044	0.035	0.01	0.0%	6.0%	0.036	1.015	1.812	0.0
	OLS-e	0.0%	0.0%	0.011	0.038	0.022	0.005	0.0%	0.0%	0.021	0.096	0.137	0.01
	EBMA-b	0.0%	0.0%	0.016	0.021	0.023	0.27	0.0%	0.0%	0.016	0.049	0.072	0.53
	EBMA-e	0.0%	0.0%	0.016	**0.027**	0.023	–	0.0%	0.0%	0.007	**0.057**	0.069	–
F_2	Elitist	2.0%	2.0%	0.030	0.075	0.033	0.61	34.0%	34.0%	0.020	0.084	0.105	0.0
	Validation-b	2.0%	2.0%	0.030	0.075	0.033	0.58	36.0%	36.0%	0.017	0.062	0.103	0.0
	OLS-b	0.0%	20.0%	0.000	0.014	0.111	0.26	0.0%	36.0%	0.015	0.742	1.536	0.0
	OLS-e	0.0%	4.0%	0.001	0.027	0.040	0.0	0.0%	14.0%	0.019	0.327	1.108	0.0
	EBMA-b	0.0%	0.0%	0.034	0.073	0.040	0.35	0.0%	0.0%	0.013	0.051	0.103	0.80
	EBMA-e	0.0%	0.0%	0.032	0.070	0.036	–	0.0%	0.0%	0.011	**0.057**	0.108	–
F_3	Elitist	2.0%	2.0%	0.100	0.144	0.033	0.06	24.0%	24.0%	0.039	0.136	0.011	0.02
	Validation-b	2.0%	2.0%	0.102	0.144	0.033	0.03	26.0%	26.0%	0.039	0.135	0.009	0.03
	OLS-b	0.0%	0.0%	0.032	0.134	0.035	0.0	0.0%	14.0%	0.034	0.875	1.636	0.0
	OLS-e	0.0%	0.0%	0.051	0.126	0.030	0.0	0.0%	0.0%	0.024	0.230	0.248	0.0
	EBMA-b	0.0%	0.0%	0.097	0.062	0.040	0.12	0.0%	0.0%	0.074	0.134	0.026	0.54
	EBMA-e	0.0%	0.0%	0.096	**0.085**	0.032	–	0.0%	0.0%	0.060	**0.134**	0.024	–
F_4	Elitist	6.0%	6.0%	0.043	0.267	0.158	0.003	54.0%	64.0%	0.316	2.359	1.893	0.0
	Validation-b	4.0%	4.0%	0.043	0.248	0.143	0.007	62.0%	72.0%	0.316	2.541	2.335	0.0
	OLS-b	0.0%	26.0%	0.002	0.237	0.170	0.009	0.0%	92.0%	2.494	4.558	3.354	0.0
	OLS-e	0.0%	4.0%	0.016	0.222	0.158	0.01	0.0%	50.0%	0.555	2.381	1.975	0.0
	EBMA-b	0.0%	0.0%	0.060	0.099	0.165	0.47	0.0%	0.0%	0.583	1.977	0.848	0.84
	EBMA-e	0.0%	0.0%	0.054	**0.129**	0.164	–	0.0%	0.0%	0.096	**1.932**	1.720	–
F_6	Elitist	0.0%	0.0%	1.265	1.713	0.517	0.46	54.0%	60.0%	1.845	7.574	5.564	0.0
	Validation-b	0.0%	0.0%	1.265	1.766	0.504	0.41	54.0%	60.0%	1.845	7.328	4.848	0.0
	OLS-b	0.0%	42.0%	0.686	2.071	2.181	0.0	0.0%	96.0%	12.505	12.505	n/a	0.0
	OLS-e	0.0%	4.0%	1.058	1.336	0.300	0.0	0.0%	44.0%	2.060	7.555	5.765	0.0
	EBMA-b	0.0%	0.0%	1.354	1.833	0.535	0.66	0.0%	0.0%	2.381	7.861	2.177	0.87
	EBMA-e	0.0%	0.0%	1.337	1.786	0.568	–	0.0%	0.0%	1.180	**7.946**	3.070	–
F_7	Elitist	2.0%	10.0%	0.488	1.317	0.725	0.32	78.0%	84.0%	0.352	0.891	0.876	0.0
	Validation-b	2.0%	4.0%	0.470	1.302	0.736	0.55	80.0%	86.0%	0.419	1.306	1.054	0.0
	OLS-b	0.0%	24.0%	0.306	0.925	0.470	0.93	0.0%	44.0%	1.190	2.904	2.240	0.0
	OLS-e	0.0%	36.0%	0.664	1.540	0.688	0.0	0.0%	68.0%	1.827	3.236	3.632	0.0
	EBMA-b	0.0%	0.0%	0.475	1.293	0.643	0.86	0.0%	0.0%	0.405	2.342	2.113	0.07
	EBMA-e	0.0%	0.0%	0.480	1.303	0.676	–	0.0%	0.0%	0.456	**3.133**	2.906	–
F_8	Elitist	2.0%	2.0%	0.001	0.811	0.126	0.001	4.0%	4.0%	0.001	1.023	0.662	0.007
	Validation-b	0.0%	0.0%	0.001	0.797	0.070	0.03	2.0%	2.0%	0.001	0.860	0.313	0.33
	OLS-b	0.0%	8.0%	0.000	0.792	0.243	0.0	0.0%	6.0%	0.000	1.483	1.758	0.0
	OLS-e	0.0%	8.0%	0.000	1.037	0.616	0.0	0.0%	8.0%	0.000	1.921	1.653	0.0
	EBMA-b	0.0%	0.0%	0.001	0.725	0.104	0.84	0.0%	0.0%	0.001	0.895	0.249	0.71
	EBMA-e	0.0%	0.0%	0.000	**0.719**	0.106	–	0.0%	0.0%	0.000	0.879	0.226	–
F_{10}	Elitist	8.0%	14.0%	0.146	0.647	0.275	0.09	8.0%	8.0%	0.072	0.590	0.394	0.72
	Validation-b	2.0%	2.0%	0.162	0.629	0.177	0.07	2.0%	2.0%	0.072	0.697	0.405	0.01
	OLS-b	0.0%	20.0%	0.155	0.697	0.309	0.03	0.0%	10.0%	0.115	0.681	0.629	0.01
	OLS-e	0.0%	24.0%	0.185	0.758	0.501	0.0	0.0%	12.0%	0.093	0.870	0.924	0.0
	EBMA-b	0.0%	0.0%	0.144	0.610	0.290	0.85	0.0%	0.0%	0.066	0.601	0.395	0.75
	EBMA-e	0.0%	0.0%	0.155	**0.608**	0.276	–	0.0%	0.0%	0.072	0.606	0.408	–
F_{12}	Elitist	0.0%	4.0%	0.280	0.488	0.222	0.17	0.0%	10.0%	1.878	5.170	1.852	0.005
	Validation-b	0.0%	4.0%	0.226	0.463	0.229	0.77	0.0%	4.0%	2.547	5.051	1.103	0.007
	OLS-b	0.0%	10.0%	0.375	0.865	0.442	0.0	0.0%	40.0%	1.913	5.907	3.096	0.0
	OLS-e	0.0%	6.0%	0.443	0.863	0.413	0.0	0.0%	48.0%	2.377	6.657	2.742	0.0
	EBMA-b	0.0%	0.0%	0.274	0.455	0.246	0.71	0.0%	0.0%	1.767	4.250	1.605	0.82
	EBMA-e	0.0%	0.0%	0.278	0.453	0.256	–	0.0%	0.0%	2.646	**4.333**	1.699	–

for estimating BMA parameters.

3) For the case of $EBMA_{ensemble}$, investigation of different ways of forming ensembles from the final population. A potential method would be to fit regularised least-squares linear model to the program outputs of the programs populating each cluster using different amount of regularisation. This technique can also be applied to the final population as a whole.
4) Study of different combination methods for ensemble output other than simple averaging in the case of $EBMA_{ensemble}$. Some form of weight optimisation (i.e. using a GA) is the obvious next step.
5) Implementation of supplementary mechanisms to enforce uncorrelated errors between the individuals of a population, like *negative correlation learning* and *pairwise failure crediting* in a Pareto-based multi-objective optimisation context.

ACKNOWLEDGEMENT

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number 08/SRC/FM1389.

REFERENCES

[1] R. Poli, W. B. Langdon, and N. F. McPhee, *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008.

[2] P. Sollich and A. Krogh, "Learning with ensembles: How overfitting can be useful." in *NIPS*, D. S. Touretzky, M. Mozer, and M. E. Hasselmo, Eds. MIT Press, 1995, pp. 190–196.

[3] M. Keijzer and V. Babovic, "Genetic programming, ensemble methods and the bias/variance tradeoff - introductory investigations," in *Genetic Programming, Proceedings of EuroGP'2000*, ser. LNCS, R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, Eds., vol. 1802. Edinburgh: Springer-Verlag, 15-16 Apr. 2000, pp. 76–90.

[4] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *Transactions Systems, Man and Cybernetics, Part C*, vol. 40, pp. 121–144, March 2010.

[5] L. I. Kuncheva, *Combining Pattern Classifiers, Methods and Algorithms*. New York: Wiley, 2005.

[6] U. Bhowan, M. Johnston, M. Zhang, and X. Yao, "Evolving diverse ensembles using genetic programming for classification with unbalanced data," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 368–386, Jun. 2013.

[7] K. Imamura, T. Soule, R. B. Heckendorn, and J. A. Foster, "Behavioral diversity and a probabilistically optimal GP ensemble," *Genetic Programming and Evolvable Machines*, vol. 4, no. 3, pp. 235–253, Sep. 2003.

[8] R. Thomason and T. Soule, "Novel ways of improving cooperation and performance in ensemble classifiers," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007.

[9] R. A. E., T. Gneiting, F. Balabdaoui, and M. Polakowski, "Using bayesian model averaging to calibrate forecast ensembles," *Monthly Weather Review*, vol. 133, pp. 1155–1174, 2005.

[10] J. M. Montgomery, F. Hollenbach, and M. D. Ward, "Improving predictions using ensemble bayesian model averaging," *Political Analysis*, 2012.

[11] H.-C. Kim and Z. Ghahramani, "Bayesian classifier combination," in *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, 2012.

[12] K. Monteith, J. L. Carroll, K. Seppi, and T. Martinez, "Turning bayesian model averaging into bayesian model combination," in *Proceedings of the 2011 Joint Conference on Neural Networks (IJCNN)*, 2011, pp. 2657–2663.

[13] Q. U. Nguyen, X. H. Nguyen, M. O'Neill, and A. Agapitos, "An investigation of fitness sharing with semantic and syntactic distance metrics," in *Proceedings of the 15th European Conference on Genetic Programming, EuroGP 2012*, ser. LNCS, A. Moraglio, S. Silva, K. Krawiec, P. Machado, and C. Cotta, Eds., vol. 7244. Malaga, Spain: Springer Verlag, 11-13 Apr. 2012, pp. 109–120.

[14] A. E. Raftery, Y. Zheng, N-We, M. Clyde, J. Hoeting, and D. Madigan, "Long-run performance of bayesian model averaging," *Journal of the American Statistical Association*, vol. 98, pp. 931–938, 2003.

[15] L. Breiman and L. Breiman, "Bagging predictors," *Machine Learning*, pp. 123–140, 1996.

[16] R. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[17] H. Iba, "Bagging, boosting, and bloating in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, Eds., vol. 2. Orlando, Florida, USA: Morgan Kaufmann, 13-17 Jul. 1999, pp. 1053–1060.

[18] A. Agapitos, M. O'Neill, and A. Brabazon, "Genetic programming for the induction of seasonal forecasts: A study on weather derivatives," in *Financial Decision Making Using Computational Intelligence*, ser. Springer Optimization and Its Applications, D. Michael, Z. Constantin, and P. Panos, Eds. Springer, 2012, vol. 70, ch. 6, pp. 153–182, due: July 31, 2012.

[19] D. Song, M. I. Heywood, and A. N. Zincir-Heywood, "Training genetic programming on half a million patterns: an example from anomaly detection," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 225–239, Jun. 2005.

[20] G. Folino, C. Pizzuti, and G. Spezzano, "Ensemble techniques for parallel genetic programming based classifiers," in *Genetic Programming, Proceedings of EuroGP'2003*, ser. LNCS, C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, Eds., vol. 2610. Essex: Springer-Verlag, 14-16 Apr. 2003, pp. 59–69.

[21] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Mar. 1991.

[22] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[23] W. B. Langdon, S. J. Barrett, and B. F. Buxton, "Genetic programming for combining neural networks for drug discovery," in *Soft Computing and Industry Recent Applications*, R. Roy, M. Köppen, S. Ovaska, T. Furuhashi, and F. Hoffmann, Eds. Springer-Verlag, 10–24 Sep. 2001, pp. 597–608, published 2002.

[24] ——, "Combining decision trees and neural networks for drug discovery," in *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002, volume 2278 of LNCS*. Springer-Verlag, 2002, pp. 60–70.

[25] W. B. Langdon and B. F. Buxton, "Genetic programming for combining classifiers," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann, 2001, pp. 66–73.

[26] M. Brameier and W. Banzhaf, "Evolving teams of predictors with linear genetic programming," *Genetic Programming and Evolvable Machines*, vol. 2, pp. 381–407, December 2001.

[27] Y. Zhang and S. Bhattacharyya, "Genetic programming in classifying large-scale data: an ensemble method," *Inf. Sci.*, vol. 163, pp. 85–101, June 2004.

[28] J.-H. Hong and S.-B. Cho, "The classification of cancer based on dna microarray data that uses diverse ensemble genetic programming," *Artif. Intell. Med.*, vol. 36, pp. 43–58, January 2006.

[29] S. Hengpraprohm and P. Chongstitvatana, "A genetic programming ensemble approach to cancer microarray data classification," in *Proceedings of the 2008 3rd International Conference on Innovative Computing Information and Control*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 340–.

[30] G. Folino, C. Pizzuti, and G. Spezzano, "Training distributed gp ensemble with a selective algorithm based on clustering and pruning for pattern classification," *IEEE Trans. Evolutionary Computation*, vol. 12, no. 4, pp. 458–468, 2008.

[31] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Ensemble diversity measures and their application to thinning," *Information Fusion*, vol. 6, p. 2005, 2004.

[32] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 211–218.

[33] A. R. McIntyre and M. I. Heywood, "Classification as clustering: A pareto cooperative-competitive GP approach," *Evolutionary Computation*, vol. 19, no. 1, pp. 137–166, Spring 2011.

[34] H. Trevor, T. Robert, and F. Jerome, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.

[35] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning." *IEEE Trans. Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.

[36] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, and U.-M. O'Reilly, "Genetic programming needs better benchmarks," in *GECCO '12: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, 2012.