Controlling Overfitting in Symbolic Regression Based on a Bias/Variance Error Decomposition

Alexandros Agapitos, Anthony Brabazon, and Michael O'Neill

Financial Mathematics and Computation Research Cluster,
Natural Computing Research and Applications Group,
University College Dublin, Ireland
{alexandros.agapitos,anthony.brabazon,m.oneill}@ucd.ie

Abstract. We consider the fundamental property of generalisation of data-driven models evolved by means of Genetic Programming (GP). The statistical treatment of decomposing the regression error into bias and variance terms provides insight into the generalisation capability of this modelling method. The error decomposition is used as a source of inspiration to design a fitness function that relaxes the sensitivity of an evolved model to a particular training dataset. Results on eight symbolic regression problems show that new method is capable on inducing better-generalising models than standard GP for most of the problems.

1 Introduction

Reliable learning in the field of Machine Learning (ML) revolves around the property of *generalisation*, which is the ability of a learned model to correctly explain data that are drawn from the same distribution as the training data, but have not been presented during the training process. This is the very important property that ML algorithms aim to optimise. The generalisation performance of a model relates to its prediction capability on an independent test dataset. Assessment of this performance guides the choice of a model, and provides a measure of the quality of the ultimately chosen model. The loss of generalisation is referred to as the problem of *overfitting* [4].

In the case of learning regression models, the task is to discover a target function $f(X)$ that map a vector of real-valued inputs X to a real-valued target variable Y. A prediction model $\hat{f}(X)$ is trained on a training dataset $D = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ of size n, where model accuracy on an individual training case is specified using a loss function for measuring the errors between Y and $\hat{f}(X)$ denoted by $L(Y, \hat{f}(X))$. Typical choice is the square error $(Y - \hat{f}(X))^2$, and the error over the entire set D is taken as the average of individual losses. The use of least squares, can lead to severe overfitting if complex regression models are trained over limited-sized datasets [4]. In an example of polynomial curve fitting [4](pages 4-11), model complexity is measured by the order of the polynomial. It is shown that a polynomial of a low order and few coefficients gives poor predictions on test data since the polynomial function has

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 438–447, 2012.

too little flexibility to be learning anything at all during training. On the other hand, a polynomial with too many coefficients has poor generalisation since it fits to closely to the noise on the training data. The issue of model complexity is central to overfitting. There is a trade-off between achieving a good fit to the training data, and obtaining a model which is not very complex, and thus does not overfit. Significant insight into this trade-off can be obtained by introducing the statistical concept of *bias/variance* error decomposition, under which the generalisation error of a model is decomposed into the sum of *bias* squared plus the *variance*. The *bias* measures the accuracy of the estimated $\hat{f}(X)$, while the *variance* measures the extend to which $\hat{f}(X)$ is sensitive to the particular dataset D used during training.

Genetic Programming (GP) [9] tackles regression problems by means of searching a model space for the most appropriate functional model-form along with the optimal coefficients given a training set of input-output pairs. A plethora methods for learning good-generalising models have been investigated in previous research; some are reported in the works of [1,2,3,10,11,12].

In this study, we draw inspiration from the bias/variance error decomposition and devise a method to improve on the sensitivity of an evolved model to a particular training dataset. The method is based on the bootstrap resampling method to randomly draw datasets with replacement from the training data, and calculate the variance of the error on all bootstrap samples. The variance is then used along with the error on the original training dataset in a single-objective fitness function that takes the form of their weighted sum that is to be minimised. Given the two conflicting objectives of *bias* and *variance*, a Pareto-based multi-objective fitness function would be a sensible line of attacking this problem. At this preliminary study, we chose to aggregate the two objectives in a scalar fitness function, and explicitly investigate the effect of different kinds of trade-off for biasing the search towards good-generalising regression models.

The rest of the paper is organised as follows. Section 2 introduces the statistical concept of bias/variance decomposition of regression error. Section 3 presents a new method for relaxing the sensitivity of evolved models to a particular dataset used during training. Section 4 presents the symbolic regression problems that will be used in this study, and details the experiment method. Section 5 analyses the results, and finally Section 6 draws our conclusions.

2 Bias and Variance for Regression

This section presents the basic background on the statistical concept of bias/variance regression error decomposition, and motivates the development of the new method for tackling overfitting. The material is based on the textbook of Bishop [4] (pages 147-152).

Consider we wish to model the underlying generator of a dataset, so that the best possible predictions for the target vector t can be made when a trained model is presented with a new value of the input vector x. For that, we are estimating a model $y(x)$ for a target function $\langle t|x\rangle$ using a training dataset D,

where $\langle t|x\rangle$ denotes the conditional average of the target data, so that $\langle t|x\rangle = \int tp(t|x)dt$. The most general descriptor of the generator of D is in terms of the probability density $p(x,t) = p(t|x)p(x)$ in the joint input-target space. Our training algorithm minimises the sum-of-squares error function, thus each individual error is calculated as $\{y(x) - \langle t|x\rangle\}^2$, and depends on the training dataset D and on the particular datapoint x. Integrating this quantity over x will give the usual sum-of-squares error measure.

Suppose we have a large ensemble of datasets of the same size, each drawn independently from the distribution $p(t,x)$ of D. We can eliminate the dependency of a model on a particular training dataset by measuring the performance of a model using the average of the ensemble of datasets, which we write as:

$$\mathbb{E}_D[\{y(x) - \langle t|x\rangle\}] \tag{1}$$

where $\mathbb{E}_D[\cdot]$ denotes the expectation, or ensemble average, which represents the error of model $y(x)$ when trained over equal-sized samples of D. If the trained model was a perfect predictor of the target function $\langle t|x\rangle$, then this error would be zero. Nevertheless a non-zero error can occur for two distinct reasons. It may be that the estimated model $y(x)$ is different from the target function $\langle t|x\rangle$, which is called the *bias*. Alternatively, it may be that the method is sensitive on the particular sample training dataset, and as a result, at a given x its prediction is either larger or smaller than the target t depending on the dataset used for training. This is called the *variance*. We can decompose Equation 1 into bias and variance using the notion of an *average model* $\mathbb{E}_D[y(x)]$, which is the average of all predictions at point x of various models trained on different samples of D:

$$\begin{aligned}\{y(x) - \langle t|x\rangle\}^2 &= \{y(x) - \mathbb{E}_D[y(x)] + \mathbb{E}_D[y(x)] - \langle t|x\rangle\}^2 \\ &= \{y(x) - \mathbb{E}_D[y(x)]\}^2 \\ &\quad +2\{y(x) - \mathbb{E}_D[y(x)]\}\{\mathbb{E}_D[y(x)] - \langle t|x\rangle\} \\ &\quad +\{\mathbb{E}_D[y(x)] - \langle t|x\rangle\}^2\end{aligned} \tag{2}$$

By taking the expectation of both sides over the ensemble of datasets, we can express the expected squared difference as:

$$\begin{aligned}&\mathbb{E}_D[\{y(x) - \langle t|x\rangle\}^2] = \\ &\underbrace{\{\mathbb{E}_D[\{y(x)] - \langle t|x\rangle\}^2}_{(bias)^2} + \underbrace{\mathbb{E}_D[\{y(x) - \mathbb{E}_D[y(x)]\}^2]}_{variance}\end{aligned} \tag{3}$$

The first term, the bias, measures the extent to which the average model $\mathbb{E}_D[y(x)]$ differs from the target function $\langle t|x\rangle$. The second term, the variance, measures the extent to which a model trained on a specific dataset varies around the average model, and hence measures the sensitivity of a particular model to the particular choice of dataset. There is a trade-off between bias and variance, with very flexible models having low bias and high variance, whereas relatively rigid models having

high bias and low variance. The next section introduces a simple measure that quantifies the sensitivity of a model to a particular training dataset, and presents a new fitness function to relax this sensitivity in pursue of better generalisation.

3 Minimising the Error Variance on Bootstrap Datasets

Suppose we have a model $y(x)$ trained on a dataset $D = \{(x_1, t_1), \ldots, (x_N, t_N)\}$ using an error function that takes the form of mean Canberra distance (C).

$$C(D) = \frac{1}{N} \sum_{i=1}^{N} \frac{abs(y(x_i) - t_i)}{abs(y(x_i)) + abs(t_i)} \tag{4}$$

where *abs* returns the absolute value of its argument, $y(x_i), t_i$ are the predicted and target values respectively for input x_i, and N is the size of D. Canberra distance is preferred to Mean Squared Error (MSE) because it implicitly normalises the output within the [0.0, 1.0] interval, which is necessary for applying weights in the same interval during the aggregation of the objectives into a scalar fitness function.

We employ the bootstrap resampling method [6] to randomly draw B datasets with replacement from D, each sample the same size as D. For each of the bootstrap datasets D^{*b} we calculate the error value $C(D^{*b})$, thus the mean error over all dataset is given by $\overline{C^*} = \sum_{b=1}^{B} C(D^{*b})/B$.

The variance of the error from the bootstrap sampling is then simply:

$$Var(D^*) = \frac{1}{B-1} \sum_{b=1}^{B} (C(D^{*b}) - \overline{C^*})^2 \tag{5}$$

The error variance can be seen as a measure of the sensitivity of a model to the training dataset D, with overfitted models achieving a large error variance on the bootstrap datasets, whereas more general models obtaining a lower error variance. In order to relax the dependence on a particular dataset, a new fitness function to be minimised is defined as:

$$fitness = w_b C(D) + w_v Var(D^*) \tag{6}$$

which is the weighted sum of the mean error on the original dataset plus the variance of error on the bootstrap datasets, and w_b, w_v are the coefficients for error and variance respectively.

It is important to note that previous work investigated a bias/variance decomposition in GP [8] from the point of view of ensemble learning methods like *Bagging*. The output of an *average model* was calculated by averaging the outputs of an ensemble of models so that the expected generalisation error of the ensemble reduced to the bias error alone. In addition, the work of [5] successfully employed a Pareto-based bi-objective fitness function that was based on the MSE and the variance of the independent squared errors over a single training dataset. Our fitness function differs substantially from the one used in [5] in that we calculate the variance over a collection of bootstrap datasets.

4 Experiment Design

We designed a set of experiments to assess the effectiveness of the new method (BVGP) on the generalisation ability of evolved models. The method is contrasted against standard GP (SGP). We used three datasets: *training*, *validation*, and *testing*. The training dataset is used to fit the models. At the end of every generation the best model on training data is stored in an *elitist-list*. At the end of evolution, the validation set is used to estimate the prediction error of every element of the elitist-list in order to perform model selection. The test set is used for assessing the generalisation error of the final chosen model. The size of training and validation sets is the same for a problem, and the three datasets share no common elements. In the case of BVGP, the bootstrap resampling is performed on the training dataset.

Table 2 presents eight symbolic regression problems that are tackled in this work. Problems F_2, F_4, F_5, F_7 were chosen from [13] due to their pronounced difficulty as test problems for GP. Problems F_8, F_9, F_{10}, F_{11} were chosen from [7]. For these four problems we deliberately used small training datasets (20 points) in order to render the GP systems more prone to overfitting.

Table 1 summarises the setup of the GP systems. For the fitness function of BVGP in Equation 6, both $C(D)$ and $Var(D^*)$ are normalised into the same [0.0, 1.0] interval. We considered an exhaustive set of combinations with a step of 0.1 for the w_b and w_v, in order to test the effect of different trade-offs. On the other hand, SGP used Equation 4 as the fitness function. Note that the number of program evaluations are exactly the same in both fitness function calculations. This is because Equation 6 needs to calculate $C(D)$ on the original training dataset D, and afterwards the calculation of every bootstrap $C(D^{*b})$ can be based on the individual losses that were cached during the program evaluation with the training cases of D.

We performed 50 independent evolutionary runs for each GP system on each problem. Statistical significance of the differences in performance is evaluated using the Mann-Whitney U-test, considering a confidence of 95% and a pairwise Bonferroni correction for the value of α.

Table 1. GP systems setup

GP systems under comparison	BVGP, SGP
EA used in GP systems	elitist, generational, expression-tree representation
Function set	$+$, $-$, $*$, / (protected)
Terminal set	Regressor variables, 5 random constants in [0.0, 1.0]
No. of generations	51
Population size	500
Tournament size	4
Tree creation	ramped half-and-half (depths of 2 to 6)
Max. tree depth	20
Subtree crossover	30% (90% inner nodes, 10% leaf-nodes)
Subtree mutation	40%
Point mutation	30%
Fitness function	BVGP: Equation 6 (no. of bootstrap datasets: 500) SGP: Equation 4

Table 2. 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 $[a_1, b_1] \times [a_2, b_2]$. Both training and validation sets are of the same size for a problem.

	Problem	Training/Validation	Test
F_2	$f(x) = e^{-x}x^3cos(x)sin(x)(cos(x)sin^2x - 1)$	100 points x=rand(0.05, 10)	221 points x=(-0.5 : 0.05 : 10.5)
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) x_2=rand(1, 2)	2,701 points x_1, x_3=(-0.05 : 0.15 : 2.1) x_2 = (0.95 : 0.1 : 2.05)
F_5	$f(x_1, x_2) = 6sin(x_1)cos(x_2)$	50 points x_1, x_2=rand(0.1, 5.9)	961 points x_1, x_3=(0.05 : 0.02 : 6.05)
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)	1,157 points x_1, x_2=(-0.25 : 0.2 : 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, 3)	361,201 points x_1, x_2=(-3 : 0.01 : 3)
F_9	$f(x_1, x_2) = {x_1}^4 - {x_1}^3 + {x_2}^2/2 - x_2$	20 points x_1, x_2=rand(-3, 3)	361,201 points x_1, x_2=(-3 : 0.01 : 3)
F_{10}	$f(x_1, x_2) = \frac{8}{2+{x_1}^2+{x_2}^2}$	20 points x_1, x_2=rand(-3, 3)	361,201 points x_1, x_2=(-3 : 0.01 : 3)
F_{11}	$f(x_1, x_2) = {x_1}^3/5 + {x_2}^3/2 - x_2 - x_1$	20 points x_1, x_2=rand(-3, 3)	361,201 points x_1, x_2=(-3 : 0.01 : 3)

5 Results

Table 4 summarises the performance statistics accrued from 50 independent runs of each experiment setup. The median value is preferred over the mean as it is more robust to outliers. The table reports the training Root Mean Squared Error (RMSE) obtained at the end of an evolutionary run, the test RMSE of models selected based on the validation set, the best-generalising model size in terms of number of tree-nodes, and the generation number when model selection took place. For the case of test RMSE the minimum value indicates the best-generalising model out of 50 runs. Table 3 summarises the p-values obtained by comparing the differences in the median test RMSE, median model size, median generation of model selection of BVGP against SGP using the Mann-Whitney U-test.

Observing the training error obtained by the different GP systems, results suggest that for all problems considered, both BVGP and SGP obtained a similar fit during training. Interestingly, the different trade-offs created by the different coefficient combinations did not seem to affect the training accuracy. When comparing the generalisation performance of BVGP against SGP we observe that in six out of eight problems BVGP outperformed SGP. The differences in median test RMSE are statistically significant (Table 3). For the remaining two problems the use of BVGP was deemed equivalent with that of SGP. It

Table 3. The p-values obtained by comparing the differences in the median test RMSE, median model size, median generation of model selection of BVGP against SGP using the Mann-Whitney U-test. Bold face indicates confidence of at least 95%.

		SGP		
		Test RMSE	**Model size**	**Model selection generation**
BVGP	F_2	0.45	0.15	0.36
	F_4	0.49	0.24	0.18
	F_5	**0**	0.22	0.23
	F_7	**0.04**	0.20	0.47
	F_8	**0.0004**	**0**	**0**
	F_9	**0.02**	0.12	0.14
	F_{10}	**0.0034**	0.14	0.31
	F_{11}	**0.001**	0.13	0.08

is important to note that for the problems F_8, F_9, F_{10}, and F_{11}, where small training sets were employed, all systems overfitted the training data. This is evidenced by the large degradation in test error as opposed to the cases of F_2, F_4, F_5, and F_7. However, the use of BVGP system appeared more resilient to overfitting. When inspecting the trade-off between error and variance that is required to enhance model generalisation, we observe no apparent trend to the combination of w_b and w_v coefficients that yields the best generalisation. The trade-off necessary to counteract overfitting appears to be problem dependent. Finally, the minimum test RMSE that is accrued from 50 independent runs of each system configuration suggests that for the majority of problems, BVGP produced the best-generalising model as opposed to SGP.

Early research on the relationship between model size and generalisation has advocated an intrinsic interaction between the two. The sixth column of Table 4 shows the median size of best-generalising models. For all the problems but F_8, we found no statistically significant differences in the sizes of the expression-trees representing the evolved models (Table 3). This is in accordance to the latest findings on the relationship between the expression-tree size and overfitting [5], ascertaining that in light of bloat in variable-length GP representations, model complexity is not directly associated with the number of tree-nodes.

Finally, the generation number when model selection is performed shows the point within an evolutionary run in which overfitting is becoming apparent. Table 4 shows that for the problems of F_2, F_4, F_5, F_7 that use intermediate to large training sets ranging from 50 to 300 data points, there appears to be a relationship between the size of the training set and the speed of generalisation loss. Contrasting between the case of F_4 that used a training set of 300 points against the cases of F_2, F_5, and F_7 that employed smaller training sets (sizes of 50 and 100), we noted that the smaller the training dataset, the quicker the model selection needs to be performed in order to avoid overfitting. On the other hand, for the cases of F_8, F_9, F_{10}, and F_{11} that utilised the smallest training datasets of size 20, this trend is not apparent; all GP systems were allowed to train for longer than those for the problems of intermediate sized training datasets of 50 points. Summarising, for the problems studied, we noted that the cases of very small and large training datasets allowed the models to train for longer before

Table 4. Summary of results. Statistics based on 50 independent runs. *Train RMSE* is the end-of-run training root mean squared error. *Test RMSE* is the generalisation root mean squared error of models selected in each run. *Model size* is the size of the best-generalising models in terms of number of tree-nodes. *Model selection generation* is the generation number when model selection is performed. Highlighting indicates that a statistically significant difference was found in the median generalisation RMSE between BVGP and SGP (Table 3).

Problem	w_b w_v	Train RMSE (median)	Test RMSE (median)	Test RMSE (minimum)	Model size (median)	Model selection generation (median)
F_2	0.2 0.8	0.29	0.32	0.26	15.00	2.00
	0.3 0.7	0.29	0.32	0.25	13.00	2.00
	0.4 0.6	0.29	0.32	0.26	31.00	1.00
	0.5 0.5	0.29	0.32	0.26	15.00	3.00
	0.6 0.4	0.29	0.32	0.27	45.00	3.00
	0.7 0.3	0.29	0.32	0.26	11.00	1.00
	0.8 0.2	0.30	0.32	0.26	15.00	2.00
	SGP	0.29	0.32	0.28	31.00	3.00
F_4	0.2 0.8	0.22	0.25	0.05	83.00	39.00
	0.3 0.7	0.22	0.26	0.04	81.00	40.00
	0.4 0.6	0.21	0.25	0.08	85.00	38.00
	0.5 0.5	0.22	0.26	0.09	87.00	40.00
	0.6 0.4	0.20	0.26	0.06	81.00	42.00
	0.7 0.3	0.23	0.27	0.14	99.00	39.00
	0.8 0.2	0.22	0.27	0.11	57.00	40.00
	SGP	0.19	0.22	0.09	81.00	37.00
F_5	0.2 0.8	3.45	3.03	2.66	19.00	3.00
	0.3 0.7	3.42	3.46	2.68	31.00	3.00
	0.4 0.6	3.61	3.41	2.82	25.00	2.00
	0.5 0.5	3.43	3.57	2.94	19.00	2.00
	0.6 0.4	3.66	3.59	2.36	19.00	2.00
	0.7 0.3	3.39	3.41	2.63	15.00	1.00
	0.8 0.2	3.39	3.75	2.52	29.00	2.00
	SGP	3.52	3.66	2.54	21.00	2.00
F_7	0.2 0.8	1.57	2.00	1.43	15.00	3.00
	0.3 0.7	1.55	2.41	1.77	17.00	4.00
	0.4 0.6	1.56	2.00	1.58	19.00	3.00
	0.5 0.5	1.56	1.97	1.69	15.00	2.00
	0.6 0.4	1.56	2.09	1.53	15.00	3.00
	0.7 0.3	1.58	1.98	1.44	21.00	4.00
	0.8 0.2	1.58	1.97	1.58	19.00	4.00
	SGP	1.58	2.65	1.73	15.00	3.00
F_8	0.2 0.8	0.50	0.68	0.68	17.00	11.00
	0.3 0.7	0.50	0.84	0.68	71.00	16.00
	0.4 0.6	0.50	17.46	0.68	57.00	11.00
	0.5 0.5	0.50	42.16	0.65	91.00	13.00
	0.6 0.4	0.50	17.74	0.63	75.00	14.00
	0.7 0.3	0.50	30.03	0.68	115.00	14.00
	0.8 0.2	0.49	27.52	0.67	123.00	27.00
	SGP	0.50	56.90	0.68	151.00	38.00
F_9	0.2 0.8	0.29	13.28	4.66	67.00	15.00
	0.3 0.7	0.30	12.76	3.03	103.00	14.00
	0.4 0.6	0.29	10.96	2.66	111.00	24.00
	0.5 0.5	0.31	11.02	1.75	97.00	18.00
	0.6 0.4	0.29	48.54	1.55	135.00	24.00
	0.7 0.3	0.29	10.52	2.64	79.00	24.00
	0.8 0.2	0.31	27.90	2.96	99.00	32.00
	SGP	0.28	13.57	2.40	119.00	31.00
F_{10}	0.2 0.8	0.22	94.31	4.13	173.00	47.00
	0.3 0.7	0.20	116.14	1.99	151.00	47.00
	0.4 0.6	0.16	40.22	5.64	167.00	46.00
	0.5 0.5	0.19	32.66	0.97	175.00	46.00
	0.6 0.4	0.19	48.36	3.69	171.00	49.00
	0.7 0.3	0.20	27.27	0.41	143.00	48.00
	0.8 0.2	0.21	85.04	0.77	157.00	43.00
	SGP	0.21	35.36	4.71	161.00	48.00
F_{11}	0.2 0.8	0.61	12.07	1.37	135.00	33.00
	0.3 0.7	0.62	23.50	1.44	127.00	40.00
	0.4 0.6	0.61	5.54	1.69	117.00	20.00
	0.5 0.5	0.63	5.25	1.26	115.00	23.00
	0.6 0.4	0.62	12.81	1.18	117.00	29.00
	0.7 0.3	0.62	18.54	1.44	93.00	26.00
	0.8 0.2	0.63	41.35	2.39	127.00	17.00
	SGP	0.62	16.36	2.32	107.00	34.00

they overfitted, as opposed to the cases of intermediate-sized training datasets, where overfitting was evident very quickly. Whether this is problem dependent remains to be seen in future studies.

6 Conclusions

The decomposition of regression error in bias and variance terms suggests that the generalisation error is due to the degree of difference between the average model (over all datasets) and the target function, as well as the degree of sensitivity of the particular model on the training dataset. We drew inspiration from this error decomposition and devised a method to relax the inherent sensitivity to the data used for training. In this method, boostrapping was employed to create an ensemble of datasets, and the variance of the error on the ensemble was used in combination with the error on the original dataset to form a new fitness function. Results on a suite of symbolic regression problems are encouraging, showing that this method is able to induce better-generalising models for most of the problems considered as opposed to standard GP.

The task of inducing a model from real-world data usually suffers from two problems: the degree to which the underlying data-generating distribution is statistically under-represented, and the degree of noise in the data points. Tackling noisy as well as unbalanced datasets is the immediate plan for future application of our method to classes of ill-defined learning environments.

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

References

1. Agapitos, A., O'Neill, M., Brabazon, A.: Evolutionary Learning of Technical Trading Rules without Data-Mining Bias. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part I. LNCS, vol. 6238, pp. 294–303. Springer, Heidelberg (2010)
2. Agapitos, A., O'Neill, M., Brabazon, A., Theodoridis, T.: Maximum Margin Decision Surfaces for Increased Generalisation in Evolutionary Decision Tree Learning. In: Silva, S., Foster, J.A., Nicolau, M., Machado, P., Giacobini, M. (eds.) EuroGP 2011. LNCS, vol. 6621, pp. 61–72. Springer, Heidelberg (2011)
3. Banzhaf, W., Francone, F.D., Nordin, P.: The Effect of Extensive Use of the Mutation Operator on Generalization in Genetic Programming Using Sparse Data Sets. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN IV. LNCS, vol. 1141, pp. 300–309. Springer, Heidelberg (1996)
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
5. Castelli, M., Manzoni, L., Silva, S., Vanneschi, L.: A comparison of the generalization ability of different genetic programming frameworks. In: IEEE Congress on Evolutionary Computation (CEC 2010), July 18-23. IEEE Press, Barcelona (2010)

6. Efron, B., Tibshirani, R.: An introduction to the bootstrap. Chapman and Hall (1993)
7. Keijzer, M.: Improving Symbolic Regression with Interval Arithmetic and Linear Scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003)
8. Keijzer, M., Babovic, V.: Genetic Programming, Ensemble Methods and the Bias/Variance Tradeoff - Introductory Investigations. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 76–90. Springer, Heidelberg (2000)
9. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk` (2008), `http://www.gp-field-guide.org.uk`, (With contributions by J. R. Koza)
10. Theodoridis, T., Agapitos, A., Hu, H.: A gaussian groundplan projection area model for evolving probabilistic classifiers. In: Genetic and Evolutionary Computation Conference, GECCO 2011, July 12-16. ACM, Dublin (2011) (forthcoming)
11. Tuite, C., Agapitos, A., O'Neill, M., Brabazon, A.: A Preliminary Investigation of Overfitting in Evolutionary Driven Model Induction: Implications for Financial Modelling. In: Di Chio, C., Brabazon, A., Di Caro, G.A., Drechsler, R., Farooq, M., Grahl, J., Greenfield, G., Prins, C., Romero, J., Squillero, G., Tarantino, E., Tettamanzi, A.G.B., Urquhart, N., Uyar, A.Ş. (eds.) EvoApplications 2011, Part II. LNCS, vol. 6625, pp. 120–130. Springer, Heidelberg (2011)
12. Tuite, C., Agapitos, A., O'Neill, M., Brabazon, A.: Tackling Overfitting in Evolutionary-Driven Financial Model Induction. In: Brabazon, A., O'Neill, M., Maringer, D. (eds.) Natural Computing in Computational Finance. SCI, vol. 380, pp. 141–161. Springer, Heidelberg (2011)
13. Vladislavleva, E.J., Smits, G.F., den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation 13(2), 333–349 (2009)