A Study on Fitness Representation in Genetic Programming

Thuong Pham Thi[1,3(✉)], Xuan Hoai Nguyen[2], and Tri Thanh Nguyen[3]

[1] University of Information and Communication Technology, Thainguyen University, Thai Nguyen, Vietnam
ptthuong@ictu.edu.vn
[2] Hanoi University, Hanoi, Vietnam
nxhoai@hanu.edu.vn
[3] VNU University of Engineering and Technology, Hanoi, Vietnam
ntthanh@vnu.edu.vn

Abstract. In this paper, we propose a variation on the fitness function in Genetic Programming based on Bias-Variance Genetic Programming (BVGP) [2], called BVGP*. In order to evaluate the effectiveness of this variation, we compare it with Genetic Programming [1] and Bias-Variance Genetic Programming (BVGP) [2]. The experimental results shown that the learned model by BVGP* is better than that of GP and BVGP in ability to generalize, model complexity and evaluation time.

Keywords: Genetic programming · Bias-Variance Decomposition · Regression problems

1 Introduction

Genetic Programming (GP) is one of evolutionary algorithm-based methodologies inspired by biological evolution. It uses tree-based structures and a suite of defined Genetic Algorithm-operators to generate and evolve a population of solutions to the given problem [3]. GP has produced many novel and outstanding results in various areas such as optimization, searching, sorting, quantum computing, electronic design, game playing, cyberterrorism prevention [4,6]. One of main areas of GP is Machine Learning

In Machine Learning, generalization and over-fitting are two central challenges need to be solved. Generalization error of learners directly relates to over-fitting and is referred to as the problem of over-fitting [7]. There are many researches in Machine Learning, including GP, try to improve generalization ability of learners by reducing over-fitting error as [2,8–12].

Over-fitting can be controlled by Bias-Variance trade-off [2], where bias is the error on training data set and variance is the error of difference on various data sets in the future. Over-fitting will be reduced when bias and variance are small, simultaneously. Because bias and variance are hidden in the L2-norm loss function (e.g. RMSE, MSE, ...). So, many researches in Machine Learning

M. Akagi et al. (eds.), *Advances in Information and Communication Technology*, Advances in Intelligent Systems and Computing 538, DOI 10.1007/978-3-319-49073-1_13

have used these functions [13–16] for learning. However, the combination of bias and variance in the error function L2 sometimes causes difficulties in optimizing them simultaneously because Bias and Variance are two conflicting problems. So, in GP, Alexandros et al. proposed the method (BVGP) [2] to overcome this issue. He divided the fitness function into two components: variance and squared bias which aim at bringing variance component into the evolution process more directly. However, this method faces to the over-fit issue on limited training sample. This leads to reducing the ability to generalize of the learned model. Moreover it can make the model very sensitive to noise.

In this paper, we propose a variation on the fitness function for GP which aims at improving the limits of BVGP as shown above. It is called BVGP*. Through experiments, we demonstrate that the use of BVGP* has some advantages: (1) It can help to reduce over-fitting on the problems that GP was over-fitted; (2) the program runs faster and finds the simpler solution. So, the main contribution of this paper is the variation on the fitness function for improving the effectiveness of GP based on bias-variance decomposition of training errors.

The remainder of this paper is organized as follows: In Sect. 2, we briefly present some background knowledge and related work. A variation on the fitness function is presented in Sect. 3. Section 4 are some experimental settings and problems for testing. Next, experimental results are given in Sect. 5. Finally, Sect. 6, we summarize achieved research results and present some future works.

2 Background and Related Work

2.1 Bias-Variance Decomposition

In this section we introduce the background on the statistical concept of loss function and Bias-Variance Decomposition for regression. The material is based on the book of Trevor Hastie [17].

If we assume that $Y = f(x) + \varepsilon$, where ε is prediction error; $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma_\varepsilon^2$, we can derive an expression for the expected prediction error of $\widehat{f}(x)$ at an input point $X = x_0$, using L2-loss function as follows:

$$\begin{aligned} Err(x_0) &= E[(Y - \widehat{f}(x))^2 | X = x_0] \\ &= \sigma_\varepsilon^2 + [E\widehat{f}(x_0) - f(x_0)]^2 + E[\widehat{f}(x_0) - E\widehat{f}(x_0)]^2 \\ &= \sigma_\varepsilon^2 + Bias^2(\widehat{f}(x_0)) + Var(\widehat{f}(x_0)) \\ &= Irreducible\ Error + Bias^2 + Variance \end{aligned} \tag{1}$$

The first term is the variance of the target around its true mean $f(x_0)$; the second term is the squared bias, the amount by which the average of our estimate differs from the true mean; the last term is the variance; the expected squared deviation of $\widehat{f}(x_0)$ around its mean. The last two terms need to be addressed for a good performance of the prediction model.

Generalization error is the prediction error over an independent test sample:

$$Err(T) = E[L(Y, \widehat{f}(x))|T] \tag{2}$$

where both X and Y are drawn randomly from their joint distribution (population). Here, the training set T is fixed, and test error refers to the error for this specific training set. A related quantity is the expected prediction error:

$$Err = E[L(Y, \widehat{f}(x))] = E[Err_T] \tag{3}$$

Such decomposition is known as the Bias Variance Decomposition.

2.2 Bias-Variance Genetic Programming (BVGP)

The Bias-Variance Genetic Programming proposed by Alexandros et al. is a new method for over-fitting issue based on Bias/Variance Error Decomposition which aims at relaxing the sensitivity of an evolved model to a particular training dataset. This method used the fitness function that is the combination of bias and variance as follows:

$$fitness = w_b Bias(D) + w_v Var(D^*) \tag{4}$$

where w_b, w_v are the coefficients for error and variance respectively; D is the training data set of size n; D^* includes B bootstrap datasets randomly drawn from D by the bootstrap re-sampling method; $Bias(D)$ is the mean error on the original dataset (bias); $Var(D^*)$ is variance of the error on the bootstrap datasets.

He separated regression error into two components: bias and variance to put variant error in the evolution process more directly.

3 The Improved Method: BVGP*

In this section, we propose a variation on the fitness function for GP which aims at overcoming the disadvantage of the BVGP. This function is based on BVGP and defined as follows:

$$fitness = w_b Bias(D^*) + w_v Var(D^*) \tag{5}$$

where bias and variance are calculated using the bootstrap re-sampling method. We consider $f(x)$ as the model trained on a dataset $D = \{(x_1, t_1), ..., (x_N, t_N)\}$ and use the bootstrap re-sampling method to randomly draw B datasets with replacement from D, each sample the same size as D. We denote D^* to include B the bootstrap sample sets: $D^* = \{D^{*b}, b : 1..B\}$. The estimated bias ($\mu$) and variance ($\sigma^2$) of stochastic fitness are computed as follows:

$$Bias(D^*) = \sum_{b=1}^{B} Bias^{*b}/B \tag{6}$$

where $Bias^{*b}$ is the bias of bootstrap sample D^{*b} is calculated using the error function RMSE:

$$Bias^{*b} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x_i) - t_i)^2} \tag{7}$$

So, here we use $Bias(D^*)$ rather than the mean error on the original dataset $Bias(D)$.

$$\sigma^2 = \frac{1}{B-1}\sum_{b=1}^{B}(Bias^{*b} - Bias(D^*))^2 \tag{8}$$

As shown in [5], given a data sample, statistical inference is the process to assess how systems will behave in untested situations. It permits generalizations of conclusions beyond the sample, about an unseen population from which the sample is drawn. This process is inference from statistics to parameters, where statistics are functions on samples and parameters are functions on populations. It is noted that, $Bias(D)$ is a statistic on D while $Bias(D^*)$ is a parameter inferred from this statistic. The bootstrap re-sampling method is used to construct empirical sampling distributions for parameter estimation $Bias(D^*)$ without making any troubling assumptions about sampling models and population distributions. BVGP* learns to optimize the fitness function based on $Bias(D^*)$ while the fitness function of BVGP is based on $Bias(D)$. Therefore, we believe that the generalization ability of BVGP* is better than that of BVGP. The experimental results have confirmed this is true with most of the problems to be tested.

4 Experimental Setting

4.1 Problems

In this paper, we used benchmarks in [2] as shown in Table 1. Besides, we also used three more UCI data sets as shown in Table 2 to test the generalization ability of BVGP*. With UCI data sets, we divide an original dataset into two parts randomly: $\langle$Train sample:Test sample$\rangle = \langle 1:2 \rangle$.

4.2 GP System Setup

Evolutionary parameter values for GP systems are shown in Table 3. These typical settings are often used by GP researchers and practitioners [1].

5 Results and Discussion

In this section we present results of comparing the performance of BVGP* in comparison with GP, BVGP. We evaluate the effectiveness of BVGP* on three aspects: (1) Generalization ability; (2) Model complexity; and (3) Time complexity.

Table 1. GP benchmark regression problems

ID	Name	Definition	Training data	Testing data
BEN_1	F4	$30\frac{(x_1-1)(x_3-1)}{x_2^2(x_1-10)}$	x_1 : U[0.05, 2, 200] x_2 : U[1, 2, 200] x_3 : U[0.05, 2, 200]	x_1 : E[-0.05, 2.1, 0.15] x_2 : E[0.95, 2.05, 0.1] x_3 : E[-0.05, 2.1, 0.15]
BEN_2	F5	$6\sin(x_1)\cos(x_2)$	U[0.1, 5.9, 200]	E[−0.05, 6.05, 0.02]
BEN_3	F7	$\frac{e^{(x_1-1)^2}}{1.2+(x_2-2.5)^2}$	U[0.3, 4, 200]	E[−0.2, 4.2, 0.01]
BEN_4	F8	$x_1x_2+\sin((x_1-10)(x_2-1))$	U[−3, 3, 200]	E[−3, 3, 0.01]
BEN_5	F9	$x_1^4-x_1^3+\frac{x_2^2}{2}-x_2$	U[−3, 3, 200]	E[−3, 3, 0.01]
BEN_6	F10	$\frac{8}{2+x_1^2+x_2^2}$	U[−3, 3, 200]	E[−3, 3, 0.01]
BEN_7	F11	$\frac{x_1^3}{5}+\frac{x_2^3}{2}-x_2-x_1$	U[−3, 3, 200]	E[−3, 3, 0.01]

Table 2. UCI data sets

ID	Name	No. of attributes	#train samples	#testing samples
UCI_1	Census6	6	133	267
UCI_2	No2	7	167	333
UCI_3	SkillCraft1_Dataset	19	1114	2224

Table 3. GP systems setup

Paramaters	GP	BVGP	BVGP*
Problems	See Tables 1 and 2		
EA used in GP systems	Elitist, generational,		
	expression tree representation		
Function set	+, −, *, / (PD)		
Terminal set	Regression variables and		
	one random constant in [0.0, 1.0]		
No. of generations	151		
Population size	500		
Tournament size	4		
Tree creation	Ramped half-and-half		
	(depths of 2 to 6)		
Max. tree depth	15		
Sub tree crossover rate	0.9		
Sub tree mutation rate	0.1		
No. of Runs	100		
Fitness function	RMSE		
No. of bootstrap datasets		30	30
wb, wv		0.7, 0.3	0.7, 0.3

5.1 Generalization Error (Fittest)

In this section, we repeated one hundred runs independently for each GP system. Generalization error is the median of testing error of the best individual from all these runs. The Table 4 shows the generalization error or testing error (fittest) GP, BVGP and BVGP*, bold values indicate that the corresponding method is the best result. We see that with most of problems (BEN_1, BEN_2, BEN_3, BEN_4, BEN_5, BEN_7, UCI_1) fittest error of BVGP* is smaller than that of GP and BVGP or generalization ability of BVGP* is better than that of GP and BVGP. However, with UCI_2, generalization ability of BVGP* is much worse than that of GP and BVGP. The cause can be the learned model by BVGP is under-fit on this problem.

It is noted that both the GP and BVGP use the bias on the original training dataset ($Bias(D)$) as the optimal goal, this lead to over-fitting when the size of the training sample is limited or there is noise in the train data or the sampling process is bad. BVGP* rather than using $Bias(D)$, it uses the mean of the empirical bootstrap error distribution ($Bias(D^*)$) as one of the optimal goals. So, it can avoid the sampling bias issues that lead to over-fitting solution as showed above. This explains why the results by BVGP* are better than those of GP and BVGP in most of problems.

5.2 Model Complexity and Evaluation Time

In this section, we repeated one hundred runs independently for each GP system. Generalization error is the median of testing error of the best individual from these runs. The Table 5 shows the evaluation time and model complexity of the best individual by GP, BVGP, BVGP*. Bold values indicate that the corresponding method is the best.

Table 4. Summary of fittest error (median). Statistics based on 100 independent runs. Bold values indicate that the method is the best.

Problem	GP	BVGP	BVGP*
BEN_1	1.404	1.6015	**1.347**
BEN_2	387.5605	425.2415	**262.844**
BEN_3	10.054	9.0615	**3.7185**
BEN_4	134.748	87.4175	**0.7**
BEN_5	486.8505	458.1465	**450.7025**
BEN_6	2.788	**0.8295**	0.85
BEN_7	623.178	854.623	**580.851**
UCI_1	0.1955	0.1952	**0.195**
UCI_2	1.5175	**1.248**	580.851
UCI_3	15.002	**15**	22

Table 5. Evaluation time (median), model complexity (median) is the average number of nodes on the best individual. Statistics based on 100 independent runs. Bold values indicate the method is the best.

Problem	Evaluation time			Model complexity		
	GP	BVGP	BVGP*	GP	BVGP	BVGP*
BEN_1	372.50	269.50	**262.50**	204.89	**197.44**	198.49
BEN_2	112.50	68.00	**65.50**	293.57	292.53	**252.92**
BEN_3	70.00	34.00	**3.50**	155.76	151.95	**4.56**
BEN_4	53.00	14.00	**2.00**	218.64	192.74	**4.91**
BEN_5	59.50	**17.00**	18.50	244.66	240.60	**239.26**
BEN_6	44.00	**2.00**	4.00	21.60	**3.21**	91.15
BEN_7	52.50	15.02	**15.00**	**207.94**	209.42	208.10
UCI_1	64.00	4.50	**4.00**	13.71	3.32	**1.40**
UCI_2	162.50	129.00	**17.00**	251.90	267.11	**208.10**
UCI_3	140.00	47.21	**47.00**	37.89	17.89	**17.39**

Here, the evaluation time is measured in milliseconds. It is effected mainly by model complexity. Similar to fittest error, in all problems (10/10 problems, see bold lines), BVGP* is faster than GP since it leaned the smaller model (see corresponding lines at the column Evaluation time). Comparing to BVGP, BVGP* also learned the model with smaller complexity with most of problems (7/10 problems, see bold lines), so it is faster than BVGP or evaluation time is smaller. It is noted that, on BEN_6, the model complexity of BVGP* is larger

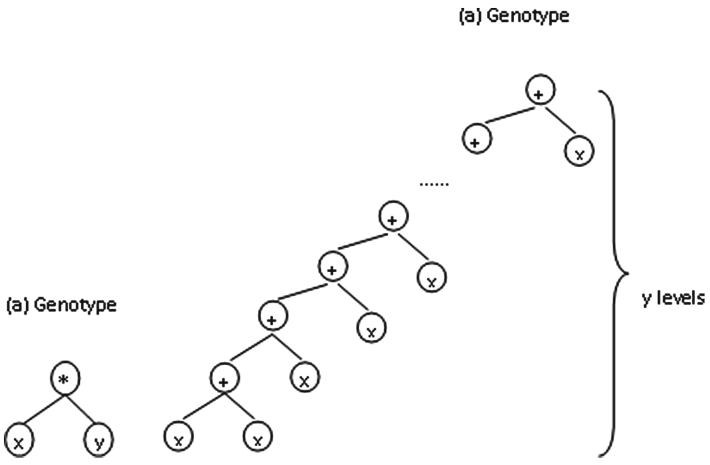

Fig. 1. The evaluation time of genotype (a) is similar to that of genotype (b), but their different model complexities are different.

while the its evaluation time is smaller than that of other methods. It can be caused by genotype of the learned model by BVGP* contains various operators that effect to evaluation time, i.e., considering two genotypes as shown in Fig. 1, although the model complexities of them are different, the evaluation time of them are similar.

6 Conclusion and Future Work

In this paper, we proposed the variation on the fitness function (BVGP*). It is based on the bias-variance decomposition and the method BVGP. Analyses of empirical results show that this approach has some advantages: (1) BVGP* can help reduce over-fitting on the problems that GP and BVGP were over-fitted; (2) It runs faster with simpler solution.

There are several future research directions arisen from this paper. First we need a more natural fitness representation way in bringing two components directly into the process of evolution. Second, we need a new selection mechanism corresponding to this fitness representation.

References

1. Koza, J.R.: Genetic Programming: on the Programming of Computers by Means of Natural Selection. MIT press, Cambridge (1992)
2. Agapitos, A., Brabazon, A., O'Neill, M.: Controlling overfitting in symbolic regression based on a bias/variance error decomposition. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012. LNCS, vol. 7491, pp. 438–447. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32937-1_44
3. Cramer, N.L.: A representation for the adaptive generation of simple sequential programs. In: Proceedings of the First International Conference on Genetic Algorithms, pp. 183–187 (1985)
4. Nordin, P.: Genetic programming iii-darwinian invention and problem solving. Evol. Comput. **7**, 451–453 (1999)
5. Cohen, P.R.: Empirical Methods for Artificial Intelligence, vol. 139. MIT press, Cambridge (1995)
6. Hansen, J.V., Lowry, P.B., Meservy, R.D., McDonald, D.M.: Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. Decis. Support Syst. **43**, 1362–1374 (2007)
7. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
8. Fitzgerald, J., Azad, R., Ryan, C.: A bootstrapping approach to reduce over-fitting in genetic programming. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 1113–1120. ACM (2013)
9. Gonçalves, I., Silva, S., Melo, J.B., Carreiras, J.M.B.: Random sampling technique for overfitting control in genetic programming. In: Moraglio, A., Silva, S., Krawiec, K., Machado, P., Cotta, C. (eds.) EuroGP 2012. LNCS, vol. 7244, pp. 218–229. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29139-5_19

10. Gonçalves, I., Silva, S.: Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A.Ş., Hu, B. (eds.) EuroGP 2013. LNCS, vol. 7831, pp. 73–84. Springer, Heidelberg (2013). doi:10.1007/978-3-642-37207-0_7
11. Nguyen, T.H., Nguyen, X.H., McKay, B., Nguyen, Q.U.: Where should we stop? An investigation on early stopping for GP learning. In: Bui, L.T., Ong, Y.S., Hoai, N.X., Ishibuchi, H., Suganthan, P.N. (eds.) SEAL 2012. LNCS, vol. 7673, pp. 391–399. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34859-4_39
12. Uy, N.Q., Hien, N.T., Hoai, N.X., O'Neill, M.: Improving the generalisation ability of genetic programming with semantic similarity based crossover. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 184–195. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12148-7_16
13. Muttil, N., Chau, K.-W.: Neural network and genetic programming for modelling coastal algal blooms. Int. J. Environ. Pollut. **28**, 223–238 (2006). Inderscience Publishers
14. Archetti, F., Lanzeni, S., Messina, E., Vanneschi, L.: Genetic programming for human oral bioavailability of drugs. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 255–262. ACM (2006)
15. Juang, C.-F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans. Syst. Man Cybern. Part B Cybern. **34**, 997–1006 (2004)
16. Whigham, P.A., Crapper, P.F.: Time series modelling using genetic programming: an application to rainfall-runoff models. In: Advances in Genetic Programming, vol. 3, pp. 89–104. MIT Press, Cambridge (1999)
17. Hastie, T., Tibshirani, R., Friedman, J., Franklin, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, New York (2005). The Mathematical Intelligencer, 27, 83–85. Springer