Nonlinear Parametric Regression in Genetic Programming

Yung-Keun Kwon
School of Computer Science & Engineering
Seoul National University
Sillim-dong, Gwanak-gu,
Seoul, 151-744 Korea
kwon@soar.snu.ac.kr

Sung-Soon Choi
School of Computer Science & Engineering
Seoul National University
Sillim-dong, Gwanak-gu,
Seoul, 151-744 Korea
sschoi@soar.snu.ac.kr

Byung-Ro Moon
School of Computer Science & Engineering
Seoul National University
Sillim-dong, Gwanak-gu,
Seoul, 151-744 Korea
moon@soar.snu.ac.kr

ABSTRACT

Genetic programming has been considered a promising approach for function approximation since it is possible to optimize both the functional form and the coefficients. However, it is not easy to find an optimal set of coefficients by using only non-adjustable constant nodes in genetic programming. To overcome the problem, there have been some studies on genetic programming using adjustable parameters in linear or nonlinear models. Although the nonlinear parametric model has a merit over the linear parametric model, there have been few studies on it. In this paper, we propose a nonlinear parametric genetic programming which uses a nonlinear gradient method to estimate parameters. The most notable feature in the proposed genetic programming is that we design a parameter attachment algorithm using as few redundant parameters as possible.

Categories and Subject Descriptors

I.1 [**Symbolic and Algebraic Manipulation**]: Applications

General Terms

Performance

Keywords

Genetic programming, system identification, mathematical modeling/curve fitting

1. INTRODUCTION

The function approximation is finding a function that best explains the relationship between independent variables and a dependent variable. Usually, many approaches assume a function model for approximation and estimate the model parameters. On the other hand, genetic programming (GP) is a model-free approach. The symbolic regression using GP involves not only the functional model but the numeric parameters also. The existing methods can be categorized in two models: the linear and the nonlinear parameter models. The advantage of linear parameter model is that parameters are estimated fast and easily by using least squares method. However, the linear model often fails to represent a nonlinear function completely and it just approximates the nonlinearity. Another important problem is that a linear model has to search larger tree structures although the underlying function is a simple nonlinear function. This places a significant burden on GP since it has to find relatively larger building blocks.

On the other hand, nonlinear parameter model is efficient in tree representation; a tree of small size can represent a sufficiently complex function. There have been some studies on nonlinear GP model and they use nonlinear optimization methods to estimate parameters. It is important to insert adjustable parameters efficiently since the nonlinear gradient algorithm for parameter estimation affected by the number of parameters and initial values of them. In this paper, we propose an efficient algorithm to attach adjustable parameters in nonlinear genetic programming. It determines whether a parameter is attached or not depending on the function types of a node and its children nodes.

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

2. THE PROPOSED GP

Two sets have to be defined to construct trees in GP: the terminal node set and non-terminal node set (function node set). We use the independent variables for terminal nodes. The non-terminal node set consists of mathematical functions: $\times$, $\div$, $+$, $-$, sin, cos, tan, and inv.

The selection operation is performed in probabilistic proportion to fitness. Crossover occurs on two parent trees A and B. The crossover operation is performed by randomly selecting an arbitrary node from each tree (say v_a and v_b, respectively) and exchanging the two subtrees rooted at v_a and v_b. Mutation is performed by picking a random node, deleting the corresponding subtree, and replacing it with a randomly generated subtree.

To evaluate a functional tree, we first attach a set of parameters to the tree in an efficient way. When a tree T is given, it can be represented as a sequence of nodes in order by depth-first search. Then, we define two arrays, P and Q, whose sizes are equally the number of nodes in T. P_i and Q_i, which are the i^{th} elements in P and Q, respectively, mean whether or not a multiplication parameter and an addition parameter are attached to the subtree T_i, the i^{th} element in T. In other words, if f_i is a function corresponding to the subtree rooted at T_i, $\widetilde{f}_i$, a function model by parameter

```
1.  AttachParameter(T, P, Q, i) {
2.     if i = 1 { // initialize P and Q
3.        P_j, Q_j ← false;   (j = 1, ..., #(T))
4.        Q_1 ← true; // for a constant term
5.     }
6.     if T_i is an independent variable return;
7.     else if T_i is a unary operator {
8.        AttachParameter(T, P, Q, i + 1);
9.        if TypeOfNode(T_{i+1}) = nonbinary
10.          P_{i+1}, Q_{i+1} ← true;
11.       else Q_{i+1} ← true;
12.    } else { // a binary operator
13.       AttachParameter(T, P, Q, i + 1);
14.       AttachParameter(T, P, Q, i + 1 + #(T_{i+1}));
15.       switch (T_i) {
16.       case ×:
17.          if TypeOfNode(T_{i+1})=nonbinary
18.             and TypeOfNode(T_{i+1+#(T_{i+1})})=nonbinary
19.          P_i ← true;
20.          break;
21.       case ÷:
22.          if TypeOfNode(T_{i+1})=nonbinary
23.          P_i ← true;
24.          if TypeOfNode(T_{i+1+#(T_{i+1})})=nonbinary
25.          P_{i+1+#(T_{i+1})}, Q_{i+1+#(T_{i+1})} ← true;
26.          else Q_{i+1+#(T_{i+1})} ← true;
27.          break;
28.       case + or −:
29.          if TypeOfNode(T_{i+1})=nonbinary
30.          P_{i+1} ← true;
31.          if TypeOfNode(T_{i+1+#(T_{i+1})})=nonbinary
32.          P_{i+1+#(T_{i+1})} ← true;
33.          break;
34.       }
35.    }
36. }
37. TypeOfNode(T_i) {
38.    if T_i is a unary operator or an independent variable
39.       return nonbinary;
40.    else return binary;
41. }
```

Figure 1: Parameter Attachment Algorithm

attachment using P and Q, is defined as follows:

$$\widetilde{f}_i = \begin{cases} p_i T_i(\widetilde{g_i^1}, \ldots, \widetilde{g_i^l}) + q_i & \text{if } P_i = true \text{ and } Q_i = true \\ T_i(\widetilde{g_i^1}, \ldots, \widetilde{g_i^l}) + q_i & \text{if } P_i = false \text{ and } Q_i = true \\ p_i T_i(\widetilde{g_i^1}, \ldots, \widetilde{g_i^l}) & \text{if } P_i = true \text{ and } Q_i = false \\ T_i(\widetilde{g_i^1}, \ldots, \widetilde{g_i^l}) & \text{if } P_i = false \text{ and } Q_i = false \end{cases}$$

where $g_i^j (j = 1, \ldots, l)$ is a function corresponding to the subtree rooted at each child node of T_i.

From the definition of P and Q, we need an efficient algorithm for specifying elements of P and Q. Figure 1 shows the algorithm proposed in this paper. Given a tree T, we identify P and Q by calling `AttachParameter(T, P, Q, 1)`. After the functional model is determined by the parameter attachment algorithm, all the parameters are initialized with random values for Levenberg-Marquardt algorithm, a derivate-based algorithm. It updates the parameters until there is hardly any update and the mean squared error is computed for the fitness.

3. EXPERIMENTAL RESULTS

We compare our approach with Koza-style GP and maximal parametric GP as shown in Table 1. "Koza-GP" is a GP that uses ephemeral random constant nodes without adjustable coefficients. "Max GP" is a GP attaching all the possible multiplication and addition parameters, and estimating them using Levenberg-Marquardt algorithm. The table shows the average and standard deviation of MSE over 50 trials. Our GP outperformed both the Koza-GP and Max GP in all the data set remarkably. We additionally analyze the influence of parameters attachment. We randomly generate 1000 solutions (trees) and attach the parameters to them by the maximal attachment and our proposed algorithm, respectively. Table 2 shows the result. In the table, P and U mean the average number of attached parameters and the average number of updating a set of coefficients in Levenberg-Marquardt algorithm, respectively. R is the ratio of the inefficient trees over the total generated trees. An inefficient tree is defined to be a tree producing a function whose MSE is abnormally large. As expected, the number of attached parameters by our approach is fewer than that by the maximal attachment approach. On the other hand, the number of updates is not different in both the approaches. However, the maximal attachment makes a tree to an inefficient function form in a much higher probability than our approach.

Table 1: Performance Comparison (50 trials)

	Koza-GP		Max GP		Our GP	
	mean	std.	mean	std.	mean	std.
BHR	32.19	4.84	28.99	12.88	**26.08**	9.84
CHF$^{\alpha}$	7.83	0.34	5.27	0.27	**4.87**	0.24
FRD	20.79	2.38	18.85	1.00	**16.80**	0.95
MPG	17.43	2.65	10.08	2.60	**9.06**	0.63
OZN	34.13	4.69	20.30	1.48	**18.45**	1.16
SER	1.08	0.16	0.70	0.09	**0.55**	0.11
IZW	2.96	1.07	1.81	0.28	**1.59**	0.10

Table 2: Efficiency of Parameter Attachment

	Maximal Attachment			Proposed Attachment		
Data	P	U	R	P	U	R
BHR	16.27	9.00	0.200	11.82	8.80	**0.011**
CHF	16.41	9.29	0.153	11.72	9.15	**0.011**
FRD	16.83	9.13	0.147	12.08	9.24	**0.003**
MPG	16.24	8.84	0.193	11.35	8.55	**0.016**
OZN	16.20	8.96	0.219	11.42	8.49	**0.034**
SER	17.24	9.18	0.151	12.05	9.28	**0.001**
IZW	15.84	9.18	0.216	10.53	8.82	**0.021**

4. CONCLUSIONS

In this paper, we proposed a genetic programming for nonlinear function approximation. In the GP, it attaches a set of parameters in the way of reducing as many redundant parameters as possible. Our approach showed a significant performance improvement over the traditional GP without the adjustable parameters. It also outperformed the GP with the maximally attached parameters. The former reduces the probability of producing inefficient functional forms by reducing redundant coefficients.

Acknowledgments

This work was supported by the Brain Korea 21 Project in 2006. This was also partly supported by grant No. (R01-2003-000-10879-0) from the Basic Research Program of the Korea Science and Engineering Foundation. The ICT at Seoul National University provided research facilities for this study.