Solving Classification and Curve Fitting Problems Using Grammatical Evolution

Passent M. El-Kafrawy

Department of Mathematics, Faculty of Science, Menoufia University, Egypt
passentmk@gmail.com

Abstract. Grammatical Evolution is related to the idea of genetic programming in that the objective is to find an executable program or function. GE offers a solution by evolving solutions according to a user specified grammar (Backus-Naur Form). In this paper GE is used to construct a classifier for some well known datasets. and curve fitting problems without the need to assume the equation shape.

Keywords: Classification, Curve fitting, Grammatical Evolution, Computational methods, Least Squares error, Backus-Naur Form.

1 Introduction

Grammatical Evolution (GE) is an extension of genetic programming in that it is an algorithm for evolving complete programs in an arbitrary language. Classification is one of the most researched questions in machine learning and data mining. The learning process in classification consists of predicting the value of the outputs from the value of the inputs as a supervised technique [12]. The goal of the classification algorithm is to find relationships between the values of the predictors and the values of the target. Different classification algorithms are employed whereas all of them represent the problem as a model, which can then be applied to different input sets in which the class assignments are unknown. The goal of regression is to find the line or curve that best predicts the values of dependent value of (Y) from the value of independent values of (X). Regression does this by finding the line or curve that minimizes the sum of the squares of the vertical distances of the points from the line or curve. Although regression technique has no understanding of the scientific context of the experiment that brings the data but it is useful in some situations when a smooth curve is required, without the need for a model. In regression techniques, the curve's equation has to be predefined to find the equation's parameters, [3]. Least-Squares is a well-known curve fitting method for a long time. The Least Squares method minimizes the square of the error between the original data and the values predicted by the equation. While this technique may not be the most statistically robust method of fitting a function to a data set, it has the advantage of being relatively simple and of being well understood. The major weakness of the Least Square method is its sensitivity to outliers in the data. For this reason, the data should always be examined for reasonableness before fitting, [5, 11]. The method of least squares assumes that the best-fit curve of a given type is the curve that has the minimal sum of the deviations squared (least square error) from a given set of data [5]. Suppose that the data points are (x_1, y_1), (x_2, y_2) … (x_n, y_n) where x is the independent variable and y is the dependent variable. The fitting curve f(x) has the deviation (error) d from each data point, d1 = y1 – f(x1), d2 = y2 – f(x2)…dn = yn– f(x_n). According to the least squares method, best fitting curve has the property that:

$$\Pi = d_1^2 + d_2^2 + \ldots + d_n^2 = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = \text{aminimum}$$

The goal of this paper is to use Grammatical Evolution to develop models of equations for curve fitting and classification. In Section 2 we introduced Grammatical Evolution and its methodology. In section 3 we presented solution of some curve fitting problems using Grammatical Evolution. Section 4 presents a solution for classification using GE, and finally in section 5 conclusions and some future work.

2 Grammatical Evolution

By utilizing a Backus Naur Form (B.N.F) grammar the advantages of defining the problem is achieved as well as a separation of genotype and phenotype, [9, 10].In Grammatical Evolution a Backus Naur Form (B.N.F) grammar is used to map the genotype to the phenotype. A separation of genotype and phenotype allows the implementation of various operators (for instance by crossover and mutation). The genotype in Grammatical Evolution is a sequence of bits, [9]. Grammatical Evolution presents a unique way of using grammar in the process of automatic programming, [7]. Variable length binary string genomes are used with each codon presenting an integer value where codons are consecutive groups of 8-bits. The integer values are used in a mapping function to select an appropriate production rule from the B.N.F. definition, the numbers generated always representing one of the rules that can be used at that time, [4, 7, 8]. To solve any problem using GE, a suitable B.N.F. definition must first be developed, [7, 8].

A Backus Naur Form (B.N.F) Grammar

B.N.F is a notation for expressing the grammar of a language in the form of production rules, [1, 4, 7]. A grammar can be represented by the tuple {N,T,P,S}, where N is the set of non-terminals, T the set of terminals , P is a set of production rules that maps the elements of N to T, and S is a start symbol that is a member of N. In a production rule, when there is a number of productions that can be applied to one particular N, the choice is delimited with the '|' symbol. In GE, the B.N.F. definition is used to describe the output language that is to be produced by the system, i.e. the compilable code produced will consist of elements of the terminal set T. The B.N.F. is a plug in component of the system, it means that GE can produce code in any language there by giving the system a unique flexibility, [7].

B Grammatical Evolution Methodology

GE is an automatic programming system similar to genetic programming (GP), in that it uses an evolutionary process to automatically generate computer programs. GE uses a population of linear genotypic binary strings, which are transformed into functional phenotypic programs, through a genotype-to-phenotype mapping process. This transformation is governed through the use of a B.N.F. grammar [6]. GE methodology consists of two important parts. The first part is mapping process and the second part is evolution algorithm that is described in the following two subsections. In the GE *mapping process* the genotype maps the start symbol onto terminals by reading codon of 8 bits to generate a corresponding integer from which an appropriate production rule is selected by using the following mapping function, [1, 4, 6, 7, 8, 9]. Rule = (codon integer value) MOD (number of rules for the current non-terminal). The evolutionary algorithm evolves over the population that comprises a simple binary strings. We do not have to employ any special crossover or mutation operators and an unconstrained search is performed on these strings due to the genotype-to-phenotype mapping process that will generate syntactically correct individuals. The Evolutionary Algorithm (EA) adopted in this case is a variable-length genetic algorithm. Individual initialization is achieved by randomly generating variable-length binary strings within a pre-specified range of codons, [7].

3 Curve Fitting and Grammatical Evolution

B.N.F. for curve fitting problem is

<prog> ::= <expr>

<expr> ::= (<expr> <op> <expr>) | <protected-op> | <pre-op>|<digit> | <var>

<op> ::= -| + | * <protected-op> ::= div(<expr>, <expr>)

<pre-op> ::= sin | cos | log | exp <var> ::= x

<digit> :: = -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |7 |8 | 9

Example 1

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	0	0.3	0.4	0.55	0.63	0.71	0.77	0.84	0.89	0.95	1

Figure 1. show the graphical drawing of the curves in every generation and figure 2. show the graphical drawing of the best curve of best function that produced from GE.

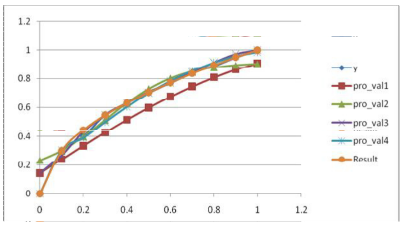

Fig. 1. Graphical drawing of curves in generations

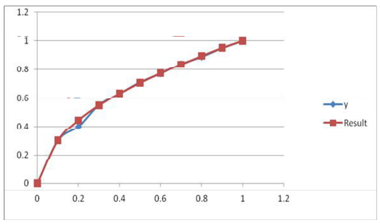

Fig. 2. Graphical drawing of the best curve

Example 2

x	-2	-1.6	-1.2	-0.8	-0.4	0	0.4	0.8	1.2	1.6	2
y	0	0	0	0.05	0.3	0.57	0.3	0.05	0	0	0

Figure 3. is a graphical representation of the curves in every generation and figure 4. presents the best curve of the best function that is produced from GE.

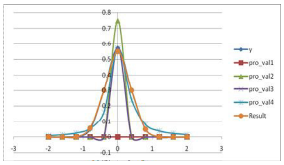

Fig. 3. Graphical drawing of curves in generation

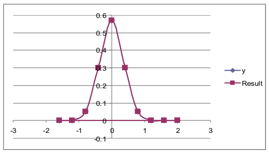

Fig. 4. Graphical drawing of the best curve

4 Classification with GE

The goal of classification is to take an input vector X and assigns it to one of K discrete classes Ck where k=1, 2, 3,…, K, [3, 8]. In our methodology we try to find a mathematical formula that defines a classifier for a problem. Suppose that we need to convert a class name or label to class value. For example, if the problem contains three classes (A, B, and C) we replace them with (1, 2, and 3) to use in the mathematical function. In the next step, we will define each record in the data set by real number, like (2.002) the integer value (2) refers to a class number and the fractional number (002) refers to the record number representing this class. By this method we can convert the class labels to class values.

x1	x2	x3	Class name	Class value
0	2	3	A	1.001
1	1	1	A	1.002
0	3	2	B	2.001
2	4	5	B	2.002

The following B.N.F. grammar is used for extracting the mathematical function.

```
<prog> ::= <expr>
<expr> ::= (<expr><op><expr>) | <protected-op> | <pre-op>|<digit> | <var>
<op> ::= -| + | *                          <protected-op> ::= div( <expr>, <expr>)
<pre-op> ::= sin | cos | log | exp         <var> ::= x1|x2|x3
<digit> :: = -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Note; <var> defined by the attributes from classification problem.

Table 1. Comparing the proposed algorithm with other well known classification algorithms

Datasets	Proposed	Simple BaysNB	Logitboost NB	Bayesian Net
Iris	78.70%	95.53%	94.87%	93.20%
Monk1	100.00%	73.38%	85.33%	73.46%
Monk2	70.60%	56.83%	59.92%	56.78%
Monk3	80.56%	93.45%	91.87%	93.45%
Haberman	72.88%	75.06%	71.48%	71.57%

The results shows that GE achieved higher accuracy in Monk1 and Monk2 and the other algorithms achieved higher in Monk3, Iris, and Haberman because they have a large number of attributes and thus the training process takes time. Also, The data in Monk3 is not filtered, whereas, Monk1 is filtered so that the accuracy reached 100

5 Conclusion and Future Work

In this paper we used Grammatical Evolution (GE) to solve curve fitting problems. Grammatical Evolution successes in solving curve fitting problems. Moreover, this paper proposed a method for the classification problem. We used grammatical evolution (GE) to extract a mathematical formula to define a classifier. This method succeeds in many problems but it takes long time in the training process when the problem contains a large number of attributes. In future work we would try to decrease the training time by using a parallel technique. In future we can benefit from this idea to solve different problems by Grammatical Evolution.

References

1. Cetinkaya, A.: Regular Expression Generation through Grammatical Evolution. In: GECCO 2007, London, England, United Kingdom, July 7-11, pp. 2643–2646 (2007)
2. Elseth, G., Baumgardner, K.: Principles of Modern Genetics. West, St. Paul (1995)
3. Kamal, H.A., Eassa, M.H.: Solving Curve Fitting problems using Genetic Programming. In: IEEE MELECON 2002, Cairo, Egypt, May 7-9, pp. 316–321 (2002)
4. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments. Springer, Heidelberg (2009)
5. Wolberg, J.: Data Analysis Using the Method of Least Squares. Springer (2006)
6. Nicolau, M., Dempsey, I.: Introducing Grammar Based Extensions for Grammatical Evolution. In: 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, July 16-21, pp. 648–655 (2006)
7. O'Neill, M., Ryan, C.: Grammatical Evolution. IEEE Transactions on Evolutionary Computation 5(4), 349–358 (2001)
8. O'Neill, M., Ryan, C.: Under the hood of Grammatical Evolution. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, July13-17, vol. 2, pp. 1143–1148. Morgan Kaufmann (1999)
9. Harper, R., Blair, A.: Dynamically Defined Functions In Grammatical Evolution. In: 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, July 16-21, pp. 2638–2645 (2006)
10. Matousek, R.: Grammatical Evolution: STE criterion in Symbolic Regression Task. In: Proceedings of the World Congress on Engineering and Computer Science, WCECS 2009, San Francisco, USA, October 20-22, vol. II (2009)
11. Kolb, W.M.: Curve Fitting for programmable Calculators. IMTEC in Bowie, Md. (1983)
12. Espejo, P.G., Ventura, S., Herrera, F.: A Survey on the Application of Genetic Programming to Classification. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews 40(2), 121–144 (2010)