Genetic Programming for Attribute Construction in Data Mining

Fernando E. B. Otero[1] Monique M. S. Silva[1] Alex A. Freitas[2] Julio C. Nievola[1]

[1] Pontificia Universidade Catolica do Parana (PUC-PR), Postgraduate Program in Applied Computer Science, Rua Imaculada Conceicao 1155, Curitiba – PR. 80215-901. Brazil
{fbo,mmonique,nievola}@ppgia.pucpr.br

[2] Computing Laboratory, University of Kent
Canterbury, Kent, CT2 7NF, UK
A.A.Freitas@ukc.ac.uk
http://www.cs.ukc.ac.uk/people/staff/aaf

Abstract. For a given data set, its set of attributes defines its data space representation. The quality of a data space representation is one of the most important factors influencing the performance of a data mining algorithm. The attributes defining the data space can be inadequate, making it difficult to discover high-quality knowledge. In order to solve this problem, this paper proposes a Genetic Programming algorithm developed for attribute construction. This algorithm constructs new attributes out of the original attributes of the data set, performing an important preprocessing step for the subsequent application of a data mining algorithm.

1 Introduction

This paper addresses the classification task of data mining [3]. In this task the goal of a data mining algorithm is to predictive the class of an example (a record, or data instance), given the values of a set of attributes for that example.

For a given data set, its set of attributes defines its data space representation. The quality of a data space representation is one of the most important factors influencing the performance of a data mining algorithm. The attributes defining the data space can be inadequate, making it difficult to discover high-quality knowledge. However, when the original attributes are individually inadequate, it is often possible to combine them in order to construct new attributes with greater predictive power than the original attributes, facilitating the discovery of knowledge with a high predictive accuracy.

This paper proposes a Genetic Programming (GP) algorithm developed for attribute construction (also called constructive induction). This algorithm constructs new attributes out of the original attributes of the data set, performing an important preprocessing step for the subsequent application of a data mining algorithm.

The main motivation for developing a GP algorithm for this task is that it performs a global search in the space of candidate solutions (new constructed attributes, in our case). In data mining, this has the advantage of coping better with attribute interaction, being less likely to get trapped into local maxima in the search space, by comparison with greedy, local search-based data mining algorithms [2], [4].

The remainder of this paper is organized as follows. Section 2 reviews attribute construction. Section 3 proposes our new GP algorithm for attribute construction. Section 4 reports computational results. Finally, section 5 concludes the paper.

2 A Review of Attribute Construction

The majority of inductive learning algorithms for the classification task discover rules (or another kind of knowledge representation) involving only the original attributes of the data being mined. In addition, the majority of rule induction methods analyze the data on a one-attribute-at-a-time basis. Hence, the performance of these methods is considerably limited by the predictive power of individual attributes, so that these methods do not cope very well with the problem of attribute interaction.

The goal of an attribute construction method is to construct new attributes out of the original ones, transforming the original data representation into a new one where regularities in the data are more easily detected by the classification algorithm, which tends to improve the predictive accuracy of the latter.

Attribute construction methods can be roughly divided into two groups, with respect to the construction strategy: hypothesis-driven methods and data-driven methods [6].

Hypothesis-driven methods construct new attributes out of previously-generated hypotheses (discovered rules or another kind of knowledge representation). In general they start by constructing a hypothesis, for instance a decision tree, and then examine that hypothesis to construct new attributes [13]. These new attributes are then added to the set of original attributes, and a new hypothesis is constructed out of this extended set of attributes. The new hypothesis is used to generate new attributes, and so on. This process is repeated until a given stopping criterion is satisfied, such as a satisfactory extended set of attributes has been found. Note that the performance of this strategy is strongly dependent on the quality of the previously-discovered hypotheses.

By contrast, data-driven methods do not suffer from the problem of depending on the quality of previous hypotheses. They construct new attributes by directly detecting relationships in the data. Two examples of data-driven attribute construction methods are GALA and GPCI.

GALA [8] constructs new attributes using two logical operators, AND and OR. First, all original attributes are transformed into boolean attributes. Then it generates new attributes by using the AND and OR operators to produce combinations of the boolean attributes. Although GALA does not use any evolutionary algorithm to construct attributes, it is interesting to note that apparently it was the “parent” of GPCI [7], a GP algorithm for attribute construction.

Like GALA, GPCI starts by transforming all original attributes into boolean attributes, and then it generates new attributes by using AND and OR operators to produce combinations of the boolean attributes. The difference is that is searches for new attributes by using a GP algorithm. In essence, each individual of GPCI repre-

sents a new attribute. The terminal set consists of the booleanized original attributes, whereas the function set consists of the AND and OR operators.

The process of attribute construction can also be roughly divided into two approaches, namely the interleaving approach and the preprocessing approach.

In the preprocessing approach the process of attribute construction is independent of the inductive learning algorithm that will be used to extract knowledge from the data. In other words, the quality of a candidate new attribute is evaluated by directly accessing the data, without running any inductive learning algorithm. In this approach the attribute construction method performs a preprocessing of the data, and the new constructed attributes can be given to different kinds of inductive learning methods.

By contrast, in the interleaving approach the process of attribute construction is intertwined with the inductive learning algorithm. The quality of a candidate new attribute is evaluated by running the inductive learning algorithm used to extract knowledge from the data, so that in principle the constructed attributes' usefulness tends to be limited to that inductive learning algorithm. An example of an attribute construction method following the interleaving approach can be found in [15].

In this paper we follow the data-driven strategy and the preprocessing approach, mainly for two reasons. First, using this combination of strategy/approach the constructed attributes have a more generic usefulness, since they can help to improve the predictive accuracy of any kind of inductive learning algorithm. Second, an attribute-construction method following the preprocessing approach tends to be more efficient than its interleaving counterpart, since the latter requires many executions of an inductive learning algorithm.

It should be noted that both GALA and GPCI also follow the data-driven strategy and the preprocessing approach. However, both these algorithms have the limitation that all attributes have to be booleanized in a preprocessing step, before the attribute construction method starts to run. Intuitively, this booleanization can lead to a significant loss of relevant information. Our proposed GP for attribute construction (described in the next section) does not have this disadvantage, since it does not require any booleanization of the original attributes.

3 A New GP for Attribute Construction

3.1 Individual Representation

We use a standard tree-structure representation for each individual [9], [1]. The GP constructs new attributes out of the continuous (real-valued) attributes of the data set being mined. Each individual corresponds to a candidate new attribute. The terminal set consists of all the continuous attributes in the data being mined. The function set consists of four arithmetic operators, namely "+", "–", "∗", "%" (where the latter is protected division [9]), and two relational comparison operators, namely "≤", "≥".

The use of these operators in the tree associated with an individual must satisfy some constraints about the data types of these operators, as shown in Table 1. As can be seen in the table, the arithmetic operators require two continuous input arguments

and produce a continuous output. The relational comparison operators also require two continuous input arguments, but they produce a boolean output.

Table 1. Data types restrictions for the function set

Operator	Input arguments	Output
$+, -, *, \%$	(continuous, continuous)	(continuous)
$\geq, \leq$	(continuous, continuous)	(boolean)

As a result of the data type restrictions shown in Table 1, there are some restrictions on the hierarchy of nodes in a tree. These restrictions are shown in Table 2. Each cell of this table indicates whether or not (Y or N, respectively) the corresponding combination of parent node and child node is allowed. Note that the relational comparison operators ("$\geq$", "$\leq$") cannot be used as child nodes. I.e., these operators can be used only in the root node of a tree.

Table 2. Restrictions on the hierarchy of nodes in the tree

		parent node					
		+	−	*	%	≥	≤
child node	+	Y	Y	Y	Y	Y	Y
	−	Y	Y	Y	Y	Y	Y
	*	Y	Y	Y	Y	Y	Y
	%	Y	Y	Y	Y	Y	Y
	≥	N	N	N	N	N	N
	≤	N	N	N	N	N	N
	terminal	Y	Y	Y	Y	Y	Y

We have also used a restriction on the size of the tree associated with an individual. This restriction consists of specifying a parameter representing the maximum tree size (number of nodes) of a tree. As will be seen later, we have done experiments with different values of this parameter, to determine how robust our GP is to variations in the setting of this parameter.

This size restriction is important for at least two reasons. First, from a predictive data mining view point, avoiding the generation of very large trees helps to combat overfitting and so potentially improves the predictive power of the candidate attribute. Second, from a GP viewpoint, this size restriction helps to avoid the effects of code bloat [12], [10] – i.e., the tendency of GP trees to grow in an uncontrolled manner.

It should be mentioned that some GP algorithms specify a predefined maximum depth for an individual's tree. There is no such maximum tree depth in our GP. Rather, we have preferred to specify a maximum tree size in terms of the number of nodes. The rationale for this choice is that, if a maximum tree depth is specified, the GP will probably be somewhat biased to produce balanced trees, growing the trees in width after the maximum tree depth has been reached. Such bias seems unnatural,

limiting the flexibility of the GP to search for solutions (trees) of different shapes. Actually, there is some evidence that predefining a maximum tree depth has some negative effects in GP [5], [11]. A size restriction based on the total number of nodes, rather than tree depth, helps to avoid this kind of problem.

Note that the function set of our GP is inclusive enough to allow the GP to construct either a continuous attribute or a boolean attribute, depending on the kind of operator used in the root node of the individual's tree. If the root node contains an arithmetic operator ("+","–","∗", or "%") the constructed attribute will be continuous, whereas if the root node contains a relational comparison operator ("≥" or "≤") the constructed attribute will be boolean.

3.2 Selection Method and Genetic Operators

We use tournament selection. In essence, this method works as follows. First, *k* individuals are randomly chosen from the population. Then the individual with the best fitness is selected. This method has an important parameter, the tournament size, *k*. This parameter determines the selective pressure of the method. Larger values of *k* correspond to larger selective pressures, favouring individuals with the best values of fitness. As will be seen later, we have done experiments with different values of this parameter, to determine how robust our GP is to variations in the setting of this parameter.

In order to create a new population from the current population we use three operators, namely reproduction, crossover and mutation. Reproduction and crossover are conventional GP operators – we use standard tree crossover [9].

The mutation operator works as follows. First, it randomly chooses a tree node. Then the current symbol in this node is replaced by a randomly chosen symbol of the same kind which is different from the current symbol. More precisely, a terminal symbol is replaced by another terminal symbol, an arithmetic operator is replaced by another arithmetic operator, and a relational comparison operator is replaced by another relational comparison operator.

3.3 Fitness Function

The fitness function used in this work is information gain ratio [14], which is a well-known attribute-quality measure in the data mining and machine learning literature. It should be noted that the use of this measure constitutes a data-driven strategy. As mentioned above, an important advantage of this kind of strategy is that it is relatively fast, since it avoids the need for running a data mining algorithm when evaluating an attribute (individual). In particular, the information gain ratio for a given attribute can be computed in a single scan of the training set.

The Information Gain Ratio of an attribute *A*, denoted by IGR(*A*), is computed by dividing the Information Gain of *A*, denoted by IG(*A*), by the amount of Information of the attribute *A*, denoted I(*A*), i.e.:

$$IGR(A) = IG(A) / I(A) \quad . \qquad (1)$$

The Information Gain of an attribute *A*, denoted IG(*A*), represents the difference between the amount of Information of the goal (class) attribute *G*, denoted I(*G*), and that amount given the knowledge of the values of an attribute *A*, denoted I(*G*|*A*). IG(A) is given by:

$$IG(A) = I(G) - I(G|A), \quad (2)$$

where

$$I(G) = - \sum_{j=1}^{n} p(G_j) \cdot \log_2 p(G_j), \quad (3)$$

and

$$I(G|A) = \sum_{i=1}^{m} p(A_i) \left(- \sum_{j=1}^{n} p(G_j|A_i) \cdot \log_2 p(G_j|A_i) \right), \quad (4)$$

where $p(G_j)$ is the estimated probability (computed in the training set) of observing the *j*-th class (i.e., the *j*-th value of the goal attribute *G*), *n* is the number of classes, $p(A_i)$ is the estimated probability of observing the *i*-th value of the attribute *A*, *m* is the number of values of the attribute *A*, and $p(G_j|A_i)$ is the empirical probability of observing the *j*-th class conditional on having observed the *i*-th value of the attribute *A*.
Finally, I(*A*) is given by:

$$I(A) = - \sum_{i=1}^{m} p(A_i) \cdot \log_2 p(A_i) \ . \quad (5)$$

Recall that the attribute constructed by the GP can be either continuous or boolean (see the last paragraph of section 3.1). When the constructed attribute is boolean, the above formulas are used in a straightforward manner to compute the IGR of the attribute. When the constructed attribute is continuous the GP computes the IGR associated with each possible cut point (attribute value) defining a candidate booleanization of the attribute, and it chooses the largest value of IGR, among all those IGR values, as the value to be assigned to IGR(*A*). For more details about this procedure and the information gain ratio measure in general, see [14].

4 Computational Results

In this section we report the results of computational experiments performed to evaluate our proposed GP for attribute construction. The experiments were performed with four public-domain data sets from the UCI (*University of California at Irvine*) data set repository, available at: http://www.ics.uci.edu/~mlearn/MLRepository.html.

Table 3 shows the main characteristics of the data sets used in our experiments. In the third column, the number before the slash ("/") is the number of continuous attributes, whereas the number after the slash is the total number of attributes in the data set. As can be seen in the table, for these experiments we have chosen data sets where all or almost all attributes were continuous. Future work will involve data sets with mixed kinds of attributes (both continuous and categorical ones).

Table 3. Data sets used in the experiments

Data set	No. of records	No. of Cont. Attrib. / Total No. of Attrib.	No. of classes
Abalone	4177	7 / 8	28
Balance-scale	625	4 / 4	3
Waveform	5000	21 / 21	3
Wine	178	13 / 13	3

In all our experiments the probabilities of reproduction, crossover and mutation were 10%, 80% and 10%, respectively. The initial population was created by using the well-known ramped half-and-half method [9]. The population size was 600, and the GP evolved for 100 generations. We made no attempt to optimize the parameters mentioned in this paragraph. However, we did experiments with different values of two other parameters, the tournament size and the maximum tree size, as will be discussed below.

The evaluation of the quality of the attributes constructed by our GP was performed by using C4.5 [Quinlan, 1993], a very well-known classification algorithm that builds a decision tree. Hence, in the experiments we compare the classification error rate of C4.5 using only the original attributes with the error rate of C4.5 using not only the original attributes but also the new attribute constructed by the GP.

The error rate was computed by a well-known 10-fold cross-validation procedure, which essentially works as follows. First, the data set is divided into 10 mutually exclusive and exhaustive partitions. Then the algorithm is run 10 times. In the *i*-th run, i=1,...,10, the *i*-th partition is used as the test set and the remaining 9 partitions are grouped and used as the training set. Finally, the reported result is the average error rate (in the test set) over the 10 runs.

In addition to the goal of evaluating the quality of the attributes constructed by the GP, our experiments also had the goal of determining how robust the GP is to variations in the setting of two important parameters, namely the tournament size k and the maximum tree size (number of nodes). Hence, we did experiments with three different values for the tournament size k (2, 4, 8) and three different values for the maximum tree size (31, 63, 127). The experiments involved all the 9 possible combinations (3 x 3) of values of these two parameters.

For each of those 9 combinations of parameter values we have run a 10-fold cross-validation procedure, as explained above. Note that each run of an entire cross-validation procedure involved running GP 10 times and running C4.5 20 times (10 times using only the original attributes and 10 times using both the original attributes and the new attribute constructed by the GP).

The results are reported in Tables 4, 5, 6, and 7, for the Abalone, Balance-Scale, Waveform and Wine data sets, respectively. In these tables, each cell contains the error rate obtained by C4.5 using the attribute constructed by GP, with the combination of parameter values corresponding to that cell. In the second line of the title of each table, between brackets, we report the error rate obtained by C4.5 using only the original attributes. In these tables, the value of a cell is shown in bold if the error rate obtained using the new attribute constructed by the GP is smaller than the error rate

obtained using only the original attributes. Therefore, cells in bold represent cases where the attribute constructed by the GP was useful to reduce the error rate associated with the original attributes, being evidence of the good quality of the constructed attribute. The numbers after the "±" symbol denote standard deviations.

In the Abalone data set (Table 4) the use of the new attribute constructed by the GP has led to a slight increase in error rate in 6 out of the 9 cases (combinations of parameter values), and has led to a slight reduction in the error rate in only 3 cases. However, in general the differences in error rates are not significant, since the corresponding error rate intervals (considering the standard deviations) overlap. (The very high error rates obtained by C4.5, with and without the attribute constructed by the GP, show that this data set represents a very difficult classification problem. This seems to be at least in part due to the relatively large number of classes, 28.)

In the Balance-Scale data set (Table 5) the use of the new attribute constructed by the GP has clearly led to a very significant reduction in the error rate in all the 9 combinations of parameter values.

In the Waveform data set (Table 6) the use of the new attribute constructed by the GP has led to a reduction in the error rate in 8 of the 9 combinations of parameter values. These reductions in error rate are significant – the corresponding error rate intervals (considering the standard deviations) do not overlap.

In the Wine data set (Table 7) the use of the new attribute constructed by the GP has led to a reduction in the error rate in all the 9 combinations of parameter values. However, these reductions in error rate are not significant.

With respect to the ability of the attributes constructed by the GP in reducing the error rate associated with the original attributes, one can summarize the above results as follows. In one data set (Abalone) the attribute constructed by GP has led to a slight increase in error rate, but this increase was not significant. In another data set (Wine) the attribute constructed by GP has lead to some reduction in error rate, but again this reduction was not significant. In the other two data sets (Balance-Scale and Waveform) the attribute constructed by GP has led to a reduction in error rate which was significant, since the corresponding error rate intervals, considering the standard deviations, do not overlap. The reduction in error rate was particularly strong in the Balance-Scale data set. Overall, we consider these results quite promising.

Recall that another goal of our experiments was to determine how robust the GP is to variations in the setting of two important parameters, namely the tournament size k and the maximum tree size (number of nodes).

First of all, as expected, there is no combination of parameters values that turned out to be the "best" one for all data sets. This is a common result in data mining and machine learning, where the best parameter setting is strongly dependent on the data being mined. In any case, the GP turned out to be quite robust to variations in the parameters k and maximum tree size. There are, of course, a few exceptions. In particular, in the Waveform data set (Table 6) the combination of $k = 8$ and maximum number of nodes = 127 led to an error rate significantly larger than the other combinations of parameter values. But this result is clearly an exception. In general, in all the four data sets the differences in error rates associated with different combinations of parameter values was not significant, which is evidence of a reasonable robustness of GP to variations in these two parameters.

Table 4. Error rate (%) of C4.5 using both original attributes and attribute constructed by the GP in Abalone data set (Error rate (%) of C4.5 using only the original attributes: 79.2 ± 0.37)

		Maximum tree size (number of nodes)		
		31	63	127
Tournament size	2	79.31 ± 0.36	**79.18 ± 0.35**	79.21 ± 0.41
	4	79.21 ± 0.33	79.21 ± 0.33	**79.16 ± 0.36**
	8	79.21 ± 0.33	79.21 ± 0.33	**79.16 ± 0.36**

Table 5. Error rate (%) of C4.5 using both original attributes and attribute constructed by the GP GP in Balance-Scale data set (Error rate (%) of C4.5 using only the original attributes: 22.42 ± 1.34)

		Maximum tree size (number of nodes)		
		31	63	127
Tournament size	2	**11.47 ± 2.13**	**7.78 ± 0.66**	**8.58 ± 0.58**
	4	**9.06 ± 0.42**	**8.26 ± 0.65**	**8.26 ± 0.44**
	8	**8.58 ± 0.67**	**8.74 ± 0.68**	**8.10 ± 0.63**

Table 6. Error rate (%) of C4.5 using both original attributes and attribute constructed by the GP in Waveform data set (Error rate (%) of C4.5 using only the original attributes: 25.06 ± 0.66)

		Maximum tree size (number of nodes)		
		31	63	127
Tournament size	2	**23.04 ± 0.40**	**22.48 ± 0.45**	**22.68 ± 0.57**
	4	**22.44 ± 0.59**	**22.86 ± 0.50**	**22.56 ± 0.57**
	8	**22.22 ± 0.49**	**22.60 ± 0.42**	28.86 ± 2.74

Table 7. Error rate (%) of C4.5 using both original attributes and attribute constructed by the GP in Wine data set (Error rate (%) of C4.5 using only the original attributes: 6.48 ± 2.05)

		Maximum tree size (number of nodes)		
		31	63	127
Tournament size	2	**5.31 ± 1.38**	**4.72 ± 1.47**	**4.72 ± 1.47**
	4	**3.54 ± 1.30**	**5.31 ± 1.63**	**3.54 ± 1.30**
	8	**3.54 ± 1.30**	**4.72 ± 1.47**	**5.30 ± 1.62**

5 Conclusions and Future Research

We have proposed a new GP for attribute construction. It constructs new attributes out of the original continuous (real-valued) attributes of the data being mined. We have also evaluated the ability of the attributes constructed by the GP in reducing the error rate associated with the original attributes. This was done by comparing the error rate obtained by C4.5 using only the original attributes with the error rate obtained by C4.5 using both the original attributes and the new attribute constructed by the GP.

Experiments were performed with four public-domain data sets. In two of those data sets there was no significant difference in the error rate of C4.5 with and without the attribute constructed by the GP. However, in the other two data sets the difference was significant, and in both of these cases the error rate of C4.5 with the constructed attribute was smaller than the error rate of C4.5 without the constructed attribute. Hence, these can be considered promising results.

In addition, GP turned out to be quite robust to variations in the settings of two important parameters, namely the tournament size and the maximum tree size (number of nodes). Experiments with 9 different combinations for the values of these two parameters showed that, in general (with a few exceptions), the differences in error rates associated with different combinations of these parameter values was not significant.

References

1. Banzhaf, W.; Nordin, P.; Keller, R.E.; Francone, F.D. Genetic Programming ~ an Introduction: On the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann, 1998.
2. Dhar, V.; Chou, D. and Provost, F. Discovering Interesting Patterns for Investment Decision Making with GLOWER – A Genetic Learner Overlaid With Entropy Reduction. Data Mining and Knowledge Discovery 4(4), 251-280. Oct. 2000.
3. Fayyad, U. M.; Piatetsky-Shapiro, G; Smith, P.; Uthurusamy, R. (Eds) Advances in Knowledge Discovery and Data Mining, 1-34. AAAI/MIT Press, 1996.
4. Freitas, A.A. Understanding the crucial role of attribute interaction in data mining. Artificial Intelligence Review 16(3), Nov. 2001, pp. 177-199.
5. C. Gathercole and P. Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. Genetic Programming 1996: Proc. 1st Annual Conf., 291-296. MIT Press, 1996.
6. Hu, Y-J. A Genetic Programming Approach to Constructive Induction . In Proceeding of 3rd Anual Genetic Programming Conference, pp. 146–151, 1998.
7. Hu, Y–J. Constructive Induction: Covering Attribute Spectrum. In: H. Liu & H. Motoda (Eds) Feature Extraction Construction and Selection, pp. 257–272. Kluwer, 1998.
8. Hu, Y-J & Kibler, D. Generation of Attributes for Learning Algorithms. In Proceeding of the 13th National Conference on Artificial Intelligence, pp. 806–811, 1996
9. Koza, J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.
10. W.B. Langdon. Quadratic bloat in genetic programming. Proc. 2000 Genetic and Evolutionary Computation Conf. (GECCO-2000), 451-458. Morgan Kaufmann, 2000.
11. Langdon, W.B. & Poli, R. An analysis of the MAX problem in genetic programming. Genetic Programming 1997: Proc. 2nd Annual Conf., 222-230. Morgan Kaufmann, 1997.
12. Langdon, W.B.; Soule, T.; Poli, R. and Foster, J.A.. The evolution of size and shape. In: L. Spector, W.B. Langdon, U-M. O'Reilly and P.J. Angeline. (Eds.) Advances in Genetic Programming Volume 3, 163-190. MIT Press, 1999.
13. Pagallo, G. & Haussler, D. Boolean Feature Discovery in Empirical Learning. In Machine Learning 5, pp. 71–99. 1990.
14. Quinlan, J.R. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
15. Zheng, Z. Constructing X-of-N attributes for decision tree learning. Machine Learning 40 (2000), 1-43.