

EMPLOYING NOMINAL ATTRIBUTES IN CLASSIFICATION USING GENETIC PROGRAMMING

Thomas Loveard and Vic Ciesielski

School of Computer Science and Information Technology
RMIT University
Melbourne, Vic., Australia, 3000
toml,vc@cs.rmit.edu.au

ABSTRACT

In this paper methods for performing classification using Genetic Programming (GP) on datasets with nominal attributes are developed and evaluated. The two methods developed included the splitting of GP program execution based upon the value of a nominal attribute (execution branching), and the conversion of a nominal attribute to a continuous or binary attribute (numeric conversion). These two methods of using nominal attributes are tested against six datasets containing either nominal and continuous attributes or nominal only attributes.

Results show that the use of the methods developed in this paper allow classifiers trained with GP to perform accurate classification of datasets containing nominal attributes. When compared to other well-known methods of classification the GP method is capable of classifying one of six datasets more accurately than any of the conventional methods tested, and accuracy close to the best achieved method on 3 other datasets.

1. INTRODUCTION

The task of classification forms a large and varied class of supervised learning problems which pervade our world in great abundance. In such a problem a set of examples, each belonging to one of a distinct number of possible classes, are used to train a classifier to distinguish between examples of such classes. Classification tasks which involve nominal attributes (attributes for which there are a known and distinct number of possible values, sometimes referred to as categorical or symbolic attributes) form an important category of classification tasks.

One method that has previously been used to perform classification tasks for continuous attributes is Genetic Programming (GP) [3, 5, 6, 7]. Such systems evolve computer programs in an evolutionary process similar to that seen in natural systems [4]. As computer programs offer a diverse and flexible representational structure they can be eas-

ily adapted to a diverse range of complex problems. Previous investigations in applying GP to classification have used numerically structured programs that fit well with datasets containing continuous attributes only. This is because the internal functions that such GP programs employ (mathematical based arithmetic and comparison operators) tend to be well suited to numeric arguments. A problem arises in the inclusion of nominal attributes in such structures as nominal attributes do not lend themselves to the mathematical arithmetic and comparison operations used by the programs (eg. for a numeric attribute "colour" you cannot evaluate "red + green" or "orange <= blue").

Given that these accurate GP based classifiers following the method developed by [7] have returned continuous output, and all internal structures of the programs within a GP system use continuous information, a problem arises when attempting to classify datasets that contain nominal attributes. Such attributes do not intuitively fit into programs with only numeric arithmetic and comparison operations and modifications must be made to the tree structure so that they may be incorporated.

The aim of this research is to determine whether methods can be developed by which to incorporate nominal attributes into a GP program tree of a predominantly numeric structure. As a secondary aim, such classifiers should be capable of using nominal attributes without degrading the performance of the classifiers over continuous attributes. Specifically this investigation aims to evaluate two separate approaches and to compare these to other well-known methods of classification for nominal attributes to evaluate the comparative performance of the GP based classifiers.

2. ENCODING METHODS

GP programs for classification have previously employed functions and terminals that operate on continuous data and these methods have shown to be capable of producing accurate classifiers [6, 7]. Because of this it was considered

important to maintain these structures so that classification using continuous attributes could still be performed accurately and the subsequent addition of nominal attributes to the classification methodology did not detract from the methods ability to perform accurate classification on continuous data. As a result a standard set of functions and terminals were adopted that have previously shown to be capable of accurately classifying continuous attribute data [6]. The terminal set consists of randomly generated constants in the range [-1,1] and continuous attribute terminals (if continuous attributes are present for the given dataset). The function set consists of the mathematical operator set +, -, *, % and the comparison operators If_LTE, If_GTE, If_Equal.to. From this base, two methods of incorporating nominal attributes into a GP tree were considered, both of which maintained the use of the continuous terminals and functions listed above.

2.1. Numeric Conversion Methods

One simple way to incorporate nominal attributes into a GP program structure that primarily uses continuous operations is to immediately convert nominal attributes into continuous data at the terminal (leaf) nodes of the program tree. This method has the advantage that no modifications to the internal structure of programs need to be made, and only the addition of a terminal, to convert nominal attributes into continuous data is required. Two methods of performing this conversion were tested, and resulted in the use of two separate terminals.

The first approach tested was to enumerate a nominal attribute into a stepped numerical attribute. The approach by which this is performed is that of labelling each distinct nominal value of a given attribute with an integer value (starting at zero and incrementing by one for each value). This method resulted in the addition of the Numeric_Conversion terminal. An example of such a terminal can be seen as can be seen in Figure 1A for a nominal attribute “colour” with 3 possible values. In this example the terminal will return a value of 0, 1 or 2, depending on the value of the “colour” attribute.

This simplistic method used by the Numeric_Conversion terminal has the unfortunate effect of giving an implied ordering to values in a nominal attribute where no ordering information is necessarily present. It does however offer the GP system the convenience of an attribute that is purely numeric and which can subsequently be compared, manipulated and operated upon by the functions of the GP system that have been shown to be capable of producing accurate classifiers for numeric only datasets.

An alternative approach to the conversion to a continuous attribute is the conversion of such an attribute to a binary (one or zero) value based upon the attribute value. The Binary_Conversion terminal was used to perform this conver-

sion (see Figure 1B for the three alternative possible terminals for the “colour” attribute). In this approach, one of the possible values for the given nominal attribute was selected as the terminal’s decision value. For a given example of data, if the value of the attribute matched the decision value then the (numeric) value of one was returned. In all other cases the value of zero was returned. The returned numeric values could then be incorporated into the GP programs and used with other numeric only functions to perform classification.

This method has the advantage of being able to single out one possible value of an attribute that might be useful for distinguishing between classes and also does not suffer from the disadvantage of inferring a ranking, as is the case with the Numeric_Conversion terminal.

2.2. Execution Branching

Similar to branching components found in decision tree structures for nominal attributes this method of utilising nominal attributes splits the program execution into several branches. Depending on the value of a given nominal attribute, only one out of a possible set of branching arguments will be executed, and thus the program tree is able to utilise the value of a nominal attribute to alter the values returned by the program and each example can be then classified based on the differing program outputs.

Within the method of execution branching two alternative functions were encoded. The first such function, called the Attribute_Split function, took a set of arguments equal in number to the number of possible values of the associated nominal attribute. When executed the function returned the argument branch associated with the value of the attribute for the current data example that was being classified. An example of such a function can be seen in Figure 1C. When the function is executed it will return a value from only one branch, which is, in this example, the branch corresponding to the value of the “colour” attribute for the given example.

A second execution branching function was termed the Binary_Split function. Similar to the Attribute_Split function this function returns one, out of a possible set of numeric valued argument branches, based on the value of a nominal attribute. The method differs from the Attribute_Split function in that it will only take two numeric arguments, regardless of the number of possible values of the nominal attribute. Instead of branching for all possible values of the attribute it instead singles out one specific value to determine which of the two branches to return. The three possible Binary_Split functions for the “colour” attribute can be seen in Figure 1D.

The Binary_Split function should have the advantage of producing narrower trees that more precisely focus on key values of attributes, rather than the selection of an attribute as a whole.

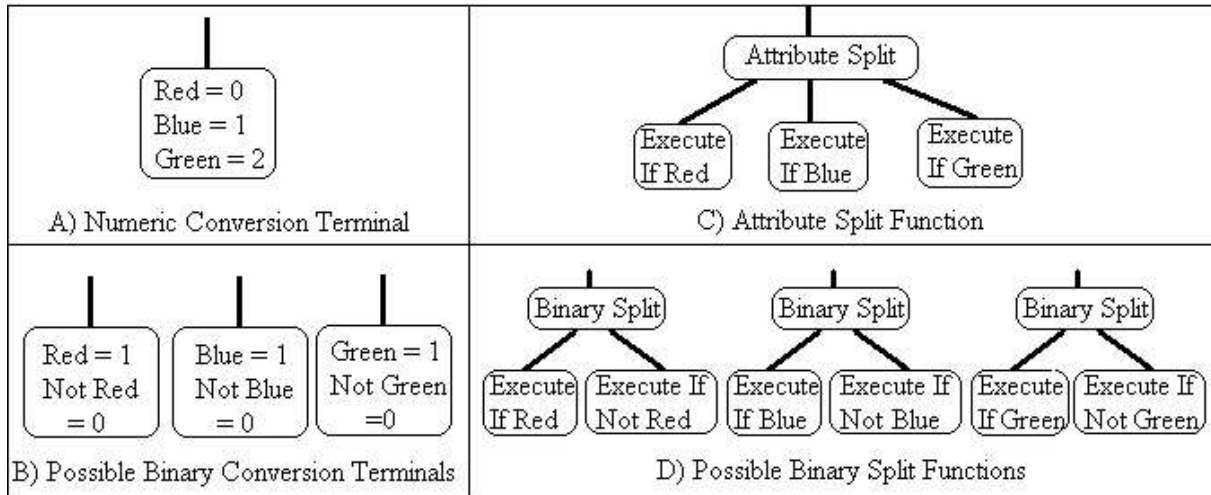


Figure 1: Two alternative encoding terminal types for numeric conversion methods and two alternate encoding function types for execution branching methods for an example attribute “colour”

Data Set Name	Training Set Size	Test Set Size	Number of Classes	Number of Continuous Attributes	Number of Nominal Attributes
Adult	4884	4884	2	6	7
Credit Card Approval	690	N.A.	2	6	9
Mushroom	4124	4000	2	0	22
Nursery	4000	4000	5	0	8
Splice	2000	1175	3	0	60
Congressional Voting	435	N.A.	2	0	16

Table 1: Datasets

2.3. Experimental configuration

A multi-class method of performing classification with GP termed dynamic range selection developed in [6] was used to evaluate the accuracy of classifiers with nominal attributes. Initially the method of classification developed in [7] (binary classification only with a zero boundary) was also used. However as classifiers trained with the dynamic range selection were more consistently accurate and capable of performing multi-class classification this method is used for all results reported in this paper.

For datasets that contained both continuous and nominal attributes, GP classifiers were first trained using only continuous attributes and programs included no components from Figure 1. This was performed to allow an indication of how much the continuous attributes contributed to the overall accuracy of classification. Any subsequent improvement in accuracy with classification using both continuous and nominal attributes could then be gauged.

The six datasets were taken from the UCI Machine Learning Repository [1]. These consisted of two mixed attribute

datasets (both nominal and continuous attributes) and four nominal only datasets. A summary of these datasets can be seen in table 1. Error estimation was performed either through the use of ten fold cross validation, or by the use of an independent test set (as indicated in table 1). The training and test datasets used in this investigation for the Adult, Mushroom, Splice and Nursery datasets have been sampled (randomly without replacement) from the larger original datasets to allow classifiers to be trained in acceptable periods of time.

Due to the probabilistic nature of GP runs, accuracy results can vary from run to run. To ensure realistic error measures each experiment was performed twenty times. The best five runs (based on training error) were then selected, and results for error rates were averaged over these five runs to give an expectation of the accuracy figures that can be reliably achieved with GP runs. Such a pruning strategy does increase the overall run time to arrive at a classifier (given that three classifiers developed are never used), but should improve the accuracy greatly over the average accuracy attained by all GP runs.

Data Set	Continuous Attributes Only	Numeric Conversion Methods		Execution Branching Methods	
		Binary Conversion Terminal	Numeric Conversion Terminal	Binary Split Function	Attribute Split Function
Adult	0.166	0.149	0.153	0.154	0.149
Credit	0.242	0.146	0.139	0.146	0.140
Mushroom	N.A.	0.0	0.0	0.0	0.0
Nursery	N.A.	0.074	0.064	0.10	0.092
Splice	N.A.	0.065	0.083	0.084	0.070
Voting	N.A.	0.051	0.052	0.051	0.050

Table 2: Classification test error rates: Average of best 5 runs (selected based on training error rate)

A population size of 500 individuals was used for all runs. Runs were terminated when perfect training accuracy was achieved or after 50 generations had progressed. When moving to the next generation, elitist reproduction accounted for 1% of the individuals, mutation for 10% whilst the remaining 89% were reproduced using crossover. Tournament selection was used for selection of individuals with a tournament size of 4. Programs were generated to an initial maximum depth of 6 with maximum depth of programs over each run limited to 17.

3. RESULTS AND DISCUSSION

The results given in table 2 show the error rate of the GP classifiers. From these it can be seen that for datasets with both continuous and nominal attributes (the Adult and Credit datasets) the use of nominal attributes in any form offers a major improvement in accuracy when compared to the GP runs that were performed on these datasets with only continuous attributes. While this result is somewhat expected (the more attributes of the problem we use the better we expect to be capable of distinguishing between the classes), it does show that the inclusion of nominal attribute functions or terminals does not detract from the classification accuracy of the GP method based only on continuous attributes. The use of nominal attribute terminals or functions in any of the encoded forms is able to make an important contribution to improving the accuracy of the GP classifier.

It can be seen from these results that none of the four functions or terminals introduced in this investigation are consistently more accurate than any other of the methods for classification using nominal attributes. The degree of variation in accuracy over most datasets for all the methods trialed is very minor. This result would indicate that regardless of the encoding method used the GP system is able to utilise the nominal attributes to perform classification. This provides support for the robustness of the GP system to be capable of utilising nominal attributes in a variety of ways

to produce results of comparable accuracy.

For the majority of datasets the Numeric Conversion Terminal performs comparably well to the other methods of classification. This result is somewhat contrary to expectation given that this terminal simply converts the nominal attribute to a continuous attribute. As a result this conversion implies some ordering to the various values of an attribute and such an ordering is not always present in the data. Some datasets do contain ordinal attributes (categorical attributes, but to which an ordering can be applied) and this ordering has been preserved in the conversion. The nursery dataset is one such dataset that contains many ordinal attributes and the accuracy of the Numeric Conversion Terminal for this dataset is comparatively high. However in all other datasets all, or a majority of the attributes, are purely nominal with no ordering. Even with such an implied ordering present the results of the Numeric Conversion Terminal are relatively accurate. This result would again tend to indicate that the GP classifier training process is a flexible one as it is capable of utilising aspects of the classification problem when they are advantages, while able to ignore these aspects when they are extraneous or irrelevant to the classification task.

3.1. Comparisons to Other Methods of Classification

Classifiers trained using GP tend to have very long training times when compared with some other methods of classification such as decision tree builders like C4.5. However, given the diverse and complex range of classification problems the class of problems that GPs can be applied, in which a large amount of training time is available if a high degree of accuracy is required, is still a valuable and important set of problems.

Table 3 shows the performance of the single best achieved test accuracy attained by any GP classification method (selected by the best performance on the training set) over these six datasets compared to four other established methods of performing classification. These classification algorithms

Data Set	Best GP	C4.5	ID 3	Naive Bayes	Back-Prop Neural Net
Adult	0.143	0.152	0.182	0.172	0.174
Credit	0.147	0.136	0.162	0.224	0.168
Mushroom	0.0	0.0	0.0	0.056	0.0
Nursery	0.050	0.061	0.045	0.105	0.023
Splice	0.059	0.072	0.077	0.054	0.057
Voting	0.057	0.041	0.062	0.099	0.046

Table 3: Best GP Performance comparisons with other classification methods

from [2] consisted of two decision tree builders (C4.5 with default parameters and ID3), one statistical classifier (Naive Bayes) and a neural network classifier (1 layer of internal nodes equal to number of attributes of the datasets, trained with back propagation of error for 100 epochs). In the Adult datasets the GP classification is higher than any other method used. The ability of GP to outperform other well-known methods of performing classification points to a definite capability of GP to perform classification for datasets with nominal attributes. Additionally, in the Credit, Mushroom and Splice datasets the GP trained classifiers appear capable of achieving a performance close to that of the best method.

These results are encouraging as the GP system tested in this investigation is a relatively simplistic model, with no attempt to control factors such as bloat, early convergence or overtraining, all of which could significantly impact the performance of the GP classifiers (GP classifiers for the Credit and Voting datasets both appeared to suffer from overtraining of the data).

4. CONCLUSIONS

The aim of this investigation was to determine whether methods could be developed by which to incorporate nominal attributes into a GP program tree of a predominantly numeric structure. Results show that the two methods of numeric conversion and execution branching tested here are capable of performing classification using both a mixture of continuous and nominal data, as well as nominal only data to perform accurate classification. The inclusion of nominal attributes does not detract from the accuracy of the classifier over continuous attributes. Of the two methods tested in this investigation, neither was consistently better than the other. The approach capable of producing the most accurate classification results varied from one dataset to another.

The GP method of performing classification with nominal attributes is capable of producing classifiers with accuracy close to other methods of classification over a majority of datasets used in this investigation. It is also able to outperform such established methods of classification for one of these datasets highlighting GPs potential usefulness in

the area of classification using nominal attributes.

5. REFERENCES

- [1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases <http://www.ics.uci.edu/~mllearn/mlrepository.html>.
- [2] Eibe Frank et. al. Weka 3. machine learning software in java <http://www.cs.waikato.ac.nz/ml/weka/>.
- [3] Chris Gathercole. *An Investigation of Supervised Learning in Genetic Programming*. Ph.D. thesis, University of Edinburgh, 1998.
- [4] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
- [5] W. B. Langdon and B. F. Buxton. Genetic programming for combining classifiers. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon and Edmund Burke (editors), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 66–73, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.
- [6] Thomas Loveard and Victor Ciesielski. Representing classification problems in genetic programming. In *Proceedings of the Congress on Evolutionary Computation*, Volume 2, pages 1070–1077, COEX, Seoul, Korea, 27-30 May 2001. IEEE Press.
- [7] Walter Alden Tackett. Genetic programming for feature discovery and image discrimination. In Stephanie Forrest (editor), *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 303–309, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann.