Evolving the input space for decision tree building

C.J.Hinde and A.I.Bani-Hani
Computer Science
Research School of Informatics
Loughborough University
Loughborough, LE11 3TU
UK
Email: {C.J.Hinde, A.I.Bani-Hani}@lboro.ac.uk

T.W.Jackson
Information Science
Research School of Informatics
Loughborough University
Loughborough, LE11 3TU
UK
Email: {T.W.Jackson@lboro.ac.uk}

Y.P.Cheung
School of Business Systems
Faculty of Information Technology
Monash University
Melbourne
Australia
Email: {Yen.Cheung@infotech.monash.edu.au}

***Abstract*—The aim of this research is to extend the discrimination of a decision tree builder by adding polynomials of the base inputs to the inputs. The polynomials used to extend the inputs are evolved using the quality of the decision trees resulting from the extended inputs as a fitness function. Our approach generates a decision tree using the base inputs and compares it with a decision tree built using the extended input space. Results show substantial improvements.**

I. Introduction

This paper addresses the well-known problem of data mining where given a set of data; the expected output is a set of rules. Decision trees using the ID3 approach [1], [2] are popular and in most cases successful in generating rules correctly. Extensions to ID3 such as C4.5 and CART are developed to cope with uncertain data. Fu et al [3] used C4.5 followed by a Genetic Algorithm (GA) to evolve better quality trees; in Fu's work C4.5 was used to seed a GA, which were then used as a basis for evolving better trees then using Genetic Programming (GP) techniques to cross over the trees. Many rule discovery techniques combining ID3 with other intelligent techniques such as genetic algorithms and genetic programming have also been suggested [4], [5], [6]. Generally, when using ID3 with genetic algorithms, individuals which are usually fixed length strings are used to represent decision trees and the algorithm evolves to find the optimal tree. When Genetic Programming is used to generate decision trees, individuals are variable length trees, which represent the decision tree. Variations in these approaches can be found in the gene encoding. One rule per individual as done in Greene [7], Freitas et al [8], [9] is a simple approach but the fitness of a single rule is not necessarily the best indicator of the quality of the discovered rule set. Encoding several rules in an individual requires longer and more complex operators [10], [11]. In genetic programming, a program can be represented by a tree with rule conditions and/or attribute values in the leaf nodes and functions in the internal nodes. Here the tree can grow dynamically and pruning of the tree is necessary [12]. Papagelis & Kelles [13] used a gene to represent a decision tree and the GA then evolves to find the optimal tree, similar to Fu et al [3]. To further improve the quality of the trees, Eggermont et al [14] applied several fitness measures and ranked them according to their importance in to tackle uncertain data. Previous work has taken the input space as a given and used evolution to produce the trees. In this work, as we shall see, the trees are generated using a variant of C4.5 and the input space is evolved rather than the trees, in direct contrast to other workers.

A vast majority of the approaches use decision trees as a basis for the search in conjunction with either a GA or GP to further improve the quality of the trees. Our approach described in this paper addresses continuous data and adds polynomials of the input values to extend the input set. A GA is used to search the space for these polynomials based on the quality of the tree discovered using a version of C4.5.

II. Iterative discrimination

ID3, C4.5 and their derivatives proceed by selecting an attribute that results in an information gain with respect to the dependent variable. A simple data set with 2 continuous attributes that are linearly separable is shown in Figure 1.

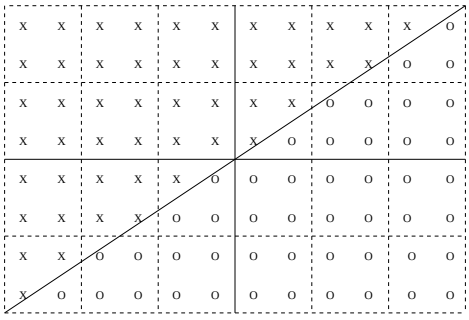

Fig. 1. A granularised version of the linearly separable set of data based on a 2 dimensional data set.

Applying C4.5 to the data set gives the result shown in Figure 2, which was first documented in [15]. If no errors are required over a large training set then the complexity of the decision tree grows with the size of the training set. This is unsatisfactory.

Anticipating the results of the proposed system a higher level discriminant of $x-y$ in addition to the two basic variables x and y would give the result shown in Figure 3.

```
x <= -0.25 :
|   y > -0.75 : in (36.0)
:
:
:
x > -0.25 :
|   y <= 0.75 : out (40.0)
|   y > 0.75 :
:
:
:

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------  ------------------------
Size   Errors   Size   Errors   Estimate
27     1( 0.8%) 27     1( 0.8%) (13.3%)
```

Fig. 2. The decision tree produced by C4.5 from the linearly separable data shown in Figure 1. The size of 27 indicates why this tree is not replicated here.

```
x-y <= -0.5 : out (64.0)
x-y > -0.5 : in (64.0)

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------   ----------------------
Size   Errors    Size  Errors   Estimate
3      0( 0.0%)  3     0( 0.0%) ( 2.1%)
```

Fig. 3. The decision tree produced by C4.5 from the linearly separable data using the discriminant value $x - y$.

III. More complex discriminants

So far we have made no more progress than Konstam [16] who used a GA to find linear discriminants. He makes the comment that the technique can be applied to quadratic discriminants. However he makes no statements about focussing the search. A set of data was prepared using the same data points as above to explore higher order and higher dimensional discriminants. The data prepared used a torus such that points inside the torus were in the concept and points outside the torus, including those that are within the inner part of the torus, were deemed outside the concept. Figure 4 illustrates the data set although, as above, does not show all the points.

Applying C4.5 to the data set represented in figure 4 gives the decision tree shown in Figure 5. This decision tree is smaller than the decision tree derived from the linearly separable data although the function used to produce the data is much more complex, and the predictions from the tree show fewer errors. The decision tree is difficult to interpret.

Taking the toroidal data set, Figure 4, and adding another attribute computed from the sum of squares of x and y gives better discrimination and a more interpretable tree shown in Figure 6. Notice that the decision tree is much smaller with

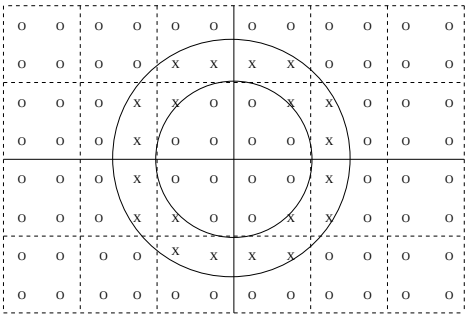

Fig. 4. A granularised version of the torus illustrating a quadratic form.

```
x <= -3.25 : out (16.0)
x > -3.25 :
|   x > 2.75 : out (16.0)
|   x <= 2.75 :
|   |   y <= -3.25 : out (12.0)
|   |   y > -3.25 :
|   |   |   y <= 2.75 : in (72.0/24.0)
|   |   |   y > 2.75 : out (12.0)

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------  ------------------------
Size   Errors   Size   Errors   Estimate
9      24(18.8%)9      24(18.8%)(25.5%)
```

Fig. 5. The decision tree produced by C4.5 from the toroidal data.

5 decision points compared to 9, and has no errors compared with 18.8% in the original tree, Figure 5.

IV. Non projectable data sets.

Thus far we have seen data sets that can be projected onto 1 dimension and which result in large trees but are nonetheless useful predictors. Section III shows that these trees can be reduced in size considerably by adding higher dimensional combined functions of the original data elements.

With the data sets shown in Figures 7 and 11 C4.5 does not produce a tree at all. Of the two data sets presented a higher order combined attribute results in a concise tree where no tree is produced without the higher order attribute. In the case

```
r2 > 8.125 : out (72.0)
r2 <= 8.125 :
|   r2 <= 0.625 : out (8.0)
|   r2 > 0.625 : in (48.0)

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------  ------------------------
Size   Errors   Size   Errors   Estimate
5      0( 0.0%) 5      0( 0.0%) ( 3.1%)
```

Fig. 6. The decision tree produced by C4.5 from the augmented toroidal data. r2 is the sum of the squares of x and y.

of the quadrant data set, a concise decision is possible with the unaugmented data set, one is not produced by C4.5.

A. Banded data set

This test shows a data set that does not project down onto 1 dimension. This 2 dimensional data set results in the following tree from C4.5, Figure 8.

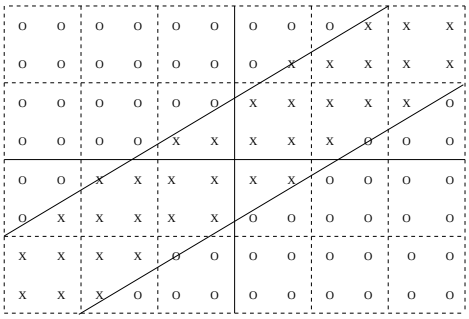

Fig. 7. A granularised version of banded linearly separable data.

```
out (128.0/52.0)

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------  ------------------------
Size   Errors   Size   Errors   Estimate
1      52(40.6%)1      52(40.6%)(44.1%)
```

Fig. 8. The decision tree produced by C4.5 from the banded data.

The decision tree produced from the banded data, Figure 8, is shown in Figure 8 and is almost useless. It does not reveal any useful information from the data. The most that can be gained from this data is that there are 52 elements in the concept and the rest are out. Adding the attribute $x - y$ gives the tree shown in Figure 9, this is a good predictor and also makes the information held in the data clear.

B. Quadrant data set

This test shows a data set that cannot be doscriminated by C4.5, however a decision tree does exist. It is shown if Figure 10. This clear 2 dimensional data set results in the following tree from C4.5, Figure 12.

```
x-y <= -2 : out (38.0)
x-y > -2 :
|   x-y <= 1.5 : in (52.0)
|   x-y > 1.5 : out (38.0)

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------  ------------------------
Size   Errors   Size   Errors   Estimate
5      0( 0.0%) 5      0( 0.0%) ( 3.2%)
```

Fig. 9. The decision tree produced by C4.5 from the banded data given the added input feature of x-y.

```
x <= 0.0 : (64.0)
|   y <= 0.0 : in (32.0)
|   y > 0.0 : out (32.0)
x > 0.0 : (64.0)
|   y <= 0.0 : out (32.0)
|   y > 0.0 : in (32.0)

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------  ------------------------
Size   Errors   Size   Errors   Estimate
6      0( 0.0%) 6      0( 0.0%) ( 3.2%)
```

Fig. 10. The decision tree which could be used to discriminate the quadrant data, but cannot be produced by C4.5.

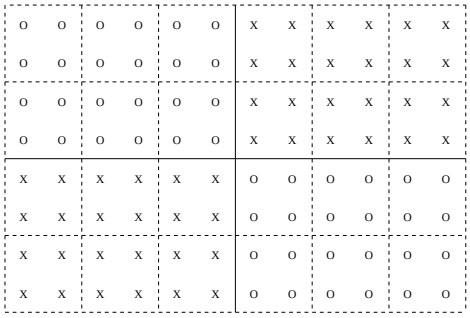

Fig. 11. A granularised version of quadrant data set.

```
out (128.0/52.0)

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------  ------------------------
Size   Errors   Size   Errors   Estimate
1      52(40.6%)1      52(40.6%)(44.1%)
```

Fig. 12. The decision tree produced by C4.5 from the quadrant data.

```
x*y <= -2 : out (38.0)
x*y > -2 :
|   x*y <= 1.5 : in (52.0)
|   x*y > 1.5 : out (38.0)

Evaluation on training data (128 items):

Before Pruning        After Pruning
--------------  ------------------------
Size   Errors   Size   Errors   Estimate
5      0( 0.0%) 5      0( 0.0%) ( 3.2%)
```

Fig. 13. The decision tree produced by C4.5 from the quadrant data given the added input feature of x*y.

V. Genetic Algorithm

The genetic algorithm attached to the front of c4.5 has a few special features. It follows most of the guidelines in [17], [18] and so has aspects designed to preserve inheritability and to ensure that no part of the genome has an inordinate effect on the phenome. With this in mind the structure of the genome is made up from a set of integers, rather than a binary genome.

A. Genetic structure.

The chromosome can deliver several genes corresponding to several combined attributes. The chromosome is a fixed maximum length and achieves a variable number of genes by an activation flag. Each gene delivers one new attribute and each variable is a linear combination of simpler variables.

1) Variable.: If the number in the variable slot is N and there are K basic continuous variables in the data set and M variables in the gene prior to this one then $N \bmod (K+M)$ refers to variable within those $K+M$ variables.

2) Function.: If the function is a monadic function then it is applied to variable 1, otherwise to both. The prototype system has a set of simple arithmetic functions, power, multiplication, division and subtraction. This is sufficient to extract all the decision trees we have considered.

3) Number of genes and gene length.: The variable length chromosome has disadvantages as the effect on the gene itself of the two fields that determine the length of the gene is considerably more than any other field and can be destructive. The variable length gene has similar disadvantages. The gene structure finally chosen for the system is shown in Figure 14

Active	
	Variable1
	Function
	Variable2

Fig. 14. This shows the basic structure of the gene adopted. The Active/Variable/Function/Variable is repeated up to the gene length.

This potentially has some of the properties of recessive genes that are attributed to diploid gene structures although no experiments have been conducted to determine this. An example gene is shown in Figure 15. This gene has 4 segments, 1 of which is active. Each segment has 2 attributes, some active and some not. The function field is interpreted as 2 for plus, 3 for minus, 5 for multiply. No other function types are illustrated.

VI. Exemplar data

The system described above was applied to some data sets taken from the Machine Learning Repository [19] in order to compare the capability of the system with other known decision tree generators.

The experiment compares the decision trees generated by C4.5 and the decision trees generated by C4.5 with the enhanced input space. The results consist of

- the percentage of correct results on the training set
- the percentage of correct results on the test set

1		active
	1	x
	3	minus
	2	y
0		inactive
	1	x
	5	times
	1	x
0		inactive
	2	y
	5	times
	2	y
0		inactive
	4	x^2
	2	plus
	5	y^2

Fig. 15. An exemplar gene. x and y are variables number 1 and 2. The first new variable is $x - y$ and is variable number 3. As this is activated then it will be made available as an input to the decision tree generator. If variable 6 is activated then because it relies on variables 4 and 5 they will also be kept but not necessarily activated.

- the number of degrees of freedom for the decision space
- the probability that the result could have arisen by chance
- the decision tree size

A. Experimental results

Each data set was split randomly into two sets, the training set which comprised 90% of the data and the test set, which comprised 10% of the data. The split was generated by choosing whether a particular data point was to be in the training set or the test set using a random number generator. This way any temporal aspects that may be in the data are accounted for. Notice the degrees of freedom are different for the training set and the test set, this is because there were no data elements belonging to one of the categories in the test set, where there were elements in the training set.

TABLE I
Experimental results for Glass data set

	C4.5	C4.5+GP
Train Correct	92.8	98.5
Test Correct	75	100
DOF Train	30	30
DOF Test	25	25
probability of not null test set	1.0	1.0
Tree size	43	61

The glass data set shows a considerable improvement for the enhanced input space, however the decision tree is larger.

The iris data set shows an improvement for the enhanced input space, but the improvement is marginal, however the decision tree is smaller.

The Pima indians data set shows a considerable improvement for the enhanced input space. Both the training set and the test set show improvement. The enhanced decision tree is also considerably bigger, by a factor of nearly 4.

TABLE II
EXPERIMENTAL RESULTS FOR IRIS DATA SET

	C4.5	C4.5+GP
Train Correct	98	100
Test Correct	100	100
DOF	6	6
probability of not null test set	0.99	0.99
Tree size	9	7

TABLE III
EXPERIMENTAL RESULTS FOR PIMA INDIANS DATA SET

	C4.5	C4.5+GP
Train Correct	82.7	98.3
Test Correct	74.5	84.2
DOF	2	2
probability of not null	1.0	1.00
Tree size	33	119

The experiments have shown that the enhanced system is able to significantly improve the quality of the decisions made, however this is often at the expense of a larger tree. The test on the iris data set indcates that the decision tree can be smaller, as shown by some of the demonstration data sets earlier in the paper.

VII. CONCLUSIONS

This paper has extended the capability of decision tree induction systems where the independent variables are continuous. The incremental decision process has been shown to be inadequate in explaining the structure of several sets of data without enhancement. The paper has shown that introducing variables based on higher order and higher dimensional combinations of the original variables can result in significantly better decision trees. This can all be accomplished by introducing these variables at the start of the decision tree generation and a suitable method for generating these would be a genetic algorithm. A fitness function for a genetic programming system has been introduced and serves to discover structure in the continuous domain.

REFERENCES

[1] J. Quinlan, "Discovering rules from large collections of examples: a case study," in *Expert systems in the microelectronic age*, D. Michie, Ed. Edinburgh University Press, 1979.

[2] J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, 1986.

[3] Z. Fu, B. Golden, and S. Lele, "A GA based approach for building accurate decision trees," *INFORMS Journal on Computing*, vol. 15, no. 5, pp. 3–23, 2003.

[4] M. Ryan and V. Rayward-Smith, "The evolution of decision trees," in *Proceedings of the Third Annual Conference on Genetic Programming*, J. Koza, Ed. San Francisco, CA.: Morgan Kaufmann, 1998, pp. 350–358.

[5] R. Marmelstein and G. Lamont, "Pattern classification using a hybrid genetic program-decision tree approach," in *Proceedings of the Third Annual Conference on Genetic Programming*, J. Koza, Ed. San Francisco, CA 94104, USA: Morgan Kaufmann, 1998.

[6] G. Folino, C. Puzzuti, and G. Spezzano, "Genetic programming and simulated annealing: a hybrid method to evolve decision trees," in *Proceedings of the Third European Conference on Genetic Programming*, R. Poli, W. Banzhaf, W. Langdon, J. Miler, P. Nordin, and T. Fogarty, Eds. Edinburgh, Scotland, UK: Springer-Verlag, 2000, pp. 294–303.

[7] D. Greene and S. Smith, "Competition-based induction of decision models from examples," *Machine Learning*, vol. 13, pp. 229–257, 1993.

[8] A. Freitas, "A GA for generalised rule induction," in *Advances in Soft Computing, Engineering Design and Manufacturing*. Berlin: Springer, 1999, pp. 340–353.

[9] D. Carvalho and A. Freitas, "A genetic-algorithm based solution for the problem of small conjuncts," in *Principles of Data Mining and Knowledge Discovery (Proc. 4th European Conf., PKDD-2000. Lyon France)*, ser. Lecture Notes in Artificial Intelligence, vol. 1910. Springer-Verlag, 2000, pp. 345–352.

[10] K. De Jong, W. Spears, and D. Gordon, "Using a genetic algorithm for concept learning," *Machine Learning*, vol. 13, pp. 161–188, 1993.

[11] C. Janikow, "A knowledge intensive GA for supervised learning," *Machine Learning*, vol. 13, pp. 189–228, 1993.

[12] Y. Hu, "A genetic programming approach to constructive induction," in *Genetic Programming, Proceedings 3rd Annual Conference*, San Mateo, California, 1998, pp. 146–151.

[13] A. Papagelis and D. Kalles, "GA tree: Genetically evolved decision trees," in *Tools with AI, ICTAI Proceedings 12th IEEE International Conference*, 13–15 Nov 2000, pp. 203–206.

[14] J. Eggermont, J. Kok, and W. Kosters, "Genetic programming for data classification: Partitioning the search space," in *ACM Symposium on Applied Computing*, 2004, pp. 1001–1005.

[15] D. Michie and R. Chambers, "Boxes: an experiment in adaptive control," in *Machine Intelligence 2*, E. Dale and D. Michie, Eds. Edinburgh: Oliver and Boyd, 1968.

[16] A. Konstam, "Linear discriminant analysis using GA," in *Proceedings Symposium on Applied Computing*, Indiannapolis.IN, 1993.

[17] M. Withall, "Evolution of complete software systems," Ph.D. dissertation, Computer Science, Loughborough University, Loughborough, UK, 2003.

[18] M. Withall, C. Hinde, and R. Stone, "An improved representation for evolving programs," *Genetic Programming and Evolvable Machines*, vol. 10, no. 1, pp. 37–70, 2009.

[19] "Machine learning repository," http://archive.ics.uci.edu/ml/datasets.html, 2010.