Using Genetic Algorithms for Inductive Learning

R. J. ALCOCK and Y. MANOLOPOULOS

Data Engineering Laboratory, Department of Informatics,
Aristotle University of Thessaloniki,
54006, Thessaloniki,
GREECE.
rob@skyblue.csd.auth.gr http://skyblue.csd.auth.gr

Abstract:- Inductive learning techniques can be utilised to build a set of IF-THEN rules from a given example data set. This paper presents a new technique for inductive learning called GAIL (Genetic Algorithm for Inductive Learning) which is based on Genetic Algorithms. Common algorithms for inductive learning are briefly reviewed. The GAIL algorithm is described and results are shown for two benchmark data sets. The results of GAIL are compared against standard inductive learning algorithms.

Keywords:- Data mining, inductive learning, genetic algorithms, machine learning, rule-based systems.

1 Introduction

Knowledge discovery is the process of finding useful knowledge from large amounts of data. The number of databases in the world is increasing rapidly and valuable information, which could be used in decision making, is often hidden in the masses of stored data. The knowledge discovery process is composed of a number of stages [1]:

- learn the problem domain;
- create data sets – select relevant examples from the overall data;
- pre-process the data – remove outliers and noise and handle missing values;
- select attributes – determine which attributes are important;
- decide the function – e.g. classification, prediction, clustering or association;
- choose the algorithm – e.g. neural networks, inductive learning or statistical techniques;
- perform data mining – run the algorithm;
- visualise results - present the results in a graphical format;
- obtain end-user reaction.

Two techniques that have been frequently employed in data mining for classification are neural networks and inductive learning. Both techniques are derived from the field of artificial intelligence [2]. Neural networks are similar to the human brain in that they have a parallel-distributed architecture and the ability to learn. Inductive learning techniques build up either a decision tree or a set of IF-THEN rules from a given example data set. These are closely related because decision trees can be easily converted into IF-THEN rule format. The main advantage of inductive learning over neural networks is that it extracts information in a form that is readable to a human user. Whereas neural networks create numerical weight values that cannot easily be deciphered, rules produced by an inductive learning algorithm are relatively easy to understand.

This paper presents a new technique for inductive learning called GAIL (Genetic Algorithm for Inductive Learning). Section two summarises common algorithms used for inductive learning. The third section presents the GAIL algorithm. Finally, GAIL is tested against standard inductive learning algorithms on benchmark problems and the results are given.

2 Inductive Learning

Inductive learning algorithms create rules from a training set containing a number of examples. Each example consists of several attribute values and a class type. Thus, the problem of inductive learning is to generate rules which, given the attribute values of an example, can determine the type. The rules generated are of the format:

IF attribute x has value y
THEN class type is z

or:

IF attribute x_1 has value y_1 AND attribute x_2 has value y_2 THEN class type is z

The first rule is said to have one *condition* and the second rule, two conditions. Normally, the more conditions that a rule has, the fewer examples it covers. Therefore, rules with less conditions usually have a greater generalisation ability.

Several inductive learning algorithms and families of algorithms have been developed. These include ID3, AQ and RULES. The ID3 algorithm, developed by Quinlan [3], produces a decision tree. At each node of the tree, an attribute is selected and examples are split according to the value that they have for that attribute. The attribute to employ for the split is the one with the highest *information gain* for the examples. In later work, the ID3 algorithm was improved to become C4.5 [4]. The AQ15 algorithm, created by Michalski et al. [5], searches for rules which can classify the examples in the training set correctly.

Recently, Pham and Aksoy [6-8] developed the first three algorithms in the RULES (RULe Extraction System) family of programs. These programs were called RULES-1, 2 and 3. Later, the rule forming procedure of RULES-3 was improved by Pham and Dimov [9] and the new algorithm was called RULES-3 PLUS. Rules are generated and those with the highest 'H measure' are kept. The H-measure is calculated based on three factors:

- the number of examples in the training set;
- the number of examples classified, either correctly or incorrectly, by the rule;
- the number of correctly classified examples covered by the rule.

The first incremental learning algorithm in the RULES family was RULES-4 [10]. RULES-4 employs a Short Term Memory (STM) to store training examples. The STM is given a user-specified size called the STM size. When the STM is full, the RULES-3 PLUS procedure is used to generate rules. RULES-4 is an incremental algorithm because when rules have been formed, new examples can be presented and existing rules can be updated or added to.

3 Genetic Algorithm for Inductive Learning (GAIL)

Genetic algorithms (GAs) are computer programs that are based on the theory of natural evolution [11]. The basis of this theory is that animals are solutions to the problem of survival. Only animals that are able to survive can reproduce thus only fit animals can pass their genes to the next generation. Over successive generations, the collective fitness of the overall population increases.

Genetic algorithms can be applied to problems where it is required to search for a solution. Possible solutions are represented as *genes* and a *fitness function* is used to determine which solutions survive to the next iteration. Genes with a low fitness are replaced by new genes. The genes in a GA are represented in binary form, that is, as strings of ones and zeros. At any one time, a *population* of genes is stored. Reproduction is performed by an operation called *crossover*, where two genes are combined to create a new gene. Diversity is maintained in the population by the *mutation* operation that generates a new gene by selecting a gene and inverting one of its elements. The proportion of times the mutation operation is applied is determined by a user-specified parameter called the *mutation rate*.

The problem of inductive learning can be seen as a searching problem. There are a large number of possible rules that could be created and it is necessary to search for the set of rules giving the best performance. Thus, it is possible to utilise GAs for inductive learning.

Rules with different numbers of conditions generate genes of different lengths. Therefore, the GAIL algorithm separately processes rules with differing numbers of conditions and then merges these rules.

In GAIL, the maximum size of rule is set to 3 conditions to ensure that the rules do not become too specific to the training set.

The fitness, or effectiveness, of each rule is calculated by a user-defined fitness function. The function employed is:

$$\text{Fitness} = NC - 10 * IC \qquad (1)$$

where NC is the number of correctly classified examples in the training set and IC is the number of incorrectly classified examples. The GAIL algorithm is given in Fig. 1.

When the rules are sorted, if two rules have the same fitness, rules with fewer conditions are given precedence as it is considered that these will have greater generalisation ability.

4 Results

To test the GAIL algorithm, a benchmark data set, IRIS flower classification [12], was employed. This data set is relatively small, consisting of 150 examples, but is useful in preliminary testing of inductive learning algorithms. Each example consists of four attributes and a class. The three classes in the data set are Iris Setosa, Iris Versicolor and Iris Virginica. The four attributes are the Sepal Length (SL), Sepal Width (SW), Petal Length (PL) and Petal Width (PW). The data was randomly split into two approximately equal sets: a training set containing 80 examples and a test set of 70 examples. The performance criterion used in all experiments was to divide the number of correctly classified examples in the test set by the total examples in the test set.

As a benchmark, the RULES-4 algorithm was employed first to classify the examples. RULES-4 has three parameters that need to be set: the STM size, number of quantisation levels (Q) and the noise level (NL). Accordingly to experiments carried out in [10], for the IRIS problem, the values of Q and NL should be 6 and 0.2, respectively. Table 1 shows experiments carried out with different STM sizes. It can be seen that as the STM size increases, the performance improves. The best result (94.3%) was obtained when the STM size equaled the training set size. The number of rules produced was sixteen.

Next, GAIL was utilised to classify the data. GAIL has four parameters that need to be set. These are the number of quantisation levels (Q), the mutation rate (MR), the number of iterations (IT) and the population size (POP). As these parameters could take any value, there are a very large number of combinations. It is impossible to test every parameter combination, therefore arbitrary values for each parameter were chosen and all combinations of these were tested. The values chosen were $Q \in \{4, 6, 8\}$, $MR \in \{0.1, 0.2\}$, $POP \in \{10, 20\}$ and $IT \in \{10000, 20000\}$. Table 2 shows the results of the experiments sorted into performance order. The conclusions made from these experiments were:

- the maximum performance achieved was 94.3%. In all four such cases, the commonality was that POP=20 and Q=6;
- the next best performance was 84.3%. In the five cases with this performance, the major similarity is that POP=20 and Q=8.

Therefore, it can be concluded that the most critical parameter was Q and the next most important was POP. Values of Q=5, Q=7 and POP=30 were tried but these gave no further improvement.

GAIL generated six rules:

1. IF PL < 2.0666 THEN class is IRIS SETOSA
2. IF 0.9 <= PW < 1.3 THEN class is IRIS VERSICOLOR
3. IF 1.3 < PW < 1.7 THEN class is IRIS VERSICOLOR
4. IF PW >= 2.1 THEN class is IRIS VIRGINICA
5. IF PL >= 5.9333 THEN class is IRIS VIRGINICA
6. IF 1.7 <= PW < 2.1 THEN class is IRIS VIRGINICA

All the rules generated have just one condition. It is considered that this is because rules with one condition have higher fitness values than multi-condition rules, which are less general. The rules generated employ just two attributes (PL and PW). Therefore, GAIL can also be employed in knowledge discovery as an attribute selection method.

To further test the ability of GAIL, data about the heart condition of patients in Cleveland, USA, was utilised [13]. The data set contains thirteen attributes, regarding information about patients and a class field showing whether the patient has a heart condition or not. Previous results with this data set show that it is more difficult to classify than the IRIS data set. Aha et al. [14] tested the C4 algorithm, developed by Quinlan, on this data set and achieved an accuracy of 75.5%. In the same paper, they developed an algorithm called IB3, a robust extension of the nearest neighbour classifier, obtaining a performance of 78.0%. Gennari et al. [15] employed the CLASSIT conceptual clustering system for this data set, recording an accuracy of 78.9%.

The heart data set consists of 303 examples. Six examples contain unknown data values and these were removed as the current version of GAIL does not deal with unknown values. The remaining examples were randomly split into a training set and a test set, each consisting of 148 examples. The first experiments performed using GAIL were to determine the value for Q as this was found to be the most sensitive parameter on the IRIS data set. For each value of Q, tests were carried out for an arbitrary population size of 30 and 30000 iterations. The average test set performance for each value of Q is shown in Table 3. The optimum value of Q for this data set was found to be 3.

Next, using the value of three for Q, the values of POP and IT were altered to determine if they could improve the performance. Each experiment was run five times and the average results are shown in Table 4. The best performance was obtained with a population size of 60 and 50000 iterations. It was seen that as the population size was increased, the performance improved.

GAIL produced a large number of unclassified examples on the heart data set, i.e. examples for which no rules were found to classify them either correctly or incorrectly. It was considered that a major contributory factor to this was the adopted fitness function that placed a heavy penalty on misclassifications. Therefore, the fitness function was changed to:

$$\text{Fitness} = \text{NC} - \text{GAIN} * \text{IC} \qquad (2)$$

Then, experiments were carried out, with a population size of 60 and using 50000 iterations, to determine the optimal value of GAIN. Results are shown in Table 5. The best performance was 81.8%, with a GAIN of 3. The large variation in performances with different values for GAIN indicates the importance of adopting an appropriate fitness function in obtaining optimal performance.

5 Conclusions

This paper has introduced a new technique, called GAIL, for inductive learning based on genetic algorithms. On the benchmark IRIS data set, GAIL is able to achieve the same performance as a recently-developed inductive learning algorithm (RULES-4). Also, GAIL is able to generate a more compact rule set. With the heart data set, GAIL gave a higher performance than previously-reported results with the aid of a suitable fitness function. These results show the potential for using genetic algorithms for inductive learning.

It is considered that with the IRIS test data set employed with six equally-spaced quantisation levels (Q=6), 94.3% may be the maximum accuracy achievable. It was also seen that the choice of the correct number of quantisation levels was crucial in obtaining optimal performance. Therefore, future work on GAIL and RULES-4 should be focussed on improving quantisation techniques. In particular, quantisation levels which are different for each attribute and inequally-spaced quantisation levels could be adopted.

A problem with using genetic algorithms for rule induction is the large number of iterations required to reach a solution. Further work should be carried out in this area into guiding the search more effectively. Research should also be performed into determining the most effective fitness function.

Acknowledgement

The authors would like to thank Robert Detrano, M.D., Ph.D from the V.A. Medical Center, Long

Beach and Cleveland Clinic Foundation for collecting the data on heart patients used in these experiments.

References

[1] Fayyad U, Piatetsky-Shapiro G and Smyth P. The KDD Process for Extracting Useful Knowledge from Volumes of Data. ***Communications of the ACM***. Vol. 39, No. 11, 1996, pp. 27-34.

[2] Pham D T, Pham P and Alcock R J. Intelligent Manufacturing. in ***Novel Intelligent Automation and Control Systems***, Vol. I. ed. Pfeiffer J. Papierflieger, Clausthal-Zellerfeld, Germany. 1998. pp. 3-18. ISBN 3-89720-201-8.

[3] Quinlan J R. Learning Efficient Classification Procedures and their Applications to Chess End Games. in ***Machine Learning, an Artificial Intelligence Approach***. Eds. Michalski R S, Carbonell J G and Mitchell T M. Morgan Kaufmann, San Mateo, California, 1983, pp. 463-482.

[4] Quinlan J R. ***C4.5: Programs for Machine Learning***. Morgan Kaufmann, San Mateo, California. 1993.

[5] Michalski R S, Mozetic I, Hong J and Lavrac N. The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. ***Proc. 5th Int. Conf. On Artificial Intelligence***. Philadelphia, Pennsylvania. (Morgan Kaufman, San Mateo, California). 1986. pp. 1041–1045.

[6] Pham D T and Aksoy M S. An Algorithm for Automatic Rule Induction. ***Artificial Intelligence in Engineering***, Vol. 8, 1994, pp. 277-282.

[7] Pham D T and Aksoy M S. RULES: A Rule Extraction System. ***Expert Systems Applications***. Vol. 8, 1995, pp. 59-65.

[8] Pham D T and Aksoy M S. A New Algorithm for Inductive Learning. ***Journal of Systems Engineering***. Vol. 5, 1995, pp. 115-122.

[9] Pham D T and Dimov S S. An Efficient Algorithm for Automatic Knowledge Acquisition. ***Pattern Recognition***. Vol. 30, No. 7, 1997. pp. 1137–1143.

[10] Pham D T and Dimov S S. An Algorithm for Incremental Inductive Learning, ***Proc. IMech E***, Vol. 211, No. 3, 1997, pp. 239-249.

[11] Goldberg D E. ***Genetic Algorithms in Search, Optimization and Machine Learning***. Addison-Wesley, Reading MA. 1989. ISBN: 0201157675

[12] Fisher R A. The Use of Multiple Measurements in Taxonomic Problems. ***Annual Eugenics***, Vol. 7, Part II, 1936, pp. 179-188.

[13] Blake C, Keogh E and Merz C J. ***UCI Repository of Machine Learning Databases***. Irvine, CA: University of California, Department of Information and Computer Science. 1998. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

[14] Aha D W, Kibler D and Albert M K. Instance-based Learning Algorithms. ***Machine Learning***, Vol. 6, 1991, pp. 37-66.

[15] Gennari J H, Langley P and Fisher D. Models of Incremental Concept Formation. ***Artificial Intelligence***, Vol. 40, 1989, pp. 11-61

STM size	Average % (3 runs)
20	86.2
40	91.4
60	93.3
80	94.3

Table 1 Effect of Varying the STM Size with RULES-4

Q	MR	POP	IT (000)	Average % (3 runs)
6	0.1	20	10	94.3
6	0.2	20	10	94.3
6	0.1	20	20	94.3
6	0.2	20	20	94.3
6	0.1	10	10	84.3
8	0.1	20	10	84.3
8	0.2	20	10	84.3
8	0.1	20	20	84.3
8	0.2	20	20	84.3
6	0.2	10	20	82.4
4	0.1	20	20	80.5
4	0.2	20	20	80.5
8	0.2	10	20	79.5
8	0.2	10	10	78.5
4	0.1	10	10	75.7
4	0.2	10	10	75.7
4	0.1	10	20	75.7
4	0.2	10	20	75.7
6	0.2	10	10	75.2
4	0.1	20	10	75.2
4	0.2	20	10	74.3
8	0.1	10	20	73.4
6	0.1	10	20	72.4
8	0.1	10	10	71.4

Table 2 Different Parameter Settings for GAIL in Sorted Order

Q	Average test set performance % (5 runs)
3	54.7
4	41.9
5	42.3
6	45.7
7	50.8
8	49.3

Table 3 Determining Q for the Heart Data Set

IT (000) / POP	30	40	50	60	Av. %
30	61.1	60.4	56.8	49.5	57.0
40	57.4	67.4	61.2	57.6	60.9
50	67.0	64.7	67.8	67.8	66.8
60	68.2	69.5	71.1	67.3	69.0
Av.	63.4	65.5	64.2	60.6	

Table 4 Determining POP and IT for the Heart Data Set

GAIN	Average test set performance % (3 runs)
0	54.5
1	64.2
2	78.4
3	81.8
5	80.2
10	71.8

Table 5 Altering the Gain of the Fitness Function

```
read in the training data set
quantise the attributes into Q quantisation levels
for (Number_of_Conditions is from 1 to 3)
{
        randomly generate an initial population of rules
        for (Iterations is from 1 to Number_of_Iterations)
        {
                calculate the fitness of the rules using the fitness function
                calculate R, a random number between 0 and 1
                if R < mutation rate (MR) then generate a new gene by mutation
                        else generate a new gene by crossover
                replace rule with lowest fitness with the new gene
        }
}
sort the rules into order according to fitness
test the rules on the training set and remove any not used (pruning phase)
```

Fig. 1 GAIL Algorithm