Evolutionary Approach to Data Mining

Y.P.Singh
Faculty of Information Technology
Multimedia University
63100 Selangor, Malaysia
ypsingh@mmu.edu.my

Norhana Abdul Rahman Araby
Faculty of Information Technology
Multimedia University
63100 Selangor, Malaysia
norhana.abdulrahman@mmu.edu.my

Abstract — **Data mining is the process of extracting previously unknown information from exceeding large data set with minimum human interference. The useful information may be expressed as relationships between propositions or variables or data elements, which can be used to predict future patterns or behaviour. The present paper investigates evolutionary computing techniques for data mining tasks in form of discovery of association rules and presents a brief review of evolutionary computation techniques for machine learning systems. The evolution of association rules as subset selection in the best form is comprehensible and modular knowledge for understanding. The experimental results and examples for binary data set are provided to demonstrate the effectiveness of evolutionary computation for rule discovery tasks in form of association rules.**

I. INTRODUCTION

Data mining is currently an area of great interest which combines databases, artificial intelligence and machine learning. Data mining (also known as Knowledge Discovery in Data set - KDD) has been defined as "The nontrivial extraction of implicit, previously unknown, and potentially useful information from large data set" [1], [2] and a fairly large class of data mining tasks can be described as search for interesting and frequently occurring patterns or rules from database. It uses machine learning, statistical and visualization techniques to discover and present knowledge in a form which is easily comprehensible to humans [2]. Knowledge discovery in data set is often called data mining. The discovered knowledge can be rules describing properties of data or relationships among data, frequently occurring patterns, clusterings of the objects in the data set, etc. Most data mining systems in use todate have been designed using variants of traditional machine learning techniques [3]. Many algorithms have been discussed in the literature for discovering association rules [4, 5,] and they have very poor response time [6].

The present paper gives a short discussion of evolutionary techniques and recent developments in data mining techniques for database applications. We present genetic algorithms for mining association rules as subset selection. Examples and experimental results for binary data set are provided to demonstrate effectiveness (rule quality) and response time (spool) of genetic algorithms for data mining applications.

0-7803-5812-0/00/$10.00©2000 IEEE

II. ASSOCIATION RULES

An association rule in large collection of data in database applications is represented in the following way [7], [8]:

Butter, milk $\Rightarrow$ *bread.*

This means that somebody that buys *butter* and *milk* is also likely to buy *bread*. So association rules describe on a given collection of items possibility of various combination of items are to occur together in the same sets. Let $R = \{i_1, i_2, \ldots, i_m\}$ be a set of attributes called items defined over the binary domain $\{0,1\}$. The input $r \{ t_1, t_2, \ldots, t_n\}$ for data mining method is a set of binary vectors of size m. Each vector can be considered as set of properties or items of R (that is, $t(i) = 1 \Leftrightarrow ri_i \in t$).

Let $X \subseteq R$ be a set of attributes called itemset and $t \in r$ be a vector of the relation. If $t(A) = 1$ for all $A \in X$, we represent $t(X) = \bar{1}$. An association rule over r is an expression $X \Rightarrow Y$, where $X \subseteq R$ and $Y \subseteq R \backslash X$. Given real number γ (confidence threshold) and σ (support threshold), we say that r satisfies $X \Rightarrow Y$ with respect to γ and σ, if

$$\frac{|\{i \mid t_i[XY] = \bar{1}\}|}{|r|} \geq \sigma_n$$

and

$$\frac{|\{i \mid t_i[XY] = \bar{1}\}|}{|\{i \mid t_i[X] = \bar{1}\}|} \geq \gamma$$

That is, we look for association (vector of r) that have 1's in all the attributes of XY and at least a fraction of the rows having a 1 in all attributes of X also have a 1 in Y (confidence). Given a set of attributes (items) $X \subseteq R$, we say that X is covering (with respect to database and the given support σ, if

$$\frac{|\{i \mid t_I[X] = \bar{1}\}|}{|r|} \geq \sigma_n$$

Example 1: Association rule

Butter, milk ⇒ bread (0.84,0.34).

States that 84% of all the customers that have bought butter and milk, also brought bread, and that 34% of all customers actually have brought butter, milk and bread.

Example 2. Let R = {A, B, C, D, E} be the items and
T = {t_1, t_2, t_3, t_4}be transactions.

where t_1 = {A,B,C}, t_2 = {A,B,D,}, t_3 = {A,D,E}, and t_4 = {A,B,D,}.

Association rules with support (0.5) and confidence (0.8) can be easily derived as

B ⇒ A and D ⇒ A.

These are the only rules which can meet support threshold (0.5) and confidence threshold (0.8). This can be interpreted as those who bought B also bought A in database applications with support (0.5) and confidence (0.8). This may be very significant information to promote products or may be used for strategic purposes.

III. EVOLUTIONARY COMPUTING TECHNIQUES

The currently most important and widely known representatives of evolutionary computing techniques are : *genetic algorithms (**GAs**)* [9], *evolution strategies (**ESs**)* [10], *and evolutionary programming (**EP**)* [11,12]. These techniques are applied in problem solving by applying evolutionary mechanism. In the following we present a brief review of genetic algorithms, the evolutionary computing techniques and their use for machine learning problems. The basic evolutionary algorithm can be represented as given below:

```
t:= 0;
initialize P(t) ; (generalize initial population)
evaluate P(t);
While not terminate (P(t)) do
    Select:     P(t):= from_P(t);
    Recombine:  P'(t):= r(P(t);
    Mutate :    P''(t):= m(P'(t));
    Evaluate:   P''(t);
    t:= t+1;
Od
Return (best individual in P(t));
```

In this algorithm, P(t) denotes a population of individuals at generation t. Q(t) is a special set of individuals that has to be considered for selection and P" is offspring individuals.

At the present time, genetic algorithms are considered to be among the most successful machine-learning techniques and are also used as general-purpose search techniques for solving complex problems. Based upon genetic and evolutionary principles, GAs work by repeatedly modifying a population of individuals through the application of *selection*, *crossover*, and *mutation* operators. The choice of an representation of individual (encoding) for a particular problem is a major factor determining a GAs success. GAs have been used for optimization as well as to classification and prediction problems [13,14] with different kinds of encoding . A GAs *fitness function* measures the quality of a particular solution. The traditional GA begins with a population of *n* randomly generated individuals (binary string of fixed length *l)*, where each individual encodes a solution to the task at hand. The GA proceeds for a fixed number of generations. During each generation, the GA improves the individuals in its current population by performing selection, followed by crossover and mutation.

Selection is the population improvement or "survival of the fittest" operator. Basically, it duplicates structures with higher fitnesses and deletes structures with lower fitnesses. There are a number of methods which can be used as selection operators. The most common selection operators are:

(i) Proportional Selection
(ii) Tournament Selection

Crossover, when combined with selection, results in good components of good individuals yielding better individuals. The offspring are the results of cutting and splicing the parent individuals at various crossover points. There are a number of crossover operators which have been used in machine learning applications and they are:

(i) Single-point crossover
(ii) Uniform crossover

Mutation creates new individuals that are similar to current individuals. With a small, prespecified probability (p_m [0.005, 0.01] or $p_m = 1/l$ where l is the length of the string representing individual), mutation randomly alters each component of each individual.

The main issues in applying GAs to data mining tasks are selecting an appropriate representation and an adequate evaluation function.

IV. GA AND DATA MINING

Genetic algorithm-based learning has traditionally been grouped into one of the two general approaches. The most common form for a concept description is one or more production rules and GA is used to evolve sets of rules. The Pitts approach [13] uses a traditional genetic algorithm in which a population of variable rule set representing a complete solution to learning problem is maintained. Here

each entity in the population is a set of rules and it represents a complete solution to the learning problem. The Michigan approach (Holland, 1986) [9] has used a distinctly different evolutionary mechanism, in which a population of individual rules, each of which represents a partial solution to the overall learning task is maintained.

V. ASSOCIATION RULE REPRESENTATION AND GENETIC DATA MINING

In order to apply GA to a data mining problem, we need to select representation of association rules and an evaluation function which can assign fitness value to rules. These components are significant for successful application to data mining problem.

An association rules (complex concepts) can be represented as a disjunctive set of (possibly overlapping) production rules of binary types .The left side of the rules consists of conjunctions of attributes/value-set conditions. The right-hand side determines the association class (assigned class).

Association rules are represented as variable-length string. Crossover can occur anywhere (i.e., both on rule boundaries and within rules). To illustrate representation more concretely, consider example 2 given in earlier section where we have form the following association rules :

$B \Rightarrow A$ and $D \Rightarrow A$.

The simplest form of representation is to consider each item as binary type. Each individual consists of fixed-length binary string representing some subset of the given item set. An individual of length, l correspond to a l-dimensional binary vector where each bit represents the elimination or inclusion of the association claim. For example, $X_i = 0$ represents elimination and $X_i = 1$ represents inclusion of the ith item. Hence, an item set with six items can be represented as $<X_1,\ldots, X_6>$. Thus, an individual of <111111> indicates inclusion of all items and <110101> represents the subset where the third and the fifth items are eliminated.

VII. SIMULATION RESULT

A. Finding Frequent Sets using Genetic Algorithms

Let R = {A, B, C, D, E} — items

T = { {11100}, {11010}, {10011}, {11010} } — transactions in the database where each transaction is a subset of the items.

GA Parameters

Population size, $pop\_size = 10$
Individual size (number of bits) = 5
Probability of crossover, $p_c = 0.6$
Probability of mutation, $p_m = 0.01$

1) Initial population

Initial population is randomly generated where each individual represents possible solution.

10110 11000 10001 11100 10011
11010 10111 00110 01001 01010

2) Selection (Reproduction)

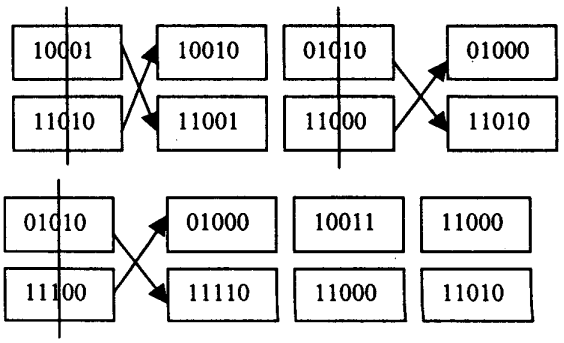
The table below gives randomly generated population, its individual fitness value and mating pool in the initial population. Fitness value for each individual is evaluated based on the number of occurrences of each individual with respect to the transactions in the database. Fitter individuals are chosen for reproduction with probability of selection proportional to fitness using Roulette-wheel selection.

Individual	Fitness value	Mating Pool
10110	0	10001
11000	3 (30%)	11010
10001	1 (10%)	01010
11100	1 (10%)	11000
10011	1 (10%)	01010
11010	2 (20%)	11100
10111	0	10011
00110	0	11000
01001	0	11000
01010	2 (20%)	11010

It is obvious from the above table that individuals with higher fitness value are more likely to be selected more than once and the weaker individuals are eliminated. These fitter individuals are used to create new population of the same size after crossover and mutation operation.

3) Crossover

Crossover which is one of the recombination operator is then applied to some members of the new population. p_c gives us the number of individuals which should undergo the crossover operation, which is $p_c * pop\_size = 0.6 * 10 = 6$. This means that only 6 individuals with $r \leq 0.6$ are randomly chosen and paired for crossover, where r is a randomly generated number for each individual in the interval [0,1]. These pairs of individuals are mated at a randomly chosen crossing point (any point between 1 to l where l is the individual bit-string size). At the crossing point, the coupled individuals exchange their bits of string as shown.



4) Mutation

After crossover, mutation is then performed on a bit-by-bit basis with a very low mutation probability. p_m gives us the expected number of bits to be mutated. A random number, r is generated for each bit in all individuals. Every bit in all individuals of the whole population has the equal chance to be mutated, i.e., change bit from 0 to 1 or vice versa. We only mutate the bit with $r \leq p_m$.

Crossovered population	Mutated population
10010	10010
11001	11001
01000	01000
11110	10110
01000	01010
11010	11010
10011	10011
11000	11000
11000	11000
11010	11010

5) Convergence

After going through crossover and mutation, the new population is ready for next generation. The same process of selection, crossover and mutation will repeat until the evolution converges to a situation where there is no more improvement in the fitness of the population.

B. Constructing Association Rules

Phase I

The frequent sets (hidden transactions) that we got using GA after several generations (10), with their fitness value are:

F_1 = {{11000}, {10010}, {01010}, {10000}, {01000}, {00010}}

Fitness value = {3, 3, 2, 4, 3, 3}

Phase II

These hidden transactions are looked as frequent sets and are used to derive association rules by genetic processing in the second phase as given below:

F_2 = {{10000}, {01000}, {00010}}

Fitness value = {4, 3, 3}

Phase II tells that items A, B and D are very frequently sold items in the hidden transactions obtained in Phase I.

From F_1 and F_2, it is easy to derive association rules as shown below:

Parameters

Support threshold, $\sigma_n = 0.5$

Confidence threshold, $\gamma = 0.8$

$$AB : A \Rightarrow B \text{ if } \frac{\text{support}(A \cup B)}{\text{support}(A)} = \frac{3}{4} = 0.75 \ (< 0.8)$$

$$B \Rightarrow A \text{ if } \frac{\text{support}(A \cup B)}{\text{support}(B)} = \frac{3}{3} = 1 \ (\geq 0.8)$$

$$AD : A \Rightarrow D \text{ if } \frac{\text{support}(A \cup D)}{\text{support}(A)} = \frac{3}{4} = 0.75 \ (< 0.8)$$

$$D \Rightarrow A \text{ if } \frac{\text{support}(A \cup D)}{\text{support}(D)} = \frac{3}{3} = 1 \ (\geq 0.8)$$

$$BD : B \Rightarrow D \text{ if } \frac{\text{support}(B \cup D)}{\text{support}(B)} = \frac{2}{3} = 0.67 \ (< 0.8)$$

$$D \Rightarrow B \text{ if } \frac{\text{support}(B \cup D)}{\text{support}(D)} = \frac{2}{3} = 0.67 \ (< 0.8)$$

From the above computation, we can derive association rules as follows:

$B \Rightarrow A$ and $D \Rightarrow A$

where: $B \Rightarrow A$ means that every transaction containing B also contains A meeting support and confidence threshold.

$D \Rightarrow A$ means that every transaction containing D also contains A meeting support and confidence threshold.

VIII. CONCLUSION

In this paper, we have presented genetic algorithms as evolutionary techniques to discover relationships or patterns in large data set in form of association rules. We proposed two phase processes for genetic mining of association rules. The experimental results and examples of binary data set have been provided to show the performance of genetic algorithms for association rule discovery tasks. Designing appropriate encoding are crucial and so future work will investigate association rule encoding and coevolution to improve response time for data mining applications.

IX. REFERENCES

[1] Pieter Adriaans and Dolf Zantinge, *Data Mining*, Addison-Wesley, New York, 1996

[2] W. Frawley and G. Piatetsky-Shapiro and C. Matheus, Knowledge Discovery in Databases: An Overview. AI Magazine, Fall 1992, pgs 213-228.

[3] "State of The Art" in Byte Magazine October 1995, contains three articles on Data Mining and Data Warehousing.

[4] Kenneth A. DeJong, William M Spears, and Diana F. Gordon, *Using Genetic algorithms for concept learning, Machnie Learning* , 1993, 13: 161-188.

[5] S.Augier, G.Venturini, and Y.Kodratoff, "Learning first order logic with a genetic algorithms," In Usama M. Fayyad and Ramasamy Uthurusamy, editors,

[6] W. H. Inmon and S. Osterfelt, *Understanding Data Pattern Processing* QED Technical Publishing Group, 1991, Wellesley, MA. (I haven't see this book myself, but I saw Gregory Piatetsky-Shapiro describe it as "a business-oriented, non-technical book")

[7] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami, "Mining association rules between sets of item in large databases," in *SIGMOD-93,* May 1993, pp. 207-216.

[8] Rakesh Agrawal and Ramakrishnan Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB-94,* September 1994.

[9] J.H Holland, "Outline for a logical theory of adaptive systems," *Journal of the Association for Computing Machinery,* 1962, 3:297-314.

[10] D.B. Fogil, "Evolutionary Computation towards a New Philosophy of Machine Intelligence," *IEEE Press,* Piscataway, N.J., 1962.

[11] H.P. Schwefce, *Numerical Optimization of Computer Models,* Wiley, Chichester, p. 198.

[12] H.P. Schwefce, *Evolution and Optimum Seeking,* Sixth-Generation Computer Technology Series, Wiley, New York, p. 199.

[13] S. Smith, A learning system based on Genetic algorithm, Ph. D. Dissertation 1980 Computer Science, University of Pittsburg.

[14] K. DeJong, Genetic Algorithm: A 10 years Perspectives, Proc. First International Group Parallel Problem Solving from Nature, 1990, 169-177, p.244.