

Design and Implementation of a Genetic-Based Algorithm for Data Mining

Sunil Choenni

Report prepared by:

Sam Beskur

Aaron Eagerman

Introduction

Data mining can be thought of as a search problem. The problem is to search a large space for interesting information(classifications). The absolute size of the search spaces involved in data mining requires that algorithms be explored that can determine interesting classifications by examining subsets of this data. Algorithms perform differently based on the characteristics of the data to be mined. It has been proposed that genetic algorithms can efficiently search certain large data sets and arrive at interesting classifications. This paper describes the design and implementation of a genetic algorithm that can be used for this purpose. The algorithm is implemented(in a proprietary tool called SHARVIND) and results are mined from 1 test and 2 real databases.

Genetic algorithms are modeled after the survival of the fittest concept observed in biology. A population consists of individuals, individuals crossover and mutate to form new populations, the fitness of individuals is evaluated, and theoretically the final population will consist of the strongest individuals. Genetic algorithm terminology, as it relates to this paper, is described in more detail in the methodology section.

Prior Work

Several researches have proposed that genetic algorithms can be applied to data mining tasks. Some have used genetic algorithms to discover first order logic rules, others have been more closely aligned with machine learning.

Contributions

This paper stresses a few contributions to the field of genetic algorithms in data mining. First, they have implemented their algorithm and applied it to actual data, none of the cited references go this far. Their algorithm differs from others that more closely resemble a hill climber algorithm. The authors state that a genetic algorithm is superior to hill climber algorithms in that by examining a larger space it is less likely to finish at a local optimum. The authors also claim that they have solved the “partitioning of attributes problem”. The “partitioning of attributes problem” is that it is difficult to mutate values while considering the domain of that attribute. This is described in more detail in the methodology section.

Methodology

The idea in classification data mining is to efficiently derive interesting rules from search spaces. In order to do this effectively on large search spaces, optimizations must be made. Individuals and populations can both be thought of as the where clause of a SQL query. An individual's where clause will likely narrow it down to a unique key in a database. The where clause of a population will select multiple tuples. The following are examples, from the paper, of where clauses that represent populations:

$$p_1 = \text{gender is ('male')} \wedge \text{age in [19, 34]}$$

$$p_2 = \text{age in [29,44]} \wedge \text{category is ('leased')} \wedge \text{town is ('rome', 'amsterdam', 'cairo')}$$

Some categories of queries can immediately be eliminated from the search space. The following rules are stated by the authors:

- 1) an attribute appears at most once in an expression (ie age = 20 \wedge age = 30 is a mutually exclusive query)
- 2) disjunctions of expressions are not allowed in the search space (ie expressions that select arbitrary tuples that do not contain common characteristics)
- 3) an expression should contain at least one conjunction (single predicate expressions are not likely to reveal previously unknown and interesting data about the search space)

Fitness Function

The success of a genetic algorithm is directly linked to the accuracy of the fitness function. The fitness function should be tailored to the specific search spaces. The authors propose a fitness function that considers major issues in evaluating an individual against its search space. The fitness of a population is the sum of the individual fitness values of that population. The fitness function is the primary performance sink in a genetic algorithm, because this is the place that the underlying data must be accessed, so optimizations should be considered wherever possible.

The authors believe that the cover of the target class and the ratio of the cover of an individual to the cover of the target class are important when considering a classification problem.

$\|\sigma_p(D)\|$ is the cover of the individuals represented by the population p in the database D

The cover of the target class refers to how well the individuals support the target class. A lot of individuals that correspond to a target class is preferable to a few.

The ratio of the cover of an individual to the cover of the target class refers to the intersection of the population and the target class individuals in the database. If this value is small it means that the target is not well supported by the individual.

$$\frac{\|Sp(D) \cap St(D)\|}{\|St(D)\|}$$

The authors define the following fitness function that grows linearly with the number of tuples that satisfy both the description of the individual and the target class, then decreases linearly with this intersection of the level of the support of the classes is reached ($\beta\|\sigma_i(D)\|$). β is a user defined value that corresponds to the number of tuples that should be satisfied for the maximum fitness level (ie if its 30% true, that is true enough). α is a user defined value that defines the number of tuples that an individual should represent in order to warrant future investigation.

$$F(p) = \left\{ \frac{\|Sp(D) \cap St(D)\|}{B \|St(D)\|} C(t) \dots \text{if } \|Sp(D) \cap St(D)\| \leq B \|St(D)\| \right.$$

$$\left. \dots \dots \left\{ \frac{\|Sp(D) \cap St(d)\| - \|St(D)\|}{\|St(D)\| (B-1)} C(t) \right. \right.$$

where $C(t)$ is 0 if the ratio of the cover of class t in the database to the database is $\leq \alpha$ and 1 otherwise.

It is important to realize that the cover values are found by executing queries on the search space based on the corresponding where clauses.

Manipulation

A population in a genetic algorithm becomes another population through a series of manipulations on the data. There are 2 types of manipulations, mutation and cross over. Mutation involves morphing a single individual into a new one. Cross over is taking 2 individuals and combining them to produce 2 new individuals.

Mutation algorithms must accommodate both the domain of the attribute and the different types of attributes in an individual. Some attributes contain no inherit relationship to one another (a list of towns), other attributes of the individual represent a range (ie age is 20-30). In the author's algorithm, if there is no order then one of the attributes values is randomly selected and replaced by another value in the attribute domain. If the attribute values represent a range then the begin or end is randomly selected and changed. Its value is changed by some number δ . The determination of δ can be found in the paper, I will not go into the details here. If the list is not successive then a random element is chosen and changed by δ . The mutation of attributes is a hard problem, the algorithm in the paper is intended as a reasonable approach.

The author's crossover algorithm selects a random point in the individual and swaps the values before and after that point with the other individual.

Optimization

In order for genetic algorithms to be feasible for large data mining problems a few optimizations should be implemented. The authors define the concept of similar individuals. $Length(p)$ is the number of elementary expressions in an individual p . p_{sim} is a similar of p if each elementary expression is contained in p or p_{sim} contains each elementary expression of p and $length(p_{sim}) \neq length(p)$. The fact that individuals are similar can be used to infer how their fitness values will relate. This is implemented in the algorithm described below. Two propositions are listed in the paper. I will not go into the details but they involve looking at similar individuals and comparing already known cover ratios and their comparative lengths to decide if the fitness of the similar individual is guaranteed to be less than or equal to the fitness of the original. These propositions can be used to increase the efficiency of the genetic algorithm. If the new individual falls under either of the above cases then it is a waste of time to compute its fitness before performing some type of manipulation.

Algorithm

The full algorithm is listed in the paper. I will provide a summary. The algorithm presented is based on the theory of elitist combination. The elitist combination approach simplifies the implementation by not requiring a probability of cross over and by allowing it to go directly to the $t+1$ population instead of requiring an intermediary population. Elitist combination works by shuffling the population, crossing over adjacent individuals, comparing the parents directly to their children, and choosing the parent and the offspring with the highest levels of fitness to survive into the next generation.

Here is an overview of the algorithm, note that it stops when the $t+1$ population is not significantly more fit than the t population :

- Initialize the population at time t as $P(t)$ with a random or psuedo random selection from the search space. This must be an even number.
- Compute the fitness of the individuals in $P(t)$
- enter a while loop that terminates when the overall fitness of population $t+1$ (sum of the individual fitness values) is not significantly greater (ϵ) than the fitness of the t population
- shuffle the population $P(t)$
- $j = 0$
- while j is less than the population size
 - cross over the individual j with the $j+1$ individual (this generates 2 offspring)
 - choose the j or $j+1$ individual with the highest fitness and mutate it with probability c
 - if necessary compute its fitness
 - put this parent into the $t+1$ population
 - for each of the offspring check whether it is similar to the chosen parent and compare fitness values
 - if they are similar and the offspring fitness is less than or greater than that of the selected parent then mutate it with probability 1, otherwise mutate with probability c (this is the optimization step described above)
 - put the offspring with the highest fitness into the population $t+1$
- compute the total fitness of the t and $t+1$ populations

Results

The authors have implemented the above algorithm in a re-targetable architecture that they call SHARVIND. It is implemented in visual basic with an access database. It is re-targetable because of a query generator piece that transforms the individuals into a search space query. Since individuals are basically a where clause, this is rather easy if the search space is accessed via SQL.

The authors used SHARVIND on 3 databases. The first was a test database that held information about driver insurance risks. The results were known ahead of time and they tested that the algorithm could arrive at the proper classification rules. They tried different methods of determining the initial population. The database contained 10,000 tuples and the algorithm arrived at near optimal solutions in less than 1000 fitness evaluations. The number of fitness evaluations required was directly related to the quality of the initial population. The values of β and α were easy to determine since the content of the database was known ahead of time.

SHARVIND and the described genetic algorithm where then tried on two real databases containing aircraft incident data. The first was ECCAIRS, which contains incident data from a Scandinavian reporting system. The second was an FAA database containing similar

information. The paper describes various data cleaning steps that were performed on the data in successive iterations. The performance of the genetic algorithm was evaluated by a flight expert working with the authors. The flight expert said that the results were promising. He was able to explain almost all of the findings of the algorithm. The mining of the FAA database revealed some information that was previously unknown to the flight expert. Data related to unlicensed pilots was also stored in this database and indicated that this class of pilot was responsible for a significant proportion of the incidents and accidents.

The results proved promising. They do not mention any significant results, only that they have proven that a genetic algorithm can be applied to data mining problems and yield accurate results.

Opinion of Paper

We do not believe that this paper contributes any significant value to the field of data mining. The main contribution is that they have implemented (and developed small performance improvements to) previously derived genetic algorithms. The results on real databases yielded very little information that a flight expert did not predict. The paper describes a real data mining process. It covers the design of the algorithm, and the iterations and data cleaning required to arrive at significant results. A major shortcoming of this paper is that no comparisons are drawn to other algorithms. The introduction and related work section of the paper claims that their approach improves upon hill climber algorithms, solves problems with data partitioning, and provides other optimizations yet they do not compare the results to any of these things.

In the spirit of assigning numbers to express our opinion, ...uhhh...lets give it a 7.