Genetic Algorithms for Rule Discovery in Data Mining

Magnus Erik Hvass Pedersen (971055)
Daimi, University of Aarhus, October 2003

1 Introduction

The purpose of this document is to verify attendance of the author to the *Data Mining* course at DAIMI, University of Aarhus. First the concept of genetic algorithms (GAs) is outlined, then a brief introduction is given to data mining in general and rule discovery in particular, and finally these are combined to describe how GAs discover rules.

The referenced litterature is used throughout, usually without explicit citation. The reader is assumed to be familiar with meta-heuristics and related topics.

2 Genetic Algorithms

The following meta-heuristic is inspired by genetics. It basically consists of combining the best solutions so far and changing them slightly. This incorporates *Darwinian* evolutionary theory with sexual reproduction. Specifically, for a population P of *chromosomes* the following *operators* are applied:

- **Selection** deals with the probabilistic *survival of the fittest*, in that more fit chromosomes are chosen to survive. Where *fitness* is a comparable measure of how well a chromosome solves the problem at hand.
- **Crossover** takes individual chromosomes from P and combines them to form new ones.
- **Mutation** alters the new solutions so as to add stochasticity in the search for better solutions.

A variant known as *elitist* GA ensures the most fit solution survives intact, leading to a higher degree of *exploitation* rather than *exploration*.

2.1 Selection

When selecting chromosomes from P, different methods are available. The first one, *roulette selection*, chooses a chromosome with probability proportional to its fitness:

$$Pr(c) = \frac{Fitness(c)}{\sum_{c' \in P} Fitness(c')}$$

This can be likened to assigning chromosomes to slices of a roulette wheel, sized according to their fitnesses, and then selecting the winning slice after spinning the wheel.

Another method is known as *tournament selection* in which two chromosomes are chosen at random, then according to a fixed probability, either the more or the less fit chromosome is selected.

Since both these methods are probabilistic, it is possible - though unlikely - that only the worst fit chromosomes are selected. And it is likewise possible that the same chromosome will be chosen every time.

2.2 Algorithm

The algorithm is as follows:

- Initialize the chromosomes in P with random values.
- Until a termination criterion is met (e.g. number of iterations, stagnation of fitness, or fitness-threshold) repeat the following:
  - Create a temporary and initially empty set, P_s.
  - Select $n \leq |P|$ chromosomes from P and add them to P_s. If $n = 0$ then P_s is completely filled by the crossover in the next step.
  - Select $\frac{|P|-n}{2}$ pairs of chromosomes. Chop up each of the chromosomes, and combine the pieces to form the two offspring that are added to P_s.
  - Mutate some of the chromosomes in P_s, i.e. alter features at random.
  - Replace the population with the newly created: $P \leftarrow P_s$.

2.3 Binary Coded

A natural way of coding the chromosomes is by binary strings. This makes the transition from genetics theory to computer implementation rather straightforward. For example, the crossover of the two strings $x = x_1x_2$ and $y = y_1y_2$ may be the two strings x_1y_2 and y_1x_2 - known as *single-point crossover* - provided $|x| = |y|$ and they are split at the same point: $|x_1| = |y_1|$. Mutation can be done by flipping one or more randomly chosen bits in the string.

Although binary encoding is of course used for the implementation of all types of values in digital computers, there are more appropriate codings for the chromosomes. Since the genetic operators can then be better tailored to the search-space, chosing another coding scheme may improve actual performance, as well as easen the understanding of how the algorithm finds the improved solutions, aiding further development of the optimization scheme.

Because the original binary realization spawned some theory[1] formalizing the validity of the algorithm as a meta-heuristic search-procedure, other coding schemes were initially disregarded by some researchers. Abstraction is however one of the main tools of not only mathematics and computer science, but most sciences - not to say of *self-organization* and life itself.

2.4 Real Coded

One such coding arises naturally when considering function optimization in continuous search-spaces, where the chromosomes are now vectors of floating point numbers instead of bit-strings.

The selection procedure depends only on the fitness and not the specific coding, but new operators are needed for crossover and mutation. A plethora of different operators are available [5], for example *flat crossover* simply selects a value between the two chromosomes to be crossed - note that this only generates one offspring though, where the algorithm of section 2.2 assumes two. Another method is *simple crossover* which is similar to the crossover described in section 2.3.

Regarding mutation, the simplest is perhaps *random mutation* in which one or more values within the chromosome is simply chosen at random. A number of variations on this seek to choose the number more sensibly; for example adding a bipolar random number covering only a fraction of the search-space, may seem more appropriate.

2.5 Observations

Notice how the population size remains constant, whereas populations in nature have a tendency to grow unless the environment prohibits it. One reason for keeping it constant is a matter of computational resources, the proper analogy to nature would be for each chromosome to execute on its own computer. Another reason is stability of convergence.

Furthermore, there is only one species and one race. Implementing race in a GA would be similar to having subsets of P with more similar chromosomes, also called *niching*. Species is more difficult as the chromosomes are normally rather precisely sized candidate solutions. But it would be interesting to allow the evolution of chromosomes with different sizes (both smaller and larger), provided there is a sensible way of using them on the original problem.

A suggestion would be to use a *window*: If the chromosome is bigger, then choose only a portion of it matching the problem size. If it is smaller, only solve a certain part of the problem of size equal to the chromosome - with the fitness also somehow reflecting that only a part of the problem was solved. The actual growing or shrinking may be built into the mutation operator, and crossover between different species could be disallowed. Alternatively, the crossover operator could instead split the two chromosomes at different points, thus creating new chromosomes of inequal length.

[1] Including the socalled *Schema-theorem*.

This is somewhat similar to the *artificial immune system* described in [3] (p. 231), in which germs are bit-strings. The protective agents, socalled *antigens*, are bit-strings of arbitrary length, offering protection against any substring they encode. The antigens may also learn from eachother, and it turns out that the information they encode gets compressed, so that substrings recognizing germs start to overlap.

However, it may very well be that the larger chromosomes provide no improvement over simply increasing the number of fixed-size chromosomes.

3 Knowledge Discovery & Data Mining

The process of retrieving data from storage, extracting useful knowledge from it and delivering an abstract analysis to the user, is known as *knowledge discovery*. One purpose is to predict values from incomplete information, for example given the temperature the past few days what will it be tomorrow, or given a customer matching a certain profile, what kind of goods will she buy.

The actual extraction of a prediction[2] model is known as *data mining*, and a common approach is to split the input data D into two mutually exclusive and exhaustive sets, the training set: $T \subset D$ and its complement, the test set: $\overline{T} = D \backslash T$. The data mining algorithm must build its model based on the training set alone, and the accuracy of the model is then evaluated by using it on the test set.

The model is also assessed according to its comprehensibility - is the model understandable by a human - aswell as interestingness. These are of less import if the model is exclusively used in *machine learning*, the automated adaption of some agent or algorithm to its surroundings.

3.1 Pre- and Post-Processing

To the logistical end of pre-processing there is the possible need for integration of several data sources with appropriate mappings. Then the data may also need some cleaning, so as to avoid too noisy or unreliable information. More intruiging is the discretization of values into fewer classes (e.g. salary into: Low, medium, and high) which is claimed to produce more comprehensible knowledge [4], aswell as the selection of perspective on D, supposedly because the algorithm may otherwise find *inaccurate* knowledge (such as the *name and cleaning-assistant* relationship described below).

After data mining, the model may be simplified for comprehensibility and interestingness, the latter seems subjective at first, but may be objectively evaluated by finding the outlying patterns - which essentially is another data mining task known as clustering.

[2] Meant in the broader sense including clustering, classification, etc. and not just prediction of actual values in e.g. time-series.

3.2 Overfitting

Overfitting is the overinfluencing of the prediction model to anomalies in the training set T, that are not representative for the entire data set D. This may occur because the model is developed *too much*, when too few samples are present in T, or if they are too noisy. The extremity of this is *memorizing* in which an uncovered pattern is so specific that it only covers a single instance. In the worst case, the entire training set may be memorized, rendering the model useless. The inverse is known as underfitting, where the model is too general to express essential subtleties of the data-set.

Although [4] mentions as an example that a person's credit can not be deduced from her name, even though the data mining algorithm may in fact find such a pattern, the purpose of knowledge discovery is precisely to uncover previously unknown patterns in vast data sets.

For example the name *Magnus* was uncommon in Denmark 10 years ago, then suddenly a large number of infant males were given this name. Thus a Danish person of this name is more likely to belong to this younger generation. The name *Olga* is even more seldomly used - perhaps there is not a single Danish woman under the age of 80 with this name. Assuming that people over 80 who live in houses employ cleaning assistants, it can then be deduced that a person whose name is Olga and who lives in a house, employs cleaning assistants. Now, this sort of rule is not *universally* predictive, but for a given era in time, it may be highly accurate.

The socalled *memetic* view of mental processes, is that ideas - like evolution - do not develop at random; there is some mutation taking place, but often it is a combination of previous ideas. Given enough data (both in the sense of attributes for a given sample and the total number of samples) and a good enough predictive model, one might be able to foresee when the name Olga comes in vogue again, and how this implicates previously discovered patterns.

> When something out of the ordinary happens, it is ridiculous to say that it is a mystery or a portent of something to come. Eclipses of the sun and moon, comets, clouds that flutter like flags, snow in the fifth month, lightning in the twelfth month, and so on, are all things that occur every fifty or one hundred years. They occur according to the evolution of Yin and Yang. The fact that the sun rises in the east and sets in the west would be a mystery, too, if it were not an everyday occurrence. It is not dissimilar. Furthermore, the fact that something bad always happens in the world when strange phenomena occur is due to people seeing something like fluttering clouds and thinking that something is going to happen. The mystery is created in their minds, and by waiting for the disaster, it is from their very minds that it occurs.
> The occurrence of mysteries is always by word of mouth.
>
> - Hagakure, Yamamoto Tsunetomo, 1716

4 Rule Discovery

The data mining task of classification revolves around discovering rules of the form:

`IF <`*antecedent*`> THEN <`*consequent*`>`

Where the consequent is a finite set whose elements are called *classes*. That is, the task is to decide what class a single *target attribute* will be, given a number of *predicting attributes*. Naturally, the target attribute can not occur in the antecedent of a rule.

For example the credit of a person may be discovered to be good, if she has a job and a positive balance on her bank-account:

`IF ((`*has_job*`) AND (`*positive_balance*`)) THEN (`*good_credit*`)`

There are direct generalizations of the classification task, such as *dependence modelling* and *association rules*. But more interesting is data mining of *first order Horn clauses* [1], or *predicate logic*, that discovers relationships with variables, for example the concepts of family relationships.

4.1 GAs For Classification

Using GAs for classification we again need a representation of the chromosomes, reasonable operators, as well as the fitness function that measure how well the discovered rules work.

First off, if a chromosome encodes only one rule, it is known as a *Michigan* model, if an entire set of rules is encoded, it is a *Pittsburgh* model. Since a normal GA has a tendency to converge its entire population to a single point, it is necessary to run a Michigan model several times to discover a set of rules - or use a niching method as described in section 2.5.

Regardless of encoding of the rules, there is the question of how to decide the consequent, and measure fitness. As always, there are several solutions, one is simply to encode the consequent in the chromosome making it subject to evolution also. Another is to choose the class that has more samples satisfying the antecedent, or even maximizes fitness. The fitness itself can be defined as a ratio of how many classes were correctly predicted, as well as how many were not predicted, as in [4]:

$$Fitness(c) = \frac{TP}{A} \cdot \frac{TP}{TP + FN}$$

Where A is the number of samples satisfying the given antecedent, the *true positives* (TP) is the number of samples satisfying both the antecedent and consequent, and the *false negatives* (FN) is the number of samples not satisfying the antecedent, but satisfying the consequent. The lefthand (TP/A) is obviously maximized by overfitting and memorizing, where the righthand side counters this. If comprehensibility is also desired, the fitness function may also take this into account, e.g. by favouring simpler rules.

The actual representation or encoding of a chromosome is suggested in [4] to be binary. That seems a tad low-level, and it may be easier to develop and maintain an algorithm working on tree-based expressions, akin to *genetic programming.* The genetic operators are then modifications on trees, that only need to ensure validity of the resulting trees. When the data-set is to be accessed, if using relational databases the tree may easily be mapped (flattened) to SQL statements - these are rather speedy with properly indexed tables. There is no need to fear more abstract chromosomes on the basis of execution speed of the genetic algorithm, as the manipulation of abstract data-types is still negligible compared to the actual data access.

References

[1] Machine Learning
Tom M. Mitchell
McGraw-Hill 1997
ISBN 0-07-042807-7

[2] Genetic Algorithms
in Search, Optimization, and Machine Learning
David E. Goldberg
Addison-Wesley Publishing 1989
ISBN 0-201-15767-5

[3] Swarm Intelligence
James Kennedy, Russell C. Eberhart
Morgan Kaufmann Publishers 2001
ISBN 1-55860-595-9

[4] A Survey of Evolutionary Algorithms for
Data Mining and Knowledge Discovery
Alex A. Freitas
Pontificia Universidade Catolica do Parana
`http://www.cs.kent.ac.uk/people/staff/aaf/`

[5] Tackling Real-Coded Genetic Algorithms:
Operators and Tools for Behavioural Analysis
F. Herrera, M. Lozano, J. L. Verdegay
Artificial Intelligence Review 12 pp. 265-319
Kluwer Academic Publishers 1998
`http://decsai.vgr.es/~herrera/`