EVOLUTIONARY DATA MINING APPLIED TO TV DATABASES: A FIRST APPROACH

PANAGIOTIS ADAMIDIS, KONSTANTINOS KOUKOULAKIS

{adamidis, coskuk}@it.teithe.gr
Dept. of Informatics,
Technological Educational Institution of Thessaloniki, Greece 541 01.

ABSTRACT

Recently, Evolutionary Algorithms (EAs) have been applied with very good results to various types of data mining problems. This paper presents the initial results of our work on data mining from TV program databases using EAs. We compare the performance of different operators and different operator parameters.

INTRODUCTION

Evolutionary Algorithms (EAs) have already proved their usefulness in scientific and real-world problems quite successfully. They are stochastic search techniques that explore combinatorial search spaces using simulated evolution. The primary objective of an EA is either to find something – whether this is known or not – or accomplish a goal, or, more generally, to solve a problem.

EAs maintain a population of individuals that evolve according to a set of rules regarding selection, recombination and mutation. The population is evolved towards the optimum, concentrating search in those areas of higher fitness (exploitation). Recombination and mutation perturb these individuals providing general heuristics for exploration. The way evolution is conducted follows the rule of the "survival of the fittest" and that's how the algorithm measures success. Figure 1 shows a somewhat formal and tangible representation of a simple EA

1. Generate an initial population of solutions (individuals) randomly
2. Repeat
    I. Evaluate the fitness of each individual
    II. Select "parent" solutions from the population based on their fitness
    III. Apply operators to parents and get generation P(I+1)
3. Until termination_criterion

Figure 1. A Simple Evolutionary Algorithm

As EAs seem to handle quite a variety of problems, their utilization in the sensitive area of computer databases was and is of tremendous interest. But, one has to take careful and well-defined steps if he is at least curious of their application. To be more specific, we could have a word or two about DBMSs (DataBase Management Systems).

The primary objective of a DBMS is to store and manipulate all sorts of data regarding a certain area of interest. This kind of generality and the formalization of processing they possess has made this field one of the most exciting and growing ones of the industry. The particular format in which data is handled and the language used to process it has varied over time for the sake of speed and size. But it took people some time to realize that data have another dynamic aspect besides their more formal one, the aspect of knowledge discovery, more formally referred to as **data mining**.

The exploration of huge quantities of stored data is now believed to be a critical factor in the decision a company or any other interested individual or organization may take. Having in mind that the actual data of a database truly give the image or a snapshot of a closed system over a given period of time, the necessity of a processing mechanism for "interesting data pattern" extraction was obvious.

The contribution of this paper is the presentation of the aforementioned topics and their example interaction for the production of simple but meaningful results in the context of a demonstration problem, together with a light discussion in the customization of the whole process.

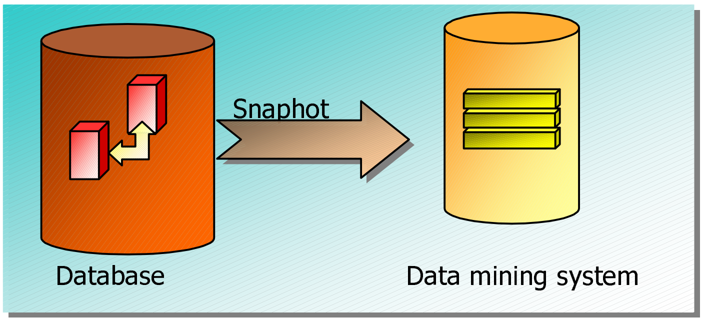

Figure 2. Basic data mining process

EVOLUTIONARY ALGORITHMS AND DATA MINING

Even though these two broad fields seem unable to connect or interact in some way, it is the revolutionary nature and the generality of the EAs that makes any notion of processing sensible.

As already mentioned, generally, an EA consists of certain steps that take place in order for a problem to be solved. It is necessary, though, to delve a bit into the terminology that surrounds such an algorithm, in order to be able to clarify and get a grasp of these ideas.

Basic biological terminology and evolutionary processes

Each living organism can be fully described by a set of chromosomes. A **chromosome** can be divided into various genes. Each **gene** contains a small descriptive information, for example hair or eye colour. A cell and each cell of an organism contain many chromosomes. Taking exactly that set of chromosomes, we are able to describe the organism in its entirety. This set is called the **genome** of the organism.

Furthermore, we call **genotype** the actual chromosomes of the genome, and **phenotype** the characteristics that the genotype eventually gives to the organism. Now, in nature we have both **haploid** and **diploid** organisms. The genotype of the haploid organisms has its chromosomes unpaired, whilst the diploid organisms pair their chromosomes.

For a new organism to be created, reproduction takes place between two haploid or diploid organisms. We can talk of the **recombination** phase, in which for diploid organisms, each pair of chromosomes of the two parents exchanges genes. That is how each parent forms a **gamete**. Then, the two gametes pair and generate the new genome. As far as haploid organisms are concerned, we merely have an exchange of genes between the unpaired chromosomes of the parents. Once the recombination phase is completed, **mutation** of the offspring occurs. During this phase, individual components (formally called **nucleotides**) of the chromosomes of the offspring change. This phase mainly exists because of the errors that may happen during reproduction and having this in mind we very well sense nature's procedures. The picture summarizes and clarifies all of the above. Finally, each organism has a measure of either its **fertility** (how many children it may have) or its **viability** (a probability that the organism may continue to exist). This measure is called the **fitness** of the organism. But how exactly an EA manipulates organisms and reaches its goals?

A very important aspect of an EA is the representation of a chromosome and eventually its genes. It must be said that there is no universally accepted proposal of a representation. Usually the representation of an individual must be tailored to the problem at hand.

An EA has to do with a set of organisms, called a **population**. Because of the search tasks the EAs commonly face, the space of all possible solutions may be called the **search space** of the EA. So, a search space consists of individual solutions to the problem and it is perfectly right to assume that a distance between those candidate solutions may exist. The calculation of the distance between two solutions is related directly to the representation of a chromosome and may well differ among various EAs.

For a reproduction to take place, a **selection** of the two (or more in some cases) parents must must occur. Again, there are many ways in which we can select reproductive partners and the majority of them relies on chromosome fitness – the fittest the chromosome, the more likely it is to be selected for reproduction.

When the algorithm has completed a certain number of reproductions, then it is said that the EA has completed a **generation**.

At first sight, a simple selection based on just the fitness of the chromosomes would give desirable enough parents. But, what if we were working in a **structured** population? The idea behind *structure* and its very implementation is **locality** (Flockhart and Radcliffe, 1995). Locality states that the population is divided into groups of chromosomes, called **demes** and chromosomes are selected based not only on their fitness but on how close they are too. The main advantage of locality is that it improves the effectiveness of the algorithm, because it reduces the number of chromosomes that must be selected to achieve a given quality.

The meaning and processes of Data Mining

In the simplest case, we have to do with a single database which contains enough data to ask ourselves: "Is this only boring data I have, or can they tell me something?", "What must I do to take a clearer picture of reality the time I gathered that data?". The importance of these questions gives rises to a more serious approach.

After some thought around this topic, we could organize those "questions" into categories [1, 2]:

- **Undirected data mining:** This is the simplest case of data mining. Mining here is set free by the user who is just interested in getting any kind of patterns his data may have, any sort of relationships that may be assumed. This category of data mining is most frequently used, although the user could slightly restrict the system.
- **Directed data mining:** In this kind of data mining, the miner is significantly more constrained, because he searches the database(s), taking into account a somewhat half-defined pattern. Therefore, he discards the patterns he finds unrelated to the description at hand and seeks for quality patterns including the elements specified by the user.
- **Hypothesis testing & refinement:** In this case, the user has a hypothesis in mind about his data and desires to find out whether it is true or not. The system not only can respond to that question but it can refine this hypothesis into a truer, thus more interesting, one.

But that's not all about mining. There are a few but important other aspects of data mining that surely affect the application and the effectiveness of it. We can see a couple of them in the next few paragraphs.

- **Database sampling:** In most of the cases that present an interest, we have to do with a large database and even a lot of them. The most time-consuming task of the whole data mining process is usually the evaluation of the patterns collected. It would surely be nice to work with smaller databases but still get the same valuable results. This is what sampling promises. Sampling is the process of identifying representative parts of a database to use in data mining. What's more, mining the whole of a large database may incur **over-fitting** (Fayyad, and Piatesky-Shapiro, 1996).
- **Database preprocessing:** The idea of preprocessing is mainly based on the need to prepare the data before they are actually used in pattern extraction. There are no standard preprocessing practices, rather some frequently used ones such as: (a) the extraction of derived attributes, that is quantities that accompany but are not directly related to the data patterns and may prove meaningful or increase the understandability of the patterns· (b) the removal of some existing attributes that should be of no concern to the mining process due to their insignificance.

Evolutionary Algorithms + Data Mining

Given all of the above, we can now set the foundation for our promising EA. Combining the general structure and methods of an EA and the needs that data mining seeks to cover, we can state that *the goal is the discovery of meaningful and interesting data patterns through a defined number of generations that the candidate solutions should evolve or until a desired pattern quality is achieved* (Fig.3).

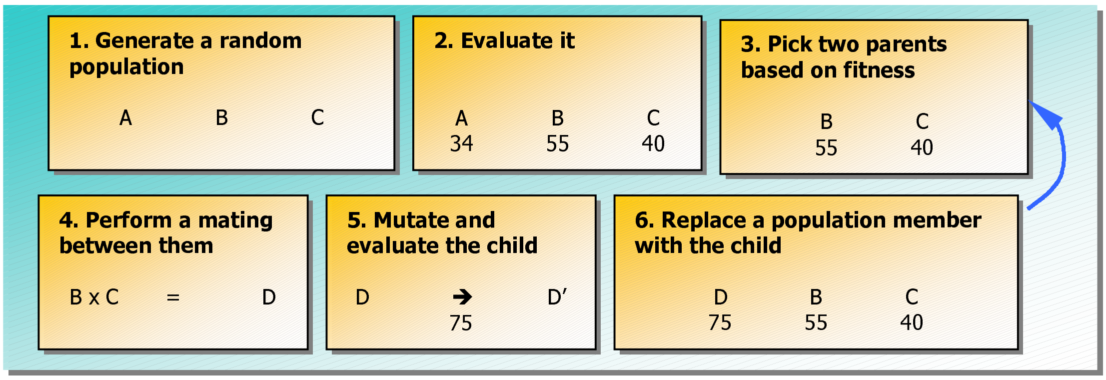

Figure 3. Evolving data patterns

The most important questions that need to be answered are outlined below:

- How will a candidate solution be represented?
- What kind of population should be used?
- What will be the selection method?
- Which factors should the fitness function consider?
- How will the various kinds of data mining be modelled into the algorithm and specifically into the syntactic operators (recombination and mutation)?

These questions will be answered in the next pages.

PROBLEM SPECIFICATION

Supposing that all of the preceding concepts have been sufficiently covered, we will now describe the problem that the EA is called to solve.

The EA mining system built should be able to extract "n" data patterns (rules) from a database that contains television broadcast data. The database has only historical data, that is, past broadcasted schedules. Therefore, the interested individual, e.g. the owner of the station, given the rules, can be informed for possible relations between attributes and plan future schedules accordingly.

Our database has been built using the widespread relational model. The Entity-Relationship diagram is shown if Figure 4.

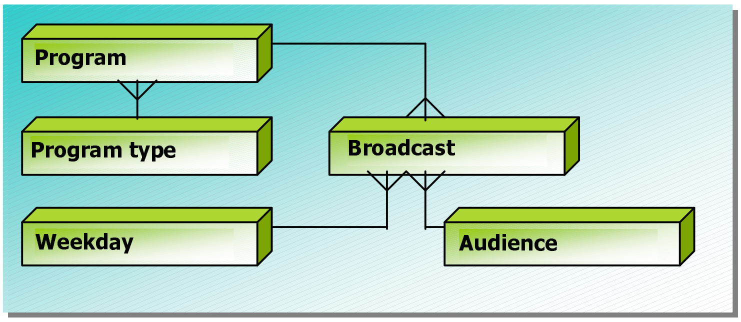

Figure 4. Entity-Relationship diagram

A brief description of each table presented above can be found in Appendix A.

The most important attribute is the *program*. This corresponds to a television program (e.g. “Sherlock Holmes”). There are *program types* (e.g. “Detective serials”) and programs are of a certain type. A *broadcast* is transmitted for a certain program, on a certain *weekday* and scores some *audience*.

It should be noted that this is only a formal description of the database. The processing is by no means performed onto data stored in that form (see sub-section II in the next section).

THE SOLUTION

The problem has many facets. It would be better to deal with one facet at a time to preserve scrutability. We begin with the configuration of the EA.

The Evolutionary Algorithm

The EA is mostly based on that of the GA-Miner project (Flockhart and Radcliffe, 1995), with a few differences.

First of all, it uses a two-dimensional structured population. It contains haploid organisms with a single chromosome and that, of course, is their genome also. “Structured” means that each chromosome mates with a chromosome that belongs to its surrounding deme. This second parent is selected using tournament selection. After the parents have been specified, mating takes place using the recombination operator and an offspring is born. This offspring is then subject to the mutation operator, which may alter it or not. The chromosome to be put in the position of the original chromosome is selected using tournament selection between the original and the offspring chromosome. That is, there is a probability to discard the offspring.

This mating procedure is performed for every chromosome of the population and after that a generation is completed. The user specifies a number of rules (data patterns/chromosomes) that he would like to obtain after a complete run of the EA. After each generation, the EA considers the best n chromosomes of the population for inclusion in that specified rule set. The rule set is updated as follows: The rule set is kept at its maximum size, while the algorithm replaces either its lower fitness rule, or the most similar one by a higher fitness rule.

It must be noted that chromosomes are subject to a kind of pruning before their evaluation. This helps keep them more concise and avoid overwhelming patterns that may confuse the user.

Representations regarding mining

Before attempting to analyze the format of a chromosome, it is better to clarify the way mining will take place.

As already mentioned, there are at least three types of data mining. At present, this paper deals with the first one, that is, undirected (pure) data mining. This kind of mining allows us to extract explicit data patterns (Flockhart and Radcliffe, 1995). The explicit data patterns consist of three subsets· S, C and P. S stands for Specificity, C for Condition and P for Prediction. Each subset refers to a portion of the database. In order to form a rule, they are connected by an implication. Figure 5 gives a conceptual form of a data pattern, which is also depicted as a Venn diagram.

When **S**, if **C** then **P**

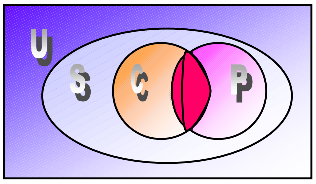

Figure 5. Single (one-way) implication

The chromosome format

The population is initialized using a template for each genome. Describing that pattern template equals to describing the format itself.

The pattern template consists of various parts, corresponding to the ones needed to fully describe a rule. Those parts and their attributes and relationships that may have with each other are given below.

Subset: A subset may contain a number of disjunctive clauses or none. In the latter case it is said that the subset forms a tautology (covers the whole database). Also, a subset may be fixed, in which case addition/deletion of the clauses it may contain is prevented.

Clause: A clause may contain a number of terms connected with the boolean 'and' operator. Again, a clause may be fixed. In this case addition/deletion of its terms as well as its own deletion is prevented.

Term: A term has the following form: '*Variable*' = '*Description*'. *Variable* refers to a database variable. *Description* is explained below. A term may be fixed, in which case its deletion by the algorithm is prevented.

Description: A description has one of the following forms: '*Value*' or '*Numeric*'...'*Numeric*'. The first one refers to an integer or string value and the second specifies an integer or real range. A *Value* or a *Numeric* may be *fixed* or *initialized*. *Fixed* means that it will surely remain unaltered by the algorithm while *initialized* means the opposite. Note that a term may have more than one values.

Finally, we are able to allow or not certain database variables to a subset and to define the maximum number of values, terms and clauses for a subset. Variables may be characterized as dependent or not but the algorithm doesn't take this into account yet.

The evaluation method

But how exactly are we going to evaluate a pattern? Before answering this daring question let's consider what factors influence how interesting a pattern is.

A common problem in data mining is avoiding "truisms". A truism is a statement that is generally known to be true, thus it is of little interest. For example, it is widely known that an audio compact disk has only audio tracks, so a pattern expressing just that would be of no use. One way to bypass truisms is to separate the variables of our database into functionally dependent and not dependent ones. If we specify that a variable is dependent we state that its value depends on the values of one or more independent ones. For example, whether it will rain tomorrow in Alaska depends on whether there are clouds above it or not.

As far as this paper is concerned, no such action has been taken to eliminate truisms, although the aforementioned variable distinction has been implemented.

A data pattern must be general, that is, it must give the user a substantial and clear image of his database. A pattern must also be accurate and that means that it must be concise and precise. However, one could rightfully say that generality and accuracy are contradictory. A measure of accuracy and one of coverage (generality) is shown below:

$$accuracy = |S \cap C \cap P| \Big/ \left(|S \cap C \cap P| + |S \cap C \cap \overline{P}|\right) \qquad (1)$$

$$coverage = |S \cap C \cap P| \Big/ \left(|S \cap C \cap P| + |S \cap \overline{C} \cap P|\right) \qquad (2)$$

The evaluation function used is the rule interest function suggested by Piatetsky-Shapiro [2, 4], a relatively simple but powerful enough function

$$|S \cap C \cap P| - \left(|S \cap C| \times |S \cap P|\right) / |S| \qquad (3)$$

Data format

In this section we give a picture of how data is stored in order to be processed by the algorithm and some details about the encoding it has followed. Also, preprocessing is analyzed and the details of creating the sample database are mentioned.

What types of variables (fields) are supported by the data mining system? A variable may be quantitative (e.g. *Car sales*) or categorical. Categorical variables may be further divided into enumerated (e.g. *Car brand*), ordinal (e.g. *Index number*) and hierarchical (e.g. *Vehicle type*) ones.

Pattern terms referring to categorical variables are called *value terms*, while terms of quantitative variables are said to be *range terms*.

Database format

In order to construct a pattern genome, the user has to supply the definition of the database, that is, the description of each field (otherwise named *variable*) of the database.

The algorithm cannot process the data when it lies in normalized tables. So, it was necessary to create a denormalized table that would contain all data. This table is stored in a simple text file with the following format: Each line of the text file represents a database record and contains its values delimited by commas. The database fields (attributes) in each record lie in the following order:

WEEKDAY(COID), DAY(COI), MONTH(COI), YEAR(COI), PROGRAM(CES), PROGRAMTYPE(CEID), DURATION(QI), SHOWHOUR(COID), SHOWMIN(COID), AUDIENCE(QID)

The number in parentheses indicates the type of the variable. Q stands for Quantitative, CE for Categorical Enumerated, CO for Categorical Ordinal and CH for Categorical Hierarchical. Also, I means that the variable accepts integer values and S that it is a string variable. A D is appended where necessary to indicate a dependent variable.

Just before the algorithm is run, the text file is loaded in main memory for faster access during the evaluation of the chromosomes. The algorithm scans the database and calculates the fitness of each individual according to the evaluation function. Figure 6 depicts the data encoding and the acceptable values for each variable.

- WEEKDAY : 1..7 (1=Sunday)
- DAY : 1..31
- MONTH : 1..12
- YEAR : 0..3000
- PROGRAM : 0..59
- PROGRAM TYPE : 0..9
- DURATION : 5..180
- SHOWHOUR : 0..23
- SHOWMIN : 0..59
- AUDIENCE : 0..10000000

Figure 6. Data encoding and acceptable values

Database preprocessing

After the database has been loaded in memory and before the specification of the pattern chromosome and the initialization of the algorithm, we take the following preprocessing steps:

- Calculation of the number of database records
- Concentration of the possible values a *value* term may include. To accomplish that, we must scan the database and collect all of the values a categorical variable has.
- Definition of the boundaries of a *range* term. To accomplish that, we must scan the database and find the minimum and maximum value for each quantitative variable.

After preprocessing, the user is free to define a pattern genome and run the algorithm.

Sample database creation

Even though the data are not real we tried to create data that approximate reality.

There are 10 program types, each with its broadcast zone. The table *ProgramTypeZone* contains these zones. Each program type has 6 programs, so another table, *Programs,* would associate programs with their types. Now, each program is available only certain weekdays, so there's another table, *WeekdayAvailability*, that contains this information. Also, each program is available only certain years and months and tables *MonthlyAvailability* and *YearlyAvailability* are created. Again, each program has a duration (table *ProgramDuration*) and 36 different sizes of duration are supported (table *Duration*).

In order to create a realistic database, we considered calculating audience onto a percentage of the available spectators. Factors that influence the spectator availability are: Weekday (table *PctWeekday*), Program type (table *PctProgramType*), Month (table *PctMonth*), Hour within which a program is starts broadcasting (table *PctHour*), Duration of a program (table *PctDuration*) and the Program itself (table *PctProgram*).

The database contains data for 2 years and those are defined in table *Years*. Finally, table *Spectators* contains the available spectators for each year. The database contains 4144 records.

The program takes into account all of the above tables, which impose their restrictions, and creates a television schedule as random as possible. It must be noted that due to the small size of the database, no sampling was considered necessary.

5. RESULTS - DISCUSSION

We run experiments for three different genomes. For each genome we run six different configurations of the EA with different probabilities of uniform recombination and one point recombination, and two different probabilities of term mutation. Table 1 shows the difference between the probabilities used by each configuration.

Table 1. Configurations of the Evolutionary Algorithm

Config. No	Uniform recombination	One point recombination	Term mutation
1	0	1	0.6
2	0.5	0.5	0.6
3	1	0	0.6
4	0	1	0.3
5	0.5	0.5	0.3
6	1	0	0.3

The population size of the EA was 100 individuals (10x10). We run each configuration 5 times.
Table 2 shows the average best fitness of each configuration.

Table 2. Average best fitness of each configuration

Config. No	Genome 1	Genome 2	Genome 3
1	9	4	4
2	9	4	4
3	9	4	4,3
4	8,5	4	4
5	9	4,7	4,5
6	6	4	3,75

We cannot compare the average best fitness values of the different genomes since a different genomes have different representations and fitness values. The algorithm is very time consuming. Average time for each run is about 2 days on a Pentium 266. We need a lot more runs in order to evaluate the significance and the results of the EA.

Our initial results show that the EA can discover previously unknown knowledge in the TV database that we used.

Even though it needs more experimentation, we can assume that the different operator probabilities have a direct impact on the performance of the EA. It seems that the worst configuration is the configuration 6, which uses only uniform crossover and smaller probability of term mutation. It seems that the best configuration is number 5 which uses both uniform recombination and one point recombination with probability 50% and smaller term mutation.

Even though we intend to continue our research using the configurations that we described earlier, we also intend to check the impact of other parameters on the performance of the algorithm.

There are even more questions that have to be answered. This paper is only a first approach to knowledge discovery in databases with TV data.

REFERENCES

U.M. Fayyad, and G. Piatesky-Shapiro, 1996. Advances in knowledge discovery and data mining. *AAAI Press: MIT Press*.

I. W. Flockhart, and N.J. Radcliffe, 1995. GA-MINER: Parallel Data Mining with Hierarchical Genetic Algorithms. Final Report, The University of Edinburgh.

T. Back, 1996. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. *Oxford Univ Press*.

G. Piatetsky-Shapiro, 1992. Knowledge discovery workbench for exploring business databases. *International Journal for Intelligent Information Systems*, 7(7):675--686.

Appendix A: Database table descriptions

Following are the tables of the sample database:

Program type
Class

Weekday
Day of week

Broadcast
Code
Day of week
Program name
Time

Program
Name
Class
Duration

Audience
Broadcast code
Date
Spectators