EVOLUTIONARY ALGORITHMS FOR SQL QUERIES

by
Monica Chiş

Abstract: Data mining is defined as the process of discovering patterns in data. The most common tasks of data mining are feature extraction, feature selection, classification and clustering. Evolutionary algorithms are randomized search procedures inspired by the mechanics of genetics and natural selection. Evolutionary algorithms are often used as optimization algorithms, and this is the role that they play in most data mining applications. In this paper a new technique for data mining using evolutionary algorithms is proposed. Evolutionary algorithms are among standard modern instruments for data mining and thus they are included in this paper. Our paper presents a methodology for applying the principles of evolutionary computation to knowledge discovery in databases by using SQL queries that describe datasets. The first step in applying an evolutionary algorithm to solve a problem is to find a representation for candidate problem solutions. In this paper, we work with the general form of the SELECT statement and we try to realize combination between evolutionary algorithms and SQL queries. The proposed approach finds a new representation for SQL queries to allow application of an evolutionary algorithm to evolve them for explore the entire databases. Another issue of this paper is the fitness function to apply evolutionary pressure to the queries, to guide them towards the correct classification rules. The starting point for the considerations presented in this paper is the possibility of explores completely a databases or a table in a databases.

1. Introduction

The large amount of data stored in databases contains valuable hidden knowledge, which could be used to improve the decision-making process of an organization. For this reason, it is necessary to improve the methods for analyzing data. Data mining is an interdisciplinary field, using methods of several research areas (specially machine learning and statistics) to extract high-level knowledge from real-world data sets. Data mining is the core step of a broader process, called knowledge discovery in databases, or knowledge discovery, for short. This process includes the application of several preprocessing methods aimed at facilitating the application of the data mining algorithm and post processing methods aimed at refining and improving the discovered knowledge ([6]).

This paper discusses the use of evolutionary algorithms in data mining and knowledge discovery. Data mining consists of the efficient discovery of knowledge from databases. This paper presents a new evolutionary for discovering a few interesting information from databases. Particularly our 2 approach proposed a new method for evolving SQL queries. We proposed codification for WHERE statement in SQL a command.

We take the word “databases” as referring to large datasets (at least tens of thousands of tuples) maintained in a DBMS. Due to the very large amount of data,

these databases must be accessed through efficiently executed DB queries (expressed in SQL in the case of relational databases).

Data mining is defined as the process of discovering patterns in data. The process must be automatic or (more usually) semi-automatic. The patterns discovered must be meaningful in that they lead to some advantage, usually economic advantage. The data is invariably present in substantial quantities. ([13])

SQL – Structured Query Language is an ANSI (American National Standards Institute) standard computer language for accessing and manipulating database systems. SQL statements are used to retrieve and update data in a database. SQL works with database programs like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase, etc. There are many different versions of the SQL language, but to be in compliance with the ANSI standard, they must support the same major keywords in a similar manner (such as SELECT, UPDATE, DELETE, INSERT, WHERE, and others). Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard.

SQL provides full database functionality. A main attraction of SQL lies in its usage flexibility and inter-operability. SQL allows a user to express a query, without the need to specify how the query is actually processed. The Structured Query Language offers database users a powerful and flexible data retrieval mechanism such as the SELECT statement.

The next section of our paper described all the preparation necessary for applying the proposed evolutionary algorithm and the general task about SQL that are used in our approach.

2. SQL queries

This paper proposed a new evolutionary representation for the general form of the SELECT statement. Our goal is to realize a combination between evolutionary algorithms and SQL queries. A database most often contains one or more tables. Each table is identified by a name. Tables contain records (rows) with data. With SQL, we can query a database and have a result set returned. SQL (Structured Query Language) is syntax for executing queries. But the SQL language also includes syntax to update, insert, and delete records. These query and update commands together form the Data Manipulation Language (DML) part of SQL:

SELECT - extracts data from a database table

UPDATE - updates data in a database table.3

DELETE - deletes data from a database table

INSERT INTO - inserts new data into a database table

Our approach works with SELECT Command. The general form of the SELECT statement appears below:

```
SELECT select_list
FROM source
WHERE condition(s)
GROUP BY expression
HAVING condition
ORDER BY expression
```

The first line of the statement tells the SQL processor that this command is a SELECT statement and that we wish to retrieve information from a database. The select_list allows us to specify the type of information we wish to retrieve. The FROM clause in the second line specifies the specific database table(s) involved and the WHERE clause gives us the capability to limit the results to those records that meet the specified condition(s). The final three clauses represent advanced features outside the scope of this article we'll explore them in future article.

In our paper the conditions clause is considered. In our approach an evolutionary algorithm is used for representing condition clauses. Consider that each field in a table (each attribute) could be part of the condition statement. In this paper we consider that a field could appear many times.

2. Solution representation for SQL queries

First of all, for applying our evolutionary algorithm it is necessary do some transformation in our data set. Consider that all data set are stored in a table. Each field will be indexing, using a unique index. In this way we obtain all the possible valued for a field in our data set. All of these possible values are encoded with a value from 1 to the total number of unique values for that field. All this preparations is made by a Relational Databases Management System and gives the value domains of our representation.

Our approach gives the possibility to explore completely a table included in a database. The proposed approach finds a representation for SQL queries to allow application of an evolutionary algorithm to evolve them for explore the entire data in the tables. Another issue of this paper is the fitness function to apply evolutionary pressure to the queries, to guide them towards the correct classification rules.

In our fist approach of this problem we proposed a representation for SQL queries in which the chromosome length is constant and is equal with 2p-1, where p is the total number of field in the table.

In this paper we proposed a different approach for chromosome representation. Genotypes were required to encode the list of conditional constraints that specify the criterion by which records should be selected. Each conditional constraint in SQL follows the structure

[attribute name] [logical operator] [value].

Each condition clauses in SQL queries could be composed of one or more structured of this form.

Each of this structured could be linked by a logical operator **and** or **o**r. For representing the chromosome it is necessary to encode the conditional constraint in SQL follows the structure

[attribute name] [logical operator] [value].

Our approach proposed for this structure a special representation. According to the proposed order for input dataset this numerical representation respects some specific requirements.

The chromosome length is not constant. We consider the chromosome length a parameter of our algorithm. We denote by l the parameter chromosome length. l could be a value between 3 and 2*p-1, where p is the maximum fields number of analyzing table.

An individual is represented as a vector:

$$c= (c_1, c_2, \ldots c_l)$$

where c_j is a string which takes different values if is in an odd or in an even position.

Because of the logical operator that links the conditions all the chromosome length will be an odd number. There is a difference between a gene in an odd positions and a gene in an even positions. Each gene in an odd position of our proposed representation of an individual represents a condition of the structure

[attribute name] [logical operator] [value].

Each gene in the odd position has three parts first represent the fields number which enter the condition, the second represent the logical operator an the third represent the encoding for the field possible value. All these parts are separated by "$". Each gene in an odd position and has the form listed below:

g1$g2$g3,

where

g1 is an integer number which represent the number of field which enter the condition;

g2 is an integer number which represent a codification of a logical operator which is part of the condition;

g3 is the codification of the field value obtained after indexing each field in analyzing table.

g3 takes one of the following values

1 for <

2 for =

3 for >

4 for <=

5 for >=

6 for <>

This value could be updated any time. We could add new codification for new logical operator.

Each gene in the even positions in the proposed chromosome is given by:

- 1 if the logical operator that linked the conditions is **and**
- 0 if the operator is **o**r.

For generating the chromosome first step is random generated of a number, denoted by f, which represented the number of fields that enter the condition. Then the chromosome length l is calculated with the relation

$$l = 2 \cdot f - l .$$

Generating a solution for our problem is a two-step process: first is generated the length of the chromosome and then is generated the chromosome.

In the next section we consider an example of our approach for encoding solution. We conside a very short number of records for our example.

Consider a table of databases, which has four fields (numeric and character type). The entire table is described in Table 1.

Field 1	Field 2	Field 3	Field 4
1	A	1	6.7
2	A	2	4.3
2	A	3	5.7
1	C	1	7.1
3	B	3	5.9
1	B	4	5.6
1	C	2	5.5
3	C	3	6.4
3	C	4	5.8
2	A	2	5.9
2	B	3	7.4
2	C	4	7.1

Table 1

Field1	Code
1	1
2	2
3	3

Table 2

field2	code
A	1
B	2
C	3

Table 3

field3	code
1	1
2	2
3	3
4	4

Table 4

field4	Code
4.3	1
5.5	2
5.6	3
5.7	4
5.8	5
5.9	6
5.9	7
6.4	8
6.7	9
7.1	10
7.1	11
7.4	12

Table 5

Table 2 – 5 contains the encoding for each value of each field after an operation of indexing using a unique index.

Consider the number of fields that are part of the condition of WHERE statement of SELECT command is 3. Then the length of the chromosome is 5.

A condition of the form

Field1=1 or field1=3 and field2='C'

has the representation listed below:

c=(1$1$1 2 1$1$3 1 2$1$3).

4. Fitness function

To realize a correct search it is necessary to compare the chromosome. For this reason is important to find a function used for evaluating the chromosome.

Our proposed evolutionary algorithm is a special case of the evolutionary algorithms. We consider the number of record that fulfilled the conditions gives by the chromosome the fitness value.

Proposed approach considers that the greater number of record represents the best the SQL queries because represent the entire data set.

5. Evolutionary Algorithm for SQL Queries

Using the proposed solution representation, an evolutionary algorithm is used to evolve a population of SQL queries encoding with number. Search operators used in proposed approach are crossover and mutation ([1], [2], [6]).

Mutation makes changes in conditions for a field or attribute and crossover operator is important for entire dataset that is analyzed.

For selection we use tournament selection operator. Proposed algorithm considers the uniform mutation and pm the mutation rate.

Proposed evolutionary algorithm uses uniform crossover operator but with some considerations ([2]). Uniform crossover does not use a predefined crossover points. For each genes of an offspring, a global parameter indicates the probability that this gene should come from either the first of the second parent. Each position of an offspring is calculated separately ([4]). We consider as well the one point crossover operator.

For crossover it is necessary to use a special operator. We defined a SQL crossover operator. This operator is used because of the chromosome length is not constant.

If the crossover uses chromosomes with the same length then we use a general uniform crossover operator. If the chromosomes that are recombined are of different length then the offsprings could take the length of boat of them parents. For this case, a parameter gives the value of the offspring length. This parameter is denoted by cl and takes value between the less length and the great length of parents. After this is established we apply the uniform crossover in special conditions. The part that is not present will take value from the parent with the greater length.

The mutation probability stands for a parameter of our evolutionary algorithm. Consider pm the mutation rate. For each gene of the chromosome population, a uniform random number q is generated. If for the i-th gene, the condition m p q <is fulfilled, that gene is selected for mutation.

Binary tournament selection is considered. Binary tournament selection implies that two individuals directly compete for selection. Tournament selection used is without reinsertion of the competing individuals into the original population.

Proposed Evolutionary Algorithm for SQL Queries **(SQLEva**l) is outlined below:

Evolutionary Algorithm for SQL Queries (SQLEval)

1. Indexing table fields successively and take the value (from 1 to the maximum number of unique indexing record);
2. Initialize t = 0 (t is the population number).

3. Initialize the population P(t). For the first generation random initialization is used. The chromosome length is calculating based on the number of fields in the conditions. Each part of each gene takes value for his own domain. Apply a procedure for transform the chromosome representation in SQL language for evaluate the number of records, which satisfy the conditions. Evaluate P(t) by using the fitness function.

while (termination condition not satisfied)

4. Apply binary tournament selection for P(t). P 1 is the set of the selected solutions.
5. Individuals from P 1 entering the mating pool based on tournament selection. Choose chromosomes from P 1 to enter the mating pool..7
6. Apply the crossover operator to the solutions from the mating pool. A new intermediate population P 2 is obtained. Mutate solutions in P 2 offspring enter the next generation P(t+1).
7. Set t = t + 1.

end while

The termination condition of our evolutionary algorithm is t=n, where n is a pre-specified number of generations.

The algorithm keeps the best individual obtained up to each generation t. The problem solution is the best individual obtained from the best individual of each generation.

A procedure, which realizes the decoding, is used after applying the proposed evolutionary algorithm. This procedure return the record specify by the best chromosome.

6. Conclusions and further research

The importance of this paper is the new representation of the chromosome for a SQL queries. We consider that the transformation or the operation that must be done before applied the evolutionary algorithm are not very complicated and takes time but could be efficiently.

Proposed approaches gives the possibility to know more about the records in databases and to find the combination of conditions that is more representative for analyzed dataset. The combinations of conditions more representative are that which give the maximum number of records.

Further research will explore different fitness functions and the possibilities to defined new genetic operators as well as the encoding for other clauses of SQL SELECT statement.

7. References and bibliography

[1]. Bäck T., Fogel D.B., Michalewicz, Z, (1997), Handbook of Evolutionary Computation, Oxford University Press, Oxford.
[2]. Dumitrescu, D., Lazzerini, B., Jain, L., and Dumitrescu, A., (2000), Evolutionary Computation, C.R.C. Press, Boca Raton, FL.
[3]. Cantú-Paz, E., Kamath, C., (2001), On the use of evolutionary algorithms in data minin**g,** in Data Mining: A Heuristic Approach, H. A. Abbass, R.A. Sarker and C. S Newton (Eds.), Idea Group Publishing.
[4]. Fayyad, U. M., Piatetsky-Shapiro, G. & Smyth, P., (1996), From data mining to knowledge discovery: an overview, In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy, R. (Eds.) Advances in Knowledge Discovery & Data Mining, 1-34. Cambridge: AAAI/MIT.
[5]. Freitas, A.A., (2001), A survey of evolutionary algorithms for data mining and knowledge discovery, Advances in evolutionary computation, A., Ghosh, S., Tsutsui, S. (Eds.), Springer-Verlag.
[6]. Goldberg, D.E., (1989), Genetic algorithms in search, optimization, and machine learning, Reading, MA: Addison-Wesley..8
[7]. Han, J., Fu, Y., Wang, W., Koperski, K., Zaiane, O., (1996), DMQL: A data mining query language for relational databases, Proceedings of the ACM SIGMOD International Conference on Management of Data.
[8]. Holsheimer, M., and Siebes, A., (1994), Data mining: the search for knowledge in databases, Report CS-R9406. Amsterdam, The Netherlands: CWI.
[9]. Kamath, C., (2001), On mining scientific data sets, Data Mining in Scientific and Engineering Applications, Kluwer Academic Publishers, Norwell, MA.
[10]. Piatetsky-Shapiro, G., Frawley, W.J., (Eds.), (1991), Knowledge Discovery, Databases Menlo Park, CA: AAAI.
[11]. Salim, M., Yao, X., (2002), Evolving SQL queries for data mining, Ideal 2002, LNCS 2412, H. Yin et al. (Eds.), pp.62-67.
[12]. Witten, I., Frank, E., (2000), Data Mining, Morgan Kaufmann Publisher.

Author:

Monica Chiş, Ph.D. Student at Faculty of Mathematics and Computer Science, Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania, Faculty of Applied Sciences, Avram-Iancu University, Cluj-Napoca, Romania, mchis@artelecom.net