STUDY ON DISCRETIZATION IN ROUGH SET BASED ON GENETIC ALGORITHM

CAI-YUN CHEN, ZHI-GUO LI, SHENG-YONG QIAO, SHUO-PIN WEN

Center for Combinatorics, LPMC, Nankai University, Tianjin, 300071, P. R. China

E-MAIL: sbickle@eyou.com

Abstract

Discretization of attributes with real values in rough set is an important problem in data mining. It is different from the traditional discretization which has particular characteristic. Nguyen S. H has given a detailed description about discretization in rough set. This paper gives a genetic algorithm aimed at discretization proposed in reference [1]. And at the same time we make an experiment on several datasets from UCI Machine Learning Repository by this method. During the experiment, we constantly optimized genetic algorithm by adopting some optimization strategies. The experiment has proved that using genetic algorithm to solve discretization of rough set is efficient whatever in time complexity or in accuracy.

Keywords

Discretization; Rough Set; Genetic Algorithm

1、 Introduction

The discretization of real value attributes is one of the important problems to be solved in data mining, especially in rough set. We know, when the value set of any attribute in a decision table is continuous value or real number, and then it is likely that there will be few objects that will have the same value of the corresponding attributes. In such a situation the number of equivalence classes based on that attribute will be large and there will be very few elements in each of such equivalence class, which will lead to the generation of a large number of rules in the classification of rough set, thereby making rough set theoretic classifiers inefficient. Discretization is a process of grouping the values of the attributes in intervals in such a way that the knowledge content or the discernibility is not lost. Many discretization approaches have been developed so far. Nguyen S. H had given some detailed description about discretization in rough set in reference [1]. He gave the complexity of discretization problem and proved that it is an NP-hard problem. And at the same time he also proposed a basic heuristic algorithm based on rough set and Boolean reasoning(for convenience of citization, we call this algorithm basic heuristic algorithm), which brought great improvement in dealing with discretization problem in rough set. Aimed at the particularity of discretization problem, this paper gives a genetic algorithm(for short GA) for discretization in rough set. Its main idea is to find possibly minimum number of discrete intervals by constantly adopting some optimization strategies during the experiment. The experiment shows that the GA is more efficient no matter in accuracy or in time complexity than basic heuristic algorithm.

This paper is organized by 5 sections. Description of discretization in rough set is introduced in section 2. In section 3 we construct a genetic algorithm for discretization problem. In section 4, an experiment for discretization is done by two methods including genetic algorithm and basic heuristic algorithm using several datasets from UCI Machine Learning Repository and the decision table in reference [1], we give an analysis about genetic algorithm for discretization. Section 5 gives a summary of this paper

2、 Description of discretization in rough set

A decision table is composed of a 4-tuple as follows: $S = < U, Q \bigcup \{d\}, V, f >$, where

U： a finite set of N objects $\{x_1, x_2, \ldots x_N\}$;

Q：a finite set of n condition attributes $\{q_1, q_2, \ldots, q_n\}$ (a nonempty set), and d is decision attribute；

$\mathrm{V} = \bigcup_{q \in Q} \mathrm{V}_q$, where V_q is a domain of the attribute q.

$f : \mathrm{U} \times \mathrm{Q} \cup \mathrm{d} \to \mathrm{V}$ is the total decision function called information function such that $f(\mathrm{x}, \mathrm{q}) \in \mathrm{V}_\mathrm{q}$ for every $q \in \mathrm{Q} \cup \mathrm{d}$, $\mathrm{x} \in \mathrm{U}$. The decision table can be represented as a finite data table, in which the columns are labeled by attributes, the rows by objects and the entry in column q_j and row x_i has the value $f(x_i, q_j)$. Each row in the table describes the information about some objects in S.

We assume $\mathrm{V}_q = [l_q, r_q) \subset R$, where R is the set of real numbers, and we also assume that S is consistent decision table[1]. The following notion and description about

discretization is referred to reference [2]. First let us give the definition of cut.

Definition 1 Any pair (q,c) , where $q \in \mathrm{Q}$ and $c \in \mathrm{R}$,defines a partition of V_q into left-hand-side and right hand-side interval. The pair (q,c) is called a cut on V_q.

For an attribute $q \in \mathrm{Q}$, $D_q = \{(q,c_1^q),(q,c_2^q),...,(q,c_{k_q}^q)\}$ is composed by all the cuts, where $k_q \in N$, and $l_q = c_0^q < c_1^q < c_2^q < ... < c_{k_q}^q < c_{k_q+1}^q) = r_q$, defines a partition on V_q into subintervals i.e. $\mathrm{V}_q = [c_0^q, c_1^q) \bigcup [c_1^q, c_2^q) \bigcup ... \bigcup [c_{k_q}^q, c_{k_q+1}^q)$. So any set of cuts on condition attributes $D = \bigcup D_a$ transforms the original decision table S into discrete decision table $S^D =< U, Q \bigcup \{d\}, V^D, f^D >$, where $f^D(x,q) = i \Leftrightarrow f(x,q) \in [c_i^q, c_{i+1}^q)$, and $x \in U, i \in \{0,1,...,k_q\}, q \in Q$.

After discretization, the original decision table is replaced with the new one. And different sets of cuts will construct different new decision table. It is obvious that discretization process is associated with loss of information. Usually the task of discretization is to determine a minimal set of cuts from a given decision table and keeping the discernibility. The selected cuts can be evaluated by the following criteria[1,3]:

1. Consistency of D. For any objects $u, v \in U$,they are satisfying if u ,v are discerned by Q, then u, v are discerned by D;
2. Irreducibility. There is no $D' \subset D$,satisfying the consistency;
3. Optimality. For any D' satisfying consistency, it follows card(D) $\leq$ card(D'), then D is optimal cuts.

Nguyen S. H had proved that the optimal discretization problem is an NP-hard problem. As we know, genetic algorithm is a better method for solving this kind of optimization problem. In the next section, we will give a design of genetic algorithm for discretization problem.

3、 Genetic algorithm for discretization problem.

Genetic algorithm(for short GA) is basically search algorithm designed to mimic the process of natural selection and evolution in nature[5]. To begin with, we need to define a few terms. A population consists of a fixed number of members, or chromosomes. Each chromosome is a fixed-length string of genes. The fitness of a chromosome is some measure of how desirable it is to have that chromosome in the population. A generation is a time step in which several events occur. Some of the most "unfit" chromosomes die and are removed from the population. To replace these, some number of crossover operations are applied to the population. A crossover is an operation analogous to mating or gene splicing. It combines part of one chromosome with part of another chromosome to create a new chromosome. Finally, some amount of mutation occurs in the population, in which genes are changed randomly with some (typically low) probability. It is also possible to have elitism in the population in which some of the fittest genes are immune from mutation between generations.

The definition of GA is as follows:

GA can be defined as an 8-tuple: $(C, E, P_0, M, \Phi, \Gamma, \Psi, \mathrm{Y})$ where:

C - code of chromosome;

E -the fitness function of every chromosome;

P_0 -the original population;

M -the scale of population;

Φ -the selection operator;

Γ -the crossover operator;

Ψ -the mutation operator;

Y -the stopping criteria of GA.

By this definition, we define a GA to find the minimal set of cuts for the rough set.

First we construct a new decision table $S^* =< U^*, Q^* \bigcup \{d\}, V^*, f^* >$ from a given decision table $S =< U, Q \bigcup \{d\}, V, f >$ as following before constructing the algorithm:

$U^* = \{(x_i, x_j) \in U \times U \,|\, d(x_i) \neq d(x_j)\}$;

$Q^* = \{c_i^q \,|\, i \in \{0,1,...,k_q\}, q \in Q\}$, where c_i^q is the cut of attribute q;

for $u = (x_i, x_j) \in U^*, c \in Q^*$ we have:

$$f^*(u,c) = \begin{cases} 1 & \min(q(x_i), q(x_j)) < c < \max(q(x_i), q(x_j)) \\ 0 & otherwise \end{cases}$$

The following genetic algorithm is designed by the new decision table $S^* =< U^*, Q^* \bigcup \{d\}, V^*, f^* >$. For convenience, we denote $L = |Q^*|$ being the length of the

chromosome.

For convenience, if $f^*(u,c)=1$ we say that the cut c can discern u or u can be discerned by cut c , otherwise we say that cut c can't discern u or u can not be discerned by c.

1. Encoding the points in the solution space

For the GA to be described in this paper, we use the following representation for a point in the solution space. For this problem, the solution space is the set of all possible combinations of 1, 2,..., L prime implicants (hence the size of the solution space is 2^L-1 , excluding the possibility where no prime implicant is chosen). We use an L-bit string to represent a particular choice of cuts. A value of 1 in the i-th string position (where $i\in\{1,...,L\}$) implies that the i-th cut is chosen to be in the discretization of rough set.

2. Determinating the scale M of the population

Here we let M=50. By experience, we know, M is larger, the result is better, but it will cost more time, and vice verse. The original population denoted by P_0 is made up of M chromosomes which are generated by random. And we denote all the chromosomes by

$$\vec{X}(0)=(X_1(0),X_2(0),\ldots X_M(0))\in H_M^L$$

where H_M^L is syngen space.

3. Designing the crossover operation and the mutation operation

The crossover operation leads to an increased diversity of the population of strings as a new individuals emerge out of this process. The intensity of crossover is characterized in terms of the probability at which the elements of the strings are affected. The higher the probability, the more individuals are affected by the crossover. The best range of this probability P_c is between 0.4 and 0.7, and in this paper we choose P_c=0.7.

The mutation operator is an example of an operator adding an extra diversity of a stochastic nature. In binary strings this mechanism is implemented by flipping the values of some randomly selected bits. Again, the mutation rate P_m is related to the probability at which the individual bits become affected. And in our experiments, we set P_m = 0.02.

Generally, the operation of crossover can be viewed as a special type of recombination operation which involves two parents (strings) and leads to two offspring.

4. Constructing the fitness function of GA

In this paper, our aim is to find the minimal set of cuts which can discern all the pairs in U^* . Since the fitness function must be positive, we choose the fitness function as: $F(X)=e^{-\frac{l}{L}}$ where $l=\sum_{i=1}^{L}x_i$, and x_i = 0 or 1, is the value of every gene of one chromosome.

5. Setting the stopping criteria

The simplest stopping criterion is to stop after a predetermined number of generations (or objective function evaluations). So we stop GA after 1000 generations. And the optimal set of cuts is the one whose value of its fitness function is minimized.

4、 Optimization Strategies

To help GA finding the best solution faster, some optimization strategies have been adopted.

(1) Elitist selection and father-offspring combined selection strategy

To make GA get the best solution faster, we apply the Elitist selection and father-offspring combined selection strategy into our algorithm. The elitist selection and father-offspring combined selection can both help GA to find the best solution [4].The algorithm is shown as follows:

Elitist Selection and Father-Offspring Combined Selection GA

Step 1 (starting) set $t\leftarrow 0$, selecting M chromosomes randomly, and evaluating them, we obtain the original generation.

$$\vec{X}(0)=(X_1(0),X_2(0),\ldots,X_M(0))\in H_M^L$$

Step 2 (Evolution)

2.1 Select M pairs of "parents" chromosomes from current generation $\vec{X}(t)$;

2.2 M "offspring" chromosomes will be generated from these M pairs of chromosomes using the "crossover" operation.

2.3 Next, these M "offspring" chromosomes are subjected to mutation. And M new chromosomes have been generated:

$$\vec{X}'(t+1)=X_1'(t+1),X_2'(t+1),\ldots,X'_M(t+1)$$

2.4 Select M-1 chromosomes $X_1(t+1)$, $X_2(t+1)$,..., $X_{M-1}(t+1)$ from $\vec{X}(t)\bigcup\vec{X}'(t+1)$, and at the same time we choose the optimal chromosome $X^*(t)$, and let $X_M(t+1)=X^*(t)$. So we get the (t+1) generation:

$$\vec{X}(t+1)=X_1(t+1),X_2(t+1),\ldots,X_{M-1}(t+1),X_M(t+1)$$

Step 3 (Stopping criteria)

If the algorithm has reached the stopping criteria, we output $X^*(t+1)$ as the optimal set of cuts for this problem, otherwise we set $t \leftarrow t+1$ and go to step 2.

(2) Restart strategy

In our algorithm, we put the best solution which has been generated before being into the original population and hope to further optimize the solution. This strategy is different from the strategy which just increases the times of generation. The former can further optimize the second best solution according to the diversity of the original population. But all the chromosomes are almost similar in the later period of GA, and it is difficult to further optimize the best solution, so the former strategy is more effective than the later one.

This strategy can also avoid the "precocious" phenomena to some extend.

(3) Penalty strategy

Because the problem of discretization in rough set is a constrained optimization problem, i.e. not every L-bit string (a chromosome) which is created randomly is a set of cuts of the rough set. We call such a chromosome an incomplete solution for our problem. And in our experiments, we use penalty function to eliminate the incomplete solutions. When the chosen chromosome can't discern all the objects of U^*, we resort to a sequential search starting at the first cut and including all the cuts that can discern all the objects which haven't been discerned by any cuts. This process continues until all objects are discerned. And for each of the newly added cut, a cost of α is multiplied to its fitness function. α is set to lower than one in order to penalize the particular candidate solution for leaving out necessary. And in our work, we set $\alpha = 0.8$.

5、 Experiments study

In our experiments, firstly, we compare the result obtained by GA and the basic heuristic algorithm on the decision table in reference [1]. The result of GA is $\{ p_2^a, p_4^a, p_2^b \}$ while the result of the basic heuristic algorithm is $\{ p_2^a, p_3^a, p_4^a, p_2^b \}$ that the different result depends on the different choice, because there exist more than one columns whose number of occurrences of 1's is maximal. So generally speaking the result obtained by GA is better than that of the basic heuristic algorithm. Next we proved that using GA is feasible dealing with the large datasets on several datasets of UCI Machine Learning Repository such as glass, iris, pima and wine (http://www.ics.uci.edu/~mlearn/ MLRepository.html). In the following table(table 1), we give the time complexity in this table.

Table 1 the results of our experiments

dataset name	number of cuts	runtime of GA	$\|U^*\|$	$\|A^*\|$
Glass	14	5 minutes	16976	930
Iris	7	2 minutes	7500	119
Pima	22	73 minutes	134121	1246
Wine	14	4 minutes	15667	992

The most complex part of GA is the part of checking and revising the incomplete solutions. In fact, the complexity of the procedure of checking is $O(|U^*|)$, while to revise the incomplete solution may reach $O(|U^*|^2)$.When $|U^*|$ become larger and larger, the runtime of GA will become longer and longer. In our experiments we notice that when $|U^*|$ is large, GA is slow in the first several generations, but it become faster and faster in the following generation, so the total time is not so long. We have found the reason of this phenomenon that the first generation is created by random, i.e. the "father" chromosomes are not very similar, thus their "offspring" chromosome have a higher probability to because incomplete. But in the late period of GA, all the chromosomes are similar, so the "children" chromosomes are very "like" their "parents", so the probability of being an incomplete solution is lower, and the speed of GA becomes faster in the later period.

From above on, we can draw the conclusion that GA is effective to solve the problem of the discretization in rough set.

6、 Acknowledgments

This work was done under the auspices of the National "973" Project on Mathematical Mechanization and the National Science Foundation of China.

7、 Conclusion

Discretization in rough set is different from the traditional discretizaton. Generally speaking, we should find possibly minimum number of cut, and at the same time it should not weaken the indiscernibility. Aimed at its particularity, we give a genetic algorithm for the discretization. We do the experiments by several datasets from the UCI Machine Learning Repository using GA. The experiment has proved that the GA is efficient no matter in accuracy or in time complexity.

References

[1] Nguyen H S,Skowron A. Quantization of real value attributes. Proceedins of Second Joint Annual Conf. on Information Science, Wrightsville Beach,North Carolina,pp34-37,1995

[2] Jian-Hua Dai,Yuan-Xiang Li. Study on discretization based on rough set theory.Proceedings of the First International Conference on Machine Learning and Cybernetics,Beijng,4-5 November 2002

[3] Holland John H. Adaptation in Natural and Artifitial System.Ann Arbor:The University of Michigan Press,1975

[4] Zong-Ben Xu, Zan-Kan Nie, Wen-Xiu Zhang, Almost Sure Strong Convergence of A Class of Genetic Algorithms With Parent-Offsprings Competition, ACTA MATHEMATICAE APPLICATAE, SINICA (2002),25(1),(pp167-175)

[5] Krzysztof J.Cios, Witold Pedrycz, Roman W. Swiniarski, Data Mining Methods for Knowledge Discovery, Kluwer Academic Publishers, 1998