基因遺傳演算式之知識整合策略
Genetic Knowledge-Integration Strategies

王景弘
Wang Ching-Hung
中華電信研究所資訊室
Chunghwa Telecomm Lab.
ching @ms.tl.gov.tw

洪宗貝
Hong Tzung-Pei
私立義守大學資訊管理系
Department of Inform. Management
I-Shou University
tphong@csa500.isu.edu.tw

曾憲雄
Tseng Shian-Shyong
國立交通大學資訊科學
Inst. of Comp. and Inform. Sci.
Chiao-Tung University
sstseng@cis.nctu.edu.tw

摘 要

在本篇論文中，我們基於達爾文的演化論，提出一個基因遺傳演算式的知識整合策略，並應用此策略於腦瘤疾病診斷方面，實驗的結果證實我們的方法非常有成效。

關鍵字：基因遺傳演算法，知識擷取，知識整合，機器學習

Abstract

In this paper, we propose a genetic knowledge-integration approach based on ***Darwin's theory of natural selection*** *to integrate multiple rule sets. The proposed approach consists of two phases : knowledge encoding and knowledge integrating. In the encoding phase, each rule set is encoded as a bit string. The combined bit strings thus form an initial knowledge population, which is then ready for integrating. In the integration phase, a genetic algorithm generates an optimal or nearly optimal rule set from these initial rule-set strings. Finally, experimental results from diagnosis of brain tumors show that the rule set derived by the proposed approach is much more accurate than each initial rule set.*

Keywords: genetic algorithm, knowledge acquisition, knowledge integration, machine learning.

1. Introduction

Recently, a great deal of research [1][3][4][10] has been devoted to the study of the reuse and integration of various existing knowledge sources to reduce knowledge-acquisition bottleneck. Especially for complex application problems, related domain knowledge is usually distributed among multiple sources. No single source may have complete domain knowledge. The use of knowledge integrated from multiple knowledge sources is thus especially important to ensure comprehensive coverage.

In this paper, we propose a genetic knowledge-integration approach to integrate multiple rule sets into one integrated knowledge base. It first encodes each rule set as a bit string, and then chooses elements from each rule set for "mating", thus gradually creating better *offspring* rule sets. The *offspring* rule sets then undergo recursive "evolution" until a really optimal or nearly optimal rule set is produced.

Finally, experimental results on the brain tumor diagnosis show that the rule set derived by the proposed integration approach is much more accurate than each initial rule set. They also show that the proposed integration approach can effectively combine multiple rule sets into a knowledge base.

The remainder of this paper is organized as follows. A Genetic knowledge-integration approach is proposed in Section 2. Experiments on brain tumor diagnosis are made in Section 3. Conclusions are given in Section 4.

2. Genetic Knowledge Integration

The objective of knowledge integration is to integrate a set of knowledge sources into a knowledge base that satisfies the following conditions:

1. *Completeness*: all the objects in the domain space can be covered by at least one rule.
2. *Correctness*: the rule set can make correct classification.
3. *Consistency*: no two rules contradict each other.
4. *Conciseness*: the number of rules is minimal.

These four conditions usually cannot be satisfied at the same time, and trade-offs exist among them. In this sense, knowledge integration can be thought of as a multi-objective optimization problem. The genetic algorithm (GA) is an adaptive search technique, which is very effective in finding optimal or nearly optimal

solutions to a variety of problems. Therefore, we propose a genetic knowledge-integration approach to deal with the knowledge-integration problem.

The proposed approach uses the genetic algorithm to maintain a population of possible rule sets, and automatically searches for the best integrated rule set. The proposed genetic knowledge integration consists of two phases: ***encoding*** and ***integration***. The encoding phase transforms each rule set into a bit-string structure. The integration phase chooses bit-string rule sets for "mating", gradually creating good offspring rule sets. The offspring rule sets then undergo recursive "evolution" until an optimal or nearly optimal rule set is found. The knowledge-integration procedure is illustrated in Fig. 1; RS_1, RS_2, ... , RS_m are the integrating rule sets obtained from different knowledge sources.

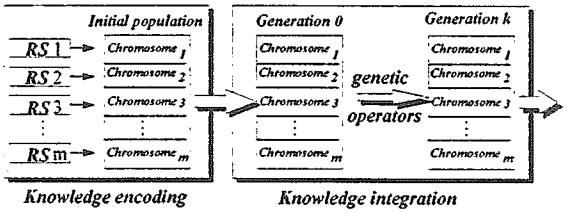

Fig. 1: The genetic knowledge-integration procedure

2.1. Knowledge Encoding

Since rule sets derived from different knowledge sources generally differ in syntax and size, designing an appropriate data structure to accommodate these rule sets is crucial. Several strategies have been proposed for representing rule-set knowledge structures for conceptual learning [8][9]. For example, the **Michigan approach** [2][11] encodes individual rules into fixed-length bit strings, with each individual in the population representing a rule. Another, the **Pittsburgh approach** [7] encodes rule sets into variable-length bit strings, with each individual in the population representing a rule set. Since multiple rule sets must be combined, the rule sets are derived from different sources, representation of variable-length rule sets is preferred in this research to preserve the syntactic and semantic constraints of *variable-length* rule sets. We thus encode knowledge using the *Pittsburgh-style approach*. The rule sets from different sources must, however, be translated into a uniform syntactical representation before being encoded. The steps for translation of rule sets are described below.

1. Collect the features and possible values occurring in the condition parts of rule sets. All features gathered together comprise the global feature set.
2. Collect classes occurring in the conclusion parts of rule sets. All classes gathered together comprise the global class set.
3. Translate each rule into an intermediary representation that retains its essential syntax and semantics. If some features in the feature set are not used by the rule, *dummy* tests are inserted into the condition part of the rule. Each intermediary representation is then composed of *N feature tests* and one ***class pattern***, where *N* is the number of global features collected. If the *feature tests* or *class patterns* are numerical, they are first discretizied into a number of possible regions.
4. Concatenate all intermediary rules to form an intermediary rule set.

An example is given below to demonstrate the translation process of forming intermediary representations.

Example 1: The brain tumors diagnosis is used to demonstrate the translation process of forming intermediary representations. There are two classes of brain tumors to be distinguished: *Adenoma* and *Meningioma*. Each rule is described by three features: *Location*, *Calcification*, and *Edema*. Each feature has the possible values shown below.

Location = {brain surface, sellar, brain stem}

Calcification = {no, marginal, vascular-like, lumpy}

Edema = {no, < 2 cm, < 0.5 hemisphere}.

Assume a rule set RS_q obtained from a knowledge source has the following two rules:

r_{q1} : If (*Location* = *sellar*) and (*Calcification* = *no*) then *Class* is *Adenoma*;

r_{q2} : If (*Location* = *brain surface*) and (*Edema* < *2 cm*) then *Class* is *Meningioma*.

This intermediary representations of r_{q1} and r_{q2} would then be constructed as follows:

r'_{q1} : If (*Location* = *sellar*) and (*Calcification* = *no*) and ***(Edema = no or Edema < 2 cm or Edema < 0.5 hemisphere)*** then *Class* is *Adenoma*;

r'_{q2} : If (*Location* = *brain surface*) and ***(Calcification = no or Calcification = marginal or Calcification = vascular like or Calcification = lumpy)*** and (*Edema* < *2 cm*) then *Class* is *Meningioma*.

The tests with underlines are *dummy tests*. Also, r_{q1} and r_{q2} are logically equivalent to r'_{q1} and r'_{q2}. After translation, each intermediary rule representation then consists of three feature tests and one class pattern. We concatenate all intermediary rules to form an intermediary rule set RS'_q.

After translation, each intermediary rule set is then ready to encode as a bit string. Using the intermediary form, we first encode each feature test into a fixed-length binary, with length equal to the number of possible feature test values. Thus, each bit represents a possible value. For example, the set of legal values for feature *Location* is {*brain surface, sellar, brain stem*}, three bits are then used to represent this feature. The bit string *101* would represent the test for *Location* being "*brain surface*" or "*brain stem*". Similarly, the class pattern is encoded into a fixed-length binary string with each bit representing a possible class. Each rule in the intermediary representation is then encoded as a fixed-length bit string. Since different rule sets might have different numbers of rules, the lengths of the intermediary rule sets might be different. An example that demonstrates the encoding process of intermediary rule set is shown below.

Example 2: Continuing from *Example* 1, assume the intermediary rule-set RS_q' is to be encoded as a bit string. The rule r_{q1}' in *Example* 1 is first encoded as follows.

	Location	*Calcification*	*Edema*	*Class*
r_{q1}'	010	1000	111	10

Since feature *Location* in r_{q1}' has only one test value, *Sellar*, the test for *Location* is then encoded as "010". Similarly, the test for *Calcification* is encoded as "1000". But, *Edema* has three possible test values, "*no*", "*< 2cm*", and "*<0.5 hemisphere*", the test for *Edema* is then encoded as "111". Corresponding, the class points to *Adenoma*, the *class pattern* is encoded as "10". As a result, each intermediary rule in RS_q' is encoded into a substring as blow.

	Location	*Calcification*	*Edema*	*Class*
r_{q1}'	010	1000	111	10
r_{q2}'	100	1111	010	01

Finally, the intermediary rule set RS_q' is encoded into a chromosome as in Fig. 2 shown.

$$\overbrace{\underbrace{0101000}_{r_{q1}'}\underbrace{1110}_{r_{q2}'}\underbrace{100111101001}}^{RS_q'}$$

Fig. 2 : Bit-representations of RS_q' for *Example* 2

2.2. Knowledge Integration

Knowledge integration uses the genetic algorithm for integration and optimization by generating a population of bit strings for each integrating rule set, and evaluating the bit strings with an evaluation function and a data set. Rule-set performance is then fed back to the genetic algorithm to control how the solution space is searched to promote rule set quality.

During integration, the genetic algorithm requires a population of feasible solutions to be initialized and updated during the evolution process. In our approach, the initial set of bit strings for rule sets comes from the multiple knowledge sources. Each rule set represents one individual in the initial population. In order to develop a "good" knowledge base from an initial population of rule sets, the genetic algorithm selects *parent* rule sets with high fitness values for mating. An evaluation function and a set of data including documentary evidence, instances or historical records, are then used to qualify the derived rule set. Two important factors are used in evaluating derived rule sets, the accuracy and the complexity of the resulting knowledge structure. Accuracy of a rule set *RS* is evaluated using data as follows:

$$Accuracy(RS) = \frac{\textit{the total number of data correctly matched by RS}}{\textit{the total number of data}}$$

The more data used, the more objective and accurate the evaluation is. The complexity of the resulting rule set (*RS*) is the ratio of rule increase, defined as follows:

$$Complexity(RS) = \frac{\textit{Number of rules in the integrated rule set RS}}{[\sum_{i=1}^{m}(\textit{Number of rules in initial } RS_i)]/m},$$

where RS_i is *i*-th the initial rule sets, and *m* is the number of initial rule sets. Accuracy and complexity are combined to represent the fitness value of the rule set. The evaluation function is defined as follows:

$$fitness(RS) = \frac{[Accuracy(RS)]}{[Complexity(RS)]^{\sigma}},$$

where σ is a control parameter, representing a trade-off between accuracy and complexity. The fitness function can also reduce the impact of noisy information that causes rule set overfitting (excessive complexity).

2.3. Genetic Operators

Two genetic operators are applied to the rule set population for knowledge integration. They are

dynamic crossover and *mutation*. The crossover operator used here selects crossover points differently from the way used in the simple genetic algorithm. The crossover operator in the simple genetic algorithm chooses the same points for both parent chromosomes, but, the crossover operator here need not use the same point positions for both parent chromosomes. The crossover points may occur within rule strings or at rule boundaries. The only requirement for crossover points is that they "match up semantically".

The crossover operator thus takes two parent rule-sets and swaps parts of their genes to produce offspring rule-sets. If the genes swapped are desirable, then the offspring rule-sets will inherit these advantages from their parents and survive after this generation. Thus, the resulting rule-sets will be closer to the one really desired from generation to generation. The detailed procedure is shown below.

1. Randomly select a crossover point in one of the parents.
2. If the point occurs at some rule boundary, then the crossover point of the other parent must also occur at the rule boundary. Otherwise, the point may be within the rule string p bits to left of a rule boundary. The crossover point for the other parent must also occurs within the rule string and p bits to left of some rule boundary.
3. Cross the genes of the parents according to the crossover points.
4. Generate new offsprings.

An example of crossover operation is shown below.

Example 2: Assume that parent rule sets RS_1 and RS_2, respectively, contain n and m rules for classifying test instances with four features (F_1, F_2, F_3, and F_4). Feature F_1 has three possible values, features F_2, F_3, and F_4 all have two possible values. Three classes are to be determined. Assume that RS_1 and RS_2 are encoded as follows:

RS_1: $\overbrace{100110110001}^{r_{11}}\cdots\overbrace{010010101010}^{r_{1i}}\cdots\overbrace{001010101100}^{r_{1n}}$
($F_1\ F_2F_3F_4$ Class)

RS_2: $\overbrace{010011001100}^{r_{21}}\cdots\overbrace{110111010101}^{r_{2j}}\cdots\overbrace{100011001101}^{r_{2m}}$

As mentioned above, the crossover points on both parents must "match up semantically". If crossover point cp_1 is the seventh bit to the left of r_{1i} in RS_1 (denoted as $cp_1 = (1i, 7)$), then crossover point cp_2 for RS_2 must be the seventh bit to the left of a certain rule r_{2j} (denoted as $cp_2 = (2j, 7)$). Thus, the crossover operator generates two offspring rule sets, O_1 and O_2 as shown in Fig. 3.

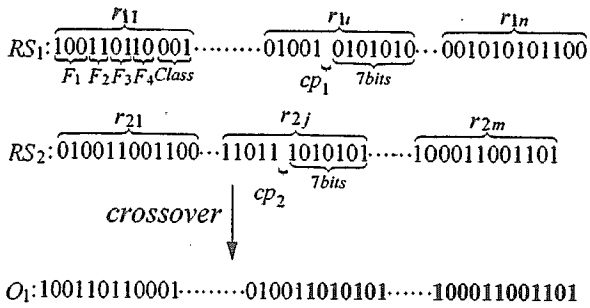

Fig. 3: An example of dynamic crossover

The mutation operator is the same as the standard one in the simple genetic algorithm, and randomly changes some elements in a selected rule set to help the integration process escape from local-optimum "traps".

3. Experimental Results

Brain tumor diagnosis [15] was used as the problem domain to test the performance of the proposed knowledge-integration approach. Due to their inherent complexity, the diagnosis of brain tumors is currently still very difficult for doctors. Thus, developing a successful brain tumor diagnostic system seems to be very important. The test instances used in these experiments to evaluate rule-set fitness values were obtained from Veterans General Hospital (VGH) in Taipei, Taiwan. One of six possible classes of brain tumors including *Pituitary Adenoma*, *Meningioma*, *Medulloblastoma*, *Glioblastoma*, *Astrocytoma*, and *Anaplastic Protoplasmic Astrocytoma* (frequently found in Taiwan), must be identified. The numbers of possible feature values and class patterns are shown in Table 1.

Table 1:Numbers of feature values and class patterns

Feature	Number	Feature	Number
Sex	2	Shape Edema	5
Location	44	Calcification	4
Precontrast	6	Enhancement Degree	4
Edema	4	Enhancement Appearance	9
Bone Change	6	General Appearance	9
Mass Effect	3	Hydrocephalus	3
Number of classes : 6			

To evaluate the performance of our knowledge-integration approach, ten initial rule sets were obtained from different groups of experts at VGH or derived from via machine learning methods [5][6][13][14][15]. Each rule, consisting of twelve feature tests and a class pattern, was encoded into a bit string 103 bits long. The accuracy of the ten initial rule sets was measured

using the test instances. The results are shown in Table 2.

Table 2: The accuracy of the ten initial rule sets

Rule Sets	Accuracy	No. of rules	Rule Sets	Accuracy	No. of rules
Rule Set 1	60.03%	52	Rule Set 6	77.89%	56
Rule Set 2	79.81%	56	Rule Set 7	68.53%	52
Rule Set 3	73.24%	56	Rule Set 8	72.83%	53
Rule Set 4	64.74%	53	Rule Set 9	76.24%	56
Rule Set 5	58.67%	52	Rule Set 10	70.19%	53

The automatic knowledge-integration algorithm was implemented in C language on a SUN SPARC/2 workstation. The experiments were made to evaluate the effectiveness of the proposed method, averaged over 50 runs. The proposed knowledge-integration algorithm obtained an accuracy rate of 85.83% after 2000 execution generations. The size and the complexity of the resulting knowledge base were respectively, 168 and 3.1168. Note that the accuracy obtained is higher than any initial rule set in Table 2.

4. Conclusion

We have shown how knowledge-integration can be effectively represented and addressed by a genetic algorithm. Experimental results showed that our genetic knowledge-integration approach is valuable for combining multiple rule sets. Our approach differs from some notable approaches [4][10][12] mainly in that it requires no human experts' intervention during integration. Our approach is thus dependent on computer execution speeds, not on human experts. This saves much time since experts may be geographically dispersed, and their deliberations may be very time-consuming. Our approach is also scalable and can be used effectively when the number of rule sets to be integrated is large, and integrating large numbers of rule sets may increase the validity of the resulting knowledge base. Our method is also objective since human experts are not involved in the integration process.

References

[1] C. Baral, S. Kraus, and J. Minker, "Combining multiple knowledge bases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 3, no. 2, pp. 208-220, 1991.

[2] L. B. Booker, *Intelligent behavior as an adaptation to the task environment*, Doctoral Dissertation, University of Michigan, 1982.

[3] J. H. Boose and J. M. Bardshaw, "Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems," *International Journal of Man-Machine Studies*, vol. 26, pp. 3-28, 1987.

[4] J. H. Boose, "Rapid acquisition and combination of knowledge from multiple experts in the same domain," *Future Computing Systems*, vol. 1, pp. 191-216, 1986.

[5] J. Cendrowska, "PRISM: An algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, vol. 27, pp. 349-370, 1987.

[6] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, pp. 261-283, 1989.

[7] K. A. DeJohn, "Learning with genetic algorithm: an overview," *Machine Learning*, vol. 3, pp. 121-138, 1988.

[8] K. A. DeJong, W. M. Spears, and D. F. Gordon, "Using genetic algorithms for concept learning," *Machine Learning*, vol. 13, pp. 161-188, 1993.

[9] K. A. DeJong and W. M. Spears, "Learning concept classification rules using genetic algorithms," *IJCAI 91*, pp. 651-656, 1991.

[10] B. R. Gaines, "Integration issues in knowledge supports systems," *International Journal of Man-Machine Studies*, vol. 26, pp. 495-515, 1989.

[11] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithm," *Machine Learning: An Artificial Intelligence Approach*, Los Altos, CA, Morgan Kaufmann Publishers, Los Altos, CA, 1983.

[12] G. J. Hwang, "Knowledge elicitation and integration from multiple experts," *Journal of Information Science and Engineering*, vol. 10, no. 1, pp. 99-109, March 1994.

[13] J. Quinlan, "Induction of decision tree," *Machine Learning*, vol. 1, pp. 81-106, 1986.

[14] C. Y. Suen, Y. S. Huang and A. Bloch, "Multiple expert systems and multi-expert systems," *The Second World Congress on Expert Systems*, pp. 207-212, 1994.

[15] C. H. Wang and S. S. Tseng, "A brain tumor diagnostic system with automatic learning abilities," *The Third IEEE Symposium on Computer-Based Medical Systems Conference*, pp. 313-320, 1990.