Rule Induction by EDA with Instance-Subpopulations

Hisashi Handa
Graduate School of Natural Science and Technology
Tsushima-Naka 3-1-1, Okayama 700-8530, JAPAN
email:handa@sdc.it.okayama-u.ac.jp

***Abstract*—In this paper, a new rule induction method by using EDA with instance-subpopulations is proposed. The proposed method introduces a notion of instance-subpopulation, where a set of individuals matching a training instance. Then, EDA procedure is separately carried out for each instance-subpopulation. Individuals generated by each EDA procedure are merged to constitute the population at the next generation. We examined the proposed method on Breast-cancer in Wisconsin and Chess End-Game. The comparisons with other algorithms show the effectiveness of the proposed method.**

I. Introduction

In recent years, as available computational resources are growing, Evolutionary Computation has attracted much attention in data mining because of its global search ability. However, it sometime tends to find out only generalized rules: For instance, the upper figure in Fig. 1 explains that certain situations could be occurred in the rule acquisition by using conventional Evolutionary Computation. In the figure, training instances are indicated to by circles (for positive instances) and crosses (for negative instances). This figure is simplified one so that problem space and rules are represented by a plane and rectangles, respectively. The generalized rules mentioned the above mean (a) and (b) in the figure. The reason why such situation is occurred is that by using single population, Evolutionary Computation tends to converge better rules, i.e., accurate rules which can correctly explain a large number of training instances. Rule (c) is accurate but can explain only a few training instances.

In order to avoid such situations, the diversification of solutions have been devised. One of such diversification mechanisms which is not restricted to rule inductions are fitness sharing and crowding [1]. By using either of methods, population will be diversified. However, they often weaken the convergence ability of Evolutionary Computation. Another approach to acquire diverged solutions is use of Evolutionary Multi-objective Optimization [2]. Evolutionary Multi-objective Optimization can separately cope with the accuracy and the complexity as objective functions. In the case of the upper figure in Fig. 1, rules (b) and (c) are definitely accurate but rule (b) is simpler than rule (c) so that EMO might not choose (c). Although the rule selection of such rule involves difficult problems from the viewpoint of the generalization property, the proposed method tries to survive such rules.

XCS by Wilson is a promising classifier systems [3][4]. Main features of XCS are 1) accuracy based fitness calculation,

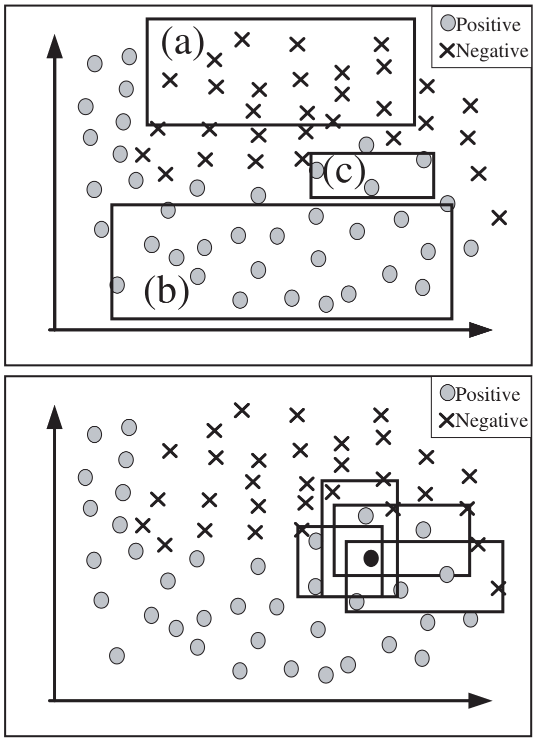

Fig. 1. Example of Acquired Rules by single population (UPPER); by proposed method (LOWER)

2) genetic operation against action sets, and 3) Q-Learning-like Reinforcement Learning. Original XCS is online algorithms while it is often used for Data Mining Problems [3][5]. Recently, XCS/BOA is proposed by Butz and Pelikan [6]. The main difference between XCS/BOA and the proposed method is that probabilistic models in the proposed method is prepared for all the training instance, i.e., action sets in XCS.

In this paper, rule induction method by EDA is proposed. At every generation, subpopulations corresponding to each of training instances, called instance-subpopulation (cf. the lower figure in Fig. 1), are constituted. Then, EDA procedure is carried out in each instance-subpopulation: Better individuals are chosen from each instance-subpopulation. The probabilistic models are estimated from the selected individuals. New individuals are sampled from the probabilistic model. All the individuals sampled by each instance-subpopulation are

```
Procedure Estimation of Distribution Algorithm
begin
  initialize $D_0$
  evaluate $D_0$
  until Stopping criterion is reached
    $D^s_l \leftarrow$ Select $N$ individuals from $D_{l-1}$
    $p_l(\mathbf{x}) \leftarrow$ Estimate the probabilistic model from $D^s_l$
    $D_l \leftarrow$ Sampling $M$ individuals from $p_l(\mathbf{x})$
    evaluate $D_l$
  end
end
```

Fig. 2. Pseudo-code of Estimation of Distribution Algorithms

merged in order to constitute the next population.

II. Estimation of Distribution Algorithms

A. General Framework of EDAs

Estimation of Distribution Algorithms are a class of evolutionary algorithms which adopt probabilistic models to reproduce individuals in the next generation, instead of conventional crossover and mutation operations. The probabilistic model is represented by conditional probability distributions for each variable. This probabilistic model is estimated from the genetic information of selected individuals in the current generation. Hence, the pseudo-code of EDAs can be written as Fig. 2, where D_l, D^s_{l-1}, and $p_l(\mathbf{x})$ indicate the set of individuals at l^{th} generation, the set of selected individuals at $l-1^{\text{th}}$ generation, and estimated probabilistic model at l^{th} generation, respectively [7] As described in this figure, the main calculation procedure of the EDAs is that (1) firstly, the N selected individuals are selected from the population in the previous generation. (2) Secondly, the probabilistic model is estimated from the genetic information of the selected individuals. (3) A new population whose size is M is then sampled by using the estimated probabilistic model. (4) Finally, the new population is evaluated. (5) Steps (1)-(4) are iterated until stopping criterion is reached.

B. MIMIC

In the proposed method, a large number of calls of the EDA procedure is required. Hence, MIMIC (Mutual Information Maximizing Input Clustering) proposed by De Bonet *et al.* is adopted as an EDA algorithm for the proposed method. MIMIC is a kind of EDAs whose probabilistic model is constructed with bivariate dependency such as COMIT (Combining Optimizers with Mutual Information Trees) [8][9] Whilst the COMIT generates a tree as dependency graph, the probabilistic model of the MIMIC is represented by a chain based upon a permutation π.

$$p_l(\mathbf{x}) = \prod_{j=1}^{n-1} p_l(x_{i_{n-j}}|x_{i_{n-j+1}}) \cdot p_l(x_{i_n}),$$

where the permutation $\pi = (i_1, i_2, \ldots, i_n)$ indicates a sequence of variable indices. This permutation is obtained in every generation. In Fig. 3, the permutation π is set to be $(i_1, i_2, \ldots, i_5) = (5, 2, 4, 1, 3)$ for instance. Furthermore, note that the conditional probability $p_l(x_{i_{n-j}}|x_{i_{n-j+1}})$ is an abbreviated form of $p_l(X_{i_{n-j}} = x_{i_{n-j}}|X_{i_{n-j+1}} = x_{i_{n-j+1}})$.

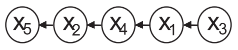

Fig. 3. Probabilistic models for MIMIC

```
Procedure EDA with Instance-Subpopulations
begin
  $D_0$, $D^i_0 \leftarrow$ Initialization $(i = 1 \ldots S)$
  until Stopping criterion is reached
    foreach training instances $i = 1 \ldots S$
      $D^{i,s}_l \leftarrow$ Select $N$ individuals from $D^i_{l-1}$
      $p^i_l(\mathbf{x}) \leftarrow$ Estimate the probabilistic model from $D^{i,s}_l$
    end
    $D_l, D^i_l \leftarrow$ Sampling from $p^i_l(\mathbf{x})$   $(i = 1 \ldots S)$
  end
end
```

Fig. 5. A pseudo-code of the proposed method

III. EDA with Instance-Subpopulations

A. Overview

Fig. 4 depicts a diagram of the proposed method. The main difference between conventional Evolutionary Computation for classification problems and the proposed method is instance-subpopulations, which consist of individuals matching the same training instance. That is, evolutionary search in the proposed method is carried out for each instance-subpopulations. Let S be the number of training instances. Subpopulation 1 in the figure denotes the instance-subpopulation matching the first training instance. After constituting instance-subpopulations, EDA procedure is carried for each of them as follows: Individuals are chosen from each instance-subpopulation. Then, probabilistic models are estimated. New individuals are sampled by using the probabilistic model. Finally, the whole population at the next generation is constituted by merging all the sampled individuals.

B. Representation and Fitness Calculation

The representation of individuals, i.e., rules, is the same as Classifier Systems: Each gene in the antecedent part of individuals is defined by values at corresponding attribute in training data. In addition, some genes are set to be “Don’t Care Symbol,” indicating that any values at corresponding attribute are matched to the rule. The decedent part of individuals represents a class to be classified. The following individuals can be listed as examples for 5 attributes with 5 values and 2

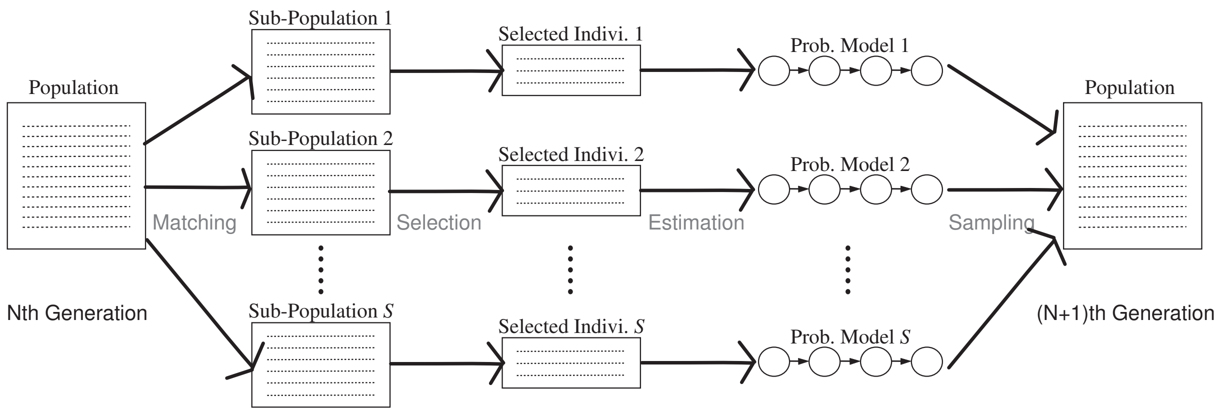

Fig. 4. A diagram of the proposed method

classes problems:

```
0  3  #  2  1   :  0,
#   2  4  #  #   :  1,
#  #   3  #  2   :  1,
```

where the first five digits and the last digit denote the antecedent part and the decedent part of individuals, respectively, and "#" denotes "Don't care symbol." "Completely match" of individuals for a training instance denotes that both of the antecedent and decedent parts of corresponding individual is the same as training instance except for "Don't Care Symbol."

Fitness function $F(x)$ is adopted simple one in this paper:

$$F(x) = n_c/n_m + C_r \cdot r_d,$$

where n_c and n_m indicate the number of correct classification and the number of matched training instances[1], respectively. C_r and r_d denote coefficient and the proportion of "Don't care symbol" in the individual to be evaluated, respectively. The coefficient C_r is set to be quite small value such as 0.001. Hence, this fitness function is designed to find out accurate rules first, then, to find out much general rules among rules with similar accuracy.

Instead of using the proportion of "Don't care symbol," as preliminary experiments, other types of fitness functions are examined, e.g., fitness function in [10] and its mutants, which take account into two kinds of errors, i.e., incorrect classification for matched training instances and the number of unmatched positive training instances. Although such fitness functions are similar to F-value and are familiar with conventional Machine Learning approaches, they did not work well in the proposed method. The reason of this is that these fitness functions are designed to search for rules which can explain a large amount of data. As consequence of this, a large number of instance-subpopulations in the proposed method converge to the same rules. In other words, simple fitness function in the this paper might help to maintain the diversity of generated rules in such subpopulation approach.

[1]Here, "match" means a corresponding rule is matched only in its antecedent part

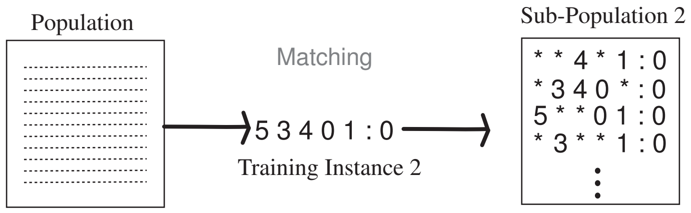

Fig. 6. An example of constitution of instance-subpopulation which is matched to a training instance `5 3 4 0 1 : 0`

C. Constitution of Instance-Subpopulation

As depicted in Fig. 4, instance-subpopulations are constituted as a set of individuals which match to each of training instances. Fig. 6 delineates constitutions of instance-subpopulation which is matched to a training instance `5 3 4 0 1 : 0` for example. Note that matching to constitute instance-subpopulations for a certain training instance means not only the match of the antecedent part but also the match of the decedent part of individuals to the instance.

As depicted in this figure, generated instance-subpopulation is composed of individuals such that each of genes in the antecedent part is the same as the value at corresponding attribute of the training instance or "Don't Care Symbol." Therefore, EDAs should find out the best allocation of "Don't Care Symbols" for each subpopulation. This property is good for EDAs. The reason of this is described as follows: EDAs use probability models to estimate the distribution of effective genetic patterns in selected individuals. That is, if the number of alleles at each locus is large, a large number of selected individuals are needed to estimate probabilistic models precisely. Therefore, in binary-encoded problems, EDAs work very well. In each instance-subpopulation, even if training instances is composed of various kinds of values at each attribute, it can be regard as binary-encoded problems in each subpopulation, i.e., whether "Don't Care Symbol" should be assigned or not.

```
Procedure Sampling ($p_l^i(\mathbf{x})$ $(i = 1 \ldots S)$ is given)
begin
  foreach training instances $i = 1 \ldots S$
    for $N_s$ times
      $D_l \leftarrow$ Add a new individual $k$ sampled by $p_l^i(\mathbf{x})$
      $D_l^i \leftarrow$ Call Matching-Evaluation for individual $k$
    end
  end
  repeat
    $i$ is randomly chosen.
    $D_l \leftarrow$ Add a new individual $k$ sampled by $p_l^i(\mathbf{x})$
    $D_l^i \leftarrow$ Call Matching-Evaluation for individual $k$
  until the size of all the subpopulation exceeds $N_{SPS}$
end

Procedure Matching-Evaluation (individual $k$)
  foreach training instances $j = 1 \ldots S$
    if  The antecedent part of $k$ matches $j$
      $Nm++$
      if  The decedent part of $k$ matches $j$
        $Nc++$
        $D_l^j \leftarrow$ reference of $k$
      endif
    endif
  end
  calculate fitness of $k$ by using $n_c$ and $n_m$
end
```

Fig. 7. Sampling, Matching, and Evaluation for generating a new population

D. Sampling Method

Technically speaking, after sampling a new individual, matching and evaluation of individuals are carried out in procedure Matching-Evaluation in Fig. 7. That is, this procedure is called a number of times in a generation in procedure Sample in the same figure. Firstly, an individual is sampled by using a conventional way in EDA with directed acyclic probabilistic graphical models, i.e., Probabilistic Logic Sampling method [7]. Sampled individual is stored in a new population. Secondly, the sampled individual is examined if its antecedent part matches to each of training instances. If the antecedent part is matched, the decedent part is also examined: If matched, the reference (or pointer in C Language) is added to the reference list of the instance-subpopulation of the corresponding training instance and n_c and n_m in section III-B are incremented. Otherwise, i.e., if the antecedent part is matched while the decedent part is not matched, only N_m is incremented. After the examinations to all the training data, we can calculate the fitness of the sampled individual.

The procedure sampling is firstly sampling N_s individuals for each probabilistic model $p_l^i(\mathbf{x})$ $(i = 1 \ldots S)$. After each sample, as mentioned in the above, the procedure Matching-Evaluation is called. Moreover, 1) the procedure sample randomly chooses training instance i. 2) For selected instance i, if the size of corresponding subpopulation is less than N_{SPS}, then a new individual is sampled by using $p_l^i(\mathbf{x})$, and the the procedure Matching-Evaluation is called. 3) go back to 1) until the size of all the subpopulation exceeds N_{SPS}. For initialization, $p_0^i(x_j)$ is set to be 0.5 $(i = 1 \ldots S, j = 1 \ldots n)$.

E. Generating final individuals

After evolution, there are a large number of individuals in the population. Therefore, final individuals are chosen from the population: First of all, instance-subpopulations are constituted by using the same method as during evolution. Secondly, individual selection is carried out for each subpopulation as follows.

1) Find out the individual which can match training instances with the greatest number.
2) If no individual is found, removed training instances in 3) are recovered and selection procedure is moved to the next subpopulation.
3) Store the individual into a set of final individual candidates and temporally remove the training instances matched by the individual.
4) Go back to 1)

Finally, duplicative individuals in the set of final individual candidates are aggregated into a single individual.

IV. Experimental Results

A. Experimentation Settings

In this paper, two datasets from UCI Machine Learning Repository, i.e., breast-cancer in Wisconsin (`bcw`) and Chess End-Game – King+Rook versus King+Pawn on a7 (`kr-vs-kp`), are used as benchmark problems [11][12]. `bcw` is composed of 699 training instances with 9 features. `kr-vs-kp` has 3196 training instances with 36 features. 10-fold cross validation is used to evaluate the proposed method. In test phase, we adopt majority voting to classify test instances, if there are a number of matched rules with different decedent parts.

Parameters are described as follows: The number of instance-subpopulations is the same as the number of training instances to be learnt. Duplicated training instances are deleted in advance. The number N_s of individuals to be sampled by using the probabilistic model in each subpopulation is set to be one of 0, 50, and 100. The number of selected individuals and the least number N_{SPS} of individuals in each subpopulation are set to be 200 and 400, respectively. Truncation Selection is used as a selection method in this paper. The number of generations is set to be 5. It might be seemed that few number of generations. However, it requires 5 $\times$ (the number of training instance to be learnt) EDA procedures for a single run.

B. Results

Table I summarizes experimental results with various [2] for `bcw` and `kr-vs-kp`. “perform.,” “pop-s,” “subpop-s,” and

[2] N_s means the number of individuals to be sampled by using the probabilistic model in each instance-subpopulation

TABLE I
CROSS-VALIDATION PERFORMANCE WITH VARIOUS N_s ON bcw (UPPER) AND ON kr-vs-kp (LOWER)

N_s	perform.	pop-s	subpop-s	final-s
0	96.0 ± 2.5	9296.1	837.1	84.8
50	96.8 ± 2.2	27423.6	807.0	209.8
100	96.7 ± 2.2	46066	1043.3	227.9
200	96.8 ± 2.1	82044.6	1804.1	241.8

N_s	perform.	pop-s	subpop-s	final-s
0	97.1 ± 10.7	20333.3	866.2	95.7
50	98.3 ± 0.6	155627.0	986.7	344.7
100	98.1 ± 0.8	291603.8	1773.9	443.1

TABLE II
COMPARISON WITH CONVENTIONAL METHODS ON bcw AND kr-vs-kp

Algorithm	perform.(bcw)	perform.(kr-vs-kp)
Proposed Method	96.8 ± 2.2	98.3 ± 0.6
XCSTS	**95.9** ± 2.3	*98.9* ± 0.6
Majority	**70.3** ± 1.2	**52.2** ± 0.1
Main Ind.	**91.9** ± 2.9	**67.1** ± 1.8
C4.5	**94.5** ± 2.6	*99.4* ± 0.4
Naive Bayes	**96.0** ± 2.1	**87.8** ± 1.9
PART	**94.7** ± 2.4	*99.1* ± 0.6
Inst.b.1	**95.6** ± 2.1	**90.5** ± 1.6
Inst.b.3	96.6 ± 2.0	**96.5** ± 1.1
SMO(poly.1)	96.7 ± 1.9	**95.8** ± 1.2
SMO(poly.3)	**95.9** ± 2.1	*99.6* ± 0.4
SMO(radial)	**96.0** ± 2.2	**91.4** ± 1.6

"final-s" in the tables indicate the result of 10-folds cross validation, the population size at the final generation, the average size of the instance-subpopulations, and the number of rules acquired by the proposed method, respectively. As described in section III-E, final rule set is chosen from the individuals at the final generation. Although instance-subpopulations in the proposed method are constituted for all the training instances, the number of final rules is smaller than the number of training instances. In terms of the result of 10-folds cross validation, $N_s = 0$ does not work well. Too much information propagation is carried out so that similar instance-subpopulations are converged rapidly. In other word, the genetic diversity in the population was easily lost. In the case of $N_s \neq 0$, performances do not differ with each other. However, the population size is significantly different. The large population size causes a large amount of computation time. We examined that $N_s = 10$. However, its performance is worse than $N_s = 0$.

As shown in Table II, we compare the proposed method with other algorithms. In the table, data except for the proposed method is cited from [3]. The bold numbers in the table denote that the proposed method outperforms by corresponding method with statistical significance. The italic numbers means that the proposed method is outperformed. The propose method is competitive with other algorithms.

V. CONCLUSIONS

In this paper, a new EDA with instance-subpopulations is proposed. One of the main features of the proposed method is that reproduction by using probabilistic model is carried out for individuals matching to each instance. By constituting such instance-subpopulations, all EDAs in each instance-subpopulation need to do is to learn whether "don't care symbol" should be set or not for corresponding training instance. We examined the proposed method on Breast-cancer in Wisconsin and Chess End-Game. The comparisons with other algorithms elucidate the effectiveness of the proposed method.

ACKNOWLEDGMENT

This work was partially supported by the Grant-in-Aid for Young Scientists (B) of MEXT, Japan (21700254).

REFERENCES

[1] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[2] H. Ishibuchi and Y. Nojima, "Accuracy-complexity tradeoff analysis by multiobjective rule selection," in *Proceedings of ICDM 2005 Workshop on Computational Intelligence in Data Mining*, 2005, pp. 39–48.

[3] M. Butz, "Rule-based evolutionary online learning systems: Learning bounds, classication, and prediction," IlliGAL, University of Illinois at Urbana-Champaign, Tech. Rep. Report No. 2004034, 2004.

[4] S. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.

[5] F. Kharbat, L. Bull, and M. Odeh, "Mining breast cancer data with xcs," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2007, pp. 2066–2073.

[6] M. V. Butz and M. Pelikan, "Studying xcs/boa learning in boolean functions: structure encoding and random boolean functions," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 1449–1456.

[7] P. Larranaga and J. Lozano, Eds., *Estimation of Distribution Algorithms, A New Tool for Evolutionnary Computation*, ser. Genetic Algorithms and Evolutionnary Computation. Kluwer Academic Publishers, 2002.

[8] S. Baluja, "Using a priori knowledge to create probabilistic models for optimization," *International Journal of Approximate Reasoning*, vol. 31, pp. 193–220(28), November 2002.

[9] J. S. De Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," *Advances in Neural Information Processing Systems*, vol. Vol. 9, 1997.

[10] K. C. Tan, Q. Yu, and T. H. Lee, "A distributed evolutionary classifier for knowledge discovery in data mining," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 35, no. 2, pp. 131–142, 2005. [Online]. Available: http://dx.doi.org/10.1109/TSMCC.2004.841911

[11] K. Bennett and O. Mangasarian, "Robust linear programming discrimination of two linearly inseparable sets," *Optimization Methods and Software*, vol. 1, pp. 23–34, 1992.

[12] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html