Parallel Distributed Multi-objective Fuzzy Genetics-based Machine Learning Mid Term Report

Bowen Zheng, Shijie Chen, Shuxin Wang
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, Guangdong, China

***Abstract*—In the second period of the project, we finished the design of fuzzy classifiers, GBML framework and the asynchronous parallel distributed system. We have implemented each part respectively and are working to integrate them together.**

I. Introduction

In this project, we aim to build a parallel distributed implementation of a multi-objective genetics based machine learning(GBML) algorithm. We choose a specific problem of a three-objective fuzzy rule-based classifier and fit it into a hybrid GBML framework. Then we develop a parallel mechanism to accelerate computation.

Code of the three parts have been completed. We will integrate them together, fix bugs and run some test problems in the next stage.

II. Fuzzy Rule-based Classifiers

The design and implementation of fuzzy classifier is based on [1].

A. Fuzzy Rules

We use the following "if-then" rules:

$$Rule\ R_q:\ if\ x_{pi}\ is\ A_{qi},\ i \in [1,n]\ then\ Class\ C_q\ with\ CF_q$$

TABLE I
Notation

Variable	Name
n	dimension of patterns
M	number of classes of patterns
S	a fuzzy classifier
I	number of membership functions
x_p	a pattern vector
x_{pi}	attribute of x_p value on i-th dimension
A_{qi}	antecedent fuzzy set
$\mu_{A_{qi}}(x)$	membership function of A_{qi}
$\mu_{A_q}(x_p)$	compatibility grade of x_p with A_q
A_q	antecedent part of q-th rule
R_q	q-th rule
C_q	consequent class for q-th rule
CF_q	rule weight for q-th rule

Input vectors are normalized to a hypercube $[0,1]^n$ using the following equation:

$$x_{pi} = \frac{x_{pi} - min(x_i)}{max(x_i) - min(x_i)}$$

The antecedent fuzzy set contains a membership function of the form:

$$\mu_{A_{qi}}(x_{pi}) = max\{1 - \frac{|a - x_{pi}|}{b}, 0\}$$

$$a = \frac{k-1}{K-1}$$

$$b = \frac{1}{K-1}$$

Where K is the number of intervals of fuzzy set A_{qi} and k is the order of the interval. As shown in fig.1. The $don't\ care$ condition is a constant function with value 1. As we will discuss in classification process, the feature x_{pi} with A_{qi} being $don't\ care$ is ignored.

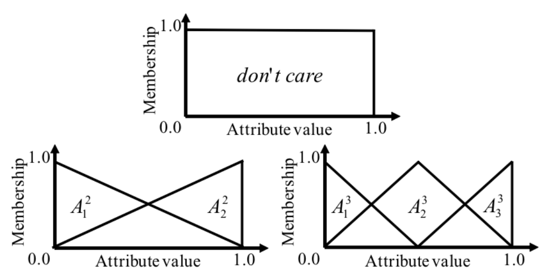

Fig. 1. Membership functions with at most 3 intervals

The antecedent part of R_q is a set of antecedent fuzzy sets.

$$A_q = \{A_{qi} | i \in [1,n]\}$$

We define the compatibility grade of an pattern x_p with rule R_q as

$$\mu_{A_q}(x_p) = \prod_{i=1}^{n} \mu_{A_{qi}}(x_{pi})$$

Then we determine the consequent class C_q and rule weight CF_q using training patterns as follows:

First, we compute the confidence of fuzzy rule R_q to each class $c(A_q \Rightarrow Class\ h), h \in [1, M]$

$$c(A_q \Rightarrow Class\ h) = \frac{\sum\limits_{x_p \in Class\ h} \mu_{A_q}(x_p)}{\sum\limits_{p=1}^{m} \mu_{A_q}(x_p)}$$

Then the consequent class C_q as the class with which R_q has the largest confidence.

$$c(A_q \Rightarrow Class\ C_q) = max\{c(A_q \Rightarrow Class\ h) | h \in [1, M]\}$$

At last, rule weight of R_q is given by the difference between its consequent class and other classes.

$$CF_q = c(A_q \Rightarrow Class\ C_q) - \sum_{h=1, h \neq C_q}^{M} c(A_q \Rightarrow Class\ h)$$

Rule weight shows the quality of a classification result given by a fuzzy rule. If a rule has negative rule weight, its abandoned.

B. Fuzzy Classifier

A fuzzy classifier S is a set of fuzzy rules. Given an input pattern x_p, the classification result C_w is produced by a winning rule R_w, chosen as follows:

$$\mu_{A_w}(x_p) \cdot CF_w = max\{\mu_{A_q}(x_p) \cdot CF_q | R_q \in S\}$$

C. Generate Fuzzy Rule From Training Patterns

At the initiation stage of our GBML algorithm, the population, set of fuzzy classifiers, is created from training data. We can use certain training patterns to create a fuzzy classifier and repeat the process to create a set of classifiers.

Each rule R_t in each classifier is generated from a training pattern x_t as follows:

We choose an antecedent fuzzy set according to each x_{ti} and form the antecedent part A_t of rule R_t so that x_t has the largest compatibility:

$$\mathsf{A}_{ti} = \arg\max_{\mu_j}\{\mu_j(x_{ti})\}, j \in [1, I]$$

Then, randomly change the antecedent fuzzy set to $dont't\ care$ according to a pre-specified probability p_{dc}. This step prevents overfitting and improves generalizing ability of fuzzy rules.

III. Hybrid Genetics-based Machine Learning Framework

After designing the fuzzy classifier, the next step is to approach our three objectives: maximizing number of correctly classified training patterns, minimizing number of fuzzy rules and minimizing total number of antecedent conditions in the fuzzy classifier S. Our current implementation only consider fist two objectives.

We use a hybrid GBML algorithm to find the non-dominated rule sets. The genetic algorithm we used is based on [1]. This hybrid GBML is implemented in the framework of non-dominated sorting genetic algorithm II(NSGA-II) and it's a Pittsburgh-style algorithm. Besides,it use Michigan-style GBML to change a rule as the mutation operation.

In our genetic algorithm, the population is a set of fuzzy classifiers, and each individual is a single fuzzy classifier. The basic steps of our algorithm are:

1) Initialize the population with size N_{pop} using training data.
2) Generate N_{pop} offsprings by crossover operation. Then apply mutation on the offsprings.
3) Combine the original population and offsprings and keep the best N_{pop} individuals according to Pareto ranking and crowding measure.
4) Repeat 2) and 3) until the stopping criterion is met.

A. NSGA-II

NSGA-II, which is a common algorithm for multi-objective optimizing problems, is introduced in detail in [2]. Compared to ordinary genetic algorithms, it uses elite-preserving, Pareto ranking, and crowding measure to control the population. Elite-preserving is a strategy that we keep the individuals with the best performance in each iteration. Pareto ranking and crowding measure are used for non-dominated sorting in the three-objective situation. Individuals are ranked by Pareto-ranking first. If two individuals have the same Pareto-ranking, on with lower crowding measure is preferred.

1) Elite-preserving: Elite-preserving is a replacing strategy. The idea is that we replace individuals with bad performance in the original population by the outstanding offsprings, and keep preeminent parents to reserve their high performance gene.

2) Pareto ranking: Pareto-optimal front is the solution set that we want to find in the multi-objective optimizing problem, and its definition can been seen at [1]. Solutions are divided into non-dominated solution sets where there is no solution dominated by another solution in the set. Pareto ranking are assigned as the number of non-nominated solution sets that dominates it. In particular, the Pareto-optimal front is not dominated and is seen as the set of best solutions.

3) Crowding measure: Crowding measure is used to find a more evenly distributed Pareto set and maintain the diversity of the population. This will enhance the algorithm's ability to escape local optimal. An individual's crowding measure is computed as the sum of its distances to its neighboring vectors on both positive and negative directions for each objective. Priority is given to individuals with larger crowding measure.

B. Michigan-style GBML

Different from the Pittsburgh-style algorithm, Michigan-style GBML sees a single fuzzy rule as the population and sees the membership function as an individual. We use Michigan-style GBML as a mutation operator acting on a single rule to enrich the diversity of our population.

IV. Asynchronous Parallel Distributed System Design

V. Computational Experiments

Our current implementation has integrated the hybrid GBML framework and the fuzzy rule-based classifier. Feasibility test is done on the Iris Flower dataset. The result is as follows:

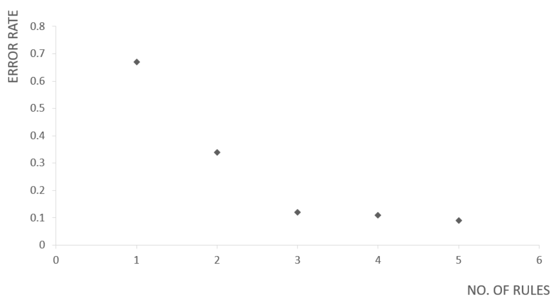

Fig. 2. Pareto front obtained on Iris Flowers

A. Dataset distributor:

This layer is responsible for the dataset distribution process, which delivering a partial of dataset to each island. During this process, the distributor picks N_{land} data from the dataset pool without replacement, then put them into an island on one slave process. Repeat this procedure until all islands get their "environment"

B. Migration Algorithm:

This layer is responsible for the population exchange process. Before slave processes run "init" part, this layer establishes connections between the master process and each other island to listen to them. When an island completes that part, it will stop, upload the current population to this layer and wait for new population feedback from this layer. After this layer receives roughly trained population from all islands, it will generate a population pool based on these population and finished the initialization of this layer.

After the initialization, this layer will delivering new population to each island and start their "main" part. The new population is generated by random sampling N_{pop_i} individuals from the population pool without replacement.

After the "main" part is started, each island can choose to update its population through this layer when finished certain iterations I_{update} by themselves, which is known as one epoch.This layer will firstly put the population uploaded from the island to the pool and randomly select the same amount of individual from the pool, then delete them from the pool. Note that comparing to the process of initialization of population pool, the control rule is given to the island rather than this control layer, therefore it is a more asynchronous way to deal with the migration of population between the island and the big population pool, which will reduce or even eliminate the barrier time cost. This will significantly improve the performance when the spent time on each epoch varies dramatically for different islands.

VI. Contribution

- Bowen Zheng - Design & Implementation of parallel system
- Shijie Chen - Design & Implementation of fuzzy classifier, Design of parallel system
- Shuxin Wang - Design & Implementation of Hybrid GBML framework

Acknowledgment

References

[1] H. Ishibuchi and Y. Nojima, "Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning," *International Journal of Approximate Reasoning*, vol. 44, no. 1, pp. 4–31, 2007.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, April 2002.