A Meta-Learning Approach based on Mean Field Genetic Algorithms

Chuleui Hong and Yeongjoon Kim

Department of Computer Science, Sangmyung University, Seoul, Korea
{yjkim, hongch}@smu.ac.kr

Abstract

Mean Field Genetic Algorithm (MGA) is a hybrid algorithm of Mean Field Annealing (MFA) and Simulated annealing-like Genetic Algorithm (SGA). It combines benefit of rapid convergence property of MFA and effective genetic operations of SGA. This paper presents an approach for building a multi-classifier system in a MGA-based inductive learning environment. Multiple base classifiers are combined to build a multi-classifier system. A base classifier consists of a general classifier and a meta-classifier. The general classifier performs regular classification task. The meta-classifier evaluates classification result of its general classifier and decides whether the base classifier participates into a final decision-making process or not. The paper discusses our approach in details and presents some empirical results that show the improvement we can achieve with our approach.

Keywords: Multi-classifier, Mean Field Genetic Algorithm, Inductive learning

1. Introduction

The concept of combining multiple classifiers into one classification system has become very popular [1-5]. The main purpose of creating a complex multi-classifier system is to obtain better classification performance than the performance offered by its components – individual base classifiers. Among those works of integrating multiple learned models, Doan et. al., [6] and Fan et. al., [7] have explored a multi-strategy learning approach that applies multiple learner modules to a given problem, then combines the predictions of modules using a meta-learner.

Mean Field Genetic Algorithm (MGA) is a hybrid algorithm based on Mean Field Annealing (MFA) and Simulated annealing-like Genetic Algorithm (SGA) [8]. MFA has the characteristics of rapid convergence to the equilibrium state while the simulated annealing takes long time to reach the equilibrium state. MGA combines benefit of rapid convergence property of MFA and effective genetic operations of SGA.

We explore a meta-learning approach for building a multi-classifier system in our MGA-based inductive learning environment. In our approach, several base classifiers are obtained from given training data set by executing MGA-based inductive learning system multiple times. Then, base classifiers are combined to build a multi-classifier system. A base classifier consists of a general classifier and a meta-classifier. The general classifier performs regular classification task. The meta-classifier evaluates classification result of its general classifier and decides whether the base classifier participates into a final decision-making process or not.

1 Corresponding author.
The paper is organized as follows. Section 2 explains how to obtain base classifiers and combine them to build a multi-classifier system. Section 3 discusses briefly an MGA-based inductive learning environment in which our approach has been explored. Section 4 presents some empirical results and Section 5 concludes the paper.

2. Multi-classifier System

2.1. Learning Classification Rules

From given examples, our system learns PROSPECTOR-style rules [9] that have the form:

\[
\text{If } E \text{ then } H \text{ with } S = s; \quad N = n
\]

where \( S \) and \( N \) are odds-multipliers, measuring the sufficiency and necessity of \( E \) for \( H \). In general, PROSPECTOR rules work with odds instead of probabilities, using the following conversion from probabilities to odds: \( O(H) = P(H)/(1-P(H)) \). That is, a probability of 0.75 is converted to odds of 3 (\(=0.75/0.25\)). If we have a rule, “If \( E \) then \( H \) with \( S=2.0; N=0.1 \)”, it expresses that presence of \( E \) increases prior odds of \( H \), \( O(H) \), by a factor 2, whereas absence of \( E \) \( (P(E')=0) \) decreases prior odds of \( H \) by a factor of 0.1. In the case that \( P(E')=1 \) (resulting in an odds-multiplier of \( N \)) or \( 0 \) (resulting in an odds-multiplier of \( S \)), it becomes necessary to interpolate.

The system learns two kinds of rules from given examples: is-high rules and is-close-to rules. An example of is-high rule is

\[
\text{If is-high (A) then D with } S=3; \quad N=0.1
\]

For a given example, the is-high rule produces an odds-multiplier between 3 and 0.1 based on relative highness of value for attribute A of the example to other examples. An example of is-close-to rule is

\[
\text{If is-close-to (A, a) then D with } S = 4; \quad N = 0.2
\]

It produces an odds-multiplier between 4 and 0.2 based on closeness of value for attribute A to a certain constant \( a \).

The posterior odds for a decision \( D \), \( O(D') \), is computed as follows:

If following rules provide evidence for the decision \( D \)

\[
(r_1) \text{ If } E_1 \text{ then } D \text{ with } S=s_1; \quad N=n_1
\]

\[
\vdots
\]

\[
(r_m) \text{ If } E_m \text{ then } D \text{ with } S=s_m; \quad N=n_m
\]

then the posterior odds of \( D \), \( O(D') \), is computed as follows:

\[
O(D') = O(D|E_1' \land \ldots \land E_m') \cdot \prod_{i=1}^{m} \lambda_i
\]

where \( \lambda_i = O(D|E_i') / O(D) \) is the odds-multiplier of rule \( r_i \).

A rule-set that consists of learned rules makes a decision as follows: assuming that we have decision candidates \( DC = \{D_1, \ldots, D_n\} \) with prior odds \( O(D_1), \ldots, O(D_n) \), these odds are updated by firing rules of the rule-set, yielding posterior odds \( O(D_1'), \ldots, O(D_n') \), and a decision candidate with the highest posterior odds is selected.
2.2. Building a Multi-classifier System

A base classifier consists of two classifiers, a general classifier, GC, and a meta-classifier, MC. GC is a classifier that learns concept descriptions from given training data set to perform regular classification task. MC learns the bias of GC so that it can decide whether a prediction made by GC is correct or not. The training process to obtain a base classifier consists of two phases. In the first training phase, GC is learned from given training data set. The training data set for GC, denoted by $T_{GC}$, is a regular training data set in which each example is represented with a list of attribute values and classification of the example (see Figure 1-(a)). In the next training phase, MC is trained for the training data set, denoted by $T_{MC}$, in which each element is represented with attribute values of an example in $T_{GC}$, a prediction made by GC for the example, and classification of GC’s prediction which is either 0 or 1, where 0 represents incorrect decision and 1 represents correct decision (see Figure 1-(b)).

![Figure 1. Training Data Set Example](image)

TMC is obtained as follows:

1. Obtain GC from given training data set $T_{GC}$.
2. Classify training examples in $T_{GC}$ with GC. For the prediction of GC for each training example, classify it into one of two classes, 0 or 1, where 0 represents an incorrect decision and 1 represents a correct decision.
3. Using training examples in $T_{GC}$, construct training data set $T_{MC}$ for MC as follows:

   Element in $T_{MC}$ = (attribute values of an example in $T_{GC}$, PGC, CPGC)
where PGC is the prediction of GC and CPGC is the classification of PGC.

A base classifier participates into a final decision-making process for a given unknown example \( t_e \) as follows:

1. GC classifies \( t_e \).
2. MC decides whether the decision made by GC is correct or not with input data that consists of attribute values of \( t_e \) and prediction of GC.
3. If MC classifies GC’s prediction as correct, the base classifier participates into a final decision-making process with classification result of GC. Otherwise it withdraws.

We build a multi-classifier system as follows. In the first step, a certain number of base classifiers are learned for the same training data set. In the second step, a search algorithm is applied to the set of base classifiers obtained in the first step to find a subset of classifiers that provides the best performance. After finding a subset of classifiers, a classification system is constructed with the classifiers in the subset.

3. Mean Field Genetic Algorithm

3.1. Simulated Annealing-like Genetic Algorithm (SGA)

In this paper, a genetic algorithm approach is used to learn appropriate \textit{is-high} rules and \textit{is-close-to} rules from given examples. A population consists of a fixed number of rule-sets and rule-sets themselves are represented in chromosomal representation as ordered sequences of rules \( r_1,\ldots,r_n \). Figure 2 depicts the chromosomal representation of rule-set and rule structure.

![Figure 2. Chromosomal Representation of Rule-set and Rule Structure](image)

1-point crossover operator and mutation operator are used as genetic operators. The 1-point crossover operator creates two offspring by exchanging some rules of selected parents. The mutation operator selects a rule \( r \) from a rule-set and replaces it with a newly generated rule \( r' \). Parents are selected based on their fitness, using the popular roulette wheel method. Fitness of a rule-set is evaluated by the percentage of examples the rule-set classifies correctly.

We modified GA such that the new evolved state is accepted with a Metropolis criterion like simulated annealing in order to keep the convergence property of MFA. The modified GA is called SGA. \( \Delta C \) is the cost change of new state from old state. It is made by subtracting the cost of new state from that of old one. \( T \) is the current temperature.

\[
\Pr[\Delta C \text{ is accepted}] = \min \left( 1, \exp \left( \frac{\Delta C}{T} \right) \right)
\]
The individuals are generated randomly with the probability as same as that of spin matrix in MFA. For example, if spin values of an arbitrary $i^{th}$ class, which is the elements of $i^{th}$ row, is 0.2, 0.4, 0.1, 0.1, 0.2, an individual is made such that for the $i^{th}$ rule, a rule-set can be rule 0 with a probability of 0.2, rule 1 with that of 0.4, rule 2 with that of 0.1, rule 3 with that of 0.1 and so on.

The pseudo code for the distributed genetic algorithm is as follows.

<Simulated Annealing-like Genetic Algorithm, SGA>
Initialize population from MFA spin matrix;
Evaluate population
for number of classes(=N) times
    Select individuals from current population;
    Reproduce next population;
    for (select 2 individuals by turns)
        Perform crossover with probability of crossover;
        Calculate the cost change ($\Delta C$);
        if (exp($-\Delta C/T$) > random[0,1]) then
            Accept new individuals;
    for (all individuals)
        Perform mutation with probability of mutation;
        Calculate the cost change ($\Delta C$);
        if (exp($-\Delta C/T$) > random[0,1]) then
            Accept new individuals;
    Keep the best individual;

3.2. Mean Field Annealing (MFA)

MFA is derived from Simulate Annealing(SA) based on mean field approximation method in physics. The spin matrix is made up of $N \times R$ where $N$ is the number of classifications and $R$ is the number of rules. The randomly selected class-$i$ is responsible for updating its spin value, $s_{ip}$. The objective function, $C(s)$, of MFA is same as that of GA. When the rule-sets of all classes are classified correctly, the objective value will be 0.

$$C(s) = \sum_{i=0}^{N-1} \sum_{j=0}^{R-1} \sum_{p=0}^{R-1} s_{ip} s_{jp} p_{ij}$$

$N$: The number of classes
$R$: The number of rules
$s_{ip}$: The probability of class $i$ mapping to rule $p$
$s_{jp}$: The probability of class $j$ mapping to rule $p$
$p_{ij}$: The percentage of falsely classified for class $i$ and $j$

The pseudo code for the mean field annealing algorithm is as follows.
< Mean Field Annealing>

while (cost change is less than $\epsilon$ for continuous $N$ annealing process)

Select a same class-$i$ at random;
Compute the local mean field of the spins at the $i^{th}$ row;

$$\phi_p = \sum_{j=1}^{N} s_{ip} p_j \quad \text{for} \quad 0 \leq p \leq R - 1$$

Compute the new spin values at the $i^{th}$ row;

$$s_{ip}^{\text{new}} = \frac{e^{\phi_i T}}{\sum_{p=0}^{R} e^{\phi_p T}} \quad \text{where} \quad T \text{ is temperature}$$

Compute the cost change due to spin updates;

$$\Delta C = \sum_{p=1}^{R-1} \phi_p (s_{ip}^{\text{new}} - s_{ip})$$

Update the spin values at the $i^{th}$ row

$$s_{ip} = s_{ip}^{\text{new}} \quad \text{for} \quad 0 \leq p \leq R - 1$$

In implementing MFA, the cooling schedule must be chosen carefully according to the characteristics of problem and cost function. Length of the Markov chain is set to the number of state transitions where the cost change is less than $\epsilon=0.5$ for continuous $N$ annealing process.

### 3.3. MGA Hybrid Algorithm

A new hybrid algorithm called MGA combines the merits of mean field annealing (MFA) and simulated annealing-like genetic algorithm (SGA). MFA can reach the thermal equilibrium faster than simulated annealing and GA has powerful and various genetic operations such as selection, crossover and mutation. First, MFA is applied on a spin matrix to reach the thermal equilibrium fast. After the thermal equilibrium is reached, the population for GA is made according to the distribution of rules of classes in the spin matrix. Next, GA operations are applied on the population while keeping the thermal equilibrium by transiting the new state with Metropolis criteria. MFA and SGA are applied by turns until the system freeze.

The followings are the pseudo code for the distributed MGA algorithm of each node.

< Distributed MGA Hybrid Algorithm >

Forms the spin matrix, $s=[s_{i1}, \ldots, s_{ip}, \ldots, s_{NK}]$;
Get the initial temperature $T_0$, and set $T=T_0$;

while ($T \geq T_f$)

Executes MFA;
Forms GA population from a spin matrix of MFA;
Executes SGA;
Forms the spin matrix of MFA from GA population;
$T = \alpha \times T$; /*decrease the temperature*/
Initial temperature, $T_0$, is set such that the probability where the cost change is less than $\varepsilon (=0.5)$ is more than 95% for the number of classes ($=N$) annealing process. Final temperature ($T_f$) is set to the temperature where the value of the cost change is in $\varepsilon/1,000$ for continuous $N$ temperature changes. A fixed decrement ratio, $\alpha$, is set to 0.9 experimentally.

4. Experiments

We performed some experiments to evaluate performance of the multi-classifier system built in our MGA-based learning environment. We used four data collections: iris data collection (FL), glass data collection (GL), radar signal data collection (RAD), and soybean diseases data collection (SBD). The characteristics of each data collection are as follows:

- Iris data collection (FL): the data set contains 4 numeric attributes (sepal length, sepal width, petal length, petal width) and class of 150 instances obtained from 3 classes, where each class refers to a type of iris plant.
- Glass data collection (GL): the data set contains 214 instances obtained from 6 different kinds (building-windows-float processed, building-windows-non-float processed, containers,...) of glass, in which each instance is represented with 9 numeric-valued attributes (Na, Fe, K, ...).
- Radar signal data collection (RAD): good and bad radar signals are described by thirty four attributes. The data set has 351 instances.
- Soybean diseases data collection (SBD): each of fifteen diseases that affect soybean plants is described by 35 attributes. The data set contains 290 instances.

For each data collection, training/testing data sets were generated by equally dividing data set into two subsets. Then, base classifiers were learned by executing the learning system multiple times with training data set. In the following step, a genetic algorithm is applied to find the subset of base classifiers that provides the best performance. Finally, a multi-classifier system was constructed with base classifiers, and performance of it was evaluated with testing data set. This whole process was repeated five times.

**Table 1. Comparison of Classification Performance**

<table>
<thead>
<tr>
<th>Data Set</th>
<th>OC (Train/Test)</th>
<th>MC (Train/Test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FL</td>
<td>98.5/94.0</td>
<td>100.0/95.4</td>
</tr>
<tr>
<td>GL</td>
<td>70.2/59.5</td>
<td>81.7/64.4</td>
</tr>
<tr>
<td>RAD</td>
<td>92.2/85.5</td>
<td>98.0/90.4</td>
</tr>
<tr>
<td>SBD</td>
<td>62.7/52.2</td>
<td>78.4/65.7</td>
</tr>
</tbody>
</table>

The results of experiments are depicted in Table 1. In the table, "OC" denotes performance of a classification system that has one classifier and "MC" denotes performance of the multi-classifier system. For the radar signal data collection, our approach improves classification performance by more than 5.8% for training data set and 5.1% for testing data set. For the soybean diseases data collection, classification performance is increased quite significantly for both training and testing data set, improving performance of the system by more than 15.7% and 13.5%, respectively.
5. Conclusions

We have explored an approach for building a multi-classifier system in a MGA-based inductive learning environment. In our approach, several base classifiers are combined to build a multi-classifier system. A base classifier consists of a general classifier and a meta-classifier and the role of a meta-classifier is to evaluate classification result of its general classifier and decide whether the base classifier participates into a final decision-making process or not. Experiments reveal that our approach improves performance of a classification system quite significantly.

Acknowledgements

This research was supported by a 2013 Research Grant from Sangmyung University.

References


Authors

Chuleui Hong received his M.S. degree and Ph.D. degree in Computer Science at New Jersey Institute of Technology, USA and University of Missouri-Rolla, USA, respectively. He was a senior researcher in Electronic and Telecommunications Research Institute, Korea. He is currently professor of Computer Science Department at Sangmyung University, Seoul, Korea. His research interests include Parallel and Distributed System, Optimization Algorithm, Multimedia Application, and Intelligent Agent, Neural Networks, Expert System.

Yeongjoon Kim received B.E. in Industrial Engineering from Korea University in 1984 and Ph. D. in Computer Science from University of Houston in 1996. He is currently a professor of Computer Science at Sangmyung University. His research interests are in the areas of Machine learning, Expert systems, and Genetic algorithms.