Optimization of fuzzy classifier parameters with a combination of gravitational search algorithm and shuffled frog leaping algorithm

M B Bardamova and I A Hodashinsky
Tomsk State University of Control Systems and Radioelectronics, 40, Lenina Ave.,
Tomsk, 634050, Russia

E-mail: 722bmb@gmail.com

Abstract. In the present article, we analyse the effectiveness of combining two metaheuristic algorithms for tuning parameters of a fuzzy classifier. To work with imbalanced data, a fitness function is used based on a compromise between the overall accuracy and the geometric mean of accuracy of each class. The experiment was performed on data sets from the “Knowledge Extraction based on Evolutionary Learning” repository with different imbalance coefficients.

1. Introduction
Metaheuristic algorithms are widely used to tune the parameters of fuzzy systems [1-4]. According to the "no free lunch" theorem, there is no algorithm which can demonstrate superior results on all data sets, so developers of intelligent systems should have a variety of different effective algorithms. The effectiveness of metaheuristics consists not only of operating time or the ability to demonstrate acceptable results on various tasks. It is also important to be able to overcome local extremes.

Various modifications are being developed to improve the performance of metaheuristics. For example, mutations or new coefficients for operators are added. Another approach is hybridization, which is combining several algorithms into a single combination. In this paper, we investigate the advantage of combining two metaheuristics for tuning the antecedents of fuzzy classifiers. The original algorithms are the gravitational search and the shuffled frog leaping algorithm, which were previously used individually in fuzzy system configuration problems [1, 5].

The aim of the work is to verify the effectiveness of the hybrid algorithm based on the gravitational search algorithm and shuffled frog leaping algorithm to tune the parameters of a fuzzy classifier of imbalanced data in comparison with the above metaheuristics working separately.

The main contributions are the following:
1) the hybrid algorithm for tuning parameters of the fuzzy classifier of imbalanced data based on two metaheuristics has been developed;
2) the hybrid has demonstrated a higher average geometric accuracy than its individual algorithms.
3) a statistical comparison of the hybrid and originative algorithms was performed.

2. Fuzzy classifier
The structure of the fuzzy classifier includes two components: a base of production rules of the "if-then" type and a list of fuzzy terms that reflect the belonging degree of the feature to the rule. The initial structure of the classifier is either set with an expert, or generated by special algorithms. However, structure
generation algorithms may not always provide high quality classification. One of the reasons is the
suboptimal position of fuzzy terms, which does not accurately describe the subject area of the problem.
Optimization methods, including metaheuristic algorithms, can be used to correct this situation.

To perform the parameter tuning an antecedent vector $\theta_0$ is formed. It consistently includes the
parameters of the linguistic terms of all the features involved in the classification. We use Gaussian
functions that have two parameters: $b$ is the coordinate of the term vertex along the abscissa axis, and $c$
is the spread of the term. If the number of terms for each attribute is equal to the amount of rules, the
parameter vector can be represented as follows: $\theta = \{b_{11}, c_{11}, \ldots, b_{1r}, c_{1r}, b_{21}, c_{21}, \ldots, b_{nr}, c_{nr}\}$, where $r$
is the amount of rules and $n$ is the number of features. This vector is input to the optimization algorithm.

3. The explored algorithms

Since the gravitational search algorithm and the shuffled frog leaping algorithm are population-based, a
population is formed for each based on the input vector with some deviation: $\Theta = \{\theta_0, \theta_1, \ldots, \theta_N\}$,
where $N$ is the population size. The quality of each solution vector is evaluated using the fitness function.
We use the following function:

$$\text{fit}(\theta) = \gamma \cdot \text{GM}(\theta) + (1 - \gamma) \cdot \text{Acc}(\theta),$$

Where $\text{GM}(\theta)$ is average geometric accuracy, $\text{Acc}(\theta)$ is overall accuracy, and $\gamma$ is the priority coefficient
[5]. We use $\gamma = 0.5$, which helps to get balanced accuracy even if there is a significant imbalance in the data.

3.1. Gravitational search algorithm

The gravitational search algorithm (GSA) was proposed by Esmat Rashedi and her colleagues in 2009
[6]. In accordance with its name, this metaheuristic uses adapted laws of gravity to find extremes.
Solution vectors imitate a system of particles that have a certain mass. The mass depends on the value
of the fitness function. The best solution has a greater mass, and attracts the other particles more strongly.
Under the influence of gravity, the particles gain some acceleration and change their location, thereby
updating the solution vector.

For the gravitational search algorithm, the following parameters are input: the initial vector of
antecedents $\theta_0$, the number of iterations $T$, the amount of particles in the population $N$, the initial value
of the gravitational constant $G_0$, the coefficient of reduction of the gravitational constant $\alpha$, and a small
constant for finding the gravitational force $e$. A population of particles is created based on the initial
vector $\theta_0$, and the fitness function is calculated for each particle.

The gravitational search algorithm can be represented as the following sequence of steps.

At the first step, the particle masses are estimated:

$$\text{mass}_i(t) = \left( \frac{\text{fit}(\theta_0(t)) - \text{fit}(\theta_{\text{worst}}(t)))}{\text{fit}(\theta_{\text{best}}(t)) - \text{fit}(\theta_{\text{worst}}(t))} \right) \cdot \text{mass}_i(t),$$

where $M_i(t)$ is the mass of the $i$th particle ($i \in [0, N]$) at the current iteration $t$, $t \in [1, T]$, $\theta_{\text{worst}}$ and
$\theta_{\text{best}}$ are the value of the fitness function of the worst and best particle in this iteration.

The second step is to calculate the acceleration $a^d_i$ for the $d$th element of the $i$th particle:

$$a^d_i(t) = G(t) \cdot \sum_{j=0, j \neq i}^{N} \text{rand} \cdot M_j \cdot \left( \theta^d_j(t) - \theta^d_i(t) \right) \left( \left\| \theta^d_j(t) - \theta^d_i(t) \right\| - e \right)$$

where $\text{rand}$ is a uniformly distributed random number in $[0; 1]$, $G(t)$ is the value of the gravitational
constant, updated at each iteration as follows:

$$G(t) = G_0 \cdot e^{-\alpha \frac{t}{T}}.$$  

In the third step, the speed of the $d$th element of the $i$th particle is calculated as the sum of the
components of the current speed $V^d_i(t)$ and the accumulated acceleration:

$$V^d_i(t+1) = \text{rand} \cdot V^d_i(t) + a^d_i(t).$$

2
The particle elements are updated in the fourth step:

$$\theta^f_i(t+1) = \theta^f_i(t) + V_i^d(t).$$

Then the fitness function is recalculated and the stop criterion is checked. If the iteration counter does not reach the maximum value, the algorithm returns to the first step.

3.2. Shuffled frog leaping algorithm

The memetic algorithm, the idea of which is to simulate the process of searching for food by a population of frogs, was first described in 2003 [7]. Since its creation, the algorithm has received several modifications, and the version of the algorithm described in [8] has become widespread. The original population is sorted and sequentially divided into independent groups called memeplexes. Within them, vectors with the worst fitness function move iteratively in the direction of the best ones. After the specified number of iterations expires, all vectors are combined into one population and the entire process is repeated again.

To speed up the operation of the shuffled frog leaping algorithm (SFLA), we don’t use the physical division of vectors into groups, but calculate the indexes of the necessary vectors. The realization of the algorithm is described below.

Input parameters are the following variables: the initial vector of parameters \( \theta_0 \), the number of global iterations \( T_g \), the amount of global iterations \( T_c \), the number of memeplexes \( n_{mem} \), the number of vectors in the memeplexes \( n_{agents} \), and the coefficient \( const \). As in the previous algorithm, the quality of solutions in the created population is evaluated before the iterations begins.

The first step is a global search, which consists of sorting the population of vectors in descending order of the fitness function value.

The second step is a local search. At each iteration of the local search \( t \in [1, T_c] \), a vector \( \text{New} \) is generated inside each memeplex:

$$\text{New} = \text{rand} \cdot \text{const} \cdot (\text{best}(t) - \text{worst}(t)) + \text{worst}(t),$$

where \( \text{rand} \) is a uniformly distributed random number within the range from 0 to 1, \( \text{best}(t) \) is the vector with the best value of the fitness function in the memeplex, and \( \text{worst}(t) \) is one of the vectors with the worst fitness function. The best vector for the \( k \)th memeplex is the vector \( \theta_k \) \((k \in [0, n_{mem}))\). The solution with the following index is selected as the worst vector:

$$w = N - f \cdot n_{mem} + k,$$

where \( f \) is the replacement counter, initially equal to one. Its presence allows not replacing the same vector. The vector \( \text{New} \) is generated based on the operator proposed in [8]:

$$\text{New} = \text{rand} \cdot \text{const} \cdot (\text{best}(t) - \text{worst}(t)) + \text{worst}(t).$$

The value of the fitness function of \( \text{New} \) is evaluated; if it is greater than the value of the fitness function of the vector \( \theta_w \), then \( \text{New} \) replaces \( \theta_w \), and the replacement counter \( f \) increases by one. Otherwise, \( \text{New} \) is re-generated, but the globally best vector \( \theta_0 \) is used as the \( \text{best}(t) \) vector. If the condition \( \text{fit}(\text{New}) > \text{fit}(\theta_w) \) is met, \( \text{New} \) replaces \( \theta_w \), and one is added to \( f \). If the condition isn’t met, a random vector based on \( \theta_0 \) with some deviation is generated in place of \( \theta_w \), and the \( f \) value does not change.

Thus, a local iteration ends when \( \theta_w \) is replaced in all memeplexes, and at this time the value of \( f \) must be checked. In order not to go beyond the population, its maximum value must be limited. It is enough to make sure that \( f \) does not exceed \( n_{agents} - 1 \), but we prefer to check this condition:

$$f > n_{agents} / 2,$$

and when it is executed, again equate \( f \) to one.

After all local iterations are completed, the algorithm returns to step one. The process continues until the number of global iterations expires, and the solution with the best fitness function value is sent to the output.
3.3. Combination of algorithms

The described algorithms have disadvantages. The GSA has complex calculations, and, as a result, a long work time. In addition, if a particle falls into a local extremum, it quickly pulls the whole population there. The SFLA, in contrast, has only one simple operator. It allows exploring a search area locally, without major changes to the vector elements. Since the first algorithm is a global search, and the second is a local search, their combination can achieve better results.

Schematically, the combination of algorithms can be represented as a block-diagram (figure 1).

![Block-diagram of the hybrid algorithm based on the combination of the gravitational search algorithm and local search from the shuffled frog leaping algorithm.](image)

The main part of the combination is the gravitational search that works without any changes, but with fewer iterations. After the end of each iteration, an entry condition to the local search must be checked. The following criteria can be selected as a condition: absence of any changes in the best solution over a given number of iterations, or too small distance between vectors, or some given iteration.

After getting into the local search from the SFLA, the vectors are sorted, then the worst solutions are iteratively replaced in each memeplex according to the idea of this metaheuristic. When replacing a vector, the corresponding velocity vector must be cleared. After performing all iterations of the local search the algorithm returns back to the global search.

4. Experiments

The purpose of the experiment was to test the performance of the hybrid algorithm. Twelve imbalanced data sets with two classes from the KEEL repository were selected [9]. The characteristics of data sets are shown in Table 1. Here $F$ is the number of features in the set, $inst_{all}$ is the total instance count, $inst^+$ is the number of instances of a positive class, $inst^-$ is the number of instances of a negative class, and IR is the coefficient of imbalance (the ratio of the amount of instances a larger class to the amount of instances of the smallest one).

**Table 1. Description of data sets for testing the fuzzy classifier.**

| Data sets     | Abbreviation | $F$ | $inst_{all}$ | $inst^+$ | $inst^-$ | IR  |
|---------------|--------------|-----|--------------|----------|----------|-----|
| wisconsin     | Wis          | 9   | 683          | 239      | 444      | 1.9 |
| glass0        | gl0          | 9   | 214          | 70       | 144      | 2.1 |
| haberman      | Hbr          | 3   | 306          | 81       | 225      | 2.8 |
| vehicle0      | vhc0         | 18  | 846          | 199      | 647      | 3.3 |
| newthyroid2   | nth2         | 5   | 215          | 35       | 180      | 5.1 |
| segment0      | sg0          | 19  | 2308         | 329      | 1979     | 6.0 |
| page-blocks0  | plb0         | 10  | 5472         | 559      | 4913     | 8.8 |
| vowel0        | vwl0         | 13  | 988          | 90       | 898      | 10.0|
| cleveland-0v4 | clv0-4       | 13  | 177          | 13       | 164      | 12.6|
| ecoli4        | ec14         | 7   | 336          | 20       | 316      | 15.8|
| abalone9-18   | abl9-18      | 8   | 731          | 42       | 689      | 16.4|
| yeast4        | yst4         | 8   | 1484         | 51       | 1433     | 28.1|
The experiment used five-fold cross-validation. An algorithm based on extreme values of features of classes was used to generate the classifier structure. Then the parameters of the terms were independently optimized using the following algorithms: gravitational search (GSA), shuffled frog leaping algorithm (SFLA), and hybrid algorithm (Hybrid). Each algorithm was run five times, and then the average values of the metrics were calculated. The parameters of the algorithms are shown in Table 2; they are empirically selected as the most universal for the studied data sets.

**Table 2.** Parameters of optimization algorithms.

| Parameter          | GSA                           | SFLA                                         | Hybrid                                      |
|--------------------|-------------------------------|----------------------------------------------|---------------------------------------------|
| Number of iterations | 300 iterations               | 20 global iterations, 15 localizations       | 300 iterations (100 global iterations, and 10 local iterations are applied during each fifth iteration) |
| Number of vectors  | 25 vectors                    | 25 vectors (5 memeplexes for 5 agents)       | 25 vectors (5 memeplexes for 5 agents)       |
| Variables          | $G_0 = 10, \alpha = 10, \varepsilon = 0.01$ | $const = 1.2$                                | $G_0 = 10, \alpha = 10, \varepsilon = 0.01, const = 1.2$ |

**Table 3.** Average geometric accuracy and optimization time using the studied algorithms.

| Data set | Metrics | Initial values | GSA | SFLA | Hybrid |
|----------|---------|----------------|-----|------|--------|
| wis      | $GM_{tr}$ | 75.2           | 97.3±0.2 | 64.1±2.7 | 77.1±2.2 |
|          | $GM_{ts}$ | 73.4           | 95.3±0.7 | 74.4±1.3 | 74.4±1.3 |
|          | $time$   | -              | 39.4±0.2 | 8.4±0.1 | 17.9±0.3 |
| gl0      | $GM_{tr}$ | 64.6           | 81.5±1.3 | 80.9±0.8 | 82.2±0.9 |
|          | $GM_{ts}$ | 60.1           | 77.2±3.7 | 76.7±3.9 | 77.7±2.1 |
|          | $time$   | -              | 15.1±0.1 | 8.4±0.1 | 7.4±0.1  |
| hbr      | $GM_{tr}$ | 50.6           | 64.1±2.7 | 59.6±2.7 | 61.7±2.5 |
|          | $GM_{ts}$ | 44.3           | 54.4±7.1 | 41.5±10.1 | 40.9±5.9 |
|          | $time$   | -              | 10.8±0.1 | 2.2±0.2 | 5.0±0.0  |
| vhc0     | $GM_{tr}$ | 57.0           | 81.9±4.0 | 90.3±2.0 | 91.4±2.0 |
|          | $GM_{ts}$ | 55.5           | 78.9±5.5 | 88.0±3.1 | 90.3±3.2 |
|          | $time$   | -              | 78.1±0.4 | 18.2±0.2 | 36.4±0.3 |
| nth2     | $GM_{tr}$ | 99.1           | 99.9±0.1 | 99.9±0.1 | 100.0±0.0 |
|          | $GM_{ts}$ | 99.2           | 98.0±1.0 | 99.0±0.5 | 97.5±2.1 |
|          | $time$   | -              | 10.8±0.1 | 2.4±0.1 | 5.1±0.2  |
| sgm0     | $GM_{tr}$ | 87.5           | 91.1±2.9 | 98.2±0.8 | 98.7±0.4 |
|          | $GM_{ts}$ | 88.1           | 91.3±2.7 | 98.0±0.9 | 98.3±0.5 |
|          | $time$   | -              | 202.1±2.3 | 54.2±2.2 | 98.4±0.5 |
| pbl0     | $GM_{tr}$ | 63.7           | 77.9±2.8 | 77.4±2.6 | 83.9±2.6 |
|          | $GM_{ts}$ | 63.6           | 77.2±3.2 | 77.1±3.1 | 83.1±2.8 |
|          | $time$   | -              | 277.5±1.4 | 73.7±2.1 | 139.2±6.3 |
| vw0      | $GM_{tr}$ | 84.0           | 95.5±1.6 | 94.3±0.9 | 96.0±1.3 |
|          | $GM_{ts}$ | 83.9           | 92.9±3.0 | 91.1±3.3 | 92.6±3.1 |
|          | $time$   | -              | 70.9±0.4 | 17.6±0.7 | 32.2±0.4 |
| clv0-4   | $GM_{tr}$ | 82.0           | 94.8±2.2 | 96.8±1.0 | 97.3±1.8 |
|          | $GM_{ts}$ | 69.9           | 59.9±11.6 | 56.4±4.7 | 65.8±13.2 |
|          | $time$   | -              | 17.2±0.1 | 3.5±0.2 | 8.2±0.1  |
| ecl4     | $GM_{tr}$ | 81.7           | 91.9±1.0 | 98.9±0.3 | 99.2±0.2 |
|          | $GM_{ts}$ | 68.7           | 76.7±5.5 | 89.3±4.3 | 91.4±1.3 |
|          | $time$   | -              | 16.8±0.4 | 3.8±0.2 | 9.0±0.2  |
| ab19-18  | $GM_{tr}$ | 65.7           | 72.1±2.1 | 79.0±2.3 | 78.6±3.5 |
|          | $GM_{ts}$ | 58.7           | 69.3±6.3 | 75.7±5.9 | 76.0±5.5 |
|          | $time$   | -              | 33.8±0.3 | 7.9±0.2 | 17.3±0.4 |
| yst4     | $GM_{tr}$ | 66.5           | 80.7±2.1 | 85.3±1.0 | 84.7±1.2 |
|          | $GM_{ts}$ | 65.7           | 76.7±3.1 | 78.7±2.4 | 79.6±3.2 |
|          | $time$   | -              | 69.8±2.9 | 17.5±0.3 | 35.1±0.2 |
The results of the experiment in the form of values of average geometric accuracy during training ($GM_t$) and testing ($GM_{tst}$), as well as the time spent by the presented algorithm on performing all iterations (time, in seconds), are shown in table 3. The column named “Initial values” demonstrates the results of the fuzzy classifier without using the parameter tuning operation. The best results are marked in bold.

The Friedman criterion was used to compare the algorithms. The average ranks, Chi-square values, and asymptotic significance calculated using SPSS Statistics are shown in table 4. If the asymptotic significance does not exceed the significance level equal to 0.05, we can conclude that there is a significant statistical difference in the results of the algorithms.

**Table 4. Comparison of classification results by the Friedman test.**

| Algorithm   | $GM_t$ | $GM_{tst}$ | time |
|-------------|--------|------------|------|
| Initial values | 1.00   | 1.67       | -    |
| GSA         | 2.58   | 2.58       | 3.00 |
| SFLA        | 2.71   | 2.42       | 1.00 |
| Hybrid      | 3.71   | 3.33       | 2.00 |
| $\gamma^2$ | 27.53  | 10.10      | 24.00|
| Asymptotic significance | 0      | 0.018      | 0    |

The comparison shows that the proposed hybrid algorithm ranks first in terms of average geometric accuracy in both training and testing. In terms of running time, it is inferior to the shuffled frog leaping algorithm, but ahead of the gravitational search.

The experiment was conducted on a personal computer with the following configuration: Intel Core i5-8400 machine, 2.8 GHz CPU, 16 GB of RAM, and Windows 10 operating system.

5. Conclusion

The results of the experiment show that the hybrid algorithm, which is the combination of the gravitational search algorithm and the shuffled frog leaping algorithm, demonstrates the best quality of classification in most of the studied data sets. If the time spent on training does not play a major role, it is advisable to use it.

Acknowledgments

The reported study was funded by Russian Foundation for Basic Research, project number 19-37-90064. We are grateful to the anonymous reviewer whose comments have helped to improve the quality of the paper.

References

[1] Hodashinsky I A, Bardamova M B and Kovalev VS 2019 Using shuffled frog-leaping algorithm for feature selection and fuzzy classifier design Sci. Tech. Inf. Proc. 46 381–7
[2] Mech M A and Hodashinsky I A 2017 Comparative analysis of differential evolution methods to optimize parameters of fuzzy classifiers J. Comput. Syst. Sci. Int. 56 616-626
[3] Hodashinsky IA, Filimonenko IV and Sarin KS 2017 Krill herd and piecewise-linearization initialization algorithms for designing Takagi–Sugeno systems Optoelectron. Instrument. Proc. 53 379–87
[4] Sarin K, Hodashinsky I and Slezkin A 2018 Feature selection and identification of fuzzy classifiers based on the cuckoo search algorithm RCAI 2018: Artificial Intelligence (Communications in Computer and Information Science vol 934) ed S Kuznetsov, G Osipov, V Stefanuk (Cham: Springer) pp 22-34
[5] Bardamova M, Hodashinsky I, Konev A and Shelupanov A 2019 Application of the gravitational search algorithm for constructing fuzzy classifiers of imbalanced data Symmetry 11 1458
[6] Rashedi E, Nezamabadi-pour H and Saryazdi S 2009 GSA: A gravitational search algorithm Inf. Sci. 179 2232–48
[7] Eusuff M M and Lansey K E 2003 Optimization of water distribution network design using the shuffled frog leaping algorithm J. Water Resour. Planning Mgmt 129 210–25
[8] Elbeltagi E, Hegazy T and Grieson D 2007 A modified shuffled frog-leaping optimization algorithm: applications to project management Struct. Infrastruct. Eng. 353-60
[9] Knowledge Extraction based on Evolutionary Learning, available at: http://keel.es