DESIGNING THE FUZZY RULE BASE BY SOLVING A LARGE-SCALE SINGLE-OBJECTIVE OPTIMIZATION PROBLEM WITH DIFFERENTIAL EVOLUTION [1];

Vladimir Stanovov, Shakhnaz Akhmedova, Eugene Semenkin

Reshetnev Siberian State University of Science and Technology
e-mails: vladimirstanovov@yandex.ru, shahnaz@inbox.ru, eugenesemenkin@yandex.ru
Russian Federation

Abstract: The paper presents an approach for fuzzy rule base design for classification problems, which consists in solving a single-objective unconstrained optimization task. To make this possible, the rule base is encoded by a set of sigmoid functions, 2 functions per variable, and every variable and rule has unique sigmoid functions. A special multi-component variant of differential evolution is developed, with a particular type of mutation strategy assigned to each component. The algorithm is tested on a set of benchmark functions and classification problems, and shows competitive results compared to other approaches.

Key words: fuzzy rule base, classification, differential evolution, large-scale optimization, unconstrained optimization.

1. INTRODUCTION

Fuzzy rule bases have found many applications in many areas, such as control, decision support systems and machine learning [1, 2, 3]. However, in the area of machine learning fuzzy rule bases are still not as popular as other methods, although they have one serious advantage – the rule base could be easily understood by a human. The main reason for the unpopularity of fuzzy machine learning techniques is the fact that the rule base is difficult to design, mainly due to the fact that various heuristic approaches are capable of working only with small datasets, whereas direct optimization approaches can deal with large numbers of variables and huge search spaces. For example, a rule base consists of M rules, each describing N variables of a problem, and there is a possibility to choose among K fuzzy terms for each variable, so the overall number of variants of fuzzy rules is $K^{M \cdot N}$.

[1] The paper has been presented on the International Workshop on Mathematical Models and their Applications 13-15.11.2017, Krasnoyarsk, Russian Federation and it is not published in the Conference Proceedings.

Various approaches have been proposed that make use of both heuristics and optimization approaches to design a rule base which would be accurate, compact and easy to understand [4]. Most of them focus on classification problems, because lots of real-world problems can be formulated as classification. There are two main directions here: a number of research studies, like [5] focus on so-called "accurate fuzzy systems", which usually means that the fuzzy terms in a rule base can change shape, position and also height, delivering more accurate solutions. The advantage of such an approach is that the solutions obtained by such methods are characterized by high accuracy; however, they are not as easy to interpret. Other groups of studies, such as [6, 7] focus on fuzzy rule bases which are easy to interpret. The main difference here is that these approaches fix the fuzzy sets, which allows the application of various heuristics, speeding up the convergence of the used optimization tool. As optimization tools, evolutionary approaches are often used, and these include both classic approaches such as genetic algorithms (GA), genetic programming (GP) and specially designed evolutionary algorithms [8].

In this study, we will focus on accurate fuzzy systems, i.e. fuzzy systems that are designed by tuning the positions and shapes of the fuzzy sets, and as an optimization tool, we will use differential evolution (DE). We mainly focus here on the complexity of the optimization problem obtained by the rule base encoding: it is characterized by a large number of variables. There are several known approaches to handle so-called large-scale global optimization (LSGO), and the most popular is the "divide-and-conquer" (DC) approach [9], which separates the problem into several sum-problems to be optimized by different component algorithms. We implement an approach similar to [9] which calculates the impact of each component algorithm to estimate its effectiveness and change the type of components. The results obtained show the effectiveness of the proposed approach for designing fuzzy rule bases for a number of complex classification problems compared to various simple DE algorithms as well as other approaches.

The paper is organized as follows: Section 1 describes the rule base encoding with the sigmoid function. Section 2 illustrates the optimization algorithm implementation. Section 3 contains the experimental setup and results, and Section 4 concludes the paper.

2. FUZZY RULE BASE ENCODING WITH SIGMOID FUNCTIONS

The rule base encoding is an important part of building a fuzzy classification system, because it defines the structure of the model and its features. Possible encoding variants include, but are not limited to: encoding term numbers, encoding term positions and numbers, encoding term shapes, position and numbers, and encoding a separate shape and position for every variable in every rule. The last approach is used in this paper, i.e. every rule has only one fuzzy term for every variable, so there is no need to choose the term number. Every term is encoded with two

sigmoid functions, which together may form different shapes. The sigmoid function is calculated using the following equation:

$$s(\mathrm{x},\mathrm{y},\mathrm{z}) = \begin{cases} 1 \, if \ (\mathrm{x} \geq \mathrm{y} \, and \, \mathrm{y} \geq \mathrm{z}) \, \mathrm{or} \, (\mathrm{x} \leq y \, and \, z > y), \\ 0 \, if \ (\mathrm{x} \leq z \, and \, \mathrm{y} \geq \mathrm{z}) \, \mathrm{or} \, (\mathrm{x} \geq z \, and \, z > y), \\ \dfrac{1}{1 + e^{\frac{-c}{y-z}(\mathrm{x} + \frac{z-y}{2} - \min(y,z))}} \, if \ y > z, \\ \dfrac{1}{1 + e^{\frac{-c}{y-z}(\mathrm{x} - \frac{z-y}{2} - \min(y,z))}} \, if \ z > y \end{cases} \quad (1)$$

where x is the input variable, y and z are the positions of the left and right tails of a sigmoid, and $c = 15$ is a parameter. The sigmoid tails are cut with two first conditions, as the value in this region is very close to either 0 or 1. An example of two sigmoid functions is presented in Fig. 1.

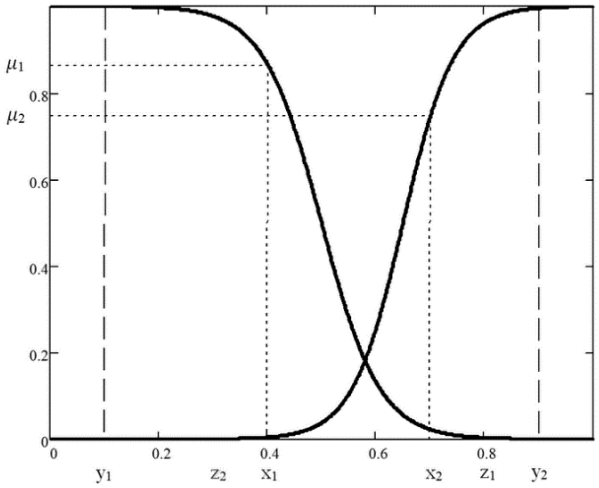

Fig. 1. Two sigmoid functions forming a fuzzy term

Here y_1, z_1 encode the first sigmoid, and z_2, y_2 encode the second one, which is rotated in the opposite direction. Next, x_1 represents the input value, and $\mu_1 = s(x_1, y_1, z_1)$ is the corresponding membership value, x_2 and $\mu_2 = s(x_2, y_2, z_2)$ are similar. The resulting membership function shape is represented as $\min(\mu_1, \mu_2)$. Fig. 2 shows possible variants of the membership function shape obtained by this method.

As can be observed, this gives us the possibility to obtain almost all the possible shapes of a fuzzy term, from flat zero, when the minimum is zero for all x, to

flat unit, when sigmoid functions are at the edges of the [0,1] interval. The first case, when the membership function is always 0, is considered as an empty part of the rule, i.e. the "Don't Care" condition.

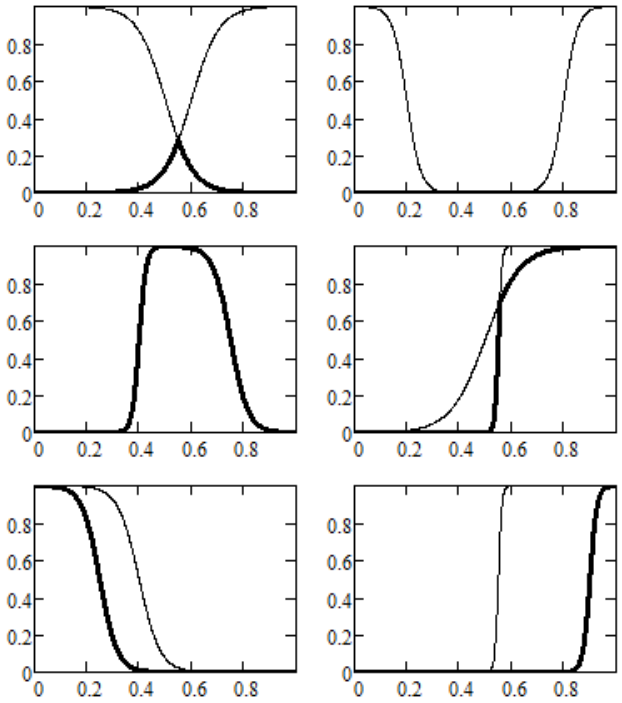

Fig. 2. Possible variants of the fuzzy term shape with a 2-sigmoid representation

The overall encoding of a rule base is shown in the following table.

Table 1.

	X_1	X_2	…	X_N	Class
R_1	$(y_1,z_1,y_2,z_2)^{1,1}$	$(y_1,z_1,y_2,z_2)^{1,2}$	…	$(y_1,z_1,y_2,z_2)^{1,N}$	C_1
R_2	$(y_1,z_1,y_2,z_2)^{2,1}$	$(y_1,z_1,y_2,z_2)^{2,2}$	…	$(y_1,z_1,y_2,z_2)^{2,N}$	C_2
…	…	…	…	…	
R_M	$(y_1,z_1,y_2,z_2)^{M,1}$	$(y_1,z_1,y_2,z_2)^{M,2}$	…	$(y_1,z_1,y_2,z_2)^{M,N}$	C_M

As stated earlier, every rule R_i and every variable X_j has its own membership function, so the total number of variables to optimize is $4*M*N+M$, because the class number C_i is also encoded. For example, for a classification problem having 24 variables and 10 rules, the number of real variables would be 970. Class numbers are represented as real values in the range $[0,|\boldsymbol{C}|]$, where $\boldsymbol{C} = [C_1,\ldots C_k]$, k is the total number of classes, and the corresponding class number is calculated as a

value rounded down. The class number C_w, which is the solution returned by the rule base, is obtained by identifying the winner-rule:

$$R_w = \arg\max_{i \in M}(\min_{j \in N}(\min(s(x, y_1, z_1), s(x, y_2, z_2)))) \quad (2)$$

3. DIFFERENTIAL EVOLUTION ALGORITHM FOR LSGO

The optimization method used in this study is based on the so-called "divide-and-conquer" (DC) strategy [10, 11], designed for large-scale global optimization problems. The key idea of this strategy is to divide the problem with variables $X = [x_1, x_2 \ldots x_N]$ into several sub-problems $X^1, X^2, \ldots X^K$, each containing a subset of the original set of search variables. This could be performed in several ways: for example, each subset may contain random variables from the original set without repetition, or variables could be selected in order, depending on the problem. Here we only considered the case when sub-problems have equal numbers of variables.

After splitting the optimization problem into sub-problems, each sub-problem is solved by a separate optimization algorithm. To calculate the goal function inside each algorithm, the missing variable values are taken from other component algorithms' best solutions. The resulting solution is the combination of best solutions from each algorithm. The best solutions for other component algorithms are defined using the same procedure, which means that the components are dependent on each other's solutions.

The Differential Evolution has been chosen for component algorithms as it shows superior performance for real-parameter optimization, which is confirmed by various competitions. We have used 6 popular [12] variants of the DE mutation scheme, namely DE/rand/1, DE/rand/2, DE/best/1, DE/best/2, DE/target-to-best/1 and the DE/rand/1 variant from a recent paper by K. Price [13]. Two variants of crossover have been applied: binomial crossover with $Cr = 0.5$ for first five mutation strategies and no crossover for the last algorithm, as proposed in [13]. The F value was set to 0.9 for all mutation schemes.

In the initialization procedure for each component algorithm, the mutation type was set randomly. The component type could be changed over time depending on the impact factor of the particular mutation scheme. The impact factors $I = [I_1, I_2, I_3, I_4, I_5, I_6]$ are assigned to every mutation scheme, but not the component algorithm, so there are always 6 impact factors. Here δ is the fitness improvement, j is the index of the component having the highest best fitness value $F_{j,best}$, F_{gbest} is the global best fitness value at the previous step, *NFEval* is the current number of goal function evaluations, *MaxFEval* is the total number of function evaluations available. The whole procedure rewards mutation schemes which produce better solutions independent of the component algorithm where these solutions were generated. The change of the impact I_j depends on the improvement size, and the impacts of other mutation schemes $I_{k \neq j}$ are decreased by the same cumulative value. If all components have the same fitness value, or the improvement is zero, then the im-

pacts of all mutation schemes are decreased by a small value. The impact update procedure is performed at the end of each generation of every algorithm. Impact factors are calculated similar to [9] as follows:

$$I_j = I_j + \delta,$$
$$I_{k\neq j} = I_{k\neq j} - \delta/5,\ k = 1\ldots6, \quad if(F_{j,best} < F_{gbest}\ \&\ \delta\neq0)$$
$$\delta = (F_{gbest} - F_{j,best})/(F_{gbest} + F_{j,best}),$$

$$I_j = I_j + \delta*NFEval/MaxFEval,$$
$$I_{k\neq j} = I_{k\neq j} - \delta*NFEval/MaxFEval/5,\ k = 1..6,\ if(F_{j,best} \geq F_{gbest}\ \&\ \delta\neq0) \quad (3)$$
$$\delta = (F_{gbest} - F_{j,best})/(F_{gbest} + F_{j,best}),$$

$$I_k = I_k - 0.1,\ k=1\ldots6, \quad if(\delta = 0\ or\ min(F_{j,best}) = max(F_{j,best}))$$

The impacts are initialized with zeros, and the Page-Hinkley (PH) statistical test is used to determine the moment when the quality of solving with a certain mutation type falls. PH detects the change in quality if the difference $max_p(|m_p|) - |m_t|$ is larger than the threshold γ. The m values are calculated as follows:

$$m_t = \sum_{p=1}^{t} e_p,\ e_t = I_t - a + \mathrm{V},\ a_t = \frac{1}{t}\sum_{p=1}^{t} I_p, \quad (4)$$

where a is the average impact over t last steps, e_t is the difference between the current and average impact, and m_t is the accumulated difference over the last t steps, Δ is the tolerance value set to 1. γ *was set to 8,* and the number of steps t equalled 10. When the PH test is triggered, all the component algorithms which have the particular type of mutation undergo the mutation scheme selection procedure. In this procedure, the tournament selection is applied, i.e. two random numbers corresponding to different mutation strategies are generated, and the strategy with a larger impact is selected. The PH test is checked every time all the components finish their generations.

4. EXPERIMENTAL SETUP AND RESULTS

The variables are divided into 5 subgroups, and the rule base contained 10 rules for all experiments. Each component algorithm optimized variables that describe 2 fuzzy rules, so each component algorithm had the possibility to solve a part of the classification problem by itself independent of the other components. However, the optimization of separate rules is interconnected through the fitness function, allowing different components to receive different solutions.

The algorithm was tested on 7 datasets taken from the UCI Machine learning repository [14]. The parameters of the datasets are presented in Table 2.

Table 2.

Name	Number of instances	Number of variables	Number of classes	Variables to optimize
Australian credit	690	14	2	570
Banknote	1372	4	2	170
Vertebral column 2c	310	6	2	250
Vertebral column 3c	310	6	3	250
German credit	1000	24	2	970
Liver	345	6	2	250
Seeds	210	7	2	290

The total computational resource was 500000 fitness function calculations. The fitness value was calculated as follows:

$$F_i = Error_i + 0.1 \cdot \frac{NUnignored_i}{NFSets}, \quad (5)$$

where $Error_i$ is the number of misclassified instances, *NUnignored* is the number of fuzzy sets in the resulting rule base which do not have the "Don't Care' condition, *NFSets* is the total number of fuzzy sets, equal to the number of variables multiplied by number of rules. The coefficient 0.1 is required to make the algorithm mainly focus on decreasing the error, as errors are always integer values. The 10-fold cross-validation was performed for every dataset to receive the training and test accuracy values.

For comparison, all 6 mutation schemes were tested separately to see if the proposed approach outperforms these basic algorithms. The number of individuals for all algorithms was set to 10*Number of variables to optimize. For the proposed approach, each component algorithm had 1/5 of the total number of individuals. Table 3 contains the accuracy comparison for the test set for all algorithms.

Table 3.

	Australian	Banknote	Column2c	Column3c	German	Liver	Seeds
1	0,848	0,966	**0,859**	0,788	0,707	0,662	0,877
2	0,857	0,994	0,807	0,769	0,705	0,621	0,887
3	0,872	**0,997**	0,814	0,775	0,706	0,659	0,902
4	0,850	0,991	0,814	**0,789**	0,715	0,671	0,921
5	0,856	0,983	0,823	0,759	0,705	0,650	0,872
6	**0,876**	0,994	0,842	0,739	0,701	**0,688**	0,892
DC	**0,876**	**0,997**	0,844	0,783	**0,720**	0,683	**0,932**

The winner-algorithms are shown in bold numbers; the proposed method is marked as DC. The algorithm with split variables was able to outperform any of its components for 2 datasets out of 7, and for the other 2 datasets the accuracy re-

ceived by the new algorithm is the same as for the best mutation scheme. Classical algorithms outperformed the proposed approach only on 3 datasets out of 7, and on 2 of them the difference in accuracy is around 0.5%.

The proposed DC approach whereby the impacts were calculated and the component algorithms were changed was also compared to other approaches: in particular, the first algorithm was the self-configuring GA (Fuzzy GA Adj), which solved the same problem of tuning membership functions, while the second algorithm (FHEA) was the specialized approach with fixed membership functions, described in [15]. The Fuzzy GA Adj used a binary representation of the membership function positions, and all the variables were encoded in one binary string. This resulted in a large amount of bits in the binary string, which significantly decreased the performance of the algorithm. FHEA algorithm, on the other hand, is not capable of tuning the membership function positions, which results in poor performance on many classification problems, because fixed membership functions may appear to be located at the wrong positions. The comparison results are presented in Table 4.

Table 4.

	Proposed DC method	Fuzzy GA Adj	FHEA [15]
Australian	**0,876**	0,852	0,857
Banknote	**0,997**	0,995	0,967
Column 2c	**0,844**	0,807	0,815
Column 3c	0,783	**0,803**	0,779
German	0,720	0,709	**0,749**
Liver	0,683	0,629	**0,696**
Seeds	**0,932**	0,917	0,909

As can be seen from Table 5, the proposed approach using differential evolution significantly outperformed the GA-based approach for 6 datasets out of 7, although they had the same computational resource and solved the same optimization problem. As for the second approach with fixed membership functions and various heuristics, it was capable of receiving better results only for 2 problems out of 7.

4. CONCLUSION

In this paper, an evolutionary fuzzy classification algorithm was presented, which uses a flexible representation of fuzzy terms with two sigmoid functions, and a differential evolution approach to tune the positions and shape of fuzzy terms. The DE algorithm also splits the optimization problem into 5 sub-problems, optimizing each pair of fuzzy rules separately. Additionally, the impacts of each used mutation strategy are calculated, allowing a better type of mutation to be chosen during the algorithm run. The resulting approach was tested on several com-

plex classification problems and has shown superior results compared to other algorithms, including a self-configuring GA and a specialized approach for fuzzy rule base generation. Further research in this direction includes automatic identification of the most suitable class number for every generated rule and the application of the presented DE algorithm to other problems, including industrial problems, where these methods have found a variety of applications [16, 17].

ACKNOWLEDGEMENT

The research is performed with the support of the Ministry of Education and Science of the Russian Federation within State Assignment project № 2.1680.2017/ПЧ.

REFERENCES

[1] Cordon O., Herrera F., Hoffmann F., Magdalena L. Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases. *Advances in Fuzzy Systems: Applications and Theory*. World Scientific, 2001.

[2] Herrera F., Magdalena L. Genetic Fuzzy Systems: A Tutorial. *Tatra Mountains Mathematical Publications*, vol. 13, 1997, p. 93-121.

[3] Fazzolari M., Alcalá R., Nojima Y., Ishibuchi H., Herrera F. A Review of the Application of Multi-Objective Evolutionary Fuzzy Systems: Current Status and Further Directions. *IEEE Transactions on Fuzzy Systems*, 21:1, 2013, p. 45-65.

[4] Alcala R., Alcala-Fernandez J., Herrera F., Otero J. Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation. *International Journal of Approximate Reasoning*, 44, 2007, p. 45–64.

[5] Berlanga F. J., Rivera A. J., del Jesus M. J., Herrera F., GP-COACH: Genetic programming-based learning of compact and accurate fuzzy rulebased classification systems for high-dimensional problems. *Inf. Sci.*, **8** (vol. 180), 2010, pp. 1183–1200.

[6] Cano J. R., Herrera F., Lozano M., Evolutionary Stratified Training Set Selection for Extracting Classification Rules with trade off Precision-Interpretability. *Data & Knowledge Engineering archive*, **1** (vol. 60), Jan. 2007, pp. 90-108.

[7] Ishibuchi H., Nojima Y. Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning*, 2007.

[8] Ishibuchi H. et al.: Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Trans. on Fuzzy Systems*, 1995.

[9] Sabar N. R., Abawajy J., Yearwood J., Heterogeneous Cooperative Co-evolution Memetic Differential Evolution Algorithms for Big Data Optimisation Problems. *IEEE Transactions on Evolutionary Computation*, **2** (vol. 21), 2017, pp. 315 – 327. – 2017.

[10] Potter M. A., De Jong K. A., Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary computation*, 8(1), 2000, pp.1–29.

[11] Mei Y., Omidvar M. N., Li X., Yao X. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software*, 42(2):13, 2016.

[12] Das S., Suganthan P.N., Differential Evolution: A Survey of the State-of-the-Art, IEEE Transactions on Evolutionary Computation, **1** (vo. 15), 2011, pp. 4-31.

[13] Price K.V., How Symmetry Constrains Evolutionary Optimizers, Black Box Differential Evolution – A Case Study. *IEEE Congress on Evolutionary computation (CEC) 2017*, Donostia - San Sebastián, Spain, June 5-8, 2017.

[14] Asuncion A., Newman D. *UCI machine learning repository*. University of California, Irvine, School of Information and Computer Sciences, 2007, URL: http://www.ics.uci.edu/~mlearn/MLRepository.html

[15] Stanovov V., Semenkin E., Semenkina O., Self-Configuring Hybrid Evolutionary Algorithm for Fuzzy Imbalanced Classification with Adaptive Instance Selection. *Journal of Artificial Intelligence and Soft Computing Research* 6(3), 2016, pp. 173-188.

[16] Salkoski, R., I. Chorbev. Design Optimization of Power Objects Based on Constrained Non-Linear Minimization, Generic Algorithms, Particle Swarm Optimization Algorithms and Differential Evolution Algorithms. *International Journal on Information Technologies and Security*, **3** (vol. 6), 2014, pp. 21-30.

[17] Nikolova-Poceva, S., A. Iliev. Hybrid Fuzzy Regression Model for Determining Specific Active Power Generation Characteristic of Hydro Power Plants. *International Journal on Information Technologies and Security*, **1** (vol. 8), 2016, pp. 55-68.

Information about the authors:

Vladimir Stanovov has a Ph.D. in Engineering Science, senior research fellow at the Reshetnev Siberian State University of Science and Technology. Research interests include evolutionary algorithms and fuzzy systems.

Shakhnaz Akhmedova has a Ph.D. in Engineering Science, associate professor at the Reshetnev Siberian State University of Science and Technology. Research interests include particle swarm optimization and neural networks.

Eugene Semenkin is a Doctor of Engineering Science, Professor at the Reshetnev Siberian State University of Science and Technology. Research interests are complex systems modelling and design, computational intelligence, evolutionary algorithms and data mining.

Manuscript (final version) received on 24 March 2018