NiaClass: Building Rule-Based Classification Models Using Nature-Inspired Algorithms

Luka Pečnik, Iztok Fister, and Iztok Fister Jr.(✉)

Faculty of Electrical Engineering and Computer Science, Koroška ul. 46, 2000 Maribor, Slovenia
luka.pecnik@student.um.si, {iztok.fister,iztok.fister1}@um.si

Abstract. Searching for a set of rules, with which the knowledge hidden in data is extracted, can also be applied for multi-class classification. In line with this, a collection of nature-inspired algorithms are selected for determining the set of rules capable of classifying the samples into three or more classes. This set is encoded into representation of individuals and undergoes acting the variation operators. The results of various nature-inspired algorithms, obtained after their application on more UCI classification databases, are compared with each other, and revealed that some of them can be potential candidates for real-world applications.

Keywords: Nature-inspired algorithms · Machine learning · AutoML · Classification pipeline

1 Introduction

A multi-class classification is a problem of classifying the instances (samples) into one of three or more classes. This is an extension of more known binary classification, where samples are classified into two classes only. The rule-based classification models identify and utilize a set of rules in the form of IF-THEN rules that, together, represent the knowledge captured by the algorithm. In the multi-class classification, the specific class to which a definite sample belongs, is determined by the consequent of the rule. Although there are a lot of traditional methods for solving this problem (e.g., Bayesian networks) [1], this study proposes the application of nature-inspired algorithms to this hard nut to crack. The main advantages of these algorithms over the traditional methods are that they are typically less computationally expensive [13]. Most of the statistical machine learning methods are described as complex mathematical functions, and they are rather incomprehensible and opaque to humans [15]. On the other hand, nature-inspired algorithms can ensure accuracy and comprehensibility – a fact that is important in Explainable Artificial Intelligence [2].

Studying the approaches based on the population-based metaheuristics for discovering the classification rules is still very popular in the research community [14]. Although the first approaches appeared more than 15 years ago with

Y. Tan and Y. Shi (Eds.): ICSI 2021, LNCS 12690, pp. 381–390, 2021.
https://doi.org/10.1007/978-3-030-78811-7_36

the rise of big data, exploring and developing new approaches is still very fruitful. One of the first algorithms for classification rule discovery was proposed by Parpinelli et al. and was called Ant-Miner [16]. Ant-miner was based on Ant Colony Optimization (ACO). On the contrary, the development of algorithms that were based on other inspirations of nature-inspired algorithms was also at full pace. The first algorithm based on Particle Swarm Optimization was proposed by Sousa back in 2004 [17]. A plethora of newer nature-inspired algorithms, in combination with some conventional machine learning methods, are also used for solving classification tasks [3,12]. It is worth mentioning that there is a very thin line between algorithms for Numerical Association Rule Mining (NARM) [9] and algorithms for discovering the classification rules. In other words, NARM algorithms with smaller modifications can also be applied for discovering classification rules.

This paper presents an extension of an approach that was published last year at the ICCS conference [8], where a new method for discovering classification rules based was proposed on the Firefly Algorithm (FA). This method was tailored only for solving binary classification problems. In this paper we step further, and extend this method for also coping with multi-class classification problems.

The main contributions of this paper are the following:

- extension of binary classification to multi-class classification,
- inspired by the recent NARM algorithm [10], new measures are modeled in the fitness function,
- extensive experimental comparison among different nature-inspired algorithms are conducted.

The structure in the remainder of the paper is as follows. Section 2 discusses the basic information needed for understanding the subjects that follow. In Sect. 3, the design of the proposed algorithm for three-class classification is explained in detail. The experiments and results are the subjects of Sect. 4, while the paper concludes with Sect. 5, that also outlines the directions for the future development.

2 Basic Information

This section is devoted to discussing information potential readers need to understand the subjects treated in the remainder of the paper. In line with this, the basics are discussed about the concept of nature-inspired algorithms. The section is concluded with a description of the NiaClass concept, within which the algorithm began to be developed.

2.1 Nature-Inspired Algorithms

Primarily, the nature-inspired population-based algorithms comprise two families: (1) Evolutionary Algorithms (EA) [7], and (2) Swarm Intelligence (SI) based

algorithms [4]. The former are inspired by the Darwinian struggle for existence, where, similarly as in nature also in simulated evolution, only the fittest individuals (i.e., solutions) can survive in the hard environmental conditions (simulated by the fitness function). One of the more prominent members of this class is undoubtedly Differential Evolution (DE) [18]. The latter mimics the behavior of social living insects and animals revealing some kinds of optimization process. Obviously, the social living swarm of birds present good examples of these behaviors (inspiration for the Particle Swarm Optimization (PSO) [11]) and swarm of insects, precisely fireflies (inspiration for the Firefly Algorithm (FA) [19]).

Indeed, this paper focuses on three nature-inspired algorithms for solving rule-based classification models, i.e., DE, PSO, and FA, operating using real-valued representation. The DE is well known evolutionary algorithm especially appropriate for continuous optimization [5]. The PSO and FA are SI-based algorithms suitable for solving problems arising in almost all application domains. Among the mentioned algorithms, the FA was successfully applied also for classification problems [8].

2.2 NiaClass - A Classification Platform in Python

The basic concept of the NiaClass classifier is to use stochastic nature-inspired population-based algorithms implemented in Python programming language in order to find the optimal set of classification rules for a given datasets [8]. The full source code of NiaClass software is available on Github[1]. Actually, the authors plan to extend a collection of those algorithms in the future as well.

3 Proposed Method

Although the proposed method enables multi-class classification of an arbitrary number of classes, here, we are focused on three-class classification problem due to its simplicity. The problem can be defined formally as follows: Let us assume a 3-class classification problem on database Db consisting of M features is given, where each feature attribute is a tuple:

$$Attr_k = \langle type, off, cont, [\mathbf{D}|\mathbf{c}] \rangle, \quad \text{for } k = 1, \ldots, M, \tag{1}$$

and

$$\begin{aligned} type &- \text{set of attribute types, i.e. } type \in \{num, cat\}, \\ off &- \text{attribute offset within solution vector } \mathbf{x}_i, \\ cont &- \text{control random variable,} \\ \mathbf{D} &- \text{vector of numeric 3-class attribute domains,} \\ \mathbf{c} &- \text{vector of discrete 3-class attributes.} \end{aligned}$$

Thus, the vector of the three-class attribute domain is defined as $\mathbf{D} = (D_1, D_2, D_3)$, where $D_l = [lb, ub]$ for $l = 1, \ldots, 3$, and the vector of discrete

[1] https://github.com/lukapecnik/NiaClass.

3-class attributes as $\mathbf{c} = (c_1, c_2, c_3)$ with elements $c_l \in AttrSet_k$ for $l = 1, \ldots, 3$. Indeed, attributes consist of rules for classifying the sample into the corresponding class. Then, the solution vector of the nature-inspired algorithm is represented as follows:

$$\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,L}), \quad \text{for } i = 1, \ldots, N, \tag{2}$$

where N is the population size, and L denotes the number of elements, calculated as:

$$L = (1 + 2 \cdot class \cdot \#numeric\_attr) + (1 + class \cdot \#category\_attr) + 1, \tag{3}$$

where $class$ denotes the number of classes (in our case, $class = 3$), $\#numeric\_attr$ the number of numerical features, and $\#category\_attr$ is the number of categorical features. Interestingly, the numerical attributes are represented as a tuple $\langle x_{Attr_k.off}, \ldots, x_{Attr_k.off+6} \rangle$ consisting of seven, and the categorical attributes as a tuple $\langle x_{Attr_k.off}, \ldots, x_{Attr_k.off+3} \rangle$ consisting of four elements.

The variable $cont \in \{0, 1\}$ of the k-th attribute is mapped from the search space to the problem space as follows:

$$Attr_k.cont = \begin{cases} 0, & \text{if } x_{Attr_k.off} \leq threshold, \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

However, when the variable $cont = 0$, the feature of this class is omitted from the classification.

The k-th element representing the numeric 3-class attribute domains is mapped from the solution vector as follows:

$$\begin{aligned} Attr_k^{(num)}.D_l.lb &= \frac{ub_l - lb_l}{UB_l - LB_l} \cdot lb_l + lb_l. \\ Attr_k^{(num)}.D_l.ub &= \frac{ub_l - lb_l}{UB_l - LB_l} \cdot ub_l + lb_l, \end{aligned} \quad \text{for } l = 1, \ldots, 3, \tag{5}$$

where $ub_l = Attr_k.off.x_{2(l-1)+2}$ and $lb_l = Attr_k.off.x_{2(l-1)+1}$. Obviously, the relation $ub_k > lb_k$ must be ensured by the equation.

Similarly, the k-th element representing the categorical attribute is mapped from the solution vector as follows:

$$Attr_k^{(cat)}.c_l = \lfloor Attr_k.off.x_{2(l-1)+1} * |Attr_k^{(cat)}.c_l| \rfloor, \tag{6}$$

where the term $|Attr_k^{(cat)}.c_l|$ denotes the number of the attributes describing the feature.

Finally, the attributes are composed into IF-THEN rules as follows:

$$P(Attr_1, S_1) \wedge \ldots \wedge P(Attr_k, S_k), \wedge \ldots \wedge (Attr_M, S_M) \Rightarrow class, \tag{7}$$

where $P(.)$ denotes a predicate returning either $true$ or $false$, $Attr_k$ the attribute of the k-th feature and S_k the corresponding value of the feature in the sample,

and $class = \{class_1, class_2, class_3\}$ is the three-class classification set. If the k-th attribute is categorical, the predicate verifies if the discrete value of the attribute is equal to one of the three predicted values of the $Attr_k^{(cat)}$, while, in the case of a numerical attribute, it verifies if the sample value is higher than the lower bound and lower than the higher bound of the numeric attribute $Attr_k^{(num)}$ in each of the observed three-classes. This means that there are three classification rules decoded per one solution vector.

The mapping from the search to the problem space is illustrated in Fig. 1, where the values of variables *cont* are represented in black, *threshold* in red, while all the other values are painted in the corresponding class colors, i.e., a set of rules for $class_1$ is represented in green, rules for $class_2$ are represented in blue and rules for $class_3$ in brown.

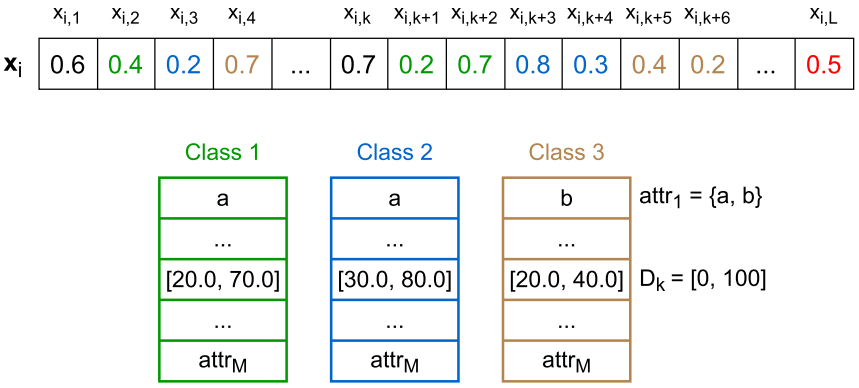

Fig. 1. Search space to problem space mapping. (Color figure online)

The fitness function in the optimization process of the proposed method is calculated after applying the mapped solution vector $Attr$ to the classification problem on problem database Db, and it is composed of three terms as follows:

$$f(Attr, Db) = -score() + 0.5 \cdot ldc - 0.5 \cdot oc, \tag{8}$$

where the first term is *score* function, that can be any classification metric such as accuracy, precision, F1-score or Cohen's κ, in other words:

$$score = \{accuracy, precision, \text{F1-}score, \kappa\}.$$

The value ldc denotes the length differences' coefficient, and oc denotes the coefficient of intervals' overlapping. Both coefficients were multiplied by the weight of 0.5, as we expect that the metric should still be the most important factor of the final fitness.

The pseudo-code of the proposed method is presented in the Algorithm 1, from which it can be see that the algorithm suits the general form of the nature-

Algorithm 1. NiaClass for building rule-based classification models

1: $\langle \mathbf{D}, \mathbf{c} \rangle \leftarrow$ preprocess characteristics of numeric/discrete features (Db)
2: $population \leftarrow$ initialize real valued vectors $\mathbf{x}_i$ randomly
3: $\langle best\_fitness, Attr\_best \rangle \leftarrow$ find the best ($population$)
4: **while** not termination condition met **do**
5: **for each** $\mathbf{x}_i \in population$ **do**
6: $\mathbf{x}_{trial} \leftarrow$ modify individual using variation operators ($\mathbf{x}_i$)
7: $Attr \leftarrow$ decode ($\mathbf{x}_{trial}$)
8: $fitness = f(Attr, Db)$)
9: **if** $fitness$ is better than 'decode ($\mathbf{x}_i$)' **then**
10: $\mathbf{x}_i \leftarrow \mathbf{x}_{trial}$ ▷ Replace the worse individual
11: **end if**
12: **if** $fitness$ is better than $best\_fitness$ **then**
13: $best\_fitness \leftarrow fitness$
14: $Attr\_best \leftarrow Attr$
15: **end if**
16: **end for**
17: **end while**
18: Return best set of rules $Attr\_best$

inspired algorithms. After initialization, evaluation of initial population, and finding the best solution (line 1 and 3), each individual undergoes acting the variation operators (line 6) by generation of trial solution. The rule-based classification model is constructed by appropriate genotype-phenotype mapping as described in Eq. 7 (line 7) and evaluated according to Eq. 8 (line 8). If the quality of constructed pipeline from the trial is better than the quality of individual, the individual is replaced with the trial (lines 9–11). In similar way, the best solution with corresponding fitness is determined in lines 12–15.

4 Experiments and Results

The goal of our experimental work was to show that the proposed NiaClass classifier is suitable for both, binary and multi-class classification problems, and that by adding certain information obtained through preprocessing of the fitness function, we can further improve the final results. In line with this, extensive experimental work was conducted, where all experiments were performed on an HP ProDesk 400 G6 MT computer running Microsoft Windows 10, an Intel (R) Core (TM) i7-9700 CPU @ 3.00 GHz processor, and 8 GB of installed physical memory.

A list of datasets from the UCI Machine Learning Repository [6] used in the experiments and their characteristics are shown in Table 1. The missing values of the numerical attributes were replaced by the mean of the feature, while the missing categorical values were replaced by the mode of the feature in the case of the Cylinder Bands dataset.

Table 1. Datasets used in the experiments.

Dataset	Type of attributes	Instances	Features	Missing data
Haberman	Integer	306	3	No
Ecoli	Real	336	8	No
Yeast	Real	1484	8	No
Cylinder Bands	Real, integer, categorical	512	39	Yes

When fitting the NiaClass classifier, we used a population size of 90 individuals using 5000 evaluations of the fitness function. In the optimization process, we used the Differential Evolution (DE), Firefly Algorithm (FA) and Particle Swarm Optimization (PSO). Settings of their parameters are shown in Table 2.

Table 2. Parameter settings of the optimization algorithms in the experiment.

Algorithm	Parameters
DE	$F = 1,\ \ CR = 0.8$
FA	$alpha = 0.5,\ \ betamin = 0.2,\ \ gamma = 1.0$
PSO	$C1 = 2.0,\ \ C2 = 2.0,\ \ w = 0.7,\ \ vMin = -1.5,\ \ vMax = 1.5$

The final results were obtained after 25 independent runs of each algorithm for each dataset using: (1) The basic fitness function that does not take into account the information about the numerical intervals, and (2) The extended fitness function capable of using the information from the dataset preprocessing step. Before each run, the dataset was divided into training and test sets in a ratio of 0.2, and after fitting, the classification was performed using the test set. The results are shown in Table 3.

Interestingly, nature-inspired algorithms using the extended fitness function failed to dominate their counterparts using the basic fitness function on the observed datasets. Although we expected that using the extended fitness function would improve the results significantly, this claim holds only for the results obtained on the Haberman dataset. Indeed, the best results were achieved by the PSO algorithm when comparing in terms of the optimization algorithms.

4.1 Discussion

As can be seen from the results of our experiments, we were unable to improve the results obtained by the nature-inspired algorithms using the basic fitness function compared with those achieved by using the the extended fitness function. The cause for this was most likely due to the fact that the fitness function for determining the class of an individual from the training set was very simple,

Table 3. Detailed results of classification according to accuracy.

Dataset	Fitness	Algorithm	Min	Max	Mean	Median	Std
Haberman	Basic	DE	0.6452	0.8226	0.7323	0.7258	0.0434
		FA	0.6290	0.8065	0.7381	0.7419	0.0462
		PSO	0.6290	0.7903	0.7335	0.7419	0.0422
	Upgraded	DE	0.6613	0.8387	0.7310	0.7419	0.0451
		FA	0.6613	0.8065	**0.7400**	0.7419	0.0346
		PSO	0.6290	**0.8710**	0.7245	0.7258	0.0605
Ecoli	Basic	DE	0.5441	0.7794	0.6476	0.6618	0.0803
		FA	0.3235	0.6471	0.5347	0.5441	0.0765
		PSO	0.5735	**0.8088**	**0.7006**	0.7059	0.0750
	Upgraded	DE	0.4412	0.7941	0.6335	0.6471	0.0717
		FA	0.3971	0.6618	0.5376	0.5441	0.0800
		PSO	0.3088	**0.8088**	0.6429	0.6471	0.1168
Cylinder Bands	Basic	DE	0.6019	0.7222	0.6778	0.6852	0.0334
		FA	0.5833	0.7685	0.6607	0.6574	0.0416
		PSO	0.6111	**0.7870**	**0.6985**	0.7037	0.0437
	Upgraded	DE	0.5370	0.7593	0.6419	0.6389	0.0483
		FA	0.5278	0.7037	0.6274	0.6204	0.0502
		PSO	0.5833	0.7130	0.6563	0.6574	0.0366
Yeast	Basic	DE	0.3367	0.4646	0.3954	0.3939	0.0335
		FA	0.2593	0.4276	0.3390	0.3401	0.0402
		PSO	0.3333	**0.4983**	**0.4168**	0.4141	0.0389
	Upgraded	DE	0.2626	0.3670	0.3244	0.3266	0.0260
		FA	0.2290	0.4074	0.3088	0.3131	0.0404
		PSO	0.3131	0.4478	0.3636	0.3603	0.0363

and, therefore, additional coefficients considered in the extended fitness function did not contain enough information about the quality of the rule based classifiers, or even the selected weights were inappropriate. It was also difficult to determine the parameter setting, and the suitability of the selected optimization algorithms for the experimental datasets is also questionable.

Indeed, the purpose of the study was to show that the problem of rule-based classification models could be solved successfully using nature-inspired algorithms beside the traditional methods. This preliminary work has proven this hypothesis. In line with this, the FA has exposed the best results among the algorithms in test, while the DE algorithm did not turn out too well. Therefore, new experiments must be conducted in order to show the best characteristics of this kind of algorithms necessary for achieving better results in solving the problem.

5 Conclusion

The rule-based multi-class classification is very interesting area of machine learning that was typically solved using traditional methods, like Bayesian networks. In this paper, the classification problem was solved using the nature-inspired algorithms, where each set of rules was encoded into a representation of individuals and indicated the property of a specific class. Three different nature-inspired algorithms were employed in this study, i.e., DE, PSO, and FA. The first algorithm belongs to a class of EAs, while the other two to the SI-based algorithms' family. Normally, all three algorithms operate with real-valued representation, and are obviously included into the NiaClass repository.

The proposed algorithms were applied to four UCI Machine Learning Repository datasets. The results were observed according to two fitness functions, i.e., the basic and extended, where the latter also explores the preprocessing information. Although the results of experiments showed that using the extended measure by the nature-inspired algorithms did not improve the results of the classification significantly, the study revealed the potential of the proposed algorithms especially in the sense of their complexity.

Although the authors are aware that the preliminary results are slightly worse than the results achieved by the traditional algorithms, they found many potential directions for improving the results in the future. For instance, better results could be achieved by finding the optimal algorithms, their parameter settings and fitness function weights using a separate optimization process. We could also try to improve the fitness function of the NiaClass classifier's fitting method by mining other potentially useful information from datasets that we could consider in the fitness value, or even by implementing a separate procedure that would literally, in some way, build a fitness function for each data set separately.

Acknowledgement. The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0041 - Digital twin).

References

1. Aly, M.: Survey on multiclass classification methods. Technical report, Caltech (2005)
2. Arrieta, A.B., et al.: Explainable Artificial Intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion **58**, 82–115 (2020)
3. Banchhor, C., Srinivasu, N.: Integrating cuckoo search-grey wolf optimization and correlative naive Bayes classifier with map reduce model for big data classification. Data Knowl. Eng. **127**, 101788 (2020)
4. Blum, C., Merkle, D.: Swarm Intelligence: Introduction and Applications, 1st edn. Springer Publishing Company Incorporate, Heidelberg (2008). https://doi.org/10.1007/978-3-540-74089-6
5. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)

6. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
7. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, 2nd edn. Springer Publishing Company Incorporated, Heidelberg (2015). https://doi.org/10.1007/978-3-662-44874-8
8. Fister, I., Fister, I., Fister, D., Vrbančič, G., Podgorelec, V.: On the potential of the nature-inspired algorithms for pure binary classification. In: Krzhizhanovskaya, V.V., et al. (eds.) ICCS 2020, Part V. LNCS, vol. 12141, pp. 18–28. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50426-7_2
9. Fister, I. Jr., Fister, I.: A brief overview of swarm intelligence-based algorithms for numerical association rule mining. arXiv preprint arXiv:2010.15524 (2020)
10. Fister Jr., I., Podgorelec, V., Fister, I.: Improved nature-inspired algorithms for numeric association rule mining. In: Vasant, P., Zelinka, I., Weber, G.-W. (eds.) ICO 2020. AISC, vol. 1324, pp. 187–195. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68154-8_19
11. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
12. Kulhari, A., Pandey, A., Pal, R., Mittal, H.: Unsupervised data classification using modified cuckoo search method. In 2016 Ninth International Conference on Contemporary Computing (IC3), pp. 1–5. IEEE (2016)
13. Liu, B., Abbass, H.A., McKay, B.: Density-based heuristic for rule discovery with ant-miner. In: The 6th Australia-Japan Joint Workshop on Intelligent and Evolutionary System, vol. 184. Citeseer (2002)
14. Martens, D., Baesens, B., Fawcett, T.: Editorial survey: swarm intelligence for data mining. Mach. Learn. **82**(1), 1–42 (2011). https://doi.org/10.1007/s10994-010-5216-5
15. Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Trans. Evol. Comput. **11**(5), 651–665 (2007)
16. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. **6**(4), 321–332 (2002)
17. Sousa, T., Silva, A., Neves, A.: Particle swarm based data mining algorithms for classification tasks. Parallel Comput. **30**(5–6), 767–783 (2004)
18. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997). https://doi.org/10.1023/A:1008202821328
19. Yang, X.-S.: Nature-Inspired Optimization Algorithms. Academic Press, Cambridge, MA, United States (2020)