Influence of the feature threshold in Evolutionary Feature Selection

Uroš Mlakar

University of Maribor, Faculty of Electrical Engineering and Computer Science
E-mail: uros.mlakar@um.si

Abstract

Evolutionary feature selection is a process employed in various fields to identify the most informative features for predictive modeling. The threshold parameter plays a crucial role in determining the selection pressure on features during the evolutionary process. This paper investigates the influence of the feature selection threshold in five popular evolutionary and swarm intelligence algorithms. The experiments are conducted on diverse UC Irvine datasets, with a detailed analysis on how varying threshold values impact the feature selection outcomes, including predictive accuracy of the trained model, the final feature subset size, and convergence properties of the used algorithms. The obtained results are statistically evaluated and suggest using a different threshold value, then the one commonly used in the literature.

1 Introduction

Feature selection is a important process in machine learning and data analysis domains, aimed at identifying a subset of relevant features from a larger set of variables to improve model performance, reduce computational complexity, and enhance the interpretability of the problem being solved [1, 8]. Many Evolutionary (EA)[4] and Swarm Intelligence (SI) [2] algorithms have been successfully applied for automating the process of feature selection by mimicking natural selection principles in various animal species [11, 5]. While EAs offer a promising approach, the selection pressure of the evolutionary process heavily relies on various parameters, among which the threshold parameter plays a pivotal role. The threshold parameter defines the criterion for selecting features during the evolutionary process, influencing the convergence behavior, solution quality, and computational efficiency of the feature selection algorithm.

In this paper, we investigate the significance of the threshold parameter in evolutionary feature selection methods. We explore its impact on the quality of the fitness function, and the final feature subset size. Additionally, we investigate how different threshold values affect the convergence characteristics of a specific evolutionary feature selection algorithm.

This paper is structured as follows. Section 2 presents the idea of evolutionary feature selection. Then in Section 3 the experiment design is detailed. Experimental results are discussed in Section 4, and the paper is concluded with future work directions in Section 5.

2 Evolutionary feature selection

Evolutionary feature selection is an application of an EA or SI algorithms for the problem of feature selection [11]. The selected EA or SI algorithm maintains a population of solutions $\mathbf{x}_i = \{x_{i,j}\}$ for $i = 1, \ldots, Np$; $j = 1, \ldots, D$, where Np denotes the population size, and D is the dimension of the problem being solved. In the case of feature selection D is usually set to the number of features in the dataset.

According to the problem for which the algorithm was designed, the solution representation is an important factor for the success of finding an optimal solution. In feature selection, the problem of a solution representation is straightforward, since the number of features in a dataset corresponds to the dimension of a solution vector $\mathbf{x}_i$. Two representations of a solution are commonly found in literature for the problem of feature selection: binary and real-coded [8]. Since the majority of EA and SI algorithms are designed to work with continuous problem domains, the real-coded representation is also mostly used for feature selection, where the presence of a feature is determined by mapping the element in the solution space to a predefined interval. Most frequently, the search space is defined within the interval $[0, 1]$, therefore a feature is selected if the value of an element $x_{i,j}$ is greater then a specified threshold, which is usually set to 0.5. This can be mathematically expressed as:

$$\mathbf{x}_s = \{x_{i,j} > threshhold\}. \tag{1}$$

The resulting vector $\mathbf{x}_s$ contains all the features, which are selected for the current solution $\mathbf{x}_i$. To evaluate the performance of the current selected feature subset $\mathbf{x}_s$, a classifier is trained on the selected training dataset using cross validation with these selected features. In the case of a wrapper-based approach, the classifiers k-Nearest Neighbor (KNN) is commonly used in literature. Therefore a fitness function for the feature selection problem can be defined as:

$$f(\mathbf{x}_i) = cross\_validation(classifier, \mathbf{x}_s) \tag{2}$$

When the algorithm run is over, the best performing feature subset is trained on the whole training set using

the selected classifier, and the final results are reported as the performance of the trained classifier on the testing set.

3 Experiment design

All experiments for this work were performed on a desktop computer with the following configuration: Intel(R) Core(TM) i9-10900KF CPU @ 3.70GHz, with 65 GB and operating system Linux Ubuntu 22.04 Jellyfish operating system. The software was written in the python programming language, utilizing the Niapy library [10] for the implementation of the EA and SI algorithms. The commonly used datasets in feature selection problems, listed in Table 1, were used for evaluating the impact of different selection thresholds [3]. Let us notice that the selected datasets exhibit different numbers of features (4-166), instances (178-1728) and classes (2-7). Each dataset in this study was split randomly into training and testing sets, with 80% of samples going to the training and 20% to the testing set, while also ensuring an equal distribution of classes.

Many EA and SI algorithms have been applied to different problems over the last couple of years, also including feature selection. Among the most common are Particle Swarm Optimization (PSO) [7], Bat Algorithm (BA) [12], Differential Evolution (DE) [9], Cuckoo Search (CS) [13] and Artificial Bee Colony (ABC) [6], which were also considered for our evaluation. The parameters of the mentioned algorithms used in the experiments were set as reported in their respective papers.

Because of the stochastic nature of EA and SI algorithms, each experiment run for a selected algorithm was repeated 30 times. To make the comparison fair, the termination condition for all algorithms was set to 100 generations, using a starting population size $Np = 30$. Due to its high computational efficiency, we chose the KNN classifier (N=5) in the experimental work.

Table 1: Datasets used during experimental work.

Dataset	#Features	#Instances	#Classes
balance_scale	4	625	3
car	6	1728	4
autompg	7	392	3
wine	13	178	3
segmentation	19	210	7
german	20	1000	2
ionosphere	34	351	3
vehicle	51	845	4
sonar	60	208	2
hill_valley	100	1212	2
musk1	166	476	2

4 Results

In this section, we present a comprehensive analysis of the results obtained by the application of different feature selection thresholds in several evolutionary and swarm intelligence algorithms. The following experiments were performed:

- influence of the feature selection threshold on the fitness function for a specific algorithm.
- influence of the feature selection threshold on the length of the obtained final feature subset for a specific algorithm.
- influence of the feature selection threshold on the convergence characteristics of the used algorithms.

All mentioned experiments are presented in detail in the remainder of this section. All experiments are reported in terms of the Friedman and Wilcox non-parametric statistical tests.

4.1 Fitness function analysis

The goal of the first experiment was to analyze how a particular feature selection threshold value influences the quality of the selected feature subsets, based on the reported fitness values of an EA or SI algorithm. Four different threshold values were employed in this experiment ($thresholds = \{0.4, 0.5, 0.6, 0.7\}$) for each of the observed algorithms. Since this is a pilot study, the thresholds, were selected empirically. Therefore each algorithm was executed 30 times for each dataset. The averages of these 30 runs over all datasets are reported in terms of average fitness values and standard deviation in Table 2. Values marked in bold face indicate the best algorithm for a specific threshold value, while the underlined value denotes the best threshold for a specific algorithm. It seems that the best performing algorithm among the selected is the DE regardless of the threshold value. Considering the threshold value, the best results were obtained with the threshold value of 0.7 on all considered algorithms, except for the PSO.

The results were also statistically evaluated using the Friedman and Wilcoxon tests. In essence, the statistical tests comparison involved four classifiers (the results obtained using different threshold values) based on 275 elements. Each classifiers size was derived from the product $5 \times 5 \times 11$, where the first number represents the number of algorithms considered, the second denotes the number of statistical measures taken into account (i.e., mean, standard deviation, minimum, maximum, and median), and the third is the number of used datasets in the study. The results of the Friedman tests are displayed graphically in Fig. 1. The results of the Nemenyi post-hoc test are represented as critical difference intervals, where the results of two methods are statistically significant if their critical difference intervals do not overlap. The results of the Wilcoxon tests are gathered in Table 3. Let us notice that a higher Friedman rank denotes better performance.

According to the results, we can infer that a higher threshold value is preferred, at least in the pool of the selected comparing algorithms. This fact is also supported by the statistical tests. The threshold value of 0.7 was ranked highest by the Friedman test, and a statistically significant difference is found when compared to other threshold values.

Algorithm \ Threshold	0.4	0.5	0.6	0.7
PSO	0.1744±0.0069	0.1737±0.0085	0.1725±0.0089	0.1726±0.0107
BA	0.1759±0.0077	0.1742±0.0073	0.172±0.0075	0.1716±0.0096
DE	**0.1676±0.0031**	**0.1669±0.0032**	**0.1663±0.0029**	**0.166±0.0032**
CS	0.1733±0.0033	0.1715±0.0033	0.1702±0.0036	0.1682±0.0035
ABC	0.1837±0.0058	0.1812±0.0047	0.179±0.0056	0.1775±0.0062

Table 2: Numerical results for the fitness function study.

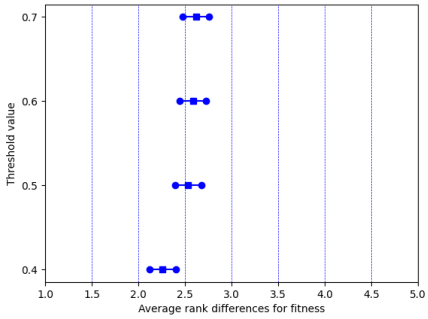

Figure 1: Graphical representation of Friedman critical distances for the fitness study on using different feature selection threshold values.

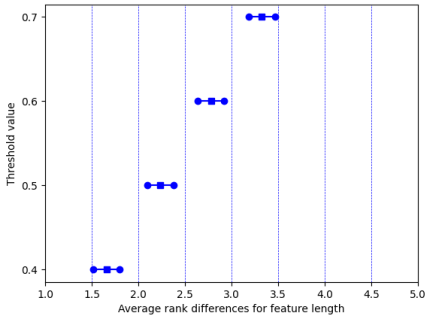

Figure 2: Graphical representation of Friedman critical distances for the feature subset size study on using different feature selection threshold values.

thr	0.4	0.5	0.6	0.7
0.4	∞	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$
0.5	-	∞	0.32	0.15
0.6	-	-	∞	0.26
0.7	-	-	-	∞

Table 3: Results of the Wilcoxon test for the fitness study on using different feature selection thresholds values.

4.2 Subset feature length analysis

The goal of the second experiment was to test how a particular feature selection threshold value influences the final size of the selected feature subsets. The same threshold values were employed in this experiments for each of the observed algorithms. As in the previous experiment, the results are presented as averages of 30 runs over all datasets and are reported in terms of average feature size values and standard deviations in Table 4.

The results in Table 4 indicate that the lowest feature subsets are obtained using a higher threshold value for all considered algorithms. This is also supported by the statistical tests presented in Fig. 2 and Table 5. A higher threshold value translates to a higher selection pressure in the algorithm's search space, so these results are somewhat expected.

4.3 Convergence analysis

The goal of the last experiment was to analyze the convergence properties of all selected algorithms considering different feature selection threshold values. The convergence graphs are presented in Fig. 3. Each subplot shows the converge plots (using different threshold values) for a selected algorithm on a specific dataset. The converge plots show that higher threshold values tend to converge faster, and while also achieving better results.

5 Conclusion

This paper investigates the importance of the feature selection threshold in EA and SI algorithms when applied in classification problems. The proposed methodology was tested using five commonly used algorithms in the literature on popular feature selection datasets. The impact of the feature selection threshold was analyzed by considering the results of fitness values of the objective function and the final feature subset sizes. The obtained results were also statistically evaluated. Our findings show that there is a statistically significant deviation in the results according to the commonly chosen value of 0.5, therefore a large scale study is needed to further analyze these conclusions.

Future work will encompass a large scale study on more recent state-of-the-art algorithms, while also considering more experimental datasets. Additionally different fitness functions will be considered, where the feature subset size is included, since an obvious connection among the classification accuracy and feature subset size was detected in the experiments.

Acknowledgments

This work was supported by the Slovenian Research and Innovation Agency (Programme No. P2-0041)

References

[1] Qasem Al-Tashi, Said Jadid Abdul Kadir, Helmi Md Rais, Seyedali Mirjalili, and Hitham Alhussian. Binary opti-

Algorithm \ Threshold	0.4	0.5	0.6	0.7
PSO	21.3939±2.4844	19.8182±2.4151	18.8879±2.473	17.7818±2.4605
BA	23.4273±2.4921	20.097±2.4677	17.1758±2.3617	13.8848±2.5165
DE	**20.2879±1.9437**	**18.8485±1.9614**	17.3303±2.0085	15.7576±2.0964
CS	22.1818±2.3522	19.4939±2.1419	16.6303±2.0509	13.903±1.9636
ABC	24.0606±2.0533	20.3394±2.0127	**16.4242±1.923**	**12.9182±1.7245**

Table 4: Numerical results for the feature subset size study.

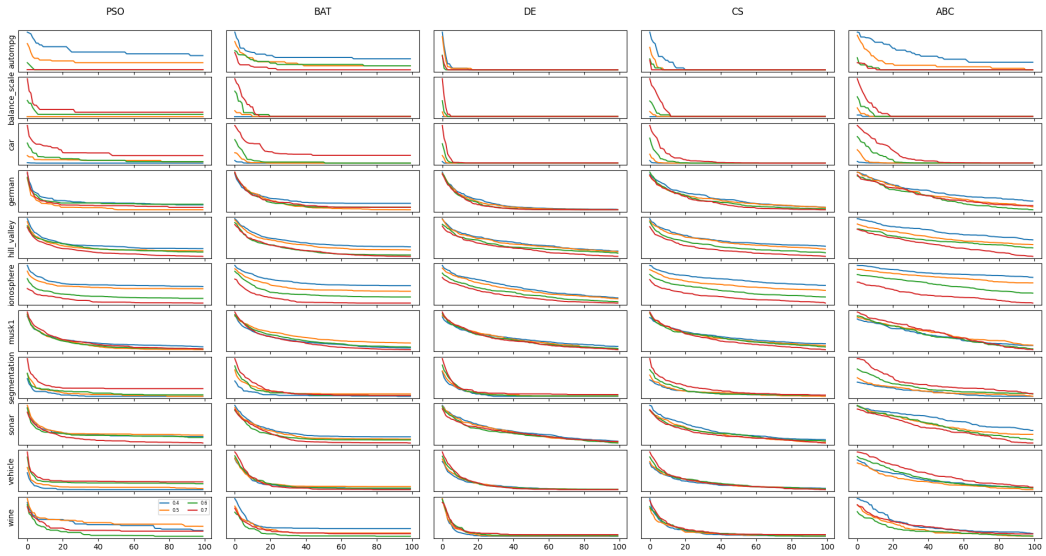

Figure 3: Convergence analysis of each algorithm when considering all feature selection threshold values.

thr	0.4	0.5	0.6	0.7
0.4	∞	$\ll 0.05$	$\ll 0.05$	$\ll 0.05$
0.5	-	∞	$\ll 0.05$	$\ll 0.05$
0.6	-	-	∞	$\ll 0.05$
0.7	-	-	-	∞

Table 5: Results of the Wilcoxon test for the feature subset size study on using different feature selection thresholds values.

mization using hybrid grey wolf optimization for feature selection. *Ieee Access*, 7:39496–39508, 2019.

[2] Christian Blum and Daniel Merkle. *Swarm intelligence: introduction and applications*. Springer Science & Business Media, 2008.

[3] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[4] Agoston E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, 2003.

[5] Miguel García-Torres, Roberto Ruiz, and Federico Divina. Evolutionary feature selection on high dimensional data using a search space reduction approach. *Engineering Applications of Artificial Intelligence*, 117:105556, 2023.

[6] Dervis Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[7] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[8] Uroš Mlakar and Iztok Fister. Impact of solution representation in nature-inspired algorithms for feature selection. *IEEE Access*, 8:134728–134742, 2020.

[9] Rainer Storn and Kenneth V. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[10] Grega Vrbančič, Lucija Brezočnik, Uroš Mlakar, Dušan Fister, and Iztok Fister Jr. NiaPy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software*, 3, 2018.

[11] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on evolutionary computation*, 20(4):606–626, 2015.

[12] Xin-She Yang and Amir Hossein Gandomi. Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, 29(5):464–483, 2012.

[13] Xin-She Yang and Amir Hossein Gandomi. Cuckoo search: recent advances and applications. *Neural Computing and Applications*, 24(1):169–174, 2014.