Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 10, No. 2, 2022, 257-268.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

A Hybridization Method of Prototype Generation and Prototype Selection for K-NN Rule Based on GSA

Mohadese Rezaei* and Hossein Nezamabadi-pour

Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran.

Article Info

Article History:
Received 21 October 2020
Revised 05 September 2021
Accepted 18 November 2021

DOI:10.22044/JADM.2021.10159.2154

Keywords:
Classification, K-nearest neighbor, Prototype generation, Prototype selection, Gravitational search algorithm, Hybrid method.

**Corresponding author: mo.rezaei@eng.uk.ac.ir (M. Rezaei).*

Abstract

In the present work, we aim to overcome some defects of the *K*-nearest neighbor (*K-NN*) rule. Two important data preprocessing methods to elevate the *K-NN* rule are the prototype selection (PS) and prototype generation (PG) techniques. Often the advantages of these techniques are investigated separately. In this work, using the gravitational search algorithm (GSA), two hybrid schemes are proposed, in which the PG and PS problems are considered together. In order to evaluate the classification performance of these hybrid models, we perform a comparative experimental study including a comparison between our proposals and some approaches previously studied in the literature using several benchmark datasets. The experimental results obtained demonstrate that our hybrid approaches outperform most of the competitive methods.

1. Introduction

The *K-NN* rule is a popular non-parametric classification technique that has been extensively used in pattern recognition [1]. This classifier is conceptually simple since for determining the class label of an unknown sample, it uses the raw training data [2]. In order to classify an unseen sample first according to a distance metric such as the Euclidean distance, its *K* nearest training samples (neighbors) are detected, and then the most frequent label occurring in the *K* neighbors is assigned to that unknown sample. Also the expected error of the *K-NN* rule is bounded by twice the Bayesian error rate [3]. Unfortunately, the simplicity and effectiveness of the *K-NN* classifier have been affected by some weaknesses. First, the *K-NN* classifier has large memory requirements in order to preserve the whole training set. Also classifying a new sample is required to compute the distances between that sample and all t\he samples in the training set, which leads to a high computational cost. The third drawback of the *K-NN* classifier is the sensitivity to noise objects [4].

These drawbacks can be dealt with using the data reduction techniques [5-7] in order to simplify the training data by removing the noisy or redundant data. The goal of these techniques is to reduce the number of instances, while trying to keep a good classification performance. It can be seen from the literature that PS and PG are the two main reduction techniques. In the PS methods the most proper samples are selected from the training set [8]. There are three main types of PS methods [9]: condensation [10], edition [11], and hybrid methods [12]. The purpose of the condensation methods is to obtain a small subset of the training set by removing the instances that are not crucial to the *K-NN* decision, while in edition methods, noisy instances that contribute to the misclassification rate are removed. In the hybrid methods, both approaches are combined. Various procedures to the PS algorithms have been suggested in the literature [13-15].

In the PG methods, some artificial prototypes are generated, and the original training set can be replaced by them [16, 17]. In the taxonomy and

experimental study on the PG methods that have been carried out in [18], these methods have been divided into four main groups: positioning adjustment [19], class relabeling [20], centroid-based [21] and space-splitting [22]. The positioning adjustment methods consist of two-step. In the first step, an initial set of prototypes is selected from the training set, and in the second step, their location is amended through an optimization procedure. The class relabeling methods suppose that some of the training samples could be suspicious of having errors and belonging to other different classes, and hence, the class labels of them must be changed. In the centroid-based techniques, new prototypes are acquired by combining a set of similar examples. Using different heuristics in the space-splitting methods, the feature space is partitioned into some regions and new prototypes are defined in each one of them.

Since the PG and PS problems can be considered as the combinatorial and optimization problems, some works have been done concerning the use of metaheuristic algorithms such as particle swarm optimization (PSO) [23], differential evolution (DE) [24], and gravitational search algorithm (GSA) [25] in order to solve these problems. In PG, the search space is continuous with real variables, and in PS, it is binary with binary variables. In [26-28], PSO and in [29, 30], DE has been applied as the optimizers that can find the best set of prototypes, Also in our previous work [31], GSA that is a population-based metaheuristic search technique of the recent years [32] has been employed in solving the PG problem. Many algorithms have been proposed in order to solve the PG problem but those based on metaheuristics have gained excellent results. In addition, to the best of our knowledge, the PS algorithm reported in [33] can get the best results.

Some intrinsic characteristics of GSA such as flexibility and capability for balancing the exploration and exploitation abilities have led to provide good results by our previous method in [31], which were the best or better than the other comparing PG methods. This success encouraged us to improve these results using the hybrid techniques that incorporate PG and PS. We have two different proposals for combining the PG and PS methods based on GSA: sequential optimization and mixed optimization. In the former approach, a PS phase is performed after the PG stage in order to select the best prototypes among prototypes generated by the PG stage. For the PS phase, a binary GSA (BGSA) [34] is used, and the same as [31], a real GSA (RGSA) [25] is applied in order to execute the PG stage. In the mixed optimization (second approach), a hybrid optimizer such as the mixed gravitational search algorithm (MGSA) [35] that contains RGSA and BGSA is employed in order to simultaneously solve the PG and PS problems. Actually, in this case, the search space includes both types of real and binary variables. Also similar to our previous work in [31], we use the "vote rule" in order to gain better classification accuracy rates. We repeat the PG phase for S times, and hence, we have S sets of the prototypes at the end of the algorithm. Classifying each test sample is done through each one of these S sets separately, and the "vote rule" is used to combine these S results. Herein, the voting rule is the majority voting, and under this rule, a test sample is assigned to the class with the most agreement between S obtained results. These two versions of the proposed methods with the vote rule are called V-SPGS and V-MPGS.

The first contribution of this paper is to examine whether the utilization of hybrid schemes can enhance the classification accuracy of the *K-NN* classifier. Our second goal is to determine which one of these two hybrid methods can achieve a better performance in the field of classification. For this reason, a total of 20 benchmark datasets with different characteristics (number of classes, instances, and features) were considered for experimentation, and the results obtained were compared with our previous method [31], nine existing PG algorithms and *K-NN* classifier (with $K = 1$). It must be mentioned that the current paper is an extension of our previous paper in [31]. In [31], we demonstrated the capabilities of GSA in the PG problem. In the current paper, we propose two new approaches in the same field, and therefore, in the background section, there is a little overlap between them.

The rest of the paper is organized as what follows. In Section, the PS and PG problems are introduced. Furthermore, the gravitational search algorithm for the binary and real-valued variables is described. Two proposed methods are introduced in Section 3. The experimentation and comparison with the competing algorithms are provided in Section 4, and finally, the paper is concluded in Section 5.

2. Background

2.1. Definition of PS and PG problems

In general, the PS problem is explained as follows: $Yp = (y_{p1}, y_{p2}, \ldots, y_{pm}, y_{pc})$ is the sample representation, where the value of the *i*-th feature of the p-th sample is given by yp_i with the real class label of ypc. Then assume that there is a

training set (called TR) that consists of M samples ($TR = \{Y_p \mid p = 1, \dots M\}$) and a test set (called TS) containing T samples ($TS = \{Y_t \mid t = 1, \dots T\}$). For the samples of *TR*, the class label yp_c is known but yt_c is unknown for the samples of *TS*.

The PS algorithm aims to select a subset called SS from *TR*. This subset is applied by the *K-NN* classifier in order to classify each new observation from *TS*. First, the *k* closest prototypes of the SS subset to that test sample are specified, and then the most frequently represented class is assigned to it.

The PG algorithm generates a set of r prototypes (called $GS = \{P_i \mid i = 1, \dots r\}$), in which $r < M$. The *GS* subset must represent the distribution of the classes in *TR* efficiently. Also the size of this subset must be small enough to improve the memory requirement and computational cost of the *K-NN* classifier. In this paper, we use $K = 1$ in the *K-NN* rule.

2.2. Gravitational search algorithm

GSA is a population-based metaheuristic search technique, which has been inspired by the Newtonian laws of gravity and motion. This algorithm simulates the mass interactions and movement of objects (agents) in a search space under the influence of gravitation. In this section, the presentation of three versions of GSA is given: real-valued gravitational search algorithm (RGSA), binary gravitational search algorithm (BGSA), and mixed gravitational search algorithm (MGSA).

2.2.1. Real-valued gravitational search algorithm (RGSA)

GSA was first developed as the continuous search space optimization problems [25]. In the present paper, this version of GSA is called RGSA. In this algorithm, the movement of *N* agents in *D*-dimensional search space is accomplished under the influence of the gravity force. The performance and the position of each agent specify the mass of a solution to the problem. The best solutions to the problem correspond to those agents with a higher performance. X_i denotes the position of the *i*-th object defined by:

$$X_i = (x_i^1, \dots, x_i^d, \dots, x_i^D); i = 1, 2, \dots, N \quad (1)$$

where x_i^d is the position of the *i*-th object in the *d*-th dimension, and *D* is the dimension of the search space. The current population's fitness is used in order to calculate the mass of each object as follows:

$$M_i(t) = \frac{fit_i(t) - worst(t)}{\sum_{j=1}^{N} (fit_j(t) - worst(t))} \quad (2)$$

The mass and the fitness value of the object *i* at iteration *t* are given by $M_i(t)$ and $fit_i(t)$, and $worst(t)$ is the worst fitness value obtained in the swarm of objects at *t*. The total force from a set of heavier objects that acts on the *i*-th agent is considered based on a modified law of gravity (Eq. (3)), and the acceleration of this agent is computed using the law of motion (Eq. (4)). Afterward, the next velocity of the *i*-th agent is calculated as a fraction of its current velocity added to its acceleration (Eq. (5)). Then its next position could be calculated using Eq. (6).

$$F_i^d(t) = \sum_{j \in Kbest, j \neq i} rand_j^d G(t) \frac{M_j(t) \times M_i(t)}{R_{ij}(t) + \varepsilon} (x_j^d(t) - x_i^d(t)) \quad (3)$$

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} = \sum_{j \in Kbest, j \neq i} rand_j^d G(t) \frac{M_j(t)}{R_{ij}(t) + \varepsilon} (x_j^d(t) - x_i^d(t)) \quad (4)$$

$$v_i^d(t+1) = rand_i^d \times v_i^d(t) + a_i^d(t) \quad (5)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (6)$$

where rand is a uniformly distributed random number in the interval [0,1], ε is a small value, and $R_{ij}(t)$ is the Euclidean distance between the two objects *i* and *j*. *Kbest* is the set of first *K* objects with the best fitness value and the heaviest mass. *K* is a decreasing function of time, initially is set to K_0 and reduced linearly over time. Here, K_0 is set to *N* (total number of agents), and *K* is decreased linearly to 1. In GSA, the initial value of the gravitational constant (*G*) is G_0 that is declined linearly during the running time of the algorithm.

$$G = G(G_0, t) \quad (7)$$

2.2.2. Binary gravitational search algorithm (BGSA)

As mentioned earlier, the basic model of GSA is suitable for the real-valued optimization problems. In [34], the binary version of GSA was introduced in order to optimize the binary structured problems. In a binary search space, every dimension has a value of 0 or 1. Moving in each dimension means that its value changes from 0 to 1 or *vice versa*. In BGSA, the force, acceleration, and velocity are updated the same as RGSA but the binary version uses the Hamming distance in order to compute the distance of *R*.

BGSA updates the velocity based on Eq. (5). Then the position is considered to be 1 or 0 with a probability according to Eq. (8). Once $S(v_i^d)$ is calculated, the object will move by the rule explained in Eq. (9).

$$S(v_i^d(t)) = |\tanh(v_i^d(t))| \quad (8)$$

$$\begin{aligned} &\textit{if} \quad rand < S(v_i^d(t+1)) \quad \textit{then} \\ &\quad x_i^d(t+1) = complement(x_i^d(t)) \quad (9) \\ &\textit{else} \quad x_i^d(t+1) = x_i^d(t) \end{aligned}$$

2.2.3. Mixed gravitational search algorithm (MGSA)

There are some optimization problems with real and binary variables. In order to solve these types of problems, the mixed optimization algorithms such as MGSA are required [35]. In MGSA, the objects move toward the optimal places in the search space with the dimensions of both the real and binary variables. An object in this complex search space is called 'super-agent', and the computation of force, velocity, and movement of each super-agent are performed separately in the real and binary parts. Also the fitness of each super-agent is the same for the real and binary parts, and is used for mass calculation. In order to compute the distance between two super-agents, the Euclidean distance and the Hamming distance are applied for the real and binary parts, respectively [35]. Different gravitational constants may be used in the real and binary parts.

3. Proposed methods

In this section, the main characteristics of the two hybrid methods will be explained. In order to do this, first, a *TR* set consisting of *M* samples with m features is considered. These *M* samples are classified into *C* classes. We will introduce two hybrid models in order to concatenate the PG and PS problems: sequential optimization for PG and PS hybridization (called in brief as SPGS) and mixed optimization for PG and PS hybridization (called in brief as MPGS). Sections 3.1 and 3.2 describe the construction of SPGS and MPGS, respectively.

3.1. Sequential optimization for PG and PS hybridization (SPGS)

The SPGS method has two distinct optimization phases. In the preliminary phase, there is a continuous search space where RGSA uses the *TR* samples in order to generate a *GS* set containing $2\times r$ prototypes. In the second phase, using BGSA, the best *r* prototypes with the highest classification accuracy rate are selected from the generated prototypes in *GS*. These r prototypes are placed in the final set of prototypes (*SS*). The second phase is performed in a binary search space. More details are interpreted in the following sub-sections.

Figure 1. Agent representation in continuous space

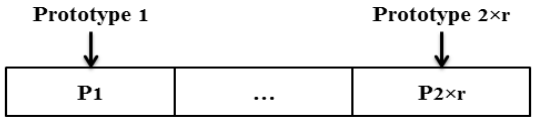

Figure 2. Agent representation in binary space.

3.1.1. Agent representation

In the continuous space, each agent is a set of $2 \times r$ prototypes, and the position of the agent corresponds to a solution to the problem. The data dimension is m, and thus the length of each agent is $(2 \times r) \times m$. The structure of an agent in the continuous space is shown in Figure 1. In the binary space, each agent is a binary string with the length of $2 \times r$ bits, shown in Figure 2, where each bit is related to a prototype. In this binary string, one and zero mean the existence and non-existence of a prototype in the final set SS, respectively.

3.1.2. Initialization

RGSA is initialized with *N* agents, each agent containing a random selection of $2 \times r$ samples from *TR*. Each class is represented by some prototypes that are related to the number of class members in the *TR* set. Also in order to initialize each agent in BGSA, a $2\times r$-bit string of 0 and 1 is produced by the uniform distribution.

3.1.3. Fitness function

RGSA and BGSA require a fitness function to guide the search process. In this work, we examine the classification accuracy rate of the *K-NN* as the fitness function. In order to compute the classification accuracy rate of an agent, first, the entire *TR* is classified by the agent's prototypes, and then the number of correct classifications (*Ncc*) is counted and divided by the total number of classifications. This proportion is regarded as the fitness of that agent. Eq. (10) shows how to calculate the fitness value for each agent.

$$fitness(i) = \frac{100}{M} \times Ncc_i, \qquad i = 1, 2, ..., N \quad (10)$$

3.1.4. Gravitational constant

The gravitational constant, *G*, is set using Eq. (11) for both RGSA and BGSA. The initial value of *G* is G_0, which declines linearly during the running time of the algorithm.

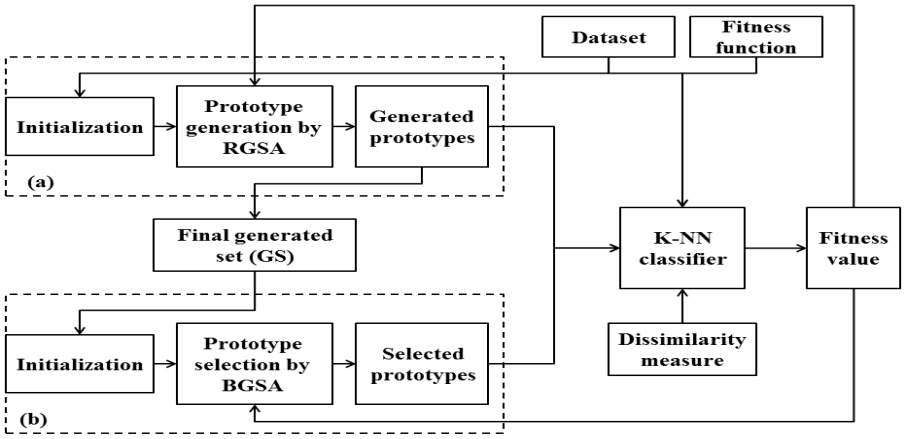

Figure 3. A block diagram of SPGS method.

$$G = G_0\left(1-\frac{t}{T}\right) \tag{11}$$

A block diagram of SPGS is given by Figure 3. As this block diagram shows, in the first phase, initialization is performed using the dataset. Then the fitness values of all agents in the swarm are computed, and these values are passed into RGSA to determine the next position of the swarm, which is the new solution of the problem and must be evaluated. A fixed number of iterations, which is a pre-determined parameter, is considered as the stopping criterion of the algorithm. At the end of the first phase, a *GS* set is obtained, which is the position of $2 \times r$ optimal prototypes. *GS* is delivered to the second phase, where the best r prototypes from *GS* are selected by BGSA and located in the final set, *SS*. Classifying the new samples are done using the prototypes of *SS* in the performance measurement phase.

Besides, the higher classification accuracy rate in SPGS can be achieved using the "vote rule". We repeat the PG phase for *S* times, and hence, we have *S* sets of the prototypes at the end of the algorithm. Classifying each test sample is done through each one of these *S* sets separately, and the "vote rule" is used in order to combine these *S* results. Herein, the voting rule is the majority voting, and under this rule, a test sample is assigned to the class with the most agreement between S obtained results. We call this version of SPGS as V-SPGS in the experiments.

3.2. Mixed optimization for PG and PS hybridization (MPGS)

As mentioned earlier, in the MPGS method, both the PG and PS problems are jointly optimized by MGSA, whereas in this case, there is a complex search space; the super-agents containing the real and binary parts move in this space to obtain the best set of prototypes. The main specifications of this algorithm are expressed in the following.

3.2.1. Super-agent representation

The structure of a super-agent is indicated by Figure 4. As indicated in this figure, a super-agent has a real part to display the position of prototypes and a binary part to select the best prototypes. In the real part, the prototypes are encoded sequentially, and in the binary part, each bit is associated with a prototype. In the MPGS method for each super-agent, those prototypes that are chosen by the binary part are presented in the final set. The length of the real part is $(2 \times r) \times m$ and the binary part is $2 \times r$.

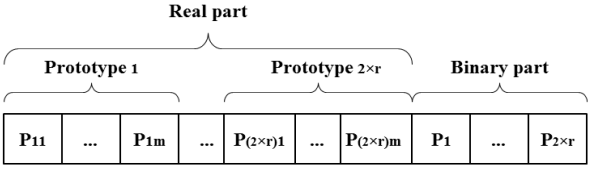

Figure 4. Super-agent representation.

3.2.2. Initialization

The initial population of MGSA has two parts, and the initialization of each part should be done separately. For each super-agent, in the real part, $2 \times r$ samples are chosen randomly from *TR*, and in the binary part, a $2 \times r$-bit string of 0 and 1 is produced by the uniform distribution. Since the binary part is produced by a uniform distribution, almost half of the prototypes of the real part are selected and the final set contains *r* prototypes. Therefore, the reduction rate remains unchanged.

3.2.3. Fitness function

The classification accuracy rate is the fitness function of this algorithm, and calculated by Eq. (10).

3.2.4. Gravitational constant

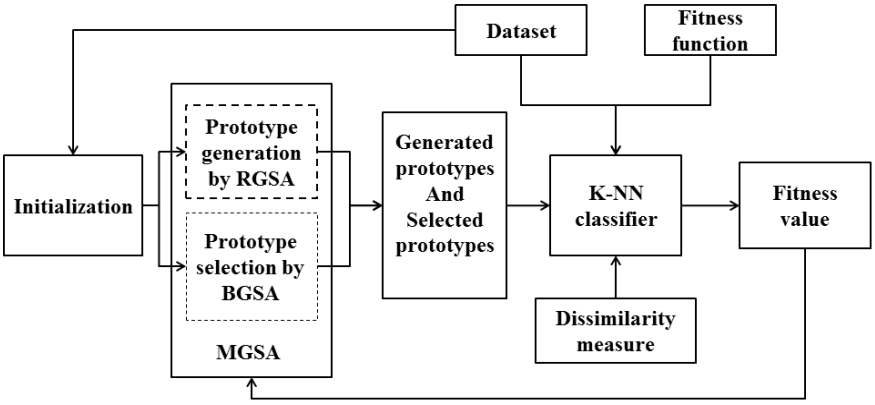

Figure 5. A block diagram of MPGS method.

In MGSA, the gravitational constant of the real and binary parts is calculated by Eq. (11).

The description of this method is demonstrated as the block diagram in Figure 5. It is clear in this figure that MGSA is applied to jointly optimize the PG and PS blocks, which cause to achieve the maximum classification accuracy rate. The super-agents of the swarm are evaluated by the fitness function, and the corresponding fitness values are applied by MGSA in order to determine the next position of the swarm. The same as SPGS, in MPGS, the maximum number of iterations is the stopping criterion.

4. Experimental results

In this section, the performance comparison is done between our two hybrid methods, 1-NN classifier, ten state-of-the-art PG methods including LVQ3 [36], MGauss [37], HYB algorithm [38], RSP3 [39], ENPC [40], PSO [26], AMPSO [27], IPADE [30], SSMA-SFLSDE [29], and our previous work called GPG, which has been proposed in [31]. The performance measurement of the competing algorithms is the classification accuracy rate, and the KEEL software tool [41] is used to carry out their experiments. Appendix A in [31] gives a brief explanation of these algorithms and their parameters.

30 benchmark datasets from the UCI repository [42] are selected to perform all the experiments. Some related studies such as [15, 29-31] have been used this repository. These datasets have been selected in such a way that the dataset sizes and the number of features and classes are varied. The characteristics of these datasets are listed in Table 1. We applied the 10-fold cross-validation schema on these 20 datasets, and in each partition, three runs of the algorithms are performed. At the end, 30 runs for each algorithm are provided, and the average of these trials is used to compare them. Furthermore, as recommended by many proposed PG methods, the feature values of the datasets are then normalized into the [0, 1] interval.

Finally, the Friedman Aligned-Rank test that is a widely-used hypothesis in the non-parametric test is used to provide a statistical analysis of the results [43]. Using this non-parametric test, $1 \times N$ comparisons have been done, and the rank is computed for each method. These ranks are used in order to determine whether the statistical differences lie among a group of results [44]. In this work, we apply the Friedman Aligned (FA) procedure with a post-hoc method called Holm's test [45] for the p-value adjustment.

Table 1. Characteristics of 30 datasets used in experiments.

Dataset	#Ex.	#feat.	#Cl.
Appendicitis	106	7	2
Australian	690	14	2
Balance	625	4	3
Bands	365	19	2
Breast-Cancer	569	31	2
Bupa	345	6	2
Cleveland	297	13	5
Dermatology	366	33	6
E.coli	327	7	5
Glass	214	9	7
Haberman	306	3	2
Hayes-Roth	160	4	3
Heart	270	13	2
Hepatitis	155	19	2
Horse	364	27	3
Ionosphere	351	33	2
Iris	150	4	3
Led7digit	500	7	10
Mammographic	830	5	2
Monks	432	6	2
Pima	768	8	2
Spectfheart	267	44	2
Sonar	208	60	2
Tae	151	5	3
Thyroid(new)	215	5	3
Vehicle	846	18	4
WDBC	569	30	2
Wine	178	13	3
Wisconsin	699	9	2
Zoo	101	16	7

Here, the null hypothesis states that there is no significant difference between the performances of two PG methods. In order to reject the null hypothesis, the p-value must be less than or equal to a level of significance of $\alpha = 0.1$, the same as [29]. This case means that a significant difference does exist between two PG methods.

4.1. Parameters of proposed methods

The parameters of the SPGS and MPGS methods must be set. The size of the initial population (N) is set to 20, and the maximum iteration number (T) is set to 250. Eq. (11) is used to set G for RGSA, BGSA, and MGSA, where G_0 is set to 1 and T is the iteration number. The initial reduction rate of these algorithms is fixed to 0.90, and the number of primary prototypes is $2 \times r = 0.1 \times M$ (M is the TR size). However, in both SPGS and MPGS, almost half of these prototypes are available in the final set. Therefore, similar to the GPG method [31], the final reduction rate of them is 0.95. In our proposed methods, we have used *1-NN* rule with the Euclidean distance in order to compute the distances between the samples. Also in order to apply the voting, the S parameter is set to 5.

4.2. Analysis of results

In this sub-section, the results obtained are analyzed by several experiments. Our goal is to compare the two current proposed methods with each other and different PG methods. Tables 2 and 3 show the train and test average classification accuracy on 30 runs for each algorithm. We also use the boldface style to highlight the best result. The last row of each table shows the average classification accuracy obtained over all datasets. The results of the statistical comparisons are presented in Table 4. In this table, the best (lowest) and the worst (highest) rank are located at the top and the bottom, respectively. Furthermore, the third column contains the adjusted p-values with the Holm post-hoc test (Holm APV). The Friedman Aligned-Rank test considers the best method that has obtained the best FA ranking as the control algorithm, and in this experiment, V-SPGS is the best one. The analysis of results has two parts including the performance comparison of the two hybrid models proposed in this paper with the previously proposed methods and also two hybrid models to each other.

4.2.1. Comparison with different PG methods

In this sub-section, we compare the two hybrid models with different PG methods. According to the results reported in Tables 2-5, we can summarize some analyses:

- The highest average classification accuracy over the test sets has been achieved by SPGS and VMPGS, respectively because in SPGS, after generating the prototypes, the best prototypes with the highest accuracy rates are inserted in the final set.
- After SPGS and V-MPGS, we can see V-SPGS that has obtained the second rank of the test average classification accuracy. This means that both hybrid models have achieved good results.
- We have observed from Table 3 that V-MPGS performs well with four large datasets that have more than 500 instances (Australian, Balance, Breast-Cancer, Wisconsin). Also V-MPGS could achieve the first rank on eight of the datasets (Appendicitis, Australian, Balance, Breast-Cancer, Heart, Iris, Wine, Wisconsin).
- In Table 4, we can see that the best FA ranking has been obtained by V-SPGS with a rank of 113.93 in comparison, and V-MPGS and SPGS could achieve the second- and third-ranking. The existence of significant differences between the competing algorithms is confirmed by p-values of the FA ranking.
- Regarding Table 4, the Holm's procedure states that the differences of V-SPGS over AMPSO, HYB, ENPC, RSP3, 1-NN, LVQ3, IPADE, and MGauss are significant. This hypothesis is rejected by two state-of-the-art algorithms (PSO and SSMA-SFLSDE) and GPG.

4.2.2. Comparison of two hybrid models

In this sub-section, we focus on comparing two hybrid models using a statistical test called the Wilcoxon test. Table 5 shows the ranking of R+ and R- values and its related p-value. Here, the same as the Fa ranking test, we consider a level of significance of $\alpha = 0.1$ for the Wilcoxon test. As it can be seen in this table, SPGS outperforms MPGS, and there are no significant differences between the V-SPGS and V-MPGS methods. Also R+ is greater for SPGS in the first row and V-SPGS in the second row of the table.

Table 2. Average classification accuracy on training set.

Dataset	**Algorithm**														
	1-NN	SPGS	MPGS	V-SPGS	V-MPGS	GPG [31]	LVQ3 [36]	MGauss [37]	HYB [38]	RSP3 [39]	ENPC [40]	PSO [26]	AMPSO [27]	IPADE [30]	SSMA-SFLSDE [29]
Appendicitis	80.61	90.32	90.29	90.38	90.30	90.71	87.31	85.61	89.38	86.27	88.78	90.95	82.03	90.88	**92.35**
Australian	80.71	88.67	88.58	88.54	88.59	88.85	83.78	85.98	83.50	86.91	87.72	88.79	84.96	84.04	**89.57**
Balance	78.95	**96.63**	91.74	94.81	91.97	94.96	80.83	87.86	97.32	86.72	91.88	90.97	62.28	83.59	91.98
Bands	73.47	**85.44**	80.11	85.74	83.76	84.88	67.08	68.73	93.26	82.66	93.01	78.46	68.91	71.10	80.25
Breast-Cancer	96.40	97.52	97.09	97.40	97.06	97.27	95.41	95.28	96.84	97.24	98.49	97.16	94.03	**97.76**	97.23
Bupa	61.22	**86.61**	80.13	85.57	80.21	85.18	58.98	63.10	90.20	80.58	86.51	75.69	61.77	67.29	80.42
Cleveland	52.77	69.32	67.43	68.35	67.35	**69.57**	56.44	60.22	70.69	65.02	69.50	67.73	58.51	64.03	67.88
Dermatology	95.63	98.88	98.63	98.88	98.70	98.83	95.97	97.70	97.94	96.18	98.71	98.30	90.66	96.75	**99.61**
E.coli	78.57	**92.84**	91.07	92.41	91.02	92.30	70.37	73.94	88.82	85.02	85.12	85.75	72.59	81.22	89.19
Glass	70.77	**82.69**	76.17	82.54	76.54	79.98	47.89	53.31	80.39	76.89	81.62	69.49	52.63	70.23	82.41
Haberman	67.14	**81.72**	80.00	81.65	80.06	81.50	70.77	74.76	68.84	74.64	72.70	78.50	74.15	73.36	79.81
Hayes-Roth	35.44	81.60	77.80	80.86	78.17	78.38	43.37	54.64	31.06	33.18	37.94	72.21	62.26	82.83	**86.03**
Heart	75.88	86.97	87.16	87.12	87.22	87.64	81.8	82.81	**90.20**	85.14	89.66	87.05	80.42	83.13	88.89
Hepatitis	80.65	92.26	91.39	92.25	90.80	91.61	75.49	79.79	**92.76**	83.37	83.44	88.96	81.43	87.89	90.18
Horse	60.25	82.18	78.40	81.62	78.24	**82.76**	68.36	67.80	78.32	80.53	72.56	81.34	66.39	75.28	78.65
Ionosphere	86.93	91.48	91.18	91.23	90.98	90.25	81.86	91.86	92.66	88.13	**97.69**	91.93	82.37	89.90	96.01
Iris	95.48	98.96	98.96	99.07	98.98	99.28	91.7	94.74	95.78	95.78	97.97	99.11	92.22	98.59	**99.41**
Led7digit	40.22	78.10	77.75	78.12	76.95	77.83	65.27	77.57	74.89	59.11	61.33	77.36	63.93	77.47	**78.84**
Mammographic	73.77	83.01	82.51	82.88	82.80	83.17	72.30	79.38	63.57	76.99	74.18	82.74	79.48	81.81	**83.87**
Monks	77.55	93.43	91.17	92.93	90.97	94.68	80.37	79.78	96.90	75.31	96.73	87.35	72.16	74.64	**97.09**
Pima	70.70	83.78	80.32	83.55	80.48	84.50	71.04	74.56	**88.26**	80.80	87.80	81.19	70.98	77.35	83.98
Spectfheart	69.46	82.99	82.72	82.96	83.40	84.60	79.07	80.67	94.57	80.44	**94.67**	88.68	80.86	85.77	82.39
Sonar	86.32	86.93	87.64	87.92	87.47	88.05	78.53	73.93	95.99	90.17	**98.40**	92.79	75.32	81.94	86.99
tae	42.11	66.00	63.90	65.74	63.84	63.94	48.79	42.68	51.81	54.75	50.05	64.17	56.44	62.25	**70.49**
Thyroid	96.64	99.53	99.00	99.72	98.91	99.41	89.94	95.59	97.73	95.86	98.03	99.17	91.09	98.22	**99.85**
vehicle	69.40	68.49	63.63	68.57	62.89	67.48	57.63	50.68	84.80	78.24	**87.73**	71.87	55.91	58.38	68.55
WDBC	95.66	97.40	97.04	97.47	97.06	97.44	92.97	95.53	97.66	97.36	98.26	98.42	94.63	97.54	**98.83**
Wine	95.57	99.17	99.02	99.10	99.03	99.19	94.78	97.44	97.11	95.57	99.48	99.75	87.77	98.63	**100.0**
Wisconsin	95.69	97.88	97.74	97.89	97.76	97.77	95.6	97.38	99.00	96.32	98.38	**98.41**	85.16	97.62	98.27
Zoo	92.08	99.24	98.50	99.28	98.25	**99.50**	88.47	98.39	94.62	88.65	94.27	98.87	85.76	96.32	98.44
Average	75.87	**88.00**	86.24	87.82	86.33	87.72	75.74	78.72	85.83	81.79	85.75	86.11	75.57	82.86	87.91

4.3. Discussion

The results obtained show that if PG and PS are considered together in solving the mentioned problem about the K-NN classifier, the performance improvement is higher than a situation with only PG or PS because in this case, we can benefit from the advantages of both of them. In the sequential method proposed here, the overall search space is explored by the algorithm, and the best prototypes are introduced to the PS phase. After that, the PS algorithm selects among this set the best of best prototypes. Hence, this approach can achieve a better performance than a competing PG algorithm. We can generalize this finding to MPGS. Again, the mixed algorithm can gain the best of best prototype using a mixed search algorithm that is able to generate and select jointly.

Table 3. Average classification accuracy on test set.

Dataset	**Algorithm**													
	1-NN	SPGS	MPGS	V-SPGS	V-MPGS	GPG [31]	LVQ3 [36]	MGauss [37]	HYB [38]	RSP3 [39]	ENPC [40]	PSO [26]	AMPSO [27]	IPADE [30]	SSMA-SFLSDE [29]
Appendicitis	79.36	86.15	85.36	86.00	**87.00**	85.67	85.09	84.42	73.91	80.18	77.18	85.70	79.09	85.76	83.27
Australian	81.45	86.09	86.28	86.09	**86.38**	85.85	82.99	85.89	71.54	82.9	77.29	85.65	83.28	84.15	86.38
Balance	79.04	89.43	88.63	89.44	**90.71**	90.39	79.51	86.51	68.10	84.31	71.56	87.08	60.32	80.19	89.28
Bands	63.09	67.36	66.89	68.14	67.46	66.69	64.96	68.10	**70.34**	69.40	69.77	68.29	66.06	65.50	69.78
Breast-Cancer	95.28	96.01	95.78	95.77	**96.48**	96.30	94.27	94.43	65.46	95.6	94.43	95.68	92.61	96.42	95.38
Bupa	61.08	67.72	**67.92**	66.86	66.58	66.32	57.75	60.70	59.83	57.72	61.92	66.43	57.35	63.60	66.00
Cleveland	53.14	52.17	54.23	53.63	54.32	54.16	54.67	53.99	50.67	53.18	52.03	55.93	53.44	55.36	**56.15**
Dermatology	95.35	**97.47**	96.90	**97.47**	97.24	96.53	96.06	95.88	94.66	95.50	95.40	93.85	89.65	95.03	95.37
Ecoli	80.70	**87.68**	87.57	87.38	86.44	87.27	68.74	72.94	77.69	81.57	76.50	77.36	71.16	77.14	83.07
Glass	73.61	66.13	65.19	72.20	66.05	66.08	46.47	47.68	**73.71**	67.11	70.89	62.27	50.88	61.99	71.98
Haberman	66.97	71.30	71.21	73.16	75.43	71.64	74.44	76.67	58.89	66.67	60.00	**77.78**	74.44	71.11	71.53
Hayes-Roth	35.70	66.46	65.00	64.38	61.88	65.63	39.67	55.36	30.93	26.48	27.68	65.74	55.13	**77.77**	75.41
Heart	77.04	82.96	81.97	82.59	**83.33**	81.48	80.37	80.37	73.95	72.59	77.16	81.23	77.16	81.60	82.22
Hepatitis	80.75	83.78	82.75	**87.00**	85.84	81.36	74.13	78.04	73.00	75.46	75.50	76.92	76.17	82.58	81.21
Horse	57.84	67.76	65.91	65.90	64.81	67.84	65.48	64.14	**72.25**	65.13	71.15	67.24	60.85	61.59	62.87
Ionosphere	85.48	88.31	87.84	**92.02**	89.45	87.07	82.03	88.04	86.32	86.32	86.33	86.33	80.04	85.21	88.02
Iris	93.33	95.11	94.44	**95.33**	**95.33**	94.22	93.78	93.33	93.33	94.00	92.89	94.44	91.78	94.67	94.00
Led7digit	40.20	73.27	**73.53**	73.40	73.42	72.20	62.93	73.27	67.07	56.8	59.27	72.20	62.33	72.73	71.40
Mammographic	73.68	81.10	81.00	81.31	81.14	**81.72**	70.24	78.67	57.02	74.09	70.55	80.75	78.15	81.27	81.27
Monks	77.91	88.22	86.34	89.55	86.35	90.06	78.06	78.50	74.10	70.92	80.19	82.96	69.41	71.40	**95.44**
Pima	70.33	75.06	**76.27**	75.41	76.01	75.05	70.53	73.84	64.17	71.88	65.50	74.75	67.51	75.24	74.89
Spectfheart	69.70	77.32	77.53	79.46	75.73	76.82	75.26	76.40	73.77	74.22	75.68	79.09	**79.83**	79.09	79.02
Sonar	85.55	78.54	77.11	81.67	76.43	74.52	72.57	70.71	86.98	82.64	**88.95**	79.71	66.74	75.50	79.29
Tae	40.50	56.38	**57.07**	53.17	55.08	56.82	45.08	32.58	42.54	46.54	43.88	63.04	52.54	53.79	56.54
Thyroid	97.23	96.46	95.53	97.27	95.84	96.48	90.87	94.44	95.35	95.41	94.74	96.63	92.45	97.11	**97.68**
vehicle	70.10	61.79	61.00	66.79	65.96	59.86	54.85	50.00	69.02	**67.73**	66.30	66.20	55.68	56.75	65.84
WDBC	95.26	96.19	95.72	96.13	95.78	95.54	91.74	95.08	94.73	95.08	93.50	97.01	94.73	97.19	**96.31**
Wine	95.52	95.87	95.34	96.08	**96.63**	95.86	96.04	96.60	96.08	93.23	95.14	96.27	87.05	94.01	94.93
Wisconsin	95.57	96.80	96.90	96.20	**97.25**	97.15	95.73	96.85	94.03	94.88	94.37	96.65	84.57	96.95	96.14
Zoo	92.81	95.95	94.74	**96.83**	94.47	96.74	92.53	94.47	92.42	88.92	93.86	92.58	85.44	93.45	95.33
Average	75.45	**80.83**	80.40	81.55	**80.83**	80.44	74.56	76.60	73.40	75.55	75.32	80.19	73.19	78.81	81.2

5. Conclusion

In this work, we aimed to improve the classification performance of the *K-NN* classifier. PG and PS were two ways of solving this problem. However, traditionally, they were considered separately. In this work, we presented two hybrid techniques that incorporated the PG and PS problems. Two different proposals for combining the PG and PS methods based on GSA were introduced: SPGS and MPGS. The first contribution of this paper was to examine whether the utilization of the hybrid schemes could improve the classification accuracy of the *K-NN* classifier. Our second objective was to determine which one of these two hybrid methods could achieve a better performance in the field of classification.

Table 4. Average ranking of algorithms (Friedman Aligned-Ranks and adjusted p-values with Holm test.

Algorithm	FA ranking	Holm APV
V-SPGS	113.93	-
V-MPGS	137.80	0.9532
SPGS	144.28	0.9532
SSMA-SFLSDE	147.48	0.9532
GPG	160.98	0.7177
MPGS	163.05	0.7177
PSO	171.10	0.5320
IPADE	224.10	0.0072
MGauss	254.58	0.0002
RSP3	291.98	10^{-6}
1-NN	295.08	10^{-6}
ENPC	299.91	0
HYB	312.32	0
LVQ3	323.93	0
AMPSO	341.95	0

Table 5. Results of Wilcoxon signed-ranks test comparing two hybrid models.

Comparison	R+	R-	p-value
SPGS *vs.* MPGS	336	99	0.0093
V-SPGS *vs.* V-MPGS	292.5	172.5	0.21

The experimental study carried out showed that both hybrid models obtained very competitive results. The results obtained confirmed that these hybrid models worked properly to tackle the PG problem with promising results. The results obtained were compared with two non-parametric statistical procedures that had supported the conclusion drawn.

As a future work, we plan to integrate the other data reduction techniques such as the feature selection and feature weighting with PG and PS in the *K-NN* performance improvement.

References

[1] T.M. Cover and P.E. Hart, "Nearest neighbor pattern classification", *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, 1967.

[2] W Li, Y Chen, and Y Song, Knowledge-Based Systems, "Boosted K-nearest neighbor classifiers based on fuzzy granules", *Knowledge-based Systems*, vol. 195, pp. 1-13, 2020.

[3] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley Inter science, New York, 1973.

[4] J. Maillo, S. Ramírez, I. Triguero, and F. Herrera, "kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data", *Knowledge-based Systems*, vol. 117, pp. 3-15, 2017.

[5] JJ. Valero-Mas and FJ. Castellanos, "Data Reduction in the String Space for Efficient k-NN Classification through Space Partitioning", *Applied Sciences*, vol. 10, pp. 33-56, 2020.

[6] Z. Su, Q. Hu, and T. Denaeux, "A Distributed Rough Evidential K-NN Classifier: Integrating Feature Reduction and Classification", *IEEE Transactions on Fuzzy Systems*, vol. 29, pp. 2322-2335, 2020.

[7] J. Hamidzadeh, N. Kashefi, and M. Moradi, "Combined weighted multi-objective optimizer for instance reduction in two-class imbalanced data problem", *Engineering Applications of Artificial Intelligence*, vol. 90, pp. 103500, 2020.

[8] D. Hwang and D. Kim, "Nearest Neighbor-based prototype classification Preserving Class Regions", *Journal of Information Processing Systems*, vol. 13, pp. 1345-1357, 2017.

[9] S. García, J. Derrac, JR. Cano, and F.Herrera "Prototype selection for nearest neighbor classification: Taxonomy and empirical study", *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 34, pp. 417–435, 2012.

[10] M. N. Ivanov, "Prototype sample selection based on minimization of the complete cross-validation functional", *Pattern Recognition and Image Analysis*, vol. 20, pp. 427–437, 2010.

[11] N. Segata, E. Blanzieri, S. J. Delany, and P. Cunningham, "Noise reduction for instance-based learning with a local maximal margin approach", *Journal of Intelligent Information Systems*, vol. 35, pp. 301–331, 2010.

[12] S. Ferrandiz and M. Boull´ e. "Bayesian instance selection for the nearest neighbor rule", *Machine Learning*, vol. 81, pp. 229–256, 2010.

[13] M. Rubbo and LA. Silva, "Prototype Selection using Self-Organizing-Maps and Entropy for Overlapped Classes and Imbalanced Data", *International Joint Conference on Neural Networks*., IJCNN, 2018, pp.1-8.

[14] P. Rosero-Montalvo and DH. Peluffo-Ordóñez, "Prototype reduction algorithms comparison in nearest neighbor classification for sensor data: Empirical study", *IEEE Second Ecuador Technical Chapters Meeting*., ETCM, 2017, pp. 408–421.

[15] JR. Rico-Juan, JJ. Valero-Mas, and J. Calvo-Zaragoza, "Extensions to rank-based prototype selection in k-nearest neighbor classification", *Applied Soft Computing*, vol. 85, pp. 105803, 2019.

[16] H. J. Escalante, M. Marin-Castro, A. Morales-Reyes, M. Graff, A. Rosales-Pe´rez, M. Montes-y-Go´mez, C. A. Reyes, and J. A. Gonzalez, "MOPG: a multi-objective evolutionary algorithm for prototype generation", *Pattern Analysis and Applications*, vol. 20, pp. 33-47, 2017.

[17] M. Elkano, M. Galar, J. Sanz, and H. Bustince, "CHI-PG: A fast prototype generation algorithm for Big Data classification problems", *Neurocomputing*, vol. 287, pp. 22-33, 2018.

[18] I. Triguero, J. Derrac, S. García, and F.Herrera, "A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification", *IEEE*

Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, vol. 42, pp. 86-100, 2012.

[19] J. Li, M.T. Manry, C. Yu, and D.R. Wilson, "Prototype classifier design with pruning", *International Journal on Artificial Intelligence Tools*, vol. 14, pp. 261–280, 2005.

[20] J.S. Sa ´nchez, R. Barandela, A.I. Marque ´ s, R. Alejo, and J. Badenas, "Analysis of new techniques to obtain quality training sets", *Pattern Recognition Letters*, vol. 24, pp. 1015–1022, 2003.

[21] H.A. Fayed, S.R. Hashem, and A.F. Atiya, "Self-generating prototypes for pattern classification", *Pattern Recognition*, vol. 40, pp. 1498–1509, 2007.

[22] T. Raicharoen and C. Lursinsap, "A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm", *Pattern Recognition Letters*, vol. 26, pp. 1554–1567, 2005.

[23] Z. Hassani, M. Alambardar Meybodi, "Hybrid Particle Swarm Optimization with Ant-Lion Optimization: Experimental in Benchmarks and Applications", *Journal of AI and Data Mining*, vol. *9*, pp. 583-595, 2021.

[24] R. Storn and K.V. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[25] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a gravitational search algorithm", *Information Sciences*, vol. 179, pp. 2232–2248, 2009.

[26] L. Nanni and A. Lumini, "Particle swarm optimization for prototype reduction", *Neurocomputing*, vol. 72, pp. 1092–1097, 2008.

[27] A. Cervantes, I.M. Galva ´n, and P. Isasi, "AMPSO: a new particle swarm method for nearest neighborhood classification", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 39, pp. 1082–1091, 2009.

[28] C. Jui-Le, T. Shih-Pang, and Y. Chu-Sing, "Using Particle Swarm Method to Optimize the Proportion of Class Label for Prototype Generation in Nearest Neighbor Classification", *Advanced Technologies, Embedded and Multimedia for Human-centric Computing, Lecture Notes in Electrical Engineering*, vol. 260, pp. 239-245, 2014.

[29] I. Triguero, S. Garcı´a, and F. Herrera, "Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification", *Pattern Recognition* vol. 44, pp. 901–916, 2011.

[30] I. Triguero, S. Garcı´a, and F. Herrera, "IPADE: iterative prototype adjustment for nearest neighbor classification", *IEEE Transactions on Neural Networks*, vol. 21, pp. 1984–1990, 2010.

[31] M. Rezaei and H. Nezamabadi-pour, "Using gravitational search algorithm in prototype generation for nearest neighbor classification", *Neurocomputing*, vol. 157, pp. 256-263, 2015.

[32] H. Nezamabadi-pour, "A quantum-inspired gravitational search algorithm for binary encoded optimization problems", *Engineering Applications of Artificial Inteligence*, vol. 40, pp. 62-75, 2015.

[33] S. García, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach", Pattern Recognition, vol. 41, pp. 2693–2709, 2008.

[34] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "BGSA: binary gravitational search algorithm", *Natural Computing*, vol. 9, pp. 727–745, 2010.

[35] S. Sarafrazi and H. Nezamabadi-pour, "Facing the classification of binary problems with a GSA-SVM hybrid system", *Mathematical and Computer Modelling*, vol. 57, pp. 270–278, 2013.

[36] T. Kohonen, "The self-organizing map", *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, 1990.

[37] M. Lozano, J. M. Sotoca, J. S. S´ anchez, F. Pla, E. Pekalska, and R. P. W. Duin, "Experimental study on prototype optimization algorithms for prototype-based classification in vector spaces", *Pattern Recognition*, vol. 39, pp. 1827–1838, 2006.

[38] S. W. Kim and J. Oomenn, "Enhancing prototype reduction schemes with LVQ3-type algorithms", *Pattern Recognition*, vol. 36, pp. 1083–1093, 2003.

[39] J. S. Sánchez, "High training set size reduction by space partitioning and prototype abstraction", *Pattern Recognition*, vol. 37, pp. 1561-1564, 2004.

[40] F. Fernández and P. Isasi, "Evolutionary design of nearest prototype classifiers", *Journal of Heuristics*, vol. 10, pp. 431–454, 2004.

[41] I. Triguero, S. González, M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M. Jesús, L. Sánchez, and F. Herrera, "KEEL 3.0: an open source software for multi-stage analysis in data mining", *International Journal of Computational Intelligence Systems*, vol. 10, pp. 1238-1249, 2017.

[42] A. Asuncion and D. Newman, "UCI machine learning repository", 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[43] D. Sheskin, *Handbook of Parametric and Non-parametric Statistical Procedures*, 2nd Ed. London, U.K.: Chapman and Hall, 2006.

[44] J. Hodges and E. Lehmann, "Ranks methods for combination of independent experiments in analysis of variance", *Annals of Mathematical Statistics*, vol. 33, pp. 482–497, 1962.

[45] S. Holm, "A simple sequentially rejective multiple test procedure", *Scandinavian Journal of Statistics*, vol. 6, pp. 65–70, 1979.

APPENDIX-A

Ten methods that have been selected to make the comparison are as follow:

1-NN: The results obtained by the 1-NN classifier without pre-processing have been shown as a base performance, and most of the rest methods should overcome it.

GPG: GPG method employs RGSA as a global searcher in order to find the best position of the prototypes [31].

LVQ3: Learning vector quantization is a competitive learning scheme, which uses a "reward-punishment" rule to update the position of prototypes [36]. The third version of this algorithm, LVQ3, reported the best results.

MGauss: This is an adaptive PG method considered in the framework of mixture modeling by Gaussian distributions, while assuming statistical independence of the features. The prototypes are chosen as the mean vectors of the optimized Gaussians, whose mixtures are fit to model each one of the classes [37].

HYB: The hybrid LVQ3 algorithm combines several data reduction techniques. In this method, first, the initial prototypes are selected by a support vector machine method, and then HYB invokes an LVQ3 phase to find the optimal position of the prototypes [38].

RSP3: Reduction by space partitioning splits the feature space, and defines new prototypes in each partition. This algorithm is proposed in order to avoid drastic changes in the form of decision boundaries associated with the original training set [39].

ENPC: The evolutionary nearest neighbor prototype classifier is a genetic-based technique that finds the optimal number of the prototypes as well as the location of these prototypes [40].

PSO: In this method, particle swarm optimization is applied in order to generate an optimal set of prototypes. The social behavior of biological organisms, the movement of bird flocking, motivates the PSO algorithm. Each potential solution of an optimization problem is a bird (or ''particle'') with a given velocity, flying trough the solution space. Each bird adjusts its flight according to its own flying experience and its companions' flying experience [26].

AMPSO: Adaptive Michigan particle swarm optimization is based on the particle swarm optimization in which each particle in the swarm represents one prototype in the solution. In AMPSO, the number of prototypes can adapt to the problem [27].

IPADE: Iterative prototype adjustment based on differential evolution determines the most proper number of prototypes per class through an incremental procedure, and adjusts their location during the evolutionary process [30]. In IPADE, each individual in the population encodes just one prototype, and the whole population is taken as the solution to the problem.

SSMA-SFLSDE: This method is a differential evolution-based approach for optimizing the positioning of the prototypes. The proposed SSMA-SFLSDE algorithm combines a prototype selection stage with an optimization of the position of prototypes [29].

In order to make a fair comparison between the algorithms, we should utilize the same number of function evaluations. Therefore, the same as our proposed method, for 1-PC, LVQ3, HYB, ENPC, PSO, AMPSO, IPADE, and DE algorithms, the number of iterations is set to 250, and for PSO, AMPSO, IPADE, and DE, the population size is equal to 20. For the other parameters, we have used the recommended parameters proposed by their respective authors. The configuration parameters are shown in Table A-1.

Table A-1. Parameters determination for comparison methods.

Algorithm	Parameters
GPG	Iterations = 250, Swarm size = 20, $G_0 = 1$
LVQ3	Iterations = 250, *alpha* = 0.1, Window Width = 0.2, *epsilon* = 0.1
MGauss	Particle Size = 5
HYB	Search Iterations = 250, Optimal search Iterations = 1000, *alpha* = 0.1, Initial *epsilon* = 0, Final *epsilon* = 0.5, Initial Window Width = 0, Final Window Width = 0.5, *delta* = 0.1, *delta* Window Width = 0.1, Initial Selecion = SVM
RSP3	Select Choice = diameter
ENPC	Iterations = 250
PSO	Iterations = 250, Swarm size = 20, $C_1 = 1$, $C_2 = 3$, $V_{max} = 0.25$, $W_{start} = 1.5$, $W_{end} = 0.5$
AMPSO	Iterations = 250, $C_1 = 1.0$, $C_2 = 1.0$, $C_3 = 0.25$, $V_{max} = 1$, W = 0.1, X = 0.5, $P_r = 0.1$, $P_d = 0.1$
IPADE	Iterations = 250, iterFSGSS = 8, iterSFHC = 20, $F_l = 0.5$, $F_u = 0.9$
SSMA-SFLSDE	Iterations = 250, Swarm size = 20, iterSFGSS = 8, iterSFHC = 20, $F_l = 0.1$, $F_u = 0.9$

یک روش ترکیبی تولید نماینده و انتخاب نماینده برای طبقه‌بند نزدیکترین همسایه مبتنی بر الگوریتم جستجوی گرانشی

محدثه رضایی* و حسین نظام‌آبادی‌پور

دانشکده مهندسی برق، دانشگاه شهید باهنر کرمان، کرمان، ایران.

ارسال ۲۰۲۰/۱۰/۲۱؛ بازنگری ۲۰۲۱/۰۹/۰۵؛ پذیرش ۲۰۲۱/۱۱/۱۸

چکیده:

در این مقاله، راه‌حلی برای مشکلات طبقه‌بند نزدیک‌ترین همسایه با استفاده از دو روش پردازش داده انتخاب نماینده و تولید نماینده ارائه می‌شود. اغلب از مزایای این دو روش به صورت جداگانه استفاده می‌شود، اما در این مقاله با استفاده از الگوریتم جستجوی گرانشی، دو روش ترکیبی پیشنهاد شده است که در آن‌ها مسائل تولید و انتخاب نماینده با یکدیگر در نظر گرفته شده‌اند. به منظور ارزیابی عملکرد طبقه‌بندی دو روش ترکیبی، آزمایش-هایی برای مقایسه روش‌های پیشنهادی با تعدادی از روش‌های موجود انجام شده است. نتایج بدست آمده نشان داد که دو روش ترکیبی عملکرد بهتری نسبت به اغلب روش‌های مورد مقایسه داشته‌اند.

کلمات کلیدی: طبقه‌بندی، الگوریتم نزدیک‌ترین همسایه، تولید نماینده، انتخاب نماینده، الگوریتم جستجوی گرانشی، روش ترکیبی.