Prog Artif Intell (2014) 2:205–215
DOI 10.1007/s13748-014-0053-6

REGULAR PAPER

An automatic methodology for construction of multi-classifier systems based on the combination of selection and fusion

Tiago P. F. de Lima · Adenilton J. da Silva · Teresa B. Ludermir · Wilson R. de Oliveira

Received: 10 December 2012 / Accepted: 10 January 2014 / Published online: 16 April 2014

Abstract We present a methodology for the automatic construction of multi-classifiers systems based on the combination of selection and fusion. The proposed methodology initially finds the optimum number of clusters for training data set and subsequently determines an ensemble for each cluster found. Self-organizing maps were used in the clustering phase, and multilayer perceptrons, in the classification phase. Adaptive differential evolution was used in order to optimize the parameters and performance of the techniques employed in the classification and clustering phases. The proposed methodology, called SFJADE, was applied on data compression of signals generated by artificial nose sensors and a variety of known classification tasks, including cancer, card, diabetes, glass, heart, horse, soybean, and thyroid. The experimental results shown that the SFJADE methodology had a better performance than some methods reported in related literature and significantly outperformed the methods commonly used to construct multi-classifier systems, for instance, bagging, boosting, and random subspaces.

T. P. F. de Lima (✉) · A. J. da Silva · T. B. Ludermir
Centro de Informática, Universidade Federal de Pernambuco, Av. Jornalista Anibal Fernandes, s/n, Cidade Universitária, CEP: 50740-560 Recife-PE, Brazil
e-mail: tpfl2@cin.ufpe.br

A. J. da Silva
e-mail: ajs3@cin.ufpe.br

T. B. Ludermir
e-mail: tbl@cin.ufpe.br

W. R. de Oliveira
Departamento de Estatística e Informática, Universidade Federal Rural de Pernambuco, Rua Dom Manoel de Mendeiros, s/n, Dois Irmãos, CEP: 52171-900 Recife-PE, Brazil
e-mail: wilson.rosa@gmail.com

Keywords Multi-classifier systems · Ensembles · Selection and fusion · Self-organizing maps · Multilayer perceptrons · Adaptive differential evolution

1 Introduction

Classification is a data mining technique used to predict group membership for data instances whose classes are unknown [35]. Pattern classification involves building a function that maps the input feature space to an output space of two or more than two classes. Decision trees, Bayesian models, and artificial neural networks (ANNs) are examples of effective classifiers in the field of pattern classification [29]. However, by the No Free Lunch theorem, there is no single classifier that can be considered optimal for all tasks. In an attempt to improve recognition performance of a single classifier, a common approach is to combine various classifiers forming what is called a multi-classifier systems (MCS) [36].

There are several reasons for combining multiple classifiers to solve a given learning task [29]. First, MCS, also known as ensembles or committees, exploits the idea that a pool of different classifiers, referred as experts, can offer complementary information about the patterns to be classified, improving the effectiveness of the overall recognition process. Second, in some cases, ensembles might not be better than the single best classifier but can diminish or eliminate the risk of picking an inadequate single classifier. Another reason for ensembles arises from limited representational capability of learning algorithms. It is possible that the classifier space considered for the task does not contain the optimal classifier.

Successful applications of MCS have been reported in various works in the literature, such as handwritten digit recognition [13], signature verification [5], image labeling [42], just

to name a few. Basically, most of these systems may take two approaches: selection and fusion (SF) [46]. In the classifier fusion, every classifier in the ensemble is used. Bagging [8], boosting [41], and the random subspace method (RSM) [22] are frequently used for the generation of members, while arithmetic rule (e.g., maximum, mean, median, minimum, product), majority vote or the use of another classifier are examples of strategies used to combine their decisions [37]. In classifier selection, each ensemble member is supposed to know well a part of the feature space and be responsible for objects in this part [29].

Most of the discussions and design methodologies of ensembles are devoted to fusion version and are concerned with how to achieve good performance by creating diversity measure and combination schema. Research is less common in selection methodology. Basically, for the selection scheme, a means of partitioning the feature space and estimating the performance of each classifier are required. In Woods' DCS-LA approach [46], for example, the classification accuracy is estimated in small regions of feature space surrounding an unknown test sample, and then the most locally accurate classifier is nominated to make the final decision. On the other hand, Kuncheva [27] presents an algorithm where the training data are clustered to form the decision regions, and a confidence interval is used to determine whether one or multiple classifiers should be used to make a final decision.

Non-automatic design of ensembles often involves a tedious trial-and-error process which might be appropriate where prior knowledge and an experienced expert are available, which might not be the case for many real-world tasks and are hard to find in practice [2]. The goal of ensembles design is to determine ensemble architectures automatically. In this paper, we report the performance of a novel automatic method, named SFJADE, to combine selection and fusion (SF) via adaptive differential evolution (JADE). JADE is a powerful stochastic real-parameter optimization algorithm in current use [16]. For the clustering phase, self-organizing maps (SOM) was chosen since it is a simple technique that has good performance [12]. For the classification phase, the attractiveness of ANNs stems from their many inherent characteristics, including nonlinearity, high parallelism, robustness, fault tolerance, learning, and their capability to generalize [21].

This paper is organized as follows. Section 2 gives the theoretical justification of selection and fusion, by means of clustering algorithms; Sect. 3 presents some related works, that show strengths and weaknesses in the development of ensembles; Sect. 4 presents the evolutionary algorithm used in the current work; Sect. 5 describes the basic idea of proposed methodology; Sect. 6 presents the experimental results; Finally, Sect. 7 presents some final considerations about the main topics covered in this work, including contributions reached and directions for future works.

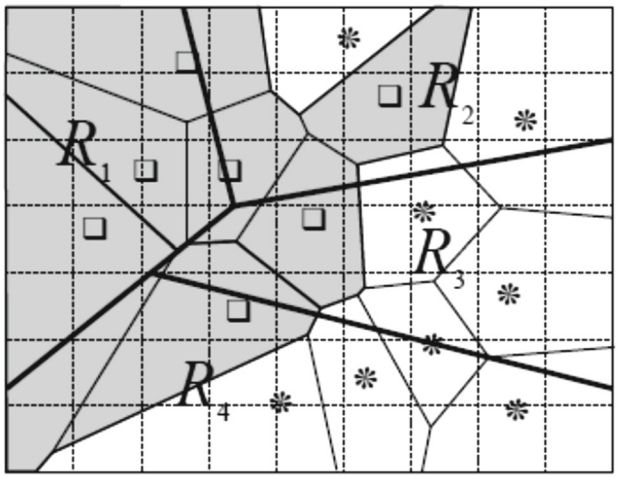

Fig. 1 An example of partitioning the feature space with two classification regions into four selection regions, from [27]

2 Theoretical justification

Let $D = \{D_1, D_2, \ldots, D_L\}$ be a set of L classifiers and $E = \{E_1, E_2, \ldots, E_M\}$ be a set of M ensembles formed from D. The construction of the system is realized by grouping the training set regardless of the class labels in $K > 1$ regions of competence denoted by $R_1, R_2, \ldots, R_K$, as show in Fig. 1.

For each region R_j, $j = 1, 2, \ldots, K$ is designated an ensemble from E which have the highest accuracy in R_j. Let $E^* \in E$ be the ensemble with the highest average accuracy over the whole features space. Denote by $P(E_i|R_j)$ the probability of correct classification by ensemble E_i in region R_j. Consider $E_{i(j)}$ the ensemble designated for region R_j. The overall probability of correct classification of a system is described in Eq. (1),

$$P_c = \sum_{j=1}^{K} P(R_j) P_c(R_j) = \sum_{j=1}^{K} P(R_j) P(E_{i(j)}|R_j) \tag{1}$$

where $P(R_j)$ is the probability that an input $\mathbf{x}$ drawn from the distribution of the task falls in R_j. To maximize P_c, we assign $E_{i(j)}$ so that

$$P(E_{i(j)}|R_j) \geq P(E_t|R_j), \quad t = 1, \ldots, M. \tag{2}$$

Thus, from Eqs. (1) and (2), we have that

$$P_c \geq \sum_{j=1}^{K} P(R_j) P(E^*|R_j) \tag{3}$$

Equation (3) shows that the combined scheme performs equal or better than the best ensemble E^*, regardless of the way the features space has been partitioned. However, the model might overtrain, giving a deceptively low training error. Hopefully, nominating an ensemble when it is better than the others will be a basis of a combination scheme less prone to overfitting and spurious errors.

3 Related works

The fusion method is widely used having several strategies that improves its performance. In contrast, no improvements have been proposed for the selection method, although their potential has already been proved in the literature [29]. In [46] is proposed the dynamic classifier selection by local accuracy (DCS-LA) where the accuracy of each classifier in the neighborhood of the test pattern is computed and the classifier with the best result is selected to give a final answer. The DCS-LA is capable of improving overall performance significantly. However, this method is time-consuming due to the accuracy estimation for each test sample calculated in real time.

Kuncheva [27] presents a method, named clustering-and-selection (CS), based on a probabilistic interpretation to statically select the best classifier. The training data set was clustered to form $k > 1$ decision regions, and a confidence interval was used to determine whether one or multiple classifiers should be used to make a final decision. The experiments showed the superior performance of the CS method when compared with some methods based on fusion methodology. However, the CS method has some drawbacks: The use of a non-automatic method and the reduced number of tasks employed in the experiments.

Liu and Yuan [31] present an algorithm based on CS, which can find in the feature space the regions where each classifier has best classification performance. In the clustering step, the feature space is partitioned by clustering separately the correctly and incorrectly classified training samples from each classifier, and the performances of the classifier in each region are calculated. In the selection step, the most accurate classifier in the vicinity of the input sample is nominated to provide the final decision. This method showed good results when compared with classical CS method, but the experiments were performed with a reduced number of tasks.

Kuncheva [28] present another important work that comprises a combination of classifier SF by using statistical inference to switch between the two. Selection is applied in those regions of the feature space where one classifier strongly dominates the others from the pool and fusion is applied in the remaining regions. Unfortunately, this model still used a non-automatic method and its real power was not explored. Even when using a non-automatic method, the results reported were better than those obtained with other methods, which encourage its use.

Dos Santos et al. [17] present a method that combines an optimization process and dynamic selection in a two-level selection phase to allow the selection of the most confident subset of classifiers to label each test sample individually. The optimization level is intended to generate a population of highly accurate candidate classifiers ensembles, while the dynamic selection level applies measures of confidence to reveal the candidate ensemble with the highest degree of confidence in the current decision.

Jackowski and Wozniak [24] present a method to split and select classifiers to compose an ensemble. Such method uses genetic algorithm to perform the feature space division and select the best classifiers existing in a pool of classifiers. An appropriated combination schema based on weighted majority voting is used. The experimental results reported pointed out that the proposed method has a better performance than single ANNs and classical ensembles. Nonetheless, the experiments were performed using only four tasks.

Lima et al. [30] presented an evolutionary method, named SFDEGL, based on a combination of selection and fusion. SFDEGL is composed by two phases: the first for designing the individual classifiers and the other for clustering patterns of training data set and search an ensemble for each cluster found. Differential evolution with global and local neighborhoods (DEGL) has been used in order to optimize the parameters and performance of the different techniques used in classification and clustering phases. The experimental results have shown that the SFDEGL method has better performance than non-automatic methods and significantly outperforms most of the methods commonly used to combine multiple classifiers for a set of four benchmark tasks. The work of [30] showed that there is a great deal of potential in the use of EA to automatic construct of ensembles.

4 Evolutionary algorithms

One way to solve optimization problems is to use exact methods. Unfortunately, these methods do not work properly in hard optimization problems. Another possibility is the application of heuristics, i.e., methods designed for a particular class of problems incorporating knowledge about the problem [44]. The drawback of this approach is the missing guarantee of optimality. But for some real-world applications, an approximate answer may be sufficient, even if it is not the best. Consequently, heuristics are widely used [7].

A keen observation of the underlying relation between optimization and biological evolution led to the development of an important paradigm of computational intelligence, the evolutionary computing techniques [16]. The evolutionary algorithms (EA) stands for a class of global optimization heuristics that search for optima using a process that is analogous to Darwinian natural selection [18]. The selection performs the "survival of the fittest" principle: population members with higher fitness are more likely to survive and participate in generating offspring. New candidate solutions are created through stochastic variation operators, which use the population information to direct the search to promising regions.

4.1 Differential evolution

Differential evolution (DE), proposed by Storn and Price [43], is a powerful, easy to use, fast, reliable EA that is used for tackling difficult optimization tasks. Since its creation, DE's reputation as an effective optimization algorithm has grown. One of the main advantages of the DE algorithm is that it has a small number of control parameters for adjustment which adds to its simplicity. Only three parameters need to be adjusted when optimizing a function: the step size known as the scale factor F, the crossover probability C_r, and the population size NP. Despite its simplicity, DE exhibits much better performance in comparison with several others EA on a wide variety of tasks including single-objective, multi-objective, unimodal, multimodal, separable, non-separable [16]. DE works through a cycle of generations, as presented in Algorithm 1.

In DE, diversity of the population is vital to avoid premature convergence. In general, selection tends to decrease the diversity of the population, whereas mutation increases it. The DE algorithm has three control parameters that maintain this diversity: the mutation scale factor F , the crossover control parameter C_r, and the population size N_P. These are fixed values for the duration of the algorithm. Proper tuning of these parameters is essential for the reliable performance of the algorithm. Trying to tune these three main control parameters and finding bounds for their values has been a topic of intensive research [32].

Algorithm 1: Differential Evolution

```
1  begin
2      Create a random initial population P of NP individuals
3      Evaluate each individual
4      while termination criterion not met do
5          for i = 1 to NP do
6              Select basis vector X_{basis,G}
7              Randomly choose X_{r1,G} ≠ X_{basis,G}
8              Randomly choose X_{r2,G} ≠ X_{r1,G} ≠ X_{basis,G}
9              Calculate the vector donor
               V_{i,G} = X_{basis,G} + F(X_{r1,G} − X_{r2,G})
10             Generate j_rand = randint(1,D)
11             for j = 1 to D do
12                 if j = j_rand or rand(0, 1) < CR then
13                     U_{j,i,g} = V_{j,i,g}
14                 else
15                     U_{j,i,g} = X_{j,i,g}
16                 end
17             end
18             if f(X_{i,G}) ≤ f(U_{i,G}) then
19                 X_{i,G+1} = X_{i,G}
20             else
21                 X_{i,G+1} = U_{i,G}
22             end
23         end
24     end
25 end
```

The mutation scale factor F controls the robustness and speed of the search. A lower value for F increases the convergence rate, but it does so at the risk of getting stuck into a local optimum and therefore failing to find the global solution. High values of C_r favor a higher mutated element crossover to current elements; as a result, the mutation factor F has a greater impact on the search. An increased value of N_P increases the diversity of the population and with it the potential to find the true optimal solution from the greater search space but at the cost of longer computation time.

The rule-of-thumb values for the control parameters given by Storn and Price [43] for F are usually between 0.5 and 1.0 and C_r between 0.8 and 1.0. These authors have proposed that the population size NP should be between $5D$ and $10D$, where D is the dimension of the problem. The suggestions by Storn and Price [43] for the control parameters are valid for many practical purposes but still lack generality. This means that, in practice, many time-consuming trial runs are required to find optimal parameters for each problem setting. As a result of the difficulty of setting appropriate control variables, research has focused on finding parameters such as F and C_r settings automatically [48].

4.2 Adaptive differential evolution

It is expected that solving a complex optimization problem itself should not be very difficult. In addition, an optimization algorithm should be able to converge to the true optimum for a variety of different tasks. Furthermore, the computing resources spent on searching for a solution should not be excessive. Thus, a useful optimization method should be easy to use, reliable, and efficient to achieve satisfactory solutions. JADE is such an optimization approach that addresses these requirements.

The JADE algorithm was proposed to improve optimization performance of DE by implementing a new mutation strategy, "*DE/target-to-pbest*" with optional external archive and updating control parameters in an adaptive manner [48]. The "*DE/target-to-pbest*" is a generalization of the classic "DE/target-to-best", while the optional archive operation utilizes historical data to provide information of progress direction. Both operations diversify the population and improve the convergence performance. The parameter adaptation automatically updates the control parameters to appropriate values and avoids a users prior knowledge of the relationship between the parameter settings and the characteristics of optimization problems. The JADE algorithm is described in algorithm 2.

5 Proposed method

The design of ensembles involves the choice of parameters as the model, number of components, combination

Algorithm 2: JADE

```
begin
    Set μ_CR = 0.5; μ_F = 0.5; A = ∅
    Create a random initial population P of NP individuals
    while termination criterion not met do
        S_F = ∅; S_CR = ∅
        for i = 1 to NP do
            Generate CR_i = randn_i(μ_CR, 0.1),
            F_i = randc_i(μ_F, 0.1)
            Randomly choose x^p_best,g as one of the 100p % best
            vectors
            Randomly choose x_r1,g ≠ x_i,g from current
            population P
            Randomly choose x_r2,g ≠ x_r1,g ≠ x_i,g from P ∪ A
            v_i,g = x_i,g + F_i(x^p_best,g − x_i,g) + F_i(x_r1,g − x_r2,g)
            Generate j_rand = randint(1, D)
            for j = 1 to D do
                if j = j_rand or rand(0, 1) < CR_i then
                    u_j,i,g = v_j,i,g
                else
                    u_j,i,g = x_j,i,g
                end
            end
            if f(x_i,g) ≤ f(u_i,g) then
                x_i,g+1 = x_i,g
            else
                x_i,g+1 = u_i,g; x_i,g → A;CR_i → S_CR; F_i → S_F
            end
        end
        Randomly remove solutions from A so that |A| ≤ NP
        μ_CR = (1 − c) · μ_CR + c · mean_A(S_CR)
        μ_F = (1 − c) · μ_F + c · mean_L(S_F)
    end
end
```

schema, and learning algorithms. In the same way, the use of ANNs involves a set of parameters that directly affects the final performance. Therefore, the design of ANN ensembles requires high number of parameters that need to be adjusted; the need for a priori knowledge on the problem domain and functioning ANN to define these parameters; and an expert when such knowledge is lacking. In some cases, the choice of parameters is manually performed through a trial-and-error method, which is a tedious, less-productive, and error-prone task. Furthermore, when the complexity of the problem domain increases and near-optimal networks are desired, non-automatic searching becomes harder and unmanageable [3].

Recently, there has been an increase in the use of EAs for the optimization problems, including the automated design of ANN ensembles. Some motivations for such research come from successful works that prove the power of EA to optimize ANNs for a large class of tasks [10,11]. In this way, EA are usually employed with the purpose of finding a set of ANNs, as diverse and accurate as possible, for composition of the ensemble.

5.1 Composition and functioning of the methodology

In Fig. 2, the flowchart of our methodology is shown, in which, given a task, in the first step, the data set is divided into training, validation, and testing. Subsequently, in the second and third steps, the search for optimized clustering maps and near-optimal ANNs take place, respectively. The parameters of each step were defined after many executions, and they were fixed for all tasks.

A functioning ensemble built with SFJADE method is shown in Fig. 2 with solid line beginning in step 1 and ending in ensemble output. The testing data set is submitted to clustering technique and the nearest cluster to data input will emit a supervised response through the fusion of all components in their respective pool. In what follows, the search process of clustering maps and ANNs will be described.

5.2 Search by clustering maps

In the second step of our method, the clustering of training data set, not considering the labels, occurs in order to discover unknown clusters and its appropriate amount. Clustering indicates the act of partitioning a data set into groups of similar objects [15].

It is worth mentioning that data clustering is an NP-hard problem when the number of clusters exceeds three [9]. Therefore, many researchers have employed EA to perform the search for the appropriated number of clusters size and

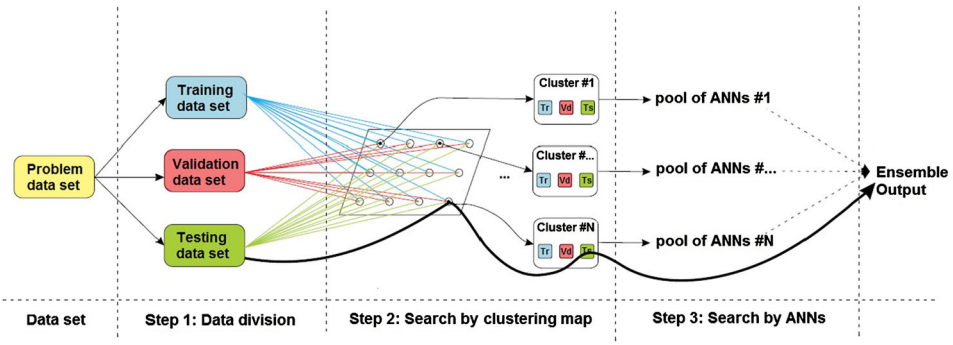

Fig. 2 Composition and functioning of SFJADE

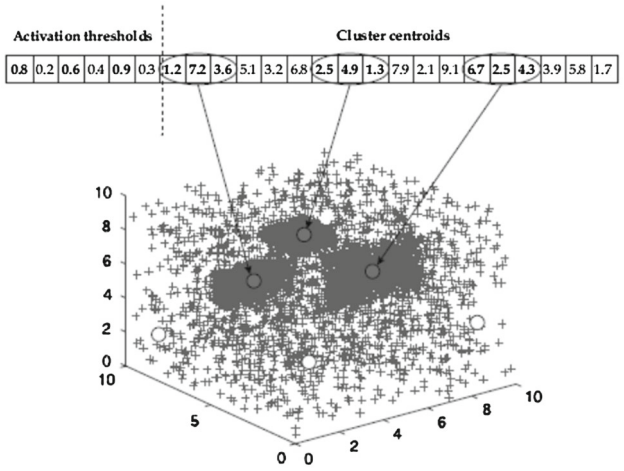

Fig. 3 Encoding scheme used in the SFJADE, adapted from [15]. Six cluster centers are encoded for a 3-D data set and activated cluster centers are shown as *gray-filled circles*

weights adjusted according to some standard learning algorithm [23]. In this way, the JADE algorithm was used for 50 generations with 35 individuals. With the use of JADE, an encoding schema and fitness function were defined.

5.2.1 Encoding schema

In the second step of SFJADE, the first $K_{\max}$ entries are floating points in [0, 1] which control whether the corresponding cluster is to be activated or not. The remainder entries are reserved for $K_{\max}$ cluster centers, each one with D dimension. In this way, the vector has the dimension of $K_{\max} + K_{\max} \times D$. In Fig. 3, this encoding schema is shown with three active clusters centers. Such encoding schema was proposed by Das et al. [15].

For each ith individual from the population used in JADE, the jth cluster center is active or selected for partitioning the associated data set if $T_{ij} > 0.5$; otherwise, the jth cluster is inactive. The T_{ij}'s values are called activation thresholds, because they govern the selection of active cluster centers [15]. If due to mutation some threshold T_{ij} in an offspring exceeds 1 or becomes negative, it is forcefully fixed to 1 or 0, respectively. In the same way, all first $K_{\max}$ values may become smaller than 0.5, in this case, randomly, a number of thresholds greater than two are selected and reinitialized to a random value >0.5.

5.2.2 Fitness function

The issue of cluster quality is a complicated one. Typically, two evaluation criteria are used: resolution and topology preservation [25,26]. The map resolution measure is the average distance between each data vector and the best match unit (BMU). For topology preservation, a measure of topographic error is used, and the proportional of all the data vectors for which first and second BMUs are not adjacent units. Beyond the quality measures presented, the numbers of active and non-representative clusters (<10 examples) are also used for fitness computation. In this way, after structure is set, the clustering algorithm is executed for ten epochs, and for a given individual, the fitness I_{clustfit} is defined by Eq. (4).

$$I_{\text{clustfit}} = \alpha \times I_{\text{rslt}} + \beta \times I_{\text{tply}} + \gamma \times I_{\text{actv}} + \delta \times I_{\text{nrep}} \quad (4)$$

where I_{rslt} is the resolution error, I_{tply} is the topological error, I_{actv} represents the perceptual number of active cluster over the maximum number of active clusters allowed, and I_{nrep} is the perceptual number of clusters that has no representative examples in validation data set. All components of Eq. (4) have values between [0.1] and the constants α, β, γ, and δ are used to control the contribution of each one over the fitness value, and consequently, to guide the search process to find solutions with an equilibrium between such information. We assigned $\alpha = 0.05$, $\beta = 0.05$, $\gamma = 0.1$, and $\delta = 0.8$ which indicates that SFJADE method finds optimized clustering maps by initially taking into account whether the cluster found will have representative samples, subsequently, the number of clusters is considered, and finally, the method prefers solutions with an equilibrium between the error measures.

5.3 Search by ANNs

The universal approximation capability presented by ANNs has been demonstrated over the years by the successful use in many types of problem with different degrees of complexity and in different fields of application [21]. However, the success of an ANNs application implies considerable time spent choosing a set of parameters that contribute toward improving the final performance. Training algorithms rates, the amount of hidden layers and nodes, and transfer functions are normally selected through a non-automatic process of trial-and-error that often fails to find the best possible set of ANN parameters for a specific task [3].

The hybridization of JADE and ANN was performed to build an automatic method capable of seeking near-optimal or even optimal neural networks for a given problem, thereby avoiding considerable human effort and difficulties stemming from a non-automatic trial-and-error search. An optimal neural network can be seen as an ANN tailored to a specific problem, thus having a smaller architecture with faster convergence and a better generalization performance [3]. Moreover, a near-optimal ANN is a neural network with the specific and correct parameters chosen to a particular task having a structure and final performance better than an ANN

Training Algorithm	Algorithm Parameters	Hidden Layers	Hidden Neurons	Transfer Function

Fig. 4 Composition of an individual in ANN optimization

discovered through the trial-and-error method [1,3]. With the use of the JADE algorithm, an encoding schema and fitness function were defined.

5.3.1 Encoding schema

The third step has a special encoding schema that comprises all ANN parameters needed for its functioning. The individual encodes the information of learning algorithm, algorithm parameters, hidden layers, hidden neurons, transfer function, as illustrated in Fig. 4.

The first part of the individual corresponds to the type of training algorithm used and its size is determined by the number of algorithms included in the search process. For this, part were considered four algorithms: backpropagation (BP) [40], resilient backpropagation (RPROP) [39], Levenberg-Marquardt (LM) [20], and scaled conjugate gradient (SGC) [34]. The BP algorithm is used to train the ANN when the highest value is in the first attribute of this part; the RPROP algorithm is used when the highest value is in the second attribute; and so on.

The second part involves the parameter values from the learning algorithm specified in the previous part. Each parameter has a predetermined position; therefore, when the algorithm is chosen, it is possible to recover: learning rate (l_r) and momentum (α), for BP; learning rate (l_r), increment to weight change (delt_{inc}), decrement to weight change (delt_{dec}), initial weight change (delt_0) and maximum weight (delt_{max}), for RPROP; initial Mu (μ), Mu decrease factor (μ_{dec}), Mu increase factor (μ_{inc}) and maximum Mu (μ_{max}), for LM; and change in weight for second derivative approximation (δ) and regulating the indefiniteness of Hessian (λ), for SGC. All these parameters have real values with the intervals described in Table 1, but they are not directly encoded. They are initialized between $[-1.0; 1.0]$ and a linear map is used to obtain the real values of the parameters.

The third part contains information about the numbers of hidden layers. According to Cybenko [14], from extension of the Kolmogorov theorem, we need at most two hidden layers, with a sufficient number of units to produce any mappings. It was also proved by Cybenko [14], that only one hidden layer is sufficient to approximate any continuous function. Nevertheless, in complex tasks, the use of three hidden layers can facilitate and improve the generalization. Therefore, in this work, the ANNs have a maximum of three hidden layers. To determinate the number of hidden layers is considered the attribute with the highest value in this part.

Table 1 Training algorithm parameters

Parameter	Min	Max
l_r	6×10^{-3}	1.4×10^{-2}
α	0	1
delt_{inc}	1	2
delt_{dec}	0.3	0.7
delt_0	0.042	0.098
delt_{max}	30	6×10^9
μ	6×10^{-4}	1.4×10^{-3}
μ_{dec}	6×10^{-2}	1.4×10^{-1}
μ_{inc}	6	14
μ_{max}	6×10^9	1.4×10^{10}
δ	3×10^{-5}	7×10^{-5}
λ	3×10^{-7}	7×10^{-7}

The fourth part encodes the number of neurons in each hidden layer. This part considers the maximum number of neurons per layer equal to n. Therefore, this part has three sections with dimension $3n$. The first n attributes correspond to number of hidden neurons in the first layer; the next n attributes correspond to the number of hidden neurons in the second layer; and the last n attributes correspond to the number of hidden neurons in the third layer. The number of hidden neurons is defined by the position of the attribute with the highest value in their respective section. The literature states that the best networks are those with a small number of neurons [6,21], so we use the maximum number of neurons $n = 12$.

The fifth part encodes the transfer function in each hidden layer, following the same principle as that of the fourth part. As three types of transfer functions are used in the search process, this part is divided into three sections, each one with three components. Each section refers to a hidden layer, and the highest value within it specifies the type of transfer function of the layer. If this value is the first attribute, then the linear function is used; if this value is the second attribute, then the tangent hyperbolic function is used; otherwise, logistic sigmoid function is used.

5.3.2 Fitness function

The information of each part of the individual is decoded to form the neural network. Furthermore, an arbitrary number of subspaces (without replacement) are selected from the original feature space. This randomization should create classifiers that are complementary. After structure is set, the neural network is trained with the training data set for up to 100 epochs, and for the fitness, I_{fit} is considered Eq. (5).

$$I_{\text{fit}} = \theta \times I_{\text{ve}} + \vartheta \times I_{\text{te}} + \iota \times I_{\text{hid}} + \kappa \times I_{\text{nod}} + \nu \times I_{\text{func}} \quad (5)$$

Equation 5 is composed by: validation error (I_{ve}); training error (I_{te}); number of hidden layers (I_{hid}); number of hidden nodes (I_{nod}); and weight of transfer functions (I_{func}) used. Each transfer function has an empirically determined weight: linear with 0.2, tangent hyperbolic with 0.3, and sigmoid logistic with 0.5.

The constants θ, ϑ, ι, κ, and ν of Eq. (5) have values between [0, 1] and control the influence of the respective factors upon the overall fitness calculation process. To favor classification accuracy regarding validating and training, the constants are defined empirically by Almeida and Ludermir [3] as follows: $\theta = 0.8, \vartheta = 0.145, \iota = 0.03, \kappa = 0.005$, and $\nu = 0.02$. These definitions imply that when apparently similar individuals are found, those that have the least training error, structural complexity, and transfer function complexity will prevail. The I_{ve} and I_{te} values are calculated using Eq. (6).

$$I_e = \frac{100}{PN} \sum_{i=1}^{P} \sum_{j=1}^{N} \left(d_{ij} - o_{ij}\right)^2 \tag{6}$$

In Eq. (6), N and P are the total number of outputs and number of patterns, respectively; d and o are the desired output and network output, respectively.

6 Experiments and results

The experiments were conducted using a real problem in which data compression of signals generated by artificial nose sensors are used to classify gases derived from oil (propane, butane, methane, and ethane) [47] and eight well-known benchmarks classification tasks found in the UCI repository [4]. The characteristics of these tasks are summarized in Table 2, which shows considerable diversity in the number of examples, attributes, and classes among the problems.

The list of methods considered in this work can be categorized into those based on SFJADE, classical methods, and methods from literature. Six versions of SFJADE were implemented in Matlab 2012a: SFJADE-MAX, SFJADE-MEAN, SFJADE-MEDIAN, SFJADE-MIN, SFJADE-PROD, and SFJADE-VOTE. The different variants of SFJADE method were suggested to investigate the best fusion technique. The classical methods considered in these experiments, executed in Java language with Weka 3.6.8, were Adaboost (ADBO) [19], Bagging (BAG) [8], MultiBoostAB (MBAB) [45], RSM [22], and single multi-layer perceptron (MLP) [21]. The values of parameters for classical methods were chosen as default from Weka 3.6.8.

Table 2 Summary of tasks used in the experiments

Task	Examples	Attributes	Classes
Artificial nose	4668	16	4
Cancer	699	9	2
Card	690	51	2
Diabetes	768	8	2
Glass	214	9	6
Heart	920	35	2
Horse	364	58	3
Soybean	683	82	19
Thyroid	7200	21	3

Table 3 Comparison of SFJADE versions

Task	Max	Mean	Median	Min	Prod	Vote
Cancer	0.047	**0.037**	**0.037**	0.047	0.049	0.038
Card	0.154	**0.138**	0.140	0.162	0.164	0.139
Diabetes	0.247	**0.241**	0.243	0.244	0.242	0.242
Glass	0.339	**0.309**	0.313	0.370	0.341	0.316
Heart	0.185	**0.172**	0.173	0.180	0.184	0.174
Horse	0.377	0.345	0.343	0.379	0.390	**0.340**
Thyroid	0.017	**0.013**	0.014	0.018	0.019	0.014
Soybean	0.115	0.063	0.063	0.114	0.105	**0.062**

To perform the experiments, we used 30 twofold iterations. For each fold, the data were randomly divided with stratification into 70 % for training and 30 % for testing. The first part was used in the construction of model (70 % for training and 30 % for validation) and the other part was used only to test the final solution (testing data set). Table 3 shows the error rate in test set of SFJADE variants, minimum values for each task are boldfaced. The SFJADE-MEAN has the lowest error rates for most of the tasks presented, except in horse and soybean data sets, and it is always statistically equivalent to the best combination method. From now, we will use only SFJADE-MEAN variant.

Table 4 presents the performance of some classical methods. In each line, according to the bootstrap hypothesis test, the boldface result means that the method has the smallest classification error for the task, and it is statistically better (with significance of 10 %) than all classical methods in italic. The results were expressed as the mean of classification error in the test set (standard deviation). There are more tasks where SFJADE-MEAN has statistically better performance than all classical methods (6 out 9 of comparisons). This shows the potential of the SF when EA are correctly employed to optimize clustering and classification techniques. However, the disadvantage of the SFJADE method, as demonstrate in Table 5, is that performing the search is very time-consuming in comparison with the classical methods. Time is measured in minutes on a computer with Microsoft Windows operational system, 8 GB of RAM and a 3.4 GHz Intel Core i7 processor .

Table 4 Comparison between classical methods

	SFJADE	[19]	[8]	[45]	[22]
Artificial	**2.3E-3**	4.81	67.76	4.83	11.21
Nose	**(1.3E-2)**	(1.68)	(11.56)	(1.68)	(1.57)
Cancer	3.70	4.85	4.27	4.43	**3.28**
	(1.00)	(1.42)	(1.17)	(1.19)	**(0.94)**
Card	**13.86**	17.42	15.25	17.15	16.04
	(1.90)	(2.17)	(1.96)	(2.13)	(2.67)
Diabetes	**0.241**	0.268	0.253	0.249	0.252
	(1.93)	(2.82)	(2.69)	(2.50)	(2.10)
Glass	**30.87**	34.97	35.18	34.3 1	35.13
	(5.00)	(5.27)	(5.09)	(5.02)	(4.93)
Heart	**17.27**	21.13	19.11	19.20	18.07
	(1.95)	(1.75)	(1.84)	(2.09)	(2.07)
Horse	*34.53*	35.87	**34.04**	35.75	*34.53*
	(3.03)	(4.02)	**(4.50)**	(4.57)	*(3.94)*
Soybean	**6.26**	7.49	21.64	7.49	*6.26*
	(1.74)	(1.60)	(3.16)	(1.60)	*(1.61)*
Thyroid	**1.38**	3.95	12.35	3.99	4.33
	(0.29)	(1.68)	(1.16)	(1.69)	(1.58)

Table 5 Average time (in minutes) of processing for each execution

	SFJADE	[19]	[8]	[45]	[22]	MLP
Art. nose	89.88	0.66	1.47	0.86	0.77	**0.15**
Cancer	9.85	0.04	0.04	0.04	0.03	**0.01**
Card	11.65	0.26	0.61	0.29	0.20	**0.06**
Diabetes	8.75	0.04	0.05	0.05	0.03	**0.01**
Glass	6.32	0.02	0.03	0.03	0.02	**0.01**
Heart	13.19	0.25	0.40	0.36	0.15	**0.04**
Horse	9.65	0.24	0.43	0.31	0.14	**0.04**
Soybean	27.20	0.48	2.00	0.48	0.81	**0.20**
Thyroid	97.00	0.48	1.60	0.50	0.73	**0.16**
Average	*30.39*	*0.23*	*0.65*	*0.26*	*0.26*	*0.06*

The length of time required is the main disadvantage of SFJADE method. While SFJADE takes on average more than 30 mins, the classical methods take <1 min to perform the same task. This huge time difference can be explained by the EA use and also by complexity of the classification and clustering techniques. For high-dimensional problems, the SFJADE method may need a long time. Thus, for applications in which time is an issue, SFJADE should not be used. SFJADE could be applied in tasks where training can be executed offline.

7 Conclusion

The work described in this paper shown, once more, that EA represents a suitable technique to solve the problem of ensemble design by yielding more than satisfactory results on a set of 8 benchmark tasks. Furthermore, in artificial nose task, the mean test error of SFJADE was 2,000 times smaller than in the best classical method used. In comparison with the result obtained in [47], SFJADE also got the best result with an improvement of 326.09 in the mean test error, but one have to be careful to compare these results because of the different experimental methodology. A key advantage of the approach proposed, which is computationally quite heavy, is the reduction in the effort required from a human expert, the most expensive resource of all, to design and set up an ensemble for a given task. Nevertheless, computation time could be further reduced by using different programming languages, as C++ or Java. Furthermore, nowadays, the monetary cost of machine time is rapidly decreasing and nothing suggests a reversal of this trend is in sight in the foreseeable future. The computational effort required by the SFJADE approach can be easily accommodated in distributed machines, able to perform parallel processing, thus increasing the performance of the computational model that has to be implemented.

To conclude and attempt to establish a working model to guide future research, this section suggests some topics, primarily for efficiency and robustness improvements of SFJADE. The extensions of this work might include:

- To perform more experiments with other types of tasks, such as time-series, for the verification of the behavior of SFJADE in prediction tasks.
- The automatic choice of the techniques/parameters through a meta-learning algorithm [33,38]. The experiments performed in this paper did not reveal a method of adjusting techniques/parameters, suggesting that these are dependent on the task.
- The multimodal and multi-objective techniques can create diversity necessary in evolution of ANNs to produce ensembles with good performance. The ability of algorithms to discover and maintain multiple optima is of great importance, in particular when several global optima exist or when other high-quality solutions might be of interest.
- The exploration of multiprocessor architectures to improve both the speed and the performance of SFJADE method. JADE, like other EAs, can be parallelized due to the fact that each individual of the population is evaluated independently. The only phase of the algorithm that requires communication with other individuals is reproduction. This phase can also be parallelized for pairs of individuals.

References

1. Abraham, A.: Meta learning evolutionary artificial neural networks. Neurocomputing **56**, 1–38 (2004)
2. Akhand, M.A.H., Murase, K.: Adaptive ensemble construction based on progressive interactive training of neural networks. Int. J. Mach. Learn. Comput. **2**(3), 283–286 (2012)

3. Almeida, L.M., Ludermir, T.B.: A multi-objective memetic and hybrid methodology for optimizing the parameters and performance of artificial neural networks. Neurocomputing **73**(7), 1438–1450 (2010)
4. Bache, K., Lichman, M.: UCI machine learning repository. University of California, School of Information and Computer Science, Irvine, CA (2013). http://archive.ics.uci.edu/ml
5. Bertolini, D., Oliveira, L., Justino, E., Sabourin, R.: Reducing forgeries in writer-independent off-line signature verification through ensemble of classifiers. Pattern Recognit. **43**(1), 387–396 (2010)
6. Bishop, C.M., et al.: Neural Networks for Pattern Recognition. Clarendon press Oxford, Oxford (1995)
7. Blickle, T.: Theory of Evolutionary Algorithms and Application to System Synthesis, vol. 17. Hochschulverlag (1997)
8. Breiman, L.: Bagging predictors. Mach. learn. **24**(2), 123–140 (1996)
9. Brucker, P.: On the complexity of clustering problems. Optim. Oper. Res. **157**, 45–54 (1978)
10. Bullinaria, J.A.: Evolving neural networks: is it really worth the effort. In: Proceedings of the European Symposium on Artificial Neural Networks, Citeseer, pp. 267–272 (2005)
11. Cantú-Paz, E., Kamath, C.: An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. IEEE Trans. Syst. Man Cybern. Part B Cybern. **35**(5), 915–927 (2005)
12. Chen, Y., Qin, B., Liu, T., Liu, Y., Li, S.: The comparison of som and k-means for text clustering. Comput. Inf. Sci. **3**(2), P268 (2010)
13. Cruz, R.M., Cavalcanti, G.D., Ren, T.I.: An ensemble classifier for offline cursive character recognition using multiple feature extraction techniques. In: The 2010 IEEE International Joint Conference Neural Networks (IJCNN), pp. 1–8 (2010)
14. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. (MCSS) **2**(4), 303–314 (1989)
15. Das, S., Abraham, A., Konar, A.: Automatic clustering using an improved differential evolution algorithm. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. **38**(1), 218–237 (2008)
16. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evolutionary Comput. **15**(1), 4–31 (2011)
17. Dos Santos, E.M., Sabourin, R., Maupin, P.: A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. Pattern Recognit. **41**(10), 2993–3009 (2008)
18. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin (2008)
19. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference, pp. 148–156. Morgan Kauffman (1996)
20. Hagan, M.T., Menhaj, M.B.: Training feedforward networks with the marquardt algorithm. IEEE Trans. Neural Netw. **5**(6), 989–993 (1994)
21. Haykin, S.: Neural networks and learning machines, 3rd edn. Pearson Education, Upper Saddle River (2009)
22. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. **20**(8), 832–844 (1998)
23. Hruschka, E.R., Campello, R.J., Freitas, A.A., De Carvalho, A.P.L.F.: A survey of evolutionary algorithms for clustering. IEEE Trans. Syst. Man Cybern.Part C Appl. Rev. **39**(2), 133–155 (2009)
24. Jackowski, K., Wozniak, M.: Algorithm of designing compound recognition system on the basis of combining classifiers with simultaneous splitting feature space into competence areas. Pattern Anal. Appl. **12**(4), 415–425 (2009)
25. Kiviluoto, K.: Topology preservation in self-organizing maps. In: IEEE International Conference on Neural Networks, vol. 1, pp. 294–299 (1996)
26. Kohonen, T.: Self-Organizing Maps, vol. 30. Springer, Berlin (2001)
27. Kuncheva, L.I.: Clustering-and-selection model for classifier combination. In: Proceedings of IEEE Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, vol. 1, pp. 185–188 (2000). doi:10.1109/KES.2000.885788
28. Kuncheva, L.I.: Switching between selection and fusion in combining classifiers: an experiment. IEEE Trans. Syst. Man Cybern. Part B Cybern. **32**(2), 146–156 (2002)
29. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience, New York (2004)
30. Lima, T.P., Silva, A.J., Ludermir, T.B.: Selection and fusion of neural networks via differential evolution. In: Pavn, J., Duque-Mndez, N., Fuentes-Fernndez, R. (eds.) Advances in Artificial Intelligence IBERAMIA 2012. Lecture Notes in Computer Science, vol. 7637, pp. 149–158. Springer, Berlin Heidelberg (2012)
31. Liu, R., Yuan, B.: Multiple classifiers combination by clustering and selection. Inf. Fusion **2**(3), 163–168 (2001)
32. Mandal, K., Chakraborty, N.: Differential evolution technique-based short-term economic generation scheduling of hydrothermal systems. Electric Power Syst. Res. **78**(11), 1972–1979 (2008)
33. Miranda, P.B.C., Prudencio, R.B.C., Carvalho, A.C.P.L.F., Soares, C.: Combining meta-learning with multi-objective particle swarm algorithms for svm parameter selection: an experimental analysis. In: Symposium on Brazilian Neural Networks (SBRN), 20–25 Oct, pp. 1–6 (2012). doi:10.1109/SBRN.2012.12
34. Møller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw. **6**(4), 525–533 (1993)
35. Phyu, T.N.: Survey of classification techniques in data mining. In; Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, pp. 18–20 (2009)
36. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits Syst. Mag. **6**(3), 21–45 (2006)
37. Ponti, M.P.: Combining classifiers: from the creation of ensembles to the decision fusion. In: 24th SIBGRAPI IEEE Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 28–30 Aug, pp. 1–10 (2011). doi:10.1109/SIBGRAPI-T.2011.9
38. Prudêncio, R.B., Ludermir, T.B.: Combining uncertainty sampling methods for supporting the generation of meta-examples. Inf. Sci. **196**, 1–14 (2012)
39. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: IEEE International Conference on Neural Networks, vol. 1 pp. 586–591 (1993). doi:10.1109/ICNN.1993.298623
40. Rumelhart, D.E., Hintont, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
41. Schapire, R.E.: The strength of weak learnability. Mach. learn. **5**(2), 197–227 (1990)
42. Singh, S., Singh, M.: A dynamic classifier selection and combination approach to image region labelling. Signal Process. Image Commun. **20**(3), 219–231 (2005)
43. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
44. Watanabe, K., Hashem, M.: Evolutionary algorithms: revisited. In: Evolutionary Computations, pp. 1–19. Springer, Berlin (2004)
45. Webb, G.I.: Multiboosting: a technique for combining boosting and wagging. Mach. learn. **40**(2), 159–196 (2000)
46. Woods, K., Kegelmeyer Jr, W.P., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. IEEE Trans. Pattern Anal. Mach. Intell. **19**(4), 405–410 (1997)
47. Zanchettin, C., Ludermir, T.: Wavelet filter for noise reduction and signal compression in an artificial nose. Appl. Soft Comput. **7**(1), 246–256 (2007

48. Zhang, J., Sanderson, A.C.: Jade: adaptive differential evolution with optional external archive. IEEE Trans. Evolutionary Comput. **13**(5), 945–958 (2009)