2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Ensembles of Evolutionary Extreme Learning Machines Through Differential Evolution and Fitness Sharing

Tiago. P. F. de Lima, and Teresa. B. Ludermir

***Abstract* — Extreme Learning Machine (ELM) is a single-hidden-layer feedforward neural network which has been applied into many real world pattern classification problems. Recently, ELMs have been built in an automatic way through evolutionary algorithms. Most works, nonetheless, do not uses all population obtained, but choose only one individual in the last generation. In an attempt to improve performance, an ensemble is a more promising choice because a pool of classifiers might produce higher accuracy than merely using the information from only one classifier among them. One of the most important factors for optimum accuracy is the diversity of the classifier pool. In this work, an enhanced Differential Evolution incorporating sharing function method is used to generate a pool of ELMs. Fitness Sharing that shares resources if the distance between the individuals is smaller than the sharing radius is a representative specification method, which produces diverse results than standard evolutionary algorithms that converge to only one solution. Experimental results on 14 well known benchmark classification tasks suggest that our method can generate ensembles that are more effective than ensembles solely through DE and traditional ensemble methods.**

I. Introduction

Machine learning has been developed and used in the past decades. Among its methods, algorithms and applications on Artificial Neural Networks (ANNs) have been widely researched. In 2004, Extreme Learning Machine (ELM) [1] was introduced as an efficient and practical learning algorithm used for single-hidden layer feedforward neural networks. ELM approach randomly generates input weights and hidden biases rather than tuning network parameters, making the learning speed thousands of times faster than traditional learning algorithms (such as back-propagation and its variants). On the other hand, ELM tends to require more hidden neurons than traditional tuning-based algorithms and lead to ill-condition problem due to the random determination of the input weights and hidden biases [2]. Thus, in [2], Zhu *et al*. used a hybrid learning algorithm to overcome the drawbacks of ELM, which uses an Evolutionary Algorithm (EA) to select the input weights and hidden biases and the Moore-Penrose (MP) generalized inverse is used to analytically calculate the output weights. Using this technique, they were able to achieve much more compact networks (with less hidden neurons) with good generalization performance. Since then, ELMs have been built in an automatic way through EAs [3-6].

Tiago. P. F. de Lima, and Teresa. B. Ludermir are with Centro de Informática, Universidade Federal de Pernambuco; Recife; PE; Brazil; 50740-560 (e-mails: {tpfl2,tbl}@cin.ufpe.br). This work was supported by FACEPE, CNPq, and CAPES (Brazilian Research Agencies).

Most works, nonetheless, do not uses all population obtained, but choose only one individual in the last generation. The selection of a single classifier can led to the choice of the worst classifier for future data. Especially when the data used to learn was not sufficiently representative in order to classify properly new objects, the test set provides just apparent errors $\hat{E}$ that differ from true errors E, in a generalization error: $\hat{E} = E \pm \Delta$. In an attempt to improve performance of single classifiers, a common approach is to combine multiple classifiers, forming ensembles.

Ensemble of classifiers, also known as Multi-classifier systems or committees, is a composite model, aggregating learning machines into one predictive model. An ensemble prediction, consequently, is a function of all included base models. Many studies showed that classification problems are often more accurate when using combination of classifiers rather than an individual base learner. For instance "weak" classifiers are capable of outperform a highly specific classifier [7]. The main motivation for using an ensemble is the fact that combining several models using averaging will eliminate uncorrelated base classifier errors, see e. g., [8]. This reasoning, however, requires the base classifiers to commit their errors on different instances - clearly there is no point in combining identical models.

Enhanced EAs incorporating multimodal techniques (which produce diverse results than standard EAs that converge to only one solution) can creates diversity in evolution of networks to produce ensembles with better performance. One class of such mechanisms goes by the name of Fitness Sharing. Sharing function method relies on a distance metric to cluster population. Individuals which are similar to each other are punished for this similarity by being required to share their fitness, while isolated individuals retain all the fitness value that they achieve.

This paper presents a hybrid system which combines the evolutionary algorithm Differential Evolution (DE) and the multimodal technique Fitness Sharing. The multimodal technique is inserted into the evolutionary algorithm to increase its exploration capacity and to improve its efficiency. The rest of this paper is organized as follows. Section II, defines terms and provides the main concepts to ELMs; Section III gives a brief review on the fundamentals of DE algorithm; The DE incorporating fitness sharing is presented in Section IV; Section V describes the hybrid learning system for evolving ELMs; Section VI presents the experimental results; Section VII concludes with a summary of the work.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

II. Extreme Learning Machines

Unlike the most known learning algorithms for ANNs, the main characteristic of ELMs is learning without iterative training, as proposed in [1]. Let $\mathcal{T} = \{(p_i, t_i) | p_i \in R^n, t_i \in R^m, i = 1,2,\dots,N\}$ be the training set, where p_i is an n-dimensional input pattern and t_i is a m-dimensional target. The training process is briefly described as follows.

Step 1: Randomly assign values to the input weights and the hidden neuron biases.

Step 2: The output weights are analytically determined through the generalized inverse operation of the hidden-layer matrices, as in (1), where a_i is the input weights, b_i is the hidden layers biases, β_i is the output weight that connects the i^{th} hidden node and output node, $f(.)$ is the activation function, L is the number of hidden neurons, and N is the number of distinct input or output data.

$$\sum_{i=1}^{L} \beta_i f(a_i, b_i, p_j) = t_j, j = 1,2,\dots,N \quad (1)$$

This is equivalent to $H\beta = T$, where

$$H = \begin{bmatrix} f(a_1, b_1, p_1) & \dots & f(a_L, b_L, p_L) \\ \vdots & \ddots & \vdots \\ f(a_1, b_1, p_N) & \dots & f(a_L, b_1, p_N) \end{bmatrix},$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}, \text{ and } T = \begin{bmatrix} t_1^T \\ \vdots \\ t_L^T \end{bmatrix}.$$

Step 3: Calculate the output weights by $\beta = H^+T$, where H^+ is the Moore-Penrose generalized inverse of H.

As analyzed by [1], ELM can reach good generalization performance by ensuring two properties of learning: the smallest norm of weights besides the smallest squared error within the training samples, while the gradient-based algorithms focus on the later property only. However, the randomness of weights and biases may lead to non-optimal performance. The search process of near-optimal ANNs is widely explored, using EAs [9]. In this approach, EAs and ANNs are combined to produce hybrid models with low error and high generalization, yielding evolutionary ANNs.

III. Differential Evolution

DE, proposed by [10], is a powerful, easy to use, fast, reliable EA that is used for tackling difficult optimization tasks. Since its inception, DE's reputation as an effective optimization algorithm has grown. One of the most important advantages of DE is that it has a small number of control parameters for adjustment, which adds to its simplicity. Despite its simplicity, DE exhibits much better performance in comparison with several other EAs on a wide variety of tasks including single-objective, multi-objective, unimodal, multimodal, separable, non-separable, and so on [11].

The DE algorithm creates a population of candidate solutions, chosen randomly in the search space, represented by $P_G = \{X_{i,G}, i = 1,2,\dots,NP\}$, where G is the index of the current generation, i is the index of the individual and NP is the population size. Each individual i is a D-dimensional vector represented as follows: $X_{i,G} = (x_{j,i,G}, j = 1,2,..,D)$, where $x_{j,i,G}$ is the attribute j of the individual i in generation G. All individuals are candidate solutions for the objective function $f(.)$ to be minimized. Algorithm 1 shows how DE affects the population until achieve the stopping criterion.

Algorithm 1 - Differential Evolution

```
// Inicialization
Create a random initial population P of NP individuals
// Evaluation
Evaluate each individual
while termination criterion not met do
  for i = 1 to NP do
    // Mutation
      Select basis vector X_{r1,G}
      Randomly choose X_{r2,G} ≠ X_{r1,G}
      Randomly choose X_{r3,G} ≠ X_{r2,G} ≠ X_{r1,G}
      Calculate the vector donor
      V_{i,G} = X_{r1,G} + F(X_{r2,G} − X_{r3,G})
    // Crossover
      Generate j_rand = randint(1, D)
      for j = 1 to D do
        if j = jrand or rand(0,1) ≤ Cr then
        U_{j,i,g} = V_{j,i,g}
        else
          U_{j,i,g} = X_{j,i,g}
        end if
      end for
    // Evaluation
      Evaluate the new individual U_{i,G}
    // Selection
      if f(U_{i,G}) ≤ f(X_{i,G})
        X_{i,G+1} = U_{i,G}
      else
        X_{i,G+1} = X_{i,G}
      end if
  end for
end while
return P
```

In this work we use the classical DE algorithm (*DE/rand/1/BIN*). The main operators of DE are:

Mutation: for each individual $X_{i,G}$, a mutant vector is defined by (2), where $r1, r2,$ and $r3$ are mutually different, randomly selected indices chosen from the range $[1, NP]$. F is the mutation factor, which provides the amplification to the difference between two individuals so as to avoid search stagnation and it is usually taken in the range of $[0,1]$.

$$V_{i,G} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}) \quad (2)$$

Crossover: in this step, each member of the population $X_{i,G}$ is crossed with a mutant vector $V_{i,G}$. They exchange attributes following the crossover probability $C_r \in [0,1]$, as shown in (3), in order to form the trial vector $U_{i,G}$.

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & if\ rand_j(0,1) \le C_r\ or\ j = j_{rand} \\ x_{j,i,G}, & otherwise \end{cases} \quad (3)$$

Selection: then the trial vector $U_{i,G}$ is evaluated and compared to the current vector $X_{i,G}$. The member for the next generation, at $G = G + 1$, is described in (4).

$$X_{i,G+1} = \begin{cases} U_{i,G}, & if\ f(U_{i,G}) \le f(X_{i,G}) \\ X_{i,G}, & otherwise \end{cases} \quad (4)$$

In each generation G, the parameters F and Cr were updated as in (5) and (6), respectively, where $rand(0,1)$ is a random number from the range [0,1], G_{max} is the maximum number of generations G.

$$F = 0.5 \times (1 + rand(0,1)) \quad (5)$$

$$Cr = (G_{max} - G)/G_{max} \quad (6)$$

IV. Fitness Sharing

The main goal of Fitness Sharing (FS) [12] is to distribute a population of individuals along a set of resources. When an individual $X_{i,G}$ is sharing resources with other individuals, its fitness $f(X_{i,G})$ is degraded in proportion to the number and closeness to individuals that surround it. The shared fitness for an individual $X_{i,G}$ is simply calculated as in (7).

$$fShar_i = \frac{f(X_{i,G})}{\sum_{j=0}^{NP} sharing_i^j} \quad (7)$$

The similarity between individuals $X_{i,G}$ and $X_{j,G}$ is measured by a distance function $sharing_i^j$, as in (8).

$$sharing_i^j = \begin{cases} 1 - (d_i^j / \sigma_{share})^{\alpha}, if\ d_i^j < \sigma_{share} \\ 0, otherwise \end{cases} \quad (8)$$

In (8), α is a constant to determine the shape of the sharing function, and σ_{share} means the sharing radius. If the difference of the individuals is large than σ_{share}, they do not share the fitness. Only the individuals who have smaller distance values than σ_{share} can share the fitness. The sharing radius is determined by a threshold value derived from (9), where d_i^j is a measure of distance between individual i and j.

$$\sigma_{share} = \frac{1}{2 \times NP^2} \sum_{i=1}^{NP} \sum_{j=1}^{NP} d_{ij} \quad (9)$$

In DE, since the evolution process aims at searching for the minimum value, therefore $fShar_i$ is modified as in (10).

$$fShar_i = f(X_{i,G}) \times \sum_{j=0}^{NP} sharing_i^j \quad (10)$$

The general flowchart of an enhanced DE algorithm incorporating sharing function method is illustrated in Fig. 1.

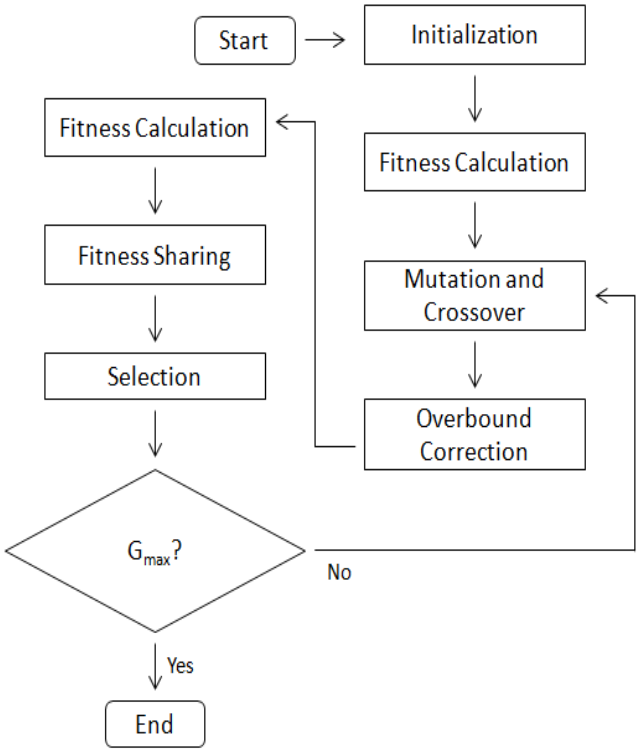

Fig. 1. General flowchart of the modified DE algorithm with FS.

V. Combining Classifiers

The research field of ensembles becomes very popular after the half of the 1990 decade, with many papers published on the creation of ensemble methods that provide some theoretical insights of why combining classifiers could be interesting. According to Dietterich [13], there are three main motivations to combine multiple classifiers, the best case, the worst case, and the computational motivations:

Representational (or best case) motivation: combination of multiple classifiers may have a better performance than the single best classifier among them. There are many experimental evidences that it is possible if the classifiers in an ensemble make different errors on a test set.

Statistical (or worst case) motivation: it is possible to avoid a wrong classifier by averaging several classifiers. It was confirmed theoretically in [13]. There is no guarantee, however, that the combination will perform better than from only one classifier among them.

Computational motivation: some algorithms perform an optimization task in order to learn and suffer from local minima. Algorithms such as the back-propagation are initialized randomly in order to avoid locally optimum solutions. In this case it is a difficult task to find the best classifier, and it is often used several (hundreds or even thousands) initializations in order to find a presumable optimal classifier. Combination of such classifiers showed to stabilize and improve the best single classifier result [13].

There are basically two principal approaches for combining classifiers: static and dynamic selection [14]. In the static approach, bagging [15], boosting [16], and random subspace [17], generates a pool of classifiers $C = \{C_1, C_2, \dots, C_l\}$, while arithmetic rule (e.g. maximum, mean, median, minimum, product), majority vote or another different classifier are examples of techniques used to combine their decisions. Given such a classifier pool, the dynamic selection has focused on finding the most relevant subset of classifiers D, for each query pattern, rather than combining all available l classifiers, where $|D| \leq |C|$.

VI. Proposed Method

The hybridization of DE and FS was performed to build an automatic method capable of seeking a diverse and accurate pool of ELMs. The DE was executed for $G_{max} = 1000$ generations. We use many generations because we wanted to provide enough time for a satisfactory fitness level to be achieved for the population. The population size used was $NP = 20$. A small number of individuals was used because we wanted to observe the effect of shared fitness function. If we use a very large number of individuals, the effect of shared fitness function could be hidden. With the use of DE, an encoding schema and objective function were defined. In encoding schema, an individual contains the ELM information organized in five parts, as illustrated in Fig. 2.

Inputs	Hidden Neurons	Activation Function	Input Weights	Hidden Biases

Fig. 2. Composition of an individual.

The first part of the individual is responsible for the process of selecting a subset of inputs extracted from the original set, in order to reduce the dimensionality of the problem and consequently the complexity of ELMs generated. The second part contains information on the hidden neurons. We use the minimum number of neurons $N_{min} = 10$ and the maximum number of neurons $N_{max} = 30$. Having too many hidden neurons is analogous to a system of equations with more equations than free variables: the system is over specified, and incapable of generalization. The third part encodes the activation function. We use the Gaussian radial basis function, hyperbolic tangent function, sigmoid function, sine function, and triangular basis function. The fourth and fifth parts correspond to the input weights and hidden biases (obtained in the range $[-1,1]$), respectively. The information of each part is decoded to form an ELM. After the structure is set, the MP generalized inverse is used to analytically calculate the output weights.

A multi-objective learning algorithm can take into account more than one objective (instead of using only the training dataset to avoid overfitting [2]). Thus, we adopt the most known error functions for the training and validation datasets, i.e., the root mean square error (RMSE) and classification error (CE), defined respectively in (11) and (12).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(t_{ij}-o_{ij})^2}{n \times m}} \tag{11}$$

$$CE = \sum_{i=1}^{c} C_i \tag{12}$$

In (11), m is the number of output units, t_{ij} is the target to pattern i in the output j, o_{ij} is the output obtained to the pattern i in the output j and n is the number of samples. In (12), c is the number of classes and C_i is the number of patterns misclassified per class. The RMSE and CE are rescaled using fitness sharing method, as in (10). To calculate the distance between the networks used to determine the σ_{share} and the value of $sharing_i^j$, we use the method of average output [18], as in (13) and (14), where n is the number of samples, m is the number of outputs units, $o_{io}(x_t)$ is the output of the o^{th} output node for the t^{th} input data.

$$O_{io} = \left(\sum_{t=1}^{n} o_{io}(x_t)\right)/_n \tag{13}$$

$$O_i = (O_{i1}, O_{i2}, \dots, O_{im}) \tag{14}$$

The distance between the two ELMs is the Euclidean distance of their average outputs. The similarity between ELM i and j can be calculated as in (15).

$$d_{ij} = \sqrt{\sum_{o=1}^{m}(O_{io} - O_{jo})^2} \tag{15}$$

VII. Experiments and Results

The experiments were conducted using 14 well-known benchmarks classification tasks found in [19]. The specifications of these tasks are summarized in Table I, which shows diversity in the number of examples, attributes and classes. All attributes have been normalized into the range $[0,1]$, while the targets have been normalized into $[-1,1]$.

TABLE I
Specification of the Tasks Used in the Experiments

Task	Examples	Attributes	Classes
Abalone	4177	8	3
Cancer	699	9	2
Car	1728	6	4
Diabetes	694	8	2
Ecoli	336	7	8
Glass	214	9	6
Iris	150	4	3
Pendigits	10992	16	10
Sat	6435	36	6
Sonar	208	60	2
Vehicle	846	18	3
Vowel	528	10	11
Wine	178	13	3
Yeast	1484	8	10

Each task was randomly divided into 50% for training, 25% for validation and 25% for test. 30 executions were done for each task. Tables II and III presents the accuracy rate in % (the best results are emphasized in **bold,** according to the empirical analysis) and the standard deviation in brackets, comparing the initial pool, final pool using DE, and final pool using DE+FS. In Table II, all ELMs were combined by mean rule (static approach). In Table III, the KNORA (K Nearest ORAcles) Eliminate [20] was used (dynamic approach).

TABLE II
CLASSIFICATION ERROR OF STATIC APPROACH IN TEST SET

Task	Initial Pool	DE	DE+FS
Abalone	0.3467 (0.0154)	**0.3364** (0.0158)	0.3381 (0.0141)
Cancer	0.0297 (0.0127)	0.0326 (0.0115)	**0.0295** (0.0114)
Car	0.2526 (0.0413)	0.0958 (0.0234)	**0.0877** (0.0202)
Diabetes	0.2385 (0.0226)	0.2312 (0.0261)	**0.2268** (0.0274)
Ecoli	0.1861 (0.0504)	0.1409 (0.0355)	**0.1333** (0.0343)
Glass	0.3792 (0.0610)	0.3453 (0.0597)	**0.3302** (0.0577)
Iris	0.0667 (0.0393)	0.0468 (0.0274)	**0.0423** (0.0272)
Pendigits	0.1062 (0.0090)	0.0586 (0.0058)	**0.0585** (0.0059)
Sat	0.1702 (0.0056)	0.1502 (0.0064)	**0.1483** (0.0076)
Sonar	0.2167 (0.0527)	0.1801 (0.0506)	**0.1744** (0.0468)
Vehicle	0.2520 (0.0247)	**0.1954** (0.0189)	0.2058 (0.0231)
Vowel	0.3487 (0.0597)	0.2707 (0.0410)	**0.2667** (0.0370)
Wine	0.0205 (0.0226)	0.0182 (0.0192)	**0.0098** (0.0154)
Yeast	0.4451 (0.0239)	0.4084 (0.0218)	**0.4067** (0.0194)

TABLE III
CLASSIFICATION ERROR OF DYNAMIC APPROACH IN TEST SET

Task	Initial Pool	DE	DE+FS
Abalone	0.3725 (0.0175)	**0.3380** (0.0152)	0.3449 (0.0130)
Cancer	0.0349 (0.0137)	0.0375 (0.0120)	**0.0316** (0.0104)
Car	0.1350 (0.0287)	0.0722 (0.0208)	**0.0630** (0.0151)
Diabetes	0.2738 (0.0312)	**0.2331** (0.0232)	0.2385 (0.0240)
Ecoli	0.1786 (0.0504)	0.1425 (0.0350)	**0.1353** (0.0352)
Glass	0.3447 (0.0718)	**0.3371** (0.0620)	0.3396 (0.0720)
Iris	0.0495 (0.0276)	0.0505 (0.0299)	**0.0441** (0.0261)
Pendigits	0.0370 (0.0067)	0.0299 (0.0048)	**0.0286** (0.0046)
Sat	0.1542 (0.0059)	0.1340 (0.0100)	**0.1288** (0.0077)
Sonar	0.2115 (0.0589)	0.1737 (0.0412)	**0.1647** (0.0512)
Vehicle	0.2545 (0.0251)	**0.1956** (0.0221)	0.2096 (0.0253)
Vowel	0.2525 (0.0507)	0.**2063** (0.0406)	0.2086 (0.0361)
Wine	0.0212 (0.0223)	0.0174 (0.0195)	**0.0098** (0.0176)
Yeast	0.4615 (0.0174)	**0.4076** (0.0225)	0.4148 (0.0216)

Tables II and III show that, in an empirical analysis, the accuracy rates obtained by DE+FS outperforms the final pool using DE for most tasks, 12 against 2 tasks in static approach and 8 against 6 tasks in dynamic approach. DE+FS achieved better results than initial pool in all tasks. DE achieved better results than initial pool, except in Cancer (both approaches) and Iris (only in dynamic approach) tasks. However, it is necessary to assess whether the performances are statistically betters. Thus, we performed the paired t-tests ($\alpha = 0.05$). In static approach, DE+FS was better than initial pool except in two tasks (Cancer and Diabetes). In dynamic approach, DE+FS was better than initial pool except in three tasks (Cancer, Glass, and Iris) and better than final pool using DE in three tasks (Cancer, Car, and Sat).

To access the accuracy of the proposed method, other techniques were used for comparison. Table IV presents the performance of some traditional ensemble methods, executed in Weka 3.6.8: AdaBoost [16] (ADBO), Bagging [15] (BAG), and Random Subspace Method [17] (RSM). The parameters values were chosen as default from Weka 3.6.8. The best results are emphasized in **bold,** according to the empirical analysis, and the standard deviation in brackets.

TABLE IV
COMPARISONS BETWEEN TRADITIONAL ENSEMBLE METHODS

Task	Proposed Method	ADBO	BAG	RSM
Abalone	**0.3381** (0.0141)	0.4321 (0.0201)	0.3620 (0.0136)	0.3568 (0.0165)
Cancer	**0.0295** (0.0114)	0.0477 (0.0144)	0.0396 (0.0141)	0.0381 (0.0131)
Car	0.0630 (0.0151)	0.2896 (0.0198)	**0.0536** (0.0112)	0.2819 (0.0290)
Diabetes	**0.2268** (0.0274)	0.2520 (0.0300)	0.2497 (0.0259)	0.2553 (0.0316)
Ecoli	**0.1333** (0.0343)	0.3587 (0.0501)	0.1829 (0.0459)	0.1984 (0.0567)
Glass	0.3302 (0.0577)	0.5792 (0.0650)	0.3346 (0.0503)	**0.3195** (0.0620)
Iris	**0.0423** (0.0272)	0.0495 (0.0225)	0.0441 (0.0251)	0.0423 (0.0323)
Pendigits	0.0286 (0.0052)	0.8011 (0.0066)	0.0329 (0.0044)	**0.0278** (0.0035)
Sat	0.1288 (0.0077)	0.5664 (0.0093)	0.1188 (0.0078)	**0.1146** (0.0083)
Sonar	**0.1647** (0.0077)	0.2609 (0.0622)	0.2519 (0.0655)	0.2526 (0.0062)
Vehicle	**0.2058** (0.0231)	0.4608 (0.0427)	0.2572 (0.0246)	0.2535 (0.0210)
Vowel	**0.2086** (0.0361)	0.8651 (0.0188)	0.3283 (0.0508)	0.2879 (0.0422)
Wine	**0.0098** (0.0176)	0.1000 (0.0483)	0.0901 (0.0697)	0.0742 (0.0462)
Yeast	**0.4067** (0.0194)	0.5979 (0.0227)	0.4178 (0.0225)	0.4472 (0.0330)

Table IV shows that, in an empirical analysis, the rates obtained by the proposed method have the lowest error for most of tasks, 10 against 4 tasks. The paired t-tests ($\alpha = 0.05$) showed that the proposed method was better than ADBO, except in Iris task (equivalent). BAG was better only in one task (Car) and equivalent in two tasks (Glass and Iris). RSM was better in only one task (Sat) and equivalent in three tasks (Glass, Iris, and Pendigits). This results show the potential of ensembles when EA are employed to optimize the classifier generation.

TABLE V
COMPARISONS BETWEEN METHODS FROM LITERATURE

Task	Proposed Method	[3]	[4]	[5]	[6]	[21]
Abalone	0.3381	-	-	-	-	-
Cancer	**0.0295**	0.0352	-	0.0360	0.0310	-
Car	0.0630	-	-	-	-	-
Diabetes	**0.2268**	0.2288	0.2278	0.2313	0.2284	0.2648
Ecoli	0.1333	0.1538	-	-	**0.1326**	-
Glass	0.3302	0.3573	-	0.3133	0.3660	**0.3130**
Iris	0.0423	0.0358	0.0310	-	**0.0280**	-
Pendigits	0.0286	-	-	-	-	-
Sat	0.1288	-	-	-	-	-
Sonar	0.1647	0.2231	-	-	-	**0.1343**
Vehicle	**0.2058**	0.2182	-	-	-	0.3182
Vowel	0.2086	-	-	-	-	-
Wine	**0.0098**	-	0.0188	-	-	-
Yeast	0.4067	-	-	-	-	-

Table V presents comparisons between some methods from the literature. This type of comparison must be made with caution, because the results are obtained with different experimental model setups as well as with different learning approaches. Thus the **boldfaced** values indicate the method that has the lowest error for each problem. In most number of tasks, the proposed method achieved better performance.

VIII. Final Remarks

This work is concerned with the development of a method that aims automatic construction of ensembles, based on an enhanced DE incorporating a sharing function method. We have considered two variants: static and dynamic approaches. Both variants have achieved better-performing when compared with the initial pool and final pool using only DE. The proposed algorithm also outperforms some traditional ensembles and methods from literature. The use of a shared function method contributed for the diversity and accuracy in the population of classifiers. For this work, we choose ELM as our base classifier but, in principle, any other classifier can be used. Furthermore, other optimization algorithms could be applied as well multimodal techniques.

References

[1] G. B. Huang, Q. Y. Zhu, C. K. Siew, Extreme Learning Machine: Theory and Applications, *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006.

[2] Q. Y. Zhu, A. Qin, P. Suganthan, G. B. Huang, Evolutionary Extreme Learning Machine, Pattern Recognition vol. 38, no. 10, pp. 1759–1763, 2005.

[3] E. M. N. Figueiredo, and T. B. Ludermir, Investigating the use of Alternative topologies on Performance of the PSO-ELM, *Neurocomputing*, vol. 127, pp. 4-12, 2014.

[4] L. D. S. Pacífico, T. B. Ludermir, Evolutionary Extreme Learning Machine Based on Particle Swarm Optimization and Clustering Strategies, *International Joint Conference on Neural Networks*, 2013

[5] T. P. F. Lima, and T. B. Ludermir, Optimizing Dynamic Ensemble Selection Procedure by Evolutionary Extreme Learning Machines and a Noise Reduction Filter, *IEEE International Conference on Tools with Artificial Intelligence*, pp. 546-552, 2013.

[6] D. N. G. Silva, L. D. S. Pacifico, and T. B. Ludermir, Na Evolutionary Extreme Learning Machine Based on Group Search Optimization, *Congress on Evolutionary Computation*, pp. 574-580, 2011.

[7] L. I. Kuncheva, J. Bezdek, and R. Duin, Decision Templates for Multiple Classifier Fusion: an Experimental Comparison, *Pattern Recognition*, vol 24, no. 2, pp. 299-314, 2001.

[8] M. P. Ponti-JR, Combining Classifiers: from the Creation of Ensembles to the Decision Fusion, Conference on Graphics Pattern, and Images Tutorials, pp. 1-10, 2011.

[9] L. M. Almeida, and T. B. Ludermir, *A multi-objective memetic and hybrid methodology for optimizing the parameters and performance of artificial neural networks*, Neurocomputing, vol. 73, no. 9, pp. 1438-1450, 2010.

[10] R. Storn, and K. Prince, *Differential evolution: a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*. Journal of Global Optimization, vol. 11, no. 4, pp. 341-359, 1997.

[11] S. Das, and P. Suganthan, Differential Evolution – A Survey of the State-of-the-art, *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4-31, 2011.

[12] D. E. Goldberg, J. Richardson, Genetic Algorithms with Sharing for Multimodal Function Optimization, *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41-49, 1987.

[13] T. G. Dietterich, Ensemble Methods in Machine Learning, *1st Int. Work. on Multiple Classifier Systems*, pp. 1-15, 2000.

[14] K. Woods, W. P. Kegelmeyer, and K. W. Bowyer, Combination of Multiple Classifiers Using Local and Accuracy estimates, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405-410, 1997.

[15] L. Breiman, Bagging predictors, *Machine Learning*, vol. 24, no. 2, pp 123-140, 1996.

[16] R. E. Schapire, The Strength of Weak Learn Ability, *Machine Learning*, vol. 5, no. 2, pp. 197-227, 1990.

[17] T. K. Ho, The Random Subspace Method for Constructing Decision Forests, IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp. 832-844, 1998.

[18] K. J. Kim, S. B. Cho, Evolutionary Ensemble of Diverse Artificial Neural Networks using Speciation, *Neurocomputing*, vol. 71, pp. 1604-1618, 2008.

[19] A. Frank, A. Asuncion.: UCI Machine Learning Repository (2010), http://archive.ics.uci.edu/lm

[20] A. Ko, R. Sabourin, and A. Britto Jr., From Dynamic Classifier Selection to Dynamic Ensemble Selection, *Pattern Recognition*, vol. 41, no.5, pp. 1718-1731, 2008.

[21] D. S. Severo, E. Veríssimo, G. D. C. Cavalvanti, and T. I. Ren, Hybrid Feature Selection and Weighting Method Based on Binary Particle Swarm Optimization, *IEEE International Conference on Tools with Artificial Intelligence*, pp. 433-438, 2013.