2014 Brazilian Conference on Intelligent Systems

Improving Classifiers and Regions of Competence in Dynamic Ensemble Selection

Tiago P. F. Lima, Anderson, T. Sergio, and Teresa B. Ludermir
Centro de Informática
Universidade Federal de Pernambuco, UFPE
Recife, Brasil
{tpfl2,ats3,tbl}@cin.ufpe.br

Abstract — **This paper evaluates some strategies to approximate the performance of dynamic ensembles based on NN-rule to the oracle performance. For this purpose, we use a multi-objective optimization algorithm, based on Differential Evolution, to generate automatically a pool of accurate and diverse classifiers in the form of Extreme Learning Machines. However, the rule defined for selecting the classifiers depends on the quality of the information obtained from regions of competence. Thus, we also improve the regions of competence in order to avoid noise and create smoother class boundaries. Finally, we employ a dynamic ensemble selection method. The performance of the proposed method was experimentally investigated using 12 benchmark datasets and results of comparative analysis are presented.**

***Keywords** — Dynamic ensembles; oracle; multi-objective optimization; differential evolution; extreme learning machine.*

I. Introduction

The research field of ensembles of classifiers become 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 different classifiers could be interesting. According to Dietterich [1], 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 theoretical and experimental evidences that it is possible if the classifiers in an ensemble make different errors on a query pattern.

Statistical (or worst case) motivation: it is possible to avoid the worst classifier by averaging several classifiers. This simple combination was demonstrated to be efficient in many applications. There is no guarantee, however, that the ensemble will outperform a single classifier.

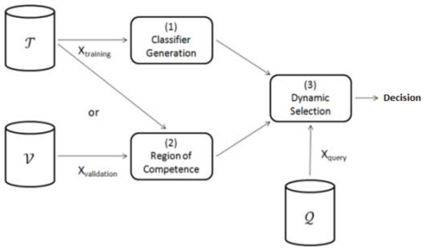
Computational motivation: some algorithms perform an optimization task in order to learn and suffer from local minima. In this case it is a difficult task to find the best classifier, and it is often used several (hundreds or even thousands) initializations in order to find a presumable optimal classifier. Combination of such classifiers showed to stabilize and improve the best single classifier result.

One of the most important issues surrounding ensembles of classifiers is ensemble selection. Given a pool of classifiers $\boldsymbol{C} = \{\boldsymbol{C_1}, \dots, \boldsymbol{C_L}\}$, the ensemble selection has focused on finding the most relevant subset of classifiers $\boldsymbol{E}$, rather than combining all available $\boldsymbol{L}$ classifiers, where $|\boldsymbol{E}| \leq |\boldsymbol{C}|$. We can perform this task either by static selection, i.e., selecting only one ensemble for all query patterns, or by dynamic selection, i.e., selecting different ensembles for different query patterns.

A way to define the upper limit of ensemble selection performance is through the concept of oracle. If at least one classifier $\boldsymbol{C_i} \in \boldsymbol{C}$ can correctly classify a given pattern $\boldsymbol{X_{query}}$, then the oracle can correctly classify $\boldsymbol{X_{query}}$. The objective of this work is to use the nature oracle, so we can only select those classifiers which might be able to correctly classify a given sample. This is accomplished in a dynamic fashion, since different patterns might require different ensembles.

To understand how we can explore the oracle concept to improve a Dynamic Ensemble Selection (DES) process, first we have to look three basic steps in Fig. 1: (1) generate a classifier pool using the training dataset $\boldsymbol{\mathcal{T}}$; (2) produce the region of competence by the training dataset $\boldsymbol{\mathcal{T}}$ or an independent validation dataset $\boldsymbol{\mathcal{V}}$; and finally (3) select the classifier(s) based on the information extracted from the regions of competence. The classifier(s) selected is(are) combined to classify the query pattern $\boldsymbol{X_{query}}$.

Fig. 1. Dynamic ensemble selection process



Despite the large number of selection methods available in the literature, the classifier generation and region of competence in DES process have not been given much attention. In addition, due to the high computational cost usually observed in the dynamic selection solutions, its application is often criticized. In fact, the decision as to whether or not use dynamic selection is still an open question.

978-1-4799-5618-0/14 $31.00 © 2014 IEEE
DOI 10.1109/BRACIS.2014.14

This paper is organized as follows: Section II defines terms and provides the main concepts of the chosen base classifier (Extreme Learning Machines); Section III presents the origins of enhanced evolutionary algorithm proposed; Section IV shows a brief review of selection methods; Section V describes the idea of proposed method; Section VI presents the experimental results; Finally, section VII gives some final considerations about the main topics covered in this work.

II. Extreme Learning Machines

There are many types of machine learning classification techniques. Artificial Neural Networks (ANNs) [2] are one of the most popular employed. This particular type of classifier has been extensively used due to its inherent characteristics: nonlinearity, high parallelism, robustness, fault and failure tolerance, learning ability to handle imprecise information, and its capability to generalize well on unseen [3].

As an important branch of neural network, Extreme Learning Machine (ELM), introduced by Huang et al. [4], plays an important role in the fields of pattern classification. The main characteristic of ELMs is learning without iterative training. Let $\boldsymbol{\mathcal{T}} = \{(\boldsymbol{P_i}, \boldsymbol{T_i}) | \boldsymbol{P_i} \in \boldsymbol{\Re^n}, \boldsymbol{T_i} \in \boldsymbol{\Re^m}, \boldsymbol{i = 1, \dots, N}\}$ be the training dataset, where $\boldsymbol{P_i}$ is a $\boldsymbol{n}$-dimensional input pattern and $\boldsymbol{T_i}$ is a $\boldsymbol{m}$-dimensional target. The training process is described briefly 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 $\boldsymbol{A_j}$ is the input weights, $\boldsymbol{B_j}$ is the hidden layers biases, $\boldsymbol{\beta_j}$ is the output weight that connects the $\boldsymbol{j^{th}}$ hidden node and output node, $\boldsymbol{f}(.)$ is the activation function, $\boldsymbol{L}$ is the number of hidden neurons, and $\boldsymbol{N}$ is the number of distinct input or output data.

$$\sum_{j=1}^{L} \boldsymbol{\beta_j f(A_j, B_j, P_j) = T_j, j = 1, \dots, N} \quad (1)$$

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

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{f(A_1, B_1, P_1)} & \dots & \boldsymbol{f(A_L, B_L, P_L)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{f(A_1, B_1, P_N)} & \dots & \boldsymbol{f(A_L, B_L, P_N)} \end{bmatrix}_{\mathbf{N \times L}}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta_1^T} \\ \vdots \\ \boldsymbol{\beta_L^T} \end{bmatrix}_{\mathbf{L \times m}} \text{ and } \boldsymbol{T} = \begin{bmatrix} \boldsymbol{T_1^T} \\ \vdots \\ \boldsymbol{T_L^T} \end{bmatrix}_{\mathbf{N \times m}}$$

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

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 Evolutionary Algorithms (EAs) [5]. EAs and ANNs are combined to produce a hybrid model with low error and high generalization, yielding evolutionary ANNs. While there is still no one EA that is universally regarded as better than others, Differential Evolution (DE) is considered to have some advantages due to its simplicity.

III. Differential Evolution

DE is one of the most powerful stochastic real-parameter optimization algorithms of current interest [6, 7]. DE operates through similar computational steps as employed by a standard EA: initialization, mutation, crossover and selection. Firstly, a number of individuals generated uniformly from the whole search space form the initial population. For each individual in the population (i.e. the target vector), a new individual (i.e. the trial vector) is generated through both mutation and crossover. The $\boldsymbol{i^{th}}$ target vector in the $\boldsymbol{g^{th}}$ generation of the population is denoted as $\boldsymbol{X_{i,g}}$, and the corresponding trial vector is represented by $\boldsymbol{U_{i,g}}$. In the selection step, $\boldsymbol{X_{i,g}}$ will be replaced by $\boldsymbol{U_{i,g}}$ if $\boldsymbol{U_{i,g}}$ is better than $\boldsymbol{X_{i,g}}$. The main two steps (i.e. mutation and crossover) are described briefly as follows.

Mutation: The mutant of $\boldsymbol{X_{i,g}}$ obtained by the mutation operation is represented by $\boldsymbol{V_{i,g}}$. In a basic DE,

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

Note that $\boldsymbol{i \neq r1 \neq r2 \neq r3}$. $\boldsymbol{F}$ is a scaling factor.

Crossover: $\boldsymbol{U_{i,g}}$ is generated from $\boldsymbol{V_{i,g}}$ and $\boldsymbol{X_{i,g}}$.

$$\boldsymbol{U_{j,i,g}} = \begin{cases} \boldsymbol{V_{j,i,g}} \textbf{ if } \boldsymbol{rand_{i,j}[0,1] \leq Cr} \textbf{ or } \boldsymbol{j = j_{rand}} \\ \qquad \boldsymbol{X_{j,i,g}} \textbf{ otherwise} \end{cases} \quad (3)$$

$\boldsymbol{U_{j,i,g}}$ represents the $\boldsymbol{j^{th}}$component of $\boldsymbol{U_{i,g}}$, $\boldsymbol{rand_{i,j}[0,1]}$ is a uniformly distributed random number, $\boldsymbol{Cr}$ is the crossover probability, and $\boldsymbol{j_{rand}}$ is a randomly chosen index which can ensure at least one parameter be copied from $\boldsymbol{V_{i,g}}$.

IV. Selection Methods

Let $\boldsymbol{C} = \{\boldsymbol{C_1, \dots, C_L}\}$ be a pool composed of $\boldsymbol{L}$ classifiers and $\boldsymbol{E} = \{\boldsymbol{E_1, \dots, E_M}\}$ be a pool composed of $\boldsymbol{M}$ ensembles formed from $\boldsymbol{C}$. Denote the regions of competence by $\boldsymbol{R_1, \dots, R_K}$. In DES process, we decide which ensemble from $\boldsymbol{E}$ we should nominate for each region $\boldsymbol{R_j}$, $\boldsymbol{j = 1, \dots, K}$. Let $\boldsymbol{E^*}$ be the ensemble with the highest average accuracy amongst the ensembles of $\boldsymbol{E}$ over the whole feature space. Denote by $\boldsymbol{P(E_i|R_j)}$ the probability of correct classification by $\boldsymbol{E_i}$ in region $\boldsymbol{R_j}$. Let $\boldsymbol{E_{i(j)}} \in \boldsymbol{E}$ be the ensemble responsible for region $\boldsymbol{R_j}$. The overall probability of correct classification $\boldsymbol{P_c}$ can be computed by (4), where $\boldsymbol{P(R_j)}$ is the probability that an input drawn from the distribution of the problem falls in $\boldsymbol{R_j}$.

$$\boldsymbol{P_c} = \sum_{j=1}^{K} \boldsymbol{P(R_j) P_c(R_j)} = \sum_{j=1}^{K} \boldsymbol{P(R_j) P(E_{i(j)}|R_j)} \quad (4)$$

To maximize $\boldsymbol{P_c}$, we assign $\boldsymbol{E_{i(j)}}$ so that

$$\boldsymbol{P(E_{i(j)}|R_j) \geq P(E_t|R_j), \forall t = 1, 2, \dots, M} \quad (5)$$

From (4) and (5), we have that

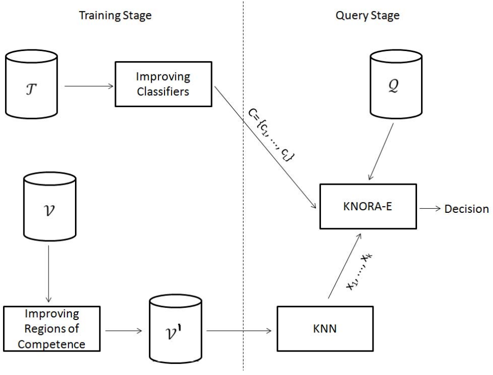
$$\boldsymbol{P_c} \geq \sum_{j=1}^{K} \boldsymbol{P}(\boldsymbol{R_j})\boldsymbol{P}(\boldsymbol{E}^*|\boldsymbol{R_j}) = \boldsymbol{P}(\boldsymbol{E}^*) \tag{6}$$

Equation (6) shows that selection methods perform equal of better than the best ensemble $\boldsymbol{E}^*$, regardless of the way the feature space has been partitioned. The only condition is to ensure that $\boldsymbol{E_{i(j)}}$ is the best among the $\boldsymbol{M}$ ensembles in $\boldsymbol{E}$ for region $\boldsymbol{R_j}$. It is worth noting that, basically, the individual-based measures most often take into account the classifier accuracy. However, the measures are conducted in different ways. For instance, one may find measures based on pure accuracy (overall local accuracy or local class accuracy) [8], ranking of classifiers [9], probabilistic information [10,11], classifier behavior calculated on output profiles [12-14], and oracle information [15,16]. Moreover, we may find measures that consider interactions among classifiers, such as diversity [17-19], ambiguity [20,21] or other grouping approaches [22].

V. Proposed Method

Fig. 2 shows the architecture of the proposed approach. The training stage generates a pool of classifiers $\boldsymbol{C} = \{\boldsymbol{C_1}, \ldots, \boldsymbol{C_L}\}$ using the training dataset $\boldsymbol{\mathcal{T}}$. Moreover, a noise reduction filter is applied to the validation dataset $\boldsymbol{\mathcal{V}}$ to remove noise patterns. In the query stage, the local region is computed using the patterns of the filtered dataset $\boldsymbol{\mathcal{V}'}$. We use the KNORA-Eliminate [16] to select the dynamic ensemble.

Fig. 2. Proposed method flowchart



A. Improving Classifiers

The hybridization of an enhanced DE and ELM was performed to build an automatic method capable of seeking a pool of ELMs, thereby avoiding difficulties stemming from a non automatic trial-and-error search. The enhanced DE was executed for $\boldsymbol{G_{max}} = \boldsymbol{1000}$ generations. We used 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 $\boldsymbol{NP} = \boldsymbol{20}$. With the use of EA, 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. 3.

Fig. 3. Composition of an individual in ELM optimization

Feature Selection	Hidden Neurons	Activation Function	Input Weights	Hidden Biases

The first part of the individual is responsible for the feature selection, in order to reduce the complexity of ELMs generated. The second part contains information on the hidden neurons. We use the minimum number of neurons $\boldsymbol{N_{min}} = \boldsymbol{10}$ and the maximum number of neurons $\boldsymbol{N_{max}} = \boldsymbol{30}$. 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 used 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 $[-\boldsymbol{1}, \boldsymbol{1}]$), 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. Finally, the objective function is computed. The enhanced DE takes into account multiple objectives (instead of using only the training dataset error to avoid overfitting [23]). Thus, we adopt the most known error functions (in training and validation datasets), i.e., the root mean square error (RMSE) and classification error (CE), defined respectively in (7) and (8). In (7), $\boldsymbol{m}$ is the number of output units, $\boldsymbol{T_{ji}}$ is the target to pattern $\boldsymbol{i}$ in the output $\boldsymbol{j}$, $\boldsymbol{O_{ji}}$ is the output obtained to the pattern $\boldsymbol{i}$ in the output $\boldsymbol{j}$ and $\boldsymbol{N}$ is the number of samples. In (8), $\boldsymbol{c}$ is the number of classes and $\boldsymbol{C_i}$ is the number of errors per class.

$$\boldsymbol{RMSE} = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{m} (\boldsymbol{T_{ji}} - \boldsymbol{O_{ji}})^2 / \boldsymbol{N} \times \boldsymbol{m}} \tag{7}$$

$$\boldsymbol{CE} = \sum_{i=1}^{c} \boldsymbol{C_i} \tag{8}$$

In the enhanced DE, at each generation $\boldsymbol{g}$, the crossover probability $\boldsymbol{Cr}$ of each individual $\boldsymbol{i}$ is independently generated according to a normal distribution of mean $\mu\boldsymbol{Cr}$ and standard deviation $\boldsymbol{0.1}$, as in (9), and then truncated to $[\boldsymbol{0}, \boldsymbol{1}]$. Denote $\boldsymbol{S_{Cr}}$ as the set of all successful crossover probabilities $\boldsymbol{Cr_i}$'s at generation $\boldsymbol{g}$. The mean $\mu\boldsymbol{Cr}$ is initialized to be $\boldsymbol{0.5}$ and then updated at the end of each generation as in (10), where $\boldsymbol{c}$ is a positive constant between $\boldsymbol{0}$ and $\boldsymbol{1}$ and $\boldsymbol{mean_A}(.)$ is the traditional arithmetic mean. Similarly, at each generation $\boldsymbol{g}$, the mutation factor $\boldsymbol{F}$ of each individual $\boldsymbol{i}$ is independently generated according to a Cauchy distribution with location parameter $\mu\boldsymbol{F}$ and scale parameter $\boldsymbol{0.1}$, as in (11), and then truncated to be $\boldsymbol{1}$ if $\boldsymbol{F_i} \geq \boldsymbol{1}$ or regenerated if $\boldsymbol{F_i} \leq \boldsymbol{0}$. Denote $\boldsymbol{S_F}$ as the set of all successful mutation factors $\boldsymbol{F_i}$'s in generation $\boldsymbol{g}$. The location parameter $\mu\boldsymbol{F}$ of the Cauchy distribution is initialized to be $\boldsymbol{0.5}$ and then updated at the end of each generation as in (12), where $\boldsymbol{mean_L}(.)$ is the Lehmer mean that is calculated as in (13).

$$\boldsymbol{Cr_i} = \boldsymbol{randn_i}(\mu\boldsymbol{Cr}, \boldsymbol{0.1}) \tag{9}$$

$$\mu\boldsymbol{Cr} = (\boldsymbol{1} - \boldsymbol{c}) \cdot \mu\boldsymbol{Cr} + \boldsymbol{c} \cdot \boldsymbol{mean_A}(\boldsymbol{S_{Cr}}) \tag{10}$$

$$\boldsymbol{F_i} = \boldsymbol{randc_i}(\mu\boldsymbol{F}, \boldsymbol{0.1}) \tag{11}$$

$$\mu\boldsymbol{F} = (\boldsymbol{1} - \boldsymbol{c}) \cdot \mu\boldsymbol{F} + \boldsymbol{c} \cdot \boldsymbol{mean_L}(\boldsymbol{S_F}) \tag{12}$$

$$\boldsymbol{mean_L}(\boldsymbol{S_F}) = \sum_{F \in S_F} \boldsymbol{F}^2 / \sum_{F \in S_F} \boldsymbol{F} \tag{13}$$

In DE, recently explored inferior solutions, when compared to the current population, can provide additional information about the promising progress direction. Thus, denote $\boldsymbol{A}$ as the set of archived inferior solutions and $\boldsymbol{P}$ as the current population. The mutant vector, in enhanced DE, $\boldsymbol{V_{i,g}}$ is generated as in (14), where $\boldsymbol{X_{i,g}}$, $\boldsymbol{X_{j,g}^{rpf}}$ and $\boldsymbol{X_{r1,g}}$ are selected from $\boldsymbol{P}$, while $\boldsymbol{X_{r2,g}}$ is randomly chosen from the union, $\boldsymbol{P} \cup \boldsymbol{A}$, of the current population and the archive.

$$\boldsymbol{V_{i,g}} = \boldsymbol{X_{i,g}} + \boldsymbol{F_i}\left(\boldsymbol{X_{j,g}^{rpf}} - \boldsymbol{X_{i,g}}\right) + \boldsymbol{F_i}\left(\boldsymbol{X_{r1,g}} - \boldsymbol{X_{r2,g}}\right) \qquad (14)$$

In (14), $\boldsymbol{X_{j,g}^{rpf}}$ is randomly chosen as one of the individuals in Pareto front. The concept of Pareto dominance and Pareto optimality are fundamental in multi-objective optimization, with Pareto dominance forming the basis of solution quality. Given the objective vectors $\boldsymbol{Y_1}, \boldsymbol{Y_2} \in \boldsymbol{\Re^m}$, then $\boldsymbol{Y_1}$ dominates $\boldsymbol{Y_2}$, denoted as $\boldsymbol{Y_1} \prec \boldsymbol{Y_2}$ if $\boldsymbol{y_{1i}} \leq \boldsymbol{y_{2i}}\, \forall \boldsymbol{i} \in \{\boldsymbol{1}, \dots, \boldsymbol{m}\}$ and $\boldsymbol{y_{1j}} < \boldsymbol{y_{2j}}\ \exists \boldsymbol{j} \in \{\boldsymbol{1}, \dots, \boldsymbol{m}\}$. The Pareto front denoted by $\boldsymbol{Z}^*$ is the set of individuals $\boldsymbol{Z}^* = \{\boldsymbol{Z_j^*} | \boldsymbol{Z_j^*} \prec \boldsymbol{Z_i}, \forall \boldsymbol{Z_i} \in \boldsymbol{Z}\}$.

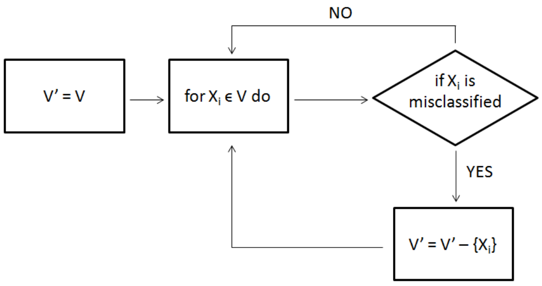
Finally, the selection operation of enhanced DE selects the best from the parent vector $\boldsymbol{X_{i,g}}$ and the trial vector $\boldsymbol{U_{i,g}}$ according to their objective functions $\boldsymbol{f}(.)$. For example, the member for the next generation, at $\boldsymbol{g} = \boldsymbol{g} + \boldsymbol{1}$, is described as in (15). The parent solutions that fail in the selection process are added to the archive. If the archive size exceeds a threshold, say $\boldsymbol{NP}$, then some solutions are randomly removed from the archive to keep the archive size at $\boldsymbol{NP}$.

$$\boldsymbol{X_{i,g+1}} = \begin{cases} \boldsymbol{U_{i,g}}, & \boldsymbol{if\ f(U_{i,g}) \prec f(X_{i,g})} \\ \boldsymbol{X_{i,g}}, & \boldsymbol{otherwise} \end{cases} \qquad (15)$$

B. Improving Regions of Competence

Although DES methods can usually achieve better classification performance than the single best classifier among them, it has been showed that there is still a large performance gap between DES and the oracle [24]. The current DES systems end up selecting the wrong classifiers when there are noise patterns near the query pattern. Based on this, we propose two techniques that remove samples that are considered noise, as shows the flowchart in Fig. 4. The first technique use the oracle concept, i.e., if all classifiers from $\boldsymbol{C}$ misclassify $\boldsymbol{X_i}$, then $\boldsymbol{X_i}$ is a noise. The second technique is based on the majority vote concept, i.e., if the majority vote of all classifiers from $\boldsymbol{C}$ misclassify $\boldsymbol{X_i}$, then $\boldsymbol{X_i}$ is a noise.

Fig. 4. Improving regions of competence flowchart



VI. Experiments and Results

For evaluating our method, the experiments were conducted using $\boldsymbol{12}$ benchmarks classification tasks found in the UCI Machine Learning Repository [25]. These tasks present different degrees of difficulties and different number of examples, attributes and classes, as summarized in Table I. All inputs (patterns) have been normalized into the range $[\boldsymbol{0}, \boldsymbol{1}]$, while the outputs (targets) have been normalized into $[-\boldsymbol{1}, \boldsymbol{1}]$. Each task was randomly divided into $\boldsymbol{50\%}$ for training, $\boldsymbol{25\%}$ for validation and $\boldsymbol{25\%}$ for test. The experiments were executed with a $\boldsymbol{30}$-dimension search space.

TABLE I. Specification of the Tasks Used in the Experiments

Task	**Number of**		
	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
Page	5473	10	5
Pendigits	10992	16	10
Sat	6435	36	6
Vehicle	846	18	3
Waveforms	5000	40	3
Yeast	1484	8	10

Table II presents the error rates in percentage, comparing the initial pool (before classifier optimization) and final pool (after classifier optimization) without improving regions of competence. The best results are emphasized in **bold**, according to paired $\boldsymbol{t}$-test ($\boldsymbol{\alpha = 0.05}$), and the standard deviation in brackets. The final pool was statistically better in all tasks, except in Glass task (statistically equivalent).

TABLE II. Error Rates in Improving Classifiers

Task	**Improving Classifiers**	
	Before	**After**
Abalone	37.62 (1.53)	**33.80 (1.66)**
Cancer	3.93 (1.52)	**3.14 (1.01)**
Car	13.77 (2.89)	**06.47 (1.76)**
Diabetes	27.15 (3.24)	**23.70 (2.52)**
Ecoli	17.50 (4.89)	**14.25 (4.25)**
Glass	34.97 (6.86)	33.27 (7.15)
Page	5.01 (0.66)	**4.26 (0.53)**
Pendigits	3.95 (0.63)	**2.55 (0.42)**
Sat	15.23 (0.75)	**13.24 (0.87)**
Vehicle	25.66 (2.50)	**19.92 (2.25)**
Waveforms	20.34 (1.23)	**14.57 (0.86)**
Yeast	45.92 (2.37)	**40.75 (2.03)**

Figs. 5 and 6 show the minimum and maximum reduction rates obtained in filtered validation dataset $\boldsymbol{\mathcal{V}'}$, as well the original dimensionality in validation dataset $\boldsymbol{\mathcal{V}}$. In an empirical analysis, the dynamic selection using $\mathcal{V}'$ presented better results with lower error rates. However, the paired $\boldsymbol{t}$-test $(\boldsymbol{\alpha = 0.05})$ showed that there is no statistically significant difference. Nevertheless $|\mathcal{V}'| < |\mathcal{V}|$, which contributes in decreasing the cost of computing the nearest neighbors in the dynamic selection. Reduction rates using the filter based on majority vote achieved the biggest absolute reductions considering both maximum and minimum values.

Fig. 5. Reduction rates (in %) using the filter based on oracle concept

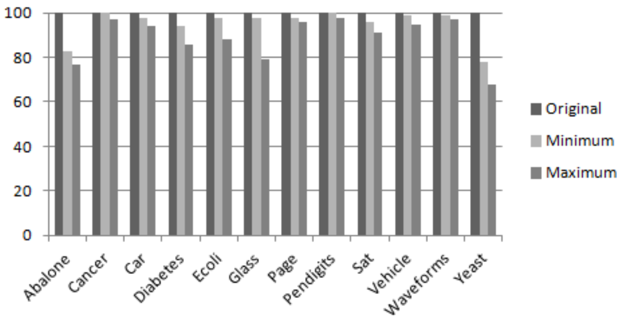
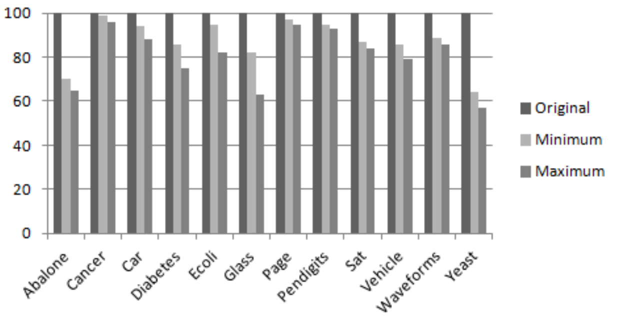


Fig. 6. Reduction rates (in %) using the filter based on majority vote



It is important to mention that one of the main problems in selection methods based on the NN-rule is the computational cost to define the neighborhood of the query pattern. When none of the classifiers correctly classifies all the neighbors, the neighborhood is reduced and the algorithm computes again. This becomes a problem when there are many noise patterns in dataset where the regions of competence are computed. The algorithm needs to reduce the neighborhood often, which increases considerably the computational time.

To access the accuracy of the proposed method, other techniques were used for comparison. Table III presents the performance of some traditional ensemble methods, executed in Weka 3.6.8: AdaBoost [26] (ADBO), Bagging [27] (BAG), and Random Subspace Method [28] (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 III. COMPARISON AMONG TRADITIONAL ENSEMBLE METHODS

Task	Proposed Method Filter Based on		Traditional Ensemble Methods		
	Oracle	**Majority Vote**	**ADBO**	**BAG**	**RMS**
Abalone	**33.71** **(1.60)**	33.78 (1.50)	43.21 (2.06)	36.20 (1.36)	35.68 (1.64)
Cancer	3.16 (1.00)	**2.99** **(1.16)**	4.77 (1.44)	3.96 (1.41)	3.81 (1.31)
Car	6.30 (1.73)	8.45 (1.99)	28.97 (1.98)	**5.36** **(1.11)**	28.19 (2.90)
Diabetes	23.60 (2.33)	**23.06** **(2.46)**	25.20 (3.00)	24.97 (2.59)	25.53 (3.16)
Ecoli	14.33 (4.47)	**13.53** **(3.49)**	35.87 (5.09)	18.29 (4.59)	19.84 (5.67)
Glass	32.96 (7.62)	32.89 (5.21)	57.92 (6.50)	33.46 (5.03)	**31.95** **(6.21)**
Page	4.22 (0.51)	4.37 (0.58)	6.73 (0.72)	**3.07** **(0.48)**	3.43 (0.55)
Pendigits	**2.37** **(0.42)**	3.84 (0.38)	80.11 (0.66)	3.29 (0.44)	2.78 (0.35)
Sat	12.92 (0.88)	14.74 (0.75)	56.64 (0.93)	11.88 (0.79)	**11.46** **(0.83)**
Vehicle	19.97 (2.18)	**19.89** **(2.31)**	46.08 (4.27)	25.72 (2.46)	25.35 (2.10)
Waveforms	14.72 (0.81)	**13.78** **(0.88)**	33.94 (3.84)	18.86 (0.99)	18.19 (1.13)
Yeast	**40.70** **(2.15)**	40.83 (2.01)	59.79 (2.27)	41.78 (2.26)	44.72 (3.30)

Table III shows that, in an empirical analysis, the proposed method have the lowest error rates for most of tasks, $\mathbf{8}$ against $\mathbf{4}$ tasks. The paired $\boldsymbol{t}$-tests $(\boldsymbol{\alpha = 0.05})$ showed that the proposed method was better than ADBO in all tasks. BAG was better in four tasks (Car, Page, Pendigits - using the filter based on majority vote, and Sat) and equivalent in two tasks (Glass and Yeast). RSM was better in three tasks (Page, Pendigits - using the filter based on majority vote, and Sat) and equivalent in only one task (Glass).

Table IV presents comparisons among 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 task. In most number of tasks, the proposed method achieved one of the best results.

TABLE IV. COMPARISON AMONG METHODS FROM LITERATURE

Task	Proposed Method	[11]	[22]	[29]	[30]	[31]
Cancer	**2.99**	3.30	3.50	-	3.60	3.52
Diabetes	23.06	-	23.80	23.96	23.13	**22.88**
Ecoli	**13.53**	-	-	-	-	15.38
Glass	32.89	35.32	-	-	**31.33**	35.73
Vehicle	19.89	-	-	**19.80**	-	21.82

VII. Final Remarks

This work is concerned with the development of a method that improve classifiers and regions of competence in DES. The method is based on an enhanced DE and techniques that remove samples that are considered noise. Through experimental results, it was possible to observe that the method reached a good improvement, especially in improving classifiers. It is relevant to mention that the evolutionary optimization avoids considerable human effort and difficulties stemming from a non-automatic trial-and-error search. Moreover, the use of techniques that remove samples that are considered noise decreased the computational cost, because reduced the cost of computing the nearest neighbor rule which can be high in some cases. Even for the methods that have a fixed neighborhood size and therefore does not need to re-compute, the use of this approach can still reduce the computational cost because the filter eliminates some patterns. 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.

Acknowledgment

The authors thank FACEPE, CNPq and CAPES (Brazilian Research Agencies) for their financial support.

References

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

[2] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*, Wiley Interscience, 2004.

[3] S. Haykin, *Neural Networks and Learning Machines*, Prentice Hall, 2009.

[4] G. B. Huang, Q. Y. Zhu, and C. K. Siew, *Extreme Learning Machine: Theory and Applications*, Neurocomputing, vol. 70, pp. 489-501, 2006.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] M. Sabourin, A. Mitiche, D. Thomas, G. Nagy, *Classifier combination for handprinted digit recognition*, Second Internation Conference on Document Analysis and Recognition, pp. 163-166, 1993.

[10] G. Giacinto, F. Roli, *Methods for dynamic classifier selection*, 10th International Conference on Image Analysis and Processing, pp. 659-664, 1999.

[11] M. Kurzynski, T. Woloszynski, R. Lysiak, *On two measures of classifier competence for dynamic ensemble selection - experimental comparative analysis*, International Symposium on Communications and Information Technologies, pp. 1108-1113, 2010.

[12] G. Giacinto, F. Roli, *Dynamic classifier selection based on multiple classifier behaviour*, Pattern Recognition, vol. 34, pp. 1879-1881, 2001.

[13] A. Nabiha, F. Nadir, *New dynamic ensemble of classifiers selection approach based on confusion matrix for arabic handwritten recognition*, International Conference on Multimedia Computing and Systems, pp. 308-313, 2012.

[14] P. R. Cavalin, R. Sabourin, C. Y. Suen, *Dynamic selection approaches for multiple classifier systems*, Neural Comput. Appl. vol. 22, pp. 673-688, 2013.

[15] L. I, Kuncheva, J. J. Rodriguez, *Classifier ensembles with a random linear oracle*, IEEE Trans. Knowl. Data Eng., vol. 19, no. 4, pp. 500-508, 2007.

[16] A. H. R. Ko, R. Sabourin, A. S. Britto Jr., *From dynamic classifier selection to dynamic ensemble selection*, Pattern Recognit. vol. 41, no. 5, pp. 1718-1731, 2008.

[17] H. W. Shin, S. Y. Sohn, *Combining both ensemble and dynamic classifier selection schemes for prediction of mobile internet subscribers*, Expert Syst. Appl. vol. 25, no. 1, pp. 63–68, 2003.

[18] A. Santana, R. G. F. Soares, A. M. P. Canuto, M. C. P. de Souto, *A dynamic classifier selection method to build ensembles using accuracy and diversity*, Ninth Brazilian Symposium on Neural Networks, pp.36–41, 2006.

[19] R. Lysiak, M. Kurzynski, T. Woloszynski, *Probabilistic approach to the dynamic ensemble selection using measures of competence and diversity of base classifiers*, Hybrid Artificial Intelligent Systems, Lecture Notes in Computer Science, vol. 6679, pp. 229–236, 2011.

[20] T. K. Ho, J. J. Hull, S. N. Srihari, *Decision combination in multiple classifier systems*, IEEE Trans. Pattern Anal. Mach. Intell. vol 16, no. 1, pp. 66–75, 1994.

[21] E. M. dos Santos, R. Sabourin, P. Maupin, *A dynamic overproduce-and-choose strategy for the selection of classifier ensembles*, Pattern Recognit, vol.41, no. 10, pp. 2993–3009, 2008.

[22] J. Xiao, C. He, *Dynamic classifier ensemble selection based on GMDH*, International Joint Conference on Computational Sciences and Optimization,vol.1, pp.731–734, 2009.

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

[24] L. Didaci, G. Giacinto, F. Roli, G.L. Marcialis, *A study on the performances of dynamic classifier selection based on local accuracy estimation*, Pattern Recognition vol. 38, no. 11, pp. 2188–2191, 2005.

[25] 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

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

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

[28] 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.

[29] R. M. O. Cruz, G. D. C. Cavalcanti, and T. I. Ren, *A method for dynamic ensemble selection based on a filter and an adaptive distance to improve the quality of the regions of competence*, International Joint Conference on Neural Networks, pp. 1126-1133, 2011.

[30] 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.

[31] 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.