A preliminary study on the use of differential evolution for adjusting the position of examples in nearest neighbor classification

Isaac Triguero, Salvador García, Francisco Herrera

***Abstract*—Nearest neighbor is one of the most successfully used techniques for performing classification and pattern recognition tasks. Its simplicity and effectiveness justify the use of this technique in certain domains but it however presents several drawbacks referring to time response, noise sensitivity and storage requirements. Several solutions have been proposed in order to alleviate these problems, such as improving the technique for speeding up or carrying out a data reduction process. Prototype generation is a suitable process for data reduction that allows to fit a data set for nearest neighbor classification. Position adjustment of prototypes is a successful technique within the prototype generation methodology.**

Evolutionary algorithms are adaptive methods based on natural evolution that may be used for search and optimization. Position adjustment of prototypes can be viewed as a search problem, thus it could be solved using evolutionary algorithms. In this paper, we perform a preliminary study on the use of differential evolution algorithms to the prototype generation problem. Differential evolution models are compared with other algorithms for adjusting the position of prototypes and the results are contrasted through non-parametrical statistical tests. The results show that some differential evolution models consistently outperform previously proposed methods.

I. Introduction

The k-Nearest Neighbors rule (kNN) [1] is one of the most known and used nonparametric classifier in Machine Learning and Data Mining (DM) tasks [2]. It is included in a more specific field of DM known as lazy learning, which refers to the set of methods that predict the class label from raw training data and do not obtain learning models. Although kNN is a simple technique, it has demonstrated itself to be one of the most interesting and effective algorithms in DM and pattern recognition.

Classification typically involves partitioning samples into training and testing categories. Let $\mathbf{x}_p$ be a training sample from n available samples in the training set. Let $\mathbf{x}_t$ be a test sample, and let ω be the true class of a training sample and $\hat{\omega}$ be the predicted class for a test sample ($\omega, \hat{\omega} = 1, 2, \ldots, \Omega$). Here, Ω is the total number of classes. During the training process, we use only the true class ω of each training sample to train the classifier, while during testing we predict the class $\hat{\omega}$ of each test sample. With the kNN rule, where k=1 (1NN), the predicted class of test sample $\mathbf{x}_t$ is set equal to the true class ω of its nearest neighbor, where $\mathbf{nn}_i$ is a nearest neighbor to $\mathbf{x}_t$ if the distance

$$d(\mathbf{nn}_i, \mathbf{x}_t) = \min_j \{d(\mathbf{nn}_j, \mathbf{x}_t)\}.$$

For kNN, the predicted class of test sample $\mathbf{x}_t$ is set equal to the most frequent true class among k nearest training samples. This forms the decision rule $D : \mathbf{x}_t \rightarrow \hat{\omega}$.

It is well known that kNN suffers from several drawbacks [3]. Mainly, four weaknesses cause a great impact in the successful application of this classifier. The first one is the necessity of high storage requirements in order to retain the set of examples which defines the training set and allows it to perform the decision rule. The second one is the low efficiency obtained during the computation of the decision rule, caused by multiple computations of similarities between the test and training samples. Third, kNN (especially 1NN) presents low tolerance to noise, due to the fact that it uses all data as relevant even when the training set contains incorrect data. Finally, kNN makes predictions over existing data and it assumes that input data perfectly delimitates the decision boundaries among classes.

Several approaches have been suggested and studied in order to tackle the drawbacks mentioned. One of the most important solutions consists of reducing data used for establishing a classification rule. Data reduction techniques [4] could be divided into two different approaches, known as prototype selection (PS) and prototype generation (PG) or abstraction. The main difference between both approaches is that PS methods assume that the best representative examples can be obtained from a subset of the original data whereas PG methods generate new representative examples if needed, tackling also the fourth disadvantage mentioned above [5].

Evolutionary Algorithms (EAs) are general-purpose search algorithms that use principles inspired by natural genetic populations to evolve solutions to problems. The basic idea is to maintain a population of chromosomes, which represent plausible solutions to the problem and evolve over time through a process of competition and controlled variation. EAs have been used in data reduction tasks with promising results [6], [7]

Differential evolution (DE) has been shown to be a simple yet efficient EA for many optimization problems in real-world applications [8]. Like other evolutionary algorithms, two fundamental processes drive the evolution of a DE population: the variation process, which enables exploring different regions of the search space, and the selection process, which ensures the exploitation of previous knowledge about the fitness landscape [9].

Some sub-processes within the PG problem can be viewed

I. Triguero and F. Herrera are with the Department of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071, Granada, Spain (email: triguero@decsai.ugr.es and herrera@decsai.ugr.es).

S. García is with the Department of Computer Science, University of Jaén, 23071, Jaén, Spain (e-mail: sglopez@ujaen.es).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

as a real valued optimization problem with high degree of difficulty. In fact, although most of the PG methods proposed are based on heuristics, some of them could be dealt with powerful optimization techniques, such the case of position adjustment of prototypes. Some of them are based on EAs, such as Genetic Algorithms [10], [11], Particle Swarm Optimization (PSO) [12], [13] and Artificial Immune Systems [14]. DE algorithms have shown an excellent performance in real valued optimization problems and, for this reason, we propose the use of DE algorithms to tackle the position adjustment of prototypes.

This paper is a preliminary study where we propose and analyze the use of all the known schemes of the basic DE algorithm [15], that is, 13 DE schemes will be studied. Recent advances in DE in real valued optimization problems are OBDE [16], SADE [17], DEGL [18], JADE [19] and SFLSDE [20], but they are out of the scope of this preliminary study. Then, we are interested in establishing the best choice for adjusting the position of prototypes in 1NN, and comparing this scheme with other two well known prototype position optimization schemes described below. The experimental study will include a statistical analysis based on nonparametric tests and we will conduct experiments involving a total of 25 classification data sets.

This paper is organized as follows. A background is given in Section II, describing the PG problem, introducing DE and a briefly summarizing the methods for adjusting prototypes positioning used in the experimental comparison. In Section III we present the DE algorithm proposed for tackling the position adjustment problem. Section IV examines and analyzes the results obtained and presents a discussion of them. Finally, Section V concludes the paper.

II. Background

This section is composed by three different parts. Subsection II-A, presents a background on PG. Next, subsection II-B shows the main characteristics of DE, and finally, subsection II-C collect a brief survey of the methods for adjusting the position of prototypes that we use to compare with our proposal.

A. Background on PG

PG is an important technique in data reduction. It has been widely applied to instance-based classifiers and can be defined as the application of instance construction algorithms over a data set to improve the classification accuracy of a nearest neighbor classifier.

More specifically, PG can be defined as follows: Let $\mathbf{x}_p$ be an instance where $\mathbf{x}_p = (\mathbf{x}_{p1}, \mathbf{x}_{p2}, ..., \mathbf{x}_{pm}, \mathbf{x}_{p\omega})$, with $\mathbf{x}_p$ belonging to a class ω given by $\mathbf{x}_{p\omega}$ and a m-dimensional space in which X_{pi} is the value of the i-th feature of the p-th sample. Then, let us assume that there is a training set TR which consists of n instances $\mathbf{x}_p$ and a test set TS composed by s instances $\mathbf{x}_t$. The purpose of PG is to obtain a prototype generate set, which consists of r, $r < n$, prototypes, which are either selected or generated from the examples of TR. The prototypes of the generated set are determined to represent efficiently the distributions of the classes and to discriminate well when used to classify the training objects. Their cardinality should be sufficiently small to reduce both the storage and evaluation time spent by a kNN classifier.

The PG approaches can be divided into several families depending on the main heuristic operation followed. The first approach we can find in the literature, called PNN [21] belongs to the family of methods that carry out merging of prototypes of the same class in successive iterations, generating centroids. One of the most important families of methods are those based on Learning Vector Quantization (LVQ) [22]. Other well-known methods are those based on a divide-and-conquer scheme, by separating the m-dimensional into two or more subspaces with the purpose of simplifying the problem in each step [23].

B. Differential Evolution

Differential evolution follows the general procedure of an EA. DE starts with a population of NP candidate solutions, so-called individuals. The initial population should better cover the entire search space as much as possible. In some problems, it is achieved by uniformly randomizing individuals, but in other problem, such as the considered in this contribution, a basic knowledge of the problem is available and the use of other initialization mechanisms is more effective. The subsequent generations in DE are denoted by $G = 0, 1, \ldots, G_{max}$.

It is usually to denote each individual as a D-dimensional vector $X_{i,G} = \{x^1_{i,G}, ..., x^D_{i,G}\}$, it is called "target vector".

1) Mutation Operation: After initialization, DE applies the mutation operator to generate a mutant vector $V_{i,G}$, with respect to each individual $X_{i,G}$, in the current population. For each target $X_{i,G}$, at the generation G, its associated mutant vector $V_{i,G} = \{V^1_{i,G}, ..., V^D_{i,G}\}$. The method of creating this mutant vector is that which differentiates one DE scheme from another. Six most frequently referred strategies are listed below:

- "DE/rand/1":

$$V_{i,G} = X_{r^i_1,G} + F \cdot (X_{r^i_2,G} - X_{r^i_3,G}) \tag{1}$$

- "DE/best/1":

$$V_{i,G} = X_{best,G} + F \cdot (X_{r^i_1,G} - X_{r^i_2,G}) \tag{2}$$

- "DE/rand-to-best/1":

$$V_{i,G} = X_{i,G} + F \cdot (X_{best,G} - X_{i,G}) + F \cdot (X_{r^i_1,G} - X_{r^i_2,G}) \tag{3}$$

- "DE/best/2":

$$V_{i,G} = X_{best,G} + F \cdot (X_{r^i_1,G} - X_{r^i_2,G}) + F \cdot (X_{r^i_3,G} - X_{r^i_4,G}) \tag{4}$$

- "DE/rand/2":

$$V_{i,G} = X_{r^i_1,G} + F \cdot (X_{r^i_2,G} - X_{r^i_3,G}) + F \cdot (X_{r^i_4,G} - X_{r^i_5,G}) \tag{5}$$

- "DE/rand-to-best/2":

$$V_{i,G} = X_{i,G} + F \cdot (X_{best,G} - X_{i,G}) + F \cdot (X_{r_1^i,G} - X_{r_2^i,G}) + F \cdot (X_{r_3^i,G} - X_{r_4^i,G}) \quad (6)$$

The indices r_1^i, r_2^i, r_3^i, r_4^i, r_5^i are mutually exclusive integers randomly generated within the range $[1, NP]$, which are also different from the base index i. These indices are randomly generated once for each mutation. The scaling factor F is a positive control parameter for scaling the difference vectors. $X_{best,G}$ is the best individual of the population in terms of fitness.

The general convention, used for naming these strategies, is $DE/a/b/c$, where DE stands for differential evolution, a represent a string denoting the vector to be perturbed, b denotes the number of difference vector considered to be perturbed, and c informs us about the crossover type used.

2) Crossover Operator: After the mutation phase, crossover operation is applied to increase the potential diversity of the population. DE algorithm can use two kinds of crossover schemes, known as 'Binomial' and 'Exponential' crossover. This operator is applied to each pair of the target vector $X_{i,G}$ and its corresponding mutant vector $V_{i,G}$ to generate a new trial vector that we denote $U_{i,G}$. The mutant vector exchanges its components with the target vector $X_{i,G}$.

Binomial crossover scheme is performed on all the components whenever a randomly picked number between 0 and 1 is less than or equal to the crossover rate (CR), The CR is a user-specified constant within the range $[0, 1)$, which controls the fraction of parameter values copied from the mutant vector. This scheme may be outlined as

$$U_{i,G}^j = \begin{cases} V_{i,G}^j & \text{if rand(0,1) } <= CR \text{ or } j = j_{rand} \\ X_{i,G}^j & \text{Otherwise} \end{cases} \quad j = 1, 2, ..., D. \quad (7)$$

In exponential crossover we select an integer N between $[1, NP]$. N is the starting point in the target Vector $X_{i,G}^N$ from where the crossover exchanges components with the mutation vector. It is necessary to choose another integer called L from the same interval $[1, NP]$. This integer L denotes the number of components that they will try to exchange.

$$U_{i,G} = \begin{cases} V_{i,G}^j & \text{for } j = \|N\|, \|N+1\|, ..., \|N+L-1\| \\ X_{i,G}^j & \text{for all other } j \in [1, D] \end{cases} \quad (8)$$

where the $\|\cdot\|$ denote a modulo function with modulus D.

There is a different kind of crossover operator called arithmetic crossover that it replace (7) and (8) with the rotationally invariant arithmetic crossover operator to generate the trial vector $U_{i,G}$ by linearly combining the target vector X_{i_G} and $V_{i,G}$, like this,

$$U_{i,G} = X_{i,G} + K \cdot (V_{i,G} - X_{i,G}) \quad (9)$$

Where K is the combination coefficient which is usually used in the interval $[0, 1]$. This strategy is known as "DE/current-to-rand/1".

3) Selection Operator: When the trial vector has been generated we must decide which individual between X_{i_G} and $U_{i,G}$ should survive in the population to the next generation $G+1$. If the new trial vector yields an equal or better solution that the target vector, it replaces the corresponding target vector in the next generation; otherwise the target is retained in the population. Therefore, the population always get better or retains the same fitness values, but never deteriorates.

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } f(U_{i,G}) \text{ is better than } f(X_{i,G}) \\ X_{i,G} & \text{Otherwise} \end{cases} \quad (10)$$

This one-to-one selection procedure is generally kept fixed in most of DE algorithms.

C. Other studied techniques for adjusting the position of prototypes

We attempt to describe LVQ and PSO as the main contributors in this field, taking into account that they will be used in the experimental study.

1) LVQ: Learning Vector Quantization: Learning vector quantization was proposed by Kohonen in [22]. LVQ can be understood as a special case of artificial neural network in which a neuron corresponds to a prototype and a competition weight based process is carried out in order to locate each neuron in a concrete place of the m-dimensional space to increase the classification accuracy.

Specifically, let us assume that $\mathbf{x}_p$ is an instance which belongs to TR, and the prototypes or codebooks are denoted $\mathbf{m}_i(t) : \mathbf{m}_i(t)\, \varepsilon\, TR$, $i = 1, 2, \ldots, r$, where r is the number of prototypes for the reduced set, and t is the time coordinate.

A sequence of patterns $\mathbf{x}_p$ will be used to learn. The r prototypes or codebooks $\mathbf{m}_i(t)$ will be the final reduced set, they are initialized in some proper way; random selection will be often suitable, and then, they are modified during the learning phase using the sequence of $\mathbf{x}_p$.

The modifications are based on the concepts of reward and punishment. For each $\mathbf{x}_p$ one finds the $\mathbf{m}_c$ closest to it, according to some distance function. The position of $\mathbf{m}_c$ is updated, if $\mathbf{x}_p$ and $\mathbf{m}_c$ belong to the same class, then $\mathbf{m}_c$ is made closer to $\mathbf{x}_p$ (reward, equation (11)) , otherwise, $\mathbf{m}_c$ is moved away from $\mathbf{x}_p$ (punishment, equation (12)). A parameter $\alpha(t)$ is used to control the movement.

$$m_c(t+1) = m_c(t) + \alpha(t)[x_p - m_c(t)] \quad (11)$$

$$m_c(t+1) = m_c(t) - \alpha(t)[x_p - m_c(t)] \quad (12)$$

This procedure is known as LVQ1, Kohonen has subsequently proposed modifications, LVQ2 and LVQ3, in order to provide an improved performance near decision borders. LVQ3 has reported the best results, and it consists on finding the two prototypes $\mathbf{m}_i(t)$ and $\mathbf{m}_j(t)$ nearest to $\mathbf{x}_p$, and which

are relatively near to themselves according to a parameter, known as *window size*. On the one hand, let $\mathbf{m}_j(t)$ be an instance with the same class as $\mathbf{x}_p$, and $\mathbf{m}_i(t)$ with different class, they are rewarded and punishment, respectively. On the other hand, if both instances belong to the same class as $\mathbf{x}_p$, both are rewarded.

2) PSO: Particle Swarm Optimization: Particle Swarm Optimization [24] is a evolutionary algorithm which is based on the social behavior of biological organisms, e.g. the movement of bird flocking. This algorithm has been applied in different fields, and it was proposed for PG by Nanni and Lumini in [13].

The basic idea for applying this technique to PG is the following. Initially, a population is initialized with NP random solutions, each solution is called "particle" $X_{i,G}$. Each particle is composed of a random small set of r prototypes which is selected from the TR. Then, the position of these prototypes is adjusted using the PSO rules, attempting to minimize the classification error.

Each particle $X_{i,G}$ has an associate velocity $V_{i,G}$ and the best previous position of each particle is recorded as $X_{best,i}$. At each iteration particles are updated according to the equation (13), previouly, the new velocity is calculated following (14). This velocity equation provides to the PSO algorithm the balance between global and local exploration. Global exploration is the social part, i.e. the particle moves according to the best particle position (P_{best}), and local exploration is the cognitive part, where the particle moves according to its best position ($X_{best,i}$).

$$X_{i,G+1} = X_{i,G} + V_{i,G+1}; \tag{13}$$

$$V_{i,G+1} = w \cdot V_{i,G} + c_1 Rand()(X_{best,i} - X_{i,G}) + c_2 Rand()(P_{best} - X_{i,G}). \tag{14}$$

The balance is controlled by two parameters, c_1 and c_2. Furthermore, the influence of the last velocity is gradually decreased with a new parameter w, known as *inertial weight* [**?**].

III. DE for Prototype Generation

In this section we propose to apply the underlying idea in DE for PG problem as a position adjusting of prototypes scheme.

First of all, it is necessary to define the solution codification. In the proposed DE algorithm, each individual in the population encodes a complete solution, that is, a reduced set of prototypes are encoded sequentially in each individual. This type of codification is known as the Pittsburgh approach [25]. The number of prototypes encoded in each individual will define its *individual size* and it is denoted r as previously. An user parameter will set this value r.

Following the notation used in subsection II-B, $X_{i,G}$ define the target vector, but in this case, the target vector could be represented as a matrix. Table I describes the structure of an individual.

TABLE I
ENCODING OF A SET OF PROTOTYPES IN A INDIVIDUAL FOR THE DE ALGORITHM

	Attribute 1	**Attribute 2**	**...**	**Attribute m**	**Class**
Prototype 1	$x_{1,p1}$	$x_{2,p1}$	...	$x_{m,p1}$	ω_{p1}
Prototype 2	$x_{1,p2}$	$x_{2,p2}$	...	$x_{m,p2}$	ω_{p2}
...					
Prototype r	$x_{1,pr}$	$x_{2,pr}$	...	$x_{m,pr}$	ω_{pr}

A. Initialization

Given that this problem provides some knowledge based on the initial arrangement of training samples, DE initializes its NP individuals by choosing r random prototypes with its respective class from the TR for each one. In this initialization process, we ensure that every class has at least a representative prototype. We use different a priori probabilities, i.e. the number of representatives instances for each class is proportional to the number of them in the TR. It is important to point out that every solution must have the same structure, thus they must have the same number of representatives per class, and they must have the same order.

B. Mutation Operator

The six mutation strategies explained in section II, have been implemented to generate the mutant matrix $V_{i,G}$, with respect to each individual $X_{i,G}$, in the current population. The operations of addition, subtraction and scalar product are carried out as typical matrices. This is the reason that justifies the individuals to have the same structure, in order to give sense to this mutation operator.

After applying this operator, it is necessary to check that the mutant matrix $V_{i,G}$ has been generated with correct values for all features of the prototypes, i.e. we need to check that the values are in the correct range. We normalize all attributes of the data set to the $[0, 1]$ range, so this procedure only needs to check if there have been values out of $[0, 1]$.

C. Crossover Operator

With the idea of increasing the potential diversity of the population, we define the three crossover operators; binomial, exponential and arithmetic, for our scheme. Instead of interchanging attributes values, the mutant matrix $V_{i,G}$ exchanges its prototypes with the target $X_{i,G}$ to generate a new trial matrix $U_{i,G}$.

D. Selection Operator

This operator must decide which individual between X_{i_G} and $U_{i,G}$ should survive in the population to the next generation $G+1$. The 1NN rule guides this operator. The instances in TR are classified with the prototype encoded in each individual X_{i_G} or $U_{i,G}$ with 1NN, and the corresponding fitness values of them is measured as the classification rate or accuracy obtained, measured as the number of successful hits (correct classifications) relative to the total number of classifications. We try to maximize this value, so the selection operator can be viewed as follow:

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{if } accuracy(U_{i,G}) >= accuracy(X_{i,G}) \\ X_{i,G} & \text{Otherwise} \end{cases} \quad (15)$$

IV. Experimental Framework and results

In this section we show the factors and issues related to the experimental study. We provide the details of the problems chosen for the experimentation and the parameters of the algorithms in subsection IV-A. In subsection IV-B we compare the different schemes of DE and identify the best DE scheme for position adjustment of prototypes by using statistical tests. Finally, subsection IV-C shows a comparative study of the best DE with other PG techniques.

A. Experimental Framework

The performance of the algorithms is analyzed by using 25 datasets from the UCI repository [26]. Table II summarizes the properties of the selected data sets. It shows, for each data set, the number of examples (#Ex.), the number of attributes (#Atts.), the number of numerical (#Num.) and nominal (#Nom.) attributes, and the number of classes (#Cl.). The data sets considered are partitioned using the ten fold cross-validation (10-fcv) procedure.

TABLE II
Summary description for classification data sets

Data Set	#Ex.	#Atts.	#Num.	#Nom.	#Cl.
appendicitis	106	7	7	0	2
balance	625	4	4	0	3
bands	539	19	19	0	2
bupa	345	6	6	0	2
car	1,728	6	0	6	4
cleveland	297	13	13	0	5
contraceptive	1,473	9	6	3	3
dermatology	366	33	1	32	6
ecoli	336	7	7	0	8
glass	214	9	9	0	7
haberman	306	3	3	0	2
hayes-roth	133	4	4	0	3
heart	270	13	6	7	2
hepatitis	155	19	19	0	2
iris	150	4	4	0	3
led7digit	500	7	0	1	10
mammographic	961	5	0	5	2
monks	432	6	6	0	2
pima	768	8	8	0	2
saheart	462	9	8	1	2
tic-tac-toe	958	9	0	9	2
wine	178	13	13	0	3
wisconsin	683	9	9	0	2
yeast	1,484	8	8	0	10
zoo	101	17	0	17	7

The parameters of the used algorithms are presented in Table IV-A. These values have been established in order to compare fairly between the algorithms. Hence, the reduction rate has been fixed to 95% with the same values for all algorithms. Furthermore, the EAs considered, PSO and DE, have been initialized with the same number of particles or individuals. The rest of parameters values are chosen according to the respective authors of the algorithms, assuming that the choice of the values of the parameter was optimally chosen. All methods used 1NN as baseline classifier with Euclidean distance.

TABLE III
Parameter specification for all the methods employed in the experimentation

Algorithm	*Parameters*
LVQ3	Iterations = 500, α = 0.1, WindowWidth=0.2, $epsilon$ = 0.1, Number of Prototypes (r) = 5%
PSO	SwarmSize = 40, Iterations = 500, ParticleSize (r) = 5%, C1 = 1, C2 = 3, Vmax = 0.25, Wstart = 1.5, Wend = 0.5
DE	PopulationSize = 40, Iterations = 500 , IndividualSize (r) = 5%, F = 0.5, CR =0.9

B. Results of DE schemes

Tables IV and V show the results in test data obtained by 13 different schemes of DE in terms of accuracy. These tables collect the average and standard deviation (SD) in accuracy obtained by them over the 25 data sets considered. The best result for each data set is remarked in bold (taking into account both tables).

Some observations can be pointed out from these tables:

- The best performing models are those based on Binomial crossover.
- Exponential crossover may produce a reduced number of perturbations, causing low exploration capabilities. Hence, exponential crossover schemes probably get stuck at a local optimum.
- Among them, schemes with RandToBest mutation scheme outperform the rest of schemes. These kind of schemes perform a good balance between exploration (random individual) and exploration (best individual) and have a fast convergence speed [17].
- In principle, we can point out the DE/RandToBest/1/Bin as the best model between both *RandToBest* models.
- The number of difference vectors to be perturbed by the mutation operator does not seems to be an important factor that influence over the final result obtained.

Considering only average results could lead us to erroneous conclusions. Due to this fact, we will accomplish statistical comparisons over multiple data sets based on non-parametric tests [27], [28]. Specifically, we will use the Wilcoxon Signed-Ranks test [29]. Table VI collects the results of applying Wilcoxon test among the 13 DE schemes studied. With a level of significance $\alpha = 0.10$, the table reflects which schemes are outperformed by other. When the scheme situated in the row is better than that situated in the column, a '+' is used. In contrary case, a '–' symbol is used. In case of ties, a '=' symbol is used. The p-value obtained is depicted in the opposite diagonal of the table. The two last

TABLE IV
RESULTS OF DE SCHEMES USING BINARY CROSSOVER

Datasets	DE/Rand/1/Bin		DE/Best/1/Bin		DE/RandToBest/1/Bin		DE/Best/2/Bin		DE/Rand/2/Bin		DE/RandToBest/2/Bin	
	Acc	SD	Acc	SD	Acc	SD	Acc	SD	Acc	SD	Acc	SD
appendicitis	86.09	9.19	85.91	9.48	**86.91**	7.34	85.09	8.31	86.09	10.84	83.27	11.28
bal	85.46	2.81	83.5	3.43	**87.52**	3.67	85.14	3.98	81.76	3.65	86.55	3.75
bands	66.81	5.8	67	6.6	70.14	4.1	67.53	3.9	67.36	6.64	**70.15**	5.64
bupa	60.75	10.34	61.8	11.6	62.31	8.87	62.89	8.38	62.66	8.81	**64.96**	3.49
car	82	2.12	83.85	2.02	83.45	2.39	**84.43**	2.59	82.64	2.53	81.77	2.22
cleveland	56.11	6.05	56.81	8.68	53.46	6.72	57.42	5.43	53.84	5.72	58.78	9.05
contraceptive	47.05	2.98	44.54	2.61	46.51	3.16	46.23	2.91	42.84	3.84	45.62	3.73
dermatology	94.8	4.13	**96.72**	1.63	96.18	2.77	95.63	4.24	96.18	2.16	95.65	3.86
ecoli	70.28	4.55	75.93	7.68	77.15	7.41	75.66	7.95	77.72	6.55	**79.2**	7.14
glass	61.51	11.16	62.03	9.82	**65.73**	8.1	64.1	12.61	61.51	8.3	62.34	9.42
haberman	73.22	4.48	69.55	6.21	71.19	5.85	69.91	3.71	72.88	3.17	71.19	4.88
hayes-roth	54.89	17.43	63.3	10.68	**68.75**	12.57	55.63	15.33	65.68	14.18	65.62	11.19
heart	81.85	8.52	81.11	9.14	81.11	9.14	80	10.37	80.37	12.29	82.59	6.21
hepatitis	**85.08**	8.95	82	11.03	83.25	10.02	78.79	9.89	81.96	6.18	84.58	8.28
iris	94.67	4	95.33	3.06	95.33	3.06	95.33	4.27	96	4.42	94	4.67
led7digit	57.4	4.29	71.4	3.9	**71.8**	4.6	**71.8**	4.51	70.2	4.33	71.2	3.71
mammographic	**81.69**	4.15	80.13	3.39	80.02	3.96	79.5	3.5	79.61	5.63	79.81	4.12
monks	**92.89**	4.11	79.18	6.74	89.43	6.13	80.6	8.59	79.11	4.32	85.54	4.43
pima	75.27	3.04	73.85	3.74	75.66	3.78	75.53	4.15	73.57	4.72	**76.84**	3.77
saheart	70.36	6.21	71.22	4.7	70.11	5.65	68.83	3.35	69.49	4.08	69.7	3.19
tic-tac-toe	69.21	3.66	72.35	4.11	**76.51**	2.94	72.55	6.4	73.39	4.85	72.65	4.6
wine	90.88	10.63	95.52	4.85	95.46	4.27	92.75	8.25	91.63	12.48	**96.63**	4.46
wisconsin	96.85	1.54	96.57	2.14	95.85	2.16	96.42	1.46	95.71	2.12	96.28	1.59
yeast	52.16	3.16	55.46	5.33	55.46	2.2	57.35	1.96	52.15	3.12	**58.09**	2.64
zoo	93.89	5.26	94.83	7.24	96.39	4.55	95.5	6.41	96.5	6.26	**98.5**	3.02
AVERAGE	75.25		76		**77.43**		75.78		75.63		77.26	

TABLE V
RESULTS OF DE SCHEMES USING EXPONENTIAL AND ARITHMETIC CROSSOVER

Datasets	DE/Rand/1/Exp		DE/Best/1/Exp		DE/RandToBest/1/Exp		DE/Best/2/Exp		DE/Rand/2/Exp		DE/RandToBest/2/Exp		DE/CurrentToRand/1	
	Acc	SD	Acc	SD	Acc	SD	Acc	SD	Acc	SD	Acc	SD	Acc	SD
appendicitis	84.18	10.75	86	9.33	84.18	10.75	86	9.33	**86.91**	8.39	86	8.4	86	9.33
bal	87.04	2.3	84.95	4.34	87.04	2.3	84.95	4.34	86.07	4.65	84.79	5.6	87.04	2.21
bands	66.61	4.28	67.54	5.06	66.61	4.28	67.54	5.06	65.69	5.57	67.36	6.48	68.28	4.53
bupa	56.09	10.84	56.24	9.29	56.09	10.84	56.24	9.29	59.37	8.62	57.15	8.05	62.8	9.61
car	81.89	2.86	82.64	2.36	81.89	2.86	82.64	2.36	82.93	2.24	79.57	2.38	82	2.24
cleveland	57.76	4.54	58.76	7.45	58.06	7.03	58.09	6.09	**59.46**	7.05	57.41	4.44	58.75	4.89
contraceptive	47.8	4.18	**49.01**	3.98	47.8	4.18	**49.01**	3.98	46.16	2.78	46.71	3.7	47.93	2.62
dermatology	90.41	4.13	90.7	3.3	93.18	3.3	92.61	4.89	93.76	4.68	91.56	4.57	91.82	3.6
ecoli	70.27	6.18	71.77	6.62	70.55	6.29	72.98	7.44	74.43	4.35	73.57	5.93	74.71	3.92
glass	57.3	8.69	56.59	9.44	58.93	13.19	61.19	10.48	55.2	9.77	59.69	13.06	57.31	8.29
haberman	**77.44**	5.35	72.19	2.51	**77.44**	5.35	72.19	2.51	70.55	4.65	72.22	4.45	74.49	7.54
hayes-roth	44.66	10.21	52.85	12.86	44.66	10.21	52.85	12.86	52.52	15.11	52.99	14.52	48.4	8.43
heart	81.11	11.88	**83.7**	6.87	81.11	11.88	**83.7**	6.87	81.48	7.59	81.11	9.14	82.59	8.61
hepatitis	81.33	6.07	78.88	10.66	81.33	6.07	78.88	10.66	79.42	8.42	80.08	7.23	77.54	9.73
iris	94.67	5.81	95.33	4.27	94.67	5.81	95.33	4.27	**96.67**	4.47	96	4.42	93.33	2.98
led7digit	51.4	7.43	47.2	7.44	51.4	7.43	47.2	7.44	50.6	7.1	53.4	5.66	49.6	7.94
mammographic	80.75	4.85	79.61	4.51	80.75	4.85	79.61	4.51	81.17	4.87	79.08	3.07	79.71	4.52
monks	83.87	8.6	80.92	8.41	83.87	8.6	80.92	8.41	83.91	8.28	75.82	7.72	91.92	6.01
pima	75.4	3.1	74.49	3.7	75.4	3.1	74.49	3.7	73.97	4.46	75.15	4.53	76.31	4.4
saheart	71.22	4.2	69.93	5.33	71.22	4.2	69.93	5.33	69.04	3.26	**72.07**	4.53	70.78	3.4
tic-tac-toe	71.3	5.14	68.59	2.93	71.3	5.14	68.59	2.93	69.81	7.05	68.16	6.32	69.63	3.42
wine	94.93	3.01	94.38	5.56	94.93	3.01	94.38	5.56	93.82	5.31	95.52	5.45	94.35	6.27
wisconsin	96.57	2.57	96.57	2.32	96.57	2.57	96.57	2.32	**96.99**	2.26	96.57	2.32	96.13	2.23
yeast	49.94	4.95	51.21	2.51	47.78	5.84	50.14	3.58	49.4	4.5	50.61	4.54	51.69	4.38
zoo	65	10.36	65.08	10.89	91.92	7.27	93.44	8.05	87.64	9.5	89.64	9.01	89.47	11.24
AVERAGE	72.76		72.61		73.95		73.98		73.88		73.69		74.5	

columns respectively show the number of ties and wins for each scheme in the corresponding row.

As we can see, the observations mentioned above are now statistically contrasted and justified. We could choose DE/RandToBest/1/Bin and DE/RandToBest/2/Bin as the best performing schemes, due to the fact that they statistically

TABLE VI
WILCOXON TEST AMONG ALL THE DE SCHEMED CONSIDERED

	DE/Rand/1/Bin	DE/Best/1/Bin	DE/RandToBest/1/Bin	DE/Best/2/Bin	DE/Rand/2/Bin	DE/RandToBest/2/Bin	DE/Rand/1/Exp	DE/Best/1/Exp	DE/RandToBest/1/Exp	DE/Best/2/Exp	DE/Rand/2/Exp	DE/RandToBest/2/Exp	DE/CurrentToRand/1	TIES	WINS
DE/Rand/1/Bin	•	=	–	=	=	–	+	+	=	=	+	+	=	6	4
DE/Best/1/Bin	0.326	•	–	=	=	–	+	+	=	=	+	+	=	6	4
DE/RandToBest/1/Bin	0.042	0.009	•	+	+	=	+	+	+	+	+	+	+	1	11
DE/Best/2/Bin	0.339	0.786	0.010	•	=	–	=	=	=	=	=	+	=	9	1
DE/Rand/2/Bin	0.976	0.282	0.001	0.443	•	–	=	=	=	=	=	+	=	9	1
DE/RandToBest/2/Bin	0.042	0.009	0.742	0.016	0.007	•	+	+	+	+	+	+	+	1	11
DE/Rand/1/Exp	0.067	0.077	0.009	0.242	0.339	0.009	•	=	=	=	=	=	=	8	0
DE/Best/1/Exp	0.069	0.055	0.002	0.126	0.207	0.006	0.689	•	=	=	=	=	–	7	0
DE/RandToBest/1/Exp	0.123	0.144	0.010	0.264	0.476	0.010	0.173	0.265	•	=	=	=	=	10	0
DE/Best/2/Exp	0.158	0.107	0.003	0.175	0.346	0.007	0.742	0.116	0.954	•	=	=	=	10	0
DE/Rand/2/Exp	0.069	0.083	0.002	0.158	0.253	0.005	0.638	0.242	0.778	0.737	•	=	=	8	0
DE/RandToBest/2/Exp	0.028	0.017	0.002	0.054	0.089	0.001	0.648	0.465	0.605	0.523	0.861	•	=	6	0
DE/CurrentToRand/1	0.607	0.276	0.040	0.397	0.757	0.034	0.166	0.072	0.658	0.511	0.242	0.440	•	9	1

TABLE VII
AVERAGE RESULTS OBTAINED BY THE BEST DE MODEL AND OTHER PG APPROACHES

Datasets	**1NN**		**LVQ3**		**PSO**		**DE/RandToBest/1/Bin**	
	Acc	**SD**	**Acc**	**SD**	**Acc**	**SD**	**Acc**	**SD**
appendicitis	79.36	11.51	83.18	8.94	**87.91**	8.1	86.91	7.34
bal	79.04	6.46	80.97	3.69	86.39	2.65	**87.52**	3.67
bands	63.09	4.65	67.01	6.85	68.29	8.25	**70.14**	4.1
bupa	61.08	6.88	57.69	8.81	**63.75**	8.39	62.31	8.87
car	**85.65**	1.81	79.22	2.58	84.72	2.41	83.45	2.39
cleveland	53.14	7.45	57.81	5	**58.11**	5.49	53.46	6.72
contraceptive	42.77	3.69	40.73	3.89	45.62	2.35	**46.51**	3.16
dermatology	95.35	3.45	94.81	4.29	95.65	2.14	**96.18**	2.77
ecoli	**80.7**	7.51	72.69	8.34	77.36	4.13	77.15	7.41
glass	**73.61**	11.91	58.92	7.22	61.17	9.8	65.73	8.1
haberman	66.97	5.46	**71.58**	4	71.22	3.71	71.19	5.85
hayes-roth	35.7	9.11	40.66	20.98	65.74	11.46	**68.75**	12.57
heart	77.04	8.89	74.44	6.51	78.15	8.19	**81.11**	9.14
hepatitis	80.75	11.09	78.08	8.74	76.92	10.28	**83.25**	10.02
iris	93.33	5.16	90.67	8.54	93.33	4.22	**95.33**	3.06
led7digit	40.2	9.48	65.8	5.47	71.6	3.98	**71.8**	4.6
mammographic	73.68	5.59	70.76	3.88	**80.75**	4.03	80.02	3.96
monks	77.91	5.42	75.49	8.86	85.65	6.89	**89.43**	6.13
pima	70.33	3.53	68.08	3.92	75.41	4.06	**75.66**	3.78
saheart	64.49	3.99	67.52	5.49	69.92	4.36	**70.11**	5.65
tic-tac-toe	73.07	2.56	69.84	2.11	72.44	3.34	**76.51**	2.94
wine	95.52	4.85	94.93	1.69	94.35	5.76	**95.46**	4.27
wisconsin	95.57	2.59	96.42	2.23	**96.71**	1.82	95.85	2.16
yeast	50.47	3.91	47.51	4.84	55.26	3.02	**55.46**	2.2
zoo	92.81	6.57	82.92	15.26	93.75	5.31	**96.39**	4.55
AVERAGE	72.07		71.51		76.41		**77.43**	

TABLE VIII
RESULTS OF THE WILCOXON TEST COMPARING THE BEST DE WITH OTHER APPROACHES

Comparison	Result ($\alpha = 0.05$)	p-value
DE vs. 1NN	Better	0.001
DE vs. LVQ3	Better	0
DE vs. PSO	Better	0.042

behave the same. However, we prefer to choose the former basing on the average accuracy results achieved in Table IV.

C. Comparison with other PG models

Table VII shows the results in test data over the 25 data sets used for 1NN, LVQ3, PSO and our best DE scheme. As before, Wilcoxon test has been applied to contrast the results. Table VIII shows the statistical comparison.

Observing the results, we can make the following analysis:

- All the DE schemes have reported better average results than LVQ3 and 1NN algorithms.
- DE/rand-To-Best/1/bin proposal obtains the best average results in accuracy measure with the same reduction rate as LVQ3 and PSO algorithms. Note that the same size of population and number of iterations have been also used for both Evolutionary schemes PSO and DE. It clearly outperforms the other techniques with a great support, given by the Wilcoxon p-value. Furthermore, the baseline algorithm, 1NN, has been outperformed with a wide difference in terms of accuracy and also with a high reduction rate.

V. CONCLUSIONS

The purpose of this contribution is to present a preliminary study of the differential evolution algorithm for data reduction tasks. In concrete, it was used for optimizing the position of prototypes for the nearest neighbor algorithm, acting as a prototype generation method. This algorithm achieves a good balance between exploration and explotation which allows it to achieve results that are statistically better than other methods shown. With a proper choice of the type of DE scheme, other prototype generation algorithms such as LVQ and PSO can be easily improved.

ACKNOWLEDGMENT

Supported by the Spanish Ministry of Science and Technology under Project TIN2008-06681-C06-01.

REFERENCES

[1] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[2] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. MIT Press, Cambridge, MA, 2010.

[3] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.

[4] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.

[5] M. Lozano, J. M. Sotoca, J. S. Sánchez, F. Pla, E. Pekalska, and R. P. W. Duin, "Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces," *Pattern Recognition*, vol. 39, no. 10, pp. 1827–1838, 2006.

[6] J.-R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561–575, 2003.

[7] S. García, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach," *Pattern Recognition*, vol. 41, no. 8, pp. 2693–2709, 2008.

[8] R. Storn and K. V. Price, "Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 10, pp. 341–359, 1997.

[9] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 61–106, 2010.

[10] X. Llora and J. Garrell, "Inducing partially-defined instances with evolutionary algorithms," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 337–344.

[11] F. Fernández and P. Isasi, "Evolutionary design of nearest prototype classifiers," *Journal of Heuristics*, vol. 10, no. 4, pp. 431–454, 2004.

[12] A. Cervantes, I. M. Galván, and P. Isasi, "AMPSO: A new particle swarm method for nearest neighborhood classification," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 39, no. 5, pp. 1082–1091, 2009.

[13] L. Nanni and A. Lumini, "Particle swarm optimization for prototype reduction," *Neurocomputing*, vol. 72, no. 4-6, pp. 1092–1097, 2008.

[14] U. Garain, "Prototype reduction using an artificial immune model," *Pattern Analysis and Applications*, vol. 11, no. 3-4, pp. 353–363, 2008.

[15] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*, ser. Natural Computing Series, 2005.

[16] S. Rahnamayan, H. Tizhoosh, and M. Salama, "Opposition-based differential evolution," *IEEE Transaction on evolutionary computation*, vol. 12, no. 1, pp. 64–79, 2008.

[17] A. K. Qin, V. Huang, and P. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transaction on evolutionary computation*, vol. 13, no. 2, pp. 398–417, 2009.

[18] S. Das, A. Abraham, U. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transaction on evolutionary computation*, vol. 13, no. 3, pp. 526–553, 2009.

[19] J. Zhang and A. Sanderson, "Adaptive differential evolution with optional external archive," *IEEE Transaction on evolutionary computation*, vol. 13, no. 5, pp. 945–958, 2009.

[20] F. Neri and V. Tirronen, "Scale factor local search in differential evolution," *Memetic Computing*, vol. 1, no. 2, pp. 153–171, 2009.

[21] C.-L. Chang, "Finding prototypes for nearest neighbor classifiers," *IEEE Transactions on Computers*, vol. 23, no. 11, pp. 1179–1184, 1974.

[22] T. Kohonen, "The self organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[23] C. H. Chen and A. Jóźwik, "A sample set condensation algorithm for the class sensitive artificial neural network," *Pattern Recognition Letters*, vol. 17, no. 8, pp. 819–823, 1996.

[24] J. Kennedy and R. Eberhart, "Learning representative exemplars of concepts: An initial case study," in *Proceedings of the IEEE International Converence on Neural Networks*, 1995, pp. 1942–1948.

[25] A. Fernández, S. García, J. Luengo, E. Bernadó-Mansilla, and F. Herrera, "Genetics-based machine learning for rule induction: State of the art, taxonomy and comparative study." *IEEE Transactions on Evolutionary Computation*, in press, doi: 10.1109/TEVC.2009.2039140.

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

[27] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[28] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[29] D. Sheskin, *Handbook of parametric and nonparametric statistical procedures*, 2nd ed. Chapman & Hall/CRC, 2006.