Computing and Informatics, Vol. 34, 2015, 1151–1167

DIFFERENTIAL EVOLUTION BASED MULTIPLE VECTOR PROTOTYPE CLASSIFIER

Pasi Luukka

Laboratory of Applied Mathematics
Lappeenranta University of Technology
P. O. Box 20, FI-53851 Lappeenranta, Finland
&
School of Business
Lappeenranta University of Technology
P.O. Box 20, FI-53851 Lappeenranta, Finland
e-mail: pasi.luukka@lut.fi

Jouni Lampinen

Department of Computer Science
University of Vaasa
P. O. Box 700, FI-65101 Vaasa, Finland
&
VSB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
e-mail: jouni.lampinen@uwasa.fi

Abstract. In this article we introduce differential evolution based multiple vector prototype classifier (shortly MVDE). In this method we extend the previous DE classifier so that it can handle several class vectors in one class. Classification problems which are so complex that they are simply not separable by using distance based algorithms e.g. differential evolution (DE) classifier or support vector machine (SVM) classifier have troubled researchers for years. In this article, we propose a solution for one area of this problem type in which we extend DE classifier in a way that we allow several class vectors to exist for optimizing one class. This way a part of such complex data can be handled by one vector and other part can be handled by another vector. Differential evolution algorithm is a clear choice for handling such a multiple vector classification tasks because of its remarkable optimization capabilities. MVDE classifier is tested with several different benchmark classification problems to show its capabilities and its performance is compared to DE classifier, SVM and backpropagation neural network classifier. MVDE classifier managed to get best classification performance of these classifiers and clearly indicates it has a potential in this type of classification problems.

Keywords: Optimization, classifier, multiple vector prototype, differential evolution algorithm, evolutionary algorithm

1 INTRODUCTION

Classification problems of complex data structures in which separation cannot be made by using simply distance based methods were already clear several decades ago. Decision tree based classification algorithms (e.g. ID3 [1], ID5R [2], CN2 [3] to mention some) were among first methods which tackled this type of classification problems. Quite soon after it was found to be a suitable area also for neural network based classification algorithms [4, 5]. These methods became popular quite quickly and people started to apply them even in such problems which were not so complex and where distance based methods could have been used also. Later in 90's and begining of this millenium faster and accurate distance based algorithms e.g. support vector machine (SVM) [6] started to gain popularity because in many problems the classification speed and accuracy were becoming more important properties regarding classification problems and separable using distance based methods. Nowadays SVM is applied in a wide variety of application areas [7, 8].

One of the latest methods in evolutionary computation is differential evolution algorithm [9]. In general, evolutionary computation research has been of interest concerned the theory and practice of classifier systems [10, 11, 12, 13, 14, 15, 16, 17]. Also from differential evolution point of view the differential evolution algorithm is fastly gaining popularity in classification problems. Some include bankruptcy prediction [18], classification rule discovery [19], feature selection, [20], edge detection in images [21]. Also in some methods classification techniques are used to improve optimization with differential evolution algorithm [22]. The global optimization problems arise in many fields of science, engineering and business [23]. In order to achieve best performance, we need to find the best algorithm. One of the emerged methods in evolutionary computation is differential evolution (DE) algorithm [9]. DE has since then been used in many areas of pattern recognition, i.e. in remote sensing imagery [24], hybrid evolutionary learning in pattern recognition systems [25], and in clustering [26, 27] to mention few. Despite this, to our knowledge, evolutionary algorithms have not been much studied in cases where in classification task the data structure is so complex that it is not separable by applying simply a distance based method but more properties are needed from the classifier.

Here in this research we concentrate to tackle such problems where the data structure is so complex that it is not separable by applying simply a distance based method but more properties are needed from the classifier. This is done by extending differential evolution (DE) classifier [28] so that there can exist multiple class vector for one class. This extension allows us to handle such situation where a part of such complex data can be handled by one vector and other part can be handled by another vector allowing us to deal with such complex data sets where distance based methods are not enough. We will show that this method will be very useful in situations where a suitable distance based classification algorithm is not enough but more is needed. We will show that depending on the problem also the number of vectors needed in classifying the class correctly is very important and by choosing the correct number of vectors per class a best accuracy can be gained. If we have too many vectors per class the results start to deteriorate quite quickly and overlearning becomes evident.

From the optimization and modelling point of view the classification problem subject to our investigations can be divided into two parts: the classification model and the optimization approach applied for fitting (or learning) the model. Generally, a multipurpose classifier can be viewed as a scalable and learnable model that can be fitted to a particular dataset by scaling it to the data dimensionality and by optimizing a set of model parameters to maximize the classification accuracy. For the optimization, simply the classification accuracy over the learning set may serve as the objective function value to be maximized. Alternatively the optimization problem can be formulated as a minimization task, as we did here, where the number of misclassified samples is to be minimized. In the literature, mostly linear or nonlinear local optimization approaches has been applied for solving the actual classifier model optimization problem, or approaches that can be viewed as such. This is the most common approach despite of the fact that the underlying optimization problem is a global optimization problem. For example, the weight set of a feed-forward neural network classifier is typically optimized with a gradient-descent based on local optimizer [29], or alternatively by some other local optimizer like Levenberg-Marquardt algorithm (see i.e. [30]). This kind of usage of limited capacity optimizers for fitting the classification model limits the achievable classification accuracy in two ways. First, the model should be limited so that local optimizers can be applied to fit them. This means that only very moderately multimodal classification models can be applied, and due to such modelling limitation, the classification capability will be limited correspondingly. Secondly, if a local optimizer is applied to optimize (to fit or to learn) even a moderately multimodal classification model, it is likely to get trapped in a local optimum to a suboptimal solution. Thereby, the only way to get the classifier models with a higher modelling capacity at disposal, and also to get a full capacity out of the current multimodal classification models by applying global optimization for fitting the classification models to the data is to be classified. For example, in case of a nonlinear feed-forward neural network classifier, the model is clearly multimodal, but practically always fitted by applying a local optimizer that is capable of providing only locally optimal solutions. Thus, we consider that applying the global optimization instead of the local optimization is an important fundamental issue that currently severely constrains the further development of classifiers. The capabilities of currently used local optimizers are limiting aspects for the selection of applicable classifier models, and also the capabilities of currently used models that include multimodal properties are limited by the capabilities of the optimizers applied to fit them to the data.

Based on the above mentioned theoretical considerations, our basic motivation for applying a global optimizer for learning the applied classifier model comes from the fact that typically local (nonlinear) optimizers have been applied for the purpose. Despite that, the underlying optimization problem is actually a multimodal global optimization problem, and a local optimizer should be expected to become trapped into a local suboptimal solution. Therefore it was considered, that by applying global optimizers (like differential evolution algorithm), it is theoretically justified to expect improved classification results (in terms of classification accuracy). Later on this was confirmed also by the obtained results. Differential evolution algorithm was chosen to solve our classification problem because it proved to be very efficient in going through the search space to find the optimal solution [9]. It is very fast compared to several other global optimization methods, which is here quite a desirable property. We investigate this new classification method with classical DE method since it is the most studied one and the 'childhood' problems of the new optimization methods have been fixed.

Another motivation was that we wanted to optimize also the parameter p of the Minkowski distance. In practice, when this is applied to several different class vectors where some of them belong to same classes, it means increased nonlinearity and increased multimodality of the classification model resulting in more locally optimal points in the search space, where a local optimizer would be even more likely to get trapped. Practically, optimizing p successfully while also getting optimal class vectors where several of them can belong to the same class, requires usage of an effective global optimizer since local optimizers are anymore unlikely to provide even an acceptably good suboptimal solution. Otherwise, by using a global optimizer, optimization of p becomes possible. Two folded advantages were expected on this. First, by optimizing (systematically) the value for p, instead of selecting it a priori by trial and error as earlier, to reach a higher classification accuracy would be possible. Secondly, the selection of the p value can be done automatically this way, and laborious trial and error experimentation by the user is removed. Furthermore, a potential for the further development is increased. The local optimization approaches severely limit the selection of classifier models to be used and the problem formulations for classifier model optimization task become also limited. Simply, local optimizers are limited to fit or learn. Only classifier models work where trapping into a local suboptimal solution is not a major problem and global optimizers do not have such fundamental limitations. For example, the range of possible class membership functions can be extended to those requiring global optimization (due to increased nonlinearity and multimodality), and which cannot be handled anymore by simple local optimizers, even with the nonlinear ones. In addition, we would like to remark, that we have not yet fully utilized the further development capabilities provided by our global optimization approach. For example, even more difficult optimization problem settings are now within possibilities, and the differential evolution have good capabilities for multi-objective and multi-constrained nonlinear optimization that provides further possibilities for our future developments.

In this work multiple vector differential evolution classifier is presented. In Section 2 we present our classification method. In Section 3 we introduce several test classification problems where we benchmark and compare our method and in Section 4 we present discussion of the method.

2 MULTIPLE VECTOR DIFFERENTIAL EVOLUTION BASED CLASSIFICATION

The DE algorithm [31, 9] was first introduced by Storn and Price in 1995, and it belongs to the family of evolutionary algorithms (EAs). The design principles of DE are simplicity, efficiency, and the use of floating-point encoding instead of binary numbers for the internal representation of solution candidates to the problem to be solved. As a typical EA, DE has a random initial population of solution candidates that is then improved using selection, mutation, and crossover operations. Several ways exist to determine a stopping criterion for EAs, but usually a predefined upper limit G_{max} for the number of generations (can be viewed as iterations) to be computed provides an appropriate stopping condition in our case. This is due to the fact that in most of the real world classification problems 100 % accuracy cannot be even expected. In these cases misclassified samples are searched through entire search space and hence homogenity of population cannot be used. Other control parameters for DE are the crossover control parameter CR, the mutation factor F, and the population size NP.

DE examines a D dimensional decision vector $v_{i,g}$ in every generation g by going through the population and forming trial vector $t_{i,g}$ by using the most common DE version, DE/rand/1/bin [32][1]:

```
Select $r_{i_1}, r_{i_2}, r_{i_3}$, where $i_1, i_2, i_3 \in \{1, 2, \ldots, NP\}$ and $i_1 \neq i_2 \neq i_3 \neq i$ randomly.
$j_\phi = \lfloor D\phi_i \rfloor + 1$
For  $j := 1$ to $D$ do
    If($\phi_i < CR \cup j = j_\phi$)
        $t_{j,i,g} = v_{j,r_{i_3},g} + F \cdot \left(v_{j,r_{i_1},g} - v_{j,r_{i_2},g}\right)$
    Else
        $t_{j,i,g} = v_{j,i,g}$
    EndIf
EndFor
```

[1] Notation refers to classical version of DE algorithm

where random number $\phi \in U(0,1)$. In applied DE version, DE/rand/1/bin, NP should be set to at least four. Crossover control parameter $CR \in [0,1]$ represents the probability that an element for the trial vector is chosen from a linear combination of three randomly chosen vectors (disregarding the old vector $v_{i,G}$). The condition "$j = j_\phi$" is added to ensure that at least one element is different compared to the elements of the old vector. The parameter F is a scaling factor for mutation and its value is typically (0, 1]^2. Parameter CR controls the rotational invariance of the search. In practice smaller values of CR are practical in separable problems and larger values in non-separable problems. Parameter F effects to the speed and robustness of the search. Here, the lower value increases the convergence rate, but at the same time also adds the risk of getting stuck in a local optimum. For convergence rate also parameters CR and NP have the same kind of effect as F.

After the mutation and crossover operations, the trial vector $t_{i,g}$ is compared to the old vector $v_{i,g}$. If the trial vector has an equal or better objective value, it replaces the old vector in the next generation. This can be presented as follows (in this paper the minimization of objectives is assumed) [32]:

$$v_{i,g+1} = \begin{cases} t_{i,g} & \text{if} \quad f(t_{i,g}) \le f(v_{i,g}) \\ v_{i,g} & \text{otherwise.} \end{cases}$$

Since DE is an elitist method, the best population member is always preserved. Due to this reason the average objective value of the population will never decline.

As the objective function f needs to be minimized, we applied the number of incorrectly classified learning set samples. Each population member $\vec{v}_{i,G}$, as well as each new trial solution $\vec{u}_{i,G}$, contains the class vectors and the power value p. In other words, DE is seeking the vector $(y_1, \ldots, y_{N_1}, p)$ that minimizes the objective function f. After the optimization process the final solution, defining the optimized classifier, is the best member of the population of the last generation G_{max}, the individual $\vec{v}_{i,G_{max}}$. The best individual is the one providing the lowest objective function value, and therefore, the best classification performance for the learning set.

The control parameters of the DE algorithm were set as follows: $CR = 0.9$ and $F = 0.5$ were applied for all classification problems. NP was chosen so that it was six times the number of the optimized parameters D. These selections were based on general recommendations in the literature [9] and practical experiences with the usage of DE.

Next, we will take a look at the actual classification. The problem of classification is basically one of partitioning the feature space into regions, one region for each category. Ideally, one would like to arrange this partitioning so that none of the decisions is ever wrong [33].

The objective is to classify a set X of objects to N different classes $C_1, \ldots, C_N$ by their features. We suppose that T is the number of different kinds of features

^{2} Notation means that the upper limit is about 1.

that we can measure from objects. The key idea is to determine the ideal vector $\mathbf{y}_i$, which is closest to sample x_j

$$\mathbf{y}_i = (y_{i1}, \ldots, y_{iT}) \tag{1}$$

where now i is the integer number such that $i \in \{1, 2, \ldots, N_1\}$ and N_1 is the total number of possible ideal vectors. In our earlier work [28] we allowed one vector per each class. In this article the key idea is that there can be several vectors for one class. This is done so that we first find the closest vector $\mathbf{y}_i$ for sample $\mathbf{x}_j$ by using minimum distance

$$g(\mathbf{x}_j) = \left\{ i \;\middle|\; \min_{i=1,\ldots,N_1} d(\mathbf{x_j}, \mathbf{y_i}) \right\} \tag{2}$$

where now the distance between the sample vector and particular ideal vector is calculated using

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{j=1}^{T} |x_j - y_j|^p \right)^{1/p}. \tag{3}$$

Next we need to refit these ideal vector indexes $g(\mathbf{x_j})$ to correspond the actual classes and this is done by the procedure given in Algorithm 1, where N is the number of classes in the given data set. Next we compare given class $g(x_j)$ of the sample to its true class by

$$B(x_j) = \begin{cases} 1 & \text{if } g(x_j) = T(x_j) \\ 0 & \text{if } g(x_j) \neq T(x_j) \end{cases} \tag{4}$$

and we denote $T(x_j)$ as the true class from sample x_j. For the objective function of differential evolution algorithm we next calculate fitness function by using

$$cost = 1 - \frac{\sum_{j=1}^{m} B(x_j)}{m} \tag{5}$$

where m is the number of samples in training set and this cost value will be minimized by DE. To summarize DE is seeking a vector

$$\mathbf{v_{i,G}} = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_{N_1}}, \mathbf{p}\} \tag{6}$$

which minimizes our objective function given in (5) and $\mathbf{y_i}$, where $i \in \{1, 2, \ldots, N_1\}$ are possible ideal vector candidates and p is a parameter coming from the Minkowski distance in (3).

In this study, we used the Differential Evolution algorithm [9] to optimize all the ideal vectors and the p value. For this purpose, we split the data into the learning set *learn* and the testing set *test*. Division was carried out so that half of the data was used in the learning set and half in the testing set. This random division was repeated 30 times and average accuracies were computed. We used the data available in the learning set to find the optimal ideal vectors $\mathbf{y}_i$, and the data

Algorithm 1 Pseudo code for fitting the samples ideal vector index to a class index

```
while $max(g(x_j)) > N$ do
    if $g(x_j) > N$ then
        $g(x_j) = i - N$
    end if
end while
```

in the testing set *test* was applied for assessing the classification performance of the proposed classifier. In short, the procedure for our algorithm is as follows:

1. Divide data into learning set and testing set.
2. Create initial multiple ideal vectors (here we simply used random numbers).
3. Compute the distance between samples in the learning set and all of the ideal vectors.
4. Fit the samples ideal vector index to correspond classes by using Algorithm 1.
5. Classify samples according to their minimum distance and using label information to which class, which vector belongs.
6. Compute classification accuracy (number of correctly classified samples/total number of samples in learning set).
7. Compute the objective function value to be minimized as $cost = 1 - accuracy$.
8. Create new ideal vectors for the next population using selection, mutation and crossover operations of the differential evolution algorithm, and go to 3. until the stopping criteria is reached (e.g. maximum number of iterations reached).
9. Classify data in testing set according to the minimum distance between class vectors and samples.

Summarizing briefly, we first apply a global optimization algorithm, differential evolution, for determining the parameters of our classifier model optimally so that the classifying accuracy over the training data will be maximized. After that the test data will be classified with these parameters of the multiple vector classifier model. A sample belongs to the class that the very nearest class vector represents. Main procedure of how the multiple vectors within classes can be used and how they can be refitted is also given shortly in Algorithm 2.

3 TEST DATA SETS AND RESULTS

In all experiments, the test data was split in half; one half for testing and one half for training. This split was randomly repeated 30 times, based on which mean classification accuracies and variances were computed. To enable further assessment of the results, we repeated the classification results also for support vector machine (SVM) [6] classifier with a linear kernel function and back propagation neural

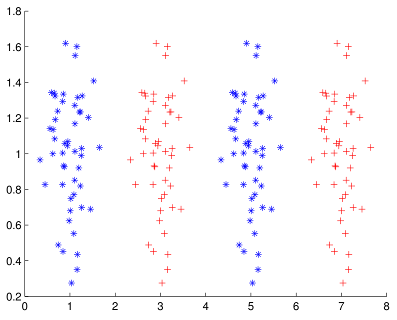
Algorithm 2 Pseudo code for classification process with optimal class vectors and parameters from DE

```
Require: $Data[1,\dots,m]$, $classvec1[1,\dots,T]$, $classvec2[1,\dots,T]$, $\dots$,
    $classvecN_1[1,\dots,T]$, $p$, $N$
    $center = [classvec1; classvec2; \dots, classvecN1]$
    Calculate distances for all the samples and all the ideal vectors:

    for $i = 1$ to $m$ do
        for $j = 1$ to $N_1$ do
            $d(i,j) = d(data(i,:), center(j,:))$
        end for
    end for
    for $i = 1$ to $m$ do
    Find the index of the ideal vector which has minimum distance to the sample

        $g(i) = find(d(i,:) == \min(d(i,:))$
    end for
    Refit ideal vector indexes to the class indexes

    while $\max(g(i))) > N$ do
        if g(i)>N then
            $g(i) = g(i) - N$
        end if
    end while
```

network (BPNN) [36] classifier for comparison. Our DE classifier was carried for $G_{max} = 1\,000$ iteration or until 100 % classification accuracy was reached.

In first experiment we decided to use test problem where data was splitted in four parts with two classes and from these four parts the two classes could not be accurately classified by using just distance based method but more was needed. In this case the multiple vectors for DE classifier are giving the answer to our problem. Data can be seen in Figure 1 a). In Table 1 one can see the results from this experiment. There results are also compared to original DE classifier, BPNN and SVM classifier. As can be seen from the results multiple vector DE (MVDE) classifier managed with the mean classification accuracy of 98.44 % which is far better than others. Original DE classifier managed with accuracy of 70.67 % and support vector machine SVM classifier managed only with 51.47 % accuracy. Back propagation neural network classifier did also quite well with 92.77 % but still got over 5 % lower classification accuracy than MVDE.

In the second experiment we tested MVDE classifier in the situation, where one class is inside the other class so basically we can say that one class is surrounding the other class. Data set created for this experiment can be seen in Figure 1 b). This experiment was already clearly more difficult than the first one and more vectors



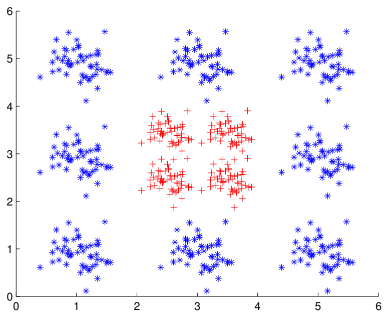


Figure 1. First two test data sets where data is not separable by using simply distance based method. Classes are marked with + or ∗ a) Two class classification problem is in four different areas b) Two classes are such that one is inside the other

Classifier	Accuracy	Variance
MVDE	98.44 %	0.037
DE_{orig}	70.67 %	0.25
BPNN	92.77 %	0.014
SVM	51.47 %	0.002

Table 1. Mean classification accuracies from first test case experiments

per class were needed than just two vectors per class as in the first experiment. Here best results were achieved with multiple vector DE when six vectors were used for one class. The mean classification accuracy of 99.09 % was achieved. Here BPNN performance was also very strong yielding the highest mean classification accuracy of 99.31 %. Original DE classifier and SVM managed both with 66.67 % classification accuracy.

Classifier	Accuracy	Variance
MV2DE	98.32 %	0.005
MV6DE	99.09 %	0.005
DE_{orig}	66.67 %	0.0012
BPNN	99.31 %	0.0003
SVM	66.67 %	0.0001

Table 2. Mean classification accuracies from second test case experiments. MV2DE means that we used two ideal vectors per class and MV6DE means we used six ideal vectors for each class.

In our third experiment we decided to test the classifier by creating data with xor-like data formulation, again making the data set such that it was not separable by simply using distance based method. Here we also decided to test our classifier by adding more such xor type squares to the data. Modifying the data in this manner also meant that we needed to add more vectors to the class to get more accurate results. Figures from the data sets used in this experiment can be seen in Figure 2 where we used 2×2, 4×4, 6×6 and 8×8 xor type data.

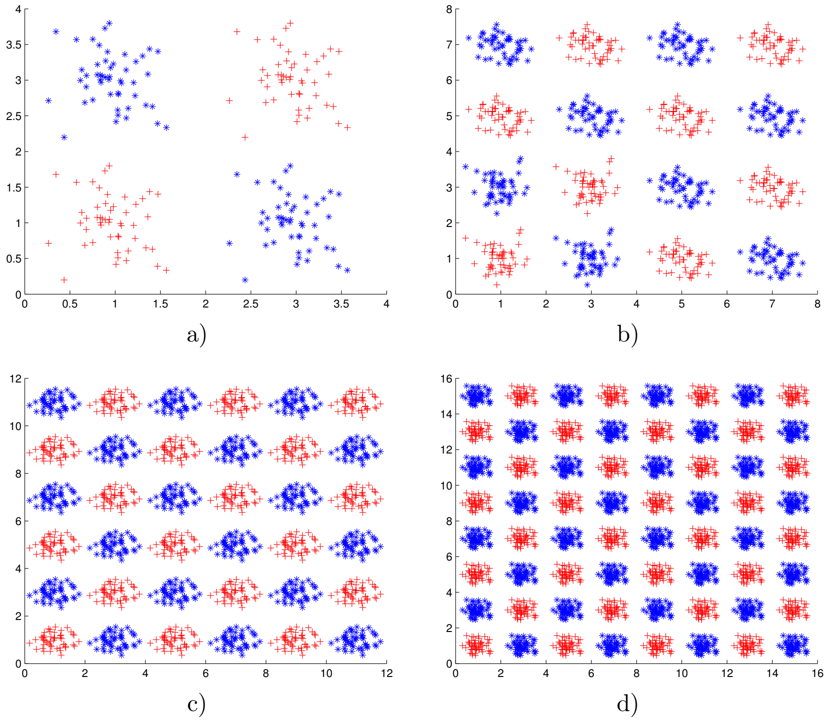


Figure 2. Xor type data where difficulty is increased by adding more squares in the data a) 2×2 XOR data b) 4×4 XOR data, c) 6×6 XOR data, d) a) 8×8 XOR data.

Classification results from this experiment can be seen in Table 3. As can be seen from the Table with 2×2 xor data both MVDE and BPNN managed quite well getting both over 98 % mean classification accuracy. Original DE and SVM on the other hand did not do so well but accuracies were around 70 %. When difficulty was increased by adding more squares to the test problem, mean classification accuracies started to decrease with all four classifiers. By increasing the number of squares we also needed to increase vectors in our MVDE. In 4×4 problem we used all together 16 vectors and in 8×8 we used 64 vectors now having 32 vectors per class. As can be seen from the Table 3, by far the best results were gained with MVDE where with 8×8 problem we had the mean classification accuracy of 84.3 %. Classification accuracy with original DE was only 57.5 % in 8×8 xor problem and BPNN and SVM did not anymore work any better than flipping the coin with this problem gaining roughly 50 % accuracy.

Data type	MVDE	DE_{orig}	BPNN	SVM
2×2	99.3	72.6	98.6	68.3
4×4	97.7	65.7	79.4	50.34
6×6	87.9	59.9	55.7	50.7
8×8	84.3	57.5	51.6	49.5

Table 3. Mean classification accuracies from xor-type test experiments

Our next testing case problem to which we decided to test our method is MONK's problem which is freely available in UCI machine learning data repository [35]. The MONK's problem was the basis of the first international comparison of learning algorithms (see [37]). One significant characteristic of this comparison is that it was performed by a collection of researchers, each of whom was an advocate of the technique they tested. This way one can say that the results are less biased than in comparisons performed by a single person advocating a specific learning method, and more accurately reflect the generalization behavior of the learning techniques as applied by knowledgeable users. In this artificial data set there are eight attributes and number of instances is 432. It is a binary classification problem.

Results with this data set and the comparison can be seen in Table 4. As can be seen, original DE classifier managed only with 82.01 % mean classification accuracy. Neural network classifier and SVM managed better with the mean accuracies 85.85 % and 92.81 %. Again, the best results were quite clearly gained with multiple vector DE classifier where now two vectors were used for each class. In this experiment MVDE classifier clearly showed its performance capabilities.

Classifier	Accuracy	Variance
DE_{orig}	82.01 %	0.0058
MVDE	99.84 %	0.0024
BPNN	85.85 %	0.0011
SVM	92.81 %	0.0001

Table 4. Mean classification accuracies from MONK's problem data set. Number of vectors used in MVDE per class is two class vectors for each class.

Next we decided to test our classifier with real world data set taken from a UCI-Repository of Machine Learning Database [35]. There we chose the Tic-Tac-Toe End game data set which clearly has problem setting which is not separable by simply using distance based method but a clear need for something else, in this case multiple vectors, is present. This is basically classification task problem on different configurations of tic-tac-toe game. This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). Data has nine different attributes and number of instances is 958. We tested our MVDE with several different number of multiple vectors. We managed to get best results by using four vectors per class.

There the mean classification accuracy was 98.03 % with variance 0.016. When we compare the classification results with BPNN and SVM we clearly see that MVDE gives over 20 % higher classification accuracy than either of these two. Also when comparing to the original DE classifier over 20 % higher classification accuracy was achieved using MVDE compared to original. From these results one a bit more general notion can also be seen. If we use too many vectors per class the results start to deteriorate. This can be seen more clearly in Table 6 where classification accuracies from both test data set and learn data set are listed. There one can see that when we have too many vectors involved, overlearning can become a problem. From learning set highest mean classification accuracy 98.71 % was gained using eight vectors per class but in testing data set mean accuracy reduced to 76.78 %. This is quite likely due to the fact that when there is no need for more vectors per class, this MVDE starts putting vectors near outliers of the learning data set and when tested with the testing data set such outliers do not exist and hence they are just deteriorating the mean classification accuracy in the test data set. The same observation was done with the other data sets. If we have too many vectors per class, results start to deteriorate and overlearning can become a problem.

Classifier	Accuracy	Variance
DE_{orig}	75.36 %	0.025
$MVDE_2$	82.72 %	0.154
$MVDE_4$	98.03 %	0.016
$MVDE_6$	98.00 %	0.003
$MVDE_8$	76.78 %	2.82
BPNN	76.85 %	0.0021
SVM	65.27 %	0.0001

Table 5. Mean classification accuracies from tic-tac-toe data set. Number of vectors used in MVDE per class is given in the subscript.

Classifier	Testing Data Set	Learning Data Set
DE_{orig}	75.36 %	79.60 %
$MVDE_2$	82.72 %	85.89 %
$MVDE_4$	98.03 %	97.96 %
$MVDE_6$	98.00 %	98.65 %
$MVDE_8$	76.78 %	98.71 %

Table 6. Mean classification accuracies from tic-tac-toe data set from both test data set and learning data set. Number of vectors used in MVDE per class is given in the subscript.

4 DISCUSSION

We have introduced multiple vector differential evolution (MVDE) based classifier where the key idea is that instead of having one vector per class to be optimized in class vector we can have several vectors per class. Need for this type of classifier became evident from the test classifiation problems which are not separable by simply using optimal class vector and suitable distance. MVDE offers a solution to such classification problems. Several test sets were given which are easy to understand and which clearly show the need for such classification methods. MVDE was also tested to real world classification problem and its performance was not only benchmarked with several different test problems but also compared to back propagation neural network (BPNN) and support vector machine (SVM) classifiers. BPNN is a classifier which can to some extent manage with this type of classification problems. But the experiments showed that MVDE managed usually with even higher classification accuracy, and when classifiers performance was tested with xor-type problem setting with increasing difficulty, we noticed that MVDE managed quite well even in larger problem settings where performance of BPNN quite quickly deteriorated. SVM is known as an accurate distance based classification algorithm but with this type of test problems it could not managed well, because problems were too complex for such algorithm where MVDE classifier, on the other hand, did perform well. One important observation from the experiments was that in order to get the best possible classification accuracy for these problems an optimal number of vectors were needed. If we have too little or too many vectors per class to a given problem, results quite quickly start to deteriorate. In case of too many vectors, overlearning can become a problem. One subject of the future research in this area is trying to optimize the correct number of vectors needed for the data sets because in many real world data sets this is not clear in advance.

Acknowledgments

The work of Jouni Lampinen was supported by the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 funded by the operational programme "Research and Development for Innovations" of the Structural Funds of the European Union and the Czech Republic Ministry of Education.

REFERENCES

[1] Quinlan, J. R.: Induction of Decision Trees. Machine Learning, Vol. 1, 1986, No. 1, pp. 81–106.

[2] Utgoff, P. E.: Incremental Induction of Decision Trees. Machine Learning, Vol. 4, 1989, No. 2, pp. 161–186.

[3] Clark, P.—Niblett, T.: The CN2 Induction Algorithm. Machine Learning, Vol. 3, 1989, No. 4, pp. 261–283.

[4] Pao, Y. H.: Adaptive Pattern Recognition and Neural Networks. Addison-Wesley Publishing Company, Inc., 1989.

[5] Bishop, C. M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford, 1995.

[6] Vapnik, V.: The Nature of Statistical Learning Theory. Springer-Verlag, 1995, ISBN 0-387-98780-0.

[7] Zhang, L.—Hu, X.: Word Combination Kernel for Text Classification with Support Vector Machines. Computing and Informatics, Vol. 32, 2013, No. 4, pp. 877–896.

[8] Hwang, B.-W.—Kwon, S.-J.—Lee, S.-W.: Facial Image Reconstruction from a Corrupted Image by Support Vector Data Description. Computing and Informatics, Vol. 32, 2013, No. 6, pp. 1212–1228.

[9] Price, K.—Storn, R.—Lampinen, J.: Differential Evolution – A Practical Approach to Global Optimization. Springer, 2005.

[10] Booker, L.: Improving the Performance of Generic Algorithms in Classifier Systems. In: Grefenstette, J. J. (Ed.): Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, July 1985, pp. 80–92.

[11] Holland, J. H.: Properties of the Bucket-Brigade Algorithm. In: Grefenstette, J. J. (Ed.): Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, July 1985, pp. 1–7.

[12] Holland, J. H.: Genetic Algorithms and Classifier Systems, Foundations and Future Directions. Proceedings of the 2nd International Conference on Genetic Algorithms, 1987, pp. 82–89.

[13] Holland, J. H.—Holyoak, K. J.—Nisbett R. E.—Thagard, P. R.: Classifier Systems, Q-Morphisms and Induction. In: Davis, L. (Ed.): Genetic Algorithms and Simulated Annealing. 1987, Chapter 9, pp. 116–128.

[14] Robertson, G.: Parallel Implementation of Genetic Algorithms in a Classifier System. In: Davis, L. (Ed.): Genetic Algorithms and Simulated Annealing. 1987, Chapter 10, pp. 129–140.

[15] Wilson, S. W.: Hierarchical Credit Allocation in a Classifier System. In: Davis, L. (Ed.): Genetic Algorithms and Simulated Annealing. 1987, Chapter 8, pp. 104–115.

[16] Fogarty, T. C.: Co-Evolving Co-Operative Populations of Rules in Learning Control Systems. In: Fogarty, T. C. (Ed.): Evolutionary Computing. AISB Workshop, Leeds, 1994, Selected Papers, Lecture Notes in Computer Science, Vol. 865, 1994.

[17] Vivekanandan, P.—Rajalakshmi, M.—Nedunchezhian, R.: An Intelligent Genetic Algorithm for Mining Classification Rules in Large Datasets. Computing and Informatics, Vol. 32, 2013, No. 1, pp. 1–22.

[18] Chauhan, N.—Ravi, V.—Chandra, D. K.: Differential Evolution Trained Wavelet Neural Networks: Application to Bankruptcy Prediction in Banks. Expert Systems with Applications, Vol. 36, 2009, No. 4, pp. 7659–7665.

[19] Su, H.—Yang, Y.—Zhao, L.: Classification Rule Discovery with DE/QDE Algorithm. Expert Systems with Applications, Vol. 37, 2010, No. 2, pp. 1216–1222.

[20] Khushaba, R. N.—Al-Ani, A.—Al-Jumaily, A.: Feature Subset Selection Using Differential Evolution and a Statistical Repair Mechanism. Expert Systems with Applications, Vol. 38, 2011, No. 9, pp. 11515–11526.

[21] Bastürk, A.—Günay, E.: Efficient Edge Detection in Digital Images Using a Cellular Neural Network Optimized by Differential Evolution Algorithm. Expert Systems with Applications, Vol. 36, 2009, No. 2, pp. 2645–2650.

[22] Liu, Y.—Sun, F.: A Fast Differential Evolution Algorithm Using k-Nearest Neighbour Predictor. Expert Systems with Applications, Vol. 38, 2011, No. 4, pp. 4254–4258.

[23] Sun, J.—Zhang, Q.—Tsang, E. P. K.: DE/EDA: A New Evolutionary Algorithm for Global Optimization. Information Sciences, Vol. 169, 2005, pp. 249–262.

[24] Maulik, U.—Saha, I.: Modified Differential Evolution Based Fuzzy Clustering for Pixel Classification in Remote Sensing Imagery. Pattern Recognition, Vol. 42, 2009, pp. 2135–2149.

[25] Zmudaa, M. A.—Rizkib, M. M.—Tamburinoc, L. A.: Hybrid Evolutionary Learning for Synthesizing Multi-Class Pattern Recognition Systems. Applied Soft Computing, Vol. 2, 2009, No. 4, pp. 269–282.

[26] Bandyopadhyay, S.—Saha, S.: GAPS, A Clustering Method Using a New Point Symmetry-Based Distance Measure Technique. Pattern Recognition, Vol. 40, 2007, No. 12, pp. 3430–3451.

[27] Das, S.—Abraham, A.—Konar, A.: Automatic Clustering Using an Improved Differential Evolution Algorithm. IEEE Transactions on Systems, Man and Cybernetics, Part A, Vol. 38, 2008, No. 1, pp. 218–237.

[28] Luukka, P.—Lampinen, J.: Differential Evolution Classifier in Noisy Settings and with Interacting Variables. Applied Soft Computing, Vol. 11, 2011, No. 1, pp. 891–899.

[29] Costa, M.—Palmisano, D.—Pasero, E.: Gradient Descent in Feed-Forward Neural Networks with Binary Neurons. Proceedings of the Neural Networks, 2000, Vol. 1, pp. 311–316.

[30] Liu, H.: On the Levenberg-Marquardt Training Method for Feed-Forward Neural Networks. 6th International Conference on Natural Computation, 2010, pp. 456–460.

[31] Storn, R.—Price, K. V.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Space. Journal of Global Optimization, Vol. 11, 1997, No. 4, pp. 341–359.

[32] Price, K. V.: An Introduction to Differential Evolution. In: Corne, D., Dorigo, M., Glover, F. (Eds.): New Ideas in Optimization. McGraw-Hill, UK, 1999, Chapter 6, pp. 79–108.

[33] Duda, R.—Hart, P.: Pattern Classification and Scene Analysis. John Wiley & Sons, 1973.

[34] Luukka, P.—Lampinen, J.: A Classification Method Based on Principal Component Analysis and Differential Evolution Algorithm Applied for Predition Diagnosis from Clinical EMR Heart Data Sets. In: Tenne, Y., Goh, C.-K. (Eds.): Computational Intelligence in Optimization, Applications and Implementations. Adaptation, Learning, and Optimization, Springer, Vol. 7, 2010, pp. 263–283.

[35] Newman, D. J.—Hettich, S.—Blake, C. L.—Merz, C. J.: UCI Repository of Machine Learning Databases. Department of Information and Computer Science, University of California, Irvine, CA, 1998. Available on: `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[36] Duin, R. P. W.—Juszczak, P.—Paclik, P.—Pekalska, E.—de Ridder, D.—Tax, D. M. J.—Verzakov, S.: PRTools4, A Matlab Toolbox for Pattern Recognition. August 2007.

[37] Thrun, S. B.—Bala, J.—Bloedorn, E.—Bratko, I.—Cestnik, B.—Cheng, J.—De Jong, K.—Dzeroski, S.—Fahlman, S. E.—Fisher, D.—Hamann, R.—Kaufman, K.—Keller, S.—Kononenko, I.—Kreuziger, J.—Michalski, R. S.—Mitchell, T.—Pachowicz, P.—Reich, Y.—Vafaie, H.—Van de Welde, W.—Wenzel, W.—Wnek, J.—Zhang, J.: The MONK's Problems – A Performance Comparison of Different Learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University, 1991.

Pasi Luukka received his M.Sc. degree in information technology in 1999 and his Dr.Sc. degree in applied mathematics in 2005 from University of Technology, Lappeenranta, Finland. He is currently Associate Professor of applied mathematics and strategic finance with the Lappeenranta University of Technology. His research is focused on soft computing methods in data analysis and decision making. His main research interests include classification, feature selection and projection methods, similarity and distance measures, fuzzy multi-criteria decision making methods, decision support systems, consensus.

Jouni Lampinen received his M.Sc. degree in economics in 1998 and his Dr.Sc. degree in economics in 2000 from the University of Vaasa, Finland. He is currently Professor of information technology at the University of Vaasa. He is also a senior researcher at the Technical University of Ostrava, Czech Republic. His research interests include intelligent systems and evolutionary computation.