Differential Evolution Classifier with Optimized Distance Measures for the Features in the Data Sets

David Koloseni[1], Jouni Lampinen[2,3], and Pasi Luukka[1]

[1] Laboratory of Applied Mathematics, Lappeenranta University of Technology, P.O. Box 20, FIN-53851 Lappeenranta, Finland
{david.koloseni,pasi.luukka}@lut.fi
[2] Department of Computer Science, University of Vaasa, P.O. Box 700, FI-65101 Vaasa, Finland
jouni.lampinen@uwasa.fi
[3] VSB-Technical University of Ostrava,17. listopadu 15, 70833 Ostrava-Poruba, Czech Republic

Abstract. In this paper we propose a further generalization of differential evolution based data classification method. The current work extends our earlier differential evolution based nearest prototype classifier that includes optimization of the applied distance measure for the particular data set at hand. Here we propose a further generalization of the approach so, that instead of optimizing only a single distance measure for the given data set, now multiple distance measures are optimized individually for each feature in the data set. Thereby, instead of applying a single distance measure for all data features, we determine optimal distance measures individually for each feature. After the optimal class prototype vectors and optimal distance measures for each feature has been first determined, together with the optimal parameters related with each distance measure, in actual classification phase we combine the individually measured distances from each feature to form an overall distance measure between the class prototype vectors and sample. Each sample is then classified to the class assigned with the nearest prototype vector using that overall distance measure. The proposed approach is demonstrated and initially evaluated with three different data sets.

1 Introduction

Differential evolution algorithm (DE) has lately gained increasing popularity in solving classification problems. The recent research in the field include *e.g.* bankruptcy prediction [1], classification rule discovery [2], nearest neighbor prototype search [3] and feature selection [4]. Since its introduction in 1995 DE have emerged among the most frequently applied evolutionary computing methods [5]. DE has also been used in many areas of pattern recognition, i.e. in remote sensing imagery [6] and hybrid evolutionary learning in pattern recognition systems [7], to mention a few examples.

The focus of this paper is in solving classification problems by extending the earlier DE based classifier [8],[9], [10] further on with an extension for optimizing the selection of applied distance measure individually for each feature in the data set to be classified. In our previous work [10] we generalized our original DE classifier version [8] by extending the optimization process to cover also the selection of the applied distance measure from a predefined pool of alternative distance measures. In [10], however, we applied the same distance measure for all features in the dataset, while the proposed approach are optimizing the distance measures individually for each feature of the classified data. The rationale behind this extension is that in classification, the data set to be classified is often in a form where each sample consists of several measurements, and it is not guaranteed that all the measured values from one particular sample obey the same optimal distance measure. It is clear that an optimal distance measure for a feature may not be optimal for another feature in the same data set. This is the underlying motivation to further generalize our earlier method so, that instead of optimizing a single vector based distance measure for all features, we concentrate on the problem at the feature level, and optimize the selection of applied distance measure individually for each particular feature.

Examples on situations where a considerably improved classification accuracy have been obtained by applying some other distance measure than a simple euclidean metric have been reported in several articles *i.e.* [11], [12], [13]. However, typically in these types of studies one has simply tested with a few different distance measures for classifying the data set at hand. So far none of them have concentrated on selecting distance measures optimally at feature level, but on data set level instead.

Thereby, in case of classifying a dataset containing T features using the proposed approach, we need to determine T different optimal distance measures. In addition we need to determine also the possible parameters related to each distance measure and the class prototype vectors representing each class. All these should be optimized so, that the classification accuracy over the current dataset will be maximized. We apply DE algorithm for solving the resulted global optimization problem in order to determine all mentioned values optimally. In particular, if the distance measure applied to a particular feature has any free parameters, also their values need to be optimized as well.

After the optimal class prototype vectors and distance measures with their related parameters have been determined by DE algorithm, the actual classification by applying the determined values takes place. After we have first computed the individual distances between a sample and the class prototype vectors individually for each feature, then we compute the overall distance value by normalizing the individual distances first, and then simply calculate the sum of all feature wisely computed and normalized distances. Finally, each sample is to be classified into the class represented by the nearest class prototype vector that is providing the lowest overall distance value.

2 Differential Evolution Based Classifier with Optimized Distances for the Features in the Data Sets

2.1 Differential Evolution Based Classification

The DE algorithm [15], [5] was introduced by Storn and Price in 1995 and it belongs to the family of Evolutionary Algorithms (EAs). As a typical EA, DE starts with a randomly generated initial population of candidate solutions for the optimization problem to be solved 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 to be computed provides an appropriate stopping condition. Other control parameters for DE are the crossover control parameter CR, the mutation factor F, and the population size NP.

In each generation G, DE goes through each D dimensional decision vector $v_{i,G}$ of the population and creates the corresponding trial vector $u_{i,G}$ as follows in the most common DE version, DE/rand/1/bin [16]:

```
$r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$, (randomly selected,
  except mutually different and different from $i$)
$j_{rand} = \text{floor}(rand_i[0,1) \cdot D) + 1$
for$(j = 1; j \leq D; j = j + 1)$
{
    if$(rand_j[0,1) < CR \vee j = j_{rand})$
        $u_{j,i,G} = v_{j,r_3,G} + F \cdot (v_{j,r_1,G} - v_{j,r_2,G})$
    else
        $u_{j,i,G} = v_{j,i,G}$
}
```

In this DE version, NP must be at least four and it remains fixed along CR and F during the whole execution of the algorithm. Parameter $CR \in [0,1]$, which controls the crossover operation, represents the probability that an element for the trial vector is chosen from a linear combination of three randomly chosen vectors and not from the old vector $v_{i,G}$. The condition "$j = j_{rand}$" is to make sure 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 $(0, 1+]$[1].

After the mutation and crossover operations, the trial vector $u_{i,G}$ is compared to the old vector $v_{i,G}$. If the trial vector has an equal or better objective value, then it replaces the old vector in the next generation. This can be presented as follows (in this paper minimization of objectives is assumed) [16]:

$$v_{i,G+1} = \begin{cases} u_{i,G} & \text{if} \quad f(u_{i,G}) \leq f(v_{i,G}) \\ v_{i,G} & \text{otherwise} \end{cases} .$$

DE is an elitist method since the best population member is always preserved and the average objective value of the population will never get worse. As the objective function, f, to be minimized we applied the number of incorrectly classified learning set samples. Each population member, $v_{i,G}$, as well as each new trial solution, $u_{i,G}$, contains the class vectors for all classes and the power value p. In other words, DE is seeking the vector $(y(1), ..., y(T), p)$ that minimizes the objective function f. After the optimization process the final solution, defining the optimized classifier, is the best member of the last generation's, G_{max}, population, the individual $v_{i,G_{max}}$. The best individual is the one providing the lowest objective function value and therefore the best classification performance for the learning set. For control parameter values see [8], [9]. Next into the actual classification. We suppose that T is the number of different kinds of features that we can measure from objects. The key idea is to determine for each class the ideal vector $\mathbf{y}_i$, $\mathbf{y}_i = (y_{i1}, \ldots, y_{iT})$ that represents class i as well as possible. Later on we call

[1] Notation means that the upper limit is about 1 but not strictly defined.

these vectors as class vectors. When these class vectors have been determined we have to make the decision to which class the sample $\mathbf{x}$ belongs to according to some criteria. This can be done e.g. by computing the distances d_i between the class vectors and the sample which we want to classify. For computing the distance usual way is to use Minkowsky metric, $d(\mathbf{x},\mathbf{y}) = \left(\sum_{j=1}^{T} |x_j - y_j|^p \right)^{1/p}$. After we have the distances between the samples and class vectors then we can make our classification decision according to the shortest distance. For $\mathbf{x},\mathbf{y} \in R^n$. We decide that $\mathbf{x} \in C_m$ if

$$d\langle \mathbf{x}, \mathbf{y}_m \rangle = \min_{i=1,\dots,N} d\langle \mathbf{x}, \mathbf{y}_i \rangle \tag{1}$$

In short the procedure for our algorithm is as follows:

1. Divide data into learning set and testing set
2. Create trial vectors to be optimized which consists of classes and parameter p, $v_{i,G}$
3. Divide $v_{i,G}$ into class vectors and parameter p.
4. Compute distance between samples in the learning set and class vectors
5. Classify samples according to their minimum distance by using (1)
6. Compute classification accuracy (accuracy = no. of correctly classified samples/total number of all samples in learning set)
7. Compute the fitness value for objective function using $f = 1 - accuracy$
8. Create new pool of vectors $v_{i,G+1}$ for the next population using selection, mutation and crossover operations of differential evolution algorithm, and goto 3. until stopping criteria is reached. (For example maximum number of iterations reached or 100% accuracy reached)
9. Divide optimal vector $v_{i,G_{max}}$ into class vectors and parameter p.
10. Repeat steps 4, 5 and 6, but now with optimal class vectors, p parameter and samples in the testing set.

For more thorough explanation we refer to [8]. The proposed extension to the earlier DE classifier will be described in detail next.

2.2 The Proposed Extension for Optimizing Distance Measures Individually for Each Feature in the Data Set

Basically the vector to be optimized consists now of following components:

$$v_{i,G} = \{\{class1, class2 \cdots classN\}, \{switch\}, \{parameters\}\}$$

where $\{class1, class2 \cdots classN\}$ are the class vectors which are to be optimized for the current data set, $\{switch\}$ is an Integer valued parameter pointing the particular distance measure to be applied from choices d_1 to d_8 (see Table 1). In the proposed approach an individual value of $\{switch\}$ is assigned for each data feature. In other words our $\{switch\}$ is T dimensional vector consisting of integer number within $[1,8]$. Since DE algorithm operates internally with floating point representations and $\{switch\}$ is a vector of Integer values, the corresponding adaptations are needed. Therefore we use in our

Algorithm 1. Pseudo code for classification process with optimal parameters from DE.

```
Require: $Data[1,\dots,T]$, $classvec1[1,\dots,T]$, $classvec2[1,\dots,T]$,...,$classvecN[1,\dots,T]$, $switch$, $p1[1,\dots,T]$, $p2[1,\dots,T]$,
  $p3[1,\dots,T]$,$p4[1,\dots,T]$ $center$=[$classvec1$;$classvec2$;...$classvecN$]
  for $j = 1$ to $T$ do
    for $i = 1$ to $N$ do
      if $switch(j) == 1$ then
        $d(:,i,j) = dist1(data, repmat(center(i,j),T,1), p1(j))$
      else if $switch(j) == 2$ then
        $d(:,i,j) = dist2(data, repmat(center(i,j),T,1), p2(j))$
      else if $switch(j) == 3$ then
        $d(:,i,j) = dist3(data, repmat(center(i,j),T,1), p3(j))$
      else if $switch(j) == 4$ then
        $d(:,i,j) = dist4(data, repmat(center(i,j),T,1))$
      else if $switch(j) == 5$ then
        $d(:,i,j) = dist5(data, repmat(center(i,j),T,1))$
      else if $switch(j) == 6$ then
        $d(:,i,j) = dist6(data, repmat(center(i,j),T,1), p4(j))$
      else if $switch(j) == 7$ then
        $d(:,i,j) = dist7(data, repmat(center(i,j),T,1))$
      else
        $d(:,i,j) = dist8(data, repmat(center(i,j),T,1))$
      end if
    end for
  end for
  $D = scale(d)$
  for $i = 1$ to $T$ do
    $d_{total}(:,i) = sum(D(:,:,i),2)$;
  end for
  for $i = 1$ to $length(d_{total})$ do
    $class(:,i) = find(d_{total}(i,j) == min(d_{total}(i,:)))$;
  end for
```

optimization real numbers that are boundary constrained $[0.5, \quad 8.499]$, and all DE operations are performed in real space. Only when we are applying the $\{switch\}$ vector to point the actual distance measures to be applied, we first round the values to the nearest integer. In addition we have the $\{parameters\}$ which is a vector of possible parameters from the distance measures. In this case $\{parameters = \{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}\}\}$ where all $\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}\}$ are again vectors of T dimensions (i.e. $\mathbf{p_1} = \{\mathbf{p_{1,1}}, \mathbf{p_{1,2}}, \cdots, \mathbf{p_{1,T}}\}$).

Each component of vector $\{switch\}$ is referring to a distance measure in the pool given in Table 1. A goal of the optimization process is to select the distance measures optimally from this pool individually for each feature in the current data set. A collection of applicable distance measure is provided in[17], from where also the measures in Table 1 have been taken.

The pseudocode Algorithm1 is describing the actual classification process after the optimal class prototype vectors, distance measures and the possible free parameters of each distance measure have been first determined by DE algorithm.

Next we discuss a bit more in detail the main modifications done to our previous method. After we had created the pool of distances we needed to find a way to optimize the selection of distance from the pool of possible choices to the data set at hand. For this we used the *switch* operator which was needed to optimize. For the eight distances in the pool now we had to add vector of parameters of length of T to be optimized in order to select the suitable distance. By optimizing the integer numbers within $[1,8]$ we performed the needed optimal selection. In addition to this, as can be noticed from the pool of distances and from the pseudo code there are different parameter values with different distances which needs to be optimized as well. For this we created additional parameters to be optimized for each of the different parameters in pool of distances. This part of the vector we call $\{parameters\}$ which is possible parameters from the distance measures. In this case $\{parameters = \{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}\}\}$. At this point we are facing the situation where we can have i.e. situation where optimal distance for feature one can be d_1 and also i.e. optimal distance for feature two can be d_1 and clearly the optimal parameter value can be different. For this reason we made $\{\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}\}$ also vectors of length T. This results in having several optimized parameter values which are not used, making the optimization task more challenging, but also makes it possible that different features with same optimal distance measure can have different optimal parameter. After the vector $v_{i,G}$ is divided in its corresponding parts we can calculate the distances between the samples and the class vectors. This results in a vector of distances for one sample and one class vector. This vector is then normalized properly and after that we aggregate this vector simply by computing the sum of normalized distances. This is now stored in d_{total} in the pseudo code. This process is repeated for all the classes and all the samples. This way we end up having a distance matrix consisting of samples and distances to each particular class. After we have created this distance matrix the selection of to which class the particular sample belongs is made according to minimum distance.

Table 1. Distance measures in a pool of distances

$d_1(x,y) = (\lvert x-y\rvert^{p_1}); p_1 \in [1,\infty)$	$d_2(x,y) = \lvert x-y\rvert^{p_2}/max\{\lvert x\rvert, \lvert y\rvert\}; p_2 \in [1,\infty)$
$d_3(x,y) = \lvert x-y\rvert^{p_3}/min\{1+\lvert x\rvert, \lvert y\rvert\}; p_3 \in [1,\infty)$	$d_4(x,y) = \lvert x-y\rvert/max\{1+\lvert x\rvert, \lvert y\rvert\};$
$d_5(x,y) = \lvert x-y\rvert/[1+\lvert x\rvert+\lvert y\rvert];$	$d_6(x,y) = \lvert x/[1+\lvert x\rvert] - y/[1+\lvert y\rvert]\rvert$
$d_7(x,y) = p_4(x-y)^2/(x+y); p_4 \in (0,\infty)$	$d_8(x,y) = \lvert x-y\rvert/(1+\lvert x\rvert)(1+\lvert y\rvert)$

3 Classification Experiments and Comparisons of the Results

The data sets for experimentation with the proposed approach were taken from UCI machine learning data repository [14]. Chosen data sets were all such where optimal distance measure was not euclidean distance. The data sets were subjected to computation of 1000 generations ($G_{max} = 1000$) of DE algorithm and the data was divided 30 times into random splits of testing sets and learning sets, based on which mean accuracies and variances were then computed. The fundamental properties of the data sets are summarized in Table 2.

Table 2. Properties of the data sets

Name	Nb of classes	Nb of features	Nb of instances
Horse-colic	2	11	368
Hypothyroid	2	25	3772
Balance scale	3	5	625

The proposed differential evolution classifier with optimal distance measures for each feature was tested with Hypothyroid, Horsecolic and Balance scale data set. The mean classification accuracies were recorded as well as the corresponding optimal distance measure that was found optimal in each individual experiment. In each case 30 repetitions were performed dividing the data randomly into learning and testing tests. The applied crossvalidation technique was two fold crossvalidation, where samples are divided randomly into folds 30 times and required statistics was calculated from that basis. Folds were normalized to achieve efficient and precise numerical computations. Results from the experiments are reported in Table 3.

As can be observed from the Table 3 for Hypothyroid data set the mean accuracy of 99.57% was reached and using 99% confidence interval (by $\mu \pm t_{1-\alpha} S_\mu / \sqrt{n}$) accuracy was 99.57 ± 0.63. With Balance scale data we observed the accuracy of 91.30 ± 0.12, and with Horsecolic data the classification accuracy of 83.35 ± 2.23 were observed.

To enable initial comparisons with some of the most frequently applied classifiers and with the previous DE classifier, we calculated the corresponding classification results also by using k-NN classifier, Back Propagation Neural Network (BPNN) and DE classifier [8]. The results and their comparisons are provided in Tables 4.

In Table 4, results from Horsecolic, hypothyroid and Balancescale data are compared between four classifiers. For Hypothyroid data, the proposed method performed significantly better than k-NN. Also in comparison with BPNN the proposed method gave significantly higher mean accuracy in 0.999 confidence interval. The original DE classifier performed here slightly better than the proposed method, but the observed difference was not found to be statistically significant.

Table 3. Classification results for the three data sets using $N = 30$ and $G_{max} = 1000$. Mean classification accuracies, variances and optimal distance found are reported in columns 2 to 6. TS is referring to test set and LS to the learning set.

Data	Mean (TS)	Variance (TS)	Mean (LS)	Variance (LS)
Horsecolic	83.35	24.67	88.59	2.36
Balance scale	91.30	0.075	92.22	0.028
Hypothyroid	99.57	1.98	99.98	0.00092

Table 4. Comparison of the results from the proposed method to other classifiers with Horse-colic data, Hypothyroid data and Balancescale data.

	Horsecolic		Hypothyroid		Balancescale	
Method	Mean accuracy	Variance	Mean accuracy	Variance	Mean accuracy	Variance
KNN	68.53	5.44	98.37	0.006	88.06	1.42
BPNN	80.85	19.27	97.29	0.018	87.90	2.22
DE classifier	71.76	107.78	99.95	0.0061	88.66	4.71
Proposed method	83.35	24.67	99.57	1.98	91.30	0.075

The results from the comparisons with the Horsecolic data set the proposed method achieved a clearly higher mean classification accuracy when compared with KNN, BPNN and DE classifiers. Improvement with mean accuracy compared to original DE classifier was significant and remarkably high, more than 10%.

In Table 4 also the results with balance scale data set are reported. Here the proposed method significantly and rather clearly outperformed the other compared classifiers by reaching the mean accuracy of 91.30%.

4 Discussion

In this paper we proposed an extension for the differential evolution based nearest prototype classifier where the selection of the applied distance measure is optimized individually for each feature of the classified data set. Earlier a single distance measure was applied for all data features, and thereby we were able to optimize the selection of the distance measure in the data set level only. The proposed generalization extends the optimization of distance measures to the feature level.

To demonstrate the proposed approach, and to enable a preliminary evaluation of it, we carried out experimentation with three different data sets. In two cases the classification accuracy of the proposed method outperformed all compared classifiers significantly. In the remaining case no statistically significant difference to the earlier DE classifier version were observed, despite the difference to the other compared classifiers were again significant and rather clear. The results are suggesting that the proposed generalization is advantageous from the classification accuracy point of view. However, this conclusion should be interpreted as a preliminary one due to limited number of data sets investigated so far.

An important aspect is, that the proposed approach is not limited to DE classifiers only, and can be applied generally in connection with any other similar type of classification method that is based on global optimization. However, this is assuming that an effective enough global optimizer like differential evolution algorithm is applied to solve the resulting optimization problem.

Despite the current results are promising, they should be considered preliminary and need to be further confirmed by applying a broader selection of data sets. That includes into our further research plans. Concerning the possibilities for further developments of the proposed approach, so far we have applied a simple summation of feature wisely computed distances to calculate the final overall distance measure. However, in future also a suitable aggregation method can be used for the purpose. This is also one of our future directions for further investigations of this method.

Acknowledgment. The work of Jouni Lampinen was supported by the IT4Innovations Center of Excellence project. CZ.1.05/1.1.00/02.0070 funded by the Structural Funds of EU and Czech Republic ministry of education.

References

1. Chauhan, N., Ravi, V., Chandra, D.K.: Differential evolution trained wavelet neural networks: Application to bankruptcy prediction in banks. Expert Systems with Applications 36(4), 7659–7665 (2009)
2. Su, H., Yang, Y., Zhao, L.: Classification rule discovery with DE/QDE algorithm. Expert Systems with Applications 37(2), 1216–1222 (2010)
3. Triguero, I., Garcia, S., Herrera, F.: Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. Pattern Recognition 44, 901–916 (2011)
4. Khushaba, R.N., Al-Ani, A., Al-Jumaily, A.: Feature subset selection using differential evolution and a statistical repair mechanism. Expert Systems with Applications 38, 11515–11526 (2011)
5. Price, K., Storn, R., Lampinen, J.: Differential Evolution - A Practical Approach to Global Optimization. Springer (2005)
6. Ujjwal, M., Saha, I.: Modified differential evolution based fuzzy clustering for pixel classification in remote sensing imagery. Pattern Recognition 42, 2135–2149 (2009)
7. Zmudaa, M.A., Rizkib, M.M., Tamburinoc, L.A.: Hybrid evolutionary learning for synthesizing multi-class pattern recognition systems. Applied Soft Computing 2(4), 269–282 (2009)
8. Luukka, P., Lampinen, J.: Differential Evolution Classifier in Noisy Settings and with Interacting Variables. Applied Soft Computing 11, 891–899 (2011)
9. Luukka, P., Lampinen, J.: A Classification method based on principal component analysis differential evolution algorithm applied for predition diagnosis from clinical EMR heart data sets. In: Computational Intelligence in Optimization: Applications and Implementations. Springer (2010)
10. Koloseni, D., Lampinen, J., Luukka, P.: Optimized Distance Metrics for Differential Evolution based Nearest Prototype Classifier. Accepted to Expert Systems With Applications
11. Shahid, R., Bertazzon, S., Knudtson, M.L., Ghali, W.A.: Comparison of distance measures in spatial analytical modeling for health service planning. BMC Health Services Research 9, 200 (2009)
12. Yu, J., Yin, J., Zhang, J.: Comparison of Distance Measures in Evolutionary Time Series Segmentation. In: Third International Conference on Natural Computation, ICNC 2007, pp. 456–460 (2007)
13. Jenicka, S., Suruliandi, A.: Empirical evaluation of distance measures for supervised classification of remotely sensed image with Modified Multivariate Local Binary Pattern. In: International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT), pp. 762–767 (2011)
14. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. University of California, Department of Information and Computer Science, CA (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
15. Storn, R., Price, K.V.: Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Space. Journal of Global Optimization 11(4), 341–359 (1997)
16. Price, K.V.: New Ideas in Optimization. An Introduction to Differential Evolution, ch. 6, pp. 79–108. McGraw-Hill, London (1999)
17. Bandemer, H., Näther, W.: Fuzzy Data Analysis. Kluwer Academic Publishers (1992)