THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

*PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,*
*Volume 10, Number 2/2009, pp. 000–000*

# NEAREST NEIGHBOR CLASSIFICATION WITH IMPROVED WEIGHTED DISSIMILARITY MEASURE

Lucian Vasile BOICULESE, Gabriel DIMITRIU, Mihaela MOSCALU

"Gr. T. Popa" University of Medicine and Pharmacy, Department of Medical Informatics and Biostatistics,
16 University Street, Iasi , Romania
Corresponding author: Lucian Vasile BOICULESE, e-mail: vlboiculese@mail.umfiasi.ro

The usefulness and the efficiency of the $k^{th}$ nearest neighbor classification procedure are well known. A less sophisticated method consists in using only the first nearby prototype. This means k=1 and it is the method applied in this paper. One way to get a proper result is to use weighted dissimilarities implemented with a distance function of the prototype space. To improve the classification accuracy and to minimize the number of parameters, functions that shape the relation of the dissimilarity with the prototypes, the features or the class belonging are proposed. Benchmark tests are also presented and the worth results encourage us to continue developing this new proposed weighted model.

*Key words:* Pattern classification; Nearest neighbor rule; Weighted dissimilarity; Gradient descent.

## 1. INTRODUCTION

The k nearest neighbor algorithm (KNN) is a supervised learning (instance-based learning) method used in applications for pattern recognition [1, 2]. The simplest version is the nearest neighbor (NN) algorithm that exploits only the first nearby prototype for classification (it can also be used for prediction and estimation).

The best performance is achieved by computing the Bayes risk, that is why the comparisons are made on this basis. Theoretically speaking, in order to get this best risk value, the number of prototypes should tend to infinity. Practically this is not reasonable; therefore, there is an interest in developing functions that ascertain the dissimilitude between elements.

Metric functions can measure the distances between prototypes and are suitable for defining dissimilarities. There are many ways to express a metric function. Let us mention some of them that are mostly applied: the well known Euclidean ($L_2$ norm), the Mahalanobis and the Minkowski distance (the well known City Block –Manhattan is a Minkowski distance of order 1).

With regards to these possibilities of measuring differences between elements, one may try to optimize the system by choosing the best matched metric function. The performance of a KNN system is assessed by the percentage of appropriate classifications ratio. The outcome depends on the training set and also on the measure of distance.

Improvements of the system performance were made by changing the distance formula [3, 4, 5, 8]. Some of the changes have the effect of not satisfying the symmetry condition on the metric function [8]. This is not a cumbersome result. By these modifications the system will manage to achieve appropriate outcome. It is important to have a suitable measure of discrepancy and also to have a scientific consistency in defining and proofing results.

The Euclidean distance with parameters that express the feature weights are presented and tested in some recent papers [3, 5, 8]. The weighted distances can model the dependency between the classes, dimensions, prototypes and are embedded in the measured dissimilarities. For the fully local weighted distance the dissimilarity function depends on any position of the prototypes. If the weight is just a parameter (named $w$), it follows to have infinity of $w$. To cope with this problem we propose to use a function that expresses the relationship between the prototypes dimensions, class or position as an alternative of the $w$

weight. The tested benchmarks reveal results that confirm the operational efficiency and robustness of the proposed algorithm.

## 2. METHOD

By means of the NN algorithm the classification of a new record is made by comparing it to the most similar record from the training set [1, 2, 7]. The weights that should be optimized are inserted in the distance function that computes the dissimilarities between elements. In addition, an error function that reveals the ratio of misclassifications is defined. To train the system, the gradient descent method may be utilized with the well known drawbacks [6].

Assume that the objects of interest are taken from a representation space denoted by $S$. The dimension of $S$ is $m$, so we can write $S = R^m$. To train the system let set a collection of prototypes (training patterns) with the associated classes:

$T = \{(x^1, c_1), (x^2, c_2), ..., (x^n, c_n)\}$ , where $x^i \in S$, $c_i = \{1,2,...C\}$, $c_i$ is an integer that represents the class of $x^i$. A dissimilarity function may be a distance or metric: $d : S \times S \to R$. Now, if this metric is the Euclidean or $L_2$ norm, it should be defined as (from now on $x$ will denote an element from $T$ and $y$ is a new element that should be classified):

$$d(y, x) = \sqrt{\sum_{j=1}^{m} \left(y_j - x_j\right)^2} \, . \tag{1}$$

To have a weighted distance, one may insert proper weights denoted by: $w_j$, if they depend on the space dimension; $w_{jx}$, if they also depend on the training prototype $x$; $w_{jxy}$ if they depend on both inputs $(x, y)$ and the space dimension as well ; and $w_{jc}$ if they depend on the prototype class and dimension ([5]).

We define the dissimilarity function with the formula:

$$d(y, x) = \sqrt{\sum_{j=1}^{m} w_{jxy}{}^2 \cdot \left(y_j - x_j\right)^2} \, . \tag{2}$$

Note that in the last situation, $w_{jxy}$ defines a dependent weight on the feature space and also on the position of $x$ and $y$ (it can be simplified to $w_j$ or to $w_{jx}$ or $w_{jc}$). So it is fully local, because it depends on the exact position of inputs vectors. Accordingly, the number of parameters $w_{jxy}$ that should be optimized is infinite.

It is important to have a dissimilarity function that takes into account the information of both inputs, but at the same time, the function should have a finite number of weights.

To preserve this requirement we have first proposed a linear dependency function as an alternative of the simple weight. First, we have substituted $w_{jxy}$ with $w_{jc}$ (class and feature dependence). The weight can be expressed as $a_{jc} \cdot x_j + b_{jc} \cdot \left(y_j - x_j\right)$. So, it also depends on $x$ and $y$ positions. Therefore, the dissimilarity function is replaced with:

$$d(y, x) = \sqrt{\sum_{j=1}^{m} \left(a_{jc} \cdot x_j + b_{jc} \cdot \left(y_j - x_j\right)\right)^2 \cdot \left(y_j - x_j\right)^2} \, , \tag{3}$$

where $c$ is the class of prototype $x$. Here, we have a term that depends on the $x$ feature ($a_{jc} \cdot x_j$) and a term that depends on the difference between $y$ and $x$ feature: $b_{jc} \cdot \left(y_j - x_j\right)$. This was meant to express that near the $x$ prototype the weight is not constant and it depends on the prototype position and class.

On the other hand, it is worth mentioning that $d(y,x)$ is an asymmetric function with the proposed weights. Therefore, the value $d(x,y)$ is different from $d(y,x)$. The most recent formula is not a metric. We have to take into account that the trained prototype should always be in the *x* position in formula (3).

Another way to express the fact that a prototype may have a proper influence on the classification task is to use nonlinear function. Thus, we have tested an exponential model inspired by the Gaussian bell shaped function. It is important to have a weight as small as we approach to the prototype studied. As a result this function should be:

$$w_{jc} = 1 - \exp\left(\frac{-(y_j - x_j)^2}{2\sigma_{jc}}\right), \tag{4}$$

where $c$ is the class number, $j$ is the space feature or dimension. This function has the minimum value for $y_j = x_j$, so that the dissimilarity is zero if the testing and training processes overlap. This seems to match our common sense. The parameter $\sigma_{jc}$ should be optimized to get proper results. It represents the influence of a prototype in the close vicinity. The new dissimilarity function is:

$$d(y,x) = \sqrt{\sum_{\substack{j=1,\\ c=class(x)}}^{m} \left(1 - \exp\left(\frac{-(y_j - x_j)^2}{2\sigma_{jc}}\right)\right)^2 \cdot \left(y_j - x_j\right)^2}\ . \tag{5}$$

## 3. ERROR FUNCTION

The dissimilarity previously defined will be used to compute the error function. In paper [5] Paredes and Vidal proposed a method named leaving-one-out (LOO) that estimates the probability of classification error. This is explained in what follows.

For a prototype $x$ from the training set, there exist two other prototypes namely $x^{=}$ and $x^{\neq}$.

$x^{=}$ is the nearest neighbor closest to $x$ that has the same class.

$x^{\neq}$ is the nearest neighbor closest to $x$ that has a different class.

The $x$ element is well classified if the dissimilarity $d(x,x^{=})$ is less than the value of $d(x,x^{\neq})$.

For each element belonging to the training set (LOO), the ratio $d(x,x^{=}) \big/ d(x,x^{\neq})$ will show an accurate classification, if it is less than 1. To compute the global error, the step function centered at 1 should be applied according to the following formula:

$$J_T(a,b) = \frac{1}{n} \cdot \sum_{x \in T} step\left(\frac{d(x,x^{=})}{d(x,x^{\neq})}\right). \tag{6}$$

For the exponential model, the error should depend on sigma: $J_T(\sigma)$. The minimization of $J_T(a,b)$ by the gradient descent technique requires that the error function should be differentiable. To accomplish this condition and at the same time, accepting a reasonable approximation, the step function will be replaced with the sigmoid function denoted by:

$$S_\beta(z) = \frac{1}{1 + \exp(\beta \cdot (1 - z))}. \tag{7}$$

Now, we have the differentiable form of the new error function:

$$J_T(a,b) \approx \frac{1}{n} \cdot \sum_{x \in T} S_\beta\left(\frac{d(x,x^{=})}{d(x,x^{\neq})}\right). \tag{8}$$

## 4. WEIGHTS ADJUSTMENT

The optimization algorithm was made by the gradient descent method. This iterative procedure updates the weights at each step, with a small quantity, proportionally to the negative direction of the gradient. For a dissimilarity defined by formula (3), the adjustment should be:

$$a_{jc}^{(t+1)} = a_{jc}^{(t)} - \eta \cdot \left( \frac{\partial J_T(a,b)}{\partial a_{jc}} \right)^{(t)}, \tag{9}$$

where $\eta$ denotes the learning rate and $t$ represents the time variable.

It is well-known that the learning rate (also named step size) is a trade-off between the speed of adaptation and accuracy at the final weight value. Also, a drawback of the gradient technique is the possible fluctuation around local minima. To cope with this, a variable training step was implemented [6].

A frequently used procedure for the variation of the training step is to start with a large value and to decrease it in time. This is known as the rate scheduling or annealing and here indicate some practical formulae:

$$\eta(t+1) = \eta(t) - \alpha,\ \eta(t+1) = \frac{\eta_0}{1 + \frac{t}{n_0}},\ \eta(t+1) = \frac{\eta(t)}{\log 10(10+t)}, \tag{10}$$

where $\eta_0$ is the initial value, $\alpha$ is a small constant, $n_0$ is an iteration count, and $t$ is the time.

The gradient with respect to the weight $a_{jc}$ is the partial derivative $\left( \frac{\partial J_T(a,b)}{\partial a_{jc}} \right)$, so that the training operation takes the form:

$$a_{jc}^{(t+1)} = a_{jc}^{(t)} - \eta \frac{1}{n} \cdot \left[ \sum_{\substack{x \in T \\ class(x^=)=c}} S_\beta^{'}(r(x)) \cdot r(x) \cdot R_1\left(x_j, x_j^{=}\right) - \sum_{\substack{x \in T \\ class(x^{\neq})=c}} S_\beta^{'}(r(x)) \cdot r(x) \cdot R_1\left(x_j, x_j^{\neq}\right) \right]. \tag{11}$$

Here, we denoted by $S_\beta^{'}$, the derivative of the sigmoid function $S_\beta^{'}(r(x))$ with respect to $r(x)$:

$$R_1\left(x_j, y_j\right) = \frac{\left[a_{jc} \cdot y_j + b_{jc} \cdot \left(x_j - y_j\right)\right] \cdot \left(x_j - y_j\right)^2 \cdot y_j}{d(x,y)^2}, \tag{12}$$

$$r(x) = \frac{d\left(x, x^{=}\right)}{d\left(x, x^{\neq}\right)}. \tag{13}$$

Similarly, we have found the gradient with respect to the weight $b_i$ and accordingly, the adjustment will take the form:

$$b_{jc}^{(t+1)} = b_{jc}^{(t)} - \eta \frac{1}{n} \cdot \left[ \sum_{\substack{x \in T \\ class(x^=)=c}} S_\beta^{'}(r(x)) \cdot r(x) \cdot R_2\left(x_j, x_j^{=}\right) - \sum_{\substack{x \in T \\ class(x^{\neq})=c}} S_\beta^{'}(r(x)) \cdot r(x) \cdot R_2\left(x_j, x_j^{\neq}\right) \right], \tag{14}$$

where $R_2$ stands for the ratio:

$$R_2\left(x_j, y_j\right) = \frac{\left[a_{jc} \cdot y_j + b_{jc} \cdot \left(x_j - y_j\right)\right] \cdot \left(x_j - y_j\right)^3}{d(x,y)^2}. \tag{15}$$

For the exponential weights, the adjustment is calculated by:

$$\sigma_{jc}^{(t+1)} = \sigma_{jc}^{(t)} - \eta \frac{1}{n} \cdot \left[ \sum_{\substack{x \in T \\ class(x^{=})=c}} - S_{\beta}^{'}(r(x)) \cdot r(x) \cdot R_3\left(x_j, x_j^{=}\right) + \sum_{\substack{x \in T \\ class(x^{\neq})=c}} S_{\beta}^{'}(r(x)) \cdot r(x) \cdot R_3\left(x_j, x_j^{\neq}\right) \right], \tag{16}$$

$$R_3\left(x_j, y_j\right) = \frac{\left(x_j - y_j\right)^4 \cdot \exp\left( \frac{-\left(x_j - y_j\right)^2}{2 \cdot \sigma_{jc}^2} \right)}{2 \cdot d(x, y)^2 \cdot \sigma_{jc}^3}. \tag{17}$$

We can see from equations (11), (14) and (16) that the adjustments on the weights also depend of the sigmoid function derivative. In fact, as we have already mentioned, the sigmoid profile represents an approximation of the step function. It has a parameter ($\beta$) that defines the slope. It is knowen that as $\beta$ increases the sigmoid is closer to the step function. As a remark, one may decide to enlarge this parameter in order to get a better approximation. On the other hand if $\beta$ increases the profile of the sigmoid derivative becomes thinner.

The argument of this derivative from equations (11), (14) and (16) is based on the ratio defined in formula (13). As a consequence, a large value of $\beta$ will adjust only the prototypes which have the ratio close to 1 (see formula (13)).

It is acceptable to start with a small value of $\beta$ and, during the training process to increase it properly. Therefore, we may have larger training steps at the beginning and thus, more prototypes will contribute in the training scheme. It is worth mentioning that this parameter value has a major influence in the training process and in optimization of the system.

## 5. TESTING DATA

The proposed weighted dissimilarity measures were tested with a set of artificial records produced on computer. We have generated a set of benchmark test data with three classes. The bivariate normal distribution was used for three classes with the following mean and covariance matrices:

Class 1: $\mu = \begin{bmatrix} 2 \\ 2.5 \end{bmatrix}$, $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$; Class 2: $\mu = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, $\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$;

Class 3: $\mu = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$.

We depicted below a graphical representation of the generated data in order to have an image of the overlapping distributions.

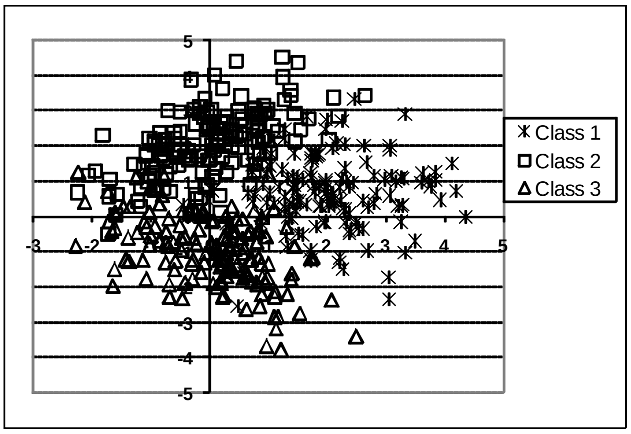

Figure 1. Three classes with normal distribution bivariate data.

The second data used for testing the proposed system, was taken from UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets.html). The Breast Cancer Wisconsin was tested for classification [9]. There are two possible classes taken under consideration: benign and malignant. The total number of prototypes is 683 ($n = 683$), after deleting the incomplete data. The space dimension is 9 ($n = 9$) and is defined by integer data (can be approximated by continuous type as they have 10 possible values).

To estimate the error rate, a method similar to the $K$ -fold cross-validation was applied, with $k = 5$. Thus, we have divided the records in 5 blocks using randomized selection. Taking into account that the number of parameters depends on the number of the training prototypes, and the optimization time may be increased exponential with increasing the prototype number, we have chosen to use one of the samples for training and the other 4 for testing (in order to decrease the training time). Each of the 5 samples was exploited for training, while the other 4 were used for testing. Therefore, the training-testing cycle was repeated 5 times.

We have tested 6 types of systems, according to the next table:

Table 1.

| **No.** | **Dissimilarity type** | **No. of weights** | **Legend** |
|---|---|---|---|
| 1 | Linear - feature dependence | $m\ (w_j)$ | LF |
| 2 | Linear - prototype and feature dependence | $n \times m\ (w_{ic})$ | LPF |
| 3 | Linear - class and feature dependence | $c \times m\ (w_{jc})$ | LCF |
| 4 | Exponential - feature dependence | $m\ (w_j)$ | EF |
| 5 | Exponential - prototype and feature dependence | $n \times m\ (w_{ic})$ | EPF |
| 6 | Exponential - class and feature dependence | $c \times m\ (w_{jc})$ | ECF |

## 6. RESULTS

The system was trained (synthetic data) with three sets of data having 50, 100 and 150 prototypes per class. As a stop criterion for the algorithm, a number of 150 steps was set up.

The optimized system was tested with a number of 9000 prototypes (3000 for each class) and the results were assessed with the classification ratio. The learning rate ($\eta$) was decreased in time, in order to multiply the chances of finding a better solution. The value of the $\beta$ coefficient in the sigmoid function was also altered during training, to fit with the optimization requirement (the range of values is between 1 and 9).

The error classification ratio has not always a decreasing evolution during training. This is a drawback of the gradient descent technique. The next figure represents the error variation during the 100 training steps for two examples (synthetic data).

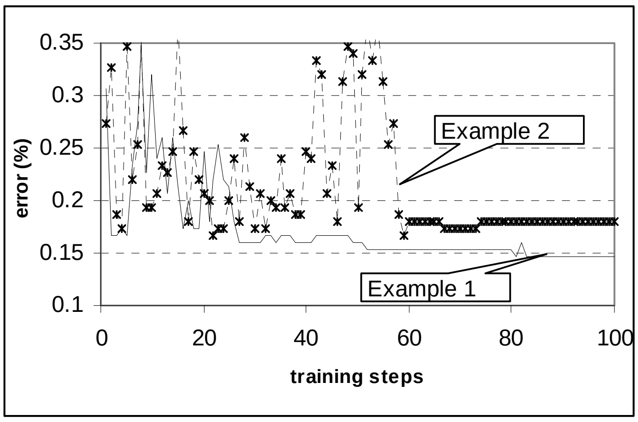

Figure 2. Error evolution during training.

Finally from the picture above, the minimum classification error is reached for the Example 1 with continuous line. This is not always true. In the first 30 training steps the fluctuation is observed (Example 1). It could continue to the end step optimization. That is why we have decided to adopt the optimized parameters, by choosing the minimum error of the entire training steps.

The best results were obtained with values between 6 and 9 for the $\beta$ sigmoid coefficient. These values are properly chosen for this example of the bivariate normal distribution.

There were not observed big differences between the results obtained with the training sets with 50, 100 and even 150 prototypes for each of the classes under study. The time needed for training has increased exponentially with the increasing volume sample.

We have generated randomly 10 training sets of 50 prototypes per class. To get a consistent comparison between the classifiers, we have chosen to compare the results by using the same sets of training-testing for all the weighted dissimilarity types. The baseline for comparison was the nearest neighbor algorithm.

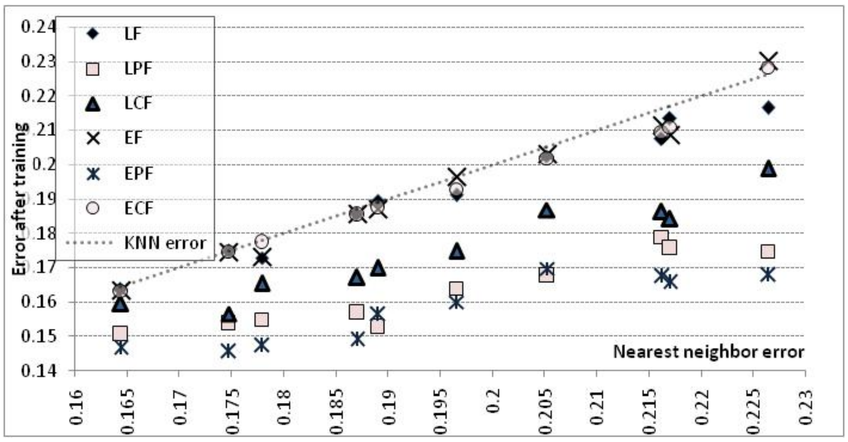

Figure 3. Error comparison for the 6 systems (10 points in the X axis).

In Figure 3 we plotted on the X axis the nearest neighbor error and on the Y axis, the optimum of error obtained for each of the 6th systems in the 10 study points. The dot line is obtained by charting the same values of nearest neighbor error on both axes. Thus, everything that is under the dotted line represents a better classification. The best performance is obtained by the EPF weighted model (mean error: 0.158), followed by the LPF (mean error: 0.163) and then by LCF (mean error: 0.175; the KNN mean error was 0.195).

To evaluate the results the statistical $t$ (Student) test was applied to get the significance value in comparison with the base line defined by the nearest neighbor classification. The differences are statistically attested ( $p < 5\%$ ) for EPF ( $p = 0.00016$ ), LPF ( $p = 0.00066$ ) and LCF ( $p = 0.0181$ ) thus these methods are better than the NN standard classification technique.

For the second data set (Breast Cancer Wisconsin), we have repeated many times the cycle of training and testing. For each selection the optimization procedure was run for 10 times (each cycle had 150 training steps) in order to get an appropriate value of the error rate classification. Because of the small data number we have applied a similar method to the $K$ -fold cross-validation procedure, described in a section above. In our experiments $k$ was set to 5 and thus we have proceeded in estimating the error. The next table (Table 2) shows the classification error for the selections made.

Table 2. Mean error and significance values for the 5 randomly subgroup selections.

| | KNN | LF | LPF | LCF | EF | EPF | ECF |
|---|---|---|---|---|---|---|---|
| Mean of classification ratio error | 0.043554 | 0.031513 | 0.030014 | 0.039165 | 0.039858 | 0.030012 | 0.042055 |
| p (significance) | | 0.0531 | **0.0356** | 0.3887 | 0.5096 | **0.0459** | 0.8068 |

The average values are less than the KNN error and the best performances are obtained with the EPF followed by the LPF method (the values are very close). For the statistical comparisons, the *p* value was 0.0356 for LPF and 0.0459 for EPF, meaning that the differences between LPF and KNN, and EPF and KNN respectively, are significant. These two methods have been found the best ones from the six procedures proposed for investigation.

There were a lot of researches in this area using artificial neural networks and the results ranges on the error classification rate between 6 and 2 percent [10,11,12]. Out performance of 3% is close to the best one presented in the literature.

To be more specific, we have made a comparison with the results (1.9% misclassification ratio) presented in the paper [12], Hussein Abbass, An evolutionary artificial neural networks approach for breast cancer diagnosis, Artificial Intelligence in Medicine, 2002.

Therefore we have used the same dataset as the Breast Cancer Wisconsin (683 patterns with 9 attributes). The first 400 instances are chosen for training whiles the remaining of 283 for testing. We have run the training for 150 steps and repeated for 5 times in order to start with new weighted parameters value. Thus, we have increased the chances in finding the global optimum solution. Finally the performances obtained with the LPF and EPF methods are the same and the error rate was 1.4134 % of misclassifications (the KNN classification has an error of 2.1201%). This is our best result that is less than 1.9% of misclassification.

We consider our results as more than satisfactory and comparing them to the others, it is obvious that they represent significant improvements for our proposed methods.

## 7. CONCLUSIONS

The functionality of the proposed systems was proved by the set of tests that were applied with synthetic and real data.

The new weighted dissimilarities (all the six methods) with linear and exponential dependency, manages to lower the error classification ratio, compared to the classic Euclidean distance (KNN classification). The best results were obtained with the prototype and feature dissimilarity paradigm (linear and exponential).

Comparing our results with the others from the Wisconsin Breast Cancer dataset from UCI Machine Learning Repository we have managed to get similar results or even a better classification ratio.

## ACKNOWLEDGEMENT

The authors would like to express their gratitude to all those who created the Breast Cancer Wisconsin data set [http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)] from UCI Machine Learning Repository.

## REFERENCES

1. DUDA, R., HART, P., STORK, D., *Pattern Classification*, New York, John Wiley & Sons, Inc., pp. 174-191, 2000.
2. LAROSE, D., *Discovering Knowledge in Data*, Hoboken, New Jersey, John Wiley & Sons, Inc., pp. 90-106, 2005.
3. PAREDES, R., VIDAL, E., *Learning prototypes and distances (LPD). A prototype reduction technique based on neighbor error minimization.* Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, **3**, pp. 442-445, 2004.
4. PAREDES, R., VIDAL, E., *Weighting prototypes. A new editing approach*, The XVth International Conference on Pattern Recognition,15th ICPR 2000, **2**, pp. 25-28, 2000.
5. PAREDES, R., VIDAL, E., *Learning weighted metrics to minimize nearest-neighbor classification error*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **28**, 7, pp. 1100–1110, 2006.
6. PRINCIPE, J., EULIANO, N., LEFEBVRE, W., *Neural and Adaptive Systems*, New York, John Wiley & Sons, Inc., pp. 173-222, 2000.
7. NONG, Y., *The Handbook of Data Mining*, Mahwah, New Jersey, London, Lawrence Erlbaum Associates, pp. 174-182, 2003.
8. WANG, J., NESKOVIC, P., COOPER, L., *Improving nearest neighbor rule with a simple adaptive distance measure*, Pattern Recognition letters, 28: 217-213, 2007.

9. WOLBERG, W., NICK STREET, W.,MANGASARIAN, O., UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)], Madison University of Wisconsin, 1993.
10. FOGEL, D., WASSON, E., BOUGHTON, E., *Evolving neural networks for detecting breast cancer*, Cancer letters, **96**, 1, pp. 49–53, 1995.
11. ABBASS, H., SARKER, R., NEWTON, C., *A pareto differential evolution approach to vector optimisation problems*, IEEE Congress on Evolutionary Computation, **2**, pp. 971–978, 2001.
12. ABBASS, H., *An evolutionary artificial neural networks approach for breast cancer diagnosis*, Artificial Intelligence in Medicine, **25**, 3, pp. 265-281, 2002.

*Received February 23, 2009*