Розроблено та досліджено універсальний класифікатор на прикладі класифікатора для діагностики захворювань, одержаний шляхом поєднання можливостей мережі з радіальною базисною функцією (РБФ-мережі) на основі функції Гауса та Дерева розв'язків CART. При його проектуванні необхідно визначити кількість РБФ-нейронів та значень параметрів цих нейронів (центру, дисперсії). Для цього запропоновано метод, який дозволяє розбити простір ознак на відносно однорідні області у вигляді гіперпаралелепіпедів, кожний з яких асоційовано з одним із РБФ-нейронів

Ключові слова: універсальний класифікатор, нейронна мережа, РБФ-мережа, Дерево розв'язків CART, підтримка прийняття рішень

Разработан и исследован универсальный классификатор на примере классификатора для диагностики заболеваний, полученный путем объединения возможностей сети с радиальной базисной функцией (РБФ-сети) на основе функции Гаусса и Дерева решений CART. При его проектировании необходимо определить количество РБФ-нейронов и значений параметров этих нейронов (центра и дисперсии). Для этого предложен метод, который позволяет разбить пространство признаков на относительно однородные области в виде гиперпараллелепипедов, каждый из которых ассоциирован с одним из РБФ-нейронов

Ключевые слова: универсальный классификатор, нейронная сеть, РБФ-сеть, Дерево решений CART, поддержка принятия решений

UDC 004.8
DOI: 10.15587/1729-4061.2017.108976

DESIGN OF THE UNIVERSAL CLASSIFIER AS A RBF NETWORK BASED ON THE CART SOLUTION TREE

L. Dobrovska
PhD, Associate Professor
Department of Biomedical Cybernetics
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»
Peremohy ave., 37, Kyiv, Ukraine, 03056
E-mail: luci.dln17@gmail.com
I. Dobrovska
UkrSibbank
Andriyvska str., 2/12, Kyiv, Ukraine, 04070
E-mail: dobrira@gmail.com

1. Introduction

Nowadays, there are many algorithms that implement Solution Trees. In the late 1970s and early 1980s, J. Ross Quinnan, a machine learning researcher, developed an ID3 algorithm for the Solution Tree construction (Iterative Dichotomiser - iterative partitioning). Later he proposed a C4.5 algorithm (based on ID3), which became a benchmark for the new learning algorithms. In 1984, a group of statisticians L. Breiman, J. Friedman, R. Olshen and C. Stone published a book called "Classification and Regression Trees (CART)".

The ID3 and CART algorithms were developed independently at about the same time, they build Solution Trees by studying the tuple data. The ID3, C4.5 and CART algorithms implement an approach when Trees are constructed recursively top-down, using the following measures of attribute selection: Gain information increment, Gain Ratio and Gini coefficient. In practice, these measures lead to the fairly good results for multi-value attributes:

1. The C4.5 algorithm with the Gain Ratio is used in an unbalanced partition (when one class is much smaller than the other).

2. The CART algorithm with Gini coefficient is able to define a multidimensional partition based on the linear combination of attributes. This partition is a structural form of the attribute when new attributes are formed based on the existing ones.

In recent years, the researchers have recognized neural networks (NN) as an instrument for solving many problems associated with biomedicine and healthcare. Among the fields of the NN using in the healthcare system are the following: biomedical signals processing, diseases diagnostics and medical systems assistance in the decision-making support.

The neural networks are able to study the relationship between the input-output mapping on a given sample of data without any prior knowledge or assumptions about statistical data distribution. This ability of data learning without any prior knowledge makes the NN suitable for solving the practical problems of classification and regression. In many biomedical applications, the problems of classification and regression take an important place. In addition, the NN are inherently non-linear, which makes them more practical for the precise modeling of complex data objects (or structures).

The neural networks are used in many real-world problems, including biomedicine, in order to surpass statistical classifiers and multiple regression methods during data analysis. Due to their ability to generalize invisible data, they are also suitable for the processing of data with outliers, as well as for the elimination (filling) of missing data and/or noisy data. The neural networks are also used together with other methods to combine the strengths and benefits of both methods.

The problem of developing universal classifiers of biomedical data in general and diagnostic data, in particular those that characterize the presence of a large number of parameters, inaccuracy and uncertainty, is relevant.

Many studies are aimed at developing methods for these data analysis, among them there are methods based on the network with radial basis function (RBF network) [1–4]. The RBF network is one of the popular tools for solving the function approximation (or data classification) problem.

Let us consider the problem of developing a universal classifier of diseases diagnostics. This classifier has the form of RBF network and combines various methods of processing the information about the same object of research (classifier in the form of CART solution tree and classifier in the form of RBF network based on the Gaussian function).

2. Literature review and problem statement

The ability of the RBF network to classify is influenced by the number of RBF neurons and the parameters of these neurons. There are various methods to define these parameters, among them are the following:

– the method that forms an RBF network using orthogonal basis functions [2];

– the method that determines the radius of RBF network taking into account the distribution of the sample data [5];

– the method that uses the C4.5 solution tree for the initial initialization of RBF network, – during the analysis of data on diabetes, the obtained accuracy of the classification was at the level of 74.8 % [6], but it is not clear from the paper whether this accuracy refers to the learning set or the testing set;

– the method that includes a data preparation step based on the selection of the corresponding sample data (the excessive sample items are removed using the threshold value, improving the accuracy of the classification); this approach helps to reduce the number of clusters and the number of RBF neuron centres; the learning efficiency is improved due to the use of hierarchical clustering method to reduce the number of clusters formed at each step of RBF neurons determination [7];

– the method that generates an RBF network using the partial least squares (PLS) method and the genetic algorithm (PLS-GA-RBF) [8].

It should be noted that the sources in which the binary classifier in the form of RBF network was investigated by combining the capabilities of RBF network based on the Gaussian function and the CART solution tree were not found. In some researches, CART classification trees show better results compared to the C4.5.

3. The aim and objectives of the study

The aim of the paper is to develop a universal binary classifier using the example of diseases diagnostics classifier in the form of RBF network by combining the capabilities of RBF network based on the Gaussian function and CART solution tree. The examples and capabilities of this binary classifier for diabetes diagnostics are considered.

To achieve this aim, the following objectives were identified:

– to develop a method for development of RBF network based on the CART solution tree, which allows splitting the space of examples into relatively homogeneous domains in the form of (hyper)parallelepipeds, each of which is associated with one of the RBF neurons;

– to check the efficiency of the method on different databases;

– to prove the possibility of using such NN in the healthcare system for the diseases diagnostics and the assistance to medical systems (or devices) in decision-making support.

4. The justification of the procedure of using the CART solution tree during RBF network development

Let us consider how the CART solution tree could be used for RBF network development. RBF network is depicted in Fig. 1. Here the activation function of the output neurons is linear, the activation function of the i-th neuron of the hidden layer has the following form

$$\varphi_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\boldsymbol{\sigma}_i^2}\right), \tag{1}$$

where $\boldsymbol{\mu}_i=[\mu_{i1},..., \mu_{in}]^T$ – centre and $\boldsymbol{\sigma}_i^2$ – dispersion of the i-th RBF neuron. In order to decide what class mark should be assigned to the input x, at first, the network is mapping x into the M-dimensional vector $\boldsymbol{\varphi}(\mathbf{x})=[\varphi_1(\mathbf{x}) \ldots \varphi_M(\mathbf{x})]^T$.

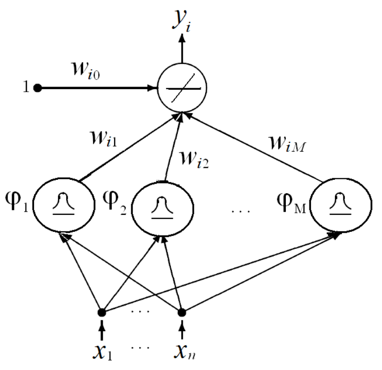

Fig. 1. RBF network with one output neuron

The value obtained in the output of the i-th output neuron is calculated as

$$y_i = \sum_{i=0}^{M} w_{ij}\varphi_j(\mathbf{x}),\ i=1, ..., m;$$

where w_{ij} – weight coefficient of the connection between the j-th hidden neuron and the i-th output neuron (w_{i0} is connected to the fixed $\varphi_0=1$). Generally, an example from the learning (or testing) set is marked with the i-th class if it is determined as:

$$y_i = \max_k\{y_1,\ ...,\ y_m\}.$$

The behavior of RBF network depends on the following parameters: the number of hidden layer neurons, the values of the Gaussian functions parameters (centres and dispersions) and the values of the output layer weight coefficients. The method of the output layer weight coefficients determination is relatively simple – for example, the network could be taught on the basis of a pseudo-inverse matrix [1].

During the RBF network development, a designer needs to know:

– how to determine the RBF neurons parameters and how many neurons are needed for the classifier to achieve an acceptable classification accuracy not only on the training examples, but also on the test data;

– how to solve the problem of difference in attributes scaling;

– how to define attributes that are not directly related to the classification problem (the classification accuracy could be improved if the classifier takes into account only important information in each part of the features space).

Typically, different dispersion values are used for each attribute. In [6], it is proposed to set the value of σ_{ik}^2 equal to the value which is proportional to the distance (along the k-th attribute) between the centre μ_{ik} of the i-th Gaussian function and the centre of its closest neighbor among other neurons.

Let μ_{ik} be the k-th coordinate of the centre $\boldsymbol{\mu}_i$, σ_{ik}^2 – dispersion of the i-th Gaussian function along the k-th attribute. Then, in order to calculate the output of the i-th hidden layer neuron in domains with n attributes, function (1) could be expressed as:

$$\varphi_i(\mathbf{x}) = \prod_{k=1}^{n} \exp\left(-\frac{\|x_k - \mu_{ik}\|^2}{2\sigma_{ik}^2}\right).$$

Obviously, it is a hang-the-expense procedure in terms of computations.

CART Solution Tree (Classification and Regression Tree) [10]. Among the various models of classifiers, classifiers based on the Solution Tree are known. The Solution Tree is a popular method of data analysis, which is based on learning from examples and the formation of an appropriate hierarchical structure. The Solution Tree divides the input space (known as the attribute space) of a data set into mutually exclusive domains, each of which is assigned a name. The decision-making mechanism is transparent, since the tree structure could easily explain how a decision is made.

The Inductive Solution Tree is a tree structure trained on the marked classes from the training set examples in the form of tuples. Each internal (nonfinite) node in this structure depicts the test value of the attribute from the training example, each branch depicts an example of model learning, each leaf (or terminal node) depicts a class mark (model output). The root node is the highest peak of the Tree. The Solution Tree can handle high dimensional data.

The training and classification stages based on the induction Solution Tree are run fast enough. The Solution Tree, which is used to solve classification problems, is called the *Classification Tree*, and each terminal node contains a mark indicating the predicted class of the given vector (example) of the attributes. In order to construct a Classification Tree, it is necessary to have a measurement error $E(t)$ which quantitatively describes the performance of node t during the splitting of data (examples) from different classes. The Classification Tree error is often called the impurity function of a given node. This error reaches its minimum value, such as zero, if all data belong to one class, and maximum, if the data are uniformly distributed among all possible classes.

In general, classifiers based on Solution Trees have acceptable precision. However, the successful use of these Trees depends on the available data. The study of various methods of forming a Solution Tree based on a data lies outside this material. Most of the existing algorithms use the recursive procedure which determines the criterion of the learning set division into two subsets so that the homogeneity of these subsets is maximized at each step. There are various software packages that perform this task.

Usually, the Solution Tree has to be reduced. During the reduction process, some of its sub-trees are either removed from the Tree or replaced by the smaller ones. In the multidimensional domains, one branch of the Solution Tree involves only a part of the attributes.

Problem statement (*diagnostic data classification*). Let the learning sample be given in the form of input-target data pairs: $\{\mathbf{x}^1, t^1\}, ..., \{\mathbf{x}^Q, t^Q\}$, that are generated by the function $t^i = f(\mathbf{x}^i)$, $i=1, ..., Q$, where $\mathbf{x}^i = [x_1^i \ \dots \ x_n^i]^T$ – the input vector with the elements $x_j^i \in \Re$; t^i – the desired response. The function $f(\times)$ is assumed to be unknown, but the set of its realizations is given:

$$T = \left\{(x_1^i, ..., x_n^i, t^i),\ i = 1, 2, ..., Q; n > 1\right\}.$$

Build an RBF network to determine the function $F(\mathbf{w}, \mathbf{x}^i)$ which approximates the function $f(\mathrm{x})$ by transforming the input signal into the output and satisfies the following condition

$$\frac{1}{Q}\sum_{i=1}^{Q}\left|F(\mathbf{w}, \mathbf{x}^i) - t^i\right| < \varepsilon,$$

where ε – some positive value which is called discrepancy.

Based on the training sample, the CART Decision Tree, which generates the function c: $\Re^n \to L$, where L – the set of class marks, is formed. For example, for a binary classifier $L=\{0, 1\}$. If the Tree's input is a vector $\mathbf{x}=\mathbf{x}^i=[x_1 \dots x_n]^T$, $i=1,..., Q$, then its output $c(\mathbf{x})$ is equal to "1", if the class mark is greater or equal to the value "0.5", and "0", if the class mark is less than "0.5".

The initialization of RBF network (function $F(\mathbf{w}, \mathbf{x}^i)$) was done using the CART Decision Tree. The weights of the output layer are determined on the basis of a pseudo-inverse rule.

5. The mechanism of Gaussian function determination using hyperparallelepipeds based on the CART Decision Tree generator

To solve the three research problems described above, a method that is based on the idea of binding each neuron with some relatively homogeneous space domain is used. To create homogeneous domains in the form of hyperparallelepipeds, the method uses the Solution Tree generator. The method that solves the problem of RBF networks parameters determination is described.

5. 1. The RBF network initialization using CART Decision Tree

There are methods of inductive Solution Trees development available to determine almost homogeneous domains [9]. It is shown that the Solution Tree based on one attribute testing determines a set of homogeneous hyperparallelepipeds, which could be transformed into the RBF network. An example of Solution Tree (Fig. 2) and its corresponding two-dimensional space $\Re^2$ of the researched objects attributes, which is split into domains in the form of rectangles, is depicted in Fig. 3. Each Tree branch consists of a set of one attribute binary tests and ends with a leaf that contains a class mark (only the following classes are considered: C_1="+", C_2="–").

To define the class that contains the attribute vector $\mathbf{x}$, the classifier checks $\mathbf{x}$ using this test starting from the root. The result of each test decides whether to continue the analysis on the left or right side. When **x** reaches the leaf, the tree assigns a class mark associated with this leaf. For this example, the square with sides (0, 10) is divided into rectangles (hyperparallelepipeds for *n* attributes), each of which contains a point depicting the neuron (Fig. 4) and marks the Gaussian function centre location.

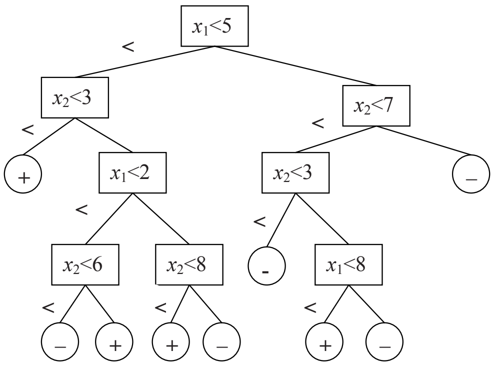

Fig. 2. The Solution Tree that divides the two-dimensional space into rectangles

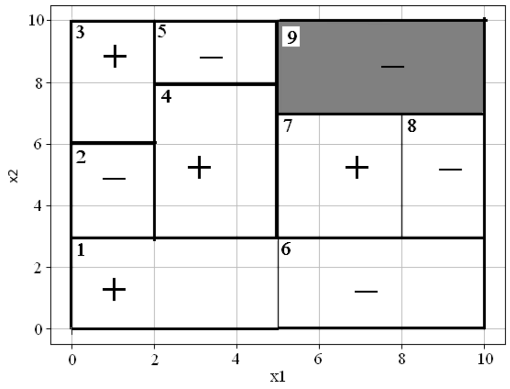

Fig. 3. Rectangles that correspond to the Solution Tree from Fig. 2

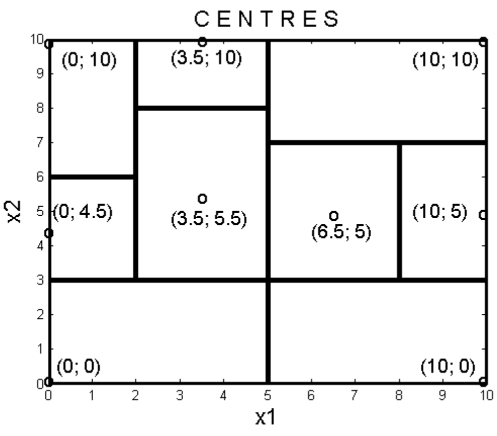

Fig. 4. The Gaussian function centres location

The neuron output is maximal for vectors located in the centre point vicinity. Its value decreases with the increasing distance to this point.

At the edge of the rectangle, the output is equal to some predetermined value *a*, which is the same for all neurons. Some rectangles are surrounded (located inside) by the others (Fig. 3). For such rectangles, the centre points are located in their geometric centres, and for other rectangles located "on the verge", a different approach is used.

The Gaussian functions values that correspond to the x_2 attribute from Fig. 3 are pictured in Fig. 5 (the *max* and *min* points are the maximum and minimum values of the x_2 attribute observed on the learning set). The Gaussian functions reach their maximum at the maximum distance to the domain boundary. Depending on whether the domain is within the space, this maximum distance is located either in the geometric centre of the rectangle, or on its edge.

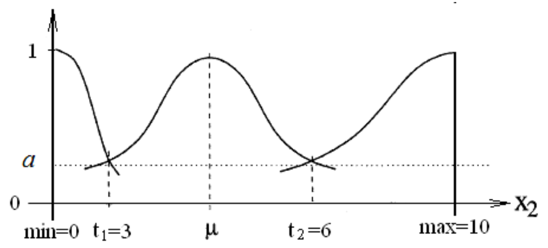

Fig. 5. The Gaussian functions that correspond to the attribute x_2 from Fig. 3 (functions reach the minimum value *a* on the edge of two rectangles)

Let us describe the algorithm that allows splitting the space of examples into relatively homogeneous domains in the form of hyperparallelepipeds, each of which is associated with one of the RBF neurons:

Step 1. The determination of the Gaussian functions centres μ *coordinates*

For each hyperparallelepiped defined based on the Solution Tree, we determine the Gaussian function centre μ location using the following rules:

1. If one side of the hyperparallelepiped is located on the space boundary, then μ is placed in the geometric centre of this side.

2. If two or more sides of the hyperparallelepiped are located on the space boundary, then μ is placed in the vertex formed by these sides.

3. If the hyperparallelepiped is bordered by other hyperparallelepipeds, then μ is placed in its geometric centre.

"Space boundary" is defined as the maximum or minimum value of a given attribute in the learning set.

Step 2. The determination of dispersion

The dispersion determines how quickly the value of the function $\varphi(x)$ decreases with increasing of the distance between x and μ. The values of this parameter depend on the length of the *k* -th side of hyperparallelepiped.

Let I_{ik} be the length of the *k*-th dimension of the *i*-th hyperparallelepiped (in Fig. 5 this value for the Gaussian function centre in the point μ is equal to $I_{ik}=t_2-t_1$). We introduce the parameter γ: $\gamma^2=I_{ik}^2/\sigma_{ik}^2$, which has the same value for all *k* attributes and all *i* neurons. Thus, the relationship between I_{ik}^2 and σ_{ik}^2 is constant within all domains. The output of the *i*-th neuron is calculated using the formula obtained by the replacement of $\sigma_{ik}^2=I_{ik}^2/\gamma^2$ in the formula

$$\varphi_i(\mathbf{x})=\prod_{k=1}^{n}\exp\left(-\frac{\left\|x_k-\mu_{ik}\right\|^2}{2\sigma_{ik}^2}\right),$$

we obtain

$$\varphi_i(x)=\prod_{k=1}^{n}\exp\left(-\frac{\gamma^2(x_k-\mu_{ik})^2}{2I_{ik}^2}\right).$$

For example, the Solution Tree branch that corresponds to the condition $x_1 \geq 5$ and $x_2 \geq 7$ (Fig. 2) describes the rectangle number «9» colored in grey, defined as $x_1 \in [5, 10]$ and $x_2 \in [7, 10]$ (on conditions that 10 is a maximum value for the attributes x_1 and x_2). It could be seen that $I_{91}=5$ and $I_{92}=3$. Using $\gamma=\sqrt{2}$ ($\gamma^2/2=1$), the activation function of the corresponding neuron is defined as ($\mu=[10\ 10]^T$):

$$\varphi_9(\mathbf{x})=\exp\left(-\frac{(x_1-10)^2}{25}\right)\cdot\exp\left(-\frac{(x_2-10)^2}{9}\right).$$

The sizes of hyperparallelepipeds separately for each attribute determine the Gaussian function dispersion.

Step 3. The determination of the output layer weight

Let the RBF transformed φ – examples from the learning set be fixed in the form of matrix $\boldsymbol{X}$ so that every row expresses one example and the i-th column contains the value φ_i of this example. Zero attribute $\varphi_0=0$.

Let $\mathbf{C}$ be the classification matrix, where every column supports one class mark: if the r-th example is marked by the j-th class, then the j-th item in the r-th row of $\mathbf{C}$ equals "1", and all other items in this row equal "–1" (or "0"). The task is to determine the weight vector $\mathbf{W}$ (matrix in the case of class encoding as "$[1\ 0]^T$" and "$[0\ 1]^T$" or "$[1\ -1]^T$" and "$[-1\ 1]^T$") that enables to minimize the mean square error $(\mathbf{XW}-\mathbf{C})$. This could be reached using pseudo inverse matrix $\mathbf{X}^+$ and the fact that the mean square error is minimized if:

$$\mathbf{W}=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{C}=\mathbf{X}^+\mathbf{C}. \qquad (2)$$

The Solution Tree, usually, has to be reduced (cropped). During the process of reduction, some of its sub-trees are either removed or replaced by the smaller sub-trees. In multidimensional domains, one branch of the Solution Tree distinguishes only a part of the attributes.

Let us define the algorithm of RBF network development based on the CART Solution Tree generator.

Step 1. To split the space of examples into almost homogeneous domains in the form of hyperparallelepipeds (this could be done using the Solution Tree generator).

Step 2. To connect each hyperparallelepiped to one RBF neuron (its parameters depend on the location and dimensions of hyperparallelepiped).

Step 3. To organize the neurons into one hidden layer. To determine the weight of the output layer.

The first step divides the space of investigated objects into almost homogeneous domains of hyperparallelepipeds (in each domain one class dominates). This division could be implemented using the CART Solution Tree generator. As soon as hyperparallelepipeds are identified, each of them a neuron with parameters depending on the dimensions of this hyperparallelepiped is assigned. The weight of the output layer is determined by the formula (2), which maximizes the classification accuracy of the network at the learning set.

5. 2. The experimental research description and results

The formulated problem is solved experimentally for $\gamma=\sqrt{2}$.

Firstly, make sure that the RBF network obtained as a result of learning based on this method is able to simulate the usual Solution Tree (Fig. 2). For this purpose, we develop the RBF network which corresponds to this classification Tree for the points from the domain $X=[0, 10]\times[0, 10]$.

Let us identify the learning set S_1 that contains the points (vectors) from all nine sub-domains and the test set S_2:

$S_1=\{[1\ 1]^T, [1\ 8]^T, [1\ 5]^T, [3.5\ 5.5]^T, [4\ 9]^T, [8\ 2]^T, [6.5\ 5]^T, [8.5\ 5]^T, [8\ 8]^T\}$;

$S_2=\{[0.1\ 0.1]^T, [0.5\ 8]^T, [0.1\ 5]^T, [3\ 5]^T, [2.5\ 8.5]^T\}$.

On the learning set, we obtained the following classes

$C=\{1; 1; -1; 1; -1; -1; 1; -1; -1\}$,

that totally correspond to their sub-domains. On the test set, RBF network modeled the following set of classes:

$C=\{1; 1; -1; 1; -1\}$

for the points from the corresponding sub-domains-branches "1", "3", "2", "4", "5" (Fig. 3).

Table 1

The result of RBF network modeling on the test set S_2

S_2	Network output y_1	Class – right answer	Network output $[y_1\ y_2]^T$
$[0.1\ 0.1]^T$	1.1990	1 or $[1\ 0]^T$	$[1.1584\ -0.0406]^T$
$[0.5\ 8]^T$	1.2510	1 or $[1\ 0]^T$	$[1.1916\ -0.0594]^T$
$[0.1\ 5]^T$	–1.3299	–1 or $[0\ 1]^T$	$[-0.0829\ 1.2470]^T$
$[3\ 5]^T$	0.9285	1 or $[1\ 0]^T$	$[0.8632\ -0.0653]^T$
$[2.5\ \ 8.5]^T$	–0.0446	–1 or $[0\ \ 1]^T$	$[0.3657\ \ 0.4103]^T$

Let us introduce changes into Fig. 2 by adding one variable $x_3 \in [0; 10]$ into the first branch (Fig. 6). Make sure that the RBF network obtained as a result of the learning based on this method is able to simulate Solution Tree (Fig. 6).

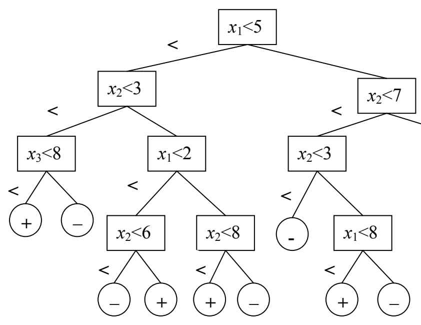

Fig. 6. The Solution Tree that splits three-dimensional space into hyperparallelepipeds

Let us develop the RBF network that corresponds to this Solution Tree for the points (vectors)

$S_1=\{[1\ 1\ 1]^T, [1\ 1\ 9]^T, [1\ 8\ 2]^T, [1\ 5\ 1]^T, [3.5\ 5.5\ 1]^T, [4\ 9\ 1]^T, [8\ 2\ 1]^T, [6.5\ 5\ 1]^T, [8.5\ 5\ 1]^T, [8\ 8\ 1]^T\}$.

These points correspond to the set of sub-domains-branches {1, 2, 4, 3, 5, 6, 7, 8, 9, 10}. The following set of classes that totally correspond to their sub-domains was obtained on the learning set

$C=\{1; -1; 1; -1; 1; -1; -1; 1; -1; -1\}]$.

On the test set

S_2={[0.1 0.1 0.1]^T, [0.1 0.1 9.5]^T, [0.5 8 3]^T, [0.1 5 3]^T, [3 5 3]^T, [2.5 8.5 6]^T},

which points belong to the branches {1, 2, 4, 3, 5, 6}, RBF network modeled the following set of classes:

C={1; –1; 1; –1; 1; –1}.

Table 2

The result of RBF network modeling on the test set S_2

S_2	Network output y_1	Class – right answer	Network output $[y_1\ y_2]^T$
[0.1 0.1 0.1]^T	1.2165	1 or [1 0]^T	[1.1763 –0.0402]^T
[0.1 0.1 9.5]^T	–1.4712	–1 or [0 1]^T	[–0.1169 1.3543]^T
[0.5 8 3]^T	1.2323	1 or [1 0]^T	[1.1909 –0.0414]^T
[0.1 5 3]^T	–1.3630	–1 or [0 1]^T	[–0.0922 1.2708]^T
[3.0 5 3]^T	0.7766	1 or [1 0]^T	[0.8526 0.0760]^T
[2.5 8.5 6]^T	–0.0970	–1 or [0 1]^T	[0.3638 0.4608]^T

It does not matter how the classes are encoded: "1" and "–1" or "[1 0]^T" and "[0 1]^T" or "[1 –1]^T" and "[–1 1]^T".

Let us make sure that the RBF-network after the learning based on this method is able to classify biomedical data. The parameter that manages the dispersion of Gaussian functions in the research has the form $\gamma=\sqrt{2}$. To test and compare the algorithm execution, machine learning researchers usually conduct experiments with publicly available benchmark domains in such a way that their colleagues can repeat these results. A well-known common source of benchmark data is a warehouse collected by the Department of Computer Science at the University of California, Irvine [10]. The accuracy of classification based on this method is demonstrated. Let us verify that the method works on the domain boundaries, where the surface of the solutions is nonlinear and the data are noisy. The cases when examples describe numerical attributes are considered.

The pseudo-inverse matrix usage has limited the number of experiments with domains – no more than a thousand examples. Taking this into account, the following recognition algorithms databases were selected [10]:

– diabetes, Pima Indians Diabetes Database (Table 3), which contains examples of data on patients with and without diabetes: the number of attributes equals 8, examples 768, classes – 2;

– breast cancer, Breast cancer database (Table 6, experiment 2): the number of attributes equals 10 (the first attribute «ID number of a patient» was deleted), examples 699, classes – 2 (examples with a question mark «?» were deleted, 683 left);

– heart diseases, Heart disease data – (Table 6, experiment 3): the number of attributes equals 13, examples 270, classes – 2;

– dermatology diseases, Dermatology Database – (Table 7, experiment 4): the number of attributes equals 34, examples 366, classes – 1÷6 (examples with a question mark «?» were deleted, 358 left).

The RBF network development algorithm was constructed. For the CART Solution Tree determination, MATLAB system with a constant (when possible) number of reductions was used.

The performance of the binary classifier is affected by the definition of sets S_1 and S_2 (S_1 – learning set, S_2 – test set or control sample). The obtained result is shown in Table 3–5:

$$error=\sum_{i=1}^{Q}\left|F(\mathbf{w},\mathbf{x}^i)-t^i\right|,$$

$$\gamma=\sqrt{2}.$$

Table 3

The result of RBF network modeling (the first experiment) using the Diabetes database

No. 1 exper.	Number of reductions, RBF neurons	Number of outputs	Number of examples	Result of modeling		Accuracy (%)	S_E (%)	S_P (%)
1.1	22 out of 34, 17 RBF neurons	1, results interpretation (y>0.5)	Learning S_1=1:380	91	31	77.63	74.59	79.07
				54	204			
			Test S_2=381:768	76	34	79.12	69.09	83.09
				47	231			
		2, results interpretation (y_1>y_2)	Learning S_1=1:380	93	37	76.58	71.54	79.2
				52	198			
			Test S_2=381:768	73	37	77.58	66.36	82.01
				50	228			
1.2	18 out of 26, 9 RBF neurons	1, results interpretation (y>0.5)	Learning S_1=381:768	79	25	82.22	75.96	83.33
				44	240			
			Test S_2=1:380	79	32	74.21	71.17	75.46
				66	203			
		2, results interpretation (y_1>y_2)	Learning S_1=381:768	83	22	84.02	79.05	85.87
				40	243			
			Test S_2=1:380	80	38	72.89	67.797	75.19
				65	197			
1.3	22 out of 34, 17 RBF neurons	1, results interpretation (y>0.5)	Learning (384), S_1-odd	74	31	76.04	70.48	78.14
				61	218			
			Test (384), S_2-even	76	27	78.13	73.79	79.72
				57	224			
		2, results interpretation (y_1>y_2)	Learning, S_1-odd	83	26	79.69	76.15	81.09
				52	223			
			Test, S_2-even	74	32	76.30	69.81	78.78
				59	219			
1.4	20 out of 29, 12 RBF neurons	1, results interpretation (y>0.5)	Learning, S_1-even	79	18	81.25	81.44	77.93
				54	233			
			Test, S_2-odd	69	34	73.96	66.99	76.51
				66	215			
		2, results interpretation (y_1>y_2)	Learning, S_1-even	83	21	81.51	79.81	82.14
				50	230			
			Test, S_2-odd	74	36	74.74	67.27	77.74
				61	213			

The modeling results (binary classification during diabetes diagnostics) based on RBF network show that the average accuracy of modeling on a learning set makes 79.87 %, on a test set – 75.87 %. If there is one output, then the classes are encoded as "1" and "0" (or "–1"); if there are two, then the classes are encoded as "$[1\ 0]^T$", "$[0\ 1]^T$" (or "$[1\ -1]^T$", "$[-1\ 1]^T$"). For instance, the data in Table 4 show that the algorithm correctly identified the classes for 91+204=295 out of 380 learning examples. The accuracy shows how many correct results were obtained after the application of the given method.

Table 4

Interpretation of the experimental results

Class	The result of RBF network modeling		Accuracy (%)
	Class 1 (True/False)	Class 2 (True/False)	
Class 1 = "1"	91 (true "1")	31 (false "0")	((91+204)/380)× ×100 %=77.63 %
Class 2 = "0" or "–1"	54 (false "1")	204 (true "0")	

The results of the research indicate that the number of reductions that lead to smaller trees and, consequently, tighter RBF networks affect the result of modeling (Table 5):

– during the learning and reduction of 22 out of 34, the number of false answers equals 92 (for the RBF network with one output) and 78 (for the RBF network with two outputs);

– during the learning and reduction of 5 out of 34, the number of false answers equals 71 and 70 respectively.

Thus, the reductions number increase makes the result of modeling worse.

Table 5

The result of RBF network modeling: the first experiment with different reductions number

Number of reductions, RBF neurons	Number of outputs	384 examples	Result of modeling		Accuracy (%)	S_E (%)	S_P (%)
22 out of 34, 17 RBF neurons	1	S_1-odd	74	31	76.04	70.48	78.14
			61	218			
		S_2-even	76	27	78.13	73.79	79.72
			57	224			
	2	S_1-odd	83	26	79.69	76.15	81.09
			52	223			
		S_2-even	74	32	76.30	69.81	78.78
			59	219			
5 out of 34, 42 RBF neurons	1	S_1-odd	86	22	**81.51**	79.63	82.25
			49	227			
		S_2-even	68	41	72.4	62.39	76.36
			65	210			
	2	S_1-odd	93	28	**81.77**	76.86	84.03
			42	221			
		S_2-even	72	50	71.09	59.02	76.72
			61	201			

After the research results from Table 5 were taken into account, during the further research the number of reductions was minimized for the other databases (the number of outputs equals "1", results interpretation: y>0.5).

Table 6

The result of RBF network modeling using other databases

No. exper.	Number of reductions, RBF neurons	Number of examples	Result of modeling		Accuracy (%)	S_E (%)	S_P (%)
2	3 out of 9, 11 RBF neurons	Learning, S_1=1:345	129	7	88.696	94.85	84.69
			32	177			
		Test, S_2=346:683	71	4	96.75	94.67	97.34
			7	256			
3	1 out of 9, 14 RBF neurons	Learning, S_1=1:135	51	2	90.37	96.23	86.59
			11	71			
		Test, S_2=136:270	39	6	81.48	86.67	78.89
			19	71			

Let us conduct the research of the given RBF network development method efficiency for the greater number of classes.

Table 7

The result of RBF network modeling using six classes data (Dermatology database)

No. exper.	Number of reductions, RBF neurons	Number of examples	Number of outputs		Accuracy (%)
			correct	false	
4.1	0 out of 13, 16 RBF neurons	Learning S_1=1:183	154	29	84.15
		Test S_2=184:358	122	53	69.71
4.2	0 out of 9, 12 RBF neurons	Learning S_1=184:358	161	14	92
		Test S_2=1:183	146	37	79.78

The efficiency of assessment algorithms of the current state of objects is one of the main characteristics of computer systems that provide solutions for the medical diagnostics problems. A convenient tool for the efficiency assessment of a diagnostic algorithm is a method based on the analysis of the so-called Receiver Operating Characteristic curve (ROC) [11–12]. Traditional ROC analysis involves a comparison of the operational characteristics of the algorithm – sensitivity (S_E) and specificity (S_P). With these characteristics, the algorithm checking results could be depicted in a two-dimensional ROC-space where the axis of ordinates represents S_E from Table 3 and the axis of abscissas represents $1-S_P$ (Fig. 7). Thus, diagnostic test (binary classifier) with fixed operational characteristics is represented by a point in a ROC-space (Fig. 7).

The ROC-space gives a graphical representation of the diagnostic value of the algorithm (test) and allows the comparison of the efficiency of different algorithms. The perfect test is located at the point with coordinates (0.1) (or vector

$[0\ 1]^T$) of the ROC-space. This test always records sick patients and doesn't record healthy people to the class of patients. Diagnostically valuable tests are located in the upper-left corner of the ROC-space: the closer the point which is defined by the operational characteristics S_E and S_P to the point (0, 1), the higher the algorithm efficiency [11]. There is an opportunity to evaluate the sensitivity and specificity of the algorithm for different values of S_1 and S_2 that correspond to the sequence of the binary classifier points in the ROC-space (Fig. 7).

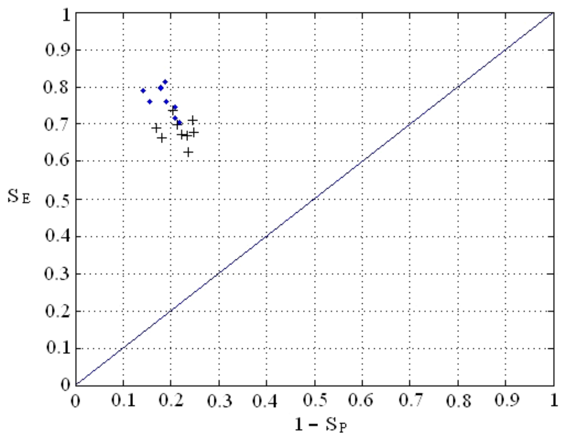

Fig. 7. Binary tests in a ROC-space: "•" corresponds to the learning regime, "+"— test regime

6. The experimental research results discussion

It should be noted that the classifier based on the RBF network was implemented on Python. MATLAB system was used to develop the CART Solution Tree. The number of RBF neurons and the values of parameters of these neurons (centre and dispersion) were determined based on the obtained Solution Tree. On the basis of the two matrices of centres and dispersions, the corresponding matrix **X** was formed on Python using formula (2). To determine the output layer weight according to the formula (2), the NumPy library (Python's language extension) functions were used. It should be noted that it is possible to implement this classifier in the programming languages C++, C# and Java.

This method of RBF network development shows a decrease in the number of connections between the input and hidden layers. While the number of RBF neurons is equal to the number of the Solution Tree branches, the number of connections with any hidden neuron would be determined by the number of fixed attributes in the corresponding tree branch.

The research results show that:

– the reductions number increase leads to the modeling result worsening (Table 5);

– the acceptable accuracy of classification on the test set (for instance, 80 % from Table 7, experiment 4.2) could be obtained when the number of classes is large.

The sensitivity to the attributes (which are not important) is very small because Solution Tree generators usually remove some branches of the tree during the reduction method application. In different parts of the objects characteristics space, the relevance of separate attributes may vary. Each branch of the Solution Tree distinguishes the part of attributes that is relevant to the corresponding subspace. The advantage of the method compared to most methods of RBF network development is that the number of parameters is significantly reduced.

The advantage of the research is that the sensitivity and specificity of the binary classifier development method were assessed for different values of S_1 and S_2, that correspond to the sequence of this classifier points in the ROC-space. The obtained research results (Tables 4, 5) show that RBF networks based on the Gaussian function and the CART Solution Tree could be used for the binary classifiers development.

Despite the fact that the conducted research continues a longstanding process of the neural and RBF networks capabilities studying, it has several potentially attractive directions of development. Among them, it is worthwhile to emphasize the following:

– the improvement of the given classifier possibilities, for instance, using the process of appropriate biomedical data preparation prior to the CART Solution Tree development;

– the research of efficiency of other RBF network development methods.

7. Conclusions

1. The universal classifier development method is described and implemented on the example of the diseases diagnostics classifier that was obtained by combining the possibilities of a radial basis function network (RBF network) based on the Gaussian function and the CART Solution Tree. This method allows splitting the space of examples into relatively homogeneous sub-domains (hyperparallelepipeds), each of which is associated with one of the RBF neurons. The number of RBF neurons and parameters of these neurons are determined automatically directly based on the CART Solution Tree.

2. It is found that:

– these classifiers show the highest efficiency on the learning set having the minimal reduction of the Solution Tree (accuracy from 80 % to 95 %); the number of reductions increase usually leads to the modeling result worsening;

– for two and more classes their accuracy on the test set makes 79 % and more (provided that a suitable data sample is selected for the learning set);

– the possibility of using such NN in the healthcare system for the diseases diagnostics and medical systems (or devices) assistance during decision-making support was proved.

References

1. Dobrovska, L. M. Teoriya ta praktyka neironnykh merezh [Text]: navch. pos. / L. M. Dobrovska, I. A. Dobrovska. – Kyiv: NTUU «KPI» Vyd-vo «Politekhnika», 2015. – 396 p.
2. Osovskiy, S. Neyronnye seti dlya obrabotki informaciy [Text] / S. Osovskiy. – Moscow: Finansy i statistika, 2004. – 343 p.
3. Haykin, S. Neyronnye seti [Text] / S. Haykin. – 2-e izd. – Moscow: Izdatel'skiy dom «Vil'yams», 2006. – 1104 p.

4. Hagan, M. Neural Network Design [Text] / M. Hagan, H. Demuth, M. Beale. – USA: Colorado University Bookstore, 2010.
5. Kitayama, Y. Pattern Classification by RBF Network [Text] / Y. Kitayama, S. Kitayama, K. Yamazaki // The Proceedings of Design & Systems Conference. – 2008. – Vol. 2008.18. – P. 330–332. doi: 10.1299/jsmedsd.2008.18.330
6. Kubat, M. Decision trees can initialize radial-basis function networks [Text] / M. Kubat // IEEE Transactions on Neural Networks. – 1998. – Vol. 9, Issue 5. – P. 813–821. doi: 10.1109/72.712154
7. Safish, M. Data classification with neural classifier using radial basis function with data reduction using hierarchical clastering [Text] / M. Safish, R. Joseph // ICTACT Journal on Soft Computing. – 2012. – Vol. 2, Issue 3. – P. 348–352. doi: 10.21917/ijsc.2012.0054
8. Jia, W. An optimized RBF neural network algorithm based on partial least squares and genetic algorithm for classification of small sample [Text] / W. Jia, D. Zhao, L. Ding // Applied Soft Computing. – 2016. – Vol. 48. – P. 373–384. doi: 10.1016/j.asoc.2016.07.037
9. Jang, J.-S. Neuro-fuzzy and soft computing: a computation approach to learning and machine intelligence [Text] / J.-S. Jang, C.-T. Sun, E. Mizutani. – USA, 1997. – 614 p.
10. Blake, C. L. UCI Repository of machine learning databases [Text] / C. L. Blake, C. J. Merz. – University of California Irvine, Department of Information and Computer Science, 1998. – Available at http://archive.ics.uci.edu/ml/datasets.html
11. Faynzil'berg, L. Garantirovannaya ocenka ehffektivnosti diagnosticheskih testov na osnove usilennogo ROC-analiza [Text] / L. Faynzil'berg // Upravlyayushchie sistemy i mashiny. – 2009. – Issue 5. – P. 3–13.
12. Flach, P. Repairing concavities in ROC curves [Text] / P. Flach, S. Wu. – Proc. UK Workshop on Comp. Intel, 2003. – P. 38–44.