Information Sciences 430–431 (2018) 65–76

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Weighted probabilistic neural network

Maciej Kusy[a,*], Piotr A. Kowalski[b,c]

[a] Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, al. Powstancow Warszawy 12, 35–959, Rzeszow, Poland

[b] Faculty of Physics and Applied Computer Science, AGH University of Science and Technology, al. A. Mickiewicza 30, 30–059, Cracow, Poland

[c] Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01–447, Warsaw, Poland

ARTICLE INFO

Article history:
Received 1 March 2017
Revised 14 September 2017
Accepted 17 November 2017
Available online 21 November 2017

Keywords:
Probabilistic neural network
Weights
Sensitivity analysis
Classification
Accuracy

ABSTRACT

In this work, the modification of the probabilistic neural network (PNN) is proposed. The traditional network is adjusted by introducing the weight coefficients between pattern and summation layer. The weights are derived using the sensitivity analysis (SA) procedure. The performance of the weighted PNN (WPNN) is examined in data classification problems on benchmark data sets. The obtained WPNN's efficiency results are compared with these achieved by a modified PNN model put forward in literature, the original PNN and selected state-of-the-art classification algorithms: support vector machine, multilayer perceptron, radial basis function neural network, *k*-nearest neighbor method and gene expression programming algorithm. All classifiers are collated by computing the prediction accuracy obtained with the use of a *k*-fold cross validation procedure. It is shown that in seven out of ten classification cases, WPNN outperforms both the weighted PNN classifier introduced in literature and the original model. Furthermore, according to the ranking statistics, the proposed WPNN takes the first place among all tested algorithms.

© 2017 Elsevier Inc. All rights reserved.

1. Introduction

Classical feedforward neural networks such as multilayer perceptron or radial basis function network have their layers linked using weighted connections. Within the training process, the utilized weights must be first initialized and then iteratively recomputed to optimize some assumed performance measure of the model for a given training data. The probabilistic neural network [36,37], however, is the model unequipped with any additional weighting factors inside its structure. PNN is therefore free from time consuming weights' update. This fact gives this network the advantage in application popularity. The usage of PNN can be found in the domains of medical diagnosis and prediction [9,15,17,18,20], image classification and recognition [4,25,45], earthquake magnitude prediction [1], multiple partial discharge sources classification [43], interval information processing [12–14], phoneme recognition [7], email security enhancement [41], intrusion detection systems [40], classification in a time-varying environment [27,28] or hardware implementation [46]. The variant of PNN in regression tasks, known as general regression neural network [38], is also studied by many authors, e.g.: in function approximation [10] or knowledge discovery in data streams [6].

Four layers create a structure of a conventional PNN: the input layer represented by data attributes, the pattern layer composed of as many neurons as training patterns, the summation layer containing a single neuron for each class and

* Corresponding author.
E-mail address: mkusy@prz.edu.pl (M. Kusy).

https://doi.org/10.1016/j.ins.2017.11.036
0020-0255/© 2017 Elsevier Inc. All rights reserved.

the output layer with a decision neuron which determines the classification outcome. In the research centered on PNN architecture, some minor emphasis is usually placed on applying the weights inside this network. The first contribution related to PNN's weights is presented in [22]. However, the coefficients are not computed explicitly. The network operates using anisotropic Gaussians to provide the output for a considered sample by utilizing the covariance matrix instead of a single smoothing parameter in the activation function. In the works of [34] and [35], the weighting coefficients are directly introduced inside the PNN model. They are placed between pattern and summation layer. These factors are calculated from soft labeling probability matrix based on Bayesian algorithm. In turn, the authors of [23–25] create weighted PNN based on the class separability in the data. The weight coefficients are defined as the ratio of 'between-class variance' and 'within-class variance' for a particular training pattern. They are included between a pattern and summation layer of PNN.

In this paper, the weighted probabilistic neural network is introduced. For the computation of the weights, we propose the use of SA procedure. As in the case of [23–25,34,35], the coefficients are placed between pattern and summation layer of the network. The formulas for the weights are analytically derived. The model with a product Cauchy kernel activation function is utilized. The performance of the proposed WPNN is examined in the classification problems of the UCI machine learning repository (UCI–MLR) data sets [19] calculating a k-fold cross validation accuracy. Furthermore, we verify WPNN accuracy with the one obtained for five state-of-the-art classifiers: support vector machine, multilayer perceptron, radial basis function neural network, k-nearest neighbor method and gene expression programming algorithm.

The current work is an extension of the preliminary study published by the same authors in [16]. We enrich the previous paper by comparing the proposed WPNN with the modified probabilistic neural network (MPNN) introduced in [23–25] where the weighing factors are determined on the basis of the class variances in the data set. Furthermore, the obtained results are averaged so that the mean and the standard deviation is computed in each data classification case. Therefore, the results are reliable and worth referring. We also introduce the ranking statistics within the comparison of WPNN, MPNN, original PNN and the state-of-the-art classification algorithms. This shows in turn, which model performs best among all tested ones. Finally, we perform the Friedman test to present that the utilized comparison method is appropriate.

At this point, it is necessary to stress the elements of novelty of our study. First of all, despite existing approaches on PNN weighting factors available in literature, the analytical formula for the weights of this network has not been provided. In this paper, such a solution is given. Secondly, the proposed formula is easy to implement and is additionally interpretable since it stems from the SA procedure. Moreover, SA has not been utilized for computing the PNN's weights up to this date. Finally, no additional iterative procedure, which relies on the minimization of some error function, is involved to compute the weights. Therefore, the loss in computational time is marginal.

The rest of this paper is organized as follows. In Section 2, the SA procedure is presented. Section 3 explains the operation of the PNN model including the structure of the network, and the computation of the smoothing parameter and the modification coefficient. In Section 4, we show the way of deriving the formula for the PNN's weights. Section 5 outlines the input data sets and the selected classifiers utilized in our study. Here, the obtained results are also discussed. The final conclusions are drawn in Section 6.

2. Sensitivity analysis

SA procedure is frequently applied in determining the influence of particular inputs of a neural network. Therefore, it can be utilized in the elimination of redundant attributes of the input data. The main idea of SA is based on establishing the importance of input attributes on a neural network output after a training process. Such an importance is defined by real coefficients [48]

$$S_{j,i}^{(p)} = \frac{\partial}{\partial x_i} y_j\left(x_1^{(p)}, x_2^{(p)}, \ldots, x_N^{(p)}\right), \tag{1}$$

where x_i is an input attribute and y_j denotes an output signal. In (1), $i = 1, \ldots, N$ and $j = 1, \ldots, J$, where N and J stand for the number of attributes and outputs, respectively.

Eq. (1) corresponds to the sensitivity of the jth neural network output on the ith attribute of a sample vector $\mathbf{x}$ obtained on the basis of the pth training pattern $\mathbf{x}^{(p)}$ for $p = 1, \ldots, P$ where P represents a cardinality of a data set. Considering an N dimensional data set, Eq. (1) can be presented as the following matrix

$$\mathbf{S}^{(p)} = \left\{S_{j,i}^{(p)}\right\}_{J\times N}. \tag{2}$$

Once $\mathbf{S}^{(p)}$ is determined for all P training patterns, we are capable of finding aggregated parameters using various types of norms. In the research, the mean square average sensitivity parameter is usually applied

$$S_{j,i}^{\text{mean}} = \sqrt{\frac{1}{P}\sum_{p=1}^{P}\left(S_{j,i}^{(p)}\right)^2}. \tag{3}$$

The absolute value average sensitivity and the maximum sensitivity parameters are also commonly used in input significance estimation [47]. The selection of a particular norm is influenced by an impact of $S_{j,i}^{(p)}$ on the aggregated outcome of $S_{j,i}^{\text{mean}}$.

The final stage of SA relies on the exploration of particular sensitivities $S_{j,i}$. For this purpose, the maximum value Q_i in the ith column of the matrix (2) must be defined with the elements aggregated according to some assumed norm, e.g.

$$Q_i = \max_{j=1,\ldots,J} \left\{ S_{j,i}^{\text{mean}} \right\}. \tag{4}$$

If we sort Q_i coefficients in descending order, it is possible to evaluate the significance of a neural network's inputs on its final output. Such a significance can be established with the use of the difference between two neighboring elements

$$\Delta Q_i = Q_i - Q_{i+1}, \tag{5}$$

for $i = 1, \ldots, N-1$. An explicit difference $\Delta Q_{\max}$ between Q_i and Q_{i+1} gives us the hint that all smaller Q_i elements have minor impact on a neural network's output.

3. Probabilistic neural network

The operation of the PNN classifier is based on the kernel density estimator (KDE) which, in general, takes the following form

$$\hat{f}(\mathbf{x}) = \frac{1}{Ph^N} \sum_{p=1}^{P} K\left(\frac{\mathbf{x} - \mathbf{x}^{(p)}}{h} \right), \tag{6}$$

where $\mathbf{x} = [x_1, \ldots, x_N]$ is a testing sample, h denotes the smoothing parameter and $K(\cdot)$ is the kernel function which performs the mapping $\mathbb{R}^N \to [0, \infty)$. For multidimensional data sets, KDE is often generalized to product notation

$$K(\mathbf{x}) = \mathcal{K}(x_1) \cdot \mathcal{K}(x_2) \cdot \ldots \cdot \mathcal{K}(x_N), \tag{7}$$

where

$$\mathcal{K}(x_i) = \frac{2}{\pi (x_i^2 + 1)^2} \tag{8}$$

stands for the one-dimensional Cauchy multiplicand. The form of the kernel in (8) should be chosen so as to guarantee its useful analytic form for derivation operation. Moreover, it has been shown that the difference in the selection of kernel function implies approximately 4% difference in the quality of the nonparametric estimation of the density function [30,44]. However, the use of kernel function is primarily dependent on application needs. In the current study, the product form of KDE in (7) with Cauchy realization (8) is used for PNN implementation.

3.1. Structure of the model

PNN consists of four layers. The attributes of an input vector $\mathbf{x}$ comprise the first input layer. The second, pattern layer is composed of as many neurons as training data. The output of the pattern neurons is fed forward to the next summation layer. There are J neurons in the summation layer, with each jth node acquiring signals from the neurons of jth class. Two approaches are usually utilized in literature to compute the output of the pattern layer: the radial kernels [18,37] and the product kernels [14,44]. In this work, we use the second approach, therefore the summation neuron output is defined in the following way

$$\hat{f}_j(\mathbf{x}) = \frac{1}{P_j \det(\mathbf{h})} \sum_{p=1}^{P_j} \frac{1}{s_p^N} K\left(\frac{\left(\mathbf{x} - \mathbf{x}_j^{(p)}\right)^T \mathbf{h}^{-1}}{s_p} \right), \tag{9}$$

where:

- P_j denotes data cardinality in the jth class ($j = 1, \ldots, J$);
- $\mathbf{h} = \text{diag}(h_1, \ldots, h_N)$ is the diagonal matrix consisting of the smoothing parameters h_i;
- $\det(\mathbf{h}) = \prod_{i=1}^{N} h_i$ defines the determinant of $\mathbf{h}$;
- s_p stands for the modification coefficient;
- $\mathbf{x}_j^{(p)} = [x_{j,1}^{(p)}, \ldots, x_{j,N}^{(p)}]$ indicates the pth training pattern belonging to the jth class.

Eq. (9) is also referred to as KDE for the PNN's jth class. If as the pattern neuron activation function we consider Eqs. (7) and (8), KDE for the PNN's jth class amounts to

$$\hat{f}_j(\mathbf{x}) = \frac{1}{P_j \det(\mathbf{h})} \sum_{p=1}^{P_j} \frac{1}{s_p^N} \prod_{i=1}^{N} \frac{2}{\pi \left(\left(\frac{x_i - x_{j,i}^{(p)}}{h_i s_p} \right)^2 + 1 \right)^2}. \tag{10}$$

Finally, the single neuron in the PNN's output layer produces the decision outcome for a new test vector $\mathbf{x}$ using Bayes theorem [37]

$$C(\mathbf{x}) = \underset{j=1\ldots J}{\operatorname{argmax}} \; f_j(\mathbf{x}), \tag{11}$$

where $C(\mathbf{x})$ denotes the predicted class.

The PNN training algorithm relies on the appropriate choice of the smoothing parameter h_i and the modification coefficient s_p. The way of how to determine these parameters is explained in Section 3.2.

3.2. Smoothing parameter

The prediction accuracy of the PNN classifier varies from the value of the smoothing parameter. Therefore, the algorithm which suitably updates the value of h_i needs to be implemented. In this paper, the smoothing parameter is treated as the N element vector whose coordinates correspond to the attributes of the input pattern; $\mathbf{h}$ is computed independently for a particular class of the data set.

In the case when KDE is determined based on the product kernel (7), the plug-in method [44] is recommended to calculate $\mathbf{h}$. Each coefficient h_i is computed according to the following formula

$$h = \left[\frac{R(\mathcal{K})}{U(\mathcal{K})^2} \frac{8\sqrt{\pi}\hat{\sigma}^9}{3P} \right]^{\frac{1}{5}}, \tag{12}$$

where

$$R(\mathcal{K}) = \int_{R^N} \mathcal{K}(x^2)dx, \tag{13}$$

and

$$U(\mathcal{K}) = \int_{R^N} x^2 \mathcal{K}(x)dx, \tag{14}$$

where $\hat{\sigma}$ denotes the estimator of the standard deviation. $R(\mathcal{K}) = 1$ and $U(\mathcal{K}) = 5/4\pi$ for the Cauchy kernel in (8). Since the type of the density of the random variable x is unknown, the approximation of $\hat{\sigma}$ is solved iteratively. In practical applications, the second order level approximation is sufficient to estimate $\hat{\sigma}$ [30,44].

Once the temporary values of the smoothing parameter vector are computed using the plug-in method, the KDE quantities are determined according to (6) for each element $\mathbf{x}^{(p)}$ for $p = 1, \ldots, P$. This allows us to calculate the modification parameter s_p for all $\mathbf{x}^{(p)}$ patterns independently in the following way

$$s_p = \left(\frac{\hat{f}(\mathbf{x}^{(p)})}{\tilde{s}} \right)^{-c}, \tag{15}$$

where $\tilde{s}$ is the geometrical mean of KDE $\hat{f}(\mathbf{x}^{(p)})$ and $c \geq 0$ is the constant of the modification intensity. The increase of c value contributes to the growth of this intensity. Note that for $c = 0$, $s_p \equiv 1$; in such a case, the modification of the smoothing parameter does not take place.

4. Computation of weights

In this section, we derive the analytical formula for the weights of the proposed WPNN. The obtained formula requires solely the knowledge on the smoothing parameter h_i and the modification coefficient s_p. The computed weighting factors must be included between pattern and summation layer of the network. They reflect the importance of a particular pattern p from the jth class.

The idea for determining the weights begins with the calculation of the sensitivity coefficients for all $r = 1, 2, \ldots, P_j$ neurons in the pattern layer

$$S = \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial \mathbf{x}_j^{(r)}}, \tag{16}$$

where $\mathbf{x}_j^{(p)} \in \left\{\mathbf{x}_j^{(1)}, \mathbf{x}_j^{(2)}, \ldots, \mathbf{x}_j^{(P_j)}\right\}$ is the vector argument from jth class. Thus, $\mathbf{S}$ for jth class takes the matrix form

$$\mathbf{S}_j = \begin{bmatrix} \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(1)}\right)}{\partial \mathbf{x}_j^{(1)}} & \cdots & \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(1)}\right)}{\partial \mathbf{x}_j^{(r)}} & \cdots & \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(1)}\right)}{\partial \mathbf{x}_j^{(P_j)}} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial \mathbf{x}_j^{(1)}} & \cdots & \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial \mathbf{x}_j^{(r)}} & \cdots & \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial \mathbf{x}_j^{(P_j)}} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(P_j)}\right)}{\partial \mathbf{x}_j^{(1)}} & \cdots & \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(P_j)}\right)}{\partial \mathbf{x}_j^{(r)}} & \cdots & \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(P_j)}\right)}{\partial \mathbf{x}_j^{(P_j)}} \end{bmatrix}. \tag{17}$$

In (17), the elements of rth column represent the sensitivities of KDE in jth class in regard to each rth pattern neuron computed for a specific input pattern p. Since the denominator of each item of $\mathbf{S}_j$ is a vector, the following gradient has to be determined

$$\nabla \hat{f}_j^{(p,r)} = \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial \mathbf{x}_j^{(r)}} = \left[\frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial x_{j,1}^{(r)}}, \ldots, \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial x_{j,N}^{(r)}}\right], \tag{18}$$

where

$$\frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial x_{j,i}^{(r)}} = \frac{1}{P_j \det(\mathbf{h})} \frac{1}{s_r^N} \frac{\partial}{\partial x_{j,i}^{(r)}} K\left(\frac{\left(\mathbf{x}_j^{(p)} - \mathbf{x}_j^{(r)}\right)^T \mathbf{h}^{-1}}{s_r}\right). \tag{19}$$

The product form of KDE in (7) expands (19) into the following formula

$$\begin{aligned} \frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial x_{j,i}^{(r)}} = \frac{1}{P_j \det(\mathbf{h})} \frac{1}{s_r^N} & \mathcal{K}\left(\frac{x_{j,1}^{(p)} - x_{j,1}^{(r)}}{h_1 s_r}\right) \cdots \\ & \frac{\partial}{\partial x_{j,i}^{(r)}} \mathcal{K}^\star\left(\frac{x_{j,i}^{(p)} - x_{j,i}^{(r)}}{h_i s_r}\right) \ldots \mathcal{K}\left(\frac{x_{j,N}^{(p)} - x_{j,N}^{(r)}}{h_N s_r}\right) \end{aligned}. \tag{20}$$

The Cauchy kernel in (8) allows us to determine the ith part of (20)

$$\frac{\partial}{\partial x_{j,i}^{(r)}} \mathcal{K}^\star\left(x_{j,i}^{(p)}\right) = \frac{8\left(x_{j,i}^{(p)} - x_{j,i}^{(r)}\right)}{\pi h_i^2 s_r^2 \left(\left(\frac{x_{j,i}^{(p)} - x_{j,i}^{(r)}}{h_i s_r}\right)^2 + 1\right)^3}. \tag{21}$$

We can see that $\nabla \hat{f}_j^{(p,r)}$ is a vector field, therefore, in order to extract an information on sensitivity of KDE of the jth class for a given rth pattern neuron, it is necessary to determine the norm of (18). Thus, $\mathbf{S}_j$ in (17) must be generalized to the following matrix

$$\mathbf{S}_j = \begin{bmatrix} \left\|\nabla \hat{f}_j^{(1,1)}\right\|_n & \cdots & \left\|\nabla \hat{f}_j^{(1,r)}\right\|_n & \cdots & \left\|\nabla \hat{f}_j^{(1,P_j)}\right\|_n \\ \vdots & & \vdots & & \vdots \\ \left\|\nabla \hat{f}_j^{(p,1)}\right\|_n & \cdots & \left\|\nabla \hat{f}_j^{(p,r)}\right\|_n & \cdots & \left\|\nabla \hat{f}_j^{(p,P_j)}\right\|_n \\ \vdots & & \vdots & & \vdots \\ \left\|\nabla \hat{f}_j^{(P_j,1)}\right\|_n & \cdots & \left\|\nabla \hat{f}_j^{(P_j,r)}\right\|_n & \cdots & \left\|\nabla \hat{f}_j^{(P_j,P_j)}\right\|_n \end{bmatrix}, \tag{22}$$

where for $n = 2$

$$\left\|\nabla \hat{f}_j^{(p,r)}\right\|_n = \left(\sum_{i=1}^{N} \left|\frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial x_{j,i}^{(r)}}\right|^n\right)^{\frac{1}{n}} \tag{23}$$

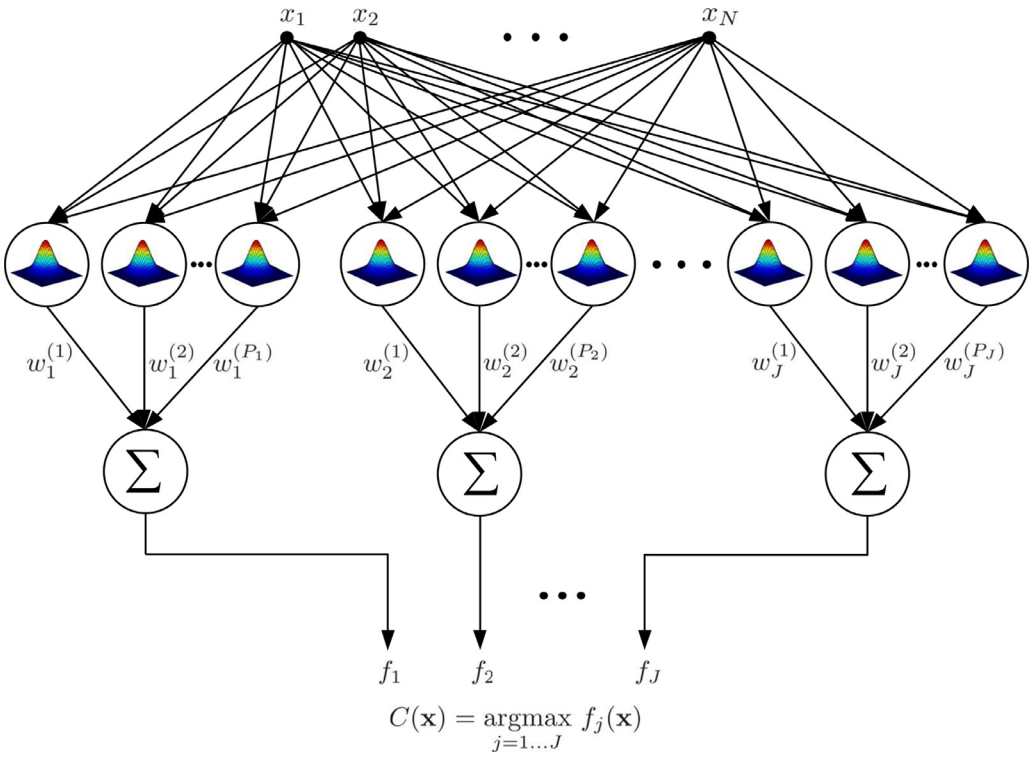

Fig. 1. The architecture of the weighted probabilistic neural network.

the Euclidean norm is utilized.[1] The matrix $\mathbf{S}_j$ is computed for $j = 1, \ldots, J$.

Now it is necessary to aggregate all $p = 1, \ldots, P_j$ entries in each rth column of $\mathbf{S}_j$ in (22) to obtain the aggregated sensitivity vector. For the mean square average sensitivity measure (3), this vector takes the following form

$$\mathbf{a}_j = \left[\sqrt{\frac{1}{P_j}\sum_{p=1}^{P_j} \left\| \nabla \hat{f}_j^{(p,1)} \right\|_n^2}, \ldots, \sqrt{\frac{1}{P_j}\sum_{p=1}^{P_j} \left\| \nabla \hat{f}_j^{(p,P_j)} \right\|_n^2} \right]. \tag{24}$$

Finally, the normalization of the elements of $\mathbf{a}_j$ to some interval introduces the weight vector for each jth class

$$\mathbf{w}_j = \frac{1}{\left\| \mathbf{a}_j \right\|_n} \left[a_j^{(1)}, a_j^{(2)}, \ldots, a_j^{(P_j)} \right]. \tag{25}$$

Formally, for $n = \infty$, $\left\| \mathbf{a}_j \right\|_n = \max \left\{ |a_j^{(1)}|, \ldots, |a_j^{(P_j)}| \right\}$, therefore, in such a case $\mathbf{w}_j$ is normalized to [0, 1] interval. Including the weights' coefficients, KDE in (9) is computed according to

$$\hat{f}_j(\mathbf{x}) = \frac{1}{P_j \det(\mathbf{h})} \sum_{\{p,r\}=1}^{P_j} w_j^{(r)} \frac{1}{s_p^N} K\left(\frac{\left(\mathbf{x} - \mathbf{x}_j^{(p)}\right)^T \mathbf{h}^{-1}}{s_p} \right), \tag{26}$$

where

$$w_j^{(r)} = \frac{1}{\left\| \mathbf{a}_j \right\|_{n_1}} \sqrt{\frac{1}{P_j}\sum_{p=1}^{P_j} \left\| \nabla \hat{f}_j^{(p,r)} \right\|_{n_2}^2}, \tag{27}$$

where r refers to rth element of $\mathbf{a}_j$, and $\{n_1, n_2\} \geq 1$ are the norms to be chosen. In the current work, we choose $n_1 = \infty$ and $n_2 = 2$. The architecture of WPNN with weights included between pattern and summation layer is illustrated in Fig. 1.

[1] In literature, p letter is used instead of n to denote the p–norm or L^p. However, in this work, p stands for the pattern index. Lower case n removes ambiguity.

Table 1
The UCI–MLR data sets used to test all compared models.

Data set	Records	Attributes	Classes	Class distribution
Breast cancer	683	9	2	444–239
Statlog heart	270	13	2	150–120
Diabetes	786	8	2	500–268
Ecoli	327	5	5	143–77–35–20–52
Parkinson	195	22	2	147–48
Iris	150	4	3	50–50–50
Breast tissue	106	9	6	21–15–18–16–14–22
Monk	432	6	2	216–216
Seeds	210	7	3	70–70–70
Cardiotocography	2126	22	3	1655–295–176

5. Computational experiments

In this section, we present the classification results obtained by the proposed WPNN model, MPNN – the modified probabilistic neural network presented in [23–25] and the original PNN classifier. The received outcomes are compared with these achieved by support vector machine (SVM), multilayer perceptron (MLP), radial basis function neural network (RBFNN), *k*-nearest neighbor method (kNN) and gene expression programming algorithm (GEP). The the input data sets and selected classifiers used in the simulations are also described.

5.1. Input data sets

The performance of the proposed WPNN, MPNN, original PNN and the state-of-the-art algorithms (SVM, MLP, RBFNN, kNN and GEP) is assessed on UCI–MLR data sets. The following well known and commonly tested databases are utilized: breast cancer, statlog heart, diabetes, ecoli, parkinson, iris, breast tissue, monk, seeds and cardiotocography. Table 1 presents the cardinality, dimensionality, number of classes and the class distributions of each considered data set.

5.2. State-of-the-art classifiers used in the comparison

This subsection shows a brief description of the MPNN model and the selected classification algorithms. We also provide the parameter settings used in the simulations for these classifiers.

5.2.1. Literature's weighted PNN

As in the case of our solution, MPNN has its weighting coefficients introduced between pattern and summation layer of the network. The weights are defined as the ratio of between-class-variance and within-class-variance for the training pattern $\mathbf{x}_j^{(p)}$. This ratio takes a high value for a high class separability among the data. In the case when low class separability is faced, this ratio is low. Therefore, all patterns are weighted based on their class separability. For more details, see [23–25].

5.2.2. SVM

SVM is the binary classification algorithm proposed by Vapnik [42]. Within the training process, SVM solves the quadratic programming (QP) optimization problem. For this purpose, various kernel functions must be explored. In this study, we test the radial basis kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \tag{28}$$

and the polynomial kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left(\alpha\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) + \beta\right)^k. \tag{29}$$

In the SVM training, the capacity control C has an impact on the solution of the QP problem. Therefore, this indicator needs to be appropriately selected. Furthermore, the kernel functions and the kernel parameters must be properly adapted since their values influence the final classification outcome. In the first row of Table 2, we present the settings of the SVM model in this work. All possible setting configurations are utilized.

5.2.3. MLP

MLP is a feedforward neural network trained with the use of a backpropagation procedure [26]. The network consists of an input layer, hidden layers, and an output layer. The number of hidden layers, the number of neurons in hidden layers and the transfer functions must be appropriately selected for MLP. In this work, we test the transfer functions from the set: {linear, logistic, hyperbolic tangent}. A scaled conjugate gradient method is utilized for the MLP training [21]. The second

Table 2
Simulation parameters of the examined state-of-the-art classification algorithms.

SVM	
	Kernel functions: –radial basis kernel (28) –polynomial kernel (29) The grid search is performed for σ, α, β and k Capacity control coefficient: $C = \left\{10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}\right\}$
MLP	
	Training method: scaled conjugate gradient Number of hidden layers: {1, 2} Number of hidden neurons: $\{2, 3, \ldots, 30\}$ Transfer functions: {linear, logistic, hyperbolic tangent}
RBFNN	
	Training method: weighted boosting search Neuron tuning parameters: Size of population: {400, 600, 800, 1000} Maximum generation: {100, 200, 500} Boosting iterations: 50 Number of hidden neurons: $\{2, 3, \ldots, 30\}$
kNN	
	Number of neighbors: $k = \{2, \ldots, K\}$ Distance measures: {Euclidean, Cityblock, Chebychev, Minkowski}
GEP	
	Head size: {2, 3, 4, 5, 6, 7, 8} Number of genes: {1, 2, ..., 15} Linking function: addition, multiplication Computing functions: $+$, $-$, $*$, $/$, $-x$, $1/x$, $\exp(-(x-a)^2/(2b^2))$ Fitness function: number of hits with penalty, mean squared error Genetic operators: set according to [29]

row of Table 2 shows the set of all parameters of MLP. In the present study, the network is simulated using all feasible parameter arrangements.

5.2.4. RBFNN

RBFNN, similar to PNN and MLP, is a feedforward neural network [2]. This model is composed of three layers: an input layer, a single hidden layer and an output layer. The hidden neurons are activated by means of a radial basis function while the output neuron produces the signal after linear activation. The number of the neurons in the hidden layer and the parameters of the RBFNN training method must be suitably selected. An evolutionary approach proposed by Chen et al. called weighted boosting search [3] is applied to train RBFNN. In the third row of Table 2, the utilized parameters of the RBFNN training algorithm are provided along with the number of explored hidden neurons. The network is trained by means of all parameter configurations.

5.2.5. kNN

kNN is a supervised learning algorithm which classifies an unknown sample vector $\mathbf{x}$ to the class represented by the majority the nearest data patterns [5]. The algorithm is based on calculation of k nearest distances between $\mathbf{x}$ and the training vectors $\mathbf{x}^{(p)}$, $p = 1, \ldots, P$ which provide k number of nearest neighbors. A greater number of these neighbors forms the prediction for $\mathbf{x}$. In our study, the distance measures are examined from the following set {Euclidean, Cityblock, Chebychev, Minkowski}. The fourth row of Table 2 reveals all kNN training parameters. K is set depending on the data set cardinality.

5.2.6. GEP

GEP is an emulating biological evolution algorithm which creates and evolves computer programs [8]. In GEP, the chromosomes are represented using linear strings of fixed length. Each chromosome consists of genes structurally organized in the head and the tail. The genes are connected to each other with use of linking functions. The individuals, which are encoded by the chromosomes, create population which undergoes evolution where the predefined genetic operators are applied to compute the fitness. The GEP's parameters are specified in the bottom row of Table 2. In all experiments, the population size is set to 30. Evolution is performed until 10,000 generations are reached.

5.3. Results and discussion

To assess the predictive capabilities of all compared classifiers, a k-fold cross validation (CV) procedure is utilized [11]. In a k-fold CV, the entire data set is randomly partitioned into k mutually exclusive subsets of possibly equal size. The model is trained on $k-1$ subsets and then tested on the subset held out by computing the CV accuracy (*Acc*). *Acc* is defined as the number of correct classifications divided by the test set cardinality. The procedure is repeated for a total of k runs, each

Table 3
10-fold *Acc*, standard deviation and ranking points achieved by WPNN, MPNN, original PNN, SVM, MLP, RBFNN, kNN and GEP in the classification tasks.

Data set	WPNN	MPNN	PNN	SVM	MLP	RBFNN	kNN	GEP
Breast cancer	97.22 0.13 7	96.70 2.15 4	96.92 0.17 5	95.02 0.35 1	97.10 0.18 6	96.38 0.32 3	**97.49** 0.15 8	95.31 0.26 2
Statlog heart	**80.85** 0.70 8	78.89 7.07 4	78.15 0.62 2	78.63 1.37 3	80.70 0.93 7	79.04 2.02 5	72.85 0.41 1	80.63 1.92 6
Diabetes	68.82 0.45 3	67.79 4.61 2	67.74 0.36 1	75.70 0.64 6	**77.29** 0.52 8	75.14 1.16 5	76.39 0.32 7	71.07 1.62 4
Ecoli	**88.91** 1.09 8	82.17 6.23 2	83.19 1.19 3	78.75 0.55 1	85.07 0.98 5	86.73 1.11 6	88.68 0.46 7	83.82 1.64 4
Parkinson	**91.20** 1.69 8	90.36 5.49 6	89.64 1.05 4	91.18 0.94 7	89.85 1.33 5	88.26 1.77 3	86.72 0.63 2	86.15 1.43 1
Iris	95.47 0.40 7	95.40 5.16 6	95.06 0.32 2	95.39 0.46 5	95.33 0.42 4	95.27 0.36 3	**97.86** 0.26 8	92.80 1.65 1
Breast tissue	69.53 2.88 7	68.69 1.19 5	**73.35** 4.13 8	68.87 2.92 6	62.36 2.94 1	65.47 3.11 3	64.25 1.81 2	67.45 2.47 4
Monk	87.61 1.39 4	88.51 6.61 5	86.69 1.55 3	99.21 0.46 7	90.02 3.55 6	75.67 0.92 1	**99.75** 0.37 8	79.70 1.88 2
Seeds	93.34 1.78 4	95.24 5.33 7	95.01 0.71 6	**95.62** 0.69 8	92.19 0.88 1	94.57 0.88 5	92.47 0.28 3	92.43 1.45 2
Cardiotocography	85.37 0.09 1	85.85 2.35 3	85.82 0.09 2	97.39 0.06 7	94.58 0.52 6	**97.42** 0.14 8	93.14 0.17 5	90.51 0.47 4
Total ranking points	57	44	36	51	49	42	51	30

time using different subset for validation. Final prediction accuracy is computed as the mean over particular accuracy scores. In this work, k is set to 10.

In Table 3, we present 10-fold *Acc* (in %) determined for WPNN, MPNN, original PNN and the state-of-the-art classifiers in the classification problems of considered benchmark data sets. For each classification task, the mean of *Acc* is reported along with the standard deviation obtained after ten separate CV simulations. Additionally, as a comparative factor, we provide the ranking points (*RP*) statistics which inform about the order of achieved results for all analyzed classification algorithms. $RP = 8$ indicates a model which yields the highest *Acc* while $RP = 1$ means that a model performs worst.

Analyzing the results in Table 3, we observe that the proposed WPNN and kNN provide the highest *Acc* value in three classification cases. The remaining models achieve the best result only once excluding MPNN and GEP which "do not win the contest at all". The above observation is confirmed by the outcomes shown in the last row of the table described as total ranking points (*TRP*). *TPR* for the proposed WPNN is 57 (the highest value) while for kNN and SVM, $TPR = 51$ (second best results). Such a good outcome of WPNN stems from $RP = \{8, 7\}$ achieved in six out of ten classification tasks. Additionally, WPNN receives $RP = 1$ only for the cardiotocography data set in contrast to SVM, MLP, and GEP for which the lowest *RP* value is reported twice. Of note is the fact that for the original PNN model, $TPR = 36$; thus, the introduction of the weights into the network's structure results in substantial improvement in terms of prediction ability.

It is necessary to point the remark that the proposed introduction of the weights into PNN structure significantly increases its accuracy in seven out of ten classification tasks. For the ecoli and statlog heart data sets this increase amounts to 5.72% and 2.70%, respectively. Only in three cases, WPNN, compared to the original network, performs worse. However, for seeds and cardiotocography databases, the accuracy gap remains acceptable (1.67% and 0.45%, respectively). For breast tissue data set, our result is 3.82% lower than the one for PNN but, in must be emphasized, the original network is the best classifier among all tested models with $Acc = 73.35\%$ in this case.

The other important observation is a higher *Acc* value received by WPNN in comparison with the one of MPNN in seven classification cases. For four data sets (breast cancer, parkinson, iris and breast tissue), the improvement of the proposed method is less than 1%. The advantage of the WPNN over MPNN is especially meaningful in the ecoli database classification problem (6.74%).

Table 4
The parameter values for the state-of-the-art classifiers for which the highest *Acc* is obtained.

Data set	SVM		MLP	RBFNN	kNN	GEP
	C	σ	structure	n	k	g
Breast cancer	10^3	0.5	9–3–2	50	7	4
Statlog heart	10^2	2.5	13–19–2	38	29	9
Diabetes	10^3	0.3	8–7–2	74	21	10
Ecoli	10^4	1.5	5–13–4–5	66	7	8
Parkinson	10^5	0.1	22–7–2	54	6	7
Iris	10^0	1.5	4–5–3	16	12	10
Breast tissue	10^1	27.1	9–3–6	34	3	6
Monk	10^2	0.7	6–14–8–2	3	5	5
Seeds	10^3	0.2	7–8–10–3	41	18	3
Cardiotocography	10^2	2.6	22–13–6–3	100	3	4

It is worth collating the result of MPNN and PNN as well. The MPNN model also outperforms the original network in seven classification cases. However, if we look more closely at the results, we can easily discover that the gain in *Acc* value is not that substantial. It is as follows: (a) for the diabetes and cardiotocography data sets – less than 0.5‰; (b) for the seeds and iris data sets – 2–3‰; (c) for the statlog heart and parkinson data sets – approximately 0.75%. Only in the monk data classification task, 1.82% of the MPNN accuracy increase is noticed. Despite relatively better performance in relation to the original PNN, MPNN results are characterized by problematic values of the standard deviation (*sd*). This parameter constitutes a multiple of the highest *sd* obtained by other methods. For example, MPNN's standard deviation received in the breast cancer and iris data set classification cases is over 6 and 11 times greater, respectively, than the highest *sd* determined by the SVM model. For the statlog heart data set, in turn, standard deviation sets up 10% of the entire *Acc* value.

Some comments need to be made on the outcomes of the state-of-the-art algorithms. As mentioned above, for the kNN classifier the highest accuracy is attained three times (breast cancer, iris and monk data sets) while the remaining models, i.e. SVM, MLP and RBFNN perform best only once. Taking *TRP* into a consideration, kNN, SVM and MLP score respectively 51, 51 and 49 points which constitutes some margin in comparison to the best classifier, i.e. WPNN ($TRP = 57$). In spite of achieving the highest *Acc* value in the classification of the cardiotocography data set, the RBFNN is the third worst among all tested models with $TRP = 42$.

In this work, in order to statistically evaluate the results obtained by all models, the non-parametric Friedman test is used. Based on the ranks of the scores presented in Table 3 for each classifier, we assume null and alternative hypotheses. The null hypothesis of the test is formulated as follows:

$$H_0 : \textit{The methods all have identical effects.}$$

The alternative hypothesis is formulated as follows:

$$H_1 : \textit{The methods do have different effects.}$$

On the basis of the *TRP* in Table 3, the Friedman Q value statistic equals 9.13, while the number of the degrees of freedom is 63. Assuming statistical significance at 0.999 level, the critical test value (*CTV*) is 33.91. The calculated Q statistic is not larger then *CTV*, therefore the alternative hypothesis can be rejected. Thus, the methods investigated for comparison purposes do not have different effects. This outcome of the test implies the consistency of the comparison of the ranking results.

Table 4 shows the parameters of the reference classifiers for which the highest *Acc* is achieved in each classification problem. The reported values are obtained after performing a vast number of simulations by exploring all possible settings' combinations according to Table 2. In the case of the SVM model, the capacity control C and the spread constant σ are shown. This gives us the possibility of discovering that for the radial basis kernel (28), $C = 10^3$ and $\sigma = 0.5$, $Acc = 95.02\%$ is obtained in the classification of the breast cancer data set. Similar information can be read for MLP. The network structure is presented in the form of the string w–x–y–z, where w and z denote the number of input and output neurons, respectively, while x and y stand for the number of neurons in the hidden layers. Thus, for the breast cancer data, $Acc = 97.10\%$ is obtained for MLP composed of 9 input neurons, 3 neurons in the hidden layer and 2 neurons in the output layer. In the case of the RBFNN and kNN classifiers, n and k stand for the number of neurons in the network's hidden layer and the number of k nearest neighbors, respectively. Finally, for the GEP algorithm, the letter g denotes the number of genes in a chromosome at which the highest accuracy is attained for this classifier. The DTREG software [29] and the Matlab Statistics Toolbox are used in simulations. The presented values of model's parameter for which the highest accuracy is obtained give the possibility of creating and testing the classifiers by other researchers.

As presented in Section 5.1, data sets of a diversified cardinality (P), dimensionality (N) and the number of classes (J) are used in this study. However, in order to verify the scalability of the proposed WPNN, further research is performed. For this purpose, the data sets with higher number of input patterns, features and classes are utilized in the simulations.

In particular, we use the following UCI–MLR data sets: the single proton emission tomography images ($P = 267,\ N = 44,\ J = 2$), the robot navigation ultrasound sensors readings ($P = 5456,\ N = 24,\ J = 4$), and the urban land cover aerial images ($P = 168,\ N = 148,\ J = 9$). WPNN tested on these data sets converges to final solution as original PNN model. The solution time strongly depends on P, N and J of each data set but this is consistent with the nature of complex classification tasks. The increase of a computational time is also observed in the case of the applied state-of-the-art algorithms.

6. Conclusions

In this paper, the weighted probabilistic neural network was proposed. The original PNN model was enriched by the weights introduced between pattern and summation layer of the network. The analytical formula for the weighting coefficients was provided with the use of the SA procedure. A 10-fold cross validation accuracy of WPNN was compared with the accuracy of the modified PNN available in literature, the traditional PNN and the state-of-the-art algorithms in ten benchmark data classification problems. Also, the ranking points statistics were calculated along with Friedman test which confirmed the consistency of the comparison. The elements of novelty of this study are as follows:

- The strict analytical formula for the weights of PNN has not been provided up to this date. In this work, we propose such a formula. It stems from the SA procedure, therefore is interpretable and easy to implement.
- Due to introduced weights, neither heuristic nor approximate but precise results are obtained.
- SA procedure has not been utilized for computing the PNN's weights so far.
- The computation of weights is strictly embedded in the operation of PNN. Therefore, no additional iterative algorithm which relies on the minimization of assumed error function is required.
- From the effectiveness point of view, the proposed WPNN is not predominant in comparison to other approaches. However, our greatest attention in this study is paid to the fact of improving the prediction ability of WPNN with respect to the modified model proposed in literature and original network. This goal is achieved in majority of classification problems since WPNN attains higher accuracy than MPNN original PNN in seven out of ten considered tasks. Moreover, in single data classification case, WPNN's accuracy is only less than 0.5% lower in contrast to other PNN based methods. This outcome makes here all the networks almost identical in terms of performance. Furthermore, if we consider the total ranking points as the comparative statistics, WPNN stacks up much better against the other models.

In a future work, we will focus on providing an approach for the PNN's weights update within the training process. Similar solution is shown for example in [32] where a backpropagation-like technique is applied to train a normalizing neural network. It will also be interesting to compare our model with neural networks which constitute a synthesis of the probabilistic rule based classifiers [33]. Finally it is worth to explore the performance of the proposed network in compound problems [31,39].

Acknowledgment

The work was supported by Rzeszow University of Technology, Department of Electronics Fundamentals Grant for Statutory Activity (DS.EP.17.001).

References

[1] H. Adeli, A. Panakkat, A probabilistic neural network for earthquake magnitude prediction, Neural Netw. 22 (7) (2009) 1018–1024.
[2] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Syst. 2 (1988) 321–355.
[3] S. Chen, X. Wang, C.J. Harris, Experiments with repeating weighted boosting search for optimization signal processing applications, Syst. Man Cybern. Part B 35 (4) (2005) 682–693.
[4] Y. Chtioui, S. Panigrahi, R. Marsh, Conjugate gradient and approximate newton methods for an optimal probabilistic neural network for food color classification, Opt. Eng. 37 (11) (1998) 3015–3023.
[5] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[6] P. Duda, M. Jaworski, L. Rutkowski, Knowledge discovery in data streams with the orthogonal series-based generalized regression neural networks, Inf. Sci. (2017).
[7] K. Elenius, H.G. Tråvén, Multi-layer perceptrons and probabilistic neural networks for phoneme recognition., EUROSPEECH, 1993.
[8] C. Ferreira, Gene expression programming: a new adaptive algorithm for solving problems, Complex Syst. 13 (2) (2001) 87–129.
[9] R. Folland, E. Hines, R. Dutta, P. Boilot, D. Morgan, Comparison of neural network predictors in the classification of tracheal–bronchial breath sounds by respiratory auscultation, Artif. Intell. Med. 31 (3) (2004) 211–220.
[10] J.Y. Goulermas, P. Liatsis, X.-J. Zeng, P. Cook, Density-driven generalized regression neural networks (dd-grnn) for function approximation, IEEE Trans. Neural Netw. 18 (6) (2007) 1683–1696.
[11] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: IJCAI, 14, Stanford, CA, 1995, pp. 1137–1145.
[12] P.A. Kowalski, P. Kulczycki, Data sample reduction for classification of interval information using neural network sensitivity analysis, in: D. Dicheva, D. Dochev (Eds.), Artificial Intelligence: Methodology, Systems, and Applications, Lecture Notes in Computer Science, vol. 6304, Springer Berlin Heidelberg, 2010, pp. 271–272.
[13] P.A. Kowalski, P. Kulczycki, A complete algorithm for the reduction of pattern data in the classification of interval information, Int. J. Comput. Methods 13 (03) (2016) 1650018–1–1650018–26, doi:10.1142/S0219876216500183.
[14] P.A. Kowalski, P. Kulczycki, Interval probabilistic neural network, Neural Comput. Appl. 28 (4) (2017) 817–834, doi:10.1007/s00521-015-2109-3.
[15] M. Kusy, J. Kluska, Assessment of prediction ability for reduced probabilistic neural network in data classification problem, Soft. Comput. 21 (1) (2017) 199–212.
[16] M. Kusy, P.A. Kowalski, Modification of the probabilistic neural network with the use of sensitivity analysis procedure, in: Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on, IEEE, 2016, pp. 97–103.

[17] M. Kusy, R. Zajdel, Probabilistic neural network training procedure based on q(0)–learning algorithm in medical data classification, Appl. Intell. 41 (3) (2014) 837–854.
[18] M. Kusy, R. Zajdel, Application of reinforcement learning algorithms for the adaptive computation of the smoothing parameter for probabilistic neural network, Neural Netw. Learn. Syst. IEEE Trans. 26 (9) (2015) 2163–2175.
[19] M. Lichman, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2013 https://archive.ics.uci.edu/ml.
[20] D. Mantzaris, G. Anastassopoulos, A. Adamopoulos, Genetic algorithm pruning of probabilistic neural networks in medical disease estimation, Neural Netw. 24 (8) (2011) 831–835.
[21] M.F. Møller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Netw. 6 (4) (1993) 525–533.
[22] D. Montana, A weighted probabilistic neural network, in: NIPS, 1991, pp. 1110–1117.
[23] D. Nanjundappan, et al., Hybrid weighted probabilistic neural network and biogeography based optimization for dynamic economic dispatch of integrated multiple-fuel and wind power plants, Int. J. Electr. Power Energy Syst. 77 (2016) 385–394.
[24] S. Ramakrishnan, M. Emary Ibrahiem, Comparative study between traditional and modified probabilistic neural networks, Telecommun. Syst. 40 (1–2) (2009) 67–74, doi:10.1007/s11235-008-9138-5.
[25] S. Ramakrishnan, S. Selvan, Image texture classification using wavelet based curve fitting and probabilistic neural network, Int. J. Imag. Syst. Technol. 17 (4) (2007) 266–275.
[26] D.E. Rumelhart, J.L. McClelland, C. PDP Research Group (eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations, MIT Press, Cambridge, MA, USA, 1986.
[27] L. Rutkowski, Adaptive probabilistic neural networks for pattern classification in time-varying environment, Neural Netw. IEEE Trans. 15 (4) (2004) 811–827.
[28] L. Rutkowski, New Soft Computing Techniques for System Modelling, Pattern Classification and Image Processing, Springer-Verlag, 2004.
[29] P.H. Sherrod, Dtreg predictive modelling software, http://www.dtreg.com.
[30] B.W. Silverman, Density Estimation for Statistics and Data Analysis, 26, CRC Press, 1986.
[31] D. Ślezak, M. Szczuka, Rough Neural Networks for Complex Concepts, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 574–582. 10.1007/978-3-540-72530-5_69.
[32] D. Ślezak, J. Wróblewski, M. Szczuka, Constructing Extensions of Bayesian Classifiers with Use of Normalizing Neural Networks, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 408–416. doi:10.1007/978-3-540-39592-8_57.
[33] D. Ślezak, J. Wrblewski, M. Szczuka, Neural network architecture for synthesis of the probabilistic rule based classifiers, Electron Notes Theor. Comput. Sci. 82 (4) (2003) 251–262, doi:10.1016/S1571-0661(04)80723-0.
[34] T. Song, C. Gasparovic, N. Andreasen, J. Bockholt, M. Jamshidi, R.R. Lee, M. Huang, A hybrid tissue segmentation approach for brain mr images, Med. Biol. Eng. Comput. 44 (3) (2006) 242–249, doi:10.1007/s11517-005-0021-1.
[35] T. Song, M. Jamshidi, R.R. Lee, M. Huang, A novel weighted probabilistic neural network for mr image segmentation, in: Systems, Man and Cybernetics, 2005 IEEE International Conference on, vol. 3, IEEE, 2005, pp. 2501–2506.
[36] D.F. Specht, Probabilistic neural networks, Neural Netw. 3 (1) (1990) 109–118.
[37] D.F. Specht, Probabilistic neural networks and the polynomial adaline as complementary techniques for classification, Neural Netw. IEEE Trans. 1 (1) (1990) 111–121, doi:10.1109/72.80210.
[38] D.F. Specht, A general regression neural network, IEEE Trans. Neural Netw. 2 (6) (1991) 568–576.
[39] M. Szczuka, D. Ślezak, Feedforward neural networks for compound signals, Theor. Comput. Sci. 412 (42) (2011) 5960–5973, doi:10.1016/j.tcs.2011.05.046.
[40] T.P. Tran, L. Cao, D. Tran, C.D. Nguyen, Novel intrusion detection using probabilistic neural network and adaptive boosting, Int. J. Comput. Sci. Inf. Security 6 (1) (2009) 83–91.
[41] T.P. Tran, T.T.S. Nguyen, P. Tsai, X. Kong, Bspnn: boosted subspace probabilistic neural network for email security, Artif. Intell. Rev. 35 (4) (2011) 369–382.
[42] V. Vapnik, The Nature of Statistical Learning Theory, Springer Science & Business Media, 2013.
[43] S. Venkatesh, S. Gopal, Orthogonal least square center selection technique–a robust scheme for multiple source partial discharge pattern recognition using radial basis probabilistic neural network, Expert Syst. Appl. 38 (7) (2011) 8978–8989.
[44] M.P. Wand, M.C. Jones, Kernel Smoothing, CRC Press, 1994.
[45] X.-B. Wen, H. Zhang, X.-Q. Xu, J.-J. Quan, A new watermarking approach based on probabilistic neural network in wavelet domain, Soft Comput. 13 (4) (2009) 355–360.
[46] F. Zhou, J. Liu, Y. Yu, X. Tian, H. Liu, Y. Hao, S. Zhang, W. Chen, J. Dai, X. Zheng, Field-programmable gate array implementation of a probabilistic neural network for motor cortical decoding in rats, J. Neurosci. Methods 185 (2) (2010) 299–306.
[47] J. Zurada, A. Malinowski, I. Cloete, Sensitivity analysis for minimization of input data dimension for feedforward neural network, in: Circuits and Systems, 1994. ISCAS '94., 1994 IEEE International Symposium on, vol. 6, 1994, pp. 447–450.
[48] J.M. Zurada, A. Malinowski, S. Usui, Perturbation method for deleting redundant inputs of perceptron networks, Neurocomputing 14 (2) (1997) 177–193.