Proceedings of the Federated Conference on Computer Science and Information Systems pp. 97–103

DOI: 10.15439/2016F280
ACSIS, Vol. 8. ISSN 2300-5963

Modification of the Probabilistic Neural Network with the Use of Sensitivity Analysis Procedure

Maciej Kusy[*]
[*]Faculty of Electrical and Computer Engineering,
Rzeszow University of Technology,
al. Powstancow Warszawy 12, 35-959 Rzeszow, Poland,
Email: mkusy@prz.edu.pl

Piotr A. Kowalski[†‡]
[†]Faculty of Physics and Applied Computer Science,
AGH University of Science and Technology,
al. A. Mickiewicza 30, 30-059 Cracow, Poland,
Email: pkowal@agh.edu.pl
[‡]Systems Research Institute,
Polish Academy of Sciences,
ul. Newelska 6, 01-447 Warsaw, Poland,
E-mail: pakowal@ibspan.waw.pl

***Abstract*—In this article, the modified probabilistic neural network (MPNN) is proposed. The network is an extension of conventional PNN with the weight coefficients introduced between pattern and summation layer of the model. These weights are calculated by using the sensitivity analysis (SA) procedure. MPNN is employed to the classification tasks and its performance is assessed on the basis of prediction accuracy. The effectiveness of MPNN is also verified by analyzing its results with these obtained for both the original PNN and commonly known classification algorithms: support vector machine, multilayer perceptron, radial basis function network and k-Means clustering procedure. It is shown that the proposed modification improves the prediction ability of the PNN classifier.**

I. Introduction

PROBABILISTIC neural network (PNN) is a data classifier proposed by Specht in [1] and [2]. It attracts researchers from the field of machine learning. In literature, one can find PNN's applications mainly in medical diagnosis and prediction [3], [4], [5], [6] and image classification and recognition [7], [8], [9]. However, a very good classification performance allows PNN to be utilized in other tasks, e.g.: earthquake magnitude prediction [10], multiple partial discharge sources classification [11], interval information processing [12], [13], phoneme recognition [14], email security enhancement [15] or intrusion detection systems [16].

In its conventional form, PNN is a multilayered feedforward network composed of four layers: input layer (represented by data features), pattern layer (consisting of as many neurons as training patterns), summation layer (with one neuron for each class) and output layer where a single neuron produces a classification result. In majority of cases, the connections between layers in PNN are not equipped with weight coefficients (or all the weights are equally set to 1). Therefore, the output response is not influenced by an impact of a particular class.

In literature, a subtle attention has been paid to determining the weights of PNN. The work of [17] is the first contribution where the weights are introduced to PNN. However, the coefficients are not computed directly; the network operates by using anisotropic Gaussians, i.e. the covariance matrix is utilized, instead of a single smoothing parameter, to compute the output for a particular class. In [18] and [19], the weighted PNN is proposed in the way it adds weighting factors between pattern and summation layer. These factors are calculated from soft labeling probability matrix to carry out a classification. The authors of [20] and [21] create weighted PNN based on their class separability. A single weight is defined as the ratio of 'between-class variance' and 'within-class variance' for a particular training pattern. As in [18], the weighting coefficients are used to connect the pattern and summation layer.

In this study, we propose the use the SA procedure in the computation of the weights for the PNN model. Similar to [18], [19], [20] and [21], the coefficients are inserted between pattern and summation layer. Their values are equal to the aggregated sensitivities normalized to $[0, 1]$ interval. The formulas for the weights are analytically derived. The idea is applied to PNN activated with a product Cauchy kernel. The proposed MPNN is tested in the classification problems of University of California, Irvine machine learning repository (UCI-MLR) data sets [22] by computing a 10-fold cross validation accuracy. The obtained outcomes are thoroughly compared to the ones obtained for original PNN. Furthermore, we verify MPNN accuracy with the accuracy of four reference classifiers: support vector machine, multilayer perceptron, radial basis function neural network and k-Means clustering algorithm.

This work is structured as follows. In section II, the SA procedure is highlighted. Section III, presents the fundamentals of the PNN model. In section IV, the procedure of computing the weights for PNN is proposed. Section V describes the input data sets and the reference classifiers used in current study. Here, the discussion regarding the results obtained in the simulations is also presented. Finally, section VI concludes the work.

This work was supported in part by the Rzeszow University of Technology under Grant No. U-596/DS

II. Sensitivity analysis

SA, in general, is one of many approaches used in determining the importance of particular inputs of a neural network. Therefore, it can be applied in the task of elimination of the irrelevant features in the input vectors. The main idea of SA is based on computing the influence of input features on a neural network output signal after a training process. This influence is characterized by real coefficients [23]

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

where x_i denotes an input feature and y_j stands for an output signal. In (1), $i = 1, \ldots, N$, $j = 1, \ldots, J$, where N and J indicate the number of features and outputs, respectively.

More specifically, equation (1) represents the sensitivity of the jth neural network output on the ith feature of the input vector $\mathbf{x}$ determined based on the pth training pattern $\mathbf{x}^{(p)}$ for $p = 1, \ldots, P$ where P is a data set cardinality. Taking into account N dimensional data set, (1) can be presented as the following matrix

$$\mathbf{S}^{(p)} = \begin{bmatrix} S_{1,1}^{(p)} & S_{1,2}^{(p)} & \cdots & S_{1,N}^{(p)} \\ S_{2,1}^{(p)} & S_{2,2}^{(p)} & \cdots & S_{2,N}^{(p)} \\ \vdots & \vdots & \ddots & \vdots \\ S_{J,1}^{(p)} & S_{J,2}^{(p)} & \cdots & S_{J,N}^{(p)} \end{bmatrix}. \qquad (2)$$

Once $\mathbf{S}^{(p)}$ is computed for all P training patterns, it is possible to find aggregated parameters after application of various types of norms. In the research, one usually utilizes the parameter of the mean square average sensitivity

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

The absolute value average sensitivity and the maximum sensitivity parameters are also frequently applied in input significance estimation [24]. The appropriate impact of $S_{j,i}^{(p)}$ on the aggregated outcome of $S_{j,i}$ implies the selection of a particular norm.

III. Probabilistic neural network

PNN is composed of four layers. The coordinates of an input vector $\mathbf{x} = [x_1, \ldots, x_N]$ constitute the first input layer. The second layer, called a pattern layer, consists of as many neurons as training examples. Pattern neurons feed their output to the next summation layer. In the summation layer, there are J neurons, however each jth neuron sums the inputs from the neurons of jth class. In literature, two approaches are usually utilized to compute the signals of the summation neurons: the additive Gaussian kernels and the product kernels. In this work, we use the second approach, therefore the summation neuron output is defined as follows

$$f_j(\mathbf{x}) = \frac{1}{P_j \det(\mathbf{h})} \sum_{p=1}^{P_j} \frac{1}{s_p^N} K \left(\frac{\left(\mathbf{x} - \mathbf{x}_j^{(p)} \right)^T \mathbf{h}^{-1}}{s_p} \right), \qquad (4)$$

where:

- $K(\cdot)$ is the kernel function calculated in the following way
$$K(\mathbf{x}) = \mathcal{K}(x_1) \cdot \mathcal{K}(x_2) \cdot \ldots \cdot \mathcal{K}(x_N), \qquad (5)$$
for which
$$\mathcal{K}(x_i) = \frac{2}{\pi (x_i^2 + 1)^2} \qquad (6)$$
denotes the one-dimensional Cauchy multiplicand;
- P_j stands for the number of cases in the jth class ($j = 1, \ldots, J$);
- $\mathbf{x}_j^{(p)} = [x_{j,1}^{(p)}, \ldots, x_{j,N}^{(p)}]$ is the pth training vector of the jth class.

If one regards (5) and (6) as the pattern neuron activation function, the summation layer output for the jth class is determined as follows

$$f_j(\mathbf{x}) = \frac{1}{P_j \det(\mathbf{h})} \sum_{p=1}^{P_j} \frac{1}{s_p^N} \prod_{i=1}^{N} \frac{2}{\pi \left(\left(\frac{x_i - x_{j,i}^{(p)}}{h_i s_p} \right)^2 + 1 \right)^2}. \qquad (7)$$

Finally, using the Bayes theorem [2], the output layer of PNN determines the label for a new test vector $\mathbf{x}$

$$C(\mathbf{x}) = \underset{j=1\ldots J}{\text{argmax}}\, f_j(\mathbf{x}), \qquad (8)$$

where $C(\mathbf{x})$ denotes the predicted class. The training algorithm for this network amounts to the appropriate choice of the smoothing parameter h_i by means of the plug-in method [25], and the computation of the modification coefficient s_p [26]. The structure of the PNN model is illustrated in Fig. 1.

IV. Proposed algorithm

In Fig. 2, we present the step-by-step data classification algorithm with the use of modified PNN model. We start with the calculation of the sensitivity coefficients for all $r = 1, 2, \ldots, P_j$ neurons in the pattern layer

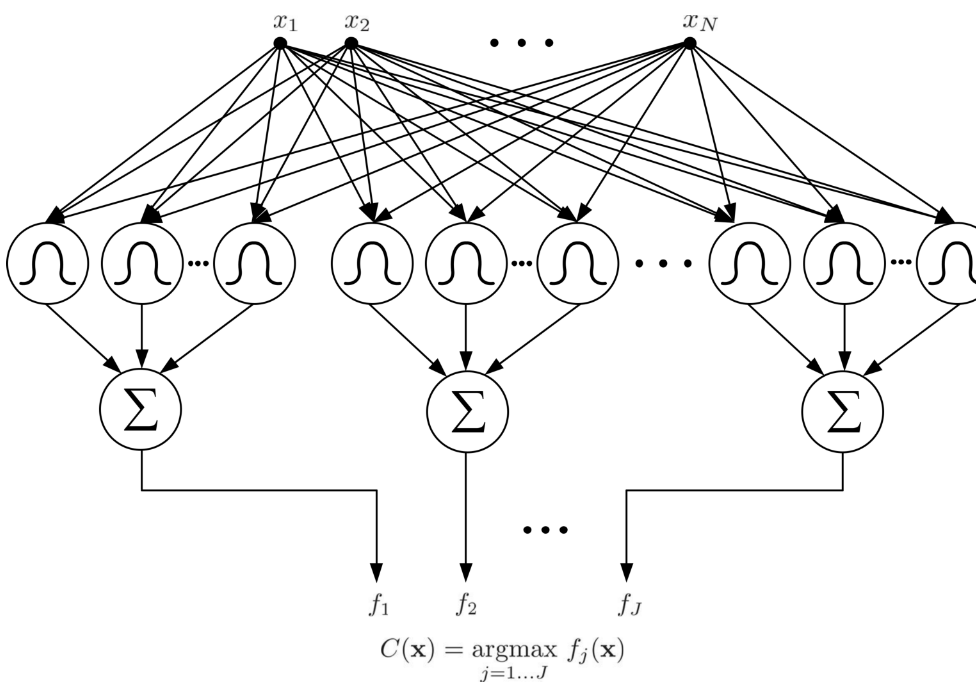
$$S = \frac{\partial \hat{f}_j \left(\mathbf{x}_j^{(p)} \right)}{\partial \mathbf{x}_j^{(r)}}, \qquad (9)$$

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

$$\mathbf{S}_j = \left\{ \frac{\partial \hat{f}_j \left(\mathbf{x}_j^{(p)} \right)}{\partial \mathbf{x}_j^{(r)}} \right\}_{P_j \times P_j}. \qquad (10)$$

In (10), 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

$$\nabla \hat{f}_j^{(p,r)} = \frac{\partial \hat{f}_j \left(\mathbf{x}_j^{(p)} \right)}{\partial \mathbf{x}_j^{(r)}} = \left[\frac{\partial \hat{f}_j \left(\mathbf{x}_j^{(p)} \right)}{\partial x_{j,1}^{(r)}}, \ldots, \frac{\partial \hat{f}_j \left(\mathbf{x}_j^{(p)} \right)}{\partial x_{j,N}^{(r)}} \right], \qquad (11)$$



x_1
x_2
x_N
f_1
f_2
f_J
$C(\mathbf{x}) = \underset{j=1\ldots J}{\operatorname{argmax}} f_j(\mathbf{x})$

Fig. 1. The architecture of probabilistic neural network.

where

$$\frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial x_{j,i}^{(r)}} = \frac{1}{P_j \det(\mathbf{h})}\frac{1}{s_r^N}\frac{\partial}{\partial x_{j,i}^{(r)}} K\left(\frac{\left(\mathbf{x}_j^{(p)} - \mathbf{x}_j^{(r)}\right)^T \mathbf{h}^{-1}}{s_r}\right). \quad (12)$$

The product form of KDE in (5) expands (12) into the following formula

$$\begin{array}{c}\frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial x_{j,i}^{(r)}} = \frac{1}{P_j \det(\mathbf{h})}\frac{1}{s_r^N}\mathcal{K}\left(\frac{x_{j,1}^{(p)} - x_{j,1}^{(r)}}{h_1 s_r}\right)\ldots \\ \frac{\partial}{\partial x_{j,i}^{(r)}}\mathcal{K}^\star\left(\frac{x_{j,i}^{(p)} - x_{j,i}^{(r)}}{h_i s_r}\right)\ldots\mathcal{K}\left(\frac{x_{j,N}^{(p)} - x_{j,N}^{(r)}}{h_N s_r}\right)\end{array}. \quad (13)$$

The Cauchy kernel in (6) allows us to determine the ith coefficient of (13)

$$\frac{\partial}{\partial x_{j,i}^{(r)}}\mathcal{K}^\star\left(x_{j,i}^{(p)}\right) = \frac{8\left(x_{j,i}^{(p)} - x_{j,i}^{(r)}\right)}{\pi h_i^2 s_r^2\left(\left(\frac{x_{j,i}^{(p)} - x_{j,i}^{(r)}}{h_i s_r}\right)^2 + 1\right)^3}. \quad (14)$$

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 jth class for a given rth pattern neuron, it is necessary to determine the norm of (11). Thus, $\mathbf{S}_j$ in (10) must be generalized to the following matrix

$$\mathbf{S}_j = \left\{\|\nabla \hat{f}_j^{(p,r)}\|\right\}_{P_j \times P_j}, \quad (15)$$

where

$$\|\nabla \hat{f}_j^{(p,r)}\| = \sqrt{\sum_{i=1}^{N}\left(\frac{\partial \hat{f}_j\left(\mathbf{x}_j^{(p)}\right)}{\partial x_{j,i}^{(r)}}\right)^2} \quad (16)$$

for the Euclidean norm. 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 $\mathbf{S}_j$ to obtain the aggregated sensitivity vector. For the mean square average sensitivity measure, this vector takes the following form

$$\mathbf{a}_j = \left[\sqrt{\frac{\sum_{p=1}^{P_j}\left(S_j^{(p,1)}\right)^2}{P_j}}, \ldots, \sqrt{\frac{\sum_{p=1}^{P_j}\left(S_j^{(p,P_j)}\right)^2}{P_j}}\right]. \quad (17)$$

Finally, the normalization of the elements of $\mathbf{a}_j$ to some interval introduces the weight vector for each jth class. In this work, we propose simple "max" normalization

$$\mathbf{w}_j = \left[\frac{a_j^{(1)}}{\max(\mathbf{a}_j)}, \frac{a_j^{(2)}}{\max(\mathbf{a}_j)}, \ldots, \frac{a_j^{(P_j)}}{\max(\mathbf{a}_j)}\right], \quad (18)$$

where $\max(\cdot)$ operator returns the maximum value of the argument.

Including the weights' coefficients, the summation layer output (7) is redefined as

$$f_j(\mathbf{x}) = \frac{1}{P_j \det(\mathbf{h})}\sum_{\{p,r\}=1}^{P_j} w_j^{(r)}\frac{1}{s_p^N}K\left(\frac{\left(\mathbf{x} - \mathbf{x}_j^{(p)}\right)^T \mathbf{h}^{-1}}{s_p}\right), \quad (19)$$

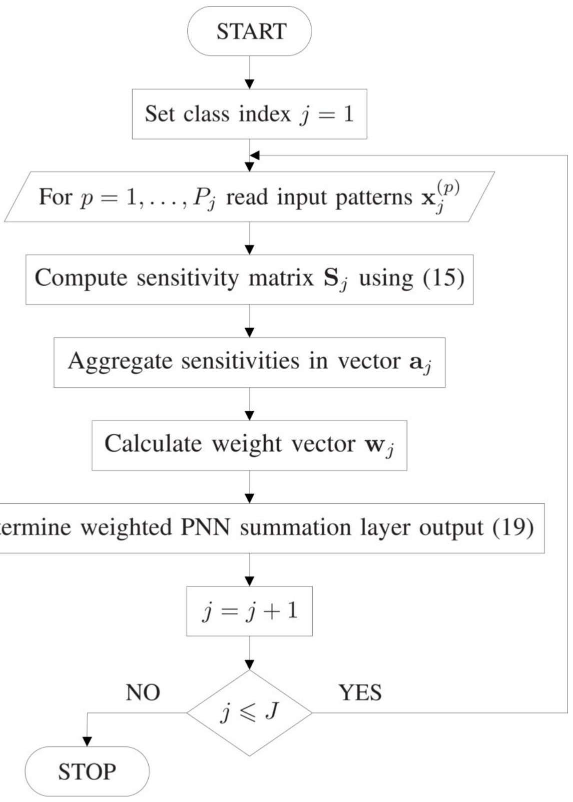

START
Set class index $j = 1$
For $p = 1, \ldots, P_j$ read input patterns $\mathbf{x}_j^{(p)}$
Compute sensitivity matrix $\mathbf{S}_j$ using (15)
Aggregate sensitivities in vector $\mathbf{a}_j$
Calculate weight vector $\mathbf{w}_j$
Determine weighted PNN summation layer output (19)
$j = j + 1$
$j \leqslant J$
NO
YES
STOP

Fig. 2. The algorithm for the computation of the weights in PNN between pattern and summation layer.

where

$$w_j^{(r)} = \frac{\sqrt{\frac{1}{P_j}\sum_{p=1}^{P_j}\left(S_j^{(p,r)}\right)^2}}{\max(\mathbf{a}_j)}, \quad (20)$$

where r refers to rth element of $\mathbf{a}_j$.

V. EXPERIMENTS

In this section, we present the classification results obtained by PNN with introduced weights and the original network. These results are compared with the outcomes achieved by other classifiers: support vector machine (SVM) algorithm, multilayer perceptron (MLP), radial basis function neural network (RBFN) and k-Means method. Both, input data sets and the reference classifiers used in the simulations are also described.

A. Input data sets

The performance of the proposed MPNN, original PNN and the reference methods (SVM, MLP, RBFN and k-Means) is evaluated on UCI-MLR data sets. Ten well known and commonly tested databases are included: Wisconsin breast cancer (WBC), Statlog heart (SH), Pima Indians diabetes (PID), Ecoli (E), Parkinsons (P), Iris (I), breast tissue (BT), monk (M), seeds (S) and cardiotocography (CTG). The cardinality, dimensionality, number of classes and the class distributions for the utilized databases are presented in Table I.

B. Reference methods

In this subsection, the reference classification models are highlighted. Additionally, we provide the parameter settings used for these classifiers in the simulations.

1) SVM: SVM is the classification algorithm proposed by Vapnik [27]. To perform the classification, SVM requires the solution of the quadratic programming optimization problem. For this purpose, various kernel functions need to be explored. In the current study, we verify the following ones:

- radial basis kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \quad (21)$$

- polynomial kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left(\alpha\left(\mathbf{x}_i \cdot \mathbf{x}_j\right) + \beta\right)^k. \quad (22)$$

In the classification tasks, the parameters of the kernels (21) and (22) and capacity control parameter C must be appropriately selected.

2) MLP: MLP is a feedforward neural network [28]. This network is composed of an input layer, hidden layers, and an output layer. The number of hidden layers, the optimal number of neurons in hidden layers and the appropriate activation functions must be determined for this model. In this work, the following functions are tested:

- linear identity

$$f(x) = x, \quad (23)$$

TABLE I
UCI-MLR DATA SETS USED TO TEST ALL COMPARED MODELS

Data set	Records	Attributes	Classes	Class distribution
WBC	683	9	2	444–239
SH	270	13	2	150–120
PID	786	8	2	500–268
E	327	5	5	143–77–35–20–52
P	195	22	2	147–48
I	150	4	3	50–50–50
BT	106	9	6	21–15–18–16–14–22
M	432	6	2	216–216
S	210	7	3	70–70–70
CTG	2126	22	3	1655–295–176

- logistic sigmoid

$$f(x) = \frac{1}{1+e^{-\alpha x}}, \tag{24}$$

- hyperbolic tangent

$$f(x) = \frac{2}{1+e^{-\beta x}} - 1, \tag{25}$$

where α and β are the coefficients used to control the slope of (24) and (25).

3) RBFNN: RBFNN, similar to PNN and MLP, is a feed-forward neural network [29]. However, this model consists of three layers: an input layer, a radial basis hidden layer and a linear output layer. The number of the neurons in the hidden layer and the parameters of the RBFNN training method must be appropriately selected.

4) k-Means: The k-Means clustering is an unsupervised learning algorithm. It partitions input data into k clusters and provides a center of each cluster [30]. As a result, the records within each cluster are similar to each other and distinct from records in other clusters. The predictions for the unknown cases are made by assigning them the category of the nearest cluster center. The parameter k is increased up to K, which depends on the number of input vectors of a given class (P_j). In the current study, K does not exceed 50% of P_j, $j = 1, \ldots, J$. The step for k is determined empirically.

The training methods and the parameters of the models are presented in Table II.

TABLE II
SIMULATION PARAMETERS OF THE EXAMINED REFERENCE CLASSIFIERS

SVM	kernel functions: – radial basis kernel (21) – polynomial kernel (22) The grid search is performed for σ, α, β and k Capacity control coefficient: $C = \{10^{-1}, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$
MLP	training method: scaled conjugate gradients number of hidden layers: $\{1, 2\}$ number of hidden neurons: $\{2, 3, \ldots, 30\}$ activation functions: – linear (23) – logistic (24) – tangent (25)
RBFN	training method: weighted boosting search [31] neuron tuning parameters: size of population: $\{400, 600, 800, 1000\}$ maximum generation: $\{100, 200, 500\}$ boosting iterations: 50 number of hidden neurons: $\{2, 3, \ldots, 30\}$
k-Means	number of clusters: $\{2, \ldots, K\}$ distance measure: Euclidean distance

C. Results and discussion

In Table III, we present the accuracy (*Acc*) determined for MPNN, PNN and the reference methods in the classification problems of WBC, SH, PID, E, P, I, BT, M, S and CTG data sets. The accuracy is computed with the use of a 10-fold cross validation procedure. Table IV shows the optimal parameters of the reference classifiers for which the highest accuracy is achieved in particular classification problems. In the case of the SVM model, the capacity control parameter C and the spread constant σ are shown for the radial basis kernel (21) since for this function, a higher accuracy values are obtained in contrast to the results achieved with the use of kernel presented in (22). For MLP, the network structure is presented in the form *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. The abbreviations "lin" and "log" refer to linear (23) and logistic (24) activation functions, respectively. For example, in the case of E data classification case, the string "5–13–4–5 log–log–log" describes MLP with 5 input neurons, 13 neurons in the first hidden layer, 4 neurons in the second hidden layer and 5 output neurons where the logistic activation function is applied in hidden and output layers. In the case of the RBFN and k-Means classifiers, n and k denote the number of neurons in the network's hidden layer and the number of cluster centers, respectively. The optimal parameters are obtained after vast number of simulations performed in DTREG software [32].

Comparing two first columns of Table III, one can definitely emphasize the fact that in almost all classification problems, the introduction of additional weight parameters to PNN

TABLE III
CROSS VALIDATION ACCURACY FOR: WEIGHTED PNN, ORIGINAL PNN AND THE REFERENCE CLASSIFIERS: SVM, MLP, RBFN AND K-MEANS IN UCI-MLR DATA CLASSIFICATION TASKS

Data set	MPNN	PNN	SVM	MLP	RBFN	k-Means
WBC	**0.9779**	0.9706	0.9546	0.9722	0.9663	0.9517
SH	**0.8148**	0.7963	0.8074	0.7926	0.7926	0.6852
PID	0.6948	0.6842	0.7474	**0.7669**	0.7435	0.6914
E	**0.8485**	0.8333	0.7982	0.8379	0.8471	0.8410
P	**0.9250**	0.8750	0.9231	0.9128	0.9026	0.8308
I	**1.000**	0.9667	0.9733	0.9733	0.9533	0.9667
BT	**0.7273**	0.7098	0.6792	0.6038	0.7075	0.6038
M	0.8605	0.8408	**0.9884**	0.9236	0.7500	0.9120
S	**0.9524**	**0.9524**	0.9429	0.9286	0.9476	0.9143
CTG	0.8568	0.8545	**0.9751**	0.9417	0.9694	0.8664

TABLE IV
THE PARAMETER VALUES FOR WHICH THE HIGHEST *Acc* IS OBTAINED FOR THE REFERENCE CLASSIFIERS

	SVM		MLP		RBFN	k-Menas
Data set	C	σ	structure	act. funct.	n	k
WBC	10^3	0.5	9–3–2	log–lin	50	57
SH	10^2	2.5	13–19–2	log–log	38	4
PID	10^3	0.3	8–7–2	log–lin	74	4
E	10^4	1.5	5–13–4–5	log–log–log	66	2
P	10^5	0.1	22–7–2	log–log	54	33
I	10^0	1.5	4–5–3	log–log	16	4
BT	10^1	27.1	9–3–6	log–log	34	11
M	10^2	0.7	6–14–8–2	log–log–lin	3	23
S	10^3	0.2	7–8–10–3	log–log–log	41	25
CTG	10^2	2.6	22–13–6–3	log–log–log	100	171

results in significant increase of its accuracy. Note, that in P data classification case, this increase is over 5% which is the highest investigated result. The above observation does not take place in S data set classification task. Here, the proposed modification of neural structure does not change obtained accuracy. Thus, the use of weights for original PNN is not beneficial in terms of prediction ability.

In general, one can note that only in three data set cases, i.e.: PID, M and CTG, the reference methods provide higher *Acc* value. However, among these cases, there is no unequivocal alternative classifier since SVM and MLP yield higher accuracy twice and once, respectively.

If we do not take the MPNN accuracy results into account, it is clear that the original PNN is an average quality classifier because in WBC, SH, E, P and I classification problems, higher *Acc* is obtained by the following methods: MLP, SVM, RBFNN, SVM, and SVM *ex aequo* MLP, respectively.

VI. CONCLUSION

In this paper, the modified probabilistic neural network was proposed. The modification relied on the introduction of the weights' coefficients between pattern and summation layers. The weights for PNN were computed by means of the SA procedure. Their values were equal to the aggregated sensitivities normalized to $[0, 1]$ interval. A PNN with product kernel estimator with Cauchy realization was used. The plug-in method was applied to determine the smoothing parameter. A 10-fold cross validation accuracies of MPNN were compared with the ones obtained for the original PNN and the reference classifiers in the UCI-MLR data sets classification tasks. The results showed, that among all tested models, MPNN achieved the highest *Acc* in seven out of ten classification cases. In contrast to the conventional PNN, the proposed network performed better in nine classification problems, while only in single task (S data set), the accuracies of MPNN and PNN were identical.

REFERENCES

[1] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.

[2] ——, "Probabilistic neural networks and the polynomial adaline as complementary techniques for classification," *Neural Networks, IEEE Transactions on*, vol. 1, no. 1, pp. 111–121, Mar 1990. doi: 10.1109/72.80210

[3] R. Folland, E. Hines, R. Dutta, P. Boilot, and D. Morgan, "Comparison of neural network predictors in the classification of tracheal–bronchial breath sounds by respiratory auscultation," *Artificial intelligence in medicine*, vol. 31, no. 3, pp. 211–220, 2004.

[4] D. Mantzaris, G. Anastassopoulos, and A. Adamopoulos, "Genetic algorithm pruning of probabilistic neural networks in medical disease estimation," *Neural Networks*, vol. 24, no. 8, pp. 831–835, 2011.

[5] M. Kusy and R. Zajdel, "Application of reinforcement learning algorithms for the adaptive computation of the smoothing parameter for probabilistic neural network," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 26, no. 9, pp. 2163–2175, 2015.

[6] ——, "Probabilistic neural network training procedure based on q(0)–learning algorithm in medical data classification," *Applied Intelligence*, vol. 41, no. 3, pp. 837–854, 2014.

[7] Y. Chtioui, S. Panigrahi, and R. Marsh, "Conjugate gradient and approximate newton methods for an optimal probabilistic neural network for food color classification," *Optical Engineering*, vol. 37, no. 11, pp. 3015–3023, 1998.

[8] S. Ramakrishnan and S. Selvan, “Image texture classification using wavelet based curve fitting and probabilistic neural network,” *International Journal of Imaging Systems and Technology*, vol. 17, no. 4, pp. 266–275, 2007.

[9] X.-B. Wen, H. Zhang, X.-Q. Xu, and J.-J. Quan, “A new watermarking approach based on probabilistic neural network in wavelet domain,” *Soft Computing*, vol. 13, no. 4, pp. 355–360, 2009.

[10] H. Adeli and A. Panakkat, “A probabilistic neural network for earthquake magnitude prediction,” *Neural networks*, vol. 22, no. 7, pp. 1018–1024, 2009.

[11] S. Venkatesh and 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 Systems with Applications*, vol. 38, no. 7, pp. 8978–8989, 2011.

[12] P. A. Kowalski and P. Kulczycki, “Data sample reduction for classification of interval information using neural network sensitivity analysis,” in *Artificial Intelligence: Methodology, Systems, and Applications*, ser. Lecture Notes in Computer Science, D. Dicheva and D. Dochev, Eds. Springer Berlin Heidelberg, 2010, vol. 6304, pp. 271–272.

[13] ——, “Interval probabilistic neural network,” *Neural Computing and Applications*, 2016. doi: 10.1007/s00521-015-2109-3 Online available.

[14] K. Elenius and H. G. Tråvén, “Multi-layer perceptrons and probabilistic neural networks for phoneme recognition.” in *EUROSPEECH*, 1993.

[15] T. P. Tran, T. T. S. Nguyen, P. Tsai, and X. Kong, “Bspnn: boosted subspace probabilistic neural network for email security,” *Artificial Intelligence Review*, vol. 35, no. 4, pp. 369–382, 2011.

[16] T. P. Tran, L. Cao, D. Tran, and C. D. Nguyen, “Novel intrusion detection using probabilistic neural network and adaptive boosting,” *International Journal of Computer Science and Information Security*, vol. 6, no. 1, pp. 83–91, 2009.

[17] D. Montana, “A weighted probabilistic neural network,” in *NIPS*, 1991, pp. 1110–1117.

[18] T. Song, M. Jamshidi, R. R. Lee, and 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.

[19] T. Song, C. Gasparovic, N. Andreasen, J. Bockholt, M. Jamshidi, R. R. Lee, and M. Huang, “A hybrid tissue segmentation approach for brain mr images,” *Medical and Biological Engineering and Computing*, vol. 44, no. 3, pp. 242–249, 2006. doi: 10.1007/s11517-005-0021-1

[20] S. Ramakrishnan and M. Emary, Ibrahiem, “Comparative study between traditional and modified probabilistic neural networks,” *Telecommun. Syst.*, vol. 40, no. 1-2, pp. 67–74, 2009. doi: 10.1007/s11235-008-9138-5

[21] 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,” *International Journal of Electrical Power & Energy Systems*, vol. 77, pp. 385–394, 2016.

[22] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[23] J. M. Zurada, A. Malinowski, and S. Usui, “Perturbation method for deleting redundant inputs of perceptron networks,” *Neurocomputing*, vol. 14, no. 2, pp. 177–193, 1997.

[24] J. Zurada, A. Malinowski, and 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, May 1994, pp. 447–450.

[25] P. A. Kowalski and P. Kulczycki, “A complete algorithm for the reduction of pattern data in the classification of interval information,” *International Journal of Computational Methods*, vol. 13, no. 03, p. 1650018, 2016. doi: 10.1142/S0219876216500183

[26] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[27] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[28] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[29] D. S. Broomhead and D. Lowe, “Multivariable functional interpolation and adaptive networks,” *Complex Systems*, vol. 2, pp. 321–355, 1988.

[30] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Applied statistics*, pp. 100–108, 1979.

[31] S. Chen, X. Wang, and C. J. Harris, “Experiments with repeating weighted boosting search for optimization signal processing applications,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 4, pp. 682–693, 2005.

[32] P. H. Sherrod, “Dtreg predictive modelling software.” [Online]. Available: http://www.dtreg.com