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.

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/

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 $S_{j,i}^{\text{mean}}$.