

Accepted Manuscript

Fuzzy c-means-based architecture reduction of a probabilistic neural network

Maciej Kusy

PII: S0893-6080(18)30211-9

DOI: <https://doi.org/10.1016/j.neunet.2018.07.012>

Reference: NN 3995

To appear in: *Neural Networks*

Received date: 8 January 2018

Revised date: 23 June 2018

Accepted date: 20 July 2018



Please cite this article as: Kusy, M., Fuzzy c-means-based architecture reduction of a probabilistic neural network. *Neural Networks* (2018), <https://doi.org/10.1016/j.neunet.2018.07.012>

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Fuzzy C–Means-Based Architecture Reduction of a Probabilistic Neural Network

Maciej Kusy

Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, al. Powstancow Warszawy 12, 35-959 Rzeszow, Poland

Abstract

The efficiency of the probabilistic neural network (PNN) is very sensitive to the cardinality of a considered input data set. It results from the design of the network's pattern layer. In this layer, the neurons perform an activation on all input records. This makes the PNN architecture complex, especially for big data classification tasks. In this paper, a new algorithm for the structure reduction of the PNN is put forward. The solution relies on performing a fuzzy c–means data clustering and selecting PNN's pattern neurons on the basis of the obtained centroids. Then, to activate the pattern neurons, the algorithm chooses input vectors for which the highest values of the membership coefficients are determined. The proposed approach is applied to the classification tasks of repository data sets. PNN is trained by three different classification procedures: conjugate gradients, reinforcement learning and the plugin method. Two types of kernel estimators are used to activate the neurons of the network. A 10–fold cross validation errors for the original and the reduced PNNs are compared. Received results confirm the validity of the introduced algorithm.

Keywords: probabilistic neural network, fuzzy c–means, architecture reduction, classification

1. Introduction

It is known that the complexity of the PNN's architecture proposed by Specht (1990) is high. This complexity is an effect of using all of the input

Email address: mkusy@prz.edu.pl (Maciej Kusy)

vectors to activate the neurons in the network's pattern layer. Therefore, to date, considerable research attention has been paid to the structure optimization of the PNN. For example, Burrascano (1990) applies the learning vector quantization procedure to find representative patterns that can be used to build neurons in PNNs. This procedure defines a number of records that are reference vectors, which approximate the probability density functions of the input classes. Chtioui et al. (1996) present cardinality reduction of the input data for a PNN by hierarchical clustering. The solution utilizes the technique of the reciprocal neighbours, allowing for the concentration of examples that are closest to each other. Zaknich (1997) introduces the quantization method for PNN structure simplification. The input space is split into a hypergrid of a fixed-size; for all hypercubes, representative cluster centers are determined. The number of training vectors in each hyper-cube is therefore reduced to one. Chang et al. (2008) show an expectation-maximization (EM) method as the training algorithm for a PNN. The idea relies on predefining a deterministic number of clusters as the input data set. A global k-means algorithm is used as the solution. Kusy and Kluska (2013, 2017) present an application of k-means clustering and support vector machines to simplify the PNN's architecture. The appropriately selected centroids and the support vectors are chosen to construct the pattern neurons for the network. Luzanin and Plancak (2014) introduce an approach that uses feature extraction to reduce the size of the PNN. The approach is performed by a clustering ensemble and provides cluster prototypes that are subsequently presented to the network as the training set. Kowalski and Kusy (2018) use a sensitivity analysis (SA) procedure to obtain a reduced set of pattern neurons. SA is applied to training vectors and relies on discarding those which are least sensitive for a given class. Kokkinos and Margaritis (2018) introduce a scalable construction method for PNNs. First, kernel-averaged gradient descent and subtractive clustering are employed to select representative data centers and their number for the PNN kernels. Then, the EM method is used to refine the PNN parameters.

The aim of this study is to propose a new algorithm for the PNN's architecture reduction. The solution consists in extracting the most representative input vectors out of the entire data set on the basis of fuzzy clustering. Selected input records are utilized to create the pattern neurons of the PNN. The algorithm is tested on a PNN trained by various procedures. Different kernel functions are applied to activate the neurons of the model. The performance of the original and reduced PNN is collated in the classification

tasks using the University of California, Irvine Machine Learning Repository (UCI-MLR) data sets (Bache and Lichman, 2015). To ensure a thorough analysis, statistical significance tests are conducted, the computational time of the algorithm is indicated and a comparison to the existing approaches is provided.

The remainder of this paper is organized as follows. Section 2 describes the architecture and the functioning principle of the PNN. In section 3, the basis of the c-means algorithm is highlighted. Section 4 introduces the algorithm by showing the general underlying idea and its detailed operation. The input data sets used in the simulations and the algorithm's parameter settings are outlined in sections 5 and 6, respectively. In section 7, a comparative analysis of the obtained results is presented. Section 8 concludes the work.

2. Probabilistic neural network

PNN is a feedforward neural network organized into four layers: (a) the input layer, where the neurons are represented by data features; (b) the pattern layer composed of as many neurons as input vectors; (c) the summation layer having a single neuron for each class; and (d) the output layer, which yields the classification result. The architecture of this model is shown in Figure 1. In general, the functioning of the PNN is based on an evaluation of the kernel density estimator (KDE) in the summation layer of the network for each class. Such a KDE can be generalized into the following form

$$\hat{f}(\mathbf{x}) = \frac{1}{L\sigma^n} \sum_{l=1}^L K\left(\frac{\mathbf{x} - \mathbf{x}^{(l)}}{\sigma}\right), \quad (1)$$

where $\mathbf{x} = [x_1, \dots, x_n]$ and $\mathbf{x}^{(l)} = [x_1^{(l)}, \dots, x_n^{(l)}]$ are an unknown sample and the l th training pattern, respectively; σ denotes the smoothing parameter, and L stands for data cardinality; $K(\cdot)$ is a kernel function that is of the unit integral and symmetrical with respect to zero, and it performs the mapping $\mathbb{R}^n \rightarrow [0, \infty)$.

The final classification verdict of the PNN is determined using the Bayes decision rule according to which \mathbf{x} is apportioned to a class j if for all classes for which $j \neq k$ the following inequality holds

$$p_j e_j f_j(\mathbf{x}) > p_k e_k f_k(\mathbf{x}), \quad (2)$$

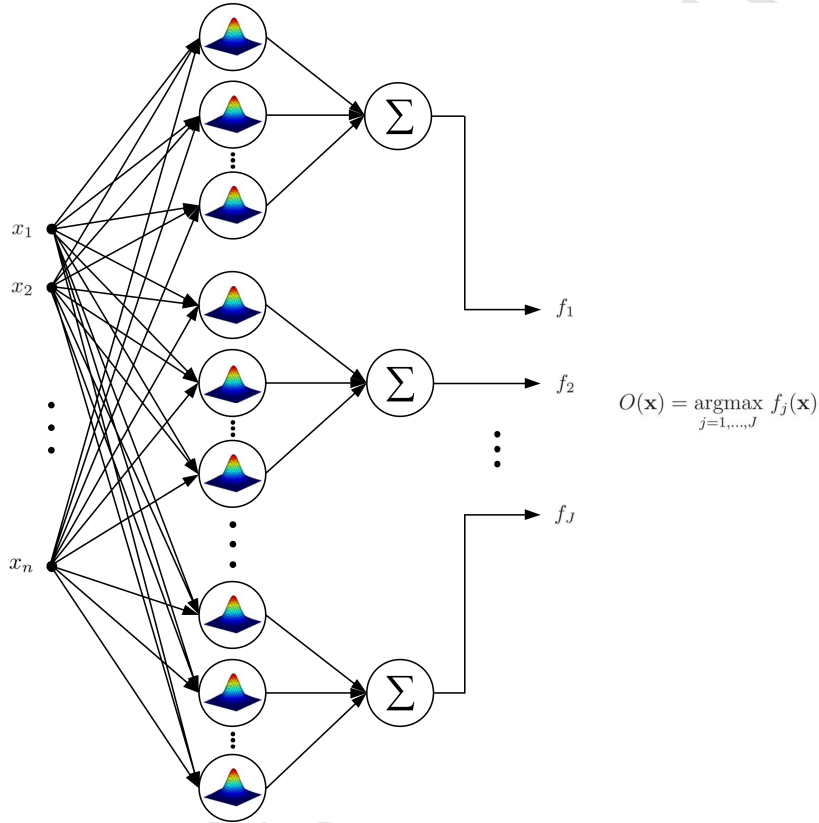


Figure 1: The architecture of probabilistic neural network including all pattern neurons.

where $j = 1, \dots, J$, p_j is the probability for \mathbf{x} to be assigned to the class j , e_j denotes the error of classifying \mathbf{x} into class j , and $f_j(\mathbf{x})$ stands for a KDE (1) for the j th class. With the assumption that $p_j = p_k$ and $e_j = e_k$, \mathbf{x} is designated as the vector belonging to the class j if $f_j(\mathbf{x}) > f_k(\mathbf{x})$.

For real, frequently complex and unbalanced data sets, one usually considers the matrix of smoothing parameters, which refer to each class and each data feature separately. This strategy is utilized in highlighting how to compute KDE precisely for PNN. Notably, there exist many types of $K(\cdot)$ functions that can be applied in KDE. In this work, only two kernels are regarded: a Gaussian kernel and a Cauchy kernel. The details are shown in two following sections.

2.1. PNN based on additive Gaussian kernel (AGK)

In this approach, KDE is mostly referred to as probability density function (PDF), where K in (1) has normal distribution. In such a case, the summation layer neuron representing the j th class computes the signal as follows (Specht, 1990)

$$f_j(\mathbf{x}) = \frac{1}{L_j(2\pi)^{n/2}\det\Sigma_j} \sum_{l=1}^{L_j} \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}_j^{(l)})^T \Sigma_j^{-2} (\mathbf{x} - \mathbf{x}_j^{(l)})\right), \quad (3)$$

where:

- L_j is the number of training cases from the j th class;
- $\Sigma_j = \text{diag}(\sigma_{j1}, \sigma_{j2}, \dots, \sigma_{jn})$ stands for the matrix in which σ_{ji} refers to the smoothing parameter determined for the j th class and the i th feature;
- $\mathbf{x}_j^{(l)} = [x_{j1}^{(l)}, \dots, x_{jn}^{(l)}]$ denotes the l th training vector from the j th class.

Thus, the j th summation neuron output amounts to the computation of the following signal

$$f_j(\mathbf{x}) = \frac{1}{L_j(2\pi)^{n/2} \prod_{i=1}^n \sigma_{ji}} \sum_{l=1}^{L_j} \exp\left(-\sum_{i=1}^n \frac{(x_i - x_{ji}^{(l)})^2}{2\sigma_{ji}^2}\right). \quad (4)$$

2.2. PNN based on product Cauchy kernel (PCK)

The second considered form for K in (1) may utilize a product notation

$$K(\mathbf{x}) = \mathcal{K}(x_1) \cdot \mathcal{K}(x_2) \cdot \dots \cdot \mathcal{K}(x_n), \quad (5)$$

where

$$\mathcal{K}(x_i) = \frac{2}{\pi(x_i^2 + 1)^2} \quad (6)$$

represents the one-dimensional Cauchy multiplicand (Kowalski and Kulczycki, 2016). Then, the generalized formula for the j th summation layer neuron is as follows

$$f_j(\mathbf{x}) = \frac{1}{L_j \det\Sigma_j} \sum_{l=1}^{L_j} \frac{1}{s_l^n} K\left(\frac{1}{s_l} (\mathbf{x} - \mathbf{x}_j^{(l)})^T \Sigma_j^{-1}\right), \quad (7)$$

where s_l is the modification coefficient, which is computed for all input vectors independently. For the details, see (Wand and Jones, 1994).

Using equations (5) and (6), the output of the summation neuron (7) is expanded to the following form

$$f_j(\mathbf{x}) = \frac{2^n}{\pi^n L_j \prod_{i=1}^n \sigma_{ji}} \sum_{l=1}^{L_j} \frac{1}{s_l^n} \prod_{i=1}^n \frac{1}{\left(\left(\frac{x_i - x_{j,i}^{(l)}}{\sigma_{ji} s_l} \right)^2 + 1 \right)}. \quad (8)$$

2.3. Operation of PNN

Once the form of the KDE is chosen to perform the activation in the summation layer and the PNN's required parameters are determined, the network yields the final output in accordance with Bayes' decision rule

$$O(\mathbf{x}) = \operatorname{argmax}_{j=1, \dots, J} f_j(\mathbf{x}), \quad (9)$$

where output $O(\mathbf{x})$ is a label of the predicted class.

As shown in Figure 1, the network is not equipped with any weighting coefficients. Therefore, no iterative algorithm that relies on the minimization of weights-based error function is required. The only parameters of the PNN to be optimized are Σ_j for the AGK and both Σ_j and s_l for the PCK. There are various methods that can be utilized for determination of PNN's parameters. For example, to compute Σ_j for PNN activated by PDF in (3), one usually applies: conjugate gradients (Chtioui et al., 1998; Sherrod, 2017), genetic algorithms (Mao et al., 2000), reinforcement learning (Kusy and Zajdel, 2014, 2015) or particle swarm optimization (Georgiou et al., 2008). In the case of the PNN with KDE in (7), Σ_j is optimized by the plug-in method (Wand and Jones, 1994; Kowalski and Kulczycki, 2016, 2017).

3. Fuzzy c-means algorithm

The fuzzy c-means algorithm (FCM) proposed by Dunn (1973) and then generalized by Bezdek (1981) belongs to the group of clustering techniques in which the data are partitioned into a number of groups, usually called clusters. Partitioning is performed in an approach that ensures a strong relationship between the input records in each cluster. Classical hard clustering techniques, such as the k-means algorithm (Hartigan and Wong, 1979), require that each record is assigned into a single cluster. In contrast, FCM

allows records to belong to more than one cluster with some degree of membership. In this way, a fuzzy partition is obtained where each cluster is associated with a membership function (MF). The MF quantifies the degree to which an individual record belongs to the cluster.

FCM performs the partitioning of a finite set of L input records $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)}\}$ into a collection of fuzzy cluster centers $\mathbf{C} = \{\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(C)}\}$ by iteratively minimizing the following objective function

$$I_m(\mathbf{U}, \mathbf{C}) = \sum_{l=1}^L \sum_{c=1}^C (u_c^{(l)})^m \|\mathbf{x}^{(l)} - \mathbf{c}^{(c)}\|_p^2, \quad (10)$$

where $\mathbf{U} = \{u_c^{(l)}\}_{L \times C}$ is the membership matrix; each $u_c^{(l)} \in [0, 1]$ expresses the degree to which $\mathbf{x}^{(l)}$ belongs to $\mathbf{c}^{(c)}$; m is a weighting exponent ($1 < m < \infty$), also called a fuzzifier; and $\|\cdot\|_p$ denotes the p -norm in \mathbb{R}^n . The membership coefficients $u_c^{(l)}$ are initialized with random values.

The fuzzy cluster's centroid is simply the mean over all input records, weighted by their degree of belonging to the cluster

$$\mathbf{c}^{(c)} = \frac{\sum_{l=1}^L (u_c^{(l)})^m \mathbf{x}^{(l)}}{\sum_{l=1}^L (u_c^{(l)})^m}, \quad (11)$$

while the membership values are updated as follows

$$u_c^{(l)} = \frac{1}{\sum_{k=1}^C \left(\frac{\|\mathbf{x}^{(l)} - \mathbf{c}^{(c)}\|_p}{\|\mathbf{x}^{(l)} - \mathbf{c}^{(k)}\|_p} \right)^{\frac{2}{m-1}}}. \quad (12)$$

Since the minimization of (10) is performed iteratively, $\mathbf{c}^{(c)}$ and $u_c^{(l)}$ are computed until I_m decreases below a specified threshold, a specified number of iterations is reached or $|\mathbf{U}(t) - \mathbf{U}(t-1)| < \epsilon$.

4. Proposed algorithm

This section presents a new algorithm for the reduction of PNN's pattern layer. The approach requires an application of the FCM to the original input

data set to determine the centroids. On the basis of the centroids, a lower number of pattern neurons is created. The performance of the reduced PNN is evaluated using a specified error measure.

4.1. General idea

As the solution, FCM provides an optimal pair (\mathbf{U}, \mathbf{C}) in the sense of I_m minimization. In the proposed algorithm, both \mathbf{U} and \mathbf{C} are directly used to determine the pattern neurons of the PNN.

In the first stage of the approach, FCM is applied to the data records of all J classes separately. $\mathbf{C}_1, \dots, \mathbf{C}_J$ sets of centroids are obtained. The number of the centroids in each \mathbf{C}_j is determined in an iterative manner using the formula suggested by Kusy and Kluska (2017)

$$C_{sj} = \text{round} \left(\frac{s}{N} L_j \right), s = 1, \dots, N - 1, \quad (13)$$

where $\text{round}(\cdot)$ rounds the argument to the nearest positive integer and N is a natural number ($N \geq 2$). Such a solution provides the possibility of utilizing an equal percentage of data in the partitioning process, which is crucial in problems with imbalanced classes. This outcome means that the algorithm includes in the analysis the input vectors that belong to all classes, even the least numerous ones. In other words, the records of every class are represented in the obtained partitions. Since the centroids are determined for each class independently, $\mathbf{U}_1, \dots, \mathbf{U}_J$ membership matrices are computed. For example, let us consider the case of three classes labeled $\{1, 2, 3\}$, storing randomly generated data in \mathbb{R}^2 with the cardinalities $L_1 = 200$, $L_2 = 300$ and $L_3 = 400$, so that for $N = 10$ and $s = 1$, one obtains $C_{11} = 20$, $C_{12} = 30$, $C_{13} = 40$ centroids, respectively. The total number of the centroids for a given s is defined as

$$C_s = \sum_{j=1}^J C_{sj}. \quad (14)$$

Therefore, for $s = 1$, $C_1 = 90$. The 90 centroids are used to create neurons in the pattern layer of the PNN, which further undergoes training. The process is repeated $N - 1$ times, each time determining the network's performance.

In the second stage, the algorithm searches the input vectors, for which the largest values of the membership coefficients $u_{cj}^{(l)}$ are found in each matrix

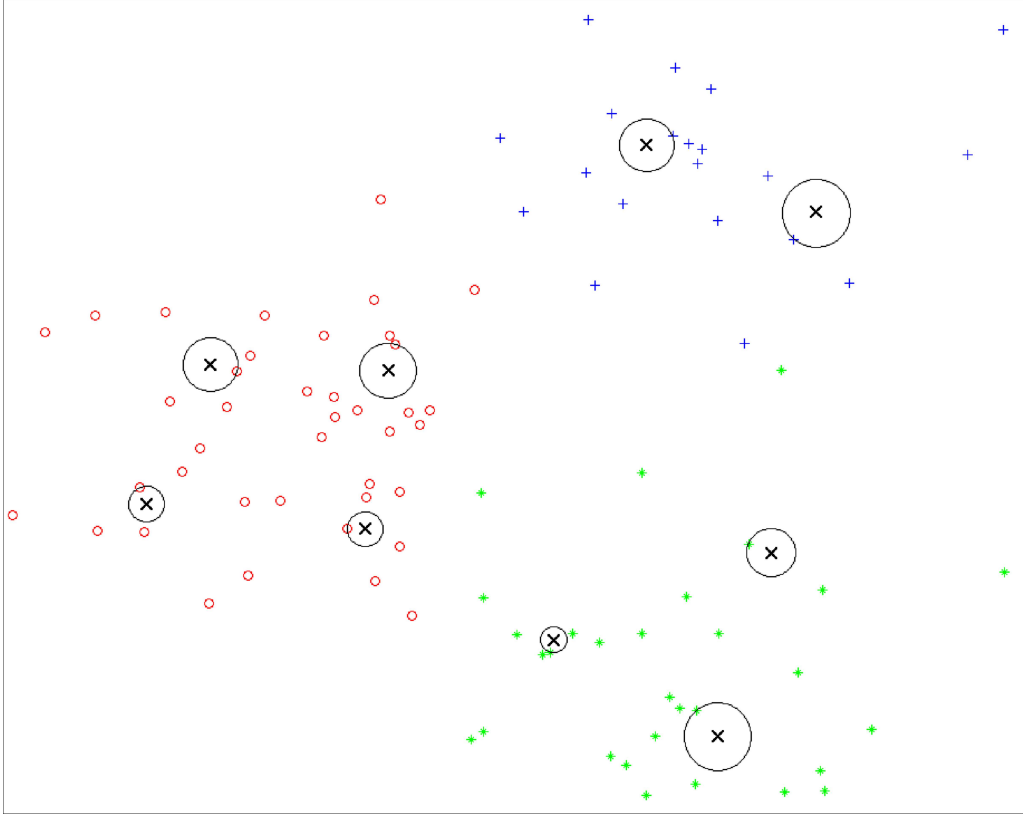


Figure 2: Graphical presentation of the idea of selecting representative input vectors for the PNN's pattern layer. The rings are the consequence of applying $\|\cdot\|_2$ norm, i.e. the Euclidean distance to obtain the centroids.

\mathbf{U}_j , taking into account the s parameter, at which the network's highest performance is attained. These vectors are the nearest to the centroids in j th class. Out of the entire j th subset, they are selected to build the pattern neurons of the final PNN. Such a selection depends on the choice of the s parameter and the m fuzzifier. The whole procedure is repeated for all J classes. For a given example, the number of pattern neurons is 90. The PNN is trained using 90 pattern neurons that, unlike in the first stage of the algorithm, are now original data. The network's performance is further assessed.

Figure 2 shows exemplary centroids (marked with \times) determined for a smaller number of records, also of three classes. For clarity, the utilized vectors are denoted by red circles, green stars and blue crosses. The data

cardinalities are reduced by 10; therefore, $L_1 = 20$, $L_2 = 30$ and $L_3 = 40$, yielding 2, 3, and 4 centroids for each class, respectively. The vectors lying on the ring indicate that they have highest $u_{c_j}^{(l)}$ and that they are placed within the smallest distance to the centroid – they are chosen to construct the pattern neurons of the network.

Unlike common clustering-based techniques that use a set of centroids to construct a PNN's pattern neurons, the current approach utilizes original input vectors that are selected on the basis of the highest similarity to the centroids.

4.2. Detailed algorithm

All the errors utilized in the proposed algorithm are the classification errors, generally computed as follows

$$\mathcal{E} = \frac{1}{L} \sum_{l=1}^L \delta [O(\mathbf{x}^{(l)}) \neq t^{(l)}], \quad (15)$$

where $O(\mathbf{x}^{(l)})$ denotes the PNN's output obtained for $\mathbf{x}^{(l)}$, and $t^{(l)}$ is the desired target. In (15), $\delta[\cdot] = 1$ when $O(\mathbf{x}^{(l)}) \neq t^{(l)}$ and 0, otherwise.

The pseudocode of the algorithm is presented in Algorithm 1. In the beginning, the input–output data pairs (\mathbf{X}, \mathbf{t}) are read (step **1**). Thereafter, for each class, one selects the data subsets $\mathbf{X}_j = \{\mathbf{x}_j^{(1)}, \dots, \mathbf{x}_j^{(L_j)}\}$, $\bigcup_{j=1}^J \mathbf{X}_j = \mathbf{X}$. In step **4**, the classification error E is computed for the PNN composed of all L pattern neurons. In step **5**, the fuzzifier m is assigned with the values in the range $(1, \infty)$. Setting $N = 10$ in the next step is a consequence of the assumption introduced into (13). Steps **7–29** present the main loop of the algorithm. This loop operates for each m and s , considering every j th class separately. Here, the total number of centroids C_s is set to 0 initially. In step **11**, the reduced set of input vectors \mathbf{P}_s is created and initialized; these will be used to construct the pattern neurons of the final PNN. Then, the C_{sj} centroids and the membership matrix \mathbf{U}_j are computed in steps **13** and **14**, respectively. \mathbf{U}_j stores the coefficients that describe the degree to which all input vectors from the j th class belong to all centroids that were determined for this class. In step **15**, in each c th column of the matrix \mathbf{U}_j (denoted here with $\{c\}$ notation), the algorithm seeks an index for which the highest membership coefficient is attained. Such an index identifies $\mathbf{x}_j^{(l)}$, which is closest to the c th centroid in the j th class in terms of fuzzy similarity. This

```

1 Read input data  $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(L)}\}$ ,  $\mathbf{t} = \{t^{(1)}, \dots, t^{(L)}\}$ 
2 Extract separate data subsets  $\mathbf{X}_j$  for each class
3 Select KDE for PNN to compute  $f_j(\mathbf{x})$ 
4 Compute  $E$  for full structure PNN
5 Assume values for fuzzifier  $m$ 
6  $N = 10$ 
7 foreach  $m$  do
8   % First stage of algorithm
9   for  $s = 1$  to  $N - 1$  do
10    Set number of centroids  $C_s = 0$ 
11    Initialize set of reduced input vectors  $\mathbf{P}_s = \emptyset$ 
12    for  $j = 1$  to  $J$  do
13     Compute  $C_{sj}$  centroids according to (13)
14     Determine membership matrix for  $j$ th class  $\{\mathbf{U}_j\}_{L_j \times C_{sj}}$ 
15     Find highest membership indices  $\mathbf{i}_j = \operatorname{argmax}_{c=1, \dots, C_{sj}} \mathbf{U}_j\{c\}$ 
16      $C_s = C_s + C_{sj}$ 
17      $\mathbf{P}_s = \mathbf{P}_s \cup \mathbf{X}_j\{\mathbf{i}_j\}$ 
18    end
19    Compute centroids-based error  $E_s$ 
20  end
21  % Second stage of the algorithm
22  Find  $s^* = \operatorname{argmin}_{s=1, \dots, N-1} E_s$ 
23  Determine:
24  – minimum centroids-based error  $E^* = E_{s^*}$ 
25  – optimal number of centroids  $C^* = C_{s^*}$ 
26  – optimal set of reduced input vectors  $\mathbf{P}^* = \mathbf{P}_{s^*}$ 
27  Compute final reduction error  $E_R^*$ 
28  Compute PNN reduction ratio  $R = L/C^*$ 
29 end
30 return  $E^*$ ,  $C^*$ ,  $\mathbf{P}^*$ ,  $E_R^*$ ,  $R$ 

```

Algorithm 1: The algorithm for the reduction of the PNN's pattern layer.

means that $\sum_{c=1}^{C_{sj}} u_{cj}^{(l)} = 1, \forall \mathbf{x}_j^{(l)}$. In step **16**, the number of the centroids is increased while step **17** allows the algorithm to update the reduced set of input vectors \mathbf{P}_s . Both steps are performed in such a way that all J classes are considered for a given value of the s parameter. Afterwards, the obtained C_s centroids are adopted to create the neurons in the PNN's pattern layer. The network's performance is evaluated by computing the centroids-based error (step **19**).

Definition 1. Let $\mathbf{C}_s = \bigcup_{j=1}^J \mathbf{C}_{sj}$ be a set of the centroids determined in s th iteration of Algorithm 1, where $\mathbf{C}_{sj} = \{\mathbf{c}_{sj}^{(1)}, \dots, \mathbf{c}_{sj}^{(C_{sj})}\}$ is a subset of the centroids computed for the j th class. The centroids-based error E_s is defined as the classification error in the sense of \mathcal{E} in (15) for the PNN, whose pattern neurons utilize C_s centroids from \mathbf{C}_s for activation.

Once the E_s errors are determined for each s , in step **22**, one finds the optimal index s^* providing the lowest centroids-based error among all E_s values. Thereafter, in steps **24–26**, with the use of the s^* index, the algorithm yields the minimum centroids-based error $E^* = E_{s^*}$, the optimal number of the centroids C^* and the set of reduced input vectors \mathbf{P}^* that are nearest to the centroids. Since $C^* = |\mathbf{P}^*|$, C^* can also be called a reduced number of pattern neurons. Next, in step **27**, the final reduction error is determined.

Definition 2. The final reduction error E_R^* is defined as the classification error in the sense of \mathcal{E} in (15) for the PNN, whose pattern neurons utilize \mathbf{P}^* input vectors for activation.

The main loop finishes with calculation of the reduction ratio $R = L/C^*$. The loop must operate until all fuzzifier values are examined. Finally, depending on the number of m values, the algorithm returns the indicators E^* , C^* , \mathbf{P}^* , E_R^* , and R .

5. Input data sets

In this work, in order to assess the performance of the presented algorithm, the reduced and original PNN models are tested in six classification tasks. The following UCI-MLR data sets are used in the experiments (Bache and Lichman, 2015): cardiocography, Parkinson, Haberman, Iris, diabetes and seeds. All of them are listed in Table 1, along with information on their cardinality, number of features and class distribution.

Table 1: The UCI-MLR data sets used to test the reduced and the original PNN models.

Data set	Records	Features	Class distribution
cardiotocography	2126	22	1655-295-176
Parkinson	195	22	147-48
Haberman	306	3	225-81
Iris	150	4	50-50-50
diabetes	786	8	500-268
seeds	210	7	70-70-70

6. Parameter settings

In this work, Algorithm 1 is applied to a PNN whose summation neuron signal is computed with the use of AGK and PCK defined in (3) and (7), respectively. For the AGK-based PNN, two training procedures are used: conjugate gradients (CG), available in (Sherrod, 2017), and reinforcement learning (RL). As the RL solution, the $Q(0)$ -learning algorithm is employed (Kusy and Zajdel, 2014, 2015). In the case of the PCK-based PNN, the plugin method (PM) is utilized (Kowalski and Kulczycki, 2016, 2017).

The placement of the centroids in \mathbb{R}^n strongly depends on the value of the m fuzzifier. However, there is no theoretical or computational recommendation for the choice of an optimal m . According to Bezdek (1981), the range of m useful values seems to be (1, 30). If a test set is available for a given task, the best strategy for selecting m is essentially performing a vast number of experiments. For most data, $m \in [1.5, 3.0]$ yields good results (Bezdek, 1981). In this study, $m = \{2, 3, 4\}$. To determine (10), $p = 2$; therefore, the Euclidean norm is utilized.

The full structure PNN classification error E , the centroids-based error E_s and the final reduction error E_R^* are determined by means of a 10-fold cross validation procedure. The simulations are repeated 10 times, and all results are averaged. As presented in step 6 of Algorithm 1, $N = 10$; therefore, $s = \{1, 2, \dots, 9\}$.

The obtained results are evaluated using statistical significance tests. The performance between the original and the reduced PNN is explored by means of a two sample t-test. The statistical significance level is assumed to be 0.95. The null hypotheses states that averaged PNN's errors before and after reduction are equal.

7. Simulation experiments

This section outlines the results obtained by the PNN in the classification tasks of the considered data sets. The network is reduced by means of Algorithm 1. The CG, RL and PM are utilized as the PNN training procedures. The appropriate KDEs are selected to activate the signals in the summation layer of the model.

In subsection 7.1, the PNN's performance is determined by computing the centroids-based error E_s for $s = \{1, 2, \dots, 9\}$ and $m = \{2, 3, 4\}$. The presented outcomes refer to the first stage of Algorithm 1 (steps **9–20**). On the basis of the lowest E_s value, the minimum centroids-based error E^* is found. E^* affects the selection of the pattern neurons for the final PNN.

Subsection 7.2 addresses the second stage of the Algorithm 1. This stage provides the final outcome of the proposed approach, i.e., the performance of the PNN composed of the ultimate reduced set of neurons, which utilize \mathbf{P}^* input vectors in the pattern layer.

7.1. PNN with pattern neurons established from centroids

Figures 3, 4 and 5 show centroids-based errors E_s [%], $\forall s$, computed at C_s centroids for the PNN trained using CG, RL and PM procedures, respectively. In each figure, six sub-plots illustrate E_s for $m = 2$ (red-circle-solid line), $m = 3$ (blue-square-dash line) and $m = 4$ (green-triangle-dash-dot line) in the classification tasks of (a)–cardiotocography, (b)–Parkinson, (c)–Haberman, (d)–Iris, (e)–diabetes, and (f)–seeds data sets. Additionally, the full structure PNN classification error E is presented – it is marked using a thin black dash-dot-dot line. For each training procedure, the following main observations can be pointed out: (i) the changes of E_s follow a similar pattern for particular values of the fuzzifier m ; (ii) it is always possible to find $E_s < E$; and (iii) the differences between particular E_s values at individual s are small. Additionally, in the case of the RL training procedure, in two classification tasks (cardiotocography and diabetes data sets), $E_s < E$ holds for all s . In the cases of the remaining training procedures, $E_s < E$ for all s occurs only for the diabetes data set.

It is difficult to read exact E^* values from sub-plots (a)–(f) in Figures 3–5. Therefore, summary Table 2 is introduced. It shows E^* (along with standard deviations in parentheses) obtained for $s^* = \underset{s=1, \dots, N-1}{\operatorname{argmin}} E_s$ and $C^* = C_{s^*}$, i.e., the three indicators determined in the early phase of the second stage of Algorithm 1 (steps **22–25**). The full structure PNN classification errors

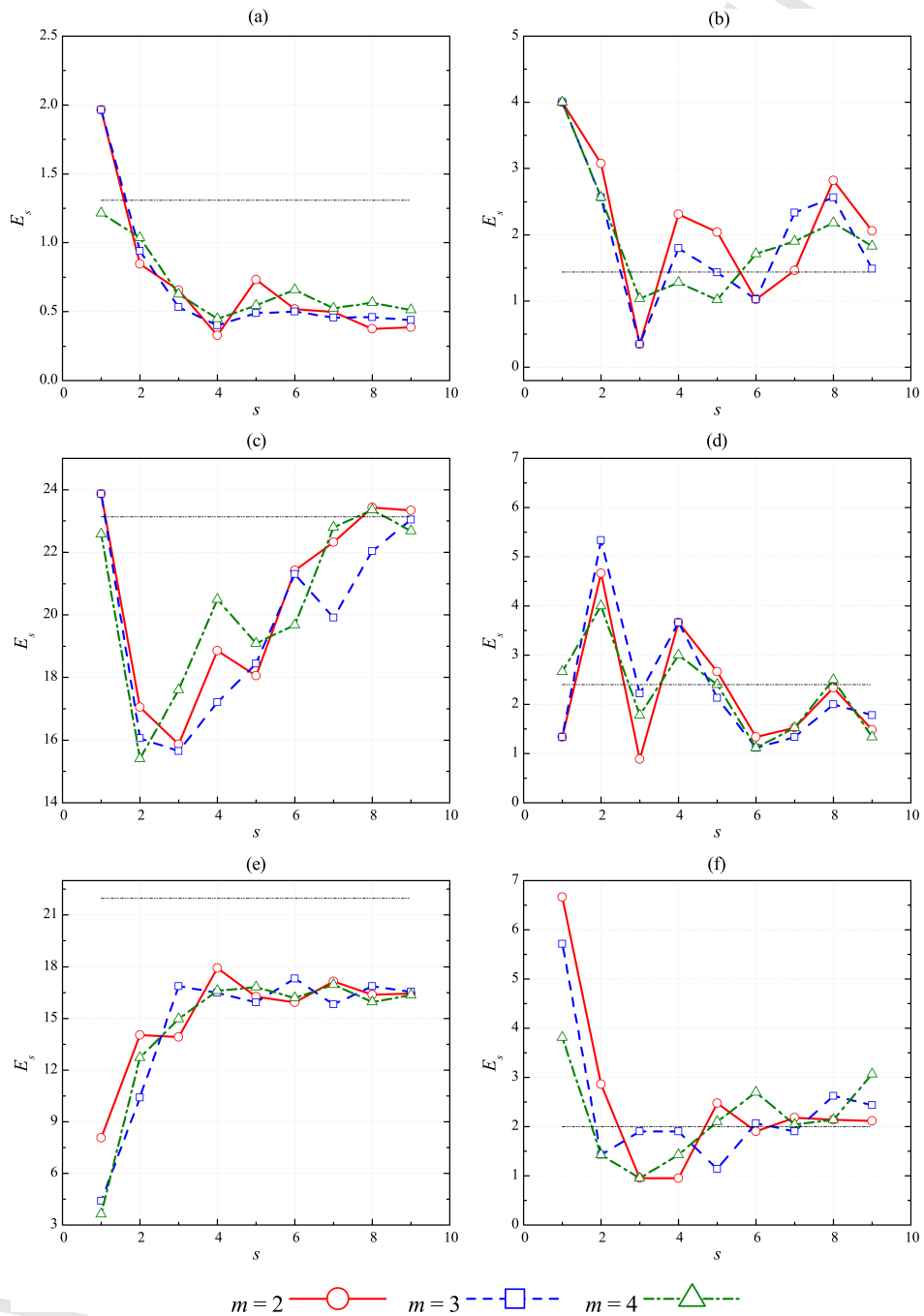


Figure 3: Conjugate gradients: centroids-based errors E_s obtained at C_s centroids, $s = \{1, 2, \dots, 9\}$ for the following data sets: (a)–cardiocography, (b)–Parkinson, (c)–Haberman, (d)–Iris, (e)–diabetes, (f)–seeds. Each sub-plot shows E_s determined for $m = \{2, 3, 4\}$ and the full structure PNN classification error E (black dash-dot-dot line).

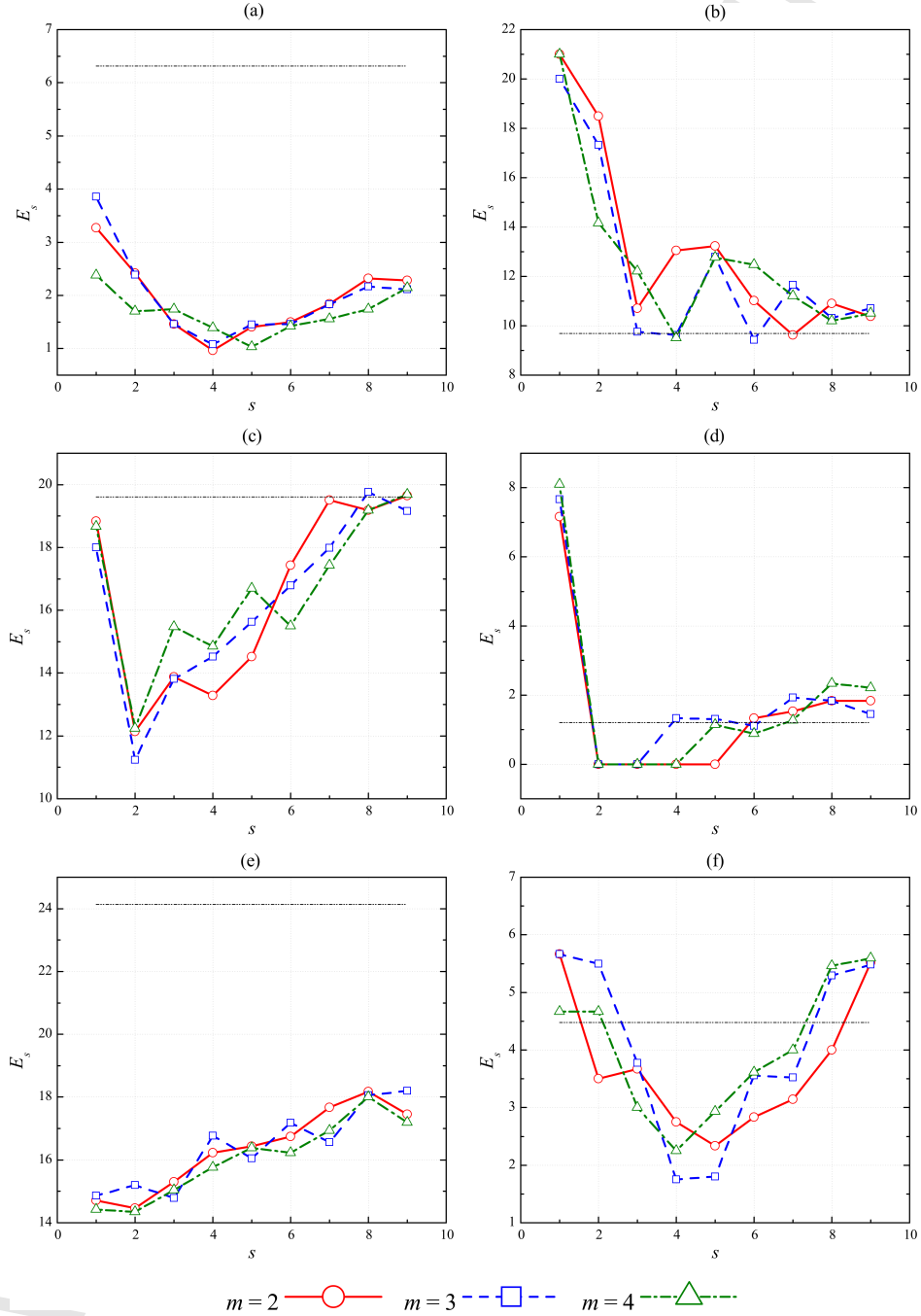


Figure 4: Reinforcement learning: centroids-based errors E_s obtained at C_s centroids, $s = \{1, 2, \dots, 9\}$ for the following data sets: (a)–cardiotocography, (b)–Parkinson, (c)–Haberman, (d)–Iris, (e)–diabetes, (f)–seeds. Each sub-plot shows E_s determined for $m = \{2, 3, 4\}$ and the full structure PNN classification error E (black dash-dot-dot line).

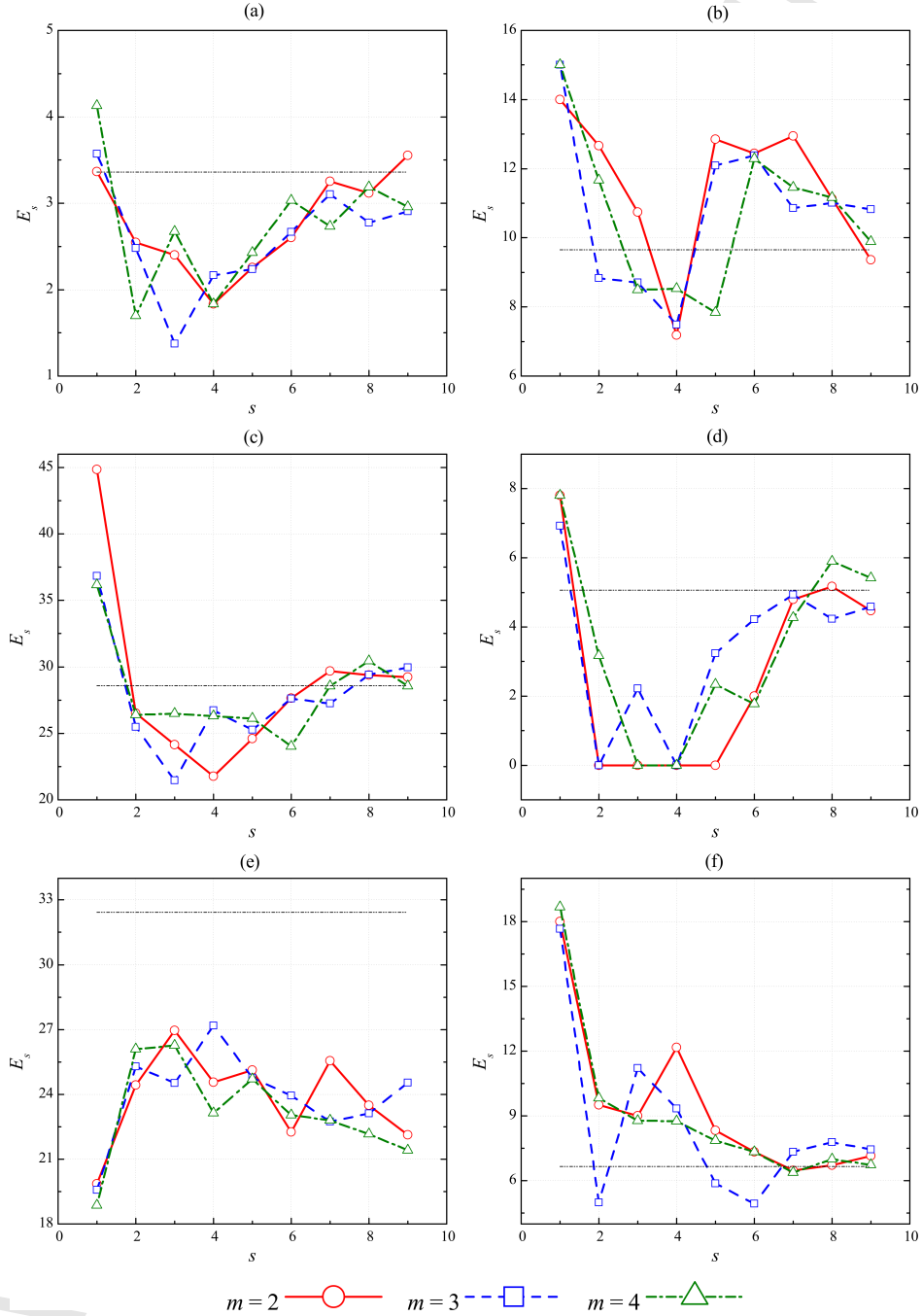


Figure 5: Plugin method: centroids-based errors E_s obtained at C_s centroids, $s = \{1, 2, \dots, 9\}$ for the following data sets: (a)–cardiocography, (b)–Parkinson, (c)–Haberman, (d)–Iris, (e)–diabetes, (f)–seeds. Each sub-plot shows E_s determined for $m = \{2, 3, 4\}$ and the full structure PNN classification error E (black dash-dot-dot line).

E are included in the last column of this table. The lowest values of all errors provided in each row are shown in bold. It can be seen that for all classification cases, each training procedure and all fuzzifier values, $E^* < E$ holds. The statistical significance is presented in this table with the use of a + symbol placed at the averaged error values. The symbol indicates that there is a difference between E and E^* errors.

Analyzing the results shown in Table 2, it is also worth noting that the lowest possible value of s^* is 1 for a PNN trained by CG and PM procedures, and 2 for a PNN trained by reinforcement learning. This means that the application of Algorithm 1 allows the network to reduce its pattern layer size by almost 10 times for the CG and PM procedures (diabetes data set, $m = \{2, 3, 4\}$) and 5 times for the RL procedure (both Haberman and Iris data sets at $m = \{2, 3, 4\}$ and diabetes data set at $m = \{2, 4\}$), increasing its performance at the same time. On the other hand, the highest value of s^* is equal to 7 for the network trained with the use of the RL (Parkinson data set, $m = 2$) and PM (seeds data set, $m = \{2, 4\}$) procedures. In these cases, the pattern layer is decreased by 1.42 times, making the PNN still improve its performance.

7.2. PNN with pattern neurons established from reduced data

In contrast to the analysis presented in subsection 7.1, the outcomes delineated in the current part of the paper pertain to the PNN in which the elements of \mathbf{P}^* are used to build the pattern neurons of the network. The number of reduced pattern neurons is also equal to C^* ; however, the original input vectors are now utilized in the PNN architecture. The discussion is based on the final reduction error E_R^* and the reduction ratio R returned by Algorithm 1.

Tables 3, 4 and 5 present the results obtained by the PNN trained with the use of CG, RL and PM procedures, respectively. For each classification task, each training procedure and all fuzzifier values, E_R^* and R are outlined in the tables. Moreover, for comparison purposes, the full structure PNN classification error is provided. The standard deviations for all error results are placed in parentheses. For each classification outcome, an additional row is inserted with the ranking score (RS), which is simply the number from the set $\{0, 1, 2, 3\}$, where 0 and 3 denote the worst and the best PNN's performance, respectively. In the last column, a decrease of the classification error after the reduction expressed in terms of % is introduced; it is defined

Table 2: The optimal s^* index, the optimal number of the centroids C^* and the minimum centroids-based error E^* determined for $m = \{2, 3, 4\}$ for the PNN trained according to Algorithm 1 with the use of CG, RL and PM procedures. The last column indicates the full structure PNN classification error.

Data set	$m = 2$			$m = 3$			$m = 4$			E
	s^*	C^*	E^*	s^*	C^*	E^*	s^*	C^*	E^*	
Conjugate gradients										
cardiotocography	4	850	0.33 ⁺ (0.08)	4	850	0.40 ⁺ (0.12)	4	850	0.45 ⁺ (0.22)	1.31 (0.16)
Parkinson	3	58	0.34 ⁺ (0.69)	3	58	0.34 ⁺ (0.69)	5	98	1.02 ⁺ (0.65)	1.74 (0.41)
Haberman	3	92	15.87 ⁺ (1.76)	3	92	15.65 ⁺ (1.11)	2	61	15.41 ⁺ (1.31)	23.14 (1.31)
Iris	3	45	0.89 ⁺ (1.09)	6	90	1.11 ⁺ (0.70)	6	90	1.11 ⁺ (0.99)	2.40 (0.33)
diabetes	1	77	8.05 ⁺ (2.52)	1	77	4.42 ⁺ (0.64)	1	77	3.64 ⁺ (0.97)	21.98 (0.39)
seeds	3	63	0.95 ⁺ (0.77)	5	105	1.14 ⁺ (0.38)	3	63	0.95 ⁺ (0.77)	2.29 (0.36)
Reinforcement learning										
cardiotocography	4	850	0.96 ⁺ (0.08)	4	850	1.08 ⁺ (0.20)	5	1064	1.03 ⁺ (0.17)	6.31 (0.14)
Parkinson	7	137	9.62 (1.45)	6	117	9.43 (0.92)	4	78	9.51 (0.74)	9.69 (0.30)
Haberman	2	61	12.14 ⁺ (1.41)	2	61	11.24 ⁺ (1.90)	2	61	12.24 ⁺ (1.12)	19.61 (0.52)
Iris	2	30	0.00 ⁺ (0.00)	2	30	0.00 ⁺ (0.00)	2	30	0.00 ⁺ (0.00)	1.20 (0.27)
diabetes	2	154	14.46 ⁺ (1.29)	3	230	14.78 ⁺ (0.91)	2	154	14.35 ⁺ (0.61)	24.14 (0.40)
seeds	5	105	2.33 ⁺ (0.92)	4	84	1.75 ⁺ (0.61)	4	84	2.25 ⁺ (1.23)	4.48 (0.71)
Plugin method										
cardiotocography	4	850	1.84 ⁺ (0.09)	3	639	1.37 ⁺ (0.24)	2	425	1.69 ⁺ (0.20)	3.36 (0.11)
Parkinson	4	78	7.18 ⁺ (1.88)	4	78	7.49 ⁺ (0.83)	5	98	7.84 ⁺ (0.61)	9.66 (0.46)
Haberman	4	122	21.76 ⁺ (2.06)	3	92	21.45 ⁺ (0.85)	6	184	24.01 ⁺ (1.37)	28.59 (1.17)
Iris	2	30	0.00 ⁺ (0.00)	2	30	0.00 ⁺ (0.00)	3	45	0.00 ⁺ (0.00)	5.07 (0.33)
diabetes	1	77	19.86 ⁺ (0.95)	1	77	19.57 ⁺ (1.67)	1	77	18.86 ⁺ (0.97)	32.43 (1.20)
seeds	7	147	6.48 (0.28)	6	126	4.94 ⁺ (0.48)	7	147	6.38 (0.47)	6.67 (0.30)

Table 3: Conjugate gradients: the full structure PNN classification error (E for L neurons), the PNN reduction ratio R , the final reduction error (E_R^* for C^* neurons) and the ranking score $RS = \{0, 1, 2, 3\}$; the values in the right most column are the decreases of the classification error after reduction.

Data set	E	$m = 2$		$m = 3$		$m = 4$		E_d [%]
		R	E_R^*	R	E_R^*	R	E_R^*	
cardiotocography	1.31	2.50	0.31 ⁺	2.50	0.52 ⁺	2.50	0.75 ⁺	76.34
	(0.16)		(0.14)		(0.21)		(0.14)	
	0		3		2		1	
Parkinson	1.74	3.62	1.03 ⁺	3.62	0.69 ⁺	1.99	1.43	60.34
	(0.41)		(0.84)		(0.84)		(1.04)	
	0		2		3		1	
Haberman	23.14	3.33	16.52 ⁺	3.33	16.74 ⁺	5.02	17.38 ⁺	28.61
	(1.31)		(1.27)		(0.87)		(0.80)	
	0		3		2		1	
Iris	2.40	3.33	3.78 ⁺	1.67	1.53 ⁺	1.67	0.25 ⁺	89.58
	(0.33)		(1.42)		(0.54)		(0.46)	
	1		0		2		3	
diabetes	21.98	9.97	9.25 ⁺	9.97	12.99 ⁺	9.97	9.25 ⁺	57.92
	(0.39)		(2.29)		(1.72)		(1.20)	
	0		2		1		3	
seeds	2.29	3.33	1.59 ⁺	2.00	1.55 ⁺	3.33	2.18	32.31
	(0.36)		(0.79)		(0.82)		(0.77)	
	0		2		3		1	

as follows

$$E_d = 1 - \frac{1}{E} \min_{m=\{2,3,4\}} E_R^*(m), \quad (16)$$

where $E_R^*(m)$ denotes the final reduction error E_R^* achieved for a particular value of m . As in the case of Table 2, Tables 3–5 show the statistical significance of the test by means of a + symbol placed at E and E_R^* , indicating that both errors differ in their values.

The application of Algorithm 1 for the PNN trained using the CG procedure yields $E_R^* < E$ in all classification cases (refer to bold numbers in each row of Table 3). The ranking score for the original network reaches the value of 1 for the Iris data set. For the remaining data sets, $RS = 0$. This means that the full structure PNN is the third best classifier only once while in other cases it is the worst prediction model. The largest decrease of the classification error after reduction is attained for the Iris data set; it occurs when the PNN’s pattern layer neurons are decreased from 150 to 90, yielding $E_d = 89.58\%$ ($m = 4$). On the other hand, the lowest decrease of the error

Table 4: Reinforcement learning: the full structure PNN classification error (E for L neurons), the PNN reduction ratio R , the final reduction error (E_R^* for C^* neurons) and the ranking score $RS = \{0, 1, 2, 3\}$; the values in the right most column are the decreases of the classification error after reduction.

Data set	E	$m = 2$		$m = 3$		$m = 4$		E_d [%]
		R	E_R^*	R	E_R^*	R	E_R^*	
cardiotocography	6.31 (0.14) 0	2.50	1.13 ⁺ (0.20) 1	2.50	1.11 ⁺ (0.09) 2	2.00	1.05 ⁺ (0.14) 3	83.36
Parkinson	9.69 (0.30) 1	1.42	9.25 (0.90) 3	2.50	9.44 (1.25) 2	2.50	10.35 (1.28) 0	4.54
Haberman	19.61 (0.52) 0	5.02	10.86 ⁺ (1.06) 3	5.02	13.45 ⁺ (1.26) 2	5.02	14.97 ⁺ (1.05) 1	44.62
Iris	1.20 (0.27) 0	5.00	0.00 ⁺ (0.00) 3	5.00	0.00 ⁺ (0.00) 3	5.00	0.00 ⁺ (0.00) 3	100.00
diabetes	24.14 (0.40) 0	4.99	20.96 ⁺ (1.36) 1	3.34	20.27 ⁺ (1.23) 2	4.99	19.28 ⁺ (1.10) 3	20.13
seeds	4.48 (0.71) 0	2.00	2.00 ⁺ (0.87) 2	2.50	2.66 ⁺ (1.10) 1	2.50	1.72 ⁺ (0.79) 3	61.61

is observed for the Haberman data set. In this case, $E_d = 28.61\%$, but such a result is obtained for the number of pattern neurons that are over 3 times lower ($m = 2$). Of note is also the fact that in the diabetes data set classification task, for all values of the fuzzifier m , almost 10 times less pattern neurons are required to achieve a better performance of the simplified PNN. The value of E_d is also remarkable since it is equal to 57.92% ($m = 4$).

In the case of Algorithm 1 applied to the PNN trained by means of the RL procedure, $E_R^* < E$ also holds in all classification problems (Table 4). Similarly, the ranking score computed for the full structure PNN is equal to 1 only once (Parkinson data set) while for the other data sets, $RS = 0$. In terms of all the attained results, the occurrence of $E_R^* = 0$ at $m = \{2, 3, 4\}$ for the Iris data set classification tasks is the most prominent observation. It is notable that this occurs when the number of the pattern neurons is 5 times smaller than that of the original PNN. A very good result is achieved for the cardiotocography data set as well. Here, $E_d = 83.36\%$ for the network whose number of neurons in the pattern layer is reduced from 2126 down to

Table 5: Plugin method: the full structure PNN classification error (E for L neurons), the PNN reduction ratio R , the final reduction error (E_R^* for C^* neurons) and the ranking score $RS = \{0, 1, 2, 3\}$; the values in the right most column are the decreases of the classification error after reduction.

Data set	E	$m = 2$		$m = 3$		$m = 4$		E_d [%]
		R	E_R^*	R	E_R^*	R	E_R^*	
cardiotocography	3.36	2.50	5.44 ⁺	3.33	6.25 ⁺	5.00	5.88 ⁺	–
	(0.11)		(0.38)		(0.24)		(0.23)	
	3		2		0		1	
Parkinson	9.66	2.50	7.68⁺	2.50	13.57 ⁺	1.99	12.60 ⁺	20.49
	(0.46)		(1.22)		(1.10)		(0.87)	
	2		3		0		1	
Haberman	28.59	2.51	19.97⁺	3.33	21.59 ⁺	1.66	23.88 ⁺	30.15
	(1.17)		(1.13)		(0.65)		(1.10)	
	0		3		2		1	
Iris	5.07	5.00	0.00⁺	5.00	0.00⁺	3.33	0.14 ⁺	100.00
	(0.33)		(0.00)		(0.00)		(0.37)	
	0		3		3		2	
diabetes	32.43	9.97	33.39	9.97	30.89 ⁺	9.97	30.62⁺	5.58
	(1.20)		(0.86)		(1.55)		(1.84)	
	1		0		2		3	
seeds	6.67	1.43	5.74 ⁺	1.67	5.31⁺	1.43	7.11 ⁺	20.39
	(0.30)		(0.80)		(0.64)		(0.56)	
	1		2		3		0	

1064 ($m = 4$). Only for the Parkinson data set ($m = 2$), both E_d and R are not impressive because they reach the values of 4.54% and 1.42, respectively. However, such outcomes still favor the proposed algorithm.

In contrast to previous examinations, the use of Algorithm 1 to simplify the PNN trained by the plugin method yields $E_R^* > E$ in a single classification case (Table 5). For the cardiotocography data set, the full structure PNN classification error has the lowest value among all compared results ($E = 3.36\%$); therefore, $RS = 3$. As in the case of the PNN trained by the RL procedure, $E_R^* = 0$ at $m = \{2, 3\}$ for the Iris data set classification problem. The size of the pattern layer is also 5 times smaller. It means that 120 neurons must be discarded to provide an ideal prediction for the network. The smallest decrease of the classification error after reduction occurs for the diabetes data set ($E_d = 5.58\%$), but it is worth emphasizing that it is observed when almost 10 times less neurons are required in the pattern layer.

7.3. Computational time

The improvement of the reduced PNN, except for the decrease of the classification error, should also be verified in terms of the run-time indicator. Tables 6, 7 and 8 show: t_{C^*} – the computational time needed to obtain the optimal number of centroids, t_{P^*} – the computational time required to complete the classification task by means of the PNN on the optimal set of reduced input vectors, and t_L – the computational time involved to train the full structure PNN for CG, RL and PM training procedures, respectively. All the results are provided in seconds for the optimal s^* index shown in Table 2 since for its particular values, the final reduction error E_R^* is determined. The simulations are conducted on a 64-bit Windows 10 Pro operating system with an Intel Core i7 2.7-GHz processor and 32-GB RAM.

Table 6: Conjugate gradients: the averaged computational times (in seconds) involved in: determining the optimal number of the centroids (t_{C^*}), training the PNN on the optimal set of the reduced input vectors (t_{P^*}), completing the classification task for the full structure PNN (t_L) in particular data set classification problems. All the values are shown for the optimal s^* index and $m = \{2, 3, 4\}$.

Data set	s^*	$m = 2$		s^*	$m = 3$		s^*	$m = 4$		t_L
		t_{C^*}	t_{P^*}		t_{C^*}	t_{P^*}		t_{C^*}	t_{P^*}	
cardiotocography	4	19.85	57.82	4	19.18	59.57	4	19.10	67.87	343.50
Parkinson	3	0.15	0.53	3	0.13	0.54	5	0.17	2.31	8.69
Haberman	3	0.21	0.36	3	0.18	0.27	2	0.15	0.21	2.47
Iris	3	0.04	0.08	6	0.07	0.55	6	0.06	0.26	1.26
diabetes	1	0.42	0.78	1	0.41	1.03	1	0.40	1.13	37.18
seeds	3	0.05	0.56	5	0.06	1.52	3	0.05	0.91	6.67

To confirm the need of the introduced architecture reduction of a PNN, the following inequality should be fulfilled

$$\forall(s^*, m) \quad t_{C^*} + t_{P^*} < t_L, \quad (17)$$

where the left side represents the total computational time required to train the simplified PNN while the right side is the original network training time. As shown in all Tables 6–8, the inequality (17) holds for all values of the s^* index and the m fuzzifier; therefore, the proposed reduction is by all means valid.

Table 7: Reinforcement learning: the averaged computational times (in seconds) involved in: determining the optimal number of the centroids (t_{C^*}), training the PNN on the optimal set of the reduced input vectors (t_{P^*}), completing the classification task for the full structure PNN (t_L) in particular data set classification problems. All the values are shown for the optimal s^* index and $m = \{2, 3, 4\}$.

Data set	$m = 2$			$m = 3$			$m = 4$			t_L
	s^*	t_{C^*}	t_{P^*}	s^*	t_{C^*}	t_{P^*}	s^*	t_{C^*}	t_{P^*}	
cardiotocography	4	19.85	1123.57	4	19.18	1241.17	5	23.73	1755.56	8198.21
Parkinson	7	0.21	18.51	6	0.19	14.15	4	0.16	6.46	36.41
Haberman	2	0.16	0.18	2	0.15	0.18	2	0.15	0.17	3.65
Iris	2	0.03	0.16	2	0.03	0.16	2	0.03	0.16	0.29
diabetes	2	0.74	4.25	3	1.05	9.24	2	0.78	4.31	102.37
seeds	5	0.06	2.62	4	0.05	1.73	4	0.04	1.74	9.73

Table 8: Plugin method: the averaged computational times (in seconds) involved in: determining the optimal number of the centroids (t_{C^*}), training the PNN on the optimal set of the reduced input vectors (t_{P^*}), completing the classification task for the full structure PNN (t_L) in particular data set classification problems. All the values are shown for the optimal s^* index and $m = \{2, 3, 4\}$.

Data set	$m = 2$			$m = 3$			$m = 4$			t_L
	s^*	t_{C^*}	t_{P^*}	s^*	t_{C^*}	t_{P^*}	s^*	t_{C^*}	t_{P^*}	
cardiotocography	4	19.85	179.01	3	14.72	99.38	2	9.94	44.59	1274.63
Parkinson	4	0.18	1.75	4	0.16	1.73	5	0.20	2.67	10.65
Haberman	4	0.22	0.83	3	0.18	0.49	6	0.22	1.76	3.52
Iris	2	0.03	0.11	2	0.03	0.12	3	0.04	0.16	0.69
diabetes	1	0.42	0.60	1	0.41	0.61	1	0.40	0.61	51.33
seeds	7	0.07	1.14	6	0.07	0.69	7	0.07	1.15	2.17

7.4. Evaluation of the performance of the proposed algorithm in comparison to existing approaches

As mentioned in section 1, ample attention has been paid to the structure reduction of the PNN with the use of k-means-based approaches. The solutions are frequently tested on UCI-MLR data sets. Therefore, to evaluate the results achieved by the proposed algorithm, the outcomes obtained by selected works that were previously published are also presented. The following methods are used in the comparison:

- clustered data-based PNN trained by means of an evolutionary algo-

rithm (Georgiou et al., 2008);

- global and fast global k-means-based PNN trained using the expectation-maximization method (Chang et al., 2008);
- k-means-based PNN trained with the conjugate gradient procedure (Kusy and Kluska, 2017);
- clustered noise-injected PNN (Mukherjee, 2017).

The results achieved by the above methods are collated in Table 9 along with the best outcomes provided by the proposed algorithm for each training method. It can be seen that for the cardiocography, Haberman and Iris

Table 9: The comparison of the lowest classification errors obtained by the reduced PNN trained by means of CG, RL, PM training procedures and the errors achieved by other solutions available in the literature.

Data set	Proposed algorithm			Existing approaches	
	CG	RL	PM	Result	Source
cardiocography	0.31	1.05	5.44	0.90	(Kusy and Kluska, 2017)
Haberman	16.52	10.86	19.97	22.20	(Kusy and Kluska, 2017)
Iris	0.25	0.00	0.00	4.29	(Chang et al., 2008)
diabetes	9.25	19.28	30.62	30.70	(Georgiou et al., 2008)
				31.00	(Chang et al., 2008)
				28.14	(Chang et al., 2008)
				9.10	(Kusy and Kluska, 2017)
				29.46	(Mukherjee, 2017)

data sets, it is possible to find at least one PNN training procedure that allows the proposed algorithm to provide a lower classification error. Only in the case of the diabetes database, Kusy and Kluska (2017) receive a better network's performance with a 0.15 error margin.

7.5. Summary

The results and detailed analysis presented in subsections 7.1–7.4 lead to the following general observations:

1. In comparison to the plugin method, the application of Algorithm 1 in the CG and RL training procedures provides much better reduction results for a PNN in which both the centroids and representative input vectors create the pattern layer.
2. In 43.75% of all obtained classification results, $E_R^* < E^*$. In particular, for:
 - (a) CG training: $E_R^* < E^*$ for two out of six data sets;
 - (b) RL training: $E_R^* < E^*$ for three out of six data sets; in a single case $E_R^* = E^*$;
 - (c) PM training: $E_R^* < E^*$ for two out of six data sets; in a single case $E_R^* = E^*$.

Such an outcome may suggest that it is enough to use the centroids to construct the pattern neurons of a PNN. However, the main objective of the current work, is to acquire original input vectors to build the pattern layer of the network.

3. On the basis of the ranking score, it is impossible to advise the best value of the m fuzzifier. If one accumulates RS values for each m separately across all classification cases, one obtains:
 - (a) CG training: the total RS is equal to 12 ($m = 2$), 13 ($m = 3$) and 10 ($m = 4$);
 - (b) RL training: the total RS is equal to 13 ($m = 2$), 12 ($m = 3$) and 13 ($m = 4$);
 - (c) PM training: the total RS is equal to 13 ($m = 2$), 10 ($m = 3$) and 8 ($m = 4$).
4. Excluding the case of the use of Algorithm 1 in the cardiocography data classification task, where no error reduction occurs within the PM training (Table 5), general comments on the reduction ratio are as follows:
 - (a) $R > 1$ in all presented results;
 - (b) $R \simeq 10$ for the PNN trained by means of the CG and PM procedures in the diabetes classification problem;

- (c) for each training procedure, the minimum and maximum values of R for $m = 3$ are similar to the corresponding values of R for $m = 4$.
- 5. The smallest two values of the decreases in the classification error after reduction are equal to only 4.54% and 5.58%. On the other hand, the largest decrease reaches 100%. In the remaining cases, $E_d > 20\%$.
- 6. For the PNN with pattern neurons established from centroids, in 49 out of 54 possible comparisons, the performed two sample t-tests indicate that the averaged values of E^* and E are significantly different.
- 7. For the PNN with pattern neurons established from reduced data, in 48 out of 54 possible comparisons, the performed two sample t-tests show that the averaged values of E_R^* and E are significantly different.
- 8. The total computational time involved in training a PNN on the optimal set of reduced input vectors is always smaller than the run-time of the original network.
- 9. The gain in computational time is the most prominent in the case of the diabetes data classification task; here, the reduced PNN operates much faster, in particular:
 - (a) CG training: 31.0 times ($m = 2$), 25.8 times ($m = 3$) and 24.3 times ($m = 4$);
 - (b) RL training: 20.5 times ($m = 2$), 9.9 times ($m = 3$) and 20.1 times ($m = 4$);
 - (c) PM training: 50.3 times ($m = 2$), 50.3 times ($m = 3$) and 50.8 times ($m = 4$).
- 10. In comparison to the results available in the literature, in all but one classification tasks, the proposed algorithm provides the lowest classification error.

It is also worth noting that Algorithm 1 can be very simply adjusted to enable selection of indices corresponding to two (or more) highest membership coefficients in $\mathbf{U}_j, j = 1, \dots, J$. This provides the possibility to identify two (or more) input vectors closest to a cluster center. The size of the pattern layer of the PNN will therefore grow, and its performance may change.

8. Conclusions

In this paper, a new algorithm for the reduction of a PNN's pattern layer was proposed. It relied on computing the centroids from the original input data by means of the FCM algorithm. The new set of the pattern neurons was established based on the input vectors that were nearest to the centroids in terms of a fuzzy similarity. Three different values of the m fuzzifier were chosen. Various PNN training procedures were selected for evaluation purposes, in particular: conjugate gradients, reinforcement learning and the plugin method. The algorithm was tested on six repository data classification tasks by collating the full structure PNN classification error, the minimum centroids-based error and the final reduction error, all obtained by means of a 10-fold cross validation procedure; the PNN reduction ratio and the decreases in the classification error after reduction were also computed. Furthermore, in order to make the comparison more comprehensive, a two sample t-test was performed to confirm the statistical significance of the results, the computational time of the algorithm was indicated and a comparison to the existing approaches was performed. As shown, the algorithm provided satisfactory outcomes in all the evaluated factors, making it useful in reduction tasks in general.

Acknowledgment

The work was supported by Rzeszow University of Technology, Department of Electronics Fundamentals Grant for Statutory Activity (DS 2018).

References

- Bache, K., Lichman, M., 2015. Uci machine learning repository. Tech. rep., School of Information and Computer Science, University of California, Irvine, CA, USA, School of Information and Computer Sciences.
URL <http://archive.ics.uci.edu/ml>
- Bezdek, J. C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell, MA, USA.
- Burrascano, P., 1990. Learning vector quantization for the probabilistic neural network. IEEE transactions on neural networks/a publication of the IEEE Neural Networks Council 2 (4), 458–461.

- Chang, R. K. Y., Loo, C. K., Rao, M., 2008. A global k-means approach for autonomous cluster initialization of probabilistic neural network. *Informatica* 32 (2).
- Chtioui, Y., Bertrand, D., Barba, D., 1996. Reduction of the size of the learning data in a probabilistic neural network by hierarchical clustering. application to the discrimination of seeds by artificial vision. *Chemometrics and Intelligent Laboratory Systems* 35 (2), 175–186.
- Chtioui, Y., Panigrahi, S., Marsh, R., 1998. Conjugate gradient and approximate newton methods for an optimal probabilistic neural network for food color classification. *Optical Engineering* 37 (11), 3015–3023.
- Dunn, J. C., 1973. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics* 3 (3), 32–57.
- Georgiou, V. L., Alevizos, P. D., Vrahatis, M. N., 2008. Novel approaches to probabilistic neural networks through bagging and evolutionary estimating of prior probabilities. *Neural Processing Letters* 27 (2), 153–162.
- Hartigan, J. A., Wong, M. A., 1979. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, 100–108.
- Kokkinos, Y., Margaritis, K. G., 2018. Simulating parallel scalable probabilistic neural networks via exemplar selection and em in a ring pipeline. *Journal of Computational Science* 25, 260–279.
- Kowalski, P. A., Kulczycki, P., 2016. A complete algorithm for the reduction of pattern data in the classification of interval information. *International Journal of Computational Methods* 13 (03), 1650018–1–1650018–26.
- Kowalski, P. A., Kulczycki, P., 2017. Interval probabilistic neural network. *Neural Computing and Applications* 28 (4), 817–834.
- Kowalski, P. A., Kusy, M., 2018. Sensitivity analysis for probabilistic neural network structure reduction. *IEEE Transactions on Neural Networks and Learning Systems* 29 (5), 1919–1932.
- Kusy, M., Kluska, J., 2013. Probabilistic neural network structure reduction for medical data classification. In: *Artificial Intelligence and Soft Computing*. Springer, pp. 118–129.

- Kusy, M., Kluska, J., 2017. Assessment of prediction ability for reduced probabilistic neural network in data classification problem. *Soft Computing* 21 (1), 199–212.
- Kusy, M., Zajdel, R., 2014. Probabilistic neural network training procedure based on $q(0)$ -learning algorithm in medical data classification. *Applied Intelligence* 41 (3), 837–854.
- Kusy, M., Zajdel, R., 2015. 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* 26 (9), 2163–2175.
- Luzanin, O., Plancak, M., 2014. Hand gesture recognition using low-budget data glove and cluster-trained probabilistic neural network. *Assembly Automation* 34 (1), 94–105.
- Mao, K., Tan, K.-C., Ser, W., July 2000. Probabilistic neural-network structure determination for pattern classification. *Neural Networks, IEEE Transactions on* 11 (4), 1009–1016.
- Mukherjee, S., 2017. Improving generalization of k-means clustering based probabilistic neural network using noise injection. In: *International Conference on Soft Computing and its Engineering Applications*. IEEE, pp. 1–5.
- Sherrod, P. H., 2017. Dtreg predictive modelling software.
URL <http://www.dtreg.com>
- Specht, D. F., 1990. Probabilistic neural networks. *Neural Networks* 3 (1), 109–118.
- Wand, M. P., Jones, M. C., 1994. *Kernel smoothing*. Crc Press.
- Zaknich, A., 1997. A vector quantisation reduction method for the probabilistic neural network. In: *Neural Networks, 1997., International Conference on*. Vol. 2. IEEE, pp. 1117–1120.