Appl Intell (2014) 41:837–854
DOI 10.1007/s10489-014-0562-9

Probabilistic neural network training procedure based on $Q(0)$-learning algorithm in medical data classification

Maciej Kusy · Roman Zajdel

Published online: 6 August 2014

Abstract In this article, an iterative procedure is proposed for the training process of the probabilistic neural network (PNN). In each stage of this procedure, the $Q(0)$-learning algorithm is utilized for the adaptation of PNN smoothing parameter (σ). Four classes of PNN models are regarded in this study. In the case of the first, simplest model, the smoothing parameter takes the form of a scalar; for the second model, σ is a vector whose elements are computed with respect to the class index; the third considered model has the smoothing parameter vector for which all components are determined depending on each input attribute; finally, the last and the most complex of the analyzed networks, uses the matrix of smoothing parameters where each element is dependent on both class and input feature index. The main idea of the presented approach is based on the appropriate update of the smoothing parameter values according to the $Q(0)$-learning algorithm. The proposed procedure is verified on six repository data sets. The prediction ability of the algorithm is assessed by computing the test accuracy on 10 %, 20 %, 30 %, and 40 % of examples drawn randomly from each input data set. The results are compared with the test accuracy obtained by PNN trained using the conjugate gradient procedure, support vector machine algorithm, gene expression programming classifier, k–Means method, multilayer perceptron, radial basis function neural network and learning vector quantization neural network. It is shown that the presented procedure can be applied to the automatic adaptation of the smoothing parameter of each of the considered PNN models and that this is an alternative training method. PNN trained by the $Q(0)$-learning based approach constitutes a classifier which can be treated as one of the top models in data classification problems.

Keywords Probabilistic neural network · Smoothing parameter · Training procedure · $Q(0)$-learning algorithm · Reinforcement learning · Accuracy

M. Kusy (✉) · R. Zajdel
Faculty of Electrical and Computer Engineering,
Rzeszow University of Technology, al. Powstancow Warszawy 12,
35-959 Rzeszow, Poland
e-mail: mkusy@prz.edu.pl

R. Zajdel
e-mail: rzajdel@prz.edu.pl

1 Introduction

Probabilistic neural network (PNN) is an example of the radial basis function based model effectively used in data classification problems. It was proposed by Donald Specht [37, 38] and, as the data classifier, draws the attention of researchers from the domain of data mining. For example, it is applied in medical diagnosis and prediction [23, 25, 28], image classification and recognition [7, 20, 27], bearing fault detection [32], digital image watermarking [45], earthquake magnitude prediction [1] or classification in a time-varying environment [31].

PNN is a feed-forward neural network with a complex structure. It is composed of an input layer, a pattern layer, a summation layer and an output layer. Despite its complexity, PNN only has a single training parameter. This is a smoothing parameter of the probability density functions (PDFs) which are utilized for the activation of the neurons in the pattern layer. Thereby, the training process of PNN solely requires a single input-output signal pass in order to compute network response. However, only the optimal value of the smoothing parameter gives the possibility of correctness of the model's response in terms of generalization ability.

The value of σ must be estimated on the basis of the PNN's classification performance which is usually achieved in an iterative manner.

Within the process of the smoothing parameter estimation two issues must be addressed. The first one pertains to the selection of σ in PDF for the pattern layer neurons of PNN. Four possible approaches are applied, i.e. a single parameter for the whole model [37, 38], a single parameter for each class [1], a single parameter for each data attribute [11, 39], and a single parameter for each attribute and a class [7, 11, 12].

The second problem related to the smoothing parameter estimation for PNN is concerned with the computation of the σ value. In literature, different procedures have been developed. For example, in [39], the conjugate gradient descent (ascent) is used to find iteratively the set of σ's which maximize the optimization criterion. Chtioui et al [7] exploit the conjugate gradient method and the approximate Newton algorithm to determine the smoothing parameters associated with each data attribute and class. In [12], the authors utilize the particle swarm optimization algorithm to estimate the matrix of the smoothing parameters for the probability density functions in the pattern layer. An interesting study is presented in [47], where the gap-based approach for smoothing parameter adaptation is proposed. The authors provide the formula for σ on the basis of the gap computed between the two nearest points of the data set. The solution is applied to PNN for which the smoothing parameter takes the form of a scalar and the vector whose parameters are associated with each data feature.

As one can observe, the choice of the smoothing parameter plays a crucial role in the training process of the probabilistic neural network. This fact is of particular importance when PNN has different σ for: each class, each attribute, and each class and attribute. The task of smoothing parameter selection can then be considered as a high-dimensional function optimization problem. The reinforcement learning (RL) algorithm is an efficient method in solving such type of problems, e.g. finding extrema of some family of functions [46] or the computation of the set of optimal weights for multilayer perceptron [40]. The RL method is also frequently applied in various engineering tasks. It is used in nonstationary serial supply chain inventory control [18], adaptive control of nonlinear objects [43], adjusting robot behavior for autonomous navigation system [26] or path planning for improving positioning accuracy of a mobile microrobot [22]. There are also studies which propose the use of RL in non-technical domains, e.g. in the explanation of dopamine neuronal activity [5] or in an educational system to improve the pedagogical policy [16].

In this work, we introduce a novel procedure for the computation of the smoothing parameter of the PNN model. This procedure uses the $Q(0)$-learning algorithm. The method adjusts the smoothing parameter according to four different strategies: single σ for the whole network, single σ for each class, single σ for each data attribute and single σ for each data attribute and each class. The results of our proposed solution are compared to the outcomes of PNN for which the smoothing parameter is calculated using the conjugate gradient procedure and, additionally, to the support vector machine classifier, gene expression programming algorithm, k–Means clustering method, multilayer perceptron, radial basis function neural network and learning vector quantization neural network in medical data classification problems.

The authors of the present study have already proposed the application of the reinforcement learning algorithm to the computation of the smoothing parameter of radial basis function based neural networks [19]. In that work, the stateless Q-learning algorithm was used for the adaptive computation of the smoothing parameter of the networks.

This paper is organized as follows. Section 2 discusses the probabilistic neural network highlighting its basics, structure, principle of operation and the problem of smoothing parameter selection. Section 3 presents the basis of one of the reinforcement learning algorithms which is applied in this work, namely the $Q(0)$-learning algorithm. In Section 4, we present the proposed procedure. Here the problem statement is provided, a general idea of applying the $Q(0)$-learning algorithm to the choice of the smoothing parameter is described and, finally, the details of the algorithm are given. Section 5 presents the data sets used in this research, the algorithm settings and the obtained empirical results along with the illustration of the PNN training process. In this part of the work, we compare the performance of our method with the efficiency of the PNN whose σ is determined by means of the conjugate gradient method and, additionally, to the efficiency of the reference classifiers and neural networks. Finally, in Section 6, we conclude our work.

2 Probabilistic neural network

Probabilistic neural network is a data classification model which implements the Bayesian decision rule. This rule is defined as follows. If we assume that: (1) there is a data pattern $\mathbf{x} \in \mathbb{R}^n$ which is included in one of the predefined classes $g = 1, \ldots, G$; (2) the probability of $\mathbf{x}$ belonging to the class g equals p_g; (3) the cost of classifying $\mathbf{x}$ into class g is c_g; (4) the probability density functions $y_1(\mathbf{x}), y_2(\mathbf{x}), \ldots, y_G(\mathbf{x})$ for all classes are known. Then, according to the Bayes theorem, when $g \neq h$, the vector $\mathbf{x}$ is classified to the class g, if $p_g c_g y_g(\mathbf{x}) > p_h c_h y_h(\mathbf{x})$. Usually $p_g = p_h$ and $c_g = c_h$, thus if $y_g(\mathbf{x}) > y_h(\mathbf{x})$, the vector $\mathbf{x}$ is classified to the class g.

In real data classification problems, any knowledge on the probability density functions $y_g(\mathbf{x})$ is not given since a data set distribution is usually unknown. Therefore, some approximation of the PDF must be determined. Such an approximation can be obtained using the Parzen method [29]. Commonly, the Gaussian function is a choice for PDF since it satisfies the conditions required by Parzen's method.

The assumption of using the Gaussian density for PDF gives the possibility of constructing a feed-forward classifier. It is composed of the input layer represented by the attributes of $\mathbf{x}$, the pattern layer and the summation layer consisting of G neurons where each one computes the signal only for patterns which belong to g-th class

$$y_g(\mathbf{x};\sigma) = \frac{1}{l_g (2\pi)^{n/2} \sigma^n} \sum_{i=1}^{l_g} \exp\left(-\sum_{j=1}^{n} \frac{\left(x_{ij}^{(g)} - x_j\right)^2}{2\sigma^2} \right), \tag{1}$$

where l_g is the number of examples of class g, σ denotes the smoothing parameter, $x_{ij}^{(g)}$ is the j-th element of the i-th training vector ($i = 1, \ldots, l_g$) which is contained in the class g and x_j is the j-th coordinate of the unknown vector $\mathbf{x}$. Finally, the output layer estimates the class of $\mathbf{x}$ in accordance with the Bayes's decision rule based on the outputs of all the summation layer neurons

$$G^*(\mathbf{x}) = \arg\max_g \left\{ y_g(\mathbf{x}) \right\}, \tag{2}$$

where $G^*(\mathbf{x})$ denotes the predicted class of the pattern $\mathbf{x}$. Since y_g defined in (1) depends on scalar σ, this type of PNN is henceforth named PNNS. The architecture of the probabilistic neural network is depicted in Fig. 1.

If we consider that the patterns of particular classes differ in their densities, then the summation layer signal defined in (1) has a different shape depending on the value of the smoothing parameter in relation to the class (such a model is called PNNC)

$$y_g(\mathbf{x};\boldsymbol{\sigma}_C) = \frac{1}{l_g (2\pi)^{n/2} \left(\sigma^{(g)}\right)^n} \sum_{i=1}^{l_g} \exp\left(-\sum_{j=1}^{n} \frac{\left(x_{ij}^{(g)} - x_j\right)^2}{2\left(\sigma^{(g)}\right)^2} \right), \tag{3}$$

where $\boldsymbol{\sigma}_C = \left[\sigma^{(1)}, \ldots, \sigma^{(G)}\right]^T$ is the smoothing parameter vector consisting of $\sigma^{(g)}$ elements associated with g-th class.

It is also possible to differentiate the smoothing parameter with respect to each attribute of the input data. In such a case, the formula in (1) takes the following form

$$y_g(\mathbf{x};\boldsymbol{\sigma}_V) = \frac{1}{l_g (2\pi)^{n/2} \prod_{j=1}^{n} \sigma_j} \sum_{i=1}^{l_g} \exp\left(-\sum_{j=1}^{n} \frac{\left(x_{ij}^{(g)} - x_j\right)^2}{2\sigma_j^2} \right), \tag{4}$$

where $\boldsymbol{\sigma}_V = [\sigma_1, \ldots, \sigma_n]$ is the smoothing parameter vector consisting of σ_j elements associated with the j-th input variable. PNN with the smoothing parameters different for each variable is denoted as PNNV.

Finally, if one regards a PNN model, whose smoothing parameter is different for each data variable and each class, the network's summation layer signal can be expressed in the most general form (such a model is named PNNVC)

$$y_g(\mathbf{x};\boldsymbol{\sigma}_{VC}) = \frac{1}{l_g (2\pi)^{n/2} \prod_{j=1}^{n} \sigma_j^{(g)}} \sum_{i=1}^{l_g} \exp\left(-\sum_{j=1}^{n} \frac{\left(x_{ij}^{(g)} - x_j\right)^2}{2\left(\sigma_j^{(g)}\right)^2} \right), \tag{5}$$

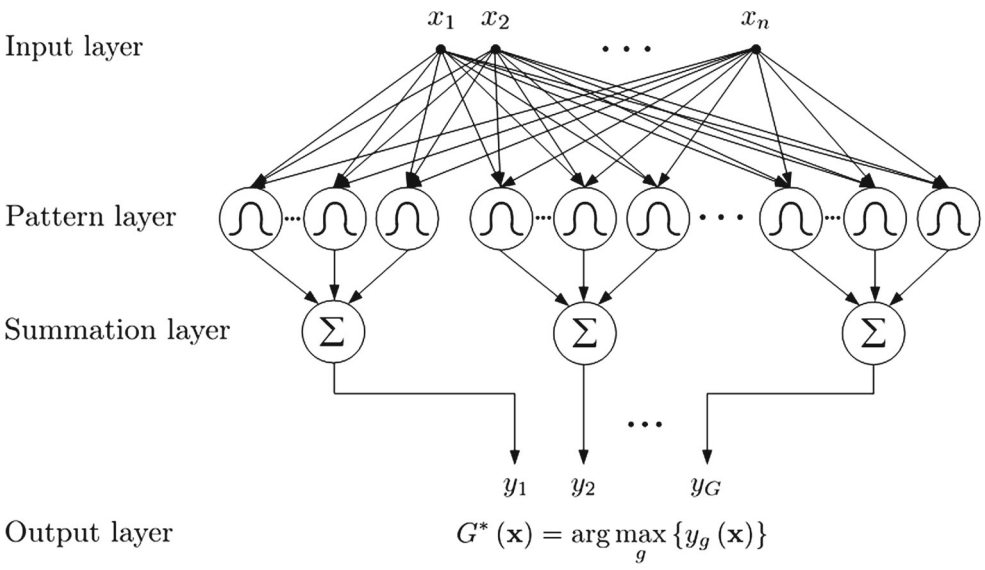

Fig. 1 The architecture of the probabilistic neural network

where

$$\boldsymbol{\sigma}_{VC} = \begin{bmatrix} \sigma_1^{(1)}, & \ldots, & \sigma_j^{(1)}, & \ldots, & \sigma_n^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_1^{(g)}, & \ldots, & \sigma_j^{(g)}, & \ldots, & \sigma_n^{(g)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_1^{(G)}, & \ldots, & \sigma_j^{(G)}, & \ldots, & \sigma_n^{(G)} \end{bmatrix} \tag{6}$$

is the matrix of the smoothing parameters where each $\sigma_j^{(g)}$ element is associated with the j-th input variable and g-th class.

Taking into account the four above possibilities of computing summation layer signal, the PNN output defined in (2) is generalized to the following form

$$G^*(\mathbf{x}; sigma) = \arg\max_g \left\{ y_g(\mathbf{x}; sigma) \right\}, \tag{7}$$

where

$$sigma = \begin{cases} \sigma & \text{for PNNS} \\ \boldsymbol{\sigma}_C & \text{for PNNC} \\ \boldsymbol{\sigma}_V & \text{for PNNV} \\ \boldsymbol{\sigma}_{VC} & \text{for PNNVC} \end{cases}, \tag{8}$$

where y_g is computed according to (1), (3), (4) and (5) for PNNS, PNNC, PNNV and PNNVC, respectively.

Figure 2 shows the difference between the PNN models expressed in terms of the smoothing parameter selection and computation for the summation neuron output signal y_g defined by the formulas (1), (3), (4) and (5). In this figure, we can see five points in $\mathbb{R}^2$ space which belong to two classes. The summation layer signals y_1 and y_2 are only marked for PNNS. One can observe that a single σ for the whole network (PNNS) does not enable the consideration of the input data densities of each class. Different shapes of PDFs for a PNNC model allow the dispersion of data of each class to be taken into account. When the smoothing parameter is different for each variable (PNNV), the particular PDFs take an elliptical form. This approach does not consider the input data densities of each class but provides information about the influence of input attribute values on y_g. Finally, PNNVC is the network which integrates data classes densities and the influence of the particular input features. This is the most general form of the model since PNNS, PNNC and PNNV are special cases of PNNVC.

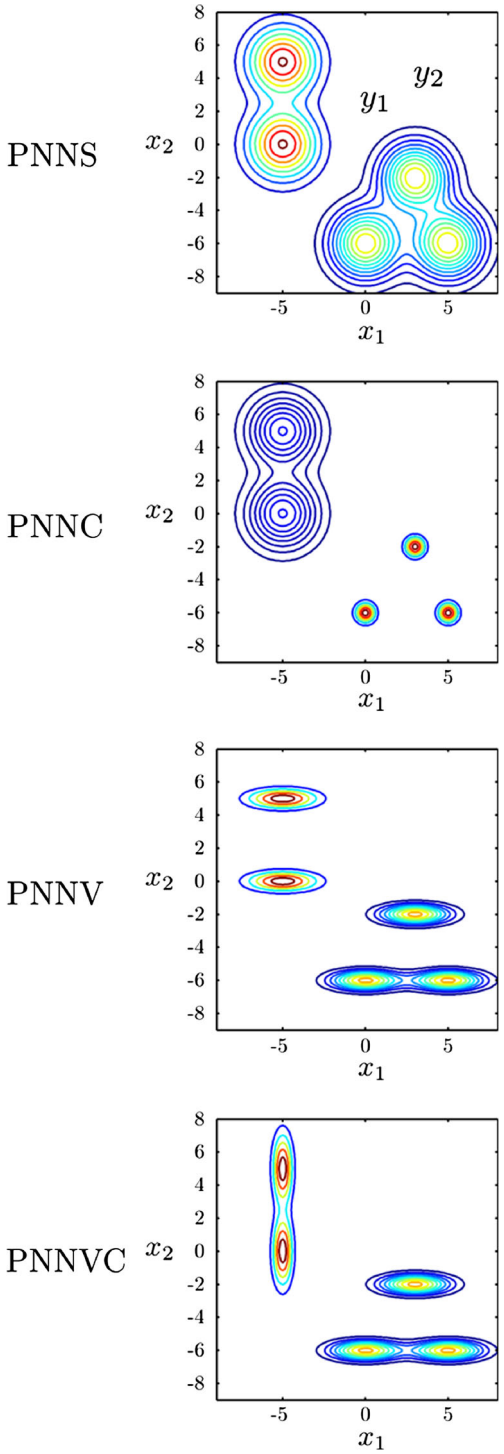

Fig. 2 The summation neuron signals y_1 and y_2 of PNNS, PNNC, PNNV and PNNVC for $\sigma = 1.4$, $\boldsymbol{\sigma}_C = \begin{bmatrix} 1.4 \\ 0.4 \end{bmatrix}$, $\boldsymbol{\sigma}_V = [1.4, 0.4]$ and $\boldsymbol{\sigma}_{VC} = \begin{bmatrix} 1.4, & 0.4 \\ 0.4, & 1.4 \end{bmatrix}$ respectively. Graphical interpretation of the signals is shown for five exemplary points of two classes: $\mathbf{x}_1^{(1)} = [-5, 5]$, $\mathbf{x}_2^{(1)} = [-5, 0]$, $\mathbf{x}_1^{(2)} = [0, -6]$, $\mathbf{x}_2^{(2)} = [5, -6]$ and $\mathbf{x}_3^{(2)} = [3, -2]$

3 Reinforcement learning

3.1 Introduction

Reinforcement learning addresses the problem of the agent that must learn to perform a task through a trial and error

interaction with an unknown environment. The agent and the environment interact continuously until the terminal state is reached. The agent senses the environment and selects an action to perform. Depending on the effect of its action, the agent obtains a reward. Its goal is to maximize the discounted sum of future reinforcements r_t received in the long run in any time step t, which is usually formalized as $\sum_{t=0}^{\infty} \gamma^t r_t$, where $\gamma \in [0, 1]$ is the agent's discount rate [41].

The mathematical model of the reinforcement learning method is a Markov Decision Process (MDP). MDP is defined as the quadruple $\langle S, A, P^{a_t}_{s_t s_{t+1}}, r_t \rangle$ where S is a set of states, A is a set of actions, $P^{a_t}_{s_t s_{t+1}}$ denotes the probability of the transition to the state $s_{t+1} \in S$ after the execution of the action $a_t \in A$ in the state $s_t \in S$.

3.2 $Q(0)$-learning

Different types of reinforcement learning algorithms exist. The $Q(0)$-learning proposed by Watkins [44] is one of the most often used. This algorithm computes the table of all $Q(s, a)$ values (called Q–table) by successive approximations. $Q(s, a)$ represents the expected pay-off that an agent can obtain in state s after it performs action a. In time step t, the Q–table is updated for the state-action pair (s_t, a_t) according to the following formula [44]

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left(r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \right), \tag{9}$$

where the maximization operator refers to the action value a which may be performed in the next state s_{t+1} and $\alpha \in (0, 1]$ is the learning rate. The formula (9) will be used as the basis of the algorithm for the PNN's smoothing parameter optimization presented in the next section.

4 Application of $Q(0)$-learning based procedure to the adaptation of PNN's smoothing parameter

4.1 Problem statement

Assume, we are given a training set in the form of the pairs $\langle \mathbf{x}_i, \hat{G}_i \rangle$, $i = 1, \ldots, l$, where $\mathbf{x}_i$ is the i-th input element and $\hat{G}_i$ is its corresponding output. Assume furthermore the following measure of the accuracy

$$Acc(sigma) = \frac{1}{l_T} \sum_{i=1}^{l_T} c_i(sigma), \tag{10}$$

where l_T is the cardinality of a training set, and c_i is the indicator of the classification's correctness defined as follows

$$c_i(sigma) = \begin{cases} 1 & \text{if } G^*(\mathbf{x}_i; sigma) = \hat{G}_i \\ 0 & \text{if } G^*(\mathbf{x}_i; sigma) \neq \hat{G}_i \end{cases}, \tag{11}$$

where $G^*(\mathbf{x}_i; sigma)$ is defined in (7).

The task is to find the optimal value of the smoothing parameter which maximizes the accuracy (10). For PNNC, PNNV and PNNVC models, this is a multivalued optimization problem. As the solution, we propose a new procedure, which is based on the $Q(0)$-learning algorithm. The set of system states S, the set of actions A and the reinforcement signal r which are required by the $Q(0)$-learning method will be defined along with the description of the algorithm.

4.2 General idea

For the adaptation of the smoothing parameter, the procedure based on the $Q(0)$-learning algorithm is proposed for PNNS, PNNC, PNNV and PNNVC models. The introduction of the $Q(0)$-learning algorithm is based on the assumption that in the PNN training process it is possible to distinguish two elements which interact with each other: the environment and the agent. The environment is composed of the data set used for the training process, the PNN model and the accuracy measure. The agent, on the basis of the policy which is represented by the action value function Q, chooses an action a_t in a state s_t. The action a_t is used to modify the smoothing parameter. In this work, the state is represented by the accuracy measure. This has some natural interpretation since the state defined in such a way is the function of PNN output, which depends on the smoothing parameter. The output of PNN is computed for the training and test set in order to determine the training and test accuracies. On the basis of the training accuracy, the next state s_{t+1} and the reinforcement signal r_t are computed. The reinforcement signal provides information about the change of the training accuracy taking the negative value when the accuracy decreases and the positive value when the accuracy increases. The effect of interaction between the agent and the environment results in both the modification of the action value function Q, and the change of the smoothing parameter.

The main assumption of the proposed procedure is to perform the training of the PNN model on the training set in order to maximize the training accuracy (10). Additionally, PNN is tested by computing the accuracy on the test set. The highest test accuracy and its corresponding value of the smoothing parameter is stored. Finding the highest

test accuracy of PNN will provide the optimal smoothing parameter in terms of the prediction ability.

The proposed procedure for the adaptation of the smoothing parameter of the network is illustrated in the form of the flowchart in Fig. 3. We only present the application of the procedure for the PNNV model, but it performs in a comparable manner for the remaining networks taking the cardinality of the smoothing parameters into account.

As shown, the procedure consists of M stages. In the first stage, the smoothing parameter vector $\boldsymbol{\sigma}_V$ of PNNV is initialized with ones. Such an initialization is proposed in the exemplary experiments in [8]. The actions from the set $A^{(1)}$ are assigned some values which should be large. The definition of the action along with the details concerning action set selection will be provided later in subsection 4.3. Then the model is trained on the training set using the proposed

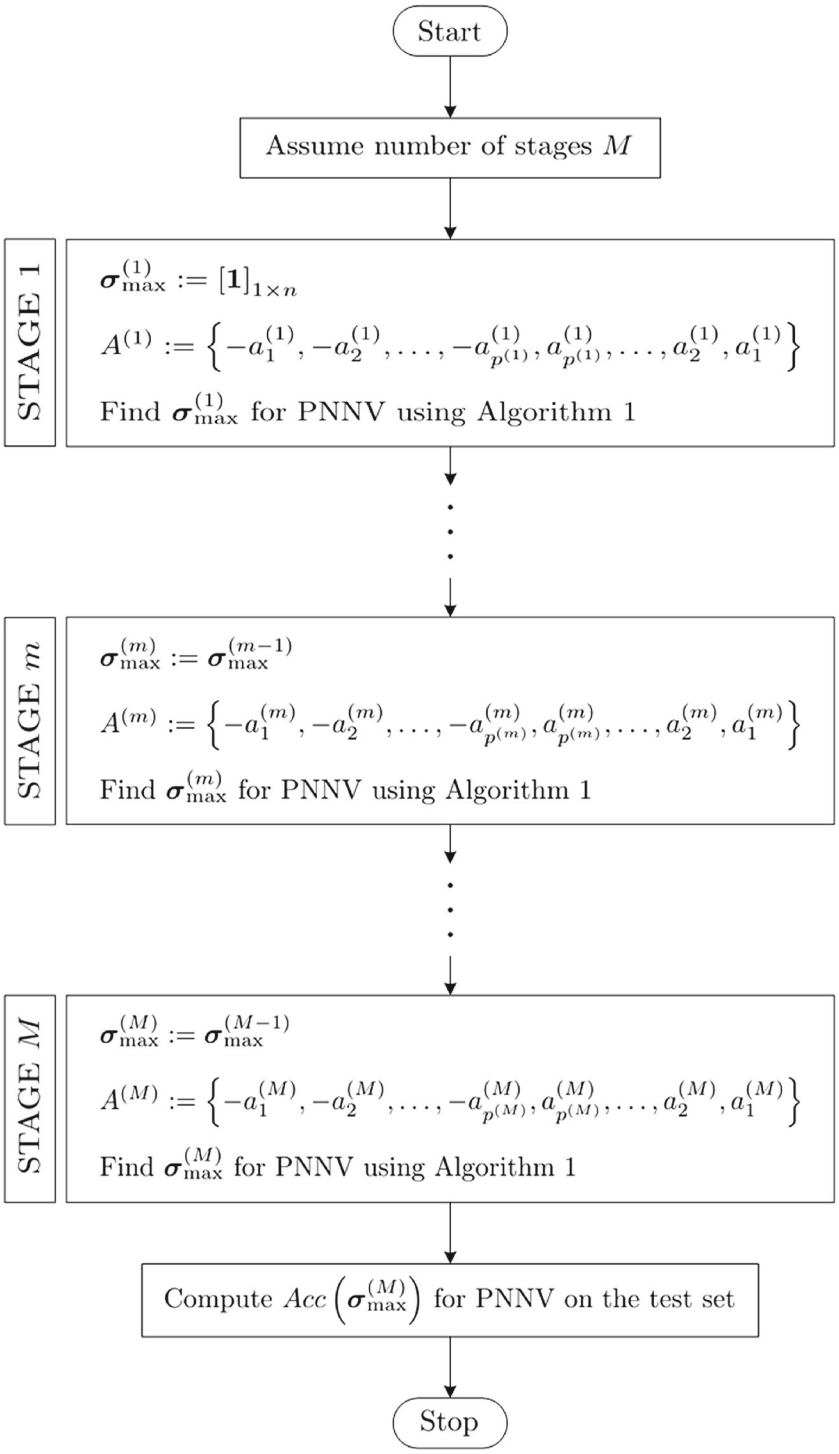

Fig. 3 The iterative procedure of the training process for the PNNV model

Algorithm 1, which will be explained later in detail, finds the smoothing parameter vector $\boldsymbol{\sigma}_{\max}^{(1)}$ which maximizes the training accuracy. Algorithm 1 is performed for each time step t and in short, for m-th stage of the procedure, consists of the following steps:

- choose a_t action using an actual policy derived from the action value function Q;
- update a single element of $\boldsymbol{\sigma}_V$ with the value of a_t;
- compute training and test accuracy Acc_t according to (10);
- actualize the maximal test accuracy $Acc_{\max}$ and the corresponding $\boldsymbol{\sigma}_{\max}^{(m)}$;
- calculate the reinforcement signal r_t on the basis of training accuracy;
- update the action value function Q.

Once the first stage of the procedure is completed, the second one begins. Here, $\boldsymbol{\sigma}_{\max}^{(2)}$ is initialized with the optimal value from the previous stage and the action set is changed. In our approach, this change relies on the decrease of all action values by an order of magnitude. The PNNV model is trained using Algorithm 1 and the smoothing parameter vector which maximizes the training accuracy is updated.

The procedure is performed M-times, each time: (1) – updating $\boldsymbol{\sigma}_{\max}^{(m)}$ on the basis of $\boldsymbol{\sigma}_{\max}^{(m-1)}$, (2) – decreasing the absolute values of the actions and (3) – finding new smoothing parameter values which maximize training accuracy. Such a type of approach, where the initial absolute values of the actions are large, gives the possibility of the selection of the smoothing parameter values within a broad range. The iterative decrease of actions in subsequent stages makes $\boldsymbol{\sigma}_{\max}^{(m)}$ narrow its values. This, in turn, allows us to search for a more optimal parameter of the PNNV model.

Once all M stages are performed, the highest test accuracy $Acc_{\max}$ is computed for $\boldsymbol{\sigma}_{\max}^{(M)}$. Such a solution provides the highest prediction ability of PNNV.

As shown, the above procedure utilizes RL in the problem of smoothing parameter adjustment to perform a classification task. However, it is also possible to combine PNN and RL in the other way. In the work of Heinen and Engel [15], a new incremental probabilistic neural network (IPNN) is proposed. The matrix of the smoothing parameters of IPNN is used for the action selection in the RL problem. IPNN is therefore utilized as the action value function approximator.

4.3 Application of the $Q(0)$-learning algorithm to adaptive computation of the smoothing parameter

In this subsection, we explain the details concerning the application of the $Q(0)$-learning algorithm to the problem of $\boldsymbol{\sigma}_V$ adaptive selection for the PNNV classifier. As mentioned before, the algorithm is solely highlighted for this type of network since $Q(0)$-learning works in a similar manner for PNNS, PNNC and PNNVC. The only difference is related to the number of the smoothing parameters which have to be updated. For PNNV, there is n parameters of the model while for PNNS, PNNC and PNNVC, there exist 1, G and $n \times G$ smoothing parameters, respectively.

The use of the $Q(0)$-learning algorithm for the choice of $\boldsymbol{\sigma}_V$ parameter requires the definition of the set of the system states, the action set and the reinforcement signal.

Definition 1 The set of system states is defined by the accuracy measure: $S = \left\{0, \frac{1}{l_T}, \frac{2}{l_T}, \ldots, \frac{l_T-1}{l_T}, 1\right\}$. S takes the real values from the interval [0, 1]. The total number of states is therefore $l_T + 1$.

Definition 2 $A^{(m)}$ is the symmetric set of actions of the following form: $A^{(m)} = \left\{-a_1^{(m)}, -a_2^{(m)}, \ldots, -a_{p^{(m)}}^{(m)}, a_{p^{(m)}}^{(m)}, \ldots, a_2^{(m)}, a_1^{(m)}\right\}$ where $p^{(m)}$ denotes the half of the cardinality of this set in stage m of the procedure.

Since $p^{(m)}$ in each iteration of the proposed procedure may be different, the cardinality of $A^{(m)}$ may vary and equals $2p^{(m)}$. The action set should be chosen so that $\max\left(A^{(1)}\right) > \ldots > \max\left(A^{(m)}\right) > \ldots > \max\left(A^{(M)}\right)$ holds. In our work, we assume the number of stages to be $M = 3$ which provides three action sets. For each m-th set, the following action values are proposed

$$\begin{aligned} A^{(1)} &= \{-10, -1, -0.1, 0.1, 1, 10\}, \\ A^{(2)} &= \{-1, -0.1, -0.01, 0.01, 0.1, 1\}, \\ A^{(3)} &= \{-0.1, -0.01, -0.001, 0.001, 0.01, 0.1\}. \end{aligned} \tag{12}$$

In each stage of the procedure, the smoothing parameters of PNNV are increased or decreased by the element values of $A^{(m)}$ action set. The proposition of the action set in the first stage ($A^{(1)}$) allows the modification of $\boldsymbol{\sigma}_V$ with large values. This gives the possibility of searching optimal parameters inside a broad range of values. Maximally, the elements of $\boldsymbol{\sigma}_V$ can be modified by the value of ± 10. The first stage of the procedure ends up with finding a candidate for optimum of the smoothing parameter. Subsequent decreases of absolute values of actions in $A^{(2)}$, shrink the domain of possible optimal parameter values. Finally, in $A^{(3)}$, the absolute values of the actions are so small so that the smoothing parameters of PNNV slightly change. A large change of $\boldsymbol{\sigma}_V$ in the third stage of the procedure is not required because the optimal modification route has already been established (in the first two stages).

In order to maximize the training accuracy of PNNV, the actual reinforcement signal r_t should reward an agent when the training accuracy increases and punish an agent when the accuracy decreases. This idea can be simply formalized as follows.

Definition 3 For the accuracy computed on the training set in the actual and previous step $Acc_t^{train}\left(\boldsymbol{\sigma}_{V,t}\right)$ and $Acc_{t-1}^{train}\left(\boldsymbol{\sigma}_{V,t-1}\right)$, the reinforcement signal is realized as follows

$$r_t = Acc_t^{train}\left(\boldsymbol{\sigma}_{V,t}\right) - Acc_{t-1}^{train}\left(\boldsymbol{\sigma}_{V,t-1}\right). \tag{13}$$

Since the training accuracy is normalized, $r_t \in [-1, 1]$.

Such a form of the reinforcement signal combined with the action value function update strengthens the confidence if the choice of an action is beneficial or not.

Algorithm 1 shows the application of the $Q(0)$-learning method to the adaptive choice of $\boldsymbol{\sigma}_V$ for the PNNV classifier. This algorithm is executed in each m-th stage of the procedure shown in Fig. 3.

The algorithm starts with the initialization of $Acc_{\max}$ on the basis of the smoothing parameter values found in the previous stage of the procedure except from the first stage, when $\boldsymbol{\sigma}_V$ is initialized with ones. $Acc_{\max}$ will store the maximal test accuracy computed on the test set during the training process. Then, in step **2**, the action value function Q is set to zero.

The main loop begins in step **4**, which runs over the maximum number of training steps $t_{\max}$. Since the PNNV training process is considered, in step **5** the inner loop begins which iterates over the number of input features. At the beginning of this loop, the actual state s_t is observed on the basis of the training accuracy (step **6**). Next, on the basis of the Q–table values, the actual action a_t is chosen at the state s_t using the ε-greedy method. Then, in step **8**, the smoothing parameter is updated by adding the value of the action a_t as follows

$$\sigma_{j,t} = \sigma_{j,t-1} + a_t. \tag{14}$$

Modification of $\sigma_{j,t}$ throughout the addition of the action value allows us to find the optimal smoothing parameter within the range determined by the extreme values of $A^{(m)}$ multiplied by the maximum number of training steps $t_{\max}$. Once a new value of the smoothing parameter is determined, the training accuracy $Acc_{j,t}^{train}\left(\boldsymbol{\sigma}_{V,t}\right)$ is calculated (step **9**), which then (step **10**) becomes the state of the system in $t+1$ time step. Next, the test accuracy $Acc_{j,t}^{test}\left(\boldsymbol{\sigma}_{V,t}\right)$ is computed on the test set. If an actual test accuracy is greater than the maximum one, both $\boldsymbol{\sigma}_{\max}^{(m)}$ and $Acc_{\max}$ are updated (steps **12–15**). Afterwards, the reinforcement signal is calculated using (13) and the actualization of the action value function is performed (steps **16** and **17**, respectively). Finally, if the current training accuracy reaches the value of 1, the algorithm stops and the next step of the procedure begins. If the algorithm is not able to find the most optimal solution ($Acc_{j,t}^{train}\left(\boldsymbol{\sigma}_{V,t}\right) < 1$), the condition in step **18** is never fulfilled. In such a case, $(m+1)$-th stage of the procedure starts after $t_{\max}$ training steps of PNNV has been performed.

Algorithm 1 The proposed algorithm of smoothing parameter adaptation of PNNV model with the use of $Q(0)$-learning method for m-th stage of the procedure

1 $Acc_{\max} := Acc\left(\boldsymbol{\sigma}_{\max}^{(m)}\right)$ according to (10) for test set
2 Initialize $Q_{j,0}(s,a) := 0$ for $j := 1,\ldots,n$, $s \in S$ and $a \in A^{(m)}$
3 Assume the maximum number of training steps $t_{\max}$
4 **for** $t := 1$ **to** $t_{\max}$ **do**
5 **foreach** j **do**
6 $s_t := Acc_{j,t-1}^{train}\left(\boldsymbol{\sigma}_{V,t-1}\right)$
7 Choose action a_t at state s_t using policy derived from $Q_{j,t-1}$ (e.g. ε-greedy$(s_t, Q_{j,t-1})$)
8 $\sigma_{j,t} := \sigma_{j,t-1} + a_t$
9 Calculate $Acc_{j,t}^{train}\left(\boldsymbol{\sigma}_{V,t}\right)$ according to (10) for training set
10 $s_{t+1} := Acc_{j,t}^{train}\left(\boldsymbol{\sigma}_{V,t}\right)$
11 Calculate $Acc_{j,t}^{test}\left(\boldsymbol{\sigma}_{V,t}\right)$ according to (10) for test set
12 **if** $Acc_{j,t}^{test}\left(\boldsymbol{\sigma}_{V,t}\right) > Acc_{\max}$ **then**
13 $\boldsymbol{\sigma}_{\max}^{(m)} := \boldsymbol{\sigma}_{V,t}$
14 $Acc_{\max} := Acc_{j,t}^{test}\left(\boldsymbol{\sigma}_{V,t}\right)$
15 **end**
16 $r_t := Acc_{j,t}^{train}\left(\boldsymbol{\sigma}_{V,t}\right) - Acc_{j,t-1}^{train}\left(\boldsymbol{\sigma}_{V,t-1}\right)$
17 Update:
 $Q_{j,t}(s_t,a_t) := Q_{j,t-1}(s_t,a_t) + \alpha\left(r_t + \gamma \max_a Q_{j,t}\left(s_{t+1},a\right) - Q_{j,t}(s_t,a_t)\right)$
18 **if** $Acc_{j,t}^{train}\left(\boldsymbol{\sigma}_{V,t}\right) = 1$ **then**
19 Go to step **23**
20 **end**
21 **end**
22 **end**
23 Start STAGE $m+1$

It is worth noting that the type of the PNN model influences the number of smoothing parameter updates. For PNNS, PNNC, PNNV and PNNVC, the number of the smoothing parameter updates is equal $t_{\max}$, $t_{\max} \times G$, $t_{\max} \times n$ and $t_{\max} \times n \times G$, respectively.

5 Experiments

In this section, we present the simulation results in the classification of medical databases obtained by PNNS, PNNC, PNNV and PNNVC trained by the proposed procedure. These results are compared with the outcomes obtained by PNN trained using the conjugate gradient procedure (PNNVC–CG), support vector machine (SVM) algorithm, gene expression programming (GEP) classifier, k–Means method, multilayer perceptron (MLP), radial basis function neural network (RBFN) and learning vector quantization neural network (LVQN). The data sets used in these experiments are also briefly described and the adjustments of the algorithm are provided. Moreover, the illustration of the PNNS training process is presented.

5.1 Data sets used in the study

In the simulations, six UCI machine learning repository medical data sets are used:

- Wisconsin breast cancer database [24] that consists of 683 instances with 9 attributes. The data is divided into two groups: 444 benign cases and 239 malignant cases.
- Pima Indians diabetes data set [36] that includes 768 cases having 8 features. Two classes of data are considered: samples tested negative (500 records) and samples tested positive (268 records).
- Haberman's survival data [21] that contains 306 patients who underwent surgery for breast cancer. For each instance, 3 variables are measured. The 5-year survival status establishes two input classes: patients who survived 5 years or longer (225 records) and patients who died within 5 years (81 records).
- Cardiotocography data set [3] that comprises 2126 measurements of fetal heart rate and uterine contraction features on 22 attribute cardiotocograms classified by expert obstetricians. The classes are coded into three states: normal (1655 cases), suspect (295 cases) and pathological (176 cases).
- Dermatology data [13] that includes 358 instances each of 34 features. Six data classes are considered: psoriasis (111 cases), lichen planus (71 cases), seborrheic dermatitis (60 cases), cronic dermatitis (48 cases), pityriasis rosea (48 cases) and pityriasis rubra pilaris (20 cases).
- Statlog heart database [3] that consists of 270 instances and 13 attributes. There are two classes to be predicted: absence (150 cases) or presence (120 cases) of heart disease.

5.2 Algorithms' settings

In the case of the proposed algorithm, the initial values of the action value function Q are set to zero. Three six–element action sets proposed in (12) are used. The maximum number of the training steps $t_{\max} = 100$ is assumed. We apply such a value of $t_{\max}$ in order to show that at a relatively small number of training steps it is possible to achieve satisfactory results. Additionally, the $Q(0)$-learning algorithm requires appropriate selection of its intrinsic parameters: the greedy parameter, the update rate and the discount factor.

The greedy parameter ε determines the probability of random action selection and must be taken from the set $[0, 1]$. If $\varepsilon = 0.05$, only 5 actions out of 100 are chosen randomly from the action set. The remaining 95 % of action selections are performed according to learned policy represented by the Q–table. If the elements of the Q–table are the same (initial iterations of Algorithm 1), the actions are selected randomly. In this work, the greedy parameter is chosen experimentally from the set $\{0.5, 0.05, 0.005\}$. Unfortunately, for $t_{\max} = 100$, the use of $\varepsilon = 0.5$ does not yield repeatable results. In turn, for $\varepsilon = 0.005$, it is observed that some actions have never been selected. Therefore, $\varepsilon = 0.05$ is utilized in the experiments.

The α parameter determines the update rate for the action value function Q. The small value of this factor increases the time of the training process. Its large value introduces the oscillations of Q elements [34]. The proper selection of α has a significant influence on the convergence of the training process. From the theoretical point of view, one requires that α is large enough to overcome any initial conditions or possible random fluctuations and it should decrease its value in time. However, in practical applications, the constant values of this factor are mostly used. Admittedly, this approach does not assure the convergence of the learning process, but a stable policy can be reached. In our study, we choose α experimentally from the set $\{0.1, 0.01, 0.001\}$. For all three parameter values, similar results are obtained. In the final simulation, we assume $\alpha = 0.01$.

The discount factor γ determines the relative importance of short and long termed prizes. This parameter is mostly picked arbitrarily near 1, e.g. 0.8 [6], 0.9 [2], [33] or 0.95 [4]. In this contribution, $\gamma = 0.95$.

PNNVC–CG used in the simulations is the probabilistic neural network trained by the conjugated gradient procedure. The model is a built-in tool of DTREG predictive

modeling software [35]. In the experiments, we use the network for which the smoothing parameter is adapted for each input feature and class separately. The starting values of the smoothing parameters for PNNVC–CG are between 0.0001 and 10 [35].

SVM algorithm [42] is used in this work as the data classifier. The model is trained by means of the SMO algorithm [30] available in Matlab's Bioinformatics Toolbox. Multiclass classification problems are solved by applying the one-against-all method. In all data classification cases, radial basis function kernel is applied with experimental grid search for both C constraint and sc spread constant: $C = \{10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}\}$ and $sc = \{0.08, 0.2, 0.3, 0.5, 0.8, 1.2, 1.5, 2, 5, 10, 50, 80, 100, 200, 500\}$, respectively.

The classification of considered data sets with the use of the GEP algorithm [9, 10] is performed in GeneXproTools software. For the simulation purposes, the GEP's parameters are chosen on the basis of Table 1. In all experiments, the number of chromosomes in the population is set to 30. For genetic computations, we use 10 random floating point constants per gene from the range $[-1000, 1000]$. Evolution is performed until 10000 generations are reached.

The k–Means clustering algorithm [14] is used in the comparison for classification purposes. The predictions are made for the unknown cases by assigning them the category of the closest cluster center. In the simulations, the optimal number of clusters is found which provides highest test accuracy.

Table 1 The head size, the number of genes within each chromosome, the linking functions between genes, the computing functions in the head, the fitness functions and the genetic operators used for the GEP classifier

Head size	3,4,5,6,7,8
Number of genes	$1, 2, \ldots, 12$
Linking function	Addition, Multiplication, Logical OR
Computing functions	$+, -, *, /, -x, 1/x$
	$b/(1+\exp(ax))$, $\exp\left(-(x-a)^2/(2b^2)\right)$
Fitness function	Sensitivity/Specificity
	Number of hits with penalty
	Mean squared error
Genetic operators	Mutation = 0.044
	Inversion = 0.1
	IS Transposition = 0.1
	RIS Transposition = 0.1
	Gene Transposition = 0.1
	One-Point Recombination = 0.3
	Two-Point Recombination = 0.3
	Gene Recombination = 0.1

MLP neural network is simulated with one or two hidden layers activated by the transfer functions from the set {*linear*, *hyperbolic tangent*, *logistic*}. The same set of transfer functions is applied for the output neurons, for which the sum squared error function is calculated. The number of hidden layer neurons is optimized in order to minimize the network error. The model is trained with gradient descent with momentum and adaptive learning rate backpropagation algorithms [8].

For RBFN and LVQN neural networks, the number of hidden neurons is selected empirically from the set $\{2, 4, 6, \ldots, 100\}$. The optimal number of hidden neurons is taken so that the sum squared error for each model is minimized. The spread constant in RBFN hidden layer activation function is chosen experimentally from the interval $[0.1, 10]$ with the step size 0.1.

5.3 Empirical results

In this study, the performance of PNN models, for which the smoothing parameter is determined using the $Q(0)$-learning based procedure, is evaluated on the input data partitioned in the following way. Firstly, the testing subsets are created by applying a random extraction of 10 %, 20 %, 30 % and 40 % of cases out of the input database. Then, the training sets are created using the rest of the patterns, i.e. 90 %, 80 %, 70 % and 60 % of data, respectively. This type of data division is introduced on purpose since considering all possible training–test subsets is complex from a computational point of view – the number of ways of dividing l training patterns into v sets, each of size k, is large and equals $l!/(v! \cdot (k!)^v)$ [17].

The remaining classifiers used in the comparative research: PNNVC–CG, SVM, GEP, k–Means, MLP, RBFN and LVQN are trained and validated on the same data subsets. The use of the same training/test sets for all the models makes the obtained results comparable.

Tables 2, 3, 4, 5, 6, 7 show the test accuracy values computed in terms of the percentage of correctly classified examples for PNNS, PNNC, PNNV and PNNVC with the smoothing parameter adapted by the proposed procedure for particular training–test set partitions on each of the six data bases. Additionally, for comparison purposes, the results are presented for PNNVC–CG, SVM, GEP, k–Means, MLP, RBFN and LVQN. For all models, the maximum (max), average (avr) and standard deviation (sd) values are provided. The results presented in the tables lead to the following observations:

1. In the classification of Wisconsin breast cancer data, out of all compared models, PNNC and PNNVC reach the highest average test accuracy which is equal 99.0 %. In the case of Haberman and dermatology data classification problems, the highest average test accuracy is

Table 2 The test accuracy values (in %) determined for four considered training-test subsets for Wisconsin breast cancer data set

	Data partitions [%]						
Model	60/40	70/30	80/20	90/10	*max*	*avr*	*sd*
PNNS	97.8	97.1	93.4	98.5	98.5	96.7	2.3
PNNC	99.6	98.5	97.8	100.0	100.0	99.0	1.0
PNNV	99.0	98.5	96.3	100.0	100.0	98.4	1.5
PNNVC	99.6	98.5	97.8	100.0	100.0	99.0	1.0
PNNVC–CG	95.3	96.6	90.5	89.7	96.6	93.1	3.4
SVM	98.9	97.6	95.6	97.1	98.9	97.3	1.4
GEP	99.3	97.6	96.3	98.5	98.5	96.8	1.5
k–Means	96.7	97.1	94.9	98.5	98.5	96.8	1.5
MLP	97.7	96.4	93.9	96.8	97.7	96.2	1.7
RBFN	97.4	96.1	95.6	95.6	97.4	96.2	0.9
LVQN	97.8	97.6	92.3	96.9	97.8	96.2	2.6

Table 3 The test accuracy values (in %) determined for four considered training-test subsets for Pima Indians diabetes data set

	Data partitions [%]						
Model	60/40	70/30	80/20	90/10	*max*	*avr*	*sd*
PNNS	69.7	66.1	69.5	75.3	75.3	70.1	3.8
PNNC	69.1	71.3	72.1	75.3	75.3	71.9	2.6
PNNV	73.9	76.5	76.0	80.5	80.5	76.7	2.8
PNNVC	74.6	76.5	75.3	79.2	79.2	76.4	2.0
PNNVC–CG	65.5	30.5	68.2	66.3	68.2	57.6	18.1
SVM	78.2	78.7	75.3	76.6	78.7	77.2	1.5
GEP	72.6	77.4	76.0	80.5	80.5	76.6	3.3
k–Means	68.3	70.0	70.8	71.4	71.4	70.2	1.2
MLP	73.2	73.6	74.8	73.2	74.8	73.7	0.8
RBFN	65.8	67.8	65.6	68.8	68.8	67.0	1.6
LVQN	65.8	65.9	67.4	66.1	67.4	66.3	0.7

Table 4 The test accuracy values (in %) determined for four considered training-test subsets for Haberman survival data set

	Data partitions [%]						
Model	60/40	70/30	80/20	90/10	*max*	*avr*	*sd*
PNNS	77.0	75.0	73.0	74.2	77.0	75.8	1.5
PNNC	79.5	77.2	78.7	77.4	79.5	78.2	1.1
PNNV	76.2	78.3	82.0	87.1	87.1	80.9	4.8
PNNVC	77.9	79.3	80.3	87.1	87.1	81.2	4.1
PNNVC–CG	50.0	69.6	68.8	51.6	69.6	60.0	10.6
SVM	78.7	76.1	78.7	80.6	80.6	78.5	1.9
GEP	75.4	75.0	73.8	77.4	77.4	75.4	1.5
k–Means	69.7	69.6	70.5	67.7	70.5	69.4	1.2
MLP	74.2	74.2	74.9	76.4	76.4	74.9	1.0
RBFN	74.6	73.9	75.4	74.2	75.4	74.5	0.7
LVQN	76.4	75.8	78.4	74.2	78.4	76.2	1.7

Table 5 The test accuracy values (in %) determined for four considered training-test subsets for cardiotocography data set

	Data partitions [%]						
Model	60/40	70/30	80/20	90/10	*max*	*avr*	*sd*
PNNS	88.9	84.4	85.9	85.0	88.9	86.0	2.0
PNNC	90.2	88.4	87.3	93.0	93.0	89.7	2.5
PNNV	97.3	93.9	93.2	97.2	97.3	95.4	2.2
PNNVC	95.8	91.9	93.2	94.9	95.8	93.9	1.7
PNNVC–CG	11.9	72.0	74.5	65.4	84.5	58.5	32.0
SVM	97.2	94.4	97.2	98.1	98.1	97.2	0.7
GEP	92.4	92.3	94.1	96.7	96.7	93.9	2.1
k–Means	89.4	89.7	85.6	93.9	93.9	89.7	3.4
MLP	89.1	89.0	89.5	91.7	91.7	89.8	1.3
RBFN	77.9	77.8	77.9	77.6	77.9	77.8	0.1
LVQN	78.1	77.8	77.9	77.6	78.1	77.9	0.2

Table 6 The test accuracy values (in %) determined for four considered training-test subsets for dermatology data set

	Data partitions [%]						
Model	60/40	70/30	80/20	90/10	*max*	*avr*	*sd*
PNNS	85.9	89.6	87.5	91.7	91.7	88.7	2.5
PNNC	89.4	90.6	90.3	94.4	94.4	91.2	2.2
PNNV	96.5	97.2	95.8	100.0	100.0	97.6	1.8
PNNVC	90.1	96.2	91.7	97.2	97.2	93.8	3.4
PNNVC–CG	55.6	89.6	86.2	94.4	94.4	81.5	17.5
SVM	98.6	96.2	97.2	94.4	98.6	96.6	1.8
GEP	97.2	98.1	94.4	94.4	98.1	96.0	1.9
k–Means	87.3	89.6	88.9	88.9	89.6	88.7	1.0
MLP	73.9	70.7	75.6	76.7	76.7	74.2	2.6
RBFN	78.9	79.3	76.4	80.6	80.6	78.8	1.8
LVQN	66.4	31.1	73.2	69.4	73.2	60.0	19.5

Table 7 The test accuracy values (in %) determined for four considered training-test subsets for Statlog heart data set

	Data partitions [%]						
Model	60/40	70/30	80/20	90/10	*max*	*avr*	*sd*
PNNS	75.9	87.6	77.8	92.6	92.6	83.5	8.0
PNNC	79.6	90.1	79.6	96.3	96.3	86.4	8.2
PNNV	83.3	91.4	83.3	100.0	100.0	89.5	8.0
PNNVC	83.3	91.4	87.0	100.0	100.0	90.4	7.2
PNNVC–CG	63.0	59.3	51.9	74.1	74.1	62.1	9.3
SVM	80.6	90.1	81.5	88.9	90.1	85.3	4.9
GEP	76.9	86.4	81.5	88.9	83.4	94.6	2.7
k–Means	63.9	76.9	66.7	81.5	81.5	70.0	7.8
MLP	75.8	86.4	76.5	87.8	87.8	81.6	6.4
RBFN	81.5	88.9	81.5	92.6	92.6	86.1	5.6
LVQN	76.9	86.9	75.2	82.6	86.9	80.4	5.4

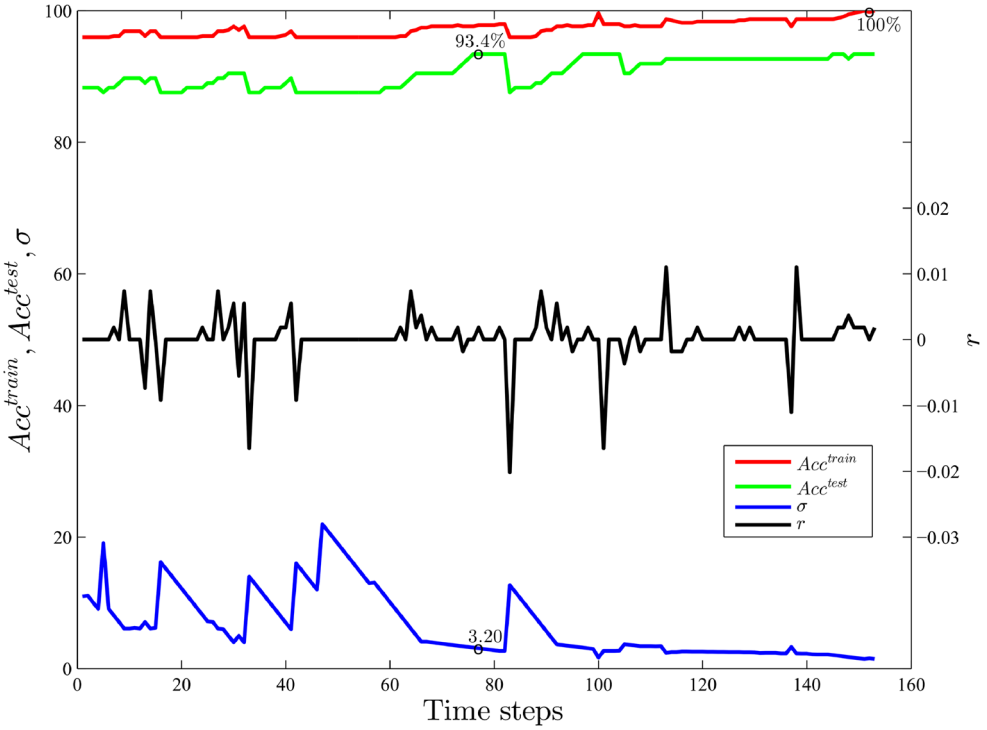

Fig. 4 The changes of Acc^{train} [%], Acc^{test} [%], σ and r within the training process of PNNS in the classification task of Wisconsin breast cancer data set (partition: 80/20)

obtained for PNNVC (81.2 %) and PNNV (97.6 %), respectively.

2. The SVM model provides the highest average test accuracy in the classification of the Pima Indians diabetes data set (77.2 %) and cardiotocography database (97.2 %). In these two classification tasks, PNNV is the second best model with the test accuracy lower by 0.5 % and 1.8 %, respectively. For the Statlog classification problem, GEP algorithm yields the highest average test accuracy which equals 94.6 %. This result

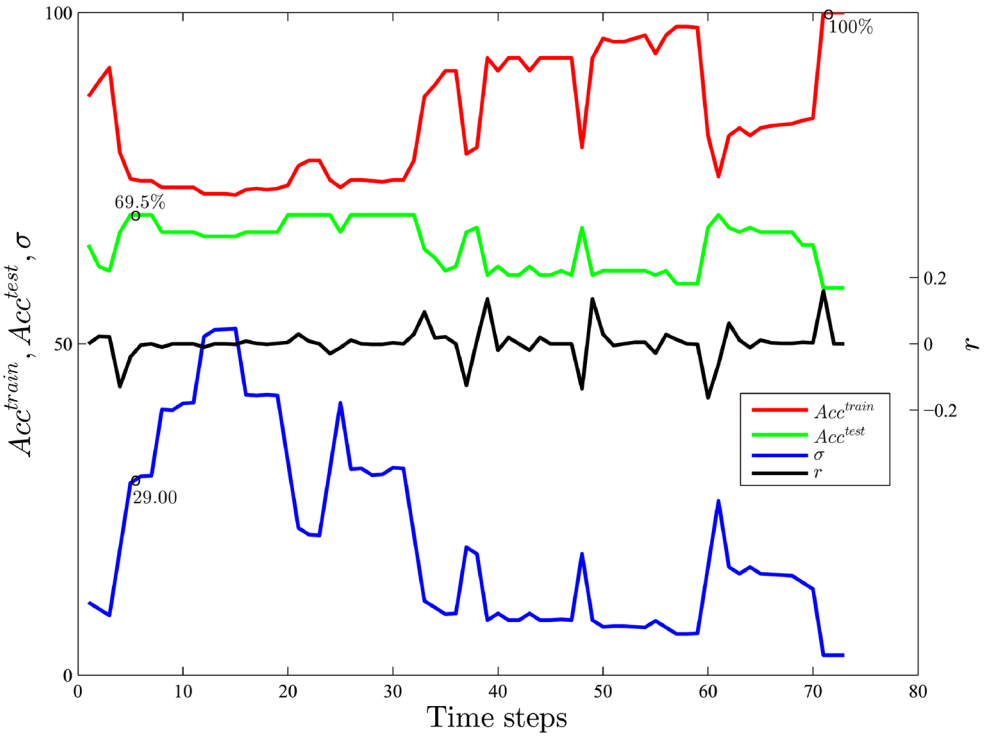

Fig. 5 The changes of Acc^{train} [%], Acc^{test} [%], σ and r within the training process of PNNS in the classification task of Pima Indians diabetes data set (partition: 80/20)

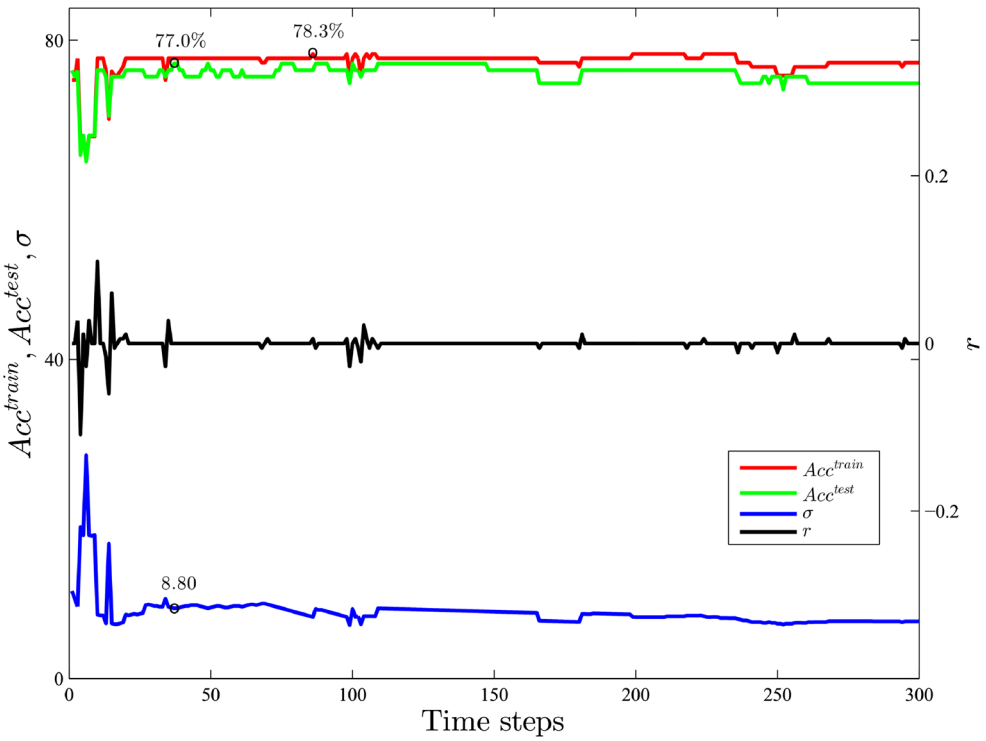

Fig. 6 The changes of Acc^{train} [%], Acc^{test} [%], σ and r within the training process of PNNS in the classification task of Haberman survival data set (partition: 60/40)

is followed by the outcomes of PNNVC, PNNV and PNNC.

3. Except for the dermatology classification problem, PNNVC–CG turns out to be the worst classifier. The k–Means algorithm and the remaining reference neural networks (MLP, RBFN and LVQN) achieve lower test accuracy than the PNNV, PNNVC, SVM and GEP classifiers.

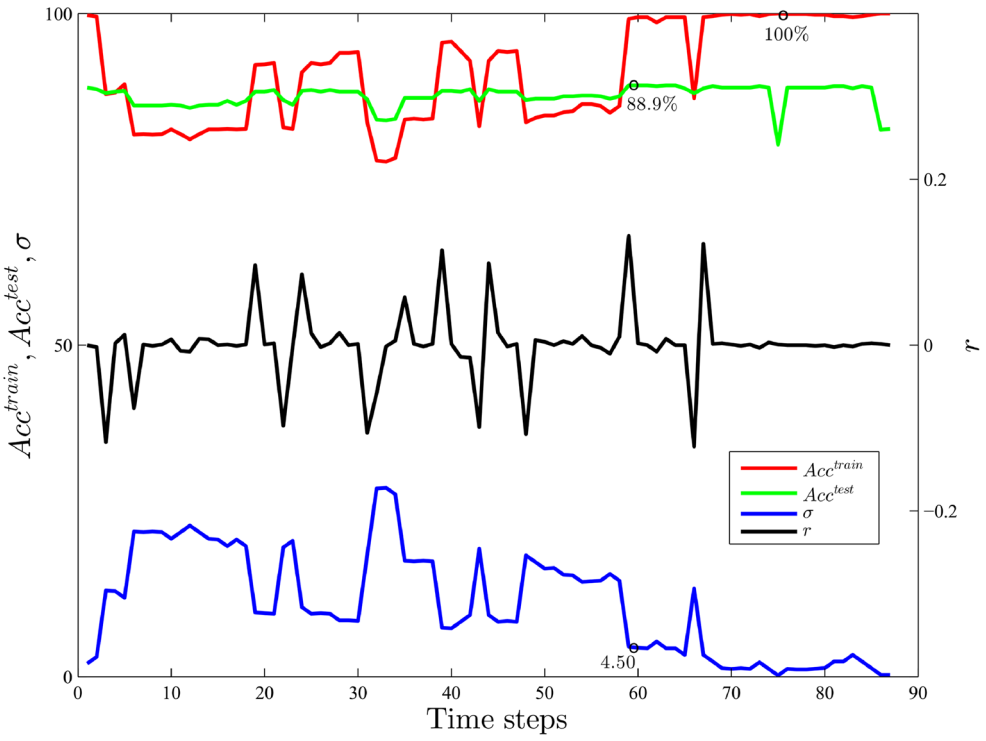

Fig. 7 The changes of Acc^{train} [%], Acc^{test} [%], σ and r within the training process of PNNS in the classification task of cardiotocography data set (partition: 60/40)

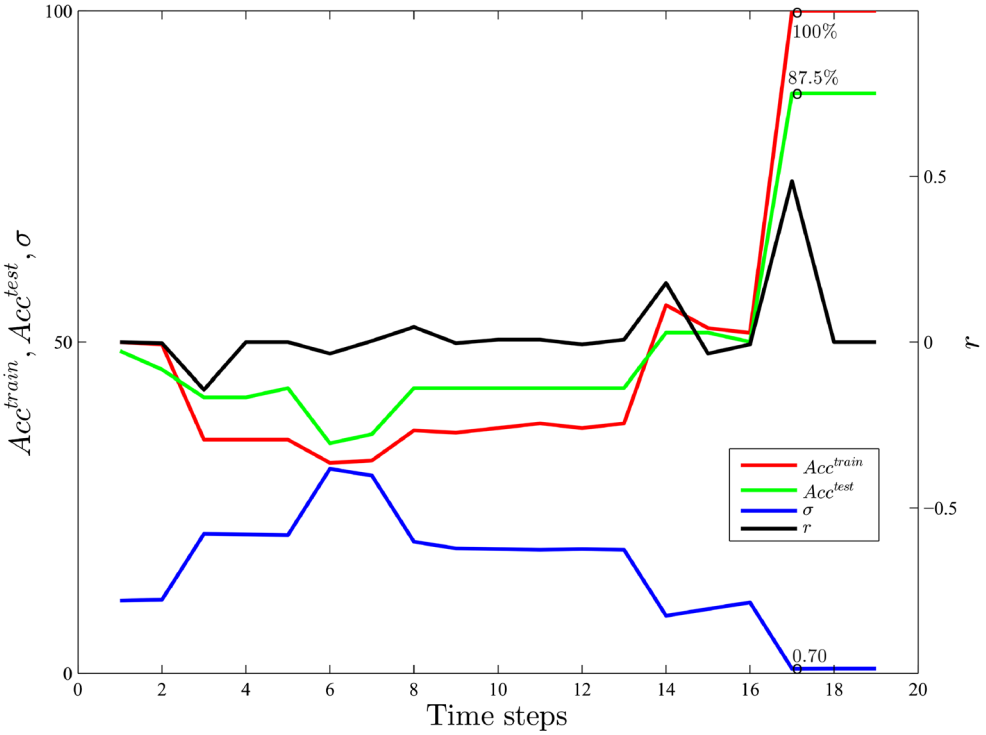

Fig. 8 The changes of Acc^{train} [%], Acc^{test} [%], σ and r within the training process of PNNS in the classification task of dermatology data set (partition: 80/20)

5.4 Illustration of the PNN training process

Figures 4, 5, 6, 7, 8, 9 illustrate the changes of Acc^{train}, Acc^{test}, σ and r as a function of time steps for six data set classification problems. These changes are only shown for one exemplary data set partition. The plots are depicted for PNNS since for this model the smoothing parameter takes the form of a scalar. In each figure, we mark the maximum

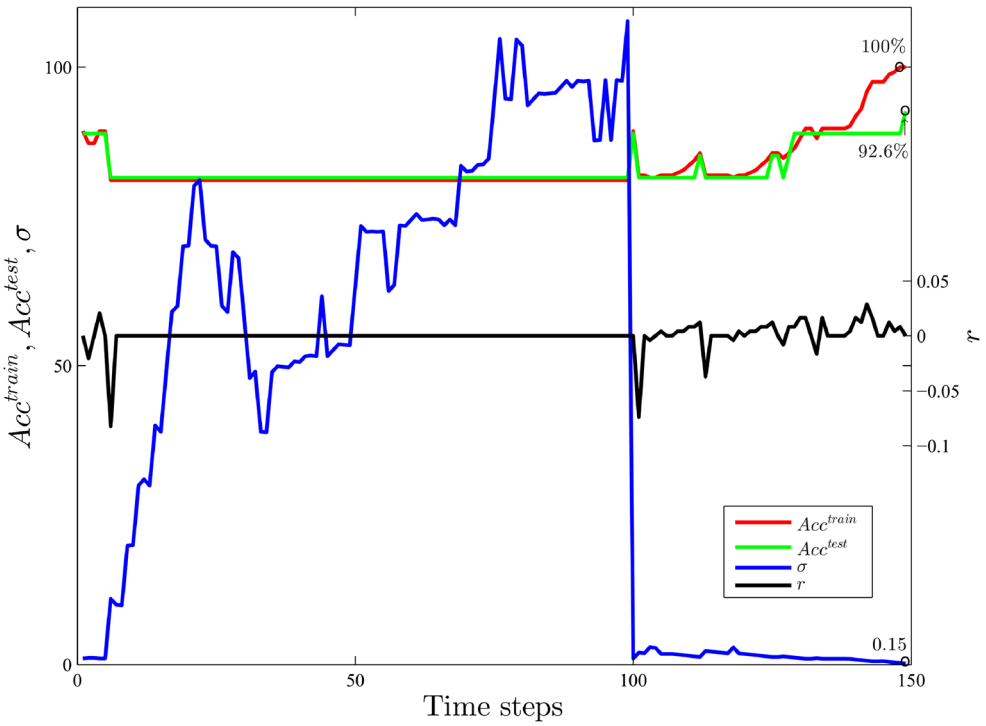

Fig. 9 The changes of Acc^{train} [%], Acc^{test} [%], σ and r within the training process of PNNS in the classification task of Statlog heart data set (partition: 90/10)

values of Acc^{train}, Acc^{test} and its corresponding smoothing parameter.

We can observe that the changes of the smoothing parameter values within the training process result from the implementation of the proposed procedure. The magnitude of these changes becomes smaller in subsequent stages of the procedure (e.g. Figs. 4, 6 and 9). Large modifications of the smoothing parameter provide the possibility of either finding optimal σ after a small number of steps ($t = 6$ in Fig. 5), or narrowing the range of its possible optimal values (Fig. 9). Another interesting feature worth noting is that the reinforcement signal follows the changes of the training accuracy. r becomes negative when Acc^{train} decreases and r takes the positive value when Acc^{train} increases.

On the basis of the figures, the following observation can also be noticed. (i) In the dermatology and Statlog heart data classification tasks, the maximal value of Acc^{test} is obtained for $Acc^{train} = 100$ %. In the remaining classification problems, the maximal value of Acc^{train} does not guarantee the highest value of the test accuracy; (ii) Only for the Haberman survival data set classification problem it is impossible to achieve 100 % of the training accuracy; (iii) The classification problem of the Haberman survival and Statlog heart data sets confirm that it is necessary to perform all stages of the procedure. In the first case, 100 % of the training accuracy is not reached in any stage. In the second one, the maximum value of $Acc^{test} = 92.6$ % is obtained in the third stage of the procedure.

6 Conclusions

In this article, the procedure based on the $Q(0)$-learning algorithm was proposed to the adaptive choice and computation of the smoothing parameters of the probabilistic neural network. All possible classes of the PNN models were regarded. These models differed in the way of the smoothing parameters representation. Application of the procedure based on the $Q(0)$-learning algorithm for PNN parameter tuning is the element of novelty. It is worth to note that the comparison of all types of probabilistic neural networks has not been presented in literature.

The proposed approach was tested on six data sets and compared with PNN trained by the conjugate gradient procedure, SVM algorithm, GEP classifier, k–Means method, multilayer perceptron, radial basis function neural network and learning vector quantization neural network. In three classification problems, at least one of the PNNC, PNNV or PNNVC models trained by the proposed procedure provided the highest average accuracy. Four out of six times, PNNS was the second to last data classifier. This means that the representation of the smoothing parameter, either in terms of a vector or a matrix, contributes to higher PNN's prediction ability. As one can observe, for PNN trained by the conjugate gradient procedure the lowest accuracy was obtained for all six data classification cases. Thus, the proposition of any alternative method for probabilistic neural network training is by all means justified.

Acknowledgments This work was supported in part by Rzeszow University of Technology Grant No. U–235/DS and U–8613/DS.

References

1. Adeli H, Panakkat A (2009) A probabilistic neural network for earthquake magnitude prediction. Neural Netw 22:1018–1024
2. Asadpour M, Siegwart R (2004) Compact Q-learning optimized for micro-robots with processing and memory constraints. Robot Auton Syst 48(1):49–61
3. Bache K, Lichman M (2013) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml. University of California School of Information and Computer Science. Irvine
4. Barto AG, Sutton RS, Anderson CW (1983) Neuronlike adaptive elements that can solve difficult learning problem. IEEE Trans SMC 13:834–847
5. Bertin M, Schweighofer N, Doya K (2007) Multiple model-based reinforcement learning explains dopamine neuronal activity. Neural Netw 20:668–67
6. Braga APS, Arauno AFR (2003) A topological reinforcement learning agent for navigation. Neural Comput Applic 12:220–236
7. Chtioui Y, Panigrahi S, Marsh R (1998) Conjugate gradient and approximate Newton methods for an optimal probabilistic neural network for food color classification. Opt Eng 37:3015–3023
8. Demuth H, Beale M (1994) Neural network toolbox user's guide. The Mathworks Inc.
9. Ferreira C (2001) Gene expression programming: a new adaptive algorithm for solving problems. Compl Syst 13(2):87–129
10. Ferreira C (2006) Gene expression programming: mathematical modeling by an artificial intelligence. Springer, Berlin
11. Georgiou LV, Pavlidis NG, Parsopoulos KE et al (2006) New self-adaptive probabilistic neural networks in bioinformatic and medical tasks. Int J Artificial Intell Tools 15:371–396
12. Georgiou LV, Alevizos PD, Vrahatis MN (2008) Novel approaches to probabilistic neural networks through bagging and evolutionary estimating of prior probabilities. Neural Process Lett 27:153–162
13. Guvenir HA, Demiroz G, Ilter N (1998) Learning differential diagnosis of Eryhemato-Squamous diseases using voting feature intervals. Artif Intell Med 13:147–165
14. Hartigan JA, Wong MA (1979) A k-means clustering algorithm. J R Stat Soc Ser C 1:100–108
15. Heinen MR, Engel PM (2010) An incremental probabilistic neural network for regression and reinforcement learning tasks. In: Diamantaras K, Duch W, Iliadis LS (eds) Lecture notes in computer science, vol 6353. Springer, Berlin, Heidelberg, pp 170–179
16. Iglesias A, Martinez P, Aler R, Fernandez F (2009) Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. Appl Intell 31:89–106
17. Jonathan P, Krzanowski WJ, McCarthy WV (2000) On the use of cross-validation to assess performance in multivariate prediction. Stat Comput 10:209–229

18. Kim CO, Kwon I-H, Baek J-G (2008) Asynchronous action-reward learning for nonstationary serial supply chain inventory control. Appl Intell 28:1–16
19. Kusy M, Zajdel R (2014) Stateless Q-learning algorithm for training of radial basis function based neural networks in medical data classification. In: Korbicz J, Kowal M (eds) Advances in intelligent systems and computing, vol 230. Springer, Berlin / Heidelberg, pp 267–278
20. Kyriacou E, Pattichis MS, Pattichis CS et al (2009) Classification of atherosclerotic carotid plaques using morphological analysis on ultrasound images. Appl Intell 30:3–23
21. Landwehr JM, Pregibon D, Shoemaker AC (1984) Graphical methods for assessing logistic regression models. J Am Stat Assoc 79:61–71
22. Li J, Li Z, Chen J (2011) Microassembly path planning using reinforcement learning for improving positioning accuracy of a 1 cm3 omni-directional mobile microrobot. Appl Intell 34:211–225
23. Maglogiannis I, Zafiropoulos E, Anagnostopoulos I (2009) An intelligent system for automated breast cancer diagnosis and prognosis using SVM based classifiers. Appl Intell 30:24–36
24. Mangasarian OL, Street WN, Wolberg WH (1995) Breast cancer diagnosis and prognosis via linear programming. Oper Res 43(4):570–577
25. Mantzaris D, Anastassopoulos G, Adamopoulos A (2011) Genetic algorithm pruning of probabilistic neural networks in medical disease estimation. Neural Netw 24:831–835
26. Mendonca M, Arruda LVR, Neves F Jr (2012) Autonomous navigation system using Event Driven-Fuzzy Cognitive Maps. Appl Intell 37:175–188
27. Nebti S, Boukerram A (2013) Handwritten characters recognition based on nature-inspired computing and neuro-evolution. Appl Intell 38:146–159
28. Orr RK (1997) Use of a probabilistic neural network to estimate the risk of mortality after cardiac surgery. Med Decis Making 17:178–185
29. Parzen E (1962) On estimation of a probability density function and mode. Ann Math Stat 36:1065–1076
30. Platt JC (1999) Sequential minimal optimization: a fast algorithm for training support vector machines. In: Schlkopf B, Burges JC, Smola J (eds) Advances in kernel methods - support vector learning. MIT Press, Cambridge, pp 185–208
31. Rutkowski L (2004) Adaptive probabilistic neural networks for pattern classification in time-varying environment. IEEE Trans Neural Netw 15:811–827
32. Samanta B, Al-Balushi KR, Al-Araimi SA (2006) Artificial neural networks and genetic algorithm for bearing fault detection. Soft Comput 10:264–271
33. Schoknecht R, Riedmiller M (2003) Reinforcement learning on explicitly specified time scales. Neural Comput & Applic 12(2):61–80
34. Schweighofer N, Doya K (2003) Meta-learning in reinforcement learning. Neural Netw. 16:5–9
35. Sherrod PH (2013) DTREG predictive modelling software. http://www.dtreg.com. Accessed 26 September 2013
36. Smith JW, Everhart JE, Dickson WC, Knowler WC, Johannes RS (1988) Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the symposium on computer applications and medical care. IEEE Computer Society Press, pp 261–265
37. Specht DF (1990) Probabilistic neural networks and the polynomial adaline as complementary techniques for classification. IEEE Trans Neural Netw 1:11–121
38. Specht DF (1990) Probabilistic neural networks. Neural Netw 3(1):109–118
39. Specht DF (1994) Experience with adaptive probabilistic neural networks and adaptive general regression neural networks. In: IEEE international conference on neural networks. USA, Orlando, pp 1203–1208
40. Starzyk JA, Liu Y, Batog S (2010) A novel optimization algorithm based on reinforcement learning. In: Tenne Y, Goh C-K (eds) Computational intelligence in optimization, ALO, vol 7, pp 27–47
41. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
42. Vapnik V (1995) The nature of statistical learning theory. Springer-Verlag, New York
43. Vien NA, Ertel W, Chung TC (2013) Learning via human feedback in continuous state and action spaces. Appl Intell 39:267–278
44. Watkins C (1989) Learning from delayed rewards. PhD Dissertation. University of Cambridge, England
45. Wen XB, Zhang H, Xu XQ, Quan JJ (2009) A new watermarking approach based on probabilistic neural network in wavelet domain. Soft Computing 13:355–360
46. Wu QH, Liao HL (2010) High-dimensional function optimisation by reinforcement learning. In: IEEE congress on evolutionary computation (CEC), Barcelona, pp 1–8
47. Zhong M, Coggeshall D, Ghaneie E et al (2007) Gap-based estimation: choosing the smoothing parameters for probabilistic and general regression neural networks. Neural Comput 19(10): 2840–2864

Maciej Kusy is an assistant professor at The Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, Poland. He received his M.Sc. degree in electrical engineering from Rzeszow University of Technology in 2000 and Ph.D. degree in biocybernetics and biomedical engineering from Warsaw University of Technology in 2008. His research interests comprise computational intelligence algorithms, decision support systems, medical applications.

Roman Zajdel received the M.Sc. degree in electrical engineering from Rzeszow University of Technology in 1990 and Ph.D in computer science from Wroclaw University of Technology in 1999. He is an assistant professor at the Institute of Control and Computer Engineering, Rzeszow University of Technology. His research interests concentrate on reinforcement learning, fuzzy logic and neural networks.