Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2007) 1424–1438
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Generalized locally recurrent probabilistic neural networks with application to text-independent speaker verification

Todor D. Ganchev[a,*], Dimitris K. Tasoulis[b], Michael N. Vrahatis[b], Nikos D. Fakotakis[a]

[a]*Wire Communications Laboratory, Department of Electrical and Computer Engineering, University of Patras, GR-26500 Rion-Patras, Greece*
[b]*Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR-26500 Rion-Patras, Greece*

Received 18 November 2005; received in revised form 17 May 2006; accepted 21 May 2006
Communicated by R. Tadeusiewicz
Available online 10 October 2006

Abstract

An extension of the well-known probabilistic neural network (PNN) to generalized locally recurrent PNN (GLR PNN) is introduced. The GLR PNN is derived from the original PNN by incorporating a fully connected recurrent layer between the pattern and output layers. This extension renders GLR PNN sensitive to the context in which events occur, and therefore, capable of identifying temporal and spatial correlations. In the present work, this capability is exploited to improve the speaker verification performance. A fast three-step method for training GLR PNNs is proposed. The first two steps are identical to the training of original PNNs, while the third step is based on the differential evolution (DE) optimization method.

Keywords: Recurrent neural networks; Hybrid architectures; Speaker verification

1. Introduction

Following the introduction of the probabilistic neural network (PNN) by Specht [27], numerous enhancements, extensions, and generalizations of the original model have been proposed. These efforts aim at improving either the learning capability [33,3], or the classification accuracy [6] of PNNs; or alternatively, at optimizing network size, thereby reducing memory requirements and the resulting complexity of the model, as well as achieving lower operational times [29,30]. An architecture, referred to as a modified PNN (MPNN), was introduced in [35] for equalization of a non-linear frequency modulated communication channel. The MPNN, which is closely related to the PNN, represents a vector-quantized form of the general regression neural networks (GRNN) [28] of Specht. Improved versions of the MPNN were employed in numerous signal processing, pattern recognition [34], and financial prediction applications [15]. A temporal updating technique for tracking changes in a sequence of images, based on periodic supervised and unsupervised updates of the PNN, has been also developed [32]. A locally recurrent global-feedforward PNN-based classifier, combining the desirable features of both feedforward and recurrent neural networks was introduced in [10]. Specifically, by incorporating a new hidden layer, comprised of recurrent neurons, we extended the original PNN architecture to locally recurrent PNN (LR PNN) [10], in order to capture the inter-frame correlations present in a speech signal.

Initially, the locally recurrent global-feedforward architecture was proposed by Back and Tsoi [1], who considered an extension of the multi-layer perceptron neural network (MLP NN) to exploit contextual information. In their work, they introduced the infinite impulse response (IIR) and finite impulse response (FIR) synapses, able to utilize time dependencies in the input data. The FIR synapse has connections to his own, current and delayed, inputs, while the IIR synapse has also connections to its past outputs. Ku and Lee [16] proposed diagonal recurrent neural networks (DRNN) for the task of system identification in real-time control applications. Their approach is based on

*Corresponding author. Wire Communications Laboratory, Department of Electrical and Computer Engineering, University of Patras, GR-26500 Rion-Patras, Greece. Tel.: +30 69 38552140; fax: +30 26 10997336.
E-mail address: tganchev@wcl.ee.upatras.gr (T.D. Ganchev).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.05.012

the assumption that a single feedback from the neuron's own output is sufficient. Thus, they simplify the fully connected neural network to render training easier. A comprehensive study of several MLP-based locally recurrent neural networks is available in Campolucci et al. [5]. The authors of [5] introduced a unifying framework for the gradient calculation techniques, called causal recursive back-propagation. All aforementioned approaches consider gradient based training techniques for neural networks, which, as it is well-known, require differentiable transfer functions.

The work presented here draws on the concept of the locally recurrent global-feedforward architecture, and the recurrent layer we propose is similar to the IIR synapse introduced in [1] and the DRNN defined by Ku and Lee [16]. Our approach differs from the previously mentioned, primarily, because we consider PNNs instead of MLP NN. Most importantly, however, in the architecture proposed here each neuron in the recurrent layer receives as input not only current and past values of its inputs, but also the N previous outputs of all neurons in that layer. This can be considered as an generalization of the locally recurrent global-feedforward PNN architecture [10] that was obtained by adding time-lagged values of inputs to recurrent layer linkage of the LR PNN. Thus, in the GLR PNN, the neurons of the recurrent layer are linked to all current and L past inputs, and to N past outputs of all neurons from the same layer—in contrast to the LR PNN where connections to the past inputs were not implemented.

In comparison to [11], the present contribution updates the GLR PNN architecture, its training procedure, and provides comprehensive results. Specifically, now all feedbacks, which origin from past outputs of neurons belonging to the recurrent layer, embrace the activation functions of these neurons. The last facilitates the training procedure of the recurrent layer weights and contributes for an improved convergence of the training process due to reduced dynamic range of values that the weight coefficients take. More importantly however, the aforementioned modification of the GLR PNN architecture leads to transformed dynamic range of the weight coefficients of the recurrent layer feedbacks. The dynamic range of their values is now comparable to the one of the weight coefficients of the inputs for that layer. In turn, the last leads to a lower sensitivity to rounding errors, an more economical hardware implementation, and most importantly, to an improved overall robustness of GLR PNNs.

Besides this improvement, in the present work the GLR PNN architecture evolves further allowing independence of the number of past inputs L and past outputs N considered in the hidden recurrent layer linkage. Thus, the earlier works [10,11] can be considered as two special cases of the generalized architecture: for $L=0$ and $L=N$, respectively. The aforementioned generalization of the GLR PNN architecture adds a new degree of freedom into the hands of researchers, and therefore contribute to improved flexibility and applicability to a wider range of classification tasks.

Another important development that we bring forward in the present work, when compared to [11], is the amended error function, which is object of minimization during the recurrent layer training. The error function was modified in a manner to seek for specific predefined balance of training among the classes, which guarantees a better steering of the learning rate for each and every class, and a better customization of the individual classes.

The layout of the present article is described as follows: The theoretical foundations of the original PNN are briefly discussed in Section 2. In Section 3, we present the updated architecture of the GLR PNN, and in Section 4, a modification of the training method is proposed. A brief description of the speaker verification (SV) task and the specific form of the GLR PNN when involved in this task is presented in Section 5. In addition, Section 5 outlines our SV system, referred to WCL-1, and discusses some measures for assessment of the SV performance. Section 6 describes the PolyCost speaker recognition database. Next, in Section 7 the experimental setup is discussed and comparative results on the task of text-independent SV are presented. Specifically, firstly the efficiency of various differential evolution (DE) operators for training the GLR PNNs recurrent layer is studied. Then, the ability of the GLR PNN to exploit correlations in the input data, for several values of the recurrence depth N, is investigated. A comparative evaluation of the GLR PNNs performance with that of the LR PNNs one, and with other locally recurrent structures, like the DRNNs, IIRs and FIR's ones, is performed. Finally, results from a comparative evaluation of the GLR PNN with respect to the original PNN, as well as to a Gaussian mixture models (GMMs)—based classifier, are reported. The article ends with concluding remarks.

2. Theoretical foundations of the PNN

In Fig. 1, the general structure of a PNN for classification in K classes, is illustrated. As presented in the figure, the first layer of the PNN, designated as an *input layer*, accepts the input vectors to be classified. The nodes of the second layer, which is designated as a *pattern layer*, are grouped in K groups depending on the class k_i they belong. These nodes, also referred to as pattern units or kernels, are connected to all inputs of the first layer. Although numerous probability density function estimators are possible [27,23,4], here we consider that each and every pattern unit can be defined as having an activation function the Gaussian basis function

$$f_{ij}(\mathbf{x}; c_{ij}, \sigma) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\frac{1}{2\sigma^2}(\mathbf{x}-c_{ij})^{\mathrm{T}}(\mathbf{x}-c_{ij})\right), \tag{1}$$

where $i=1,\ldots,K$, $j=1,\ldots,M_i$, and M_i is the number of pattern units in a given class k_i. Here, σ is the standard

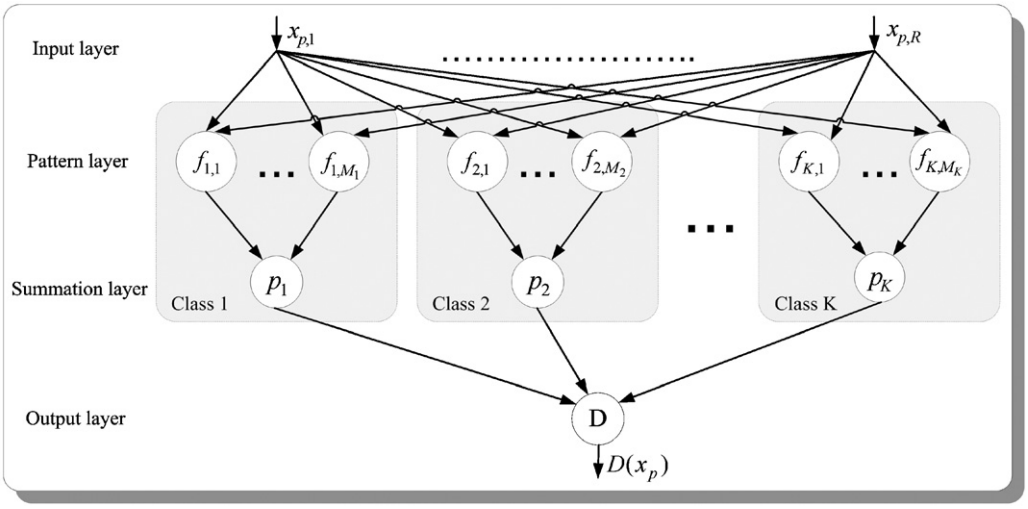

Fig. 1. Structure of the original probabilistic neural network.

deviation, also known as *spread* or *smoothing factor*. It regulates the receptive field of the kernel. The input vector $\mathbf{x}$ and the centers $c_{ij} \in R^d$ of the kernel are of dimensionality d. Finally, exp stands for the exponential function, and the superscript T denotes the transpose of the vector. Obviously, the total number of the second-layer nodes is given as a sum of the pattern units for all classes:

$$M = \sum_{i=1}^{K} M_i. \tag{2}$$

Next, the weighted outputs of the pattern units from the second layer that belong to the group k_i are connected to the summation unit of the third layer (designated as *summation layer*) corresponding to that specific class k_i. The weights are determined by the decision cost function and the a priori class distribution. In the general case, the positive weight coefficients w_{ij} for weighting the member functions of class k_i have to satisfy the requirement

$$\sum_{j=1}^{M_i} w_{ij} = 1 \quad \text{for every given class } k_i, \;\; i = 1, \ldots, K. \tag{3}$$

For a symmetrical cost function of the type "zero-one" (no loss for correct decision and a unit loss to any error), which implies minimum classification error rate, and a uniform a priori distribution, all weights for class k_i are equal to $1/M_i$. Consequently, each node of the summation layer estimates the class-conditional probability density function $p_i(\mathbf{x}|k_i)$ of each class k_i, defined as

$$p_i(\mathbf{x}|k_i) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{M_i} \sum_{j=1}^{M_i} \exp\left(-\frac{1}{2\sigma^2} (\mathbf{x} - \mathbf{x}_{ij})^{\mathrm{T}} (\mathbf{x} - \mathbf{x}_{ij}) \right), \tag{4}$$

where $\mathbf{x}_{ij}$ is the jth training vector from class k_i, $\mathbf{x}$ is the test input vector, d is the dimension of the speech feature vectors, and M_i is the number of training patterns in class k_i. Each training vector $\mathbf{x}_{ij}$ is assumed a center of a kernel function. The standard deviation σ acts as a smoothing factor, which softens the surface (4), defined by the multiple Gaussian functions (1). As obvious from (4), σ has the same value for all the pattern units, and therefore, a homoscedastic PNN is considered.

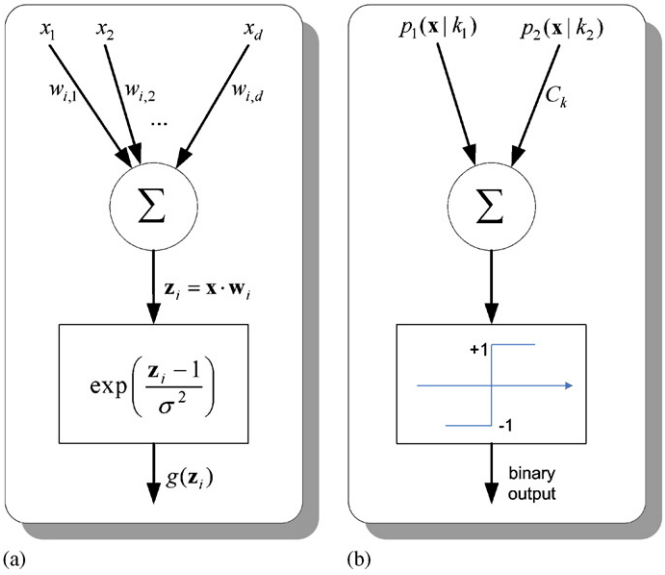

Fig. 2. Internal structure of: (a) ith neuron in the pattern layer; (b) a neuron in the output layer.

The internal structure of the pattern units, (i.e. the neurons in the pattern layer) is illustrated in Fig. 2(a). As the figure presents, the ith pattern unit forms a dot product $\mathbf{z}_i$ of the input pattern vector $\mathbf{x}$ with a weight vector $\mathbf{w}_i$

$$\mathbf{z}_i = \mathbf{x}\mathbf{w}_i. \tag{5}$$

Next, a non-linear operation is performed on $\mathbf{z}_i$ (the activation function box in Fig. 2(a)) before outputting it to

the following summation layer. In fact, the non-linear operation

$$g(\mathbf{z}_i) = \exp\left(\frac{\mathbf{z}_i - 1}{\sigma^2}\right) \tag{6}$$

used here supposes that both the vectors $\mathbf{x}$ and $\mathbf{w}_i$ are normalized to unit length, and therefore, this is equivalent to using

$$\exp\left(-\frac{1}{2\sigma^2}(\mathbf{w}_i - \mathbf{x})^{\mathrm{T}}(\mathbf{w}_i - \mathbf{x})\right) \tag{7}$$

which is in the same form as Eq. (1). Although, as originally proposed by Specht [27] alternative estimators (details available in: [23,4]) can be used instead of the one in (4), here we only consider estimators of the type specified by Eq. (4).

Comparing (1) and (4), it can be seen that the output of the summation unit for class k_i is the estimated class-conditional PDF for that class, when $c_{ij} = \mathbf{x}_{ij}$. That makes the training of the pattern layer straightforward, since the training procedure is reduced just to remembering of the training set and adjusting σ. Having the estimation (4) of the class-conditional PDF $p_i(\mathbf{x}|k_i)$ for all classes computed from the training data, and the a priori class probability $P(k_i)$, which in many applications is known (or otherwise usually assumed uniform for all classes), the best classifier which can minimize the defined cost function is given by the fundamental Bayesian decision rule [20]. Thus, in the *output layer* of the PNN, also known as *competitive layer*, the Bayesian decision rule

$$D(\mathbf{x}) = \arg\max_i \{P(k_i)p_i(\mathbf{x}|k_i)\}, \quad i = 1, \ldots, K \tag{8}$$

is applied to distinguish the class k_i, to which the input vector $\mathbf{x}$ belongs.

Fig. 2(b) presents the internal structure of a neuron from the output layer. As the figure illustrates, in a two-class problem—such as the SV problem—the output units are two-input neurons, which produce binary outputs. They have a single variable weight, C_k,

$$C_k = -\frac{P_2(k_2)c_{2,k}}{P_1(k_1)c_{1,k}}\frac{M_1}{M_2}, \tag{9}$$

where $P_1(k_1)$ and $P_2(k_2)$ are a priori class probabilities, M_1 and M_2 are the number of training patterns for class one and two, respectively, and $c_{1,k}$ and $c_{2,k}$ are the losses in case of misclassification of input vectors belonging to the respective class. In fact, the constant C_k represents the ratio of a priori probabilities, divided by the ratio of training vectors and multiplied by the ratio of losses. When the numbers M_1 and M_2 of the training vectors from the respective categories are obtained in proportion to their a priori probabilities, (9) can be rewritten [27], as

$$C_k = -\frac{c_{2,k}}{c_{1,k}}. \tag{10}$$

Hence, the constant C_k depends only from the significance of the decision. If there is no particular reason for biasing the decision, C_k may simplify to -1. Furthermore, since the class-conditional PDFs for all the K classes were already computed in (4), beside the decision $D(\mathbf{x})$, the PNN is able to provide the confidence in its decision that follows directly from the Bayes' theorem [2]. Thus, the posterior probability for each class, provided that the classes are mutually exclusive and exhaustive, is computed through

$$P(k_i|\mathbf{x}) = \frac{P(k_i)p_i(\mathbf{x}|k_i)}{\sum_{j=1}^{K} P(k_j)p_j(\mathbf{x}|k_j)}, \quad i = 1, \ldots, K. \tag{11}$$

At this point, we need to emphasize that according to the *Bayes' Postulate* the a priori class probability $P(k_i)$ is assumed equal for all classes, unless there is a compelling reason to assume otherwise.

3. The generalized locally recurrent PNN architecture

Although there exist numerous improved versions of the original PNN, which are either more economical, or exhibit a significantly superior performance, for simplicity of exposition, we adopt the original PNN as a starting point for introducing the GLR PNN architecture. The development of the PNN architecture that we propose in the present section does not interfere with the aforementioned enhancements, and therefore, it can be applied straightforwardly to the more advanced PNNs.

The GLR PNN is derived from the PNN by including a hidden recurrent layer between the radial basis (4) and competitive (8) layers. This hidden recurrent layer consists of fully linked neurons (one per each target class), which in addition to current inputs and L delayed inputs possess also feedbacks from N past outputs of all neurons belonging to that layer, including its own ones.

The simplified structure of the GLR PNN ($L = N = 1$) is presented in Fig. 3. As illustrated, the first hidden layer of the GLR PNN is identical to the one of the PNN. Specifically, in the first hidden layer, the GLR PNN, as their predecessor—the PNN, implement the Parzen window estimator by using a mixture of Gaussian basis functions. When a GLR PNN for classification in K classes is considered, the probability density function of each class k_i, which for simplicity we denote here as $\mathbf{f}_i(\mathbf{x}_p)$, is defined by

$$\mathbf{f}_i(\mathbf{x}_p) = \frac{1}{(2\pi)^{d/2}\sigma^d}\frac{1}{M_i}\sum_{j=1}^{M_i}\exp\left(-\frac{1}{2\sigma_i^2}(\mathbf{x}_p - \mathbf{x}_{ij})^{\mathrm{T}}(\mathbf{x}_p - \mathbf{x}_{ij})\right) \tag{12}$$

for $i = 1, \ldots, K$, where $\mathbf{x}_{ij}$ denotes the jth training vector from class k_i; $\mathbf{x}_p$ is the pth input vector; d is the dimension of the speech feature vectors; and M_i is the number of training patterns in class k_i. Each training vector $\mathbf{x}_{ij}$ is assumed to be a center of a kernel function, and consequently the number of pattern units in the first hidden layer of the neural network is given by the sum (2) of the pattern units for all classes. As in the original PNN, the spread σ_i, smooths the surface defined by the multiple

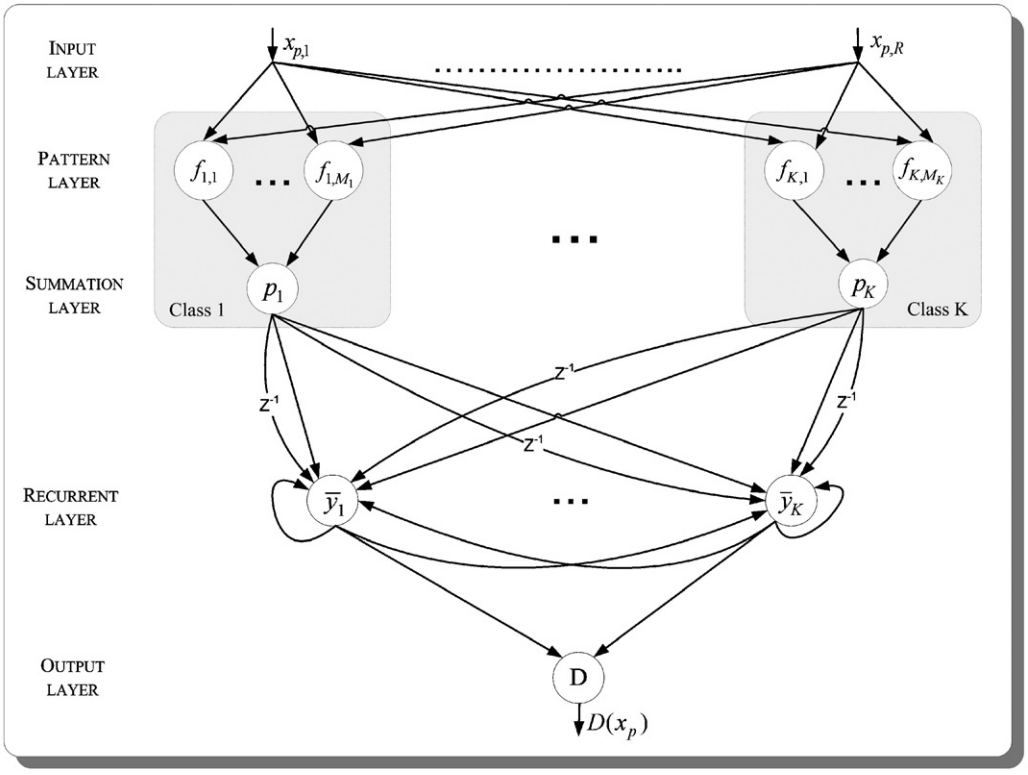

Fig. 3. Simplified structure of the generalized locally recurrent PNN.

Gaussian functions. Instead of the simple covariance matrix, $\sigma_i^2 I$, where I represents the identity matrix, the full covariance matrix can be computed using the Expectation Maximization algorithm, as proposed in [33,17]. For simplicity of exposition, we consider here a simple case, where the value of σ_i is identical for all pattern units from a specific class, or, it can even be the same for all pattern units irrespective of the class, as it was originally proposed by Specht [27].

The class-conditional PDFs (12), computed for all K classes, act as inputs for the next hidden layer, namely the fully connected recurrent layer. In fact, along with the current K values obtained through (12), each recurrent neuron also receives $L \times K$ past values of the PDFs, as well as $N \times K$ past outputs of all recurrent neurons from the same layer. Fig. 4 illustrates the linkage of a single neuron belonging to this hidden generalized recurrent layer. As illustrated, beside the current values of the PDFs from all classes, $\mathbf{f}_i(\mathbf{x}_p)$, $i = 1, \ldots, K$, this neuron also receives the past inputs $\mathbf{f}_i(\mathbf{x}_{p-t})$, $t = 1, \ldots, L$ and feedbacks from its past outputs $\overline{y}_i(\mathbf{x}_{p-t})$, $t = 1, \ldots, N$, as well as from current $y'_{j \neq i}(\mathbf{x}_p)$, $j = 1, \ldots, K$ and past $\overline{y}_{j \neq i}(\mathbf{x}_{p-t})$, $j = 1, \ldots, K$, $t = 1, \ldots, N$ outputs from all other neurons belonging to that layer. Here, the subscript i denotes the current neuron number, and the subscript p stands for the serial number of the input vector $\mathbf{x}_p$. On its own side, the current neuron i provides to the other neurons of the recurrent layer its current $y_i(\mathbf{x}_p)$ and past $\overline{y}_i(\mathbf{x}_{p-t})$, $t = 1, \ldots, N$ outputs, again with p standing for the specific input vector.

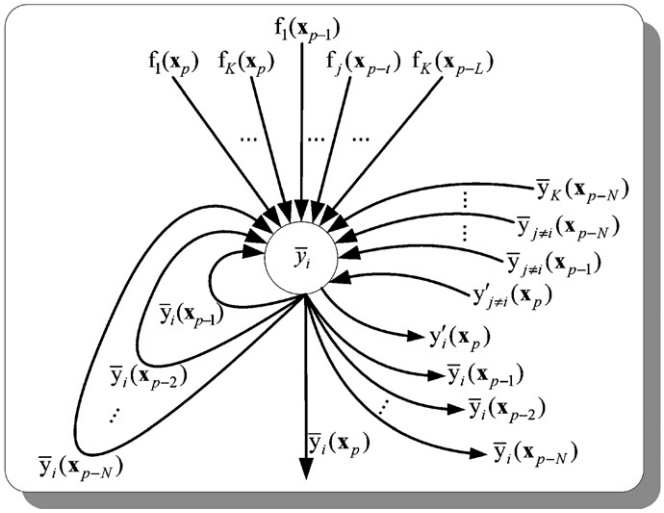

Fig. 4. Linkage of a recurrent neuron in the GLR PNN.

In Fig. 5, the detailed structure of the recurrent neurons is illustrated. The inputs $\mathbf{f}_i(\mathbf{x}_p)$ and $\mathbf{f}_i(\mathbf{x}_{p-t})$, with $i = 1, \ldots, K$, and $t = 1, \ldots, L$ denoting the current and past class conditional PDFs, respectively, are weighted by the coefficients $b_{i,j,t}$. All feedbacks $\overline{y}_i(\mathbf{x}_{p-t})$, $t = 1, \ldots, N$ that originate from the present neuron i, and the links $\overline{y}_{j \neq i}(\mathbf{x}_{p-t})$, $j = 1, \ldots, K$, $t = 1, \ldots, N$ coming from the other neurons $j \neq i$ of the recurrent layer are weighted by the coefficients $a_{i,j,t}$, $t = 1, \ldots, N$ and $a_{i,j \neq i,t}$, $j = 1, \ldots, K$, $t = 1, \ldots, N$, respectively. The three indexes of the

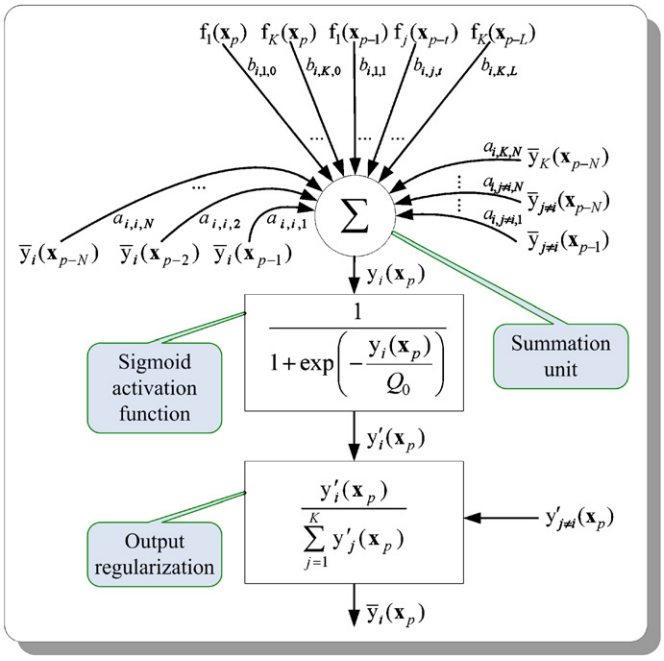

Fig. 5. Structure of a neuron from the recurrent layer of the GLR PNN.

weights $a_{i,j,t}$ and $b_{i,j,t}$ stand for: (1) $i = 1, \ldots, K$ indicates current recurrent neuron i and equals to the class index k_i; (2) $j = 1, \ldots, K$ indicates for the class to which the corresponding input or output belongs; and (3) $t = 1, \ldots, L$ shows the time delay of the specific input.

The output $y_i(\mathbf{x}_p)$ of each summation unit located in the locally recurrent layer of the GLR PNN is computed by

$$y_i(\mathbf{x}_p) = \sum_{t=0}^{L} \sum_{k=1}^{K} b_{i,k,t} \mathbf{f}_k(\mathbf{x}_{p-t}) + \sum_{t=1}^{N} \sum_{k=1}^{K} a_{i,k,t} \overline{y}_k(\mathbf{x}_{p-t}), \quad i = 1, \ldots, K. \tag{13}$$

Here, $\mathbf{x}_p$ denotes the input vector; $\mathbf{f}_k(\mathbf{x}_{p-t})$ is the probability density function of class k_i that has been computed t time steps ago; L is the number of delayed input values for a given class; K is the number of classes; N stands for the recurrence depth; $\overline{y}_k(\mathbf{x}_{p-t})$ is the normalized past output of class k_i that has been delayed on t time steps; and finally $a_{i,k,t}$ and $b_{i,k,t}$ represent weight coefficients.

Next, the output $y_i(\mathbf{x}_p)$ of each summation unit located in the recurrent layer of the GLR PNN is subject to the regularization transformation

$$\overline{y}_i(\mathbf{x}_p) = \frac{\mathrm{sgm}(y_i(\mathbf{x}_p))}{\sum_{j=1}^{K} \mathrm{sgm}(y_j(\mathbf{x}_p))}, \quad i = 1, \ldots, K, \tag{14}$$

imposed to retain an interpretation of the output of the recurrent layer in terms of probabilities. The designation sgm refers to the sigmoid activation function.

Consequently, the Bayesian decision rule (8), rewritten as

$$D(\mathbf{x}_p) = \arg \max_i \{h_i c_i \overline{y}_i(\mathbf{x}_p)\}, \quad i = 1, \ldots, K \tag{15}$$

is applied to distinguish the class k_i, to which the input vector $\mathbf{x}_p$ belongs. Here, h_i is a priori probability of occurrence of the patterns of category k_i, and c_i is the loss in case of misclassification of a vector belonging to class k_i. Finally, provided that all classes are mutually exclusive and exhaustive, we can compute the Bayesian confidence for every decision $D(\mathbf{x}_p)$ by applying the Bayes' theorem

$$P(k_i|\mathbf{x}_p) = \frac{h_i \overline{y}_i(\mathbf{x}_p)}{\sum_{j=1}^{K} h_j \overline{y}_j(\mathbf{x}_p)}, \quad i = 1, \ldots, K. \tag{16}$$

The posterior probability $P(k_i|\mathbf{x}_p)$ for the pth input vector belonging to class k_i is computed by relying on the a priori probabilities h_i and the temporally smoothed probabilities $\overline{y}_i(\mathbf{x}_p)$.

When a test trial consists of multiple feature vectors, as this happens in the SV task, the posterior probability $P(k_i|\mathbf{X})$, all vectors of a given test trial $\mathbf{X} = \{\mathbf{x}_p\}$, $p = 1, \ldots, P$ to belong to class k_i, is computed by

$$P(k_i|\mathbf{X}) = \frac{N(D(\mathbf{x}_p) = k_i)}{\sum_{j=1}^{K} N(D(\mathbf{x}_p) = k_j)}, \quad i = 1, \ldots, K, \tag{17}$$

where $N(D(\mathbf{x}_p) = k_i)$ is the number of vectors $\mathbf{x}_p$ classified by the Bayesian decision rule (15) as belonging to class k_i. Since the SV task assumes an exhaustive taxonomy, any of the inputs $\mathbf{x}_p$ falls in one of the classes k_i. Thus, the equality

$$P = \sum_{j=1}^{K} N(D(\mathbf{x}_p) = k_j), \tag{18}$$

where P is the number of test vectors in the given trial $\mathbf{X}$, is always preserved.

Considering that in many real-world applications, including the SV problem, computing the probability $P(k_i|\mathbf{X})$ is not sufficient as a final outcome from the GLR PNN, since an explicit final (in our case binary) decision about a given trial $\mathbf{X}$ is required, the outcome of (17) is assessed with respect to a predefined threshold Θ

$$P(k_i|\mathbf{X}) \lessgtr \Theta. \tag{19}$$

Most often, the threshold Θ is computed on a data set, referred to as development data, which is independent from the training and testing data. A necessary requirement for obtaining a reasonable estimate of Θ is the development data to be representative, i.e. they have to bear a resemblance to the real-world data that the GLR PNN will operate within the corresponding application.

4. The GLR PNN training

A three-step training procedure for the GLR PNN is proposed. By analogy to the original PNN, the first training step creates the actual topology of the network. Specifically, in the first hidden layer, a pattern unit for each training vector is created by setting its weight vector equal to the corresponding training vector. The outputs of the pattern units associated with the class k_i are then connected to the corresponding summation units of the second hidden

layer neurons. The number of recurrent neurons in this second hidden layer is equal to the number of classes K.

The second training step is the computation of the smoothing parameters $\{\sigma_i\}$ for each class. (In the general case of heteroscedastic PNN [33,17] the covariance matrix $\sum_{k_i}$ has to be computed). To this end, various approaches [6,29,30,19,21,18,12], etc. have been proposed. Although other methods can be employed, here we mention only the one proposed by Cain [6] due to its simplicity. According to this method, any σ_i is proportional to the mean value of the minimum distances among the training vectors in class k_i:

$$\sigma_i = \lambda \frac{1}{M_i} \sum_{j=1}^{M_i} \min\{\|\mathbf{x}_{i,j} - \mathbf{x}_{i,j\neq i}\|_2^2\}, \tag{20}$$

where $\mathbf{x}_{i,j}$ is the jth pattern unit (located in the pattern layer) for class k_i; $\|\cdot\|_2$ corresponds to the 2-norm on R^d (reminding that $\mathbf{x}_{i,j}$ are the remembered training data, and therefore, $\mathbf{x}_{i,j} \in R^d$); d is the dimensionality of the input data; the expression $\min\{\|\mathbf{x}_{i,j} - \mathbf{x}_{i,j\neq i}\|_2^2\}$ represents the smallest Euclidean distance computed between jth pattern unit of class k_i and all other pattern units from the same class; and M_i is the number of training patterns in class k_i. The constant λ, which controls the degree of overlapping among the individual Gaussian functions, is usually selected in the range $\lambda \in [1.1, 1.4]$. If the smoothing parameter σ is common for all classes, either it is chosen empirically, or it is computed by applying (20) on the entire training data set, regardless of the class belonging of the pattern units.

The third training step is the computation of the weights of the recurrent layer. This step utilizes the probabilities obtained by passing the training data exploited at step one through the pattern layer. The optimization of the recurrent layer weights is equivalent to the minimization of the composite error function

$$E(\mathbf{w}) = E_c(\mathbf{w}) + G_{\text{imp}} E_d(\mathbf{w}), \tag{21}$$

where the errors $E_c(\mathbf{w})$ and $E_d(\mathbf{w})$ are defined as follows:

$$E_c(\mathbf{w}) = \sum_{i=1}^{K} c_i P(Miss|k_i) P(k_i), \tag{22}$$

$$E_d(\mathbf{w}) = \frac{1}{K(K-1)} \sum_{i=1}^{K} \sum_{\substack{j=1 \\ j\neq i}}^{K} |c_i P(Miss|k_i) P(k_i) - c_j P(Miss|k_j) P(k_j)|. \tag{23}$$

While the $E_c(\mathbf{w})$ renders an account for the level of training for the entire recurrent layer, $E_d(\mathbf{w})$ provides for balance among the training of the individual classes. The gain factor G_{imp} allows adjusting the contribution of the $E_d(\mathbf{w})$ in the total error $E(\mathbf{w})$; K is the number of classes; c_i is the relative cost of detection error for the corresponding class k_i; $P(Miss|k_i)$ is the post probability of misclassification of the patterns belonging to class k_i; and the $P(k_i)$ is the a priori probability of occurrence of the patterns of class k_i in the training data set. The values of $P(Miss|k_i)$ are obtained in the following way: For a given weight vector $\mathbf{w} = \{a_{i,k,t}, b_{i,k,t}\}$, the values of $\bar{y}_i$, $i = 1, \ldots, K$ are computed, according to (13) and (14), and then (15) is applied. Finally, the post-probability $P(Miss|k_i)$ is computed as

$$P(Miss|k_i) = 1 - P(k_i|\mathbf{X}), \tag{24}$$

where $P(k_i|\mathbf{X})$ is obtained from (17) for the case of the training data set.

The minimization of the total error $E(\mathbf{w}) = E(a_{i,k,t}, b_{i,k,t})$ is achieved by employing the DE algorithm introduced by Storn and Price [31]. In brief, the DE method exploits a population of potential solutions to probe the search space. At each iteration, called generation g, three steps, called *mutation*, *recombination*, and *selection* are performed. According to the DE method, initially all weight vectors are randomly initialized. Then at the mutation step, new mutant weight vectors $\mathbf{v}_{g+1}^i$ are generated by combining weight vectors, randomly chosen from the population. For that purpose, one of the variation operators (25)–(30) is exploited

$$\mathbf{v}_{g+1}^i = \omega_g^{r1} + \mu(\omega_g^{r1} - \omega_g^{r2}), \tag{25}$$

$$\mathbf{v}_{g+1}^i = \omega_g^{\text{best}} + \mu(\omega_g^{r1} - \omega_g^{r2}), \tag{26}$$

$$\mathbf{v}_{g+1}^i = \omega_g^{r1} + \mu(\omega_g^{r2} - \omega_g^{r3}), \tag{27}$$

$$\mathbf{v}_{g+1}^i = \omega_g^{i} + \mu(\omega_g^{\text{best}} - \omega_g^{i}) + \mu(\omega_g^{r1} - \omega_g^{r2}), \tag{28}$$

$$\mathbf{v}_{g+1}^i = \omega_g^{\text{best}} + \mu(\omega_g^{r1} - \omega_g^{r2}) + \mu(\omega_g^{r3} - \omega_g^{r4}), \tag{29}$$

$$\mathbf{v}_{g+1}^i = \omega_g^{r5} + \mu(\omega_g^{r1} - \omega_g^{r2}) + \mu(\omega_g^{r3} - \omega_g^{r4}), \tag{30}$$

where ω_g^{r1}, ω_g^{r2}, ω_g^{r3}, ω_g^{r4} and ω_g^{r5} are randomly selected vectors, different from ω_g^i, ω_g^{best} is the best member of the current generation, and the positive mutation constant μ controls the magnification of the difference between two weight vectors. At the recombination step, each component $j = 1, \ldots, R$ of these new weight vectors is subject to a further modification. A random number $r \in [0, 1]$ is generated, and if r is smaller than a predefined crossover constant p, the jth component of the mutant vector $\mathbf{v}_{g+1}^i$ becomes jth component of the trial vector. Otherwise the jth component is obtained from the target vector. Finally, at the selection step, the trial weight vectors obtained at the crossover step are accepted for the next generation only if they yield a reduction of the value of the error function; otherwise the previous weights are retained. The training process ends when the target error margin is reached, or after completing a predefined number of iterations.

After the weight coefficients of the recurrent layer are computed, the GLR PNN is fully trained and ready for operation.

5. The SV task

The SV process, based on an identity claim and a sample of speaker's voice, provides an answer to the unambiguous question: "Is the present speaker the one s/he claims to be, or not?" The output of the verification process is a binary decision "Yes, s/he is !" or "No, s/he is not !". The actual

decision depends on the degree of similarity between the speech sample and a predefined model for the enrolled user, whose identity the speaker claims. When an enrolled user claims his own identity, we designate the input utterance as a *target* trial. When a non-user addresses a SV system, or when an enrolled user claims identity belonging to another user, we denote that utterance as a *non-target* trial. The non-target trials are also referred to as *impostor* trials.

Thus, in the SV problem we have two hypotheses—either the input speech originates from the same person, whose identity the speaker claims, or it originates from another person, which has different identity. In order to test each of these two hypotheses, we build an individual expert (e.g. a GLR PNN) for each enrolled user. Each expert incorporates two models: one build from the voice of the enrolled user, and another one representing the rest of the world. The latter one is also designated as a *reference*. Since the reference model has to be sufficiently *flat* not to interfere with the models of the individual users, it is build by exploiting large amounts of speech from multiple speakers.

With respect to linguistic contents of speech the speaker recognition process can be text-dependent or text-independent. The text-dependent SV systems examine the manner in which a specific password or a system-prompted sequence is pronounced. In the text-independent scenario, the talker is not restricted in any way, and as soon as the identity claim is provided, s/he is free to speak naturally, without any vocabulary restrictions. Although the GLR PNN is suitable for the text-dependent scenario, in the present work, we consider the text-independent one.

5.1. GLR PNNs in the context of the SV task

As it was explained above, in the SV task only two classes ($K = 2$) are considered—one for the particular enrolled user whose identity is claimed, and one for the reference, which represents the collective model of the non-users. Therefore, for each enrolled users a GLR PNN involved in two-class separation problem is considered. For easiness of illustration, here we discuss the simplest GLR PNN—with recurrence depth one.

In Fig. 6, the architecture of the GLR PNN for the case of two classes ($K = 2$), one past value of the inputs ($L = 1$), and recurrence depth one ($N = 1$) is shown. For visualization purposes the locally recurrent layer is magnified. As illustrated, there exist three distinct sections in Fig. 6 corresponding to the three major elements in the GLR PNN architecture. Starting from the input, all blocks that precede the recurrent layer, defined by (13) and (14), represent the computation of the probability density function (12). Next, the block *compet* that follows the recurrent layer stands for the Bayesian decision rule (15).

The probability density functions $\mathbf{f}_1$ and $\mathbf{f}_2$ and their past values computed by the first hidden layer act as inputs for the summation units of the recurrent layer's neurons. Both these inputs, as well as the delayed past outputs $\overline{y}_1$ and $\overline{y}_2$ of the two classes are weighted by the weights $b_{i,k,t}$ and $a_{i,k,t}$, respectively. Finally, the current output values $\overline{y}_1$ and $\overline{y}_2$ are passed as inputs to the competitive layer (15) that decides the winning class.

Since, in the speaker recognition tasks each test trial consists of P speech feature vectors, (17) is applied to accumulate the decisions made in (15). The final decision

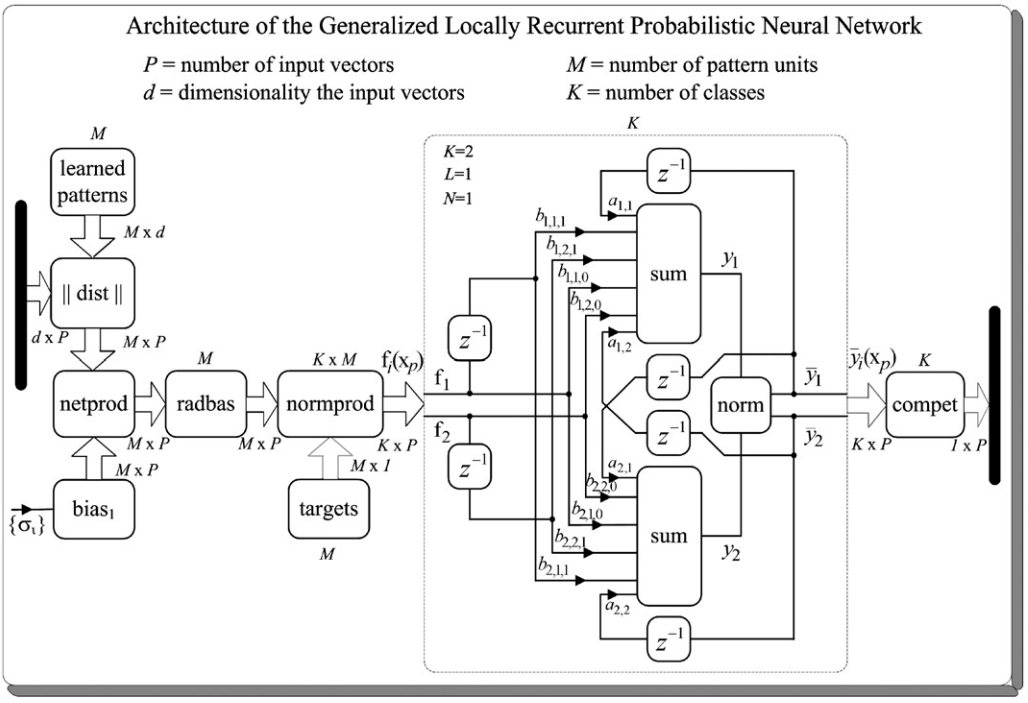

Fig. 6. Architecture of the generalized locally recurrent probabilistic neural network. The locally recurrent layer is delineated by dashed line.

(19) is made with respect to a speaker-independent threshold.

5.2. WCL-1: A text-independent SV system

Our text-independent SV system, referred to as WCL-1, briefly described in the following was used as a platform to evaluate the performance of the GLR PNN in comparison to various other classifiers. The WCL-1 system [7], a participant in the 2002 NIST speaker recognition evaluation [22] has a modular structure with an individual PNN for each enrolled user. A reference model is employed for counterbalancing the scores produced by the individual user models.

The WCL-1 system utilizes a feature vector composed of 31 Mel-frequency cepstral coefficients (MFCC). A comprehensive description of the MFCC computation steps is offered in [8], where various implementation strategies are evaluated on the SV task. In the baseline system discussed here, the MFCC FB-32 implementation (details in [8]), which is profoundly based on the Auditory Toolbox [26] of M. Slaney, is employed.

Fig. 7 summarizes the speech parameterization steps. In the present work, we deal with telephone quality speech, sampled at 8 kHz. Saturation by level is a common phenomenon for telephone speech signals. In order to reduce the spectral distortions it causes, a band-pass filtering of speech is performed as a first step of the feature extraction process. A fifth-order Butterworth filter with pass-band from 80 to 3800 Hz is used for both training and testing. Then the speech signal is pre-emphasized by the filter

$$H(z) = 1 - 0.97z^{-1} \tag{31}$$

and subsequently, windowed into frames of 40 ms duration, at a frame rate of 100 Hz using a Hamming window. The voiced/unvoiced speech separation is performed by a modification of the autocorrelation method with clipping [24]. Only the voiced speech frames are used due to their relatively better robustness to interference. Next, each voiced speech frame is subjected to 1024-point short-time discrete Fourier transform, and then is passed through a set of Q equal-area triangular band-pass filter-bank channels

$$H_i(k) = \begin{cases} 0 & \text{for } k < f_{b_{i-1}}, \\ \frac{2(k - f_{b_{i-1}})}{(f_{b_i} - f_{b_{i-1}})(f_{b_{i+1}} - f_{b_{i-1}})} & \text{for } f_{b_{i-1}} \leqslant k \leqslant f_{b_i}, \\ \frac{2(f_{b_{i+1}} - k)}{(f_{b_{i+1}} - f_{b_i})(f_{b_{i+1}} - f_{b_{i-1}})} & \text{for } f_{b_i} \leqslant k \leqslant f_{b_{i+1}}, \\ 0 & \text{for } k > f_{b_{i+1}}, \end{cases}$$

$$\text{with } i = 1, \ldots, Q, \tag{32}$$

where i stands for the ith filter; f_{b_i} are the boundary points that specify the filters; and k corresponds to the kth coefficient of the 1024-point DFT. The boundary points f_{b_i} are expressed in terms of position in order to conform with k.

We have accepted an approximation of the Mel-scale, with 13 linearly spaced filters, lowest central frequency 200 Hz, highest 1000 Hz and 19 log-spaced with highest central frequency 3690 Hz. Finally, a J dimensional feature vector C_j is formed, after applying discrete Cosine transform (33) to the log-filter-bank outputs X_i

$$C_j = \sum_{i=1}^{Q} X_i \cos\left(j(i - 1/2)\frac{\pi}{Q}\right), \quad j = 1, \ldots, J, \tag{33}$$

$$X_i = \log_{10}\left(\sum_{k=0}^{F-1} |X(k)| H_i(k)\right), \quad i = 1, \ldots, Q. \tag{34}$$

Here $X(k)$ are the DFT coefficients; $Q = 32$ is the number of filters in the filter-bank $H_i(k)$; $J = 32$ is the total number of cepstral coefficients; and $F = 1024$ is the size of the DFT. In the WCL-1 system, we compute all 32 MFCCs, but the zeroth one is not included into the feature vector. As it is well-known the zeroth cepstral coefficient is related to log-energy of the speech frame, and therefore it is much influenced by the transmission channel and handset type.

Subsequently, the feature vectors computed from the training data set, for both the user's and reference models, are compressed by using the k-means clustering technique

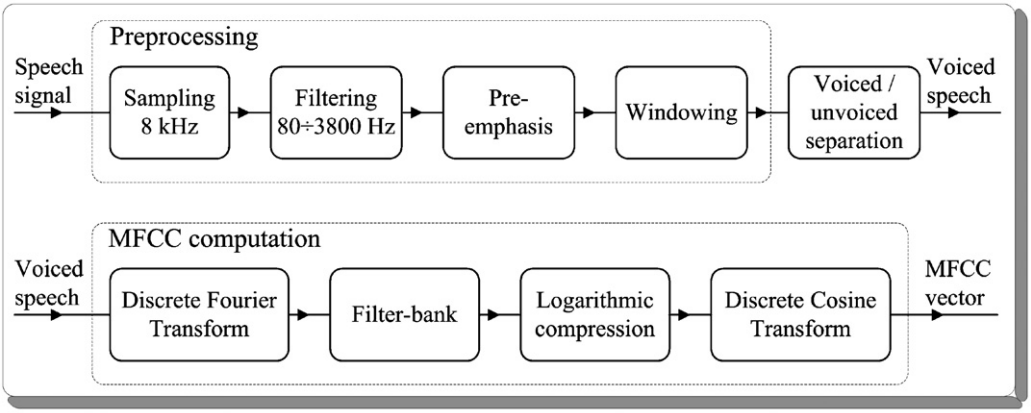

Fig. 7. Diagram of the speech preprocessing and MFCC computation.

[13], and the codebooks produced are further used for training the PNNs. In this way, the complexity of individual neural networks is greatly reduced and faster operation times are achieved. Specifically, in the WCL-1 system, we used a codebook of 128-vectors for the enrolled users, and a codebook of 256-vectors for the reference model. The size of the codebooks was chosen as a trade-off between computational demands and performance. A comprehensive description of the baseline speaker verification system, WCL-1, is available in [7]. A more sophisticated version is presented in [9].

In brief, the WCL-1 system operates as follows: Depending on the probability density function (4) computed for the user's and reference models, frame-by-frame decisions (8) are made for each segment of the speech signal. In the simplified baseline system employed here, these decisions are accumulated (17) over the whole test utterance. The final decision (19) is made with respect to a speaker independent threshold Θ. The speaker is rejected as an impostor when the probability (17) is below this threshold; otherwise, the speaker's identity claim is accepted.

5.3. SV performance assessment

Two types of errors can occur in the SV process. The first one, called a *false rejection* (FR) error, occurs when the true target speaker is falsely rejected as being an impostor, and as a result, the system misses accepting an attempt belonging to the true authorized user. The second type, called a *false acceptance* (FA) error, occurs when a try-out from an impostor is accepted as if it came from the true authorized user. The latter error is also known as a *false alarm*, because a non-target trial is accepted as a target one. The FR and FA are employed together to characterize the performance of the SV systems under investigation.

The equal error rate (EER) decision point is where the FR and the FA error probabilities are equal. When computing the EER, we assume equal values for the cost c_i (see Eq. (15)) for both the classes. In practice, the final decision is made at the location, where the distance between the FR and FA rates has its minimum. The EER decision point is widely accepted as balanced performance estimation, but its disadvantage is that the final decision is made a posteriori.

In the present work, the EER is paired off with the normalized detection cost C_{Norm} (35), a less intuitive function which is the official performance measure in the annual speaker recognition evaluation campaigns, e.g. [22]

$$C_{\text{Norm}} = \frac{c_{\text{FR}} P(FR|Tgt)P(Tgt) + c_{\text{FA}} P(FA|NonTgt)P(NonTgt)}{\min\{c_{\text{FR}} P(Tgt), c_{\text{FA}} P(NonTgt)\}}, \tag{35}$$

where c_{FR} and c_{FA} are the relative costs for detection errors FR and FA; $P(FR|Tgt)$ and $P(FA|NonTgt)$ are the probability of FR of a target and FA of an impostor trail; and finally, $P(Tgt)$ and $P(NonTgt) = 1 - P(Tgt)$ are the a priori probabilities of the target and impostor trials, respectively.

The C_{Norm} function offers multiple decision points, depending on the choice of the costs c_{FR} and c_{FA}, and threshold Θ. Specifically, the optimal detection cost, DCF_{opt}, gives impression about the prospective performance of a system if the *optimum* speaker-independent threshold is applied, and the DCF_{act} presents the actual performance of a system on a specific task, when a specific predefined threshold is applied.

Together the EER and DCF_{opt} offer a unique description of the SV performance convenient for ranking of different systems, which often exhibit different slope of the detection error trade-off (DET) curve. While the EER gives intuitive, balanced, and application-independent assessment of the potential performance of a system, DCF_{opt} and DCF_{act} are application-specific due to the detection costs c_{FR} and c_{FA} [22]. The ratio between these costs can vary from one application to another in the range of 1:10–10:1, depending on whether the emphasis is placed on security or comfort of use.

Both EER and DCF_{opt} utilize posteriori-computed thresholds, which results in an optimistic estimation of the absolute SV performance. However, since the present study aims at comparing the performances provided by various classifiers rather than comparing the absolute SV performance of different systems, the ordering and relative differences among the classifiers is of interest.

6. Speaker recognition database

Our experimentations (see Section 7) are based on the well-known PolyCost speaker recognition corpus [14]. In the present work, we use version v1.0 with bugs from 1 to 5 fixed.

PolyCost is comprised of real-world telephone quality speech recordings (English spoken by non-native speakers) collected across the international land-based telephone networks of Europe. The speech data are representative for research related to telephone-driven voice services and automated call-centers.

The database contains 1285 calls (around 10 sessions per speaker) recorded by 134 speakers (74 males and 60 females) from 13 different European countries. Each session comprises: 10 prompts with connected digits uttered in English, two prompts with sentences uttered in English, and two prompts in the speaker's mother tongue (17 different languages or dialects). One of the prompts in the speaker's mother tongue consists of free speech.

A detailed description of the PolyCost speaker recognition database is available in [14].

7. Experiments and results

The WCL-1 system outlined in Section 5.2 is used as a platform to compare the performance of various classifiers.

Initially, we study several variation operators for training the GLR PNN and the influence of the recurrence depth N over the SV performance. Next, a comparison between the distribution of output scores for the GLR PNN and original PNN is performed. Subsequently, the performance of the GLR PNN is contrasted to the one obtained by substituting the fully connected recurrent layer with several partially connected architectures, such as the DRNN, IIR and FIR MLP NN, as well as to the one obtained for the LR PNN. Finally, a comparison with a GMM-based classifier with a similar complexity is provided.

7.1. Experimental set-up

As it was discussed in Section 3, the GLR PNN inherits its pattern layer from the original PNN architecture. Therefore, for the purpose of fair comparison, we kept identical settings of the pattern layer in all experiments with the PNN and GLR PNN. This choice was motivated by our intention to evaluate the performance gain that is solely attributed to the ability of the GLR PNN architecture to exploit temporal correlations among the successive speech frames. Any difference in the smoothing parameters $\{\sigma_i\}$ during the comparative experiments might influence the performance of the classifiers, and consequently bias our conclusions. Consequently, the smoothing parameters $\{\sigma_i\}$ were set to the fixed value of 0.35, which provides a good approximation of the true underlying distribution of the training data.

In support of our decision to unify the parameters of the pattern layer are the following reasons: (1) In the present development the recurrent layer of the GLR PNN is optimized independently from the other layers (details are available in Section 4). (2) The various NN structures (FIR, IIR, DRNN, fully or partially connected recurrent layers) that are evaluated here share the same input data, which are formed by the pattern layer of the original PNN. (3) The present study aims at comparing the GLR PNN architecture to the classical PNN, rather than optimizing absolute SV performance.

The pattern layer of the GLR PNN was trained as described in Section 4, employing the feature vectors computed in Section 5.2. After that, the class-conditional probability for each speech frame is computed for the entire training set, and further serve as training data for the recurrent layer. When plenty of training data are available for each enrolled speaker, the recurrent layer's weights can be computed in a speaker-dependent manner. Since in the present experimental setup we deal with small amounts of training speech, the recurrent layer's weights were computed from a common data set, obtained through pooling together the training data of all enrolled speakers. In order to speed up the computation process, we retained only 25 000 training vectors—namely, 12 500 feature vectors for each class. Irrelevant outputs of the pattern layer are automatically discarded to facilitate recurrent layer training.

Common training and testing protocols were followed in all experiments. The speech recordings of the PolyCost v1.0 database were separated on non-overlapping training and testing sets, utilizing different sessions. The training data comprised of eighth utterances, obtained from the first two sessions of each speaker. In average, about 11 s of voiced speech per speaker were detected for creating each speaker model. The reference model was build from the combined training data of all enrolled users. In average, about 1.3 s of voiced speech per test utterance were available. The actual amount that was detected in the particular trials varied in the range of 0.4–2.1 s.

In total, there were 54 male and 45 female user models. Utterances from all the 134 speakers (74 males and 60 females) available in the database were used to perform test trials. We used 20 target trials per user model, taken from sessions 6–10 of the corresponding speaker. The number of non-target trials per model was 219 and 236, for the males and the females, respectively. The impostor trials for each model were made by both enrolled users claiming somebody else's identity and unknown to the system impostors. There were 20 males and 15 females which served as unknown impostors.

Summarily, there were 11 826 and 10 620 impostor trials, and 1080 and 900 target ones in the male and female experiments, respectively. The genders were evaluated separately and no cross-gender trials were performed.

In all experiments, the SV performance is reported in terms of both equal error rate and optimal detection cost DCF_{opt}.

7.2. Study of six variation operators for training the recurrent layer weights

Firstly, we tested the effectiveness of the variation operators (25)–(30), on the performance of the GLR PNN. Table 1 presents the EER and DCF_{opt} obtained for training the recurrent layer by each of the aforementioned variation operators, when a recurrence depth of one ($N = 1$) and one present and one delayed values of the recurrent layer inputs ($L = 1$) are considered. As Table 1 presents, for the both male and female experiments the lowest EER and DCF_{opt} is achieved for operator (25), followed by (30) and (27), and the highest for (28). The operator (26) performed almost as good as (27). Although our past experience suggested operator (28) as a candidate for the best performance, due to its good performance on numerous optimization tasks, it did not perform well here. For the operator (28), we have observed an almost perfect separation of the two classes for the training data, but also the highest error rate for the test data. The last observation suggests that when a relatively small amount of training data is available, preventive measures are required to assure that the recurrent layer weights do not become overspecialized on the training data, and thus unable to generalize. The operators (26) and (29) similarly to (28), are based on exploiting the best member ω_g^{best} of each

Table 1
The EER and DCF_{opt} depending on the variation operator

	Operator	Eq. (25)	Eq. (26)	Eq. (27)	Eq. (28)	Eq. (29)	Eq. (30)
Males	EER (%)	3.89	4.07	4.05	8.21	4.60	3.89
	DCF_{opt}	0.230	0.278	0.267	0.403	0.329	0.251
Females	EER (%)	4.70	5.00	4.98	5.12	5.10	4.78
	DCF_{opt}	0.378	0.396	0.391	0.412	0.394	0.400

Table 2
The EER and DCF_{opt} depending on the recurrence depth N

	N	0	1	2	3	4	5
Males	EER (%)	4.14	3.89	3.80	3.98	3.99	4.18
	DCF_{opt}	0.254	0.230	0.224	0.241	0.241	0.250
Females	EER (%)	5.11	4.70	5.00	5.01	5.00	5.10
	DCF_{opt}	0.387	0.378	0.422	0.416	0.426	0.422

generation, and therefore are also prone to overspecialization when they are not bounded by limitations. For achieving a high performance providing sufficient amount of training data is highly recommended, but when such data are not available, a restriction of the learning rate seems to be an effective solution for avoiding such an overspecialization.

7.3. The GLR PNN performance depending on the recurrence depth N

In principle, for optimizing the performance of the GLR PNN, the number of past input values L and the recurrence depth N can be set independently one from another. However, here we consider the simple case when these parameters are equal (i.e. $L = N$).

Table 2 presents the EER and the optimal detection cost DCF_{opt} obtained for GLR PNN-based speaker verification and various values of the recurrence depth N. In the present experimental setup, $N = 0$ corresponds to lack of recurrence (each speech frame is processed independently, exactly as in the original PNN), and $N = 5$ corresponds to 90 ms time window (one frame of 40 ms plus five overlapping frames shifted on 10 ms each).

As expected, when N increases—the EER decreases, because a larger part of the inter-frame correlation is identified and exploited. We deem, the increase of the EER observed for higher values of N, is mainly due to the insufficient amount of training data—only 12 500 train vectors per class were utilized. The number of weight coefficients $\left((L+N+1)\times K^2\right)$ in the recurrent layer depends in linear manner from N and L, but for large N and L more training data are required.

Another important constraint that restricts the recurrence depth is the time window length. As the experimental results presented in Table 2 suggest, in the male experiment, the best performance is observed for $N = 2$, which corresponds to 60 ms time window. In the female experiment the lowest EER and DCF_{opt} were observed for $N = 1$, which corresponds to a time window of 50 ms. We deem this difference between the male and female results is due to the gender-specific features of speech. Specifically, these are differences in the dynamics and the fundamental frequency of speech signal, as well as in the speaking style.

For even larger values of N and L the time window could spread across more than one phonemes, and even across syllables. In that case, the neural network becomes sensitive to the linguistic information carried by the training data, which can be very useful in the case of speech recognition or text-dependent SV, but decreases the SV performance when a text-independence is considered.

7.4. Performance comparison between the original PNN and the GLR PNN

The GLR PNN in its simplest form, with a recurrence depth one, $N = 1$, and one past value of the recurrent layer inputs, $L = 1$, trained by the variation operator (25) was compared with the original PNN. Fig. 8 presents the normalized distribution of the scores for the enrolled users (dashed line) and the impostors (solid line). The considerable spread of both users' and impostors' scores for the PNN case, shown in Fig. 8 (left), is obvious. In contrast, as Fig. 8 (right) illustrates, the GLR PNN classifier produces a smaller deviation from the mean value for both the users and the impostors. In about 55% of the cases, a zero probability for the impostor trials was produced, which is a major improvement compared to the only 38% of the original PNN. Moreover, the GLR PNN exhibited a significant concentration of the enrolled users' scores at the maximum probability point of one (in about 50% of all trials), in contrast to the PNN for which the user scores were spread out over a much wider area in the upper part of the scale. Therefore, not only a major concentration of the score distributions, but also a clearer separation of the two classes, and a decrease of the overlapping area were observed. For the male experiment, this is expressed in terms of EER (DCF_{opt}) as 4.14% (0.254) and 3.89% (0.230), for the PNN and GLR PNN, respectively.

Examining the score distributions for the PNN and GLR PNN (Fig. 8), we see that when compared to the baseline one, the performance of the GLR PNN-based system is less susceptible to the choice of decision threshold Θ. This virtue of the GLR PNN has an enormous practical value.

7.5. Comparing the fully linked recurrent layer with other recurrent structures

A comparative study of the GLR PNNs fully linked recurrent layer with other recurrent architectures was performed. The FIR and IIR structures [1], and the DRNN [16] one, were employed instead of the fully linked

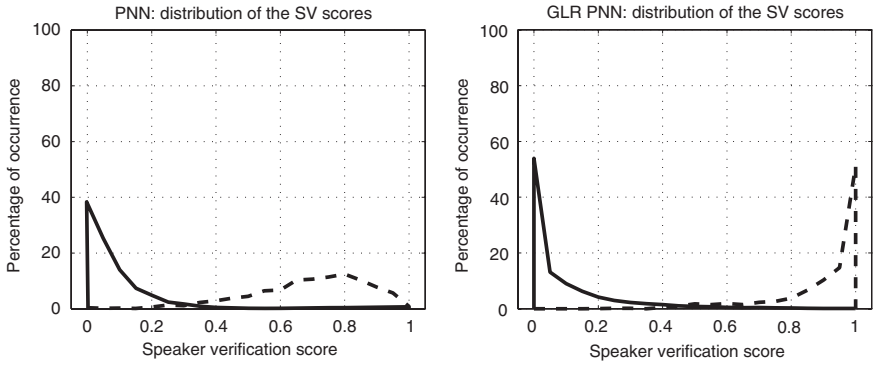

Fig. 8. Distribution of user (dashed line) and impostor (solid line) scores for the original PNN (left), and the GLR PNN for $N = 1$, $L = 1$ (right).

layer. For comprehensiveness on our study, the LR PNN architecture [10] was also included. Table 3 presents a performance comparison among these architectures over a common data set, and a recurrence depth $N = 1$ (for the GLR PNN, $N = 1$ and $L = 1$ were considered). Exploiting one past value of the input data, these structures also exploit one past value of their own output, or the outputs of all classes. The only exception here is the FIR synapse, which does not possess feedbacks. Thus, for the same time window, a different number of weighted connections are available in each structure. The symbol (*), next to the number of weight coefficients w, suggests that no biases were considered in the presented quantities.

As presented in Table 3, the best SV performance is achieved for the GLR PNN, followed by LR PNN, DRNN, IIR and FIR at the end. An increasing EER is observed, as the number of weighted connections decreases. The only deviation here is the LR PNN which possesses less connections than DRNN, but exhibits a better performance. In our opinion this is due to the presence of cross-class feedbacks from the past outputs of all classes in the LR PNN architecture. Thus, for the same size of the time window, the linkage of the LR PNN is better suited to capture the dynamics of the process.

Table 3
The EER and DCF_{opt} depending on the architecture

	Arch.	FIR	IIR	DRNN	LR PNN	GLR PNN
	# w^*	4	6	10	8	12
Males	EER (%)	4.45	4.17	4.02	3.98	3.89
	DCF_{opt}	0.307	0.268	0.249	0.241	0.230
Females	EER (%)	5.32	5.22	5.00	4.91	4.70
	DCF_{opt}	0.447	0.424	0.421	0.389	0.378

7.6. Comparison of the GLR PNN with a GMM-based classifier

For the sake of comparison the PNN classifier of the WCL-1 system was replaced by one, based on GMMs [25]. Initially, an equivalent complexity—128 mixtures for the user models, and 256 mixtures for the universal background model was considered. In attempt to optimize the SV performance, multiple other mixture sizes were tested. In the case when spherical kernels are used, the complexity of the GMM classifier is equivalent to the one of PNN. In the case of more sophisticated GMMs—when a diagonal or full covariance matrix is employed, the complexity is equivalent to the elliptical basis function network described in [17].

Table 4 presents the EERs and the detection cost DCF_{opt} that were obtained for the male voices. As shown in the table, the GMM-based system did not perform well here due to the insufficient amount of training data. When these results are compared to Table 3 we observe that the PNN-based structures (and the GLR PNN in particular) outperform entirely the GMM classifier.

Table 5 presents results from another experiment, where more training data were available. In particular, this time we have used 17 seconds of voiced speech for training, and 4 target and 292 non-target trials for testing, per user model. As shown in Table 5, when more training speech is available the GMM-based system significantly outperformed the baseline PNN-based one. When more sophisticated GMM models are trained, by exploiting a diagonal covariance matrix instead of the spherical-diagonal $\{\sigma^2 I\}$, EER drops to 2.60%, at the cost of multifold increase in the training time, and the amount of memory required for each speaker model. However, as presented in the table, the amount of training data was still insufficient for training GMMs with full covariance matrix.

In conclusion, summarizing the experimental results presented in Sections 7.2–7.6, we support the claim that the GLR PNN architecture outperforms the LR PNN and traditional PNN, and that the fully connected recurrent layer is more effective than the partially linked DRNN, IIR, and FIR structures. For small amounts of training data the original PNN and GLR PNNs demonstrate a better performance than the one of a GMM-based classifier. However, when more training data are available the best absolute performance was achieved by the GMM classifier with diagonal covariance matrix.

Table 4
The EER and DCF_{opt} for the GMM-based classifier

Kernel	Spherical diagonal			Diagonal covariance			Full covariance		
# mix.	32	64	128	8	16	32	2	4	8
EER (%)	4.93	4.56	4.60	5.44	4.77	6.64	7.23	8.34	11.2
DCF_{opt}	0.271	0.280	0.442	0.400	0.307	0.388	0.423	0.429	0.613

Table 5
The EER and DCF_{opt} for the PNN, GLR PNN, and GMM classifiers when more training data are available

	PNN	GLR PNN $N=1$	GMM Sph. diag. 128	GMM Diag. covar. 128	GMM Full covar. 128
EER (%)	3.43	3.07	3.01	2.60	8.14
DCF_{opt}	0.227	0.213	0.219	0.176	0.778

8. Conclusion

Introducing the generalized locally recurrent PNN, we extended the traditional PNN architecture to exploit the temporal correlation among the features extracted from successive speech frames. When compared to earlier work, a further development of the GLR PNN architecture and a revised training method was presented. Comparative experimental results for text-independent speaker verification (SV) confirmed the practical value of the proposed GLR PNN. For both male and female speakers, a better speaker verification performance expressed as a relative reduction of the EER with 9% was achieved, without significantly increasing the operational complexity of the original PNN, and without requiring additional training data. This work focused on the PNN-based structures (PNN, LR PNN, GLR PNN, etc). Due to their simpler training, the ability to learn from small amounts of data, and the intrinsic parallel architecture, they are very attractive for hardware implementations.

References

[1] A.D. Back, A.C. Tsoi, FIR and IIR synapses, a new neural network architecture for time series modeling, Neural Comput. 3 (3) (1991) 375–385.

[2] T. Bayes, An essay towards solving a problem in the doctrine of chances, Philos. Trans. R. Soc. London 53 (1763) 370–418.

[3] M. Berthold, J. Diamond, Constructive training of probabilistic neural networks, Neurocomputing 19 (1–3) (1998) 167–183.

[4] T. Cacoullos, Estimation of multivariate density, Ann. Instit. Statist. Math. 18 (1966) 179–189.

[5] P. Campolucci, A. Uncini, F. Piazza, B.D. Rao, On-line learning algorithms for locally recurrent neural networks, IEEE Trans. Neural Networks 10 (2) (1999) 253–271.

[6] B.J. Cain, Improved probabilistic neural networks and its performance relative to the other models, in: Proceedings SPIE, Applications of Artificial Neural Networks, vol. 1294, 1990, pp. 354–365.

[7] T. Ganchev, N. Fakotakis, G. Kokkinakis, Text-independent speaker verification based on probabilistic neural networks, in: Proceedings of the Acoustics 2002, Patras, Greece, September 30th–October 1st, 2002, pp. 159–166.

[8] T. Ganchev, N. Fakotakis, G. Kokkinakis, Comparative evaluation of various MFCC implementations on the speaker verification task, in: Proceedings of the SPECOM-2005, vol. 1, 2005, pp. 191–194.

[9] T. Ganchev, I. Potamitis, N. Fakotakis, G. Kokkinakis, Text-independent speaker verification for real fast-varying noisy environments, Int. J. Speech Technol. 7 (2004) 281–292.

[10] T. Ganchev, D.K. Tasoulis, M.N. Vrahatis, N. Fakotakis, Locally recurrent probabilistic neural networks for text independent speaker verification, in: Proceedings of the EuroSpeech-2003, pp. 1673–1676.

[11] T. Ganchev, D.K. Tasoulis, M.N. Vrahatis, N. Fakotakis, Generalized locally recurrent probabilistic neural networks for text independent speaker verification, in: Proceedings of the ICASSP-2004, vol. 1, 2004, pp. 41–44.

[12] V.L. Georgiou, N.G. Pavlidis, K.E. Parsopoulos, Ph.D. Alevizos, M.N. Vrahatis, Optimizing the performance of probabilistic neural networks in a bioinformatic task, in: Proceedings of the EUNITE 2004, 2004, pp. 34–40.

[13] J.A. Hartigan, M.A. Wong, A k-means clustering algorithm, Appl. Statist. 28 (1) (1979) 100–108.

[14] J. Hennebert, H. Melin, D. Petrovska, D. Genoud, Polycost: a telephone-speech database for speaker recognition, Speech Commun. 31 (1–2) (2000) 265–270.

[15] T. Jan, T. Yu, J. Debenham, S. Simoff, Financial prediction using modified probabilistic learning network with embedded local linear model, in: Proceedings of the IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, CIMSA 2004, Boston, MD, USA, 14–16 July, 2004.

[16] C.C. Ku, K.Y. Lee, Diagonal recurrent neural networks for dynamic system control, IEEE Trans. Neural Network 6 (1) (1995) 144–156.

[17] M.W. Mak, S.Y. Kung, Estimation of elliptical basis function parameters by the EM algorithm with application to speaker verification, IEEE Trans. Neural Networks 11 (4) (2000) 961–969.

[18] T. Masters, Practical Neural Network Recipes in C++, Academic Press, London, UK, 1993.

[19] W. Meisel, Computer-Oriented Approaches to Pattern Recognition, Academic Press, New York, 1972.

[20] A.M. Mood, F.A. Graybill, D.C. Boes, Introduction to the Theory of Statistics, McGraw-Hill, New York, USA, 1962.

[21] M. Musavi, K. Kalantri, W. Ahmed, Improving the performance of probabilistic neural networks, in: Proceedings of IEEE International Joint Conference on Neural Networks, Baltimore, MD, USA, vol. 1, June 7–11, 1992, pp. 595–600.

[22] NIST, The NIST year 2002 speaker recognition evaluation plan, National Institute of Standards and Technology of USA, February 2002, ⟨http://www.nist.gov/speech/tests/spk/2002/doc/2002-spkrec-evalplan-v60.pdf⟩.

[23] E. Parzen, On estimation of a probability density function and mode, Ann. Math. Stat. 33 (3) (1962) 1065–1076.

[24] L.R. Rabiner, M.J. Cheng, A.E. Rosenberg, C.A. McGonegal, A comparative performance study of several pitch detection algorithms, in: IEEE Transactions on Acoustics Speech and Signal Processing, vol. 24(5), 1976, pp. 399–418.

[25] D. Reynolds, R. Rose, Robust text-independent speaker identification using gaussian mixture speaker models, in: IEEE Transactions on Speech and Audio Processing, vol. 3(1), 1995, pp. 72–83.

[26] M. Slaney, Auditory Toolbox. Version 2, Technical Report 1998-010, Interval Research Corporation, 1998.

[27] D.F. Specht, Probabilistic neural networks, Neural Networks 3 (1) (1990) 109–118.

[28] D.F. Specht, A general regression neural network, IEEE Trans. Neural Networks 2 (6) (1991) 568–576.

[29] D.F. Specht, Enhancements to probabilistic neural networks, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Baltimore, MD, June 7–11, 1992.
[30] D.F. Specht, H. Romsdahl, Experience with adaptive PNN and adaptive GRNN, in: Proceedings of the IEEE International Conference on Neural Networks, Orlando, FL, vol. 2, June 28–July 2, 1994, pp. 1203–1208.
[31] R. Storn, K. Price, Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[32] B. Tian, M.R. Azimi-Sadjadi, T.H. Vonder Haar, D. Reinke, Temporal updating scheme for probabilistic neural network with application to satellite cloud classification, IEEE Trans. Neural Networks 11 (4) (2000) 903–920.
[33] Z.R. Yang, S. Chen, Robust maximum likelihood training of heteroscedastic probabilistic neural networks, Neural Networks 11 (4) (1998) 739–748.
[34] A. Zaknich, The modified probabilistic neural network for signal processing and pattern recognition, Ph.D. Dissertation, Department of Electrical and Electronics Engineering, University of Western Australia, Nedlands, May 1995.
[35] A. Zaknich, C. deSilva, Y. Attikiouzel, The probabilistic neural network for nonlinear time series analysis, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Singapore, November 17–21, 1991, pp. 1530–1535.

Todor D. Ganchev received his Diploma Engineer degree in Electrical Engineering from the Technical University of Varna, Bulgaria, in 1993. From February 1994 to August 2000, he was with Technical University of Varna, where he consequently occupied engineering, research, and teaching staff positions. During the period, his research activities were mainly in the area of low-bit-rate speech coding. Since September 2000, he is with the Wire Communications Laboratory, University of Patras, Greece. In 2005 he received his Ph.D. degree in the area of Speaker Recognition. Presently, he is a post-doctoral researcher at the same laboratory. His current research interests include Speech Processing, Neural Networks, and Differential Evolution.

Dimitris K. Tasoulis received his Degree in Mathematics form the Department of Mathematics, University of Patras, Greece in 2000. He is currently a post-graduate student in the post-graduate course ''Mathematics of Computers and Decision Making'' from which he was awarded a postgraduate fellowship. His research activities focus on Unsupervised Clustering, Neural Networks, Data-Mining and Applications. He was a Visiting Research Fellow at the INRIA, Sophia-Antipolis, France, in 2003. He is co-author of more than 50 publications (12 of which are published in international refereed journals). His research publications have received more than 40 citations.

Michael N. Vrahatis is a Professor at the Department of Mathematics, University of Patras, Greece, since August 2000. He received the Diploma and Ph.D. degrees in Mathematics from the University of Patras, in 1978 and 1982, respectively. He is the author or (co-author) of more than 270 publications (more than 120 of which are published in international refereed journals) in his research areas, including computational mathematics, optimization, neural networks, evolutionary algorithms, data mining, and artificial intelligence. His research publications have received more than 1000 citations. He has been a principal investigator of several research grants from the European Union, the Hellenic Ministry of Education and Religious Affairs, and the Hellenic Ministry of Industry, Energy, and Technology. He is among the founders of the ''University of Patras Artificial Intelligence Research Center (UPAIRC)'', established in 1997, where currently he serves as director. He is the founder of the ''Computational Intelligence Laboratory (CI Lab)'', established in 2004 at the Department of Mathematics of University of Patras, where currently he serves as director.

Nikos D. Fakotakis received the B.Sc. degree in Electronics from the University of London (UK) in 1978, the M.Sc. degree in Electronics from the University of Wales (UK), and the Ph.D. degree in Speech Processing from the University of Patras, Greece, in 1986. From 1986 to 1992 he was lecturer in the Electrical and Computer Engineering Department of the University of Patras, from 1992 to 1999 Assistant Professor, from 2000 to 2003 Associate Professor, and since 2004 he has been Professor in the area of Speech and Natural Language Processing and Head of the Speech and Language Processing Group at the Wire Communications Laboratory. He is author of over 200 publications in the area of Signal, Speech and Natural Language Processing. His current research interests include Speech Recognition/Understanding, Speaker Recognition, Speech Modeling, Spoken Dialogue Processing, Natural Language Processing and Optical Character Recognition. Dr. Fakotakis is a member of the Executive Board of ELSNET (European Language and Speech Network of Excellence), Editor-in-Chief of the ''European Student Journal on Language and Speech'', WEB-SLS.