Locally Recurrent Probabilistic Neural Network for Text-Independent Speaker Verification

Todor Ganchev[1,3], Dimitris K. Tasoulis[2,3], Michael N. Vrahatis[2,3], Nikos Fakotakis[1,3]

[1]Wire Communications Laboratory, [2]Department of Mathematics,
[3]University of Patras Artificial Intelligence Research Center,
University of Patras, GR-26500 Patras, Greece

tganchev@wcl.ee.upatras.gr

Abstract

This paper introduces Locally Recurrent Probabilistic Neural Networks (LRPNN) as an extension of the well-known Probabilistic Neural Networks (PNN). A LRPNN, in contrast to a PNN, is sensitive to the context in which events occur, and therefore, identification of time or spatial correlations is attainable. Besides the definition of the LRPNN architecture a fast three-step training method is proposed. The first two steps are identical to the training of traditional PNNs, while the third step is based on the Differential Evolution optimization method. Finally, the superiority of LRPNNs over PNNs on the task of text-independent speaker verification is demonstrated.

1. Introduction

The contemporary classification techniques, such as Vector Quantization (VQ) and Gaussian Mixture Models (GMM), used in the text-independent speaker verification task, assume context independence between neighbouring feature vectors extracted from a speech utterance. It is well-known, that speech signals contain an abundance of short- and long-term correlations, which if identified could be exploited to produce a superior speaker verification performance.

At present, the most popular speech features, like cepstral coefficients and PLP, used in both speaker and speech recognition tasks, represent the static spectrum for a given speech frame. The Δ and Δ^2 derivatives of these parameters are widely used in addition to the basic parameters in an attempt to capture the inter-frame correlation inherent to the speech signal. A more effective approach to utilize inter-frame information is to use a classifier sensitive to it. For example: HMM, time-delay neural networks or recurrent neural networks. Here we only consider neural networks. Time-delay neural networks are able to capture the inter-frame correlations at the cost of a significant increase of network size and computational requirements, when compared to their static counterparts. Recurrent neural networks are much more efficiently, but suffer from stability problems, and their training is computationally more demanding compared to time-delay neural networks.

In this work, we propose a locally recurrent global-feedforward PNN-based classifier, combining the desirable features of both feedforward and recurrent neural networks. More specifically, we extend the traditional PNN architecture, proposed by Specht [1], to Locally Recurrent PNN (LRPNN), in order to capture the inter-frame correlations present in speech signals, without imposing extra computational burden for training.

The local recurrent global-feedforward architecture was originally proposed by Back and Tsoi [2], who considered an extension of the Multilayer Perceptron (MLP) neural network to exploit contextual information. In the work of Back and Tsoi each recurrent neuron has only connections to his own current and delayed inputs and outputs. The work presented here is based on the local recurrent global-feedforward architecture and the locally recurrent layer we introduce is similar to the IIR synapse proposed in [2]. Our approach differs from the one proposed in [2] primarily because we consider PNNs instead of MLPs. Most importantly, though, in our architecture each summation unit in the recurrent layer receives as input not only current and past values of its input and output, but it is also fully connected to the other neurons of the same layer. In other words, each neuron in the recurrent layer also receives as input the previous output of all other neurons in that layer. Overall, the input signal, acting on a recurrent neuron located in the third hidden layer of a LRPNN, is a sum of two differences. The first difference is between the weighted probability of the given class and the sum of probabilities computed for all the other classes. The second difference is between the past output values of the given unit and the sum of the past output values of all other neurons in this layer.

Finally, a comparative evaluation of LRPNN's performance, with that of PNNs, was performed, on the task of text-independent speaker verification. The experimental results are discussed in Section 4.

2. The LRPNN architecture

The LRPNN is derived from the PNN by including a third hidden layer, which consists of summation neurons possessing feedbacks. The LRPNNs, as their predecessor -- the PNNs, implement the Parzen window estimator by using a mixture of Gaussian basis functions (see [1] for details). If a LRPNN for classification in K classes is considered, the probability density function $f_i(\boldsymbol{x}_p)$ of each class κ_i is defined by (1), where

$$f_i(\boldsymbol{x}_p) = \frac{1}{(2\pi)^{d/2}\sigma_i^d}\frac{1}{M_i}\sum_{j=1}^{M_i}\exp\left[-\frac{1}{2\sigma_i^2}(\boldsymbol{x}_p-\boldsymbol{x}_{ij})^T(\boldsymbol{x}_p-\boldsymbol{x}_{ij})\right], \quad i=1,2,...,K \tag{1}$$

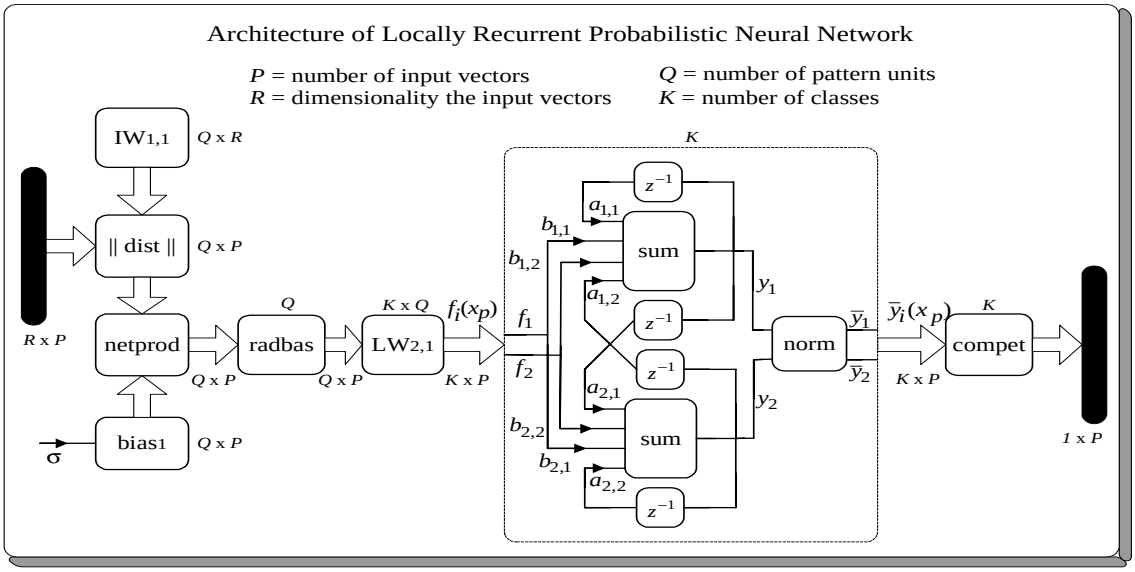

Figure 1: Architecture of the Locally Recurrent Probabilistic Neural Network

$\boldsymbol{x}_{ij}$ is the *j*-th training vector from class κ_i, $\boldsymbol{x}_p$ is the *p*-th input vector, d is the dimension of the speech feature vectors, and M_i is the number of training patterns in class κ_i. Each training vector $\boldsymbol{x}_{ij}$ is assumed to be a centre 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 of the pattern units for all the classes. The variance σ_i acts as a smoothing factor, which softens the surface defined by the multiple Gaussian functions. The value of σ_i is the same for all pattern units from a specific class, or, as it was originally proposed by Specht, it can be the same for all pattern units regardless of the classes they belong to. The architecture of the LRPNN for the case of two classes (K=2) and one-time-step-deep recurrence (N=1), is shown in Fig. 1. The dashed line delineates the locally recurrent layer.

The summation units output $y_i(x_p)$ of the locally recurrent layer is computed by (2), where $f_i(\boldsymbol{x}_p)$ is the probability density function of each class k_i, $\boldsymbol{x}_p$ is the input vector, K is the number of classes, N is the recurrence deepness, z^{-t} is a time delay of t steps, and $a_{i,i,t}$ and $b_{i,i}$ are weight coefficients. Further, the output $y_i(x_p)$ of each summation unit is subject to a regularization transformation $\overline{y}_i(\boldsymbol{x}_p) = sgm(y_i(\boldsymbol{x}_p)) \Big/ \sum_{i=1}^{K} sgm(y_i(\boldsymbol{x}_p))$, which keeps the probabilistic sense at the recurrent layer output. The designation *sgm* stands for a sigmoid activation function. As a whole, the recurrent layer can be considered as a form of Infinite Impulse Response filter, which smoothes the difference between the probabilities generated for each class, by using one or more past values of the summation outputs.

Finally, the Bayesian decision rule (3) is applied to distinguish class k_i, to which the input vector $\boldsymbol{x}_p$ belongs:

$$D(\boldsymbol{x}_p) = \underset{i}{\operatorname{argmax}}\{h_i c_i \overline{y}_i(\boldsymbol{x}_p)\},\ i = 1, 2, ..., K \tag{3}$$

where h_i is a-priori probability of occurrence of the patterns of category k_i, and c_i is the cost function in case of misclassification of a vector belonging to class k_i. The averaged for all test vectors $\boldsymbol{X} = \{\boldsymbol{x}_p\}$, $p = 1, 2, ..., P$ probability $\mathrm{P}(k_i \mid \boldsymbol{X})$, $\boldsymbol{X}$ to belong to class k_i is computed by:

$$\mathrm{P}(k_i \mid \boldsymbol{X}) = \frac{1}{P} \sum_{p=1}^{P} \boldsymbol{D}(\boldsymbol{x}_p) \tag{4}$$

3. The LRPNN training

A three-step training procedure for the LRPNN is proposed. By analogy to the original PNN, the first training step creates the actual topology of the network. 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 one of the second hidden layer summation units. The number of summation units is equal to the number of classes K. We consider a modification of the PNN, where only the n-best results are summed together, with n usually ranging between 1 and 6.

The second training step is to compute the smoothing parameter σ_i for each class, through Cain's rule [3]. Therefore, σ_i is proportional to the mean value of the minimum distances between the training vectors in class κ_i:

$$y_i(\boldsymbol{x}_p) = \left[b_{i,i}\, f_i(\boldsymbol{x}_p) - \sum_{\substack{k=1 \\ i \neq k}}^{K} b_{i,k}\, f_k(\boldsymbol{x}_p) \right] + \sum_{t=1}^{N} \left[a_{i,i,t}\, y_i(\boldsymbol{x}_p) z^{-t} - \sum_{\substack{k=1 \\ i \neq k}}^{K} a_{i,k,t}\, y_k(\boldsymbol{x}_p) z^{-t} \right],\quad i = 1, 2, ..., K \tag{2}$$

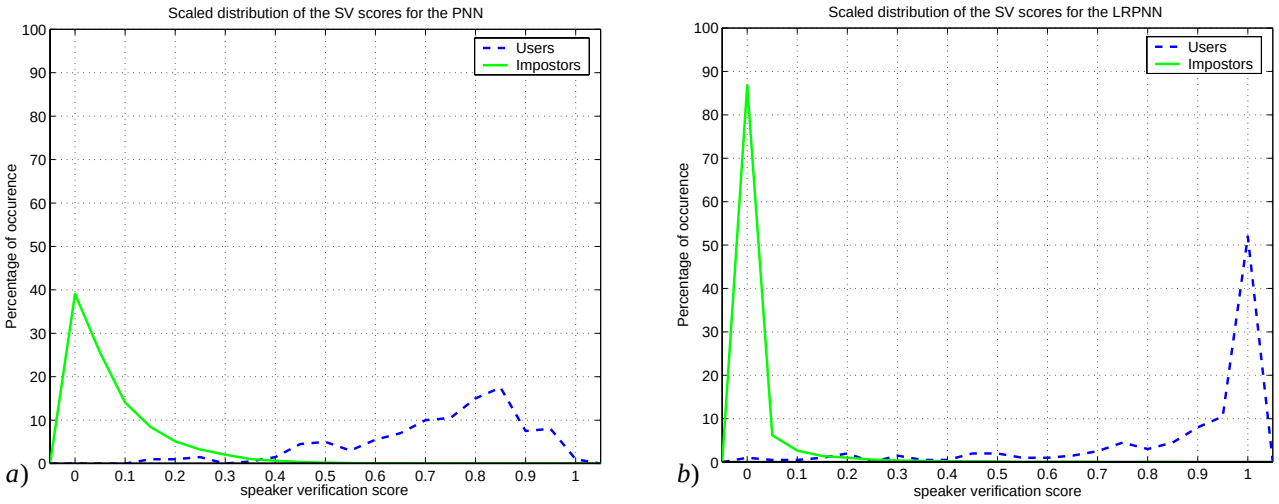

Figure 2: Speaker verification score distribution for the: a) PNN system, and b) LRPNN system.

$$\sigma_i = \lambda \frac{1}{M_i} \sum_{j=1}^{M_i} d_{ij} \tag{5}$$

where d_{ij} is the minimum Euclidean distance for each pattern unit from class k_i, with the pattern units from the same class. Usually, the constant λ is between 1.1 and 1.4. In case σ_i is common for all classes it is chosen empirically, or it is computed by applying (5) on the entire training data.

The third step is adjusting the weights of the locally recurrent layer by using the same training data, exploited at step one to construct the vectors utilized in the Radial Basis layer training. As a training method, we selected the Differential Evolution method (DE) introduced by Storn and Price [4]. Supervised training of the recurrent layer is equivalent to the minimization of the error function (6):

$$E(w) = \sum_{i=1}^{K} c_i P(Miss \mid k_i) P(k_i) \tag{6}$$

where the parameter c_i is the relative cost of detection error for the corresponding class k_i, $P(Miss \mid k_i)$ 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 the patterns of class k_i into the training data set. The values of $P(Miss \mid k_i)$ are obtained in the following way: For a given weight vector $\boldsymbol{w}=\{\boldsymbol{a}, \boldsymbol{b}\}$, the values of y_i are computed, according to (2), and then (3) is applied. Finally, $P(Miss \mid k_i)$ is computed as $P(Miss \mid k_i)=1-P(k_i \mid \boldsymbol{X})$, where $P(k_i \mid \boldsymbol{X})$ is obtained from (4) for the case of the training data set.

The total error $E(w)=E(a,b)$ is reduced by adjusting the weight vectors $\boldsymbol{b}$ and $\boldsymbol{a}$ by means of the DE algorithm. The DE method exploits a population of potential solutions, which have a fixed number of members over the whole training process, to probe the function space. At each iteration, called generation g, three steps, named *mutation*, *recombination*, and *selection* are performed [5]. First, all weight vectors are initialized by using a random number generator. Then at the mutation step, new mutant weight vectors v_{g+1}^i are generated by combining weight vectors, randomly chosen from the population:

$$v_{g+1}^i = w_g^i + \mu(w_g^{best} - w_g^i) + \mu(w_g^{r1} - w_g^{r2}) \tag{7}$$

where w_g^{r1} and w_g^{r2} are two randomly selected vectors, different from w_g^i, w_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=1,2,…,L 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 predefined crossover constant p, the j-th component of the mutant vector v_{g+1}^i becomes j-th component of the trial vector. Otherwise the j-th component is picked up 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.

4. Experiments and results

Our text-independent speaker verification system WCL-1 [6], which participated in the 2002 NIST Speaker Recognition Evaluation [7], was used as a platform to compare the performances of the LRPNN and the traditional PNN.

In the experiments, the LRPNN in its simplest form, with only one-step-deep recurrence (N=1), was compared with the traditional PNN. In the speaker verification task we consider two classes (K=2), one for the enrolled user and one for the collective model of non-users.

Fifty male speakers, extracted from the PolyCost v1.0 telephone-speech speaker recognition corpus [8], were enrolled as authorized users. As a training data, ten utterances obtained from the first session of each speaker, containing both num-

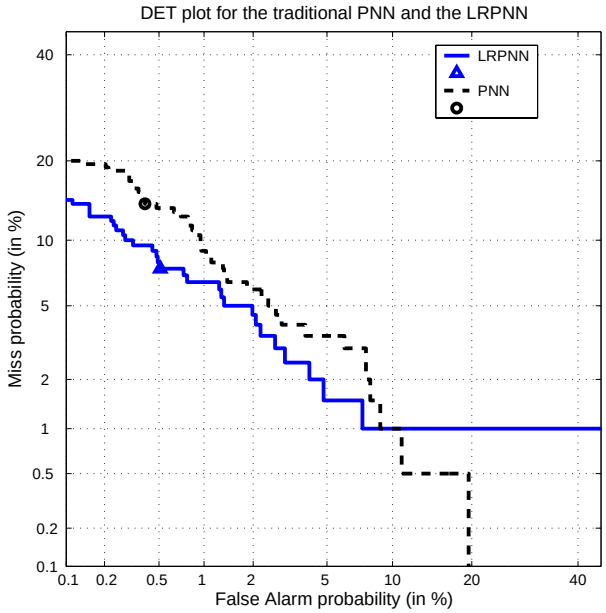

Figure 3: DET plots for the PNN and LRPNN systems.

bers and sentences, were used. In average, about 17 seconds of voiced speech per speaker were available for training each user model. Utterances from all male speakers available in the database were used to perform test trials. Each user model was tested by four utterances extracted from the second session of the corresponding enrolled user, and by 292 trials from impostors. About 1.3 seconds of voiced speech per test utterance were available. The actual amount of voiced speech in the particular trials was in the range of 0.4 to 2.1 seconds. Impostor trials from opposite sex speakers were not performed.

Fig. 2 presents the normalized distribution of the scores for the enrolled users (dashed line) and the impostors (solid line). The considerable spread of the scores of both users' and impostors' for the PNN case, shown in Fig. 2 *a*), is obvious. In contrast, as Fig. 2 *b*) demonstrates, the LRPNN classifier produces a smaller deviation from the mean value for both the users and the impostors. In 86% of the cases, a zero probability for the impostor trials was produced, which is a major improvement compared to the only 40% of the traditional PNN. Moreover, the LRPNN exhibited a significant concentration of the enrolled users' scores at the maximum probability point (more than 50 % of all trials), in contradistinction to the PNN case where the user scores were spread out over a much wider area in the upper part of the scale. Therefore, not only major concentration of the score distributions, but also a clearer separation of the two classes, and a decrease of the overlapping area were detected.

In Fig. 3, the Detection Error Trade-Off (DET) curves for the baseline PNN (dashed line) and the LRPNN (solid line) are shown. The 'triangle' and the 'circle' marks in the DET plot show the optimal performance point for the corresponding system (as it is defined in [7]). At the optimal performance point, an improvement of the absolute speaker verification performance by more than 5 % for the LRPNN, when compared to the PNN, is observed. This corresponds to relative reduction of the error rate by more than 30 %.

From a practical point of view, the low false acceptance rate section of the DET plot, which corresponds to the high security zone, is more important. In authorised access applications false acceptance rates of around 0.1% or less are considered plausible. At that level of security the proposed LRPNN in its simplest form, with one level deep recurrence, has exhibited a significant advantage over the PNN considered here. In fact the improvement was smaller than expected, due to some files containing saturated speech signal, for which the target users' models in both the systems produced extremely low probabilities.

5. Conclusion

Introducing the Locally Recurrent PNN, we extended the traditional PNN architecture to exploit the inter-frame correlation among the features extracted from successive speech frames. Beside the LRPNN architecture definition, a fast three-step training method was proposed. Comparative experimental results for text-independent speaker verification were presented. They demonstrated superiority of the new LRPNN architecture over the traditional PNN one. Reduction of the absolute error rate by more than 5 %, at the optimal decision point, due to better grouping of both classes of users' and impostors' scores were observed. This corresponds to relative decrease of the error rate by more than 30 %.

6. Acknowledgement

This work was supported by the "Infotainment management with Speech Interaction via Remote microphones and telephone interfaces" - INSPIRE project (IST-2001-32746).

7. References

[1] Specht, D. F., "Probabilistic Neural Networks", *Neural Networks*, 3(1): 109-118, 1990

[2] Back, A. D., Tsoi, A. C., "FIR and IIR Synapses, a New Neural Network Architecture for Time Series Modeling", *Neural Computation*, 3: 375-385, 1991

[3] Cain, B. J., "Improved probabilistic neural network and its performance relative to the other models", In *Proc SPIE, Applications of Artificial Neural Networks*, vol.1294, 354-365, 1990

[4] Storn, R. and Price, K., "Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces", *Journal of Global Optimization*, 11: 341-359, 1997

[5] Plagianakos, V.P. and Vrahatis, M.N., "Parallel evolutionary trained algorithms for 'hardware-friendly' neural networks", *Natural Computing*, 1: 307-322, 2002.

[6] Ganchev T., Fakotakis N., Kokkinakis G., "Text-Independent Speaker Verification Based on Probabilistic Neural Networks", In *Proc. of the Acoustics 2002*, Patras, Greece, 159-166, 2002

[7] "The NIST Year 2002 Speaker Recognition Evaluation Plan", *NIST of USA*, February 2002. Available: http://www.nist.gov/speech/tests/spk/2002/doc/2002-spkrec-evalplan-v60.pdf

[8] Hennebert J., Melin H., Petrovska D., Genoud D., "POLYCOST: A Telephone-Speech Database for Speaker Recognition", *Speech Communication* 31: 265-270, 2000