HEURISTIC SEQUENTIAL CLASSIFIERS BASED ON MULTILAYER PERCEPTRON AND PROBABILISTIC NEURAL NETWORK

TOMASZ WALOKWIAK
Institute of Engineering Cybernetics
Wroclaw University of Technology
ul. Janiszewskiego 11/17,
50-372 Wroclaw, Poland
e-mail: twalkow@ict.pwr.wroc.pl

WOJCIECH ZAMOJSKI
Institute of Engineering Cybernetics
Wroclaw University of Technology
ul. Janiszewskiego 11/17
50-372 Wroclaw, Poland
e-mail: zamojski@ict.pwr.wroc.pl

ABSTRACT

This paper presents a method of sequential classification. We assumed that we have a classifier which gives classification results for a single feature vector. We are analyzing five different algorithms of integrating these single classification outputs to give a final classification decision after a given sequence of feature vectors. As a base classification rule (for single feature vector) we used the multilayer perceptron and the probability neural network (PNN). The performance of the proposed integration of sequences of classification outputs from neural network classifiers was tested in an automatic, text independent, speaker identification task. Achieved results are presented..

Keywords: sequential classification, multilayer perceptron, Probabilistic Neural Network, speaker recognition

1 INTRODUCTION

In this paper, we would like to focus on the problem of object recognition [1] based on some signals incoming in time. The real examples of such problem are speaker recognition, recognition of vehicles based on acoustic signals or recognition of vehicles based on seismic signals. Assume, that we have a signal generated by the object, which could be interpreted as a change of some physical parameter in time (i.e. pressure of air or ground vibrations). The aim is to classify the signal to a given group, which represents the object generating the signal. One solution of such problem could be so called long period recognition, which is based on estimating some parameters for long periods of signal. However, we want to focuses on the other approach. Assume that we have a method, which extracts some parameters from short parts of signal. For each parameter vector, representing a short part of signal, we could have a classifier (in this case we will focus on neural networks [1,2]), which generates classification answers. However, it is worth to investigate the classification results for longer parts of signals. Therefore, a method for sequential integration of classification results will be required. It could be assumed, that integration of several classification results for following parameter vectors would give much better results than results based on one parameter vector.

In this paper we present a set of heuristic sequential classifiers which allows solving described above problem. The framework of the classification problem with predefined classes is as follows. Certain objects are to be classified as coming from one of a fixed number of classes. Each object gives rise to certain measurements forming a feature vector. This is an input for a classification rule. We investigate, in chapter 2, two kinds of classifiers probabilistic neural networks [3,4] and classical multilayer perceptron [1,2]. Answers from these neural network classifiers for following feature vectors are integrated by a sequential classification rules (chapter 3) giving the final classification answer. Performance of proposed algorithms is tested in an automatic, text independent, speaker recognition task. The speaker recognition system uses an auditory based pre-processing described in chapter 4. The achieved results for speaker identification are presented in chapter 5.

2 NEURAL NETWORK CLASSIFIERS

2.1 MULTILAYER PERCEPTRON

One of the most popular neural networks is a multilayer perceptron (MLP). It is a very useful tool in approximation and recognition problems [1,2]. The topology of this net is a structured hierarchical layered network (Fig. 1). It consists of several distinct layers of nodes including an input layer and an output layer. Connections within a layer or from higher to lower layers are not permitted. Each node in a layer is connected to all the nodes in the layer above it. The algorithms for multilayer perceptron processing can be divided into two phases: learning and working (giving answers for specified stimulates). The output of each neuron is calculated based on the outputs of all neurons connected with that one and the weights on those connections. A rule that defines the output of each neuron is called the update rule. There are a variety of different kinds of model update rules, such as sigmoid or tangent hyperbolic.

Training is equivalent to finding proper weights for all the connections such that a desired output is generated for a corresponding input. Multilayer perceptrons are used because there is known learning algorithm for them named as Back Propagation Algorithm [2]. The idea of this algorithm is learning proper weights by computing the discrepancy between the desired and actual outputs and feeding back this error signal level-by-level to the inputs, changing the connection weights in such a way as a modify them in proportion to their responsibility for the output error. There exist a large number of different representations of the error measure as a function of connection weights ($\boldsymbol{w}$). The most common function is given by a sum of the square of the difference between desired (d_i) and actual (o_i) output over all output units (index i) and all examples (index j):

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_j \sum_i (d_i^j - o_i^j)^2 .$$

We used a two-layer perceptron for classification of features vectors. Number of neurones in input layer depends on the pre-processing algorithm. The number of inputs equals to the length of the feature vector. The net has as many output neurons as many classes should recognize. It is expected to light only one proper output neuron (answer is in code 1-of-n). It means that classifier recognizes class k when k-th neuron is the most active.

For training the Levenberg-Marquardt method has been chosen [2].

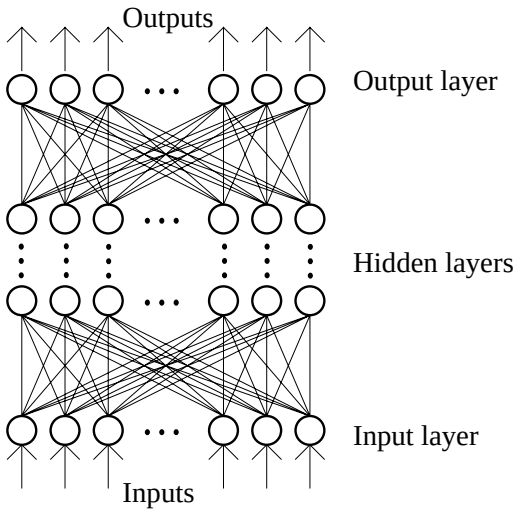

Fig. 1. Neural network for classification of feature vectors

2.2 PROBABILISTIC NEURAL NETWORKS

Assume that we have some objects from one of 1,...,K classes described by a feature vector $\boldsymbol{x}$. The process of classification could be understood as taking one of K possible decisions. Assuming that a probability density function for each class is given (denote it by $f_i(\boldsymbol{x})$) optimal (minimum-error-rate) Bayes theory (described in many standard textbooks, for example [2]) could be used for classification. If there is no cost associated with being correct, all the prior probabilities (probabilities that an example is drawn from a given class) are equal, the optimal Bayes decision rule is choosing a class with largest values of probability density function. Therefore the Bayes rule is given by:

$$\boldsymbol{x} \text{ belongs to class} = \left\{ k \quad \text{if } f_k(\boldsymbol{x}) = \max_j f_j(\boldsymbol{x}) \right. .$$

Having some estimator of probability density function for each class the optimal Bayes decision rule could be a bases for a probabilistic neural network (PNN) presented in Fig. 2.

The density estimator for each class is independent from the other classes. Therefore, from this point we will be talking only about the density for one class.

Assume, that an unknown density could be represented as a linear combination of component densities $F(\boldsymbol{x},\theta_i)$ (whereas θ_i is a parameter vector):

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} a_i F(\boldsymbol{x}, \theta_i),$$

where a_i presents the *prior* probability of the data having been generated from component i of the mixture. These priors must sum to 1.

The component densities (kernels) $F(\boldsymbol{x},\theta_i)$ could be selected from any density function. However, we limit our attention to the Gaussian distribution function.

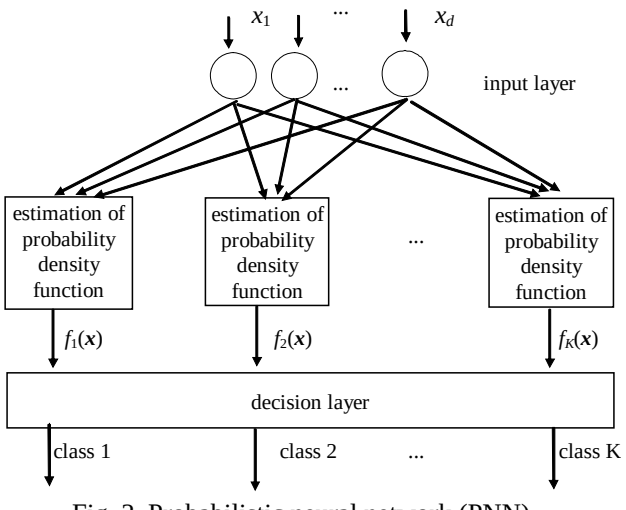

Fig. 2. Probabilistic neural network (PNN)

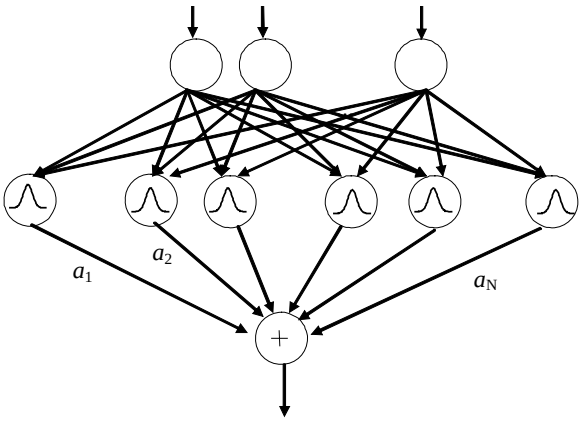

Fig.3. The RBF network for density estimation

Moreover, the covariance matrix of Gaussian function will be limited to a diagonal matrix (a matrix having a non-zero values only on its diagonal) so that Σ=diag(s^1, s^2,..., s^d) and hence:

$$F(\boldsymbol{x},\theta_i) = F(\boldsymbol{x},\boldsymbol{m}_i,\boldsymbol{s}_i) = \frac{1}{(2\pi)^{d/2}\prod_{j=1}^{d} s_i^j} \exp\left\{-0.5\sum_{j=1}^{d}\left(\frac{x^j - m_i^j}{s_i^j}\right)^2\right\}.$$

Based on above equations a feed-forward RBF neural network (RBFNN) presented in Fig. 3 could be defined. The number of neurons in the input layer is set equal to a dimension of the feature vector (denoted here as *d*). Each neuron from input layer is connected with each of *N* neurons in the hidden layer. With each hidden neuron a centre vector $\boldsymbol{m}_i$ and a width vector $\boldsymbol{s}_i$ is associated. The activation of the each hidden neuron is defined by the above equation. All hidden neurons activation are multiplied by weights a_i and summed giving an output from the network.

The PNN network in Fig. 2 requires a separate RBFNN for each class. The parameter values a_i, $\boldsymbol{m}_i$, $\boldsymbol{s}_i$ for each RBFNN are determinated separately based on the training set for a given class. This is done using a probability density estimation algorithm, described in work [5], which lays in the framework of Expectation-Maximisation (EM) algorithm [6].

3 SEQUENTIAL RECOGNITION ALGORITHMS

As it was stated in the introduction, we want to integrate a sequence of answers from classifiers to make the final classification decision. At first, we will introduce some notations. Let $Cer_k(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N)$ be a value, we will called it certainty of class *k* for *N* following features vectors $\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N$, which is a base for the final sequential classification. We propose to use a heuristic sequential classification rule, which resembles the Bayes rule, i.e. select a class with highest certainty:

$\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N$ belongs to class $= k$
if $Cer_k(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N) = \max_{j=1..K} Cer_j(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N)$.

Certainty for class *k* is calculated based on answer of the neural network output $f_k(\boldsymbol{x}_n)$ (value from of *k*-th output neuron in case of MLP and the output of *k*-th RBF network in case of PNN) for each feature vector $\boldsymbol{x}_n$ (*n*=1..*N*) and the classification result for class *k* , i.e.:

$$c_k(\boldsymbol{x}_n) = \begin{cases} 1 & \text{if } f_k(\boldsymbol{x}_n) = \max_{j=1..K} f_j(\boldsymbol{x}_n) \\ 0 & \text{elsewhere} \end{cases}.$$

We propose five algorithms for calculation of the certainty of class *k* for *N* following features vectors $\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N$:

Algorithm 1. Summing of classifier outputs:

$$Cer_k(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N) = \sum_{n=1}^{N} c_k(\boldsymbol{x}_n).$$

Algorithm 2. Summing of neurons outputs:

$$Cer_k(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N) = \sum_{n=1}^{N} f_k(\boldsymbol{x}_n).$$

Algorithm 3. Multiplying of neurons outputs:

$$Cer_k(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_N) = \prod_{n=1}^{N} f_k(\boldsymbol{x}_n).$$

In case of PNN, this is justify by the probabilistic rule, stating that the cumulative probability density is equal for the multiplication of each probability density function in case when the following feature vectors $\boldsymbol{x}_n$ are probabilistic independent. It is worth to mention that this assumption is not always true in case of sequential classification. This algorithm was proposed by us in [4].

Next, two algorithms, proposed in [7], combines the classifier outputs of *N* previous feature vectors, understood as long history of classification, with *M* (where 0<*M*<*N*) last outputs, say short history. It could be useful for detection of class changes for on-line classification, when at given moment a object generating signals changes to another. They are recurrent algorithms and could be calculated only for *N*>*M*, for *N*<=*M*, certainty for each class is equal to 0 for algorithm 4 and 1 for algorithm 5, and is impossible to undertake final classification decision.

Algorithm 4. Certainty modified by short-time history:

$$Cer_k(x_1,x_2,...,x_N) = Cer_k(x_2,...,x_N) + \frac{1}{M}\sum_{i=N-M+1}^{N} c_i(x_n)\cdot\sum_{n=1}^{N} c_k(x_n).$$

Algorithm 5 Weighted certainty modified by short-time history:

$$Cer_k(x_1,x_2,...,x_N) = \left(Cer_k(x_2,...,x_N) + \frac{1}{M}\sum_{i=N-M+1}^{N} c_i(x_n)\cdot\sum_{n=1}^{N} c_k(x_n)\right)\cdot\left(0.5+\frac{1}{N}\sum_{n=1}^{N} c_k(x_n)\right)$$

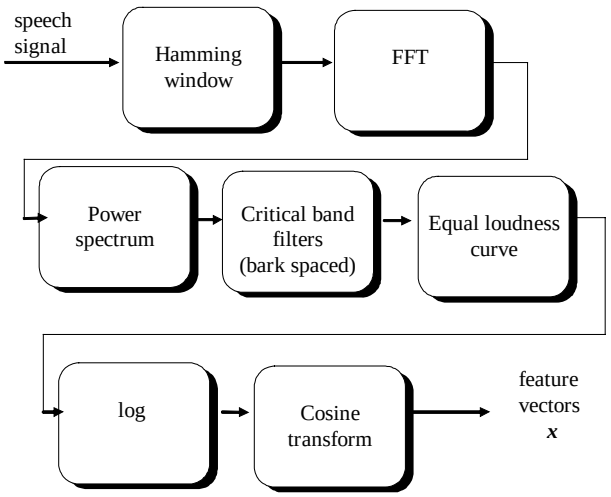

Fig. 4. Auditory based pre-processing

4 SPEAKER RECOGNITION TASK

The presented in previous chapter methods of integration sequences of neural network classification outputs have been applied to a speaker recognition task. We have use special way of pre-selecting input data. Only selected parts of speech signal, we call it speech events, are used for further processing. Speech was digitized at 20 kHz on 16 bits and presented to the system. A short-time energy function is calculated using 40 msec frames with 10 mces steps in an overlapping mode. Next the energy function is smoothed using a 7-point Hanning window. From this smoothed energy function the peaks are located. The peaks with energy value below some threshold are rejected, regarded as noise. Remained peaks are treated as speech events and only 23 msec of signal around such event is further used by pre-processing. In performed experiments approximately 6 speech events occurred in 1 second of speech.

Moreover, experiments showed that around 95% of speech events lay in the middle of vowels. It allows to achieve an independent sequence of feature vectors [3,4]. Next, each of selected speech event was pre-processed to achieve feature vectors which are inputs for neural network classifiers. The pre-processing was developed on the basis of empirical studies of the human ear. In the computational model, the goal is to convert the short-time spectra from a conventional form to a so-called perceptual domain, with the resultant spectra referred to as *auditory spectra*. A number of facts originating from psychological experiments lay on the basis of perceptually processing, including: critical band, masking and equal-loudness. The computational model of this method is shown in a block diagram of Fig. 4. First, speech signal is converted into the power spectrum. Next, the signal is filtered throughout a bank of filters (in the performed experiments a number of 32 filters were used), which are equally spaced in the Bark domain [8]. Next, the filter outputs are multiplied by the equal loudness function approximated from date given in [9] and achieved signal is converted into logarithm spectrum. Finally, the logarithm perceptually spectrum is cosine transformed to produce the cepstral coefficients. In performed experiments a number of 14 cepstral coefficients are used in the feature vector.

5 EXPERIMENT AND RESULTS

The proposed in chapter 3 algorithms have been tested on a closed set, text independent, speaker identification task. The population of speakers consists of 15 persons. Each person produced a set of three 10s utterance of text. Text consisted of freely spoken sentences in Polish language. This set has been used for training each neural network. In case of the multilayer perceptron we have used 17 hidden neurons (selected by experiments) with sigmoid update rule. And in case of PNN, each RBF network consists of 8 neurons in the hidden layer. Next, after three months, the same speakers produced next set, consisting of three 10s utterance of text. This text was uncorrelated with the text used for training and was used for testing. Achieved results, i.e. percent of correct recognition, for each certainty calculation algorithm for multilayer perceptron is presented in Table 1, and for PNN in Table 2. Length of short-time history (M) in algorithms 4 and 5 was set to 3.

Table 1. Percentage of correct classification for multilayer perceptron

Algori-thm	Number of speech events (feature vectors)				
	2	3	4	5	6
1	61.0%	73.1%	80.7%	94.6%	87.9%
2	74.1%	82.4%	87.1%	90.8%	92.3%
3	70.2%	75.4%	76.5%	76.6%	76.4%
4	-	73.2%	80.7%	82.6%	83.7%
5	-	73.2%	80.7%	82.6%	84.2%
	Number of speech events (feature vectors)				
	7	8	9	10	11
1	90.2%	92.8%	93.6%	94.5%	95.7%
2	94.6%	95.7%	96.7%	97.7%	98.1%
3	75.0%	74.3%	73.7%	71.7%	70.1%
4	84.7%	85.0%	85.6%	86.3%	86.8%
5	86.1%	87.2%	88.1%	89.0%	90.2%

Table 2. Percentage of correct classification for PNN

Algori-thm	Number of speech events (feature vectors)				
	2	3	4	5	6
1	62.4%	72.5%	79.6%	84.5%	87.6%
2	66.9%	70.4%	72.2%	73.1%	74.2%
3	75.0%	82.9%	87.3%	90.0%	92.0%
4	-	72.8%	80.0%	82.3%	83.6%

5	-	72.8%	80.0%	82.3%	84.0%
	Number of speech events (feature vectors)				
	7	8	9	10	11
1	89.8%	92.3%	93.7%	94.7%	95.6%
2	74.5%	74.9%	75.6%	76.0%	76.5%
3	94.1%	95.4%	96.2%	97.3%	98.5%
4	84.1%	84.8%	85.4%	86.2%	86.7%
5	85.1%	86.7%	87.9%	88.6%	89.4%

6 CONCLUSION AND FURTHER WORK

The achieved results are quite good. As expected, taking into consideration larger number of speech events (longer text) gives lower recognition error. The system for 11 speech event, approximately 2 s of text, gives for the best of proposed algorithm 98.5% of correct recognition. It's hard to say which algorithm is better in general, tests on different tasks would be very helpful with proposed algorithm analysis. However, the achieved results suggest that summing of multilayer perceptron neuron outputs and multiplication of RBF network outputs in case of PNN classifier are the best methods.

Further work will include the test of presented methods on other data set, i.e. recognition of vehicles based on acoustic signals [10]. In case of presented speaker recognition task, the method of speech event selection, which in most cases were vowels, allows us to assume the probabilistic independence of following vowels. However, in the planned task, i.e. in recognition of vehicles, it is not hold. Therefore, the proposed method requires some investigations to allow recognitions in cases when the assumption of probabilistic independence of following vectors is not true. Also some theoretical work on sequential classifier presented here would be needed to justify the achieved results.

REFERENCES

[1] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[2] Ch. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

[3] T. Walkowiak T, Probabilistic Neural Network for Open Set Classification, *IV National Conference Neural Networks and their Application*, Zakopane, Poland, pp. 232-237, 1999.

[4] T. Walkowiak T, Sequential recognition in neural networks – a speaker recognition case study, *6th International Conference on Soft Computing*, Mendel'2000, Brno, Czech Republic, pp. 385-390, 2000.

[5] T. Walkowiak, RBF neural networks for density estimation, *XIX KKTOiUE*, Krakow-Krynica, Poland, pp.657-662, vol. 2, 1996.

[6] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, vol B 39 , pp. 1-38, 1977.

[7] J. Zagorny, *Analysis of sequential recognition methods* (in Polish), MSc thesis, Institute of Engineering Cybernetics, Wroclaw University of Technology, Wroclaw, Poland, 2001.

[8] R. Schroeder, Recognition of Complex Acoustic Signals, *Life Science Research Report 5*, edited by T. H. Bullock, Abakon Verlag, Berlin, pp. 307-352, 1977.

[9] D. Robinson, R. Dadson, A predetermination of the equal-loudness relations for pure tones, *British Journal of Applied Physics*, vol. 7, pp. 166-181, 1956.

[10] T. Walkowiak, Neural networks for vehicle recognition based on acoustic signals. *First International Conference on Soft Computing Applied in Computer and Economic Environments ICSC'2003*, Kunovice, Czech Republic, pp. 129-139, 2003.