World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:13, No:7, 2019

Optimizing the Probabilistic Neural Network Training Algorithm for Multi-Class Identification

Abdelhadi Lotfi, Abdelkader Benyettou

Open Science Index, Information and Communication Engineering Vol:13, No:7, 2019 publications.waset.org/10010615.pdf

***Abstract*—**In this work, a training algorithm for probabilistic neural networks (PNN) is presented. The algorithm addresses one of the major drawbacks of PNN, which is the size of the hidden layer in the network. By using a cross-validation training algorithm, the number of hidden neurons is shrunk to a smaller number consisting of the most representative samples of the training set. This is done without affecting the overall architecture of the network. Performance of the network is compared against performance of standard PNN for different databases from the UCI database repository. Results show an important gain in network size and performance.

***Keywords*—**Classification, probabilistic neural networks, network optimization, pattern recognition.

I. Introduction

IN the world of pattern recognition, classification is known to be one of the major factors which can affect directly and dramatically the performance of any given system. PNN has been introduced by Specht in [1]. Because of their good classification properties, they soon became a reference in classification as neural networks. PNNs offer many advantages: they do not suffer from the problem of local minimums [2] as MLPs do, the training is very fast since the network is created after one pass through the training set [1], [3], they can be used interactively and the principle itself has a good mathematical basis [4] (function approximation, interpolation and probability density functions estimation). On the other hand, there are some important drawbacks: the number of hidden neurons is equal to the number of training samples. This can be very restrictive for certain problems with a big training set which includes most of time many redundancies (usually the training sets are created from the same physical observation and that means the same sample is duplicated with slight differences). Another problem is the choice of the smoothing parameter which can reflect seriously the generalization of the network [5].

Actually, PNN are considered good classifiers and can be used with little knowledge of artificial intelligence techniques [6]. This is why they were used recently for so many classification problems. Examples of areas where PNNs were used successfully are email security enhancement [7], intrusion detection within computer networks [8], water quality assessment [9], hepatitis disease diagnosis [10], detection of resistivity for antibiotics [11], speech recognition [12], biometrics applications [13]-[16] and military applications [5], [17], [18].

To solve the problem of generalization and parameters choice in PNN, a lot of work has been carried out by many researchers [19]. Kim [18] proposed a new architecture to enhance generalization of standard PNNs. In [20], a learning vector quantization (LVQ) algorithm was used to train a PNN in order to make the network's size smaller. Other examples of PNN variants are Fuzzy PNNs [19] and Particle Swarm Optimization stochastic algorithm for PNN parameters choice [21]. Most of the time, solving the initial problem may cause a change in the architecture of the PNN and makes it loose its advantages. RKPNN (Rotated Kernel PNNs, see [22]) is an example of such a change. These networks have good generalization qualities which are claimed to be compared to support vector machines (SVMs) (with a reduced number of classes [23]) but the training is extremely slow [25]. Applying this type of networks interactively seems impossible.

The training algorithm presented in this paper reduces significantly the number of hidden neurons without affecting the architecture of the network which makes adding new samples or classes possible at any time without re-training. The testing is, as a result, faster since it depends on the number of neurons in the network after training. This training algorithm can be used for a wide range of problems such as biometrics, character recognition and speech recognition.

In the following sections, the training algorithm is presented and tested on some benchmarking databases from the UCI repository. Results from classical PNN training algorithm are presented for comparison purposes.

II. Standard PNNs

A. Background

Unlike multi-layer perceptron (MLP) networks, radial basis function networks (including PNN) use radial basis functions instead of sigmoid activation functions to make a local decision function centered on a subset of the input space [8]. The global decision function is the sum of all local functions [6] [23]. This way, the problem of local minimums is solved.

A supervised pattern classification system operates as follows: place the observed vector x in one of the predefined category classes C_m; (m number classes). The accuracy of the cluster classifier is limited by the dimension of the input space and the number of classes. This problem is formulated by the Bayes classifier as [24]:

Abdelhadi Lotfi is with the National Institute of Telecommunication and ICT (INTTIC), Oran, 31000 Algeria (corresponding author, phone: +213541253300; e-mail: alotfi@ito.dz).

Abdelkader Benyettou is with The University of Sciences and Technology (USTO-MB), Oran, 31000 Algeria (e-mail: a_benyettou@yahoo.fr).

Open Science Index, Information and Communication Engineering Vol:13, No:7, 2019 publications.waset.org/10010615.pdf

$$P(C_i \mid x) = \frac{P(x \mid C_i)P(C_i)}{\sum_{j=1}^{m} P(x \mid C_j)P(C_j)} \quad (1)$$

where $P(x|C_i)$ is the conditional probability density function of x given the set C_i and P(C_j) is the probability of drawing data from the class C_j.

An input vector x is classified as belonging to class Ci if:

$$P(C_i|x) > P(C_j|x);\ \forall\, j = 1, 2, \ldots, m; j \neq i \quad (2)$$

The difficulty with this relationship is that the prior probabilities P(x | C_i) (probability that the sample x comes from a population C_i) are usually unknown. A solution to this problem is the estimation of these probabilities from the training dataset. This can be done using the Parzen windowing technique for pdf estimation [6]. Parzen showed that a class of pdf estimators asymptotically approaches the underlying parent density provided that it is smooth and continuous [1]. The estimator used for PNN is:

$$f_A(x) = \frac{1}{2\pi^{p/2}\sigma^p}\frac{1}{m}\sum_{i=1}^{m}\exp\left[-\frac{(X - X_{ai})^t(X - X_{ai})}{2\sigma^2}\right] \quad (3)$$

with X_{ai} is the pattern i from class A; σ is a parameter used for smoothing [24]. As a result, the global decision function is the sum of small Gaussian functions centered on the training samples. In the same context, PNN use all the training data set to estimate probability density functions. The densities are used then to estimate the likelihood function [1].

B. Network architecture

Fig. 1 represents the architecture of a standard PNN as stated by Specht [1]. This is a 3-layer neural network: The number of neurons in the **input layer** is the number of features needed to describe the observation.

The **pattern (hidden) layer** organizes the learning set by representing each input vector by a hidden neuron which records the parameters of this vector. The activation function used in this layer is the exponential function:

$$f(x) = e^{-(w_i - x_i)^t(w_i - x_i)/2\sigma^2} \quad (4)$$

where: x_i represent the variables of the model; w_i represent the weights; σ represents a smoothing parameter chosen according to each case [24].

The computation of the output of hidden units is realized by:

$$\Phi_j^i(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} e^{-\frac{(x_j^i - x)^t(x_j^i - x)}{2\sigma^2}} \quad (5)$$

where x_j^i is the j^{th} training pattern vector from patterns in class C_i and d is the dimension of the input space.

The number of neurons in the **output layer is** equal to the number of classes. All neurons from the hidden layer belonging to the same class are connected to the same output neuron. Neurons in the output layer calculate their activations which represent the probability of x being drawn from class i [25]. So:

$$P(C_i \mid x) = \frac{1}{N_i(2\pi)^{d/2}\sigma^d}\sum_{j=1}^{N_i}\exp\left(-\frac{(x_j^i - x)^t(x_j^i - x)}{2\sigma^2}\right) \quad (6)$$

Note that N_i is the number of training patterns from class C_i.

A normalization layer may be needed if the inputs are not already normalized [3].

The classification decision for the input vector X is given in accordance with the Bayes decision rule using:

$$\hat{C}(x) = \arg\max_{i=1,2,\ldots}\left\{\frac{1}{N_i(2\pi)^{d/2}\sigma^d}\sum_{j=1}^{N_i}\exp\left(-\frac{(x_j^i - x)^t(x_j^i - x)}{2\sigma^2}\right)\right\} \quad (7)$$

where m is the number of classes presented in the training set [1].

The training consists of creating a hidden neuron for each training vector with a Gaussian activation function centered on this vector.

Each neuron in the output layer is connected to all neurons from the hidden layer representing this class [15].

The Gaussian activation function used by the hidden neurons was shown in (4) [23].

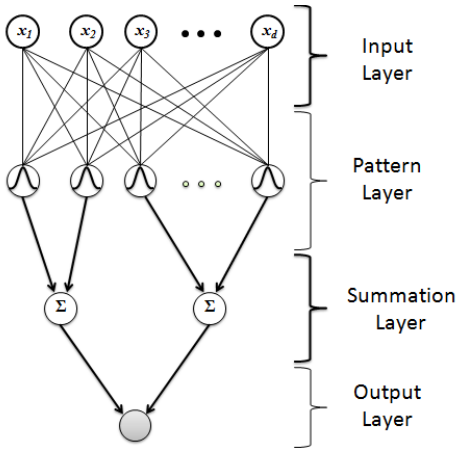

Fig. 1 Mapping nonlinear data to a higher dimensional feature space

C. Training Algorithm

It consists of looping over all training vectors and creating Gaussian functions centered on each one of these vectors. After that, a summation neuron is added for each class and connected with the hidden neurons from the same class [24]. The training algorithm is as follows:

Begin

Open Science Index, Information and Communication Engineering Vol:13, No:7, 2019 publications.waset.org/10010615.pdf

Initialization
j = 0;
n = number of samples;
Do
j ← j+1 ;
Normalization: $x_{jk} \leftarrow \frac{x_{jk}}{\sqrt{\sum_{i}^{d} x_{ji}^{2}}}$;
Learning process: $w_{jk} \leftarrow x_{jk}$;
If $x \in w_i$ then $a_{ic} \leftarrow 1$;
Until *j* = *n*;
End

D. Testing Algorithm

Once the training is completed, the normalized samples are classified using the formulae:

$$z_k = w_k^t x$$

The final output of the hidden neuron is given by:

$$\varphi(z_k) = e^{\frac{z_k - 1}{\sigma^2}}$$

The only user-defined parameter is the smoothing parameter σ [15].

The summation neuron calculated its activation by a normalized sum of all hidden neurons connected to it [15]. The testing algorithm is given by:

Begin
Initialization
k = 0;
x = test sample;
Do
k ← k+1 ;
$z_k \leftarrow w_k^t x$
If $a_{kc} = 1$ then $g_c \leftarrow g_c + e^{\frac{z_k - 1}{\sigma^2}}$;
Until *k* = *n*;
Return $class \leftarrow \arg\max_i (g_i(x))$;
End

III. Modified Training Algorithm

The main purpose of this work is to reduce significantly the number of hidden neurons without changing the overall architecture which will make it as effective as the standard PNN. To achieve this purpose, the network is trained with a standard training algorithm. The unnecessary neurons are removed from the hidden layer by removing a neuron at a time from the hidden layer and measuring its contribution in the decision of the new network. If the contribution is not significant, the neuron is removed definitively from the network, otherwise, it is retuned back.

After all hidden neurons are treated; the hidden layer is reduced to a small number of neurons. This step mostly removes noisy and redundant data.

Finally, the process is reversed to add neurons to the hidden layer if the classification is not correct. At this stage, classification of the training data is a 100% accurate.

Another advantage of this new network (PNN*) is that all boosting algorithms for standard PNNs are applicable without major modifications.

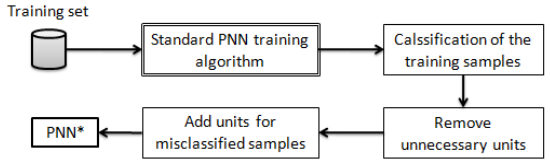

Fig. 1 The training algorithm

The new training algorithm consists of two steps:

A. **Build a standard PNN:** This is achieved by implementing the standard training algorithm. At this stage, the number of hidden neurons is equal to the number of training vectors.
B. **B. Shrink the PNN's hidden layer:** The size of the network is gradually reduced by applying the reduction phase of the modified training algorithm (see Fig. 2). This algorithm is given by:

Begin
i ← 0;
x ← first training sample;
While (i<n) Do
i ← i+1;
Delete i^{th} node from the hidden layer
Classify the i^{th} from the network
If the classification confidence is acceptable then
Consider this network
Else ignore the last modification;
End While;
Return
the new network;
End [23]
3rd step: network re-checking
Begin
i ← 0;
x ← first training sample;
While (i<n) do
i ← i+1 ;
If x is not correctly classified then
Add x to the hidden layer;
End While;
Return
the new network;
End [23]

Note that the testing algorithm is the same as a standard PNN testing algorithm.

IV. Simulation Results

A. Test of the Algorithm for a Bi-Dimensional Dataset

In this experiment three classes are represented by 33 data points. Two networks are created to separate the plan

(100*100 points) using the 33 labeled samples. Each class is represented by a color (black, grey and white).

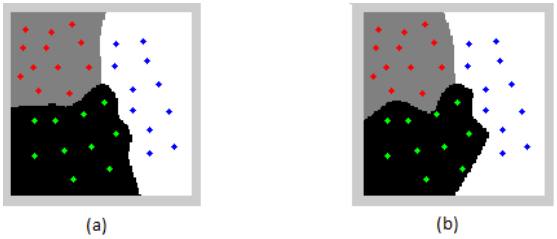

Fig. 2 Classification of (100*100) data points with a standard PNN algorithm (a) and a modified algorithm (b)

In Fig. 3 (a) the number of hidden neurons is equal to the number of training samples. That is 33 hidden units. In Fig. 3 (b) the number of hidden neurons used for classification is 10 neurons. We can see that the classification is reduced to about 30% of the size of a standard PNN. This is because adjacent data points are simply replaced by one neuron in the network. The time for testing is proportional to the number of hidden neurons. Thus, the network in Fig. 3 (b) is faster. In this experiment, the dimension of the input space is small. To validate our results, databases with higher input dimensions are needed with more classes.

B. Databases Description

To illustrate the effectiveness of the proposed algorithm, we used different data sets from the UCI database repository. The data sets are chosen with different properties (number of classes, number of inputs …) to test the new algorithm for a wide range of situations. The databases used are:

- **Letter recognition database (Letter):** The database consists of 26 capital back-and-white English alphabet letters. There are 20,000 instances in the data set with an input dimension of 16 features representing 26 letters for 20 different fonts. All data are numeric (in the integer range 0-15) with no missing data [26].
- **Glass identification database (Glass):** Classification of types of glass motivated by criminological investigation. The database contains 214 samples distributed on 7 different classes of glass types [25].
- **Balance scale database (Balance):** The database contains 625 instances of scale measures based on a physiological experiment. The input vector has 4 different features. The database was taken from the UCI machine learning repository [25].
- **Breast cancer database (Breast):** This data set includes 201 instances of one class and 85 instances of another class. The instances are described by 9 attributes, some of which are linear and some are nominal [25].
- **Iris database (Iris):** This database represents data used for iris plant classification. There are 50 instances belonging to 3 different classes of iris plants [25].

Table I gives more details about the size and input dimensions of all 5 databases.

TABLE I
PROPERTIES OF THE DATA SETS USED IN EXPERIMENT

Data Set	Input dimension	Number of samples
Letter	16	20000
Glass	9	214
Balance	4	625
Breast	9	286
Iris	4	150

C. Results

Both training algorithms (standard and modified) are applied on the datasets separately. We focused on the performance of the network (classification rate) and the number of neurons created in the hidden layer. About 2/3 of samples are used for training and the remaining (1/3) samples are used for test. The samples in the testing set are new (unseen before) to the network and they were chosen randomly from the entire database to make the situation as realistic as possible. It is not necessary to present the classification rate for the training set since both networks can classify all samples successfully.

TABLE II
CLASSIFICATION RATES FOR ALL DATA SETS

Data Set	# samples	Standard algorithm		New algorithm	
		% correct	% error	% correct	% error
Letter	20000	95.96	4.04	93.06	6.94
Glass	214	76.59	23.40	78.72	21.27
Balance	625	94.87	5.12	91.66	8.33
Breast	286	96.18	3.81	95.03	4.96
Iris	150	96.42	3.57	100.00	0.00

TABLE III
PERFORMANCE IN TERMS NETWORK SIZE (NUMBER OF HIDDEN UNITS) AND TIME NECESSARY FOR CLASSIFICATION

Data Set	# training samples	Standard algorithm		New algorithm	
		# hidden units	t (s)	# hidden units	t (s)
Letter	15000	15000	9472.75	2433	1443.74
Glass	167	167	2.7767	91	1.5131
Balance	469	469	24.8078	171	8.9744
Breast	437	437	42.1040	50	4.8678
Iris	85	85	0.6776	36	0.2886

D. Discussion

Results in Table II show that for all databases, performances of the standard and new algorithm are very close. For instance, performance for Letter, Balance and Breast cancer databases is better with the standard PNN. On the other hand, performance for Glass and Iris databases is better with the proposed algorithm. Regardless to the number of samples or the size of the input space, both networks have quite the same behavior and results are almost similar.

Table III gives details about the size of each network (number of hidden units) and the time necessary for classification of all test samples (t(s)). We can observe that networks created with the proposed algorithm are all smaller than those created with a standard PNN training algorithm. For letter recognition the proposed network is more than 6 times smaller in size with almost the same performance. For

Open Science Index, Information and Communication Engineering Vol:13, No:7, 2019 publications.waset.org/10010615.pdf

Open Science Index, Information and Communication Engineering Vol:13, No:7, 2019 publications.waset.org/10010615.pdf

the glass identification database (which is a database with a lot of perturbations), the size is reduced to 54% with better classification rate for the proposed algorithm. The reduction for balance and iris data sets is about 36% and 42% respectively with a better performance in the case of iris database. In the case of breast cancer database, there is a great reduction in the number of hidden units only 11% of the hidden units were created. Table III gives also information about the time necessary to classify all test examples for both algorithms. The gain in execution time is very clear for all experiments. The proposed algorithm creates fewer hidden neurons which imply a smaller time to calculate the output of the hidden layer. The gain in processing time is proportional to the number of hidden neurons for all cases presented. For example: time necessary with the proposed algorithm is 11% of the time elapsed when using a standard algorithm for the breast cancer dataset. This is the exact proportion in the number of hidden units.

V. Conclusion

The algorithm presented here apparently gives similar classification rates as a standard PNN training algorithm for almost all cases. However, the number of hidden neurons in the second layer is very small and depends on the nature of the database. Databases with many redundancies need fewer hidden units to represent all the training samples. The reduction of the number of neurons implies a gain in the processing time which is proportional to it.

The algorithm presents a solution to the problem of PNN size and keeps the same architecture and advantages of a standard PNN. To add new classes to the new PNN, the same process can be executed for the new training samples only. All PNN boosting algorithms are applicable to this network without major modifications on its structure.

References

[1] D. F. Specht, « Probabilistic neural networks », Neural networks, vol. 3, no. 1, p. 109–118, 1990.

[2] Y.-qun Deng and P.-ming Wang, « Predicting the shrinkage of thermal insulation mortar by probabilistic neural networks », Journal of Zhejiang University SCIENCE A, vol. 11, no. 3, p. 212-222, 2010.

[3] M. Bazarghan and R. Gupta, « Automated classification of sloan digital sky survey (SDSS) stellar spectra using artificial neural networks », Astrophysics and Space Science, vol. 315, no. 1-4, p. 201-210, 2008.

[4] C. M. Bishop, Neural networks for pattern recognition. Oxford University Press, 1995.

[5] Abdelhadi Lotfi and Abdelkader Benyettou. “Cross validation probabilistic neural network based face identification”. Journal of Information Processing Systems. Page: 1075~1086, Vol. 14, No.5, 2018. DOI: 10.3745/JIPS.04.0085.

[6] R. O. Duda, Pattern Classification 2nd Edition with Computer Manual 2nd Edition Set. John Wiley & Sons Inc, 2004.

[7] T. P. Tran, T. T. S. Nguyen, P. Tsai, and X. Kong, « BSPNN: boosted subspace probabilistic neural network for email security », Artificial Intelligence Review, 2011.

[8] T. P. Tran, L. Cao, D. Tran, and C. D. Nguyen, « Novel Intrusion Detection using Probabilistic Neural Network and Adaptive Boosting », 0911.0485, nov. 2009.

[9] M. R. Nikoo, R. Kerachian, S. Malakpour-Estalaki, S. N. Bashi-Azghadi, and M. M. Azimi-Ghadikolaee, « A probabilistic water quality index for river water quality assessment: a case study », Environmental Monitoring and Assessment, 2010.

[10] M. S. Bascil and H. Oztekin, « A Study on Hepatitis Disease Diagnosis Using Probabilistic Neural Network », Journal of Medical Systems, 2010.

[11] F. Budak and E. D. Übeyli, « Detection of Resistivity for Antibiotics by Probabilistic Neural Networks », Journal of Medical Systems, vol. 35, no. 1, p. 87-91, 2009.

[12] N. Neggaz and A. Benyettou, « Hybrid models based on biological approaches for speech recognition », Artificial Intelligence Review, vol. 32, no. 1-4, p. 45-57, 2009.

[13] M.-I. Faraj and J. Bigun, « Synergy of Lip-Motion and Acoustic Features in Biometric Speech and Speaker Recognition », IEEE Transactions on Computers, vol. 56, no. 9, p. 1169-1175, 2007.

[14] S. Meshoul and M. Batouche, « A novel approach for Online signature verification using fisher based probabilistic neural network », in Computers and Communications, IEEE Symposium on, Los Alamitos, CA, USA, 2010, vol. 0, p. 314-319.

[15] Abdelhadi Lotfi, Abdelkader Benyettou, "Over-fitting Avoidance in Probabilistic Neural Networks", World Congress of Information Technology and Computer Applications, 11-13 June 2015 Hammamet Tunisia.

[16] N. Neggaz, M. Besnassi, and A. Benyettou, « Application of Improved AAM and Probabilistic Neural network to Facial Expression Recognition », Journal of Applied Sciences, vol. 10, no. 15, p. 1572-1579, 2010.

[17] L. F. Araghi, H. Khaloozade, and M. R. Arvan, « Ship Identification Using Probabilistic Neural Networks (PNN) ».

[18] M. W. Kim et M. Arozullah, « Generalized probabilistic neural network based classifiers », in Neural Networks, 1992. IJCNN, International Joint Conference on, 2002, vol. 3, p. 648–653.

[19] V. Georgiou, P. Alevizos, and M. Vrahatis, « Fuzzy Evolutionary Probabilistic Neural Networks », Artificial Neural Networks in Pattern Recognition, p. 113–124, 2008.

[20] P. Burrascano, « Learning vector quantization for the probabilistic neural network », IEEE Transactions on Neural Networks / a Publication of the IEEE Neural Networks Council, vol. 2, no. 4, p. 458-461, 1991.

[21] I. De Falco, A. Della Cioppa, and E. Tarantino, « Facing classification problems with Particle Swarm Optimization », Applied Soft Computing, vol. 7, p. 652–658, juin. 2007.

[22] I. Galleske and J. Castellanos, « A rotated kernel probabilistic neural network (RKPNN) for multi-class classification », in Proceedings of the Artificial and natural neural networks 7th international conference on Computational methods in neural modeling - Volume 1, Berlin, Heidelberg, 2003, p. 152–157.

[23] Abdelhadi Lotfi and Abdelkader Benyettou. “A Reduced Probabilistic Neural Network for Classification of Large Databases”. Turkish Journal of Electrical Engeneering and Computer Science, 2014.

[24] Abdelhadi Lotfi and Abdelkader Benyettou. “Using Probabilistic Neural Networks for Handwritten Digit Recognition”. Journal of Artificial Intelligence, 4: 288-294. 2011.

[25] Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository (http://archive.ics.uci.edu/ml). Irvine, CA: University of California, School of Information and Computer Science.

[26] P. W. Frey and D. J. Slate. "Letter Recognition Using Holland-style Adaptive Classifiers". (Machine Learning Vol 6 #2 March 91).