J Inf Process Syst, Vol.14, No.5, pp.1075~1086, October 2018
https://doi.org/10.3745/JIPS.04.0085

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Cross-Validation Probabilistic Neural Network Based Face Identification

Abdelhadi Lotfi* and Abdelkader Benyettou**

Abstract

In this paper a cross-validation algorithm for training probabilistic neural networks (PNNs) is presented in order to be applied to automatic face identification. Actually, standard PNNs perform pretty well for small and medium sized databases but they suffer from serious problems when it comes to using them with large databases like those encountered in biometrics applications. To address this issue, we proposed in this work a new training algorithm for PNNs to reduce the hidden layer's size and avoid over-fitting at the same time. The proposed training algorithm generates networks with a smaller hidden layer which contains only representative examples in the training data set. Moreover, adding new classes or samples after training does not require retraining, which is one of the main characteristics of this solution. Results presented in this work show a great improvement both in the processing speed and generalization of the proposed classifier. This improvement is mainly caused by reducing significantly the size of the hidden layer.

Keywords

Biometrics, Classification, Cross-Validation, Face Identification, Optimization, Probabilistic Neural Networks

1. Introduction

For a pattern recognition system in general and for biometric identification systems in particular, in addition to the employed strategy for feature extraction, the choice of the classification method to be used in the decision phase is a determining factor in the effectiveness of the proposed solution. To address this problem, the user must select a decision strategy that meets his or her needs. These methods include: probabilistic methods, Bayesian methods and neural methods. Each method has advantages and disadvantages. Therefore, choosing the appropriate strategy is not always obvious.

Progress in the field of pattern recognition recently resulted in problems with large training databases. Databases containing thousands and sometimes millions of samples have become a common reality for applications such as biometrics, information retrieval, semantic web, etc. For such applications, the choice of the most appropriate method is essential for a robust solution. Therefore, the scientific community has recently reconsidered many old techniques to make them meet these new requirements. Unfortunately, many methods have been neglected from this refinement and need to be

Manuscript received August 25, 2017; first revision January 25, 2018; accepted March 15, 2018.
Corresponding Author: Abdelhadi Lotfi (alotfi@ito.dz)
* National Institute of Telecommunication and Information and Communication Technology, Oran, Algerie (alotfi@ito.dz)
** Dept. of Computing, Faculty of Mathematics and Computing, University of Sciences and Technology of Oran - Mohamed Boudiaf, Oran, Algerie (a_benyettou@yahoo.fr)

re-explored more in depth.

Since they were created by Specht [1] probabilistic neural networks (PNNs) have been used extensively to solve many problems related to classification, mapping and associative memory. The reader can find so many papers in the literature referring to PNNs, especially for classification applications. Actually, they have been used in many fields, namely: automatic emails security enhancement [2], intrusion detection in computer networks [3], water quality assessment [4], illness diagnosis [4], antibiotics resistance detection [4], biometrics applications [5-7], handwriting recognition [8], military applications (target detection) [9,10], etc.

The main reason behind their popularity is the fact that unlike multi-layer neural networks (MLP), these classifiers do not suffer from the problem of local minimums frequently encountered with other neural networks [11]. In fact, in PNNs the global decision function is a sum of many local decision functions centered on the training samples, which makes the situation of local minimums very unlikely to happen [12]. Moreover, these networks have a stronger mathematical background than most other neural networks [12].

Unfortunately, PNNs cannot be used effectively to classify databases with a big number of samples in the training set because the number of hidden neurons is equal to the number of training samples. If a situation like this appears, another technique is used to reduce the number of training samples for a later use by PNNs. This solution does not guarantee that these samples are good for the PNN since it does not consider the PNN parameters.

As a result, PNNs are generally used for small benchmarking databases. When it comes to bigger real world databases, other types of classifiers like MLPs are preferred [13]. Despite this major drawback, the PNN have several important good features such as the probabilistic nature of its outputs, the possibility of learning by small chunks without retraining, parallelism in the hidden layer (for separate classes), robustness against the problem of local minima, and multi-dimensional interpolation, etc.

A lot of work has been done so far to make PNNs more efficient for certain problems by changing either the architecture of the network or the training algorithm. Most of time, these changes affect seriously the network making it lose some of its major advantages and adds more complexity to its implementation. As a response to this concern, a learning vector quantization (LVQ) training algorithm for PNNs was proposed in [14]. By using a Bayesian model for estimating parameters and a particle swarm optimization to determine the weight of the outputs of a PNN network, a solution has been proposed by De Falco et al. [15]. The solution requires a very expensive computational complexity to estimate the network's parameters from the training samples. In [16], authors proposed a PNN that uses fuzzy logic to measure the significance of each neuron in the hidden layer by giving it an output weight towards the summation layer. Other propositions include Gram-Charlier PNNs, generalized PNNs [17], and rotated kernel PNNs [18].

All of these solutions were proposed to solve a particular problem and they modified the architecture of the network which makes it lose its major advantages. In this paper, we propose a new training algorithm for PNNs that does not change its architecture at all. It uses the principle of cross validation to reduce the number of hidden neurons of the PNN and hence speed up the response of the network in the testing step. Of course, our training algorithm is slower (in training) than the standard algorithm. In fact, cross-validation PNN (CVPNN) has an algorithmic complexity of $O(n^2)$ while standard PNN's complexity is $O(n)$; n being the number of training examples.

The proposed solution was tested for a variety of benchmarking databases. They were also tested for some biometric applications. Performances of the CVPNN and the standard PNN were compared in terms of the size of the hidden layer, the processing speed and the generalization of the network to show the difference between the new and old training algorithms.

2. Cross-Validation PNNs

2.1 Basics

In a classification context, let's suppose we have a training data set of examples belonging to m different classes and we want to classify a new vector x of dimension d.

According to the Bayes conditional property rule, the probability of samples x to belong to class C_i is given by:

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

The problem here is that the prior probabilities $P(x|C_i)$ are unknown (since the probability densities of the patterns in the categories to be separated are unknown) [1]. The Parzen windowing technique gives a solution for the estimation of probability density functions from the training examples. The formula used in this paper for the estimation of probability density functions (pdfs) is given by:

$$g_i(x) = \frac{1}{2m\pi^{d/2}\sigma^d} \sum_{j=1}^{m} \exp\left(-\frac{\left\| x - x_{ij} \right\|^2}{2\sigma^2} \right) \tag{2}$$

where $g_i(x)$ is the global decision function for class i; X_{ij} is sample j belonging to class i; and σ is a smoothing parameter.

Example x is classified as belonging to class i if:

$$\arg\max_{j=1,2,\ldots} P\left(C_j \mid x\right) = i \tag{3}$$

When we apply this principle to neural networks, the result will be a multi-layer neural network with four layers (Fig. 1).

The *input layer* has as much neurons as there are features in the input space (d dimensional space). All neurons are connected to all neurons in the hidden layer.

The *pattern (hidden) layer* has neurons representing each example in the training set. All neurons belonging to a certain class are connected to the neuron in the summation layer which represents this class. The hidden layer computes its outputs using an exponential activation function.

The formula of activation in the hidden layer is given by:

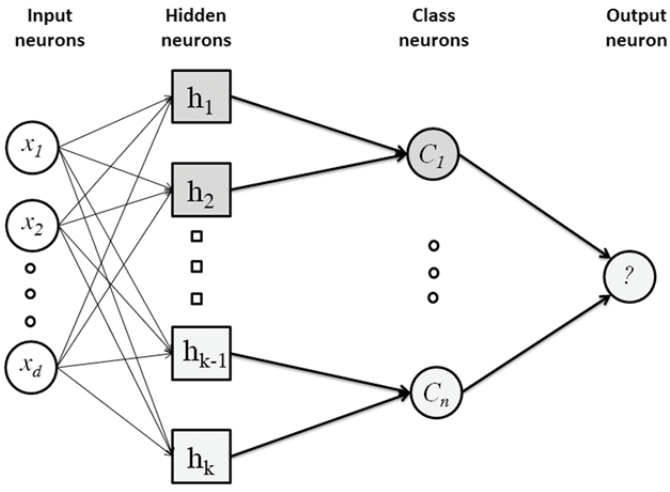

Fig. 1. Main architecture of a CVPNN.

$$\Phi_j^i(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} e^{-\frac{\left(x_j^i - x\right)^t \left(x_j^i - x\right)}{2\sigma^2}} \tag{4}$$

where x_j^i is the j^{th} example from the i^{th} class.

The *summation (class) layer* contains m neurons; each one representing a separate class. The activation of neuron i (class i) is calculated here using:

$$P(C_i \mid x) = \frac{1}{N_i (2\pi)^{d/2}\sigma^d} \sum_{j=1}^{N_i} \exp\left(-\frac{\left(x_j^i - x\right)^t \left(x_j^i - x\right)}{2\sigma^2} \right) \tag{5}$$

With N_i the total number of examples in class i.

The *output layer* calculates the wining class using the formula:

$$\hat{C}(x) = \arg\max_{j=1,2,\ldots} P\left(C_j \mid x\right) \tag{6}$$

Or more formally:

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

2.2 Training Algorithm

The training algorithm presented here is inspired from a former work where we presented a network called reduced PNN (RPNN) for classification of large databases [12]. The CVPNN training algorithm

is much faster and generates fewer hidden neurons. The former algorithm starts its training using a standard PNN and reduces the number of hidden units iteratively which takes much time to delete the unnecessary neurons after measuring their contribution in the final decision of the network (because the size of the original PNN's hidden layer is equal to the number of training vectors).

The idea here is to start instead with an empty PNN (no neurons in the hidden layer) and use a cross-validation algorithm while constructing the hidden layer in order to select only the representative examples from the training set. This is done by presenting the training examples one at a time. Each training vector presented to the network is tested before creating the corresponding hidden neuron. If this new neuron does not add a significant value to the generalization of the network, it is automatically ignored. It is a test-and-train process that is repeated so many times and for all the training data set (Fig. 2).

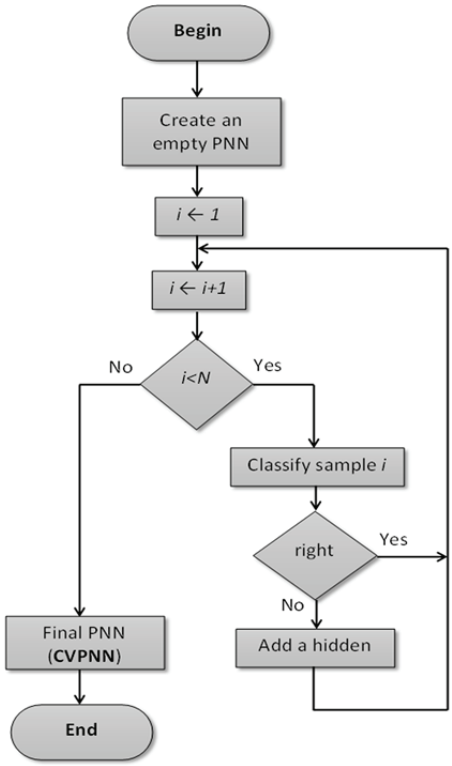

Fig. 2. CVPNN training algorithm.

For initialization, we start here with an empty PNN and neurons are added to the hidden layer when necessary. Adding a new neuron to the hidden layer depends on the accuracy of classifying the corresponding example by the current network. Three different cases are possible:

- If the network is unable to classify the current example accurately, this example is added to the hidden layer.
- If the classification is accurate with low confidence, the neuron's positions are changed according to Fig. 3.
- In case we have good classification with high confidence, the current PNN is kept unchanged.

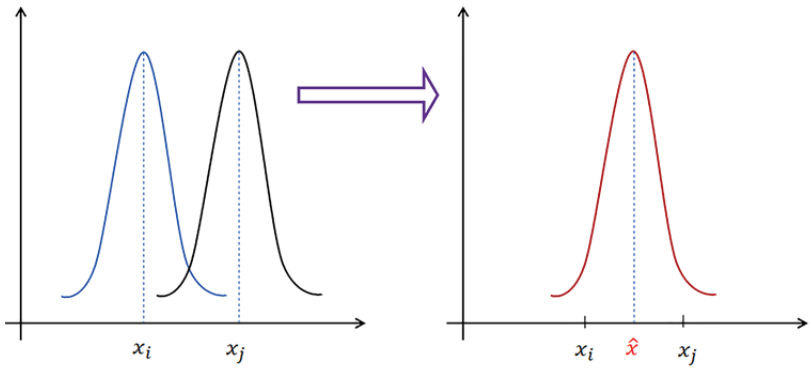

Fig. 3. Changing the neuron's position in a one-dimensional space.

The major drawback of this new training algorithm is the training time which can be very long in comparison with a standard training algorithm. However, the training can be done by classes of samples at separate points of time without a retraining each time. This means that new examples and/or classes can be added as needed in different points in time. Actually, this is one of the major advantages of this solution.

The result of this training is a classifier which has a smaller hidden layer (with less neurons); is more robust and resistant to over-fitting, and has better generalization capacities (better classification rates).

The reduced number of neurons in the hidden layer makes the testing (time of response) of the network very fast according to the reduction ratio. A small number of hidden neurons needs a reduced time to calculate the activation in the hidden and summation layers and produces a quick response.

The main idea of the training algorithm is further explained in the following algorithm:

Input: Training samples
Output: Neural network
Begin
  Create an empty CVPNN;
  $i \leftarrow 1$;
  $x \leftarrow$ *first training sample* ;
  Repeat
    $i \leftarrow i+1$;
    If classification (i^{th} sample) ≠ target **then**
      Add a new neuron in the hidden layer;
    End if
  Until $i=n$;
  Return CVPNN;
End

2.3 Testing Algorithm

The testing algorithm is similar to the one usually used in standard PNNs. The same algorithm is also used in the training step and is given hereafter:

Input: A test sample
Output: Its class
Begin
i ← *0*;
x ← Test vector (sample);
Repeat
i ← *i*+*1*;
$z_i \leftarrow w_i^t x$;
If a_{ic}=*1* **then**
$g_c \leftarrow g_c + e^{\frac{z_i - 1}{\sigma^2}}$;
End
Until i=n;
Ruturn $class \leftarrow \arg\max_i(g_i(x))$;
End

In the following section, the CVPNN will be used for classification of simple data points in two-dimensional space to show its performance compared to a standard PNN. After that, it will be used for face identification using some well-known databases.

3. Experiments

3.1 Test on a 2-Dimentional Space

In order to show the difference between the CVPNN and the standard PNN, we present the classification of 2-dimentional data points into three different classes. Each class is represented in the graph with a different color. There are 137 training examples from all classes (Fig. 4).

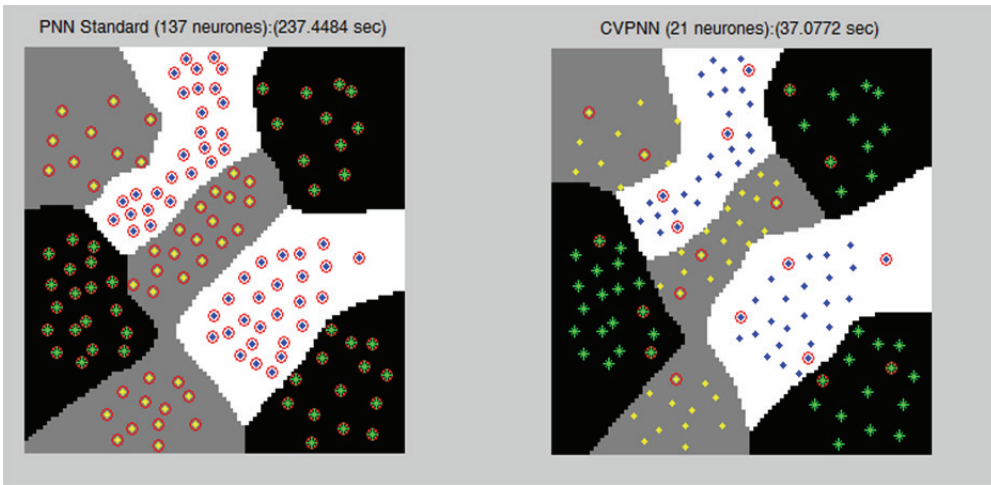

Fig. 4. Classification of 100*100 data points using standard PNN (left) and CVPNN (right). Neurons from the hidden layer are represented by circles.

3.2 The ORL Faces Database

The face database Oliveti Research Laboratory (ORL) was collected by a laboratory of AT & T, based in Cambridge. The database contains images of 40 subjects; each registered under 10 different views (Fig. 5). For some subjects, the images were collected at different times, with variations in lighting conditions, facial expressions (neutral expression, smile and eyes closed) and facial details (with/without glasses). All images were collected on a dark background. The size of each image is 112×92 pixels gray-scale. The poses of the head have some variation in depth with respect to the frontal pose. However, these variations relate only to certain individuals and are not systematic. The files are in PGM format.

Fig. 5. Samples from the ORL faces database.

3.2.1 Results

For features extraction (and dimensionality reduction), two algorithms were used namely Principal Component Analysis (PCA) and Kernel Principal Component Analysis (KPCA) [19]. After feature extraction, two training algorithms (PNN, CVPNN) were used to construct PNNs.

The ORL database samples were divided into:

- A training set containing 280 samples chosen randomly from the overall database.
- A test set which contains 120 totally different from those used for training.

Results of these experiments are presented in Table 1. A comparison between the two networks is shown in Fig. 6.

Table 1. Results for the ORL faces database

	PNN			CVPNN		
	Classification rate (%)	**Time (s)**	**Hidden neurons**	**Classification rate (%)**	**Time (s)**	**Hidden neurons**
PCA	95.83	6.188	280	95.83	2.792	130
KPCA	96.67	6.005	280	96.67	2.323	110

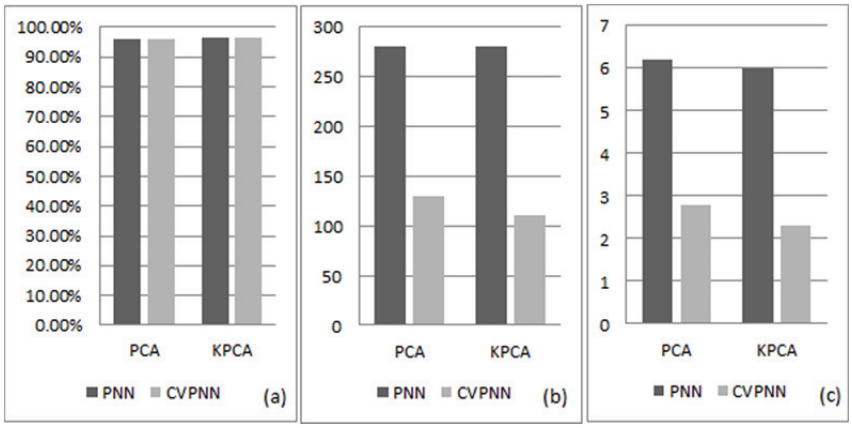

Fig. 6. Comparison between networks in terms of: (a) classification rate, (b) number of hidden neurons and (c) time necessary for classification.

3.3 The GT Faces Database

Georgia Tech faces database contains images of 50 subjects taken in two or three sessions at the Signal Processing Center and the image in the Georgia Institute of Technology. All subjects in the database are represented by 15 color JPEG images with a noisy background and a resolution of 640×480 pixels. The average size of the faces in these images is 150×150 pixels. The images show the front taken and/or inclined with different facial expressions, lighting conditions and scale. Each image is manually labeled to determine the position of the face in the image. Fig. 7 shows a sample of the GT database.

Fig. 7. Some samples from the GT faces database.

3.3.1 Results

The algorithms, Linear Discriminant Analysis (LDA), PCA, and KPCA were used for the extraction of features from images of the GT database.

As in the previous experiment, the two training algorithms (PNN and CVPNN) were used to construct three PNNs each time.

The GT database samples were divided into:

- A training set containing 500 examples chosen randomly from the overall database.
- A test set which contains 250 remaining examples.

Table 2 shows the results of this study and Fig. 8 shows a comparison between both algorithms in terms of classification rate, processing speed and size of the hidden layer.

Table 2. Results for the GT faces database

	PNN			CVPNN		
	Classification rate (%)	**Time (s)**	**Hidden neurons**	**Classification rate (%)**	**Time (s)**	**Hidden neurons**
LDA	94.17	10.551	500	95.00	3.907	180
PCA	91.67	11.612	500	94.17	5.125	230
KPCA	90.00	12.556	500	93.33	5.150	220

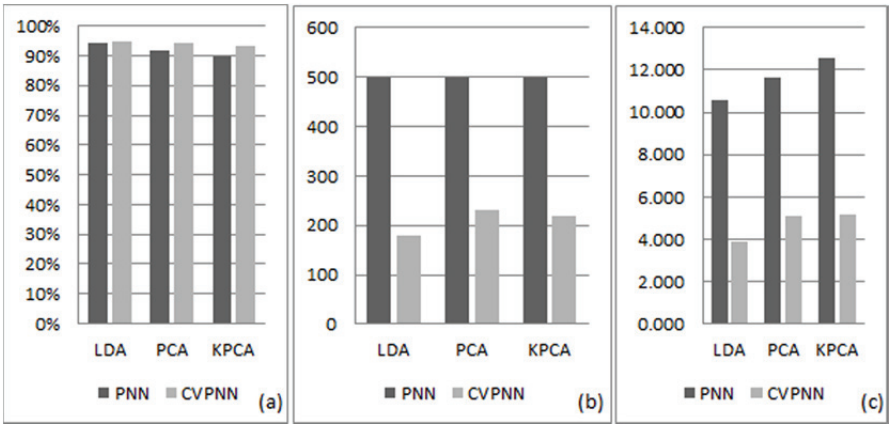

Fig. 8. Comparison between networks in terms of: (a) classification rate, (b) number of hidden neurons and (c) time necessary for classification.

4. Discussion

From Fig. 4, we can observe that the CVPNN produces almost the same separation areas like the standard PNN for all 3 classes. However, the number of neurons in the hidden layer of a CVPNN (21 neurons) is much lower than the number of neurons in the hidden layer of the standard PNN (137= number of samples). Thus, the size of the hidden layer is reduced to only 15% of the training samples and the time for classification of all points of the plan (100×100=10000) is reduced consequently. It is clear here that the CVPNN outperforms standard PNNs for classifying these data points both in the hidden layer's size and classification speed.

According to the results in Table 1 and Fig. 7, we notice that the classification results of the standard PNN network and the CVPNN network (proposed algorithm) are similar for both PCA and KPCA. However, the number of hidden neurons is very low for CVPNN (130 and 110 for the PCA and the

KPCA respectively against 280 hidden neurons in a standard PNN network). The gain in execution time is about 60% which is proportional to the number of neurons in the hidden layer.

From Table 2 and Fig. 8, we observe that despite the reduced number of neurons in the CVPNN network, this latter gives better performance compared to a standard PNN. Here, the gain is double: a gain in generalization and classification speed. Table 2 shows that for the LDA, for example, the CVPNN network gives 95% as classification rate with only 40% of hidden neurons. This is clearly a network with a better generalization than the standard PNN.

In this experiment, it was shown that by using the proposed training algorithms, one can easily gain in speed with almost the same performance in classification rate. The fact of using fewer neurons in the hidden layer while keeping the same classification rate means that the new classifier has high generalization ability compared to standard PNNs. This characteristic becomes more apparent for bigger training databases. Indeed, the reader can notice that the larger the training set the greater is the gain. Thus, the problem of over-fitting is reduced significantly for CVPNNs.

5. Conclusions

Results presented in this paper show that the new proposed CVPNN outperforms standard PNNs for all cases treated in this work. Significant improvements were reported not only in the size of the network and its processing speed but also in the overall generalization of the network. Actually, the proposed solution can be used for all cases where the database is large and where standard PNNs find serious problems with their huge hidden layer. It has been also shown that this network performs also well for very small databases with a more added complexity for the training algorithm.

For future works, the response of the network can be made even faster by using a parallel testing algorithm instead of a single threaded testing algorithm. In fact, parallelism is possible since connections to the summation layer are independent for each class. Hence each neuron in the class (summation) layer can calculate its activation using a separate dedicated execution thread. As a result, the processing time will depend greatly on the number of processors in the machine.

References

[1] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no 1, pp. 109-118, 1990.

[2] T. P. Tran, T. T. S. Nguyen, P. Tsai, and X. Kong, "BSPNN: boosted subspace probabilistic neural network for email security," *Artificial Intelligence Review*, vol. 35, no. 4, pp. 369-382, 2011.

[3] T. P. Tran, L. Cao, D. Tran, and C. D. Nguyen, "Novel intrusion detection using probabilistic neural network and adaptive boosting," 2009 [Online]. Available: https://arxiv.org/abs/0911.0485.

[4] F. Budak and E. D. Ubeyli, "Detection of resistivity for antibiotics by probabilistic neural networks," *Journal of Medical Systems*, vol. 35, no. 1, pp. 87-91, 2011.

[5] M. I. Faraj and J. Bigun, "Synergy of lip-motion and acoustic features in biometric speech and speaker recognition," *IEEE Transactions on Computing*, vol. 56, no 9, pp. 1169-1175, 2007.

[6] S. Meshoul and M. Batouche, "A novel approach for online signature verification using fisher based probabilistic neural network," in *Proceedings of IEEE Symposium on Computers and Communications*, Los Alamitos, CA, 2010, pp. 314-319.

[7] K. T. Blackwell, T. P. Vogl, H. P. Dettmar, M. A. Brown, G. S. Barbour, et D. L. Alkon, "Identification of faces obscured by noise: comparison of an artificial neural network with human observers," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no 4, pp. 491-508, 1997.

[8] A. Lotfi and A. Benyettou, "Using probabilistic neural networks for handwritten digit recognition," *Journal of Artificial Intelligence*, vol. 4, no. 4, pp. 288-294, 2011.

[9] L. F. Araghi, H. Khaloozade, and M. R. Arvan, "Ship identification using probabilistic neural networks (PNN)," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong, China, 2009, pp. 18-20.

[10] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*. San Rafael, CA: Morgan & Claypool Publishers, 2009.

[11] K. J. Nishanth and V. Ravi, "A computational intelligence based online data imputation method: an application for banking," *Journal of Information Processing Systems*, vol. 9, no. 4, pp. 633-650, 2013.

[12] A. Lotfi and A. Benyettou, "A reduced probabilistic neural network for the classification of large databases," *Turkish Journal of Electrical Engineering and Computer Science*, vol. 22, no. 4, pp. 979-989, 2014.

[13] K. Valentin and M. Maly, "Network firewall using artificial neural networks," *Computing and Informatics*, vol. 32, no. 6, pp. 1312-1327, 2014.

[14] P. Burrascano, "Learning vector quantization for the probabilistic neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 4, pp. 458-461, 1991.

[15] I. De Falco, A. Della Cioppa, and E. Tarantino, "Facing classification problems with particle swarm optimization," *Applied Soft Computing*, vol. 7, no. 3, pp. 652-658, 2007.

[16] V. L. Georgiou, P. Alevizos, and M. N. Vrahatis, "Fuzzy evolutionary probabilistic neural networks," in *Artificial Neural Networks and Pattern Recognition*. Heidelberg: Springer, 2008, pp. 113-124.

[17] M. W. Kim and M. Arozullah, "Generalized probabilistic neural network based classifiers," in *Proceedings of International Joint Conference on Neural Networks*, Baltimore, MD, 1992, pp. 648-653.

[18] I. Galleske and J. Castellanos, "A rotated kernel probabilistic neural network (RKPNN) for multi-class classification," in *Computational Methods in Neural Modeling.* Heidelberg: Springer, 2003, pp. 152-157.

[19] Y. Qi and J. Zhang, "Supervised kernel locally principle component analysis for face recognition," *Computing and Informatics*, vol. 31, pp. 1465-1479, 2013.

Abdelhadi Lotfi https://orcid.org/0000-0001-8229-0539

He is an Associate Professor at the National Institute of Telecommunication and Information and Communication Technology of Oran (INTTIC), Algeria. He is also member of the LaRATIC research laboratory at the INTTIC institute. He received his Ph.D. from the USTO-MB University and works now mainly on pattern recognition and intelligent systems.

Abdelkader Benyettou https://orcid.org/0000-0003-4370-2259

He is a Professor of Computer Science at the Department of Computing, Faculty of Mathematics and Computing, University of Sciences and Technology of Oran, Algeria. He is also head of the SIMPA laboratory. His research area includes signal processing, speech recognition and artificial intelligence.