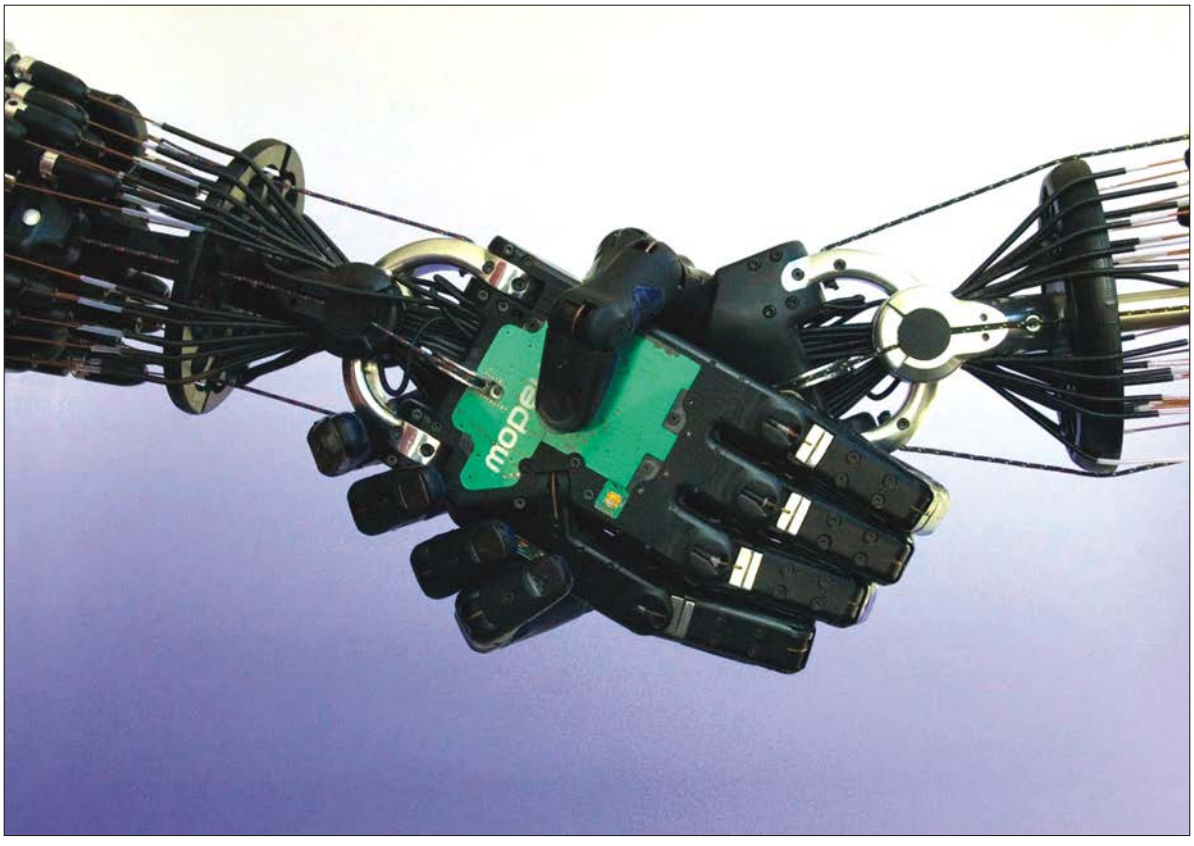

# Journal of Artificial Intelligence

ISSN 1994-5450

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Artificial Intelligence 4 (4): 288-294, 2011
ISSN 1994-5450 / DOI: 10.3923/jai.2011.288.294


# Using Probabilistic Neural Networks for Handwritten Digit Recognition

Abdelhadi Lotfi and Abdelkader Benyettou
21, Rue Miloud ben Med Gambetta 31000 Oran, Algeria

*Corresponding Author: Abdelhadi Lotfi, 21, Rue Miloud ben Med Gambetta 31000 Oran, Algeria*

## ABSTRACT

Artificial neural networks are well known in the field of pattern recognition and machine learning. Multi-layer neural networks are usually used as universal neural classifiers even though probabilistic neural networks represent a special type of artificial neural networks and have been designed to be used mainly in classification problems. In this article a study has been conducted to train a probabilistic neural network to recognize handwritten digits taken from the MINST database for handwritten digits. Results presented in this paper show good performance and generalization capacity of the proposed network for a real-world big database and no deep tuning of the parameters is required.

**Key words:** Pattern recognition, digit recognition, probabilistic neural network, classification, MINST database

## INTRODUCTION

The problem of pattern recognition depends essentially on the nature of the problem (complexity of the resolution) and the number of input arguments (size of the input space) (Bishop, 2006). For this reason, any proposed solution should take into account these two parameters. Most of pattern recognition techniques are used to solve a particular kind of problems. Artificial neural Networks (AN) and especially Multi-layer Merceptrons (MLP), are used for a wide variety of problems but for each problem, the parameters of the network have to be changed (usually by carrying a lot of experiments) to suite the problem in question (Galleske and Castellanos, 2003). This makes it difficult to use them for all kinds of problems.

The MLP neural network has been used extensively during the last decade to solve problems related to prediction, classification, regression, time series prediction... etc. (Li *et al*., 2011; Dawson and Wilby, 1999). Thus, they are considered as the standard in neural networks. But there are so many other types of ANNs in literature that are sometimes left behind. Probabilistic Neural Networks (PNN) have been designed by Specht (1990) "probabilistic neural networks for classification, mapping and associative memory" (Specht, 1990). They are a special kind of Radial Basis neural networks (RBF) (Tran *et al*., 2011). When it comes to classification, probabilistic neural networks are probably the best neural classifiers due to their design architecture (Bascil and Oztekin, 2010). They benefit also from a good mathematical basis since the idea is similar to that of interpolating a function with multi-variables (Duda, 2004).

The problem of handwriting recognition falls mainly into field of classification and pattern recognition (Dhandra *et al*., 2010). Recognizing handwriting is still a challenge when it comes to

noisy and deteriorated characters and digits (Bazarghan and Gupta, 2008). The MINST database is a good reference to test a classification technique on real world data. Standard MLPs were used in many papers to classify MINST digits (LeCun *et al.*, 1998) with a reported difficulty in the choice of the network parameters (initial parameters, number of hidden units, preprocessing techniques). In this work we tried to show that a probabilistic neural network can do the same job with less complexity and better accuracy than a standard MLP. Experimental results show that PNNs perform quite well without any deep concern about the choice of the network parameters.

## PROBABILISTIC NEURAL NETWORKS

**Mathematical background:** Probabilistic neural networks use radial basis functions as activation functions in the hidden layer to make a local decision function centered on a subset of the input space (Tran *et al.*, 2009). The global decision function is constructed by summing all local functions (Duda, 2004; Kim *et al.*, 2007). For this reason PNNs have the advantage over other multi-layer networks. The problem of local minimums does not affect the decision, of a PNN.

In a pattern classification context, each observed vector of cluster data x (d dimensional vector) is placed in one of the predefined cluster classes $C_i$ i = 1,2,..., m; where m is the number of possible classes in which x can belong to. The dimension of the input vector x and the number of possible classes m have a great impact on the performance of the network since a high number of input parameters affect the speed and accuracy of the network (Lotfi *et al.*, 2010). The Bayes pattern classifier implements the Bayes conditional probability rule that the probability P ($C_i|x$) of $x$ to belong to class $C_i$. This probability is given by:

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

When P ($C_i|x$) 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 $C_i$ if:

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

The major problem in Eq. 2 is that the prior probabilities P ($C_i|x$) (probability that the sample x comes from a population $C_i$) are most of time unknown for the system. To solve this problem, we can estimate these probabilities from the training samples. The Parzen windowing technique for pdf estimation (Duda, 2004) may present a good solution. Parzen proved that a class of pdf estimators asymptotically approaches the underlying parent density provided that it is smooth and continuous (Specht, 1990). The Parzen estimator used here is:

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

where, $X_{ai}$ is the $i^{th}$ pattern from class $C_A$ and $\sigma$ is a smoothing parameter.

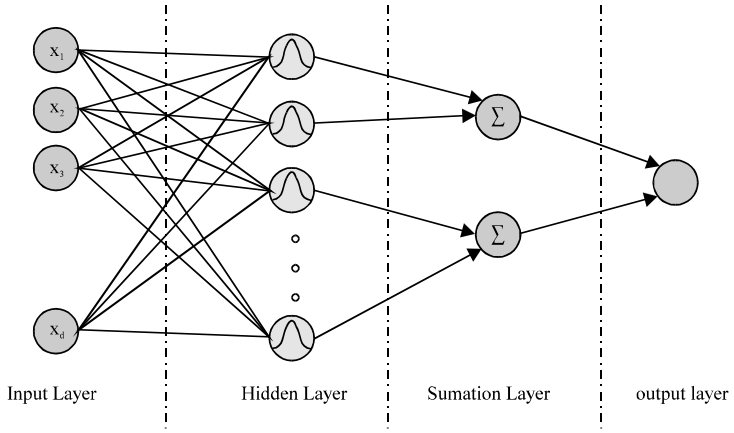


Fig. 1: Architecture of a standard probabilistic neural network

From Eq. 3, we can see that the global decision function $f_A$ (x) is the sum of small Gaussian functions centered on the training samples. This is similar to multi-dimensional interpolation.

Probabilistic neural networks use all the training dataset to estimate probability density functions P (x| $C_j$). These functions are then used to estimate the likelihood of the input sample as belonging to a given class (Specht, 1990).

**Architecture:** A typical architecture of a standard probabilistic neural network is given in Fig. 1. A PNN has four layers (Specht, 1990):

The input layer has as many neurons as the parameters (variables) of the system. Each neuron is fully connected to the neurons in next layer.

There is one hidden layer which contains the pattern units. The number of units in this layer is equal to the number of samples in the training set. The radial units are copied directly from the training set, one per case. Each one model a Gaussian function centered on a training sample. There is an output unit for each class which is connected to all hidden units belonging to this class and without connections to other units. This layer uses the following activation function:

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

where, the $w_i$ are the weights, the $x_i$ are the variables of the model and σ is a smoothing parameter. The smoothing parameter is the only parameter of the network that needs to be fixed at the beginning of the training.

The output of the pattern units is given by:

$$\phi_j^I(x) = \frac{1}{(2\pi)^{d/2}\sigma^2} \exp\left(-\frac{1}{2\sigma^2}\left(x - x_j^i\right)^t \left(x - x_j^i\right)\right) \tag{5}$$

where, $x_j^i$ is the jth training pattern vector from patterns in class $C_i$ and d is the size of the input space (number of parameters in the system).

In the summation layer, the number of neurons is the number of classes of data. Each neuron is connected to all neurons in the pattern layer that belong to the class represented by this unit. The output of each neuron in this layer represents the probability for the input vector X to belong to class $C_i$ and we can write:

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

where, $N_i$ is the number of training patterns from class $C_i$.
The input vector must be normalized to give the adequate separation in the hidden layer.

The output layer contains one neuron which gives the classification decision for the input vector X in accordance with the Bayes decision rule. The output is calculated by:

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

m is the number of classes presented in the system (Specht, 1990).

**Algorithm for training:** Unlike other neural networks' training algorithms that require a choice of many parameters (most of time with no exact method), the only parameter that is needed for probabilistic neural networks is the smoothing parameter which does not affect dramatically the performance of the network. In our case, the smoothing parameter was chosen using a small set from the training samples. The best results were given for $\sigma = 0.9$. The actual training consists simply of copying the training samples in the hidden units (one hidden node for each sample) by one pass through the training set. This makes a PNN very fast to train (Specht, 1990). The training algorithm is given by:

---

Start
- Initialization
  - j = 0
  - n = number of samples
  - Choice of σ
- 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 $x \in w_t$ then $a_k \leftarrow 1$
- Until j = n

End

---

**Algorithm for the test:** Each neuron in the hidden layer calculates its output using the formula:

$$\varphi(Z_k) = e^{\frac{z_k - 1}{\sigma^2}} \text{ with } Z_k = W_k^t x$$

σ is a smoothing parameter defined in the initialization step (using a subset of the training database).

In the summation layer, each neuron sums the contributions of all hidden neurons connected to the neuron representing a given class.

The output neuron gives the index (label) of the winner class.

As a result, the testing algorithm is then:

---

Start

Initialization

$k = 0$

$x$ = sample under test

Do

$k \leftarrow k+1$

$z_k \leftarrow w_k^t x$

If $a_{kc} = 1$ then $g_c \leftarrow g_c + e^{\frac{z_k - 1}{\sigma^2}}$

While $k < n$

Return $\leftarrow \arg\max(g_i(x))$

End

---

## DATABASE DESCRIPTION

Handwritten digit recognition represents a challenge for a probabilistic neural network since PNNs are usually applied on small benchmarking datasets. There are only few applications of these networks on real case large databases (Neggaz *et al*., 2010). Both the number of neurons in the input layer (variables) and the number of samples can affect dramatically a neural network from any type (Li *et al*., 2011).

The MINST database for handwritten digit recognition is a well known database in pattern recognition. It contains a large number of samples (60.000 samples for training and 10.000 samples for test). These are images of the size 20×20 black and white images of handwritten digits Fig. 2 written by 500 different writers and size normalized. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting (LeCun *et al*., 1998).

Fig. 2: The first 50 samples from the MINST database

Table 1: Number of samples for each class for 60.000 training samples and 10.000 testing samples

| Class | "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" |
|---|---|---|---|---|---|---|---|---|---|---|
| Training | 5922 | 6739 | 5957 | 6129 | 5840 | 5419 | 5918 | 6265 | 5851 | 5948 |
| Testing | 978 | 1133 | 1031 | 1010 | 980 | 891 | 957 | 1027 | 974 | 1007 |

Table 2: Results for digit recognition with different subsets of the MINST training database

| # training samples | # correct | # incorrect | % correct | % incorrect | Time |
|---|---|---|---|---|---|
| 1000 | 850 | 150 | 85 | 15 | 00:03:11 |
| 10000 | 820 | 180 | 82 | 18 | 00:33:50 |
| 20000 | 803 | 197 | 80,3 | 19,7 | 01:09:13 |
| 30000 | 1574 | 426 | 78,7 | 21,3 | 01:48:03 |
| 40000 | 1878 | 122 | 93,9 | 6,1 | 02:30:16 |
| 50000 | 1573 | 427 | 78,65 | 21,35 | 03:17:51 |
| 60000 | 1606 | 394 | 80,3 | 19,7 | 03:59:40 |

Some statistics about the training and testing database are given in Table 1. We notice that the number of samples in each class is almost the same which means that the database is more or less balanced. The digits "1" and "7" have more training samples than the others because they have almost the same form and can lead to misclassifications.

## EXPERIMENTAL RESULTS

The probabilistic neural network was tested with different subsets of the training and testing samples to observe the performance of the network as the number of training examples goes bigger. The smoothing parameter was fixed at 0.9 after many tests with a small set of samples. Table 2 shows the experiments carried out in this study. We tried to use the same testing samples for many experiments to show the effect of network size on the performance of the system.

## DISCUSSION

Results show that with a minimum of parameters (only the smoothing parameter), the probabilistic neural network performs well for handwritten digit recognition. It achieved 93.9% of correct detections for a PNN with 40.000 hidden neurons and 400 neurons in the input layer. For almost all cases performance decreases as the size of the network grows bigger. This mainly caused by the fact that noisy samples in the training samples lead to misclassifications. A good solution to this problem is to reduce the size of the network by taking out the neurons that contribute to misclassifications.

The time column represents the time necessary to classify 1000 images from the testing dataset. We observe that the time necessary for processing the test patterns is proportional to the size of the network. As the size of the network goes bigger, the time necessary for the test is important. This is why an effective probabilistic neural network should only take the most representative samples as training patterns. In this case, the first 1000 samples from the training set can lead to good classification with less hidden units which implies a faster response of the system.

## CONCLUSION

In this study probabilistic neural networks were used for handwritten digit recognition from the MINST database. Results show that PNNs perform well even for a large input space (number of parameters $20 \times 20 = 4000$ in this study) and with only the smoothing parameter as a variable

for the network (fixed after experiments at 0.9). No tuning of parameters is required for the stability of the proposed solution. Nevertheless, the size of the training set can affect dramatically the performance of the network if no prior choice of the representative samples is done. A smaller PNN with the right training samples is most of time better than a bigger one.

**ACKNOWLEDGMENT**

The authors would like to acknowledge all members of the editorial board and the reviewers for the great effort done in publishing high quality papers. We are thankful to all the members of the SIMPA laboratory for their constant help and support.

**REFERENCES**

Bascil, M.S. and H. Oztekin, 2010. A study on hepatitis disease diagnosis using probabilistic neural network. J. Med. Syst., (In Press).

Bazarghan, M. and R. Gupta, 2008. Automated classification of Sloan Digital Sky Survey (SDSS) stellar spectra using artificial neural networks. Astrophys. Space Sci., 315: 201-210.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. 1st Edn., Springer, Heidelberg, ISBN: 0-387-31073-8.

Dawson, C.W. and R.L. Wilby, 1999. A comparison of artificial neural networks used for river forecasting. Hydrol. Earth Syst. Sci., 3: 529-540.

Dhandra, B.V., R.G. Benne and M. Hangarge, 2010. Kannada, Telugu and Devanagari handwritten numeral recognition with probabilistic neural network: A novel approach. Int. J. Comput. Appl., 26: 83-88.

Duda, R.O., 2004. Pattern Classification 2nd Edition with Computer Manual 2nd Edition Set. John Wiley and Sons Inc., USA., ISBN-13: 9780471703501, Pages: 134.

Galleske, I. and J. Castellanos, 2003. A Rotated Kernel Probabilistic Neural Network (RKPNN) for multi-class classification. Comput. Methods Neural Model., 2686: 1040-1040.

Kim, D.K., D.H. Kim, S.K. Chang and S.K. Chang, 2007. Modified probabilistic neural network considering heterogeneous probabilistic density functions in the design of breakwater. KSCE J. Civil Eng., 11: 65-71.

LeCun Y., L. Bottou, Y. Bengio and P. Haffner, 1998. Gradient-based learning applied to document Recognition. Proc. IEEE, 86: 2278-2324.

Li, J., K. Ouazzane, H. Kazemian, Y. Jing and R. Boyd, 2011. A neural network based solution for automatic typing errors correction. Neural Comput. Appl., 20: 889-896.

Lotfi, A., K. Mezzoug and A. Benyettou, 2010. Rotated kernel neural networks for radar target detection in background noise. J. Applied Sci., 10: 1331-1335.

Neggaz, N., M. Besnassi and A. Benyettou, 2010. Application of improved aam and probabilistic neural network to facial expression recognition. J. Applied Sci., 10: 1572-1579.

Specht, D.F., 1990. Probabilistic neural networks. Neural Network, 3: 109-118.

Tran, T.P., L. Cao, D. Tran and C.D. Nguyen, 2009. Novel intrusion detection using probabilistic neural network and adaptive boosting. Int. J. Comput. Sci. Inform. Secur., 6: 83-91.

Tran, T.P., T.T.S. Nguyen, P. Tsai and X. Kong, 2011. BSPNN: Boosted subspace probabilistic neural network for email security. Artif. Intell. Rev., 35: 369-382.