Probabilistic Neural Network in Solving Various Pattern Classification Problems

Nabha B Nimbhorkar#1, Satish J Alaspurkar*2

Computer Science & Engineering Department,
Amravati University , India

Abstract

A probabilistic neural network (PNN) is a feed forward neural network, which was derived from the Bayesian network and a statistical algorithm called Kernel Fisher discriminant analysis. The feed forward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. Basically there consists of four layers : input layer, pattern layer, summation layer and output layer. The Network structure determination is an important issue in pattern classification based on a probabilistic neural network. In this study, a supervised network structure determination algorithm is proposed. The proposed algorithm consists of two parts and runs in an iterative way. The first part identifies an appropriate smoothing parameter using a genetic algorithm, while the second part determines suitable pattern layer neurons using a forward regression orthogonal algorithm. The proposed algorithm is capable of offering a fairly small network structure with satisfactory classification accuracy**.**

Keywords

Genetic Algorithm, orthogonal algorithm, pattern classification, probabilistic neural network (PNN), Feed forward Regression Network

I. INTRODUCTION

Probabilistic Neural Network PNN often learn more quickly than many neural network models such as back propagation networks and have had success on a variety of applications. PNNs are a special form of Radial Basis Function (RBF) network used for classification which is the major job of this paper. The network learns from a training set T which is a collection of examples called instances [1]. Each instance i has an input vector yi and an output class denoted as class i. During execution, the network receives additional input vectors denoted as x and outputs the class that x seems most likely to belong to. With regards to the real time classification problem which is the main attention of this paper, PNN has proven to be more time efficient than conventional back propagation based networks. In order to classify a feature pattern [2] vector x € Rm, that is to assign the pattern to one among k predefined classes, the conditional density P (x¦ Ck) of each class Ck is estimated since it represents the uncertainty associated to class attribution; then these estimates are combined by the rule of Baye's to yield a posterior class probabilities P (Ck¦ x) that allow to make optimal decisions

II. A BRIEF REVIEW OF THE PNNS

Consider a classification problem where the pattern data, designated by the vector x belong to different classes,
designed by the letters A, B, C,..., etc. The Bayesian classifier is the classifier that minimizes the probability of misclassifying the labels of unseen data. The Bayesian classifier chooses as the predicted label of an unseen pattern the label l that maximizes the following a-posterior probability. In order to calculate the above probabilities for every label l = A, B, C, one needs to compute the class conditional probabilities p(x | l) for every l , and the a-priori probabilities P(l) . The a-priori probabilities P(l) can be estimated from the available training data. The class conditional probabilities p(x | l) can also be estimated using the training data, by using an approximation for the probability density function formula, suggested by Parzen and depicted below, for the class label l = A.

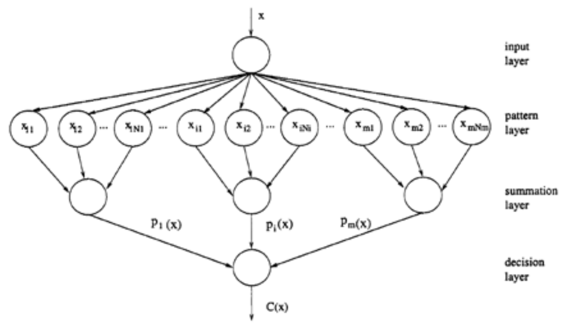

Fig. 1. Diagram of a PNN.

The PNN was first proposed in . The architecture of a typical PNN is as shown in Fig. 1. The PNN architecture is composed of many interconnected processing units or

Manuscript received March 5, 2014
Manuscript revised March 20, 2014

neurons organized in successive layers. The input layer unit does not perform any computation and simply distributes the input to the neurons in the pattern layer. On receiving a pattern from the input layer, the neuron of the pattern layer computes its output

$$P(l \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid l) \cdot P(l)}{p(\mathbf{x})}$$

$$p(\mathbf{x} \mid A) = \frac{1}{(2\pi)^{D/2}} \cdot \frac{1}{N_{tr}(A)} \cdot \sum_{t \in S_{tr}(A)} \prod_{d=1}^{D} \frac{1}{\sigma_t^A(d)} \exp - \left(\frac{(\mathbf{x}(d) - \mathbf{x}_t^A(d))^2}{2(\sigma_t^A(d))^2} \right)$$

III. Determining THE PNN STRUCTURE USING ORTHOGONAL ALGORITHM AND THE GENETIC ALGORITHM

In this section, we propose a supervised PNN structure determination algorithm that incorporates an appropriate constraint on classification error rate.

A. CONSTRUCT THE PATTERN USING FEEDFORWARD NETWORK

At this stage, it is assumed that the smoothing parameter has been chosen. The objective is to select representative pattern layer neurons from the training samples. As described in the previous section, for the kth training pattern in class Ci denoted by vector Xik , the maximum likelihood to be classified to is Ci

$$\begin{aligned} p_i(x_{ik}) &= \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{N_i} \sum_{j=1}^{N_i} \\ &\quad \cdot \exp\left[-\frac{(x_{ik} - x_{ij})^T (x_{ik} - x_{ij})}{2\sigma^2} \right] \\ &= \sum_{j=1}^{N_i} \phi_{ij}(x_{ik}) \end{aligned}$$

where

$$\phi_{ij}(x_{ik}) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{N_i} \exp\left[-\frac{(x_{ik} - x_{ij})^T (x_{ik} - x_{ij})}{2\sigma^2} \right]$$

B. Selecting the Smoothing Parameter Using Genetic Algorithms

Typically, a genetic algorithm consists of the following operations; encoding, fitness value assignment, reproduction, crossover and mutation. Details of the GA-based PNN structure detection algorithm are described below.

Encoding: GA works with the coding of parameters rather than the parameter themself. If samples are normalized, the smoothing parameter should be smaller than one, only fraction part needs to be coded. A four-bit decimal coding is employed in the present study to encode the smoothing parameter. For example, one individual is, this value can be represented by the following decimal string:.

$$\overbrace{6}^{b_1} \quad \overbrace{2}^{b_2} \quad \overbrace{5}^{b_3} \quad \overbrace{7}^{b_4}$$

where bi denote the bit of 10to the power –i

Fitness Evaluation: Each individual represents a smoothing parameter value. With the use of neuron selection algorithm developed in Section III-A and smoothing parameters defined by all individuals, a number of candidate network structures can be obtained. The objective is to minimize the neural-network size, therefore the fitness function should be inversely proportional to the number of selected neurons. The fitness can be computed using the following mapping scheme:

$$\rho_i = \rho_{\max} - \frac{\rho_{\max} - \rho_{\min}}{n_{\max} - n_{\min}} (n_i - n_{\min})$$

where pi Nmin and Nmax denotes the fitness value of the th individual. ,,Pmin and Pmax , are the minimum and maximum sizeof candidate network structure in the current population, and the minimum and maximum fitness values, respectively. In this study andPmin and Pax are set to 0.5 and 1, respectively

Reproduction: The roulette wheel approach is employed to implement the reproduction procedure. Each string is allocated a slot of the roulette wheel subtending an angle proportional to its fitness. A random number in the range of 0 to 2 is generated. A copy of string goes to the mating pool if the random number falls in the slot corresponding to the string. The reproduction is repeated to generate a mating pool with a prespecified size.

Crossover: The purpose of crossover operation is to generate new solutions by exchanging bits between individuals. Assuming two randomly selected parent individuals are given by string i and string j

$$\overbrace{6}^{b_1} \quad \overbrace{2}^{b_2} \quad \overbrace{5}^{b_3} \quad \overbrace{7}^{b_4} \qquad \text{string } i$$

$$\overbrace{0}^{b_1} \quad \overbrace{4}^{b_2} \quad \overbrace{3}^{b_3} \quad \overbrace{2}^{b_4} \qquad \text{string } j.$$

First, randomly select the bit at which the two strings will be changed, for example b2 . Then exchanging the values at bit b2 for the two strings yields two offspring strings:

$$\overbrace{6}^{b_1}\ \overbrace{4}^{b_2}\ \overbrace{5}^{b_3}\ \overbrace{7}^{b_4} \quad \text{string } k$$

$$\overbrace{0}^{b_1}\ \overbrace{2}^{b_2}\ \overbrace{3}^{b_3}\ \overbrace{2}^{b_4} \quad \text{string } l.$$

Mutation: The purpose of employing mutation is to generate an individual that is not easy to achieve by the crossover operation. In this study, the mutation is achieved by changing the selected bit with a random number between zero to nine. For example if the bit b1 of string k is supposed to mutate, changing this bit to a random generated number, say eight, yields the following string:

$$\overbrace{8}^{b_1}\ \overbrace{4}^{b_2}\ \overbrace{5}^{b_3}\ \overbrace{7}^{b_4} \quad \text{string } z.$$

iv .PIN Architecture AND THEORY OPERATION

The probabilistic Neural Network used in this paper is shown in Fig. 1. The first (leftmost) layer contains one input node for each input attribute in an application. All connections in the network have a weight of 1, which means that the input vector is passed directly to each hidden node [1]. A novel PNN (Probabilistic Neural Networks) hardware architectureis also as called 'Sigma Parallel Architecture' (SPA). Different values of the network parameter are calculated in parallel in the SPA and it speeds up the PNN learning as well as recognition overcoming the difficulty in the VLSI implementation. The hardware prototype is developed using FPGA chips and it shows a high speed leaning of about 10 seconds that satisfies the requirements in the real world image recognition tasks

In PNN, there is one hidden node for each training instance i in the training set. Each hidden node hi has a center point yi associated with it, which is the input vector of instance i. A hidden node also has a spread factor, si , which determines the size of its respective field. There are a variety of ways to set this parameter. Si is equal to the fraction f of the distance to the nearest neighbor of each instance i. The value of f begins at 0.5 and a binary search is performed to fine tune this value. At each of five steps, the value of f that results in the highest average confidence of classification is chosen [1]. A hidden node receives an input vector x and outputs an activation given by the Gaussian function g, which returns a value of 1 if x and yi are equal and drops to an insignificant value as the distance grows [1]

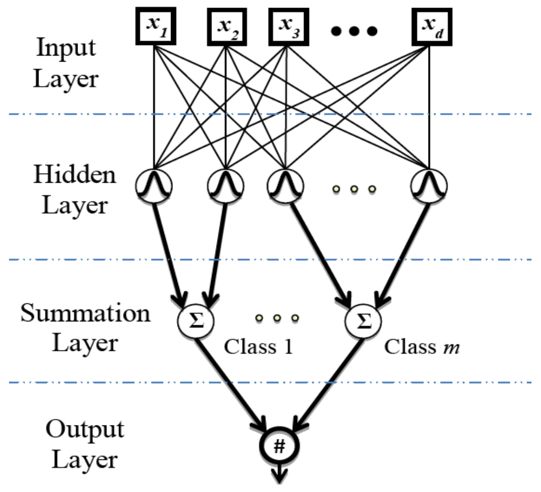

Fig:2 A General Architecture Of PNN

$$g(x, y_i, s_i) = \exp(-D^2(x, y_i)/2s_i^2)$$

The distance function D determines how far apart the two vectors are. By far the most common distance function used in PNNs is Euclidean distance. However, in order to appropriately handle applications that have both linear and nominal attributes, a heterogeneous distance function HVDM [3,4] is used to normalize Euclidean distance for linear attributes and the Value Difference Metric (VDM) [5] for nominal attributes. It is defined as:

$$\mathrm{HVDM}(x, y) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)}$$

Where m is the number of attributes. The function da (x, y) returns a distance between the two values x and y for attribute a and is defined as:

$$d_a(x, y) = \begin{cases} 1 & \text{if x or y is unknown} \\ \mathrm{vdm}_a & \text{if a is nominal} \\ \mathrm{diff}_a & \text{if a is linear} \end{cases}$$

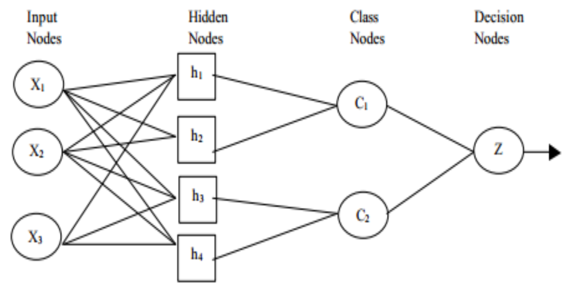

Fig. 3: Probabilistic neural network architecture

Where m is the number of attributes. The function da (x, y) returns a distance between the two values x and y for attribute a and is defined as:

1 if x or y is unknown
d (x, y) = vdm if a is nominal
diff if a is linear

The function da (x, y) uses the following function, based on the Value Difference Metric (VDM) [5]for nominal (discrete, unordered) attributes

$$\mathrm{Vdm}_a(x, y) = \sqrt{\sum_{c=1}^{C} |(N_{a,x,c}/N_{a,x}) - (N_{a,y,c}/N_{a,y})|^2}$$

Where Na, x is the number of times attribute a had value x; Na, x, cis the number of times attribute a had value x and the output class was c and C is the number of output classes.

For linear attributes, the following function is used.

$$\mathrm{Diff}_a(x, y) = |x-y|/4s_a$$

Where sa is the sample standard deviation of the value occurring for attribute a in the training set. Each hidden node hi in the network is connected to a single class node. If the output class of instance i is j, then hi is connected to class node Cj . Each class node Cj computes the sum of the activations of the hidden nodes that are connected to it (i.e., all the hidden nodes for a particular class) and passes this sum to a decision node. The decision node outputs the class with the highest summed activation. One of the greatest advantages of this network is that it doesn't require any iterative training and thus can learn quite quickly. However, one of the main disadvantages of this network is that it has one hidden node for each training instance and thus requires more computational resources (storage and time) during execution than many other models. When simulated on a serial machine, O(n) time is required to classify a single input vector. On parallel system, only O(log n) time is required, but n nodes and nm connections are still required (where n is the number of instances in the training set and m is the number of input attributes) .The most directly way to reduce storage requirement and speed up execution is to reduce the number of nodes in the network. One common solution to this problem is to keep only a randomly selected subset of the original training data in building thenetwork. However, arbitrary removing instances can reduce generalization accuracy. In addition, it is difficult to know how many nodes can be safely removed without a reasonable stopping criterion. Other subset selection algorithm exist in linear regression theory, including forward selection, in which the network starts with no nodes and nodes are added one at a time to the network. Another method that has been used is k-means clustering densities are then used in a Bayes decision rule to perform the classification. If the probability density function (pdf) of each of the population is known, then an unknown x belong to class i if fi (x)>fj (x), for all j¹i.

V. DATABASE PROCESSING

Since the PNN is usually applied on small benchmarking datasets, handwritten digit recognition represents a challenge for a standard probabilistic neural network. There are only few applications of this type of networks on real world big databases [20]. Both the number of neurons in the input layer (dimensionality) and the number of samples (available labeled data) can affect dramatically a neural network from any type

In order to test the reduced probabilistic neural network on a real world big database, the new classifier was tested on the MNIST database for handwritten digit recognition. This database is well known in pattern recognition. It contains a large number of samples (60,000 samples for training and 10,000 samples for test). The samples are 20×20 pixels black and white images of handwritten digits (see figure 3). The different Arabic digits were written by 500 different writers and size normalized. This is a good database for people who want to try pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting of data. Database search post-processing by neural network was employed in peptide mapping experiments. The database search was performed using both the known algorithms and score functions, such as Bayesian, MOWSE, Z-score, correlations between calculated and actual peptide length fractional abundance, and, in addition, the probability of protein digest pattern in peptide fingerprint, all embedded in locally developed program. The new signal-processing algorithm based on neural network improves signal-noise separation and is acceptable for automatic protein identification in mixtures. Its power was tested on Helicobacter pylori protein inventory after preceding protein separation by sodium dodecyl sulfate-polyacrylamide gel electrophoresis (SDS-PAGE). Increase in protein identification success rate was observed, and about 100 proteins were identified with no need of human participation in database search estimation

Fig: 4. some samples from the MNIST database.

A. High Level Algorithm:

We are given the exemplar feature vectors that make up the training set. For each one we know the class to which it belongs. The following sets up the PNN.

Step 1. Read in the file of exemplar vectors and class numbers

Step 2. Sort these into the K sets where each set contains one class of vectors

Step 3. For each k define a Gaussian function centered on each exemplar vector in set k define the summed Gaussian output function

Once the PNN is defined, then we can feed vectors into it and classify them as follows.

Step 1. Read input vector and feed it to each Gaussian function in each class

Step 2. For each group of hidden nodes, compute all Gaussian functional values at the hidden nodes

Step 3. For each group of hidden nodes, feed all its Gaussian functional values to the single output
node for that group
Step 4. At each class output node, sum all of the inputs and multiply by constant
Step 5. Find maximum value of all summed functional values at the output nodes

VI. CONCLUSION

A self adaptive model for probabilistic neural networks was proposed. The proposed approach incorporates the Orthogonal Algorithm to _nd an appropriate spread parameter for the probabilistic neural network with respect to the resulting classification accuracy. The effectiveness of the proposed model is assessed on two data sets from the _eld of bioinformatics (E.coli and Yeast), with encouraging results. Among the three sampling techniques considered, strati_ed random sampling proved to yield the best classification performance despite the fact that the training sets it produces are smaller in magnitude compared to 10.fold cross validation. To evaluate further the capabilities of this model, comparisons with feed forward neural network models trained using the R.PROP algorithm were performed. These comparisons showed that the self adaptive model proposed achieves statistically signi_cant superior performance on the Yeast data set, while on the E.coli data set the performance of the two models is not found to be statistically different. Future work will include the generalization of the proposed self adaptive scheme to determine the corresponding matrix of spread parameters, to further optimize the performance of PNNs. We also intend to apply different evolutionary computation techniques on this problem.

ACKNOWLEDGMENT

The author is thankful Mr.S.J.Alaspurkar of Computer Science Department of G..H.Raisoni College Of engineering for his perfect logistic support provided for the neural networks and the pattern classification problems based on it

REFERENCES

[1] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley, 1989.
[2] C. Kramer, B. Mckay, and J. Belina, “Probabilistic neural network arrayarchitecture for ECG classification,” in Proc. Annu. Int. Conf. IEEE Eng.Medicine Biol., vol. 17, 1995, pp. 807–808.
[3] L. I.Kuncheva and J. C. Bezdek, “Nearest prototype classification: Clustering, genetic algorithms, or random search,” IEEE Trans. Syst., Man, Cybern. C, vol. 28, pp. 160–164, Feb. 1998
[4] S. Ma and C. Ji, “Performance and efficiency: Recent advances in supervised learning,” Proc. IEEE, vol. 87, pp. 1519–1535, 1999.
[5] C.G. Atkeson. (1991) Using locally weighted regression for robot learning. Proceedings of the 1991 IEEE Conference on Robotics and Automation, pp. 958{963. Los Alamitos, CA: IEEE Computer Society Press
[6] D. Goldberg. (1988) Genetic Algorithms in Machine Learning, Optimization and Search. Redwood City, CA: Addison-Wesley.
[7] M. T. Musavi, K. H. Chan, D. M. Hummels, and K. Kalantri, “On the generalization ability of neural-network classifier,” IEEE Trans. Pattern Anal. Machine Intell., vol. 16, no. 6, pp, 1994.
[8] P. P. Raghu and B. Yegnanarayana, “Supervised texture classification using a probabilistic neural network and constraint satisfaction model,” IEEE Trans. Neural Networks, vol. 9, pp. 516–522, May 1998.
[9] B.V. Dasarathy. (1991) Nearest Neighbor (NN) Norms: NN Pattern Classi_cation Techniques. Los Alamitos, CA: IEEE Computer Society Press.
[10] K. Z. Mao, K.-C. Tan, and W. Ser “Probabilistic Neural-Network Structure Determination for Pattern Classification” IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 11, NO. 4, JULY 2000
[11] Maida,Anthony S.”Identifieng causual structure in a biological neural network” ICTAI 2000 Proceedings 12th IEEE International Conference.