Probabilistic Neural Network

Sayeda Suaiba Anwar
Lecturer, Department of CSE
East Delta University

Introduction

- Probabilistic Neural Network (PNN) is a feed forward neural network which is used for classification and pattern recognition problems.
- In mathematical terms, an input vector (called a feature vector) is used to determine a category and the network classifiers are trained by being shown data of known classifications.
- The PNN uses the training data to develop distribution functions that are in turn used to estimate the likelihood of a feature vector being within several categories.
- Ideally this can be combined with a priori probability (relative frequency) of each category to determine the most likely category for a given feature vector.

Bayesian Probability

- PNN is a neural network implementation of Bayesian classifiers.
- Bayesian inversion formula gives

$$P(X|Y) = \frac{P(Y|X) \cdot P(X)}{P(Y)}$$

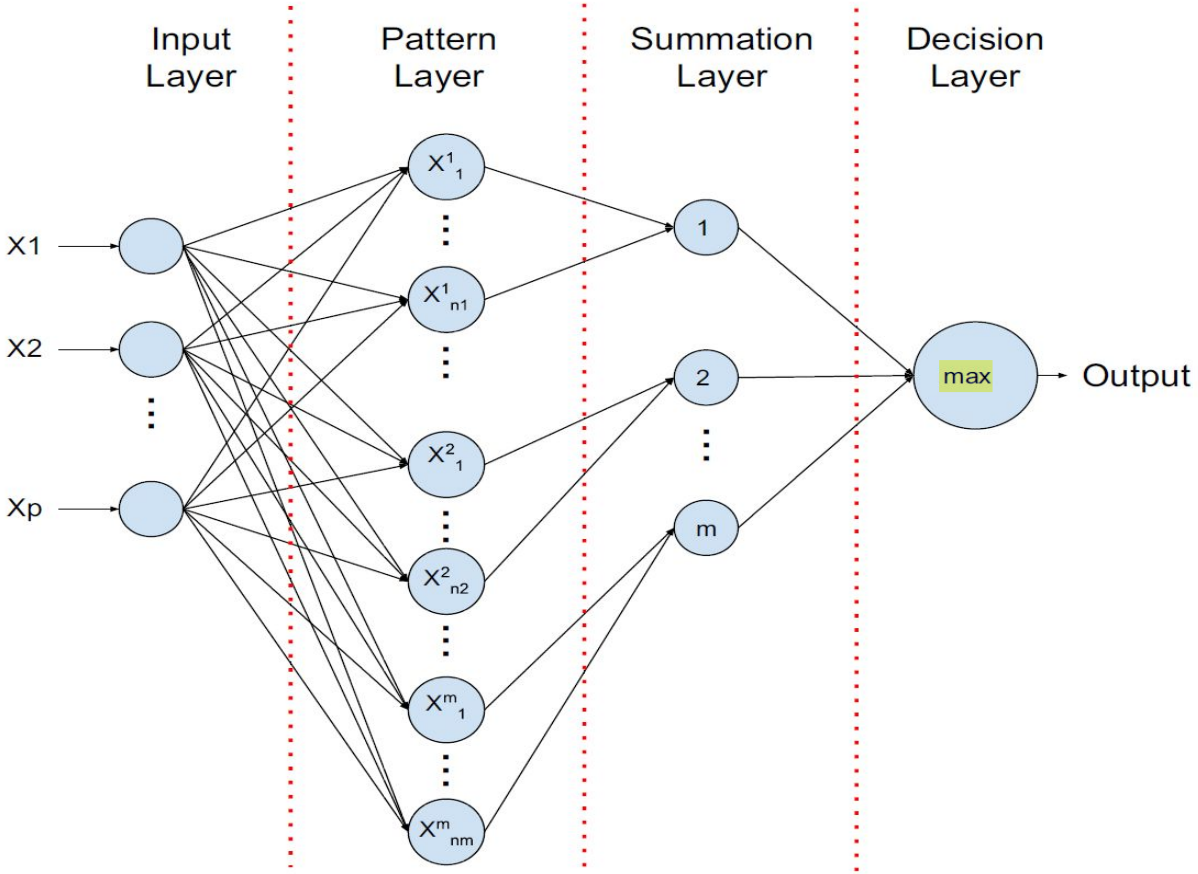
The relationship P(Y|X) is called the posteriori (the posterior) probability indicating that the probability is known only after the event X itself has occurred.

Bayesian formula also provides a method for categorizing patterns.

Bayesian Probability

- In this formulation, Y is interpreted as a possible category into which a pattern might be placed and X is interpreted as the pattern itself.
- The decision function can be associated with each possible category (all values of Y).
- Bayesian decision theory tries to place a pattern in the category that has the greatest value of its decision function.
- In real world problems, we rarely have known probabilities and must estimate or approximate such Bayesian probabilities.
- A probabilistic neural network has this capability.

Structure of PNN



Structure of PNN

PNN consists of four layers.

1. Input Layer :

   P neurons represent the input vector and distribute it to the next layer.
   P equals the number of input features.

2. Pattern Layer:
   - Fully connected with input layer, with one neuron for each pattern in the training set.
   - Each of the neurons in the pattern layer performs a weighted sum of the incoming signals from the input layer and then applies a nonlinear activation function to give the neuron's output.

Structure of PNN

3. Summation Layer:

- This layer computes the average of the output of the pattern units for each class.
- There's one neuron for each class. Each class neuron is connected to all neurons in the pattern layer of that class.

4. Output Layer:

- This layer selects the maximum value from the summation layer, and the associated class label is determined accordingly.

Structure of PNN

The nonlinear activation function used by pattern layer neutrons is not a sigmoidal function but instead is an exponential function as shown in Figure 9.14. This activation function is given by

$$\Phi(I_i) = \exp\left[(I_i - 1)/\sigma^2\right] \tag{9.6-2}$$

where I is the weighted input to the neuron and the σ is the smoothing parameter that determines how smooth the surface separating categories will be. A reasonable range of values for σ is 0.1 to 10. The reason for the exponential activation function is that it is a simplification of the Parzan estimator of a Bayesian surface. Using a Bayesian estimating function in the pattern layer neurons allows the PNN to approximate Bayesian probabilities in categorizing patterns.

Structure of PNN

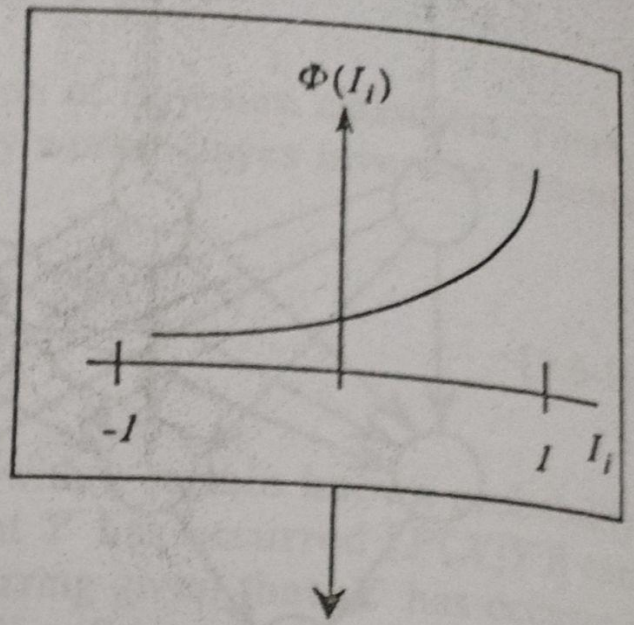


Figure 9.14 Exponential activation function for pattern layer neurons in a probabilistic neural network.

$$\Phi(I_i) = \exp\left[\frac{I_i - 1}{\sigma^2}\right]$$

Structure of PNN

The pattern layer has one neuron for each pattern in the training set. if there are 20 patterns in the training set, 12 in category A, and 8 in category B, then there are 20 neurons in the pattern layer. Each of these neurons has a set of weighted connections between it and the input layer. Each pattern layer neuron is assigned to one of the 20 training patterns, which connects to the summation layer neuron that represents its patterns category. Since in this case the summation layer has two neurons, the category A neuron receives inputs only from the 12-pattern layer neurons that represent category A, and the category B neuron receives inputs only from the eight-pattern layer neurons that represent categories B patterns. The weights on the connections from the pattern layer to the summation layer are fixed at unity.

Each neuron in the

Structure of PNN

unity.

Each neuron in the output layer received only two inputs, one from each of two summation units. One weight is fixed with a strength of unity; the other weight has a variable strength equal to

$$w' = -[h_B/h_A][l_B/l_A][n_A/n_B] \qquad (9.6\text{-}3)$$

where h refers to *a priori* probability of patterns being in category A or B, l is the loss associated with identifying a pattern as being in one category when it is in reality in the other category, and n is the number of A or B patterns in the training set. The values for h_A, h_B, n_A, and n_B are determined by the data pattern themselves, but the losses must be based on knowledge of the application. In many real-world cases there is no difference in loss if the categorization is wrong in one direction or the other. If so and if the training samples are present at approximately the ratio of their overall likelihood of occurrence, this weight reduces to unity.

Smoothing Parameter

A smoothing parameter which affects the generality of decision boundaries can be modified without retraining. The PNN usually needs a reasonable number of training samples for good generalization, but it can give good results with a small number of training samples. Since each training sample is represented by a neuron in the pattern layer, this serial implementation of a PNN will typically take longer in the recall mode than a backpropagation model. Inputs need not be normalized, which in some cases may distort the inputs space in an undesirable way. However, some implementations of PNN do normalize inputs for convenience.

The smoothing parameter σ varies between zero and infinity, but neither limit provides an optimal separation. A degree of averaging of nearest neighbors provides better generalization where the degree of averaging is dictated by the density of the training samples. Figure 9.15 shows the smoothing parameter σ as it varies between 0.1 and 1.0. For the lowest value the estimated probability density function has five distinct neurons, whereas for the largest variable there is a very severe flattening of the probability density function between -3 and $+3$.

Smoothing Parameter

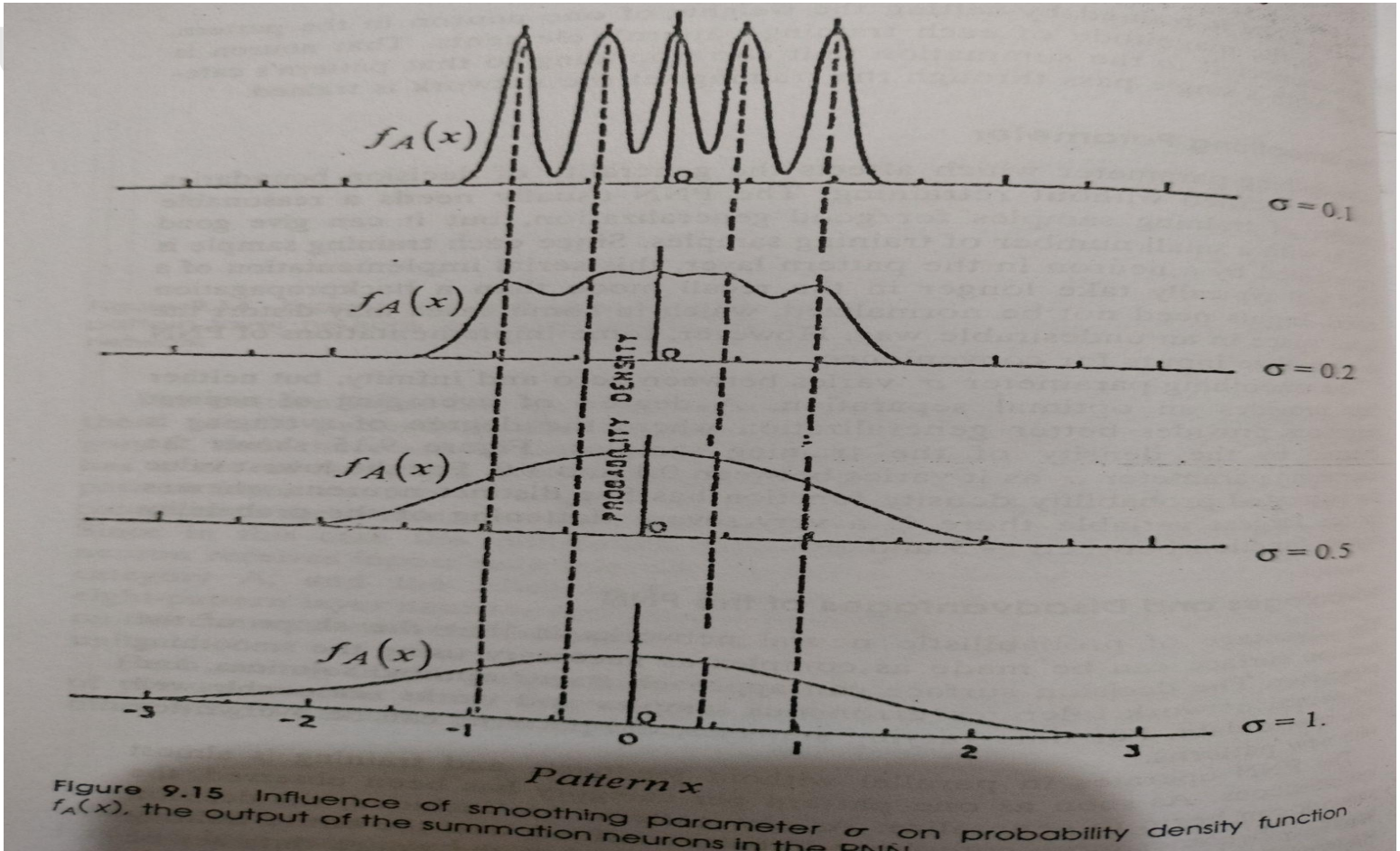


Figure 9.15 Influence of smoothing parameter σ on probability density function $f_A(x)$, the output of the summation neurons in the PNN.

Advantages and Disadvantages of PNN

Advantages :

1. In a PNN, there's no extensive training computation time associated with networks that use back-propagation. Instead, each data pattern is represented with a unit that measures the similarity of the input pattern to the data patterns.
2. PNN learns from the training data instantaneously. With this speed of learning, PNN has the capability to adapt its learning in real time, deleting or adding training data as new conditions arise.
3. Additionally, PNNs are relatively insensitive to outliers and approach Bayes optimal classification as the number of training samples increases.
4. PNNs are guaranteed to converge to an optimal classifier as the size of the representative training set increases.

Advantages and Disadvantages of PNN

Disadvantages :

1. PNN can’t deal with extremely large training datasets. Because there's one hidden node for each training instance, more computational resources (storage and time) during inference.
2. Additionally, the performance of the system usually decreases in terms of the classification accuracy and speed with a very big hidden layer.

Suggested Reading

https://devopedia.org/probabilistic-neural-network