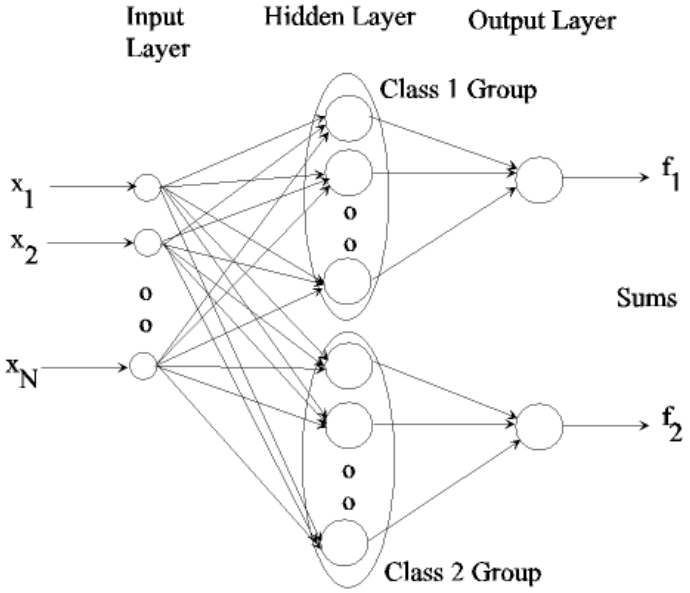
Probabilistic Neural Network Tutorial

The Architecture of Probabilistic Neural Networks

A *probabilistic neural network* (PNN) has 3 layers of nodes. The figure below displays the architecture for a PNN that recognizes K = 2 classes, but it can be extended to any number K of classes. The input layer (on the left) contains N nodes: one for each of the N input features of a feature vector. These are fan-out nodes that branch at each feature input node to all nodes in the hidden (or middle) **layer** so that each hidden node receives the complete input feature vector **x**. The hidden nodes are collected into groups: one group for each of the K classes as shown in the figure.



Each hidden node in the group for Class k corresponds to a Gaussian function centered on its associated feature vector in the k^{th} class (there is a Gaussian for each exemplar feature vector). All of the Gaussians in a class group feed their functional values to the same output layer node for that class, so there are K output nodes.

How the PNN Works

At the output node for Class k (k = 1 or 2 here), all of the Gaussian values for Class k are summed and the sum is scaled to so the probability volume under the sum function is unity so that the sum forms a *probability density function*. Here we temporarily use special notation for clarity. Let there be P exemplar feature vectors $\{\mathbf{x}^{(p)}: p = 1,...,P\}$ labeled as Class 1 and let there be Q exemplar feature vectors $\{\mathbf{y}^{(r)}: r = 1,...,R\}$ labeled as Class 2. In the hidden layer there are P nodes in the group for Class 1 and R nodes in the group for Class 2. The equations for each Gaussian centered on the respective Class 1 and Class 2 points $\mathbf{x}^{(p)}$ and $\mathbf{y}^{(q)}$ (feature vectors) are (where N is the dimension of the vectors) are, for any input vector **x**

$$g_1(\mathbf{x}) = [1/\sqrt{(2\pi\sigma^2)^N}]\exp\{-\|\mathbf{x} - \mathbf{x}^{(p)}\|^2/(2\sigma^2)\} \quad (1)$$

$$g_2(\mathbf{y}) = [1/\sqrt{(2\pi\sigma^2)^N}]\exp\{-\|\mathbf{y} - \mathbf{y}^{(q)}\|^2/(2\sigma^2)\} \quad (2)$$

The σ values can be taken to be one-half the average distance between the feature vectors in the same group or at each exemplar it can be one-half the distance from the exemplar to its nearest other exemplar vector. The k^{th} output node sums the values received from the hidden nodes in the k^{th} group, called *mixed Gaussians* or *Parzen windows*. The sums are defined by

$$f_1(\mathbf{x}) = [1/\sqrt{(2\pi\sigma^2)^N}](1/P)\Sigma_{(p=1,P)} \exp\{-\|\mathbf{x} - \mathbf{x}^{(p)}\|^2/(2\sigma^2)\} \quad (3)$$

$$f_2(\mathbf{y}) = [1/\sqrt{(2\pi\sigma^2)^N}](1/Q)\Sigma_{(q=1,Q)} \exp\{-\|\mathbf{y} - \mathbf{y}^{(q)}\|^2/(2\sigma^2)\} \quad (4)$$

where $\mathbf{x}$ is any input feature vector, σ_1 and σ_2 are the spread parameters (standard deviations) for Gaussians in Classes 1 and 2 , respectively, N is the dimension of the input vectors, P is the number of center vectors in Class 1 and R is the number of centers in Class 2, $\mathbf{x}^{(p)}$ and $\mathbf{y}^{(r)}$ are centers in the respective Classes 1 and 2, and $\|\mathbf{x} - \mathbf{x}^{(p)}\|$ is the Euclidean distance (square root of the sum of squared differences) between $\mathbf{x}$ and $\mathbf{x}^{(p)}$.

Any input vector $\mathbf{x}$ is put through both sum functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ and the maximum value *(maximum a posteriori*, or MAP value) of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ decides the class. For K > 2 classes the process is analogous.

There is no iteration nor computation of weights. For a large number of Gaussians in a sum, the error buildup can be significant. Thus the feature vectors in each class may be reduced by thinning those that are too close to another one and making σ larger.

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