An Introduction to Probabilistic Neural Networks

Vincent Cheung

Kevin Cannons

Signal & Data Compression Laboratory

Electrical & Computer Engineering

University of Manitoba

Winnipeg, Manitoba, Canada

Advisor: Dr. W. Kinsner

June 10, 2002

University of Manitoba

Outline

- Introduction
- Classifier Example
- Theory and Architecture
- Training
- Program Implementations
- Conclusion

What is a PNN?

- A probabilistic neural network (PNN) is predominantly a classifier
  - Map any input pattern to a number of classifications
  - Can be forced into a more general function approximator
- A PNN is an implementation of a statistical algorithm called kernel discriminant analysis in which the operations are organized into a multilayered feedforward network with four layers:
  - Input layer
  - Pattern layer
  - Summation layer
  - Output layer

Advantages and Disadvantages

- Advantages
  - Fast training process
    - Orders of magnitude faster than backpropagation
  - An inherently parallel structure
  - Guaranteed to converge to an optimal classifier as the size of the representative training set increases
    - No local minima issues
  - Training samples can be added or removed without extensive retraining
- Disadvantages
  - Not as general as backpropagation
  - Large memory requirements
  - Slow execution of the network
  - Requires a representative training set
    - Even more so than other types of NN's

Classification Theory

- If the probability density function (pdf) of each of the populations is known, then an unknown, X, belongs to class "i" if:

  $f_i(X) > f_j(X)$, all $j \neq i$ $\quad$ f_k is the pdf for class k

- Other parameters may be included
  - Prior probability (h)
    - Probability of an unknown sample being drawn from a particular population
  - Misclassification cost (c)
    - Cost of incorrectly classifying an unknown
  - Classification decision becomes:

    $h_i c_i f_i(X) > h_j c_j f_j(X)$, all $j \neq i$
    (Bayes optimal decision rule)

PDF Estimation

- Estimate the pdf by using the samples of the populations (the training set)
- PDF for a single sample (in a population):

$$\frac{1}{\sigma}W\left(\frac{x-x_k}{\sigma}\right)$$

x = unknown (input)
x_k = "k^{th}" sample
W = weighting function
σ = smoothing parameter

- PDF for a single population:

$$\frac{1}{n\sigma}\sum_{k=1}^{n}W\left(\frac{x-x_k}{\sigma}\right)$$

(average of the pdf's for the "n" samples in the population)

(Parzen's pdf estimator)

- The estimated pdf approaches the true pdf as the training set size increases, as long as the true pdf is smooth

Weighting Function

- Provides a "sphere-of-influence"
  - Large values for small distances between the unknown and the training samples
  - Rapidly decreases to zero as the distance increases
- Commonly use Gaussian function
  - Behaves well and easily computed
  - **Not** related to any assumption about a normal distribution
- The estimated pdf becomes:

$$g(x) = \frac{1}{n\sigma} \sum_{k=1}^{n} e^{-\frac{(x-x_k)^2}{\sigma^2}}$$

Multivariate Inputs

- Input to the network is a vector
- PDF for a single sample (in a population):

$$\frac{1}{(2\pi)^{p/2}\sigma^p} e^{-\frac{\|X - X_k\|^2}{2\sigma^2}}$$

X = unknown (input)
X_k = "kth" sample
σ = smoothing parameter
p = length of vector

- PDF for a single population:

$$g_i(X) = \frac{1}{(2\pi)^{p/2}\sigma^p n_i} \sum_{k=1}^{n_i} e^{-\frac{\|X - X_{ik}\|^2}{2\sigma^2}}$$

(average of the pdf's for the "n_i" samples in the "ith"population)

- Classification criteria:
  $g_i(X) > g_j(X)$, all $j \neq i$

$$\therefore g_i(X) = \frac{1}{n_i} \sum_{k=1}^{n_i} e^{-\frac{\|X - X_{ik}\|^2}{\sigma^2}}$$

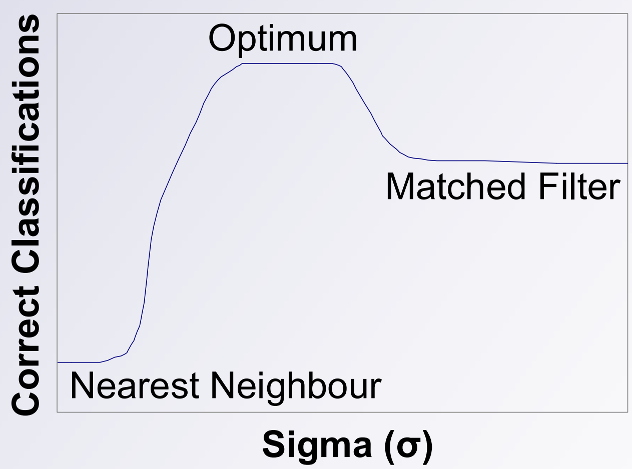
(eliminate common factors and absorb the "2" into σ)

Training Set

- The training set must be thoroughly representative of the actual population for effective classification
  - More demanding than most NN's
  - Sparse set sufficient
  - Erroneous samples and outliers tolerable
- Adding and removing training samples simply involves adding or removing "neurons" in the pattern layer
  - Minimal retraining required, if at all
- As the training set increases in size, the PNN asymptotically converges to the Bayes optimal classifier
  - The estimated pdf approaches the true pdf, assuming the true pdf is smooth

Training

- The training process of a PNN is essentially the act of determining the value of the smoothing parameter, sigma
  - Training is fast
    - Orders of magnitude faster than backpropagation



- Determining Sigma
  - Educated guess based on knowledge of the data
  - Estimate a value using a heuristic technique

Estimating Sigma Using Jackknifing

- Systematic testing of values for sigma over some range
  - Bounding the optimal value to some interval
  - Shrinking the interval
- Jackknifing is used to grade the performance of each "test" sigma
  - Exclude a single sample from the training set
  - Test if the PNN correctly classifies the excluded sample
  - Iterate the exclusion and testing process for each sample in the training set
    - The number of correct classifications over the entire process is a measure of the performance for that value of sigma
  - Not unbiased measure of performance
    - Training and testing sets not independent
    - Gives a ball park estimate of quality of sigma

Implementations

- Current Work
  - Basic PNN coded in Java
    - Simple examples
      - Boy/Girl classifier (same as perceptron)
      - Classification of points in **R**^{2} or **R**^{3} into the quadrants
  - Multithreaded PNN
    - For parallel processing (on supercomputers)
    - One thread per class
- Future Work
  - Artificially create a time series of a chaotic system and use a PNN to classify its features in order to reconstruct the strange attractor
    - Further test the classification abilities of PNN
    - Test the PNN's tolerance to noisy inputs

Conclusion

- PNN's should be used if
  - A near optimal classifier with a short training time is desired
  - Slow execution speed and large memory requirements can be tolerated
- No extensive testing on our implementation of PNN's have been done
  - Once chaotic time series have been obtained, we will have more challenging data to work with

References

[Mast93] T. Masters, *Practial Neural Network Recipes in C++*, Toronto, ON: Academic Press, Inc., 1993.

[Specht88] D.F. Specht, “Probabilistic Neural Networks for Classification, Mapping, or Associative Memory”, *IEEE International Conference on Neural Networks*, vol. I, pp. 525-532, July 1998.

[Specht92] D.F. Specht, “Enhancements to Probabilistic Neural Networks”, *International Joint Conference on Neural Networks*, vol. I, pp. 761-768, June 1992.

[Wass93] P. D. Wasserman, *Advanced Methods in Neural Computing*, New York, NY: Van Nostrand Reinhold, 1993.

[Zak98] Anthony Zaknich, *Artificial Neural Networks: An Introductory Course*. [Online]. http://www.maths.uwa.edu.au/~rkealley/ann_all/ann_all.html (as of June 6, 2002).

Simple Classifier Example

- Idea behind classification using a PNN
- Three classes or populations
  - X, O, and □
- The “?” is an unknown sample and must be classified into one of the populations
- Nearest neighbour algorithm would classify the “?” as a □ because a □ sample is the closest sample to the “?”
  - In other words, with nearest neighbour, the unknown belongs to the same population as the closest sample

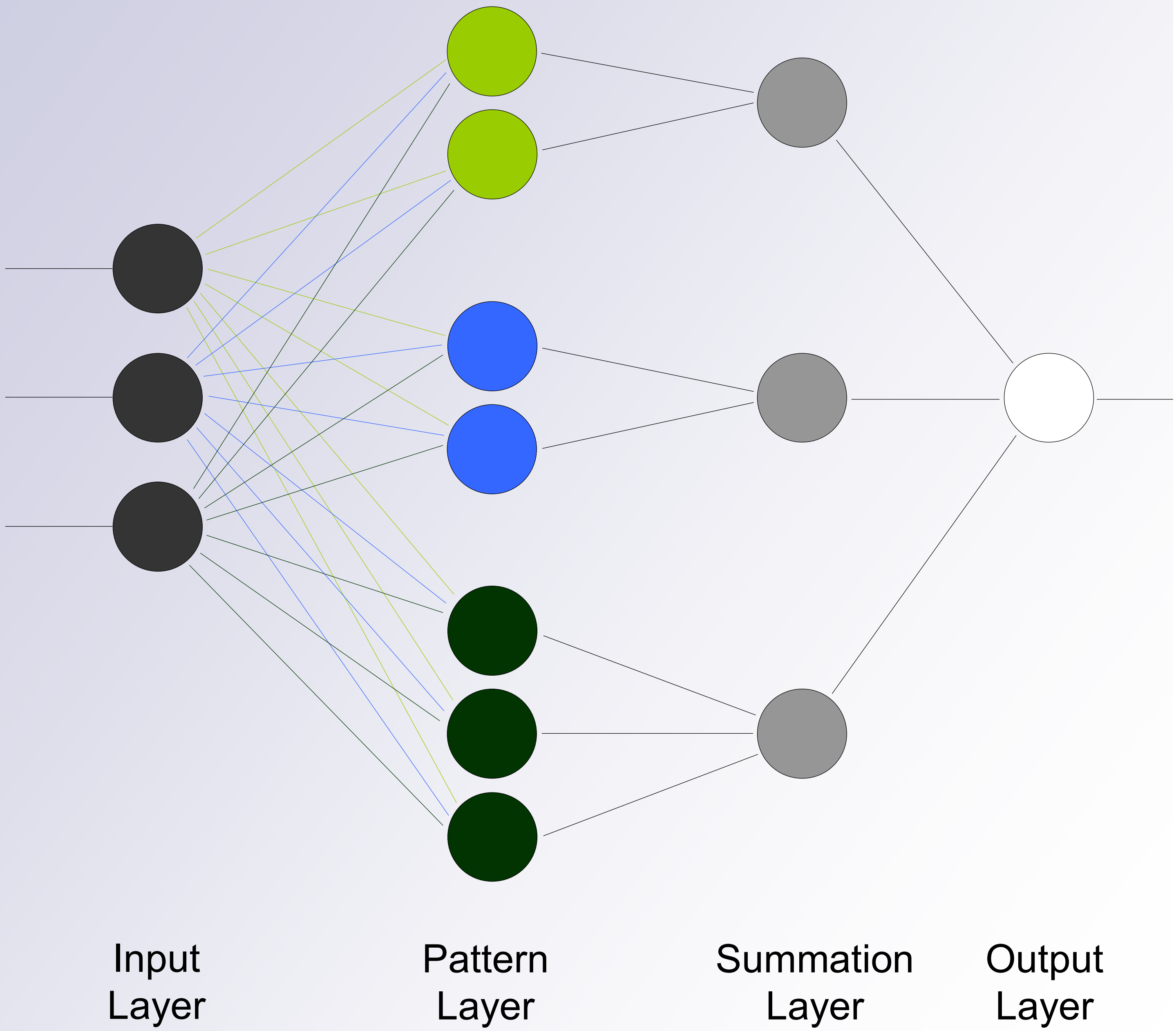
Simple Classifier Example

- A more effective classifier would also take the other samples into consideration in making its decision
- However, not all samples should contribute to the classification of a particular unknown the same amount
  - Samples close to the unknown should have a large contribution (increase the probability of classifying the unknown as that population)
  - Samples far from the unknown should have a small contribution (decrease the probability of classifying the unknown as that population)
  - A “sphere-of-influence”

Simple Classifier Example

- What the more effective classifier would then do is, for each population, calculate the average of all the contributions made by the samples in that population
- The unknown sample is then classified as being a member of the population which has the largest average

Architecture



Architecture

$$g_i(X) = \frac{1}{n_i}\sum_{k=1}^{n_i} e^{-\frac{\|X - X_{ik}\|^2}{\sigma^2}}$$