Machine intelligence

Dr. Ahmad Al-Mahasneh

5th lecture

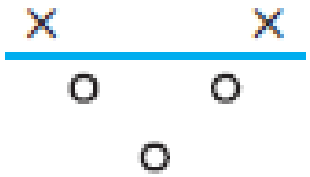
Review of the last lecture

- Activation functions.
- Feed Forward Neural Networks (FFNNs)
- Radial Basis Functions Neural Network (RBFNNs)

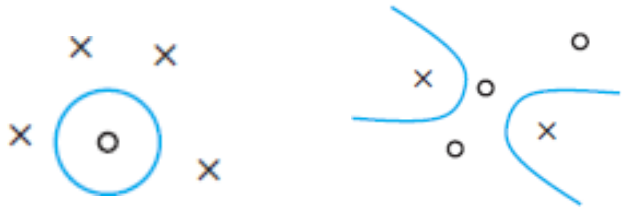
Outline

- Radial Basis Functions Neural Networks (RBFNN)
- Probabilistic Neural Network (PNN)
- Hamming Distance network

Linear vs. nonlinear separation



Linear Separation



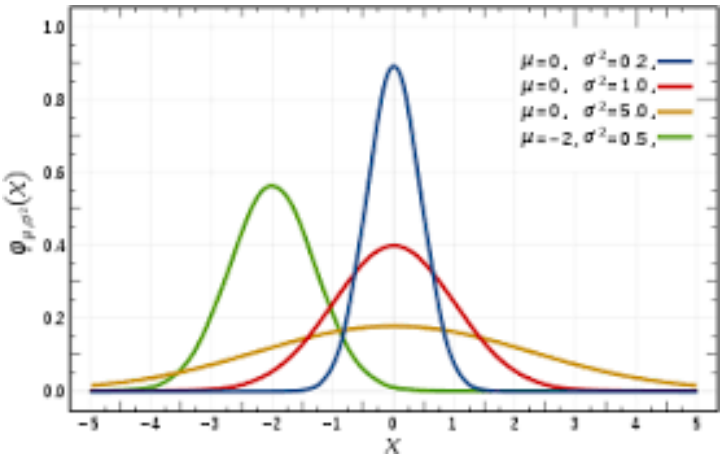
Nonlinear Separation

XOR is linearly separable ?

Gaussian Functions

In probability theory, a normal distribution is a continuous probability distribution for a real- valued random variable.

The general form of its probability density **function** is ... A random variable with a **Gaussian** distribution is said to be normally distributed and is called a normal deviate.



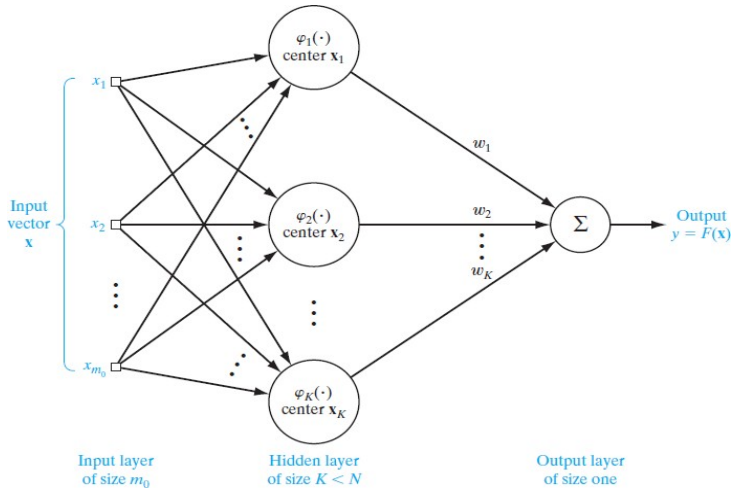
$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Radial Basis Functions

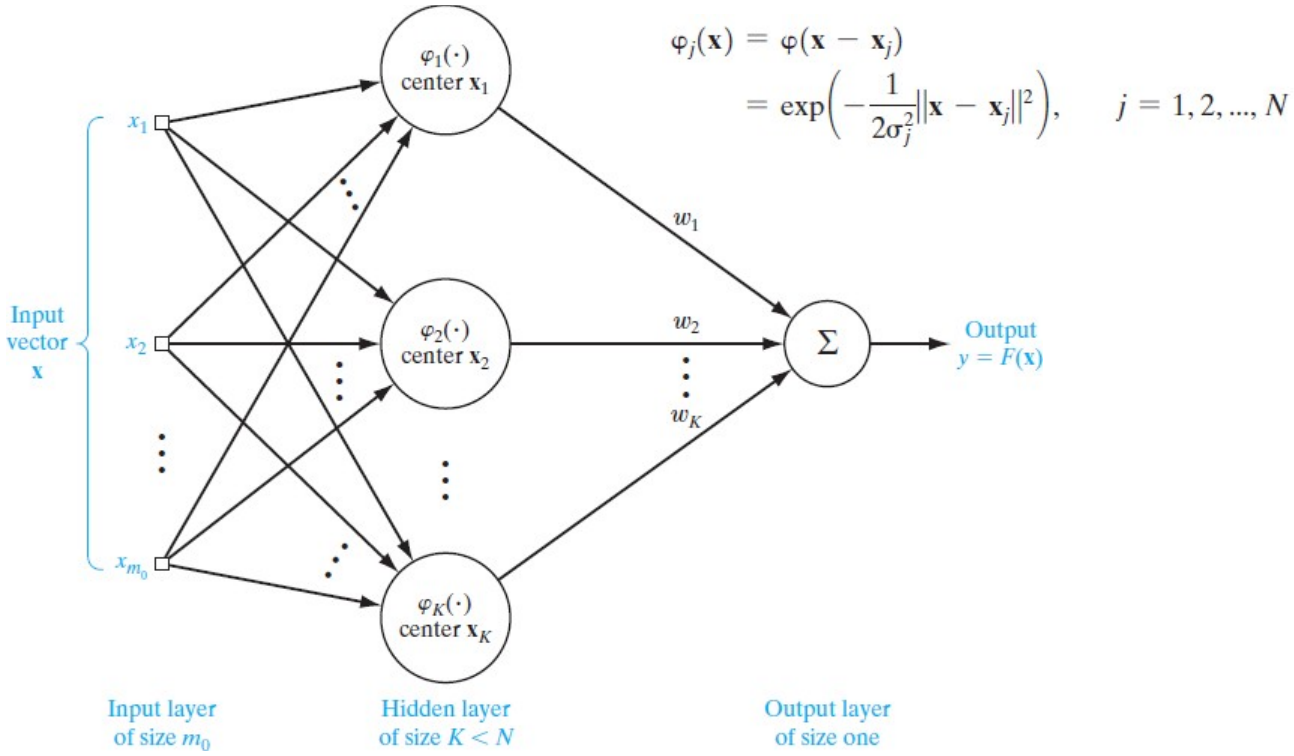
- RBF Functions use Gaussian-like functions
- RBF network can be used to perform complex pattern classification task and regression tasks also.
- Pattern Classification for nonlinear separation can be performed in two stages:
  - **Transform** nonlinear patterns into new linearly separated space
  - **Separate** the data using least-squares estimation

RBF Network

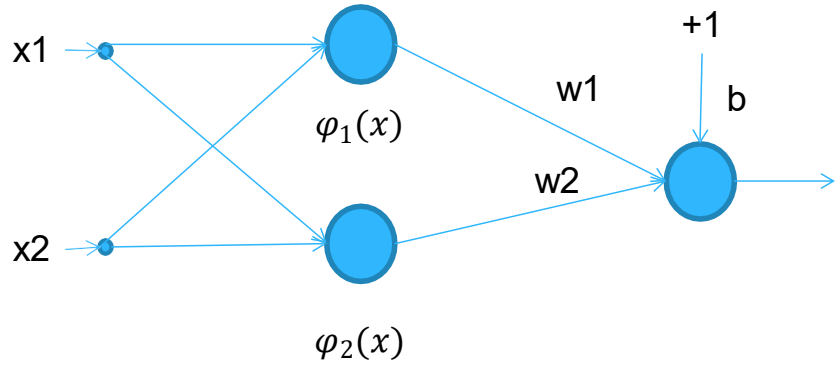
- RBF Network is composed of **three** layers
  - **Input layer** is made up of sensor inputs
  - **Hidden layer** applies nonlinear transformation from input space to hidden space
  - **Output layer** is linear



RBF Network Structure



Simple RBF Network

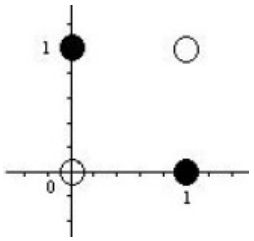


$$\varphi_i(x) = exp\left(-\frac{\|x - c_i\|^2}{2\sigma^2}\right)$$

$$y = w_1\varphi_1(x) + w_2\varphi_2(\mathrm{x}) + \mathrm{b}$$

XOR Problem Revisited

x2	x1	y
0	0	0
0	1	1
1	0	1
1	1	0



Linearly unseparable

$\varphi_1(x) = exp(- \|x - c_1\|^2)$ with centers *c1* = (1,1)

$\varphi_2(x) = exp(- \|x - c_2\|^2)$ with center *c2* = (0,0)

Pattern x	φ_1	φ_2
(0,0)	0.1353	1
(0,1)	0.3678	0.3678
(1,0)	0.3678	0.3678
(1,1)	1	0.1353

XOR Problem Revisited

Let w1 = w2 = -2.5, b = 2.84
c1=(1,1), c2=(0,0)

$$\varphi_i(x) = exp\left(-\frac{\|x - c_i\|^2}{2(0.5)}\right)$$

$$y = -2.5\ \varphi_1(x) - 2.5\ \varphi_2(\mathrm{x}) + 2.84$$

Input x=(0,1) or x=(1,0)

$$d_1 = (0-1)^2 + (1-1)^2 = 1$$

$$\varphi 1 \quad = e^{-1} = 0.3678$$

$$d_2 = (0-0)^2 + (1-0)^2 = 1$$

$$\varphi_2 = e^{-1} = 0.3678$$

y = 2.84 - 2.5(0.3679) - 2.5(0.367) = **1**

Input x=(0,0) or x=(1,1)

$$d_1 = (0-1)^2 + (0-1)^2 = 2$$

$$\varphi_1\ x \quad = e^{-2} = 0.1353$$

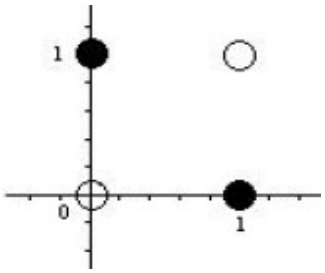
$$d_2 = ((0-0)^2 + (0-0)^2 = 0$$

$$\varphi_2\ x \quad = e^{0} = 1$$

y = 2.84 – 2.5(1) -2.5(0.1353) = **0**

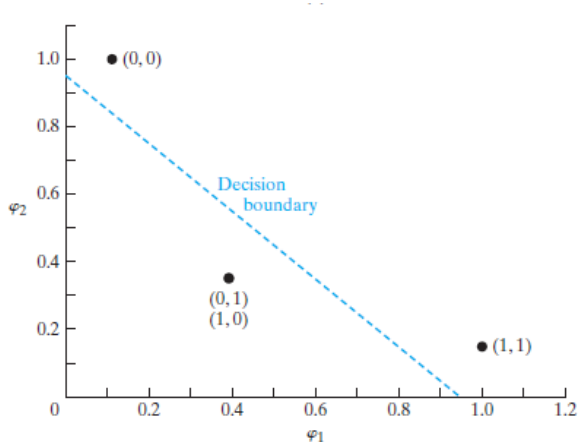
XOR Problem Re-visted

x2	x1	y
0	0	0
0	1	1
1	0	1
1	1	0



Transformation from input space to hidden space

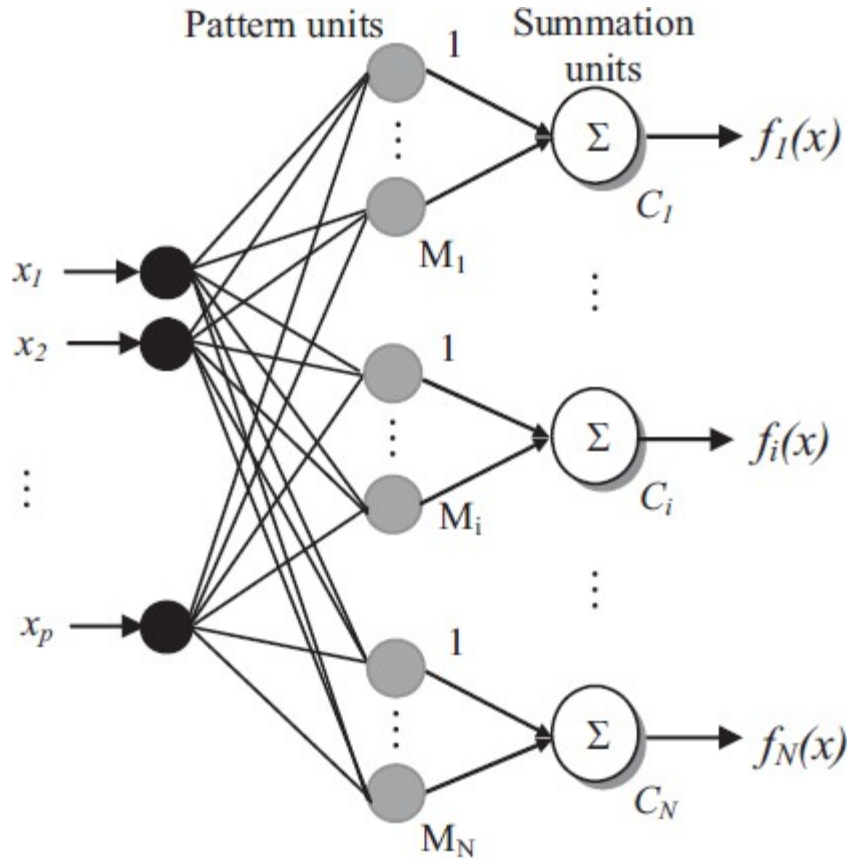
Pattern x	φ_1	φ_2
(0,0)	0.1353	1
(0,1)	0.3678	0.3678
(1,0)	0.3678	0.3678
(1,1)	1	0.1353



Probabilistic Neural Network (PNN)

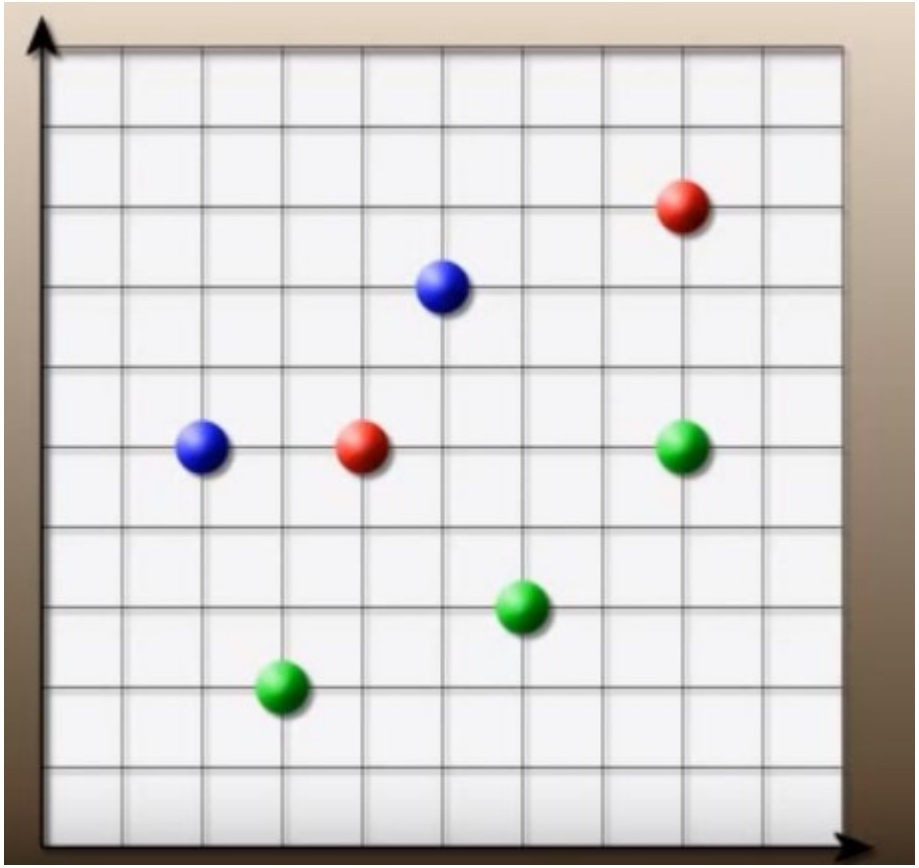
- PNN is a feedforward network based on probability theory
- PNN use probability density functions (PDF)
- PNN uses sums of Gaussian functions centered at the training vector patterns
- PNN is composed of three layers
  - Input layer
  - Pattern layer that uses Gaussian function
  - Output layer that uses linear summation

PNN Architecture



$$f_i(x) = \frac{1}{(2\pi)^{\frac{p}{2}}\sigma^p}\frac{1}{M}\sum_{j=1}^{M}\exp\left[\frac{-(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}\right]$$

PNN Example



- **Class I**: (0.5, 0.7)
  (0.2, 0.5)
- **Class II**: (0.8, 0.8)
  (0.4, 0.5)
- **Class III**: (0.8, 0.5)
  (0.6, 0.3)
  (0.3, 0.2)

PNN Example

$$f_i(x) = \frac{1}{(2\pi)^{\frac{p}{2}}\sigma^p}\frac{1}{M}\sum_{j=1}^{M}\exp\left[\frac{-(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}\right]$$

- **Class I**: (0.5, 0.7)
  (0.2, 0.5)

$$f_1(x_1,x_2) = \frac{1}{2}\,e^{-\left(\frac{((x_1-0.5)^2+(x_2-0.7)^2)}{2*0.1^2}\right)} + \frac{1}{2}\mathrm{e}^{-\left(\frac{\left((x_1-0.2)^2+(\mathrm{x}_2-0.5)^2\right)}{2*0.1^2}\right)}$$

$$f_1(x_1,x_2) = \left(\frac{1}{2\,\pi\,0.1^2}\right)\frac{1}{2}\,e^{-\left(\frac{((x_1-0.5)^2+(x_2-0.7)^2)}{2*0.1^2}\right)} + \frac{1}{2}\mathrm{e}^{-\left(\frac{\left((x_1-0.2)^2+(\mathrm{x}_2-0.5)^2\right)}{2*0.1^2}\right)}$$

PNN Example

$$f_i(x) = \frac{1}{(2\pi)^{\frac{p}{2}}\sigma^p}\frac{1}{M}\sum_{j=1}^{M}\exp\left[\frac{-(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}\right]$$

- **Class II**: (0.8, 0.8)
  (0.4, 0.5)

$$f_2(x_1, x_2) = \frac{1}{2}\, e^{-\left(\frac{((x_1-0.8)^2+(x_2-0.8)^2)}{2*0.1^2}\right)} + \frac{1}{2}\mathrm{e}^{-\left(\frac{\left((x_1-0.4)^2+(\mathrm{x}_2-0.5)^2\right)}{2*0.1^2}\right)}$$

PNN Example

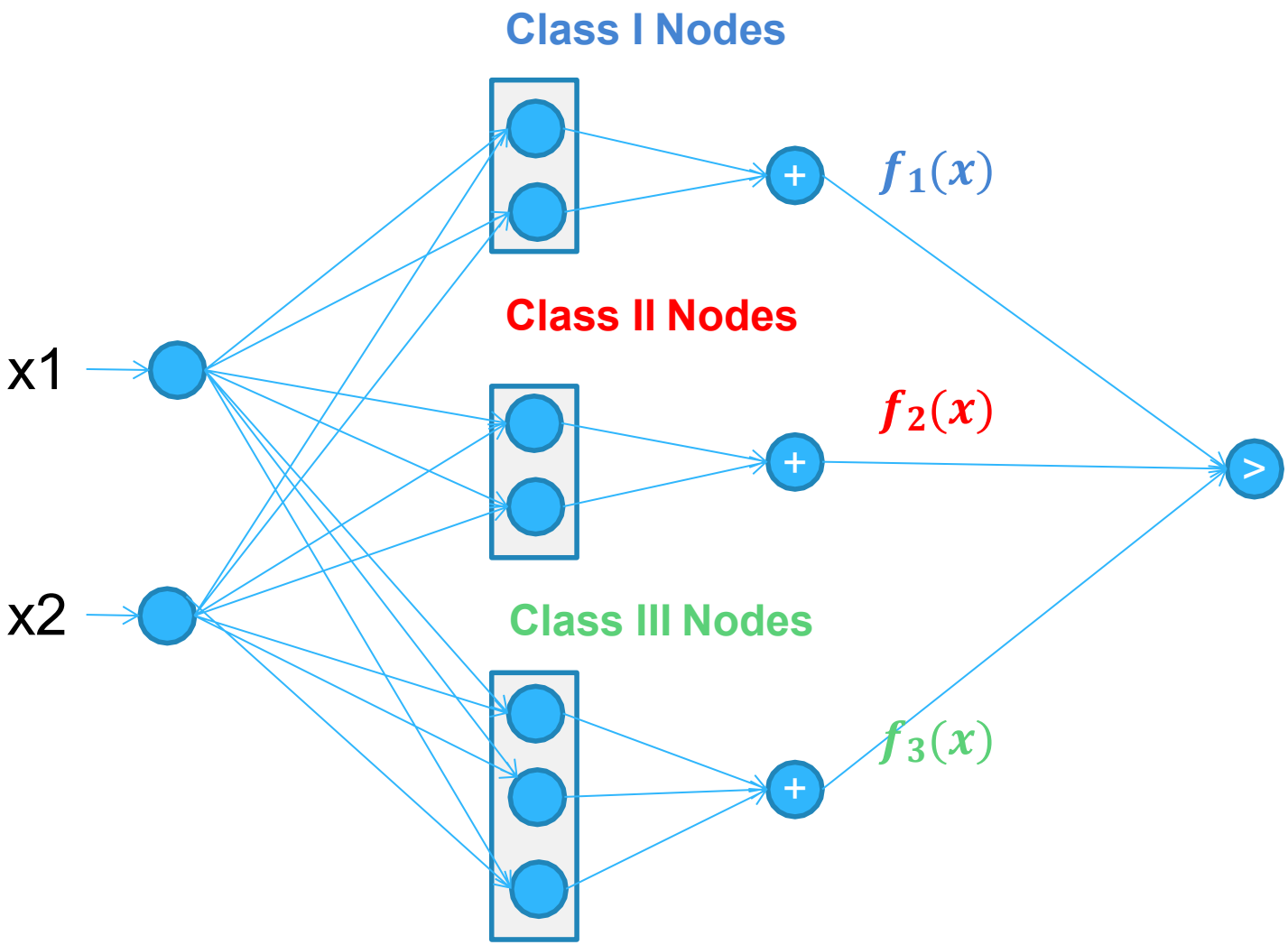
$$f_i(x) = \frac{1}{(2\pi)^{\frac{p}{2}}\sigma^p}\frac{1}{M}\sum_{j=1}^{M}\exp\left[\frac{-(x-x_{ij})^T(x-x_{ij})}{2\sigma^2}\right]$$

- **Class III**: (0.8, 0.5)
  (0.6, 0.3)
  (0.3, 0.2)

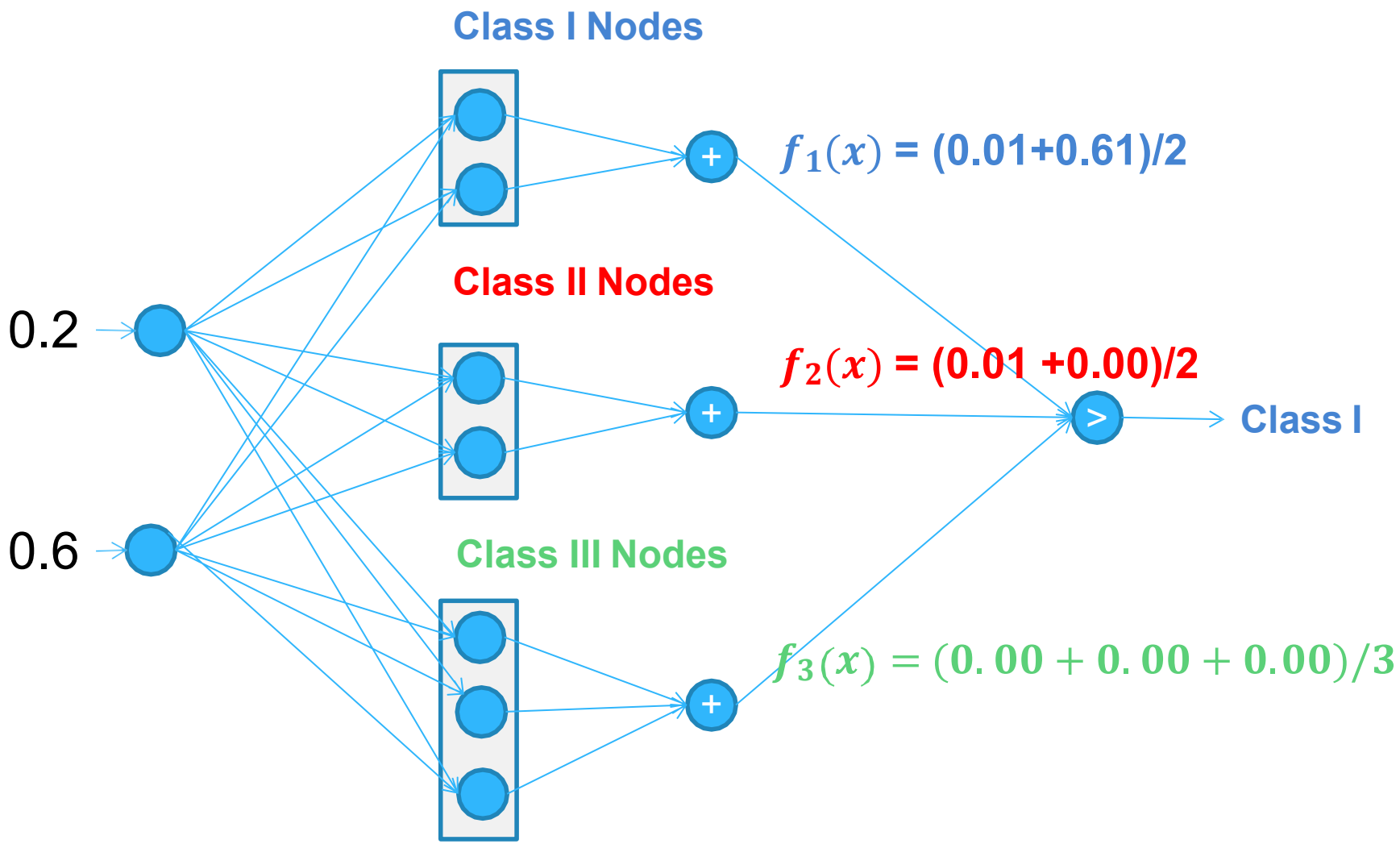
$$f_3(x1,x2) = \frac{1}{2}\,e^{-\left(\frac{((x_1-0.8)^2+(x_2-0.5)^2)}{2*0.1^2}\right)} + \frac{1}{2}\,e^{-\left(\frac{\left((x_1-0.6)^2+(x_2-0.3)^2\right)}{2*0.1^2}\right)} + \frac{1}{2}e^{-\left(\frac{\left((x_1-0.3)^2+(x_2-0.2)^2\right)}{2*0.1^2}\right)}$$

PNN Example

- **Class I**: (0.5, 0.7), (0.2, 0.5)
- **Class II**: (0.8, 0.8), (0.4, 0.5)
- **Class III**: (0.8, 0.5), (0.6, 0.3), (0.3, 0.2)



PNN Example


Class I Nodes
Class II Nodes
Class III Nodes
0.2
0.6
+
+
+
>
$f_1(x)$ = (0.01+0.61)/2
$f_2(x)$ = (0.01 +0.00)/2
$f_3(x) = (0.00 + 0.00 + 0.00)/3$
Class I

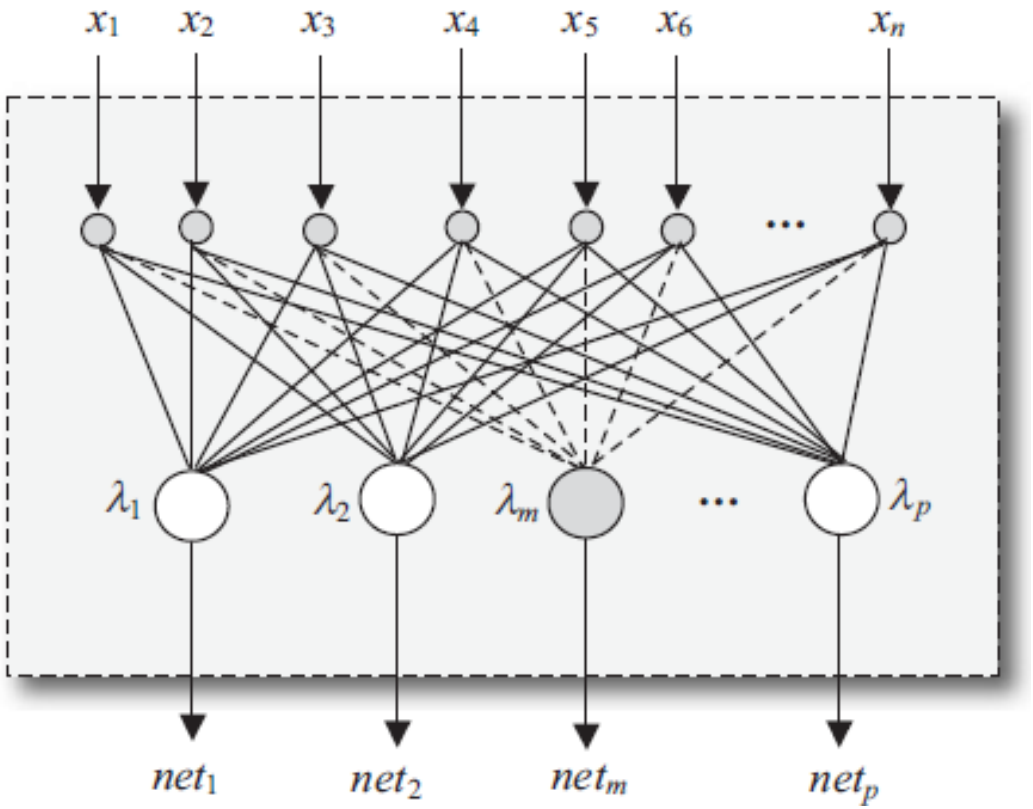
Hamming Network

- Hamming Network is a **two-layer feedforward** network
  - The **first layer** is the input layer for *n-tuple* (± 1) input vectors
  - The **second layer** stores *p* memory patterns

- Hamming Network is used for classification using minimum Hamming distance, $D_H(x, \lambda_m) = \frac{1}{2} x^T \lambda_m - \frac{n}{2}$

- The distance between the ***input*** vector ***x*** and the memory ***pattern*** λ_m stored in the network and selects the memory with the smallest Hamming distance

Hamming Network Structure



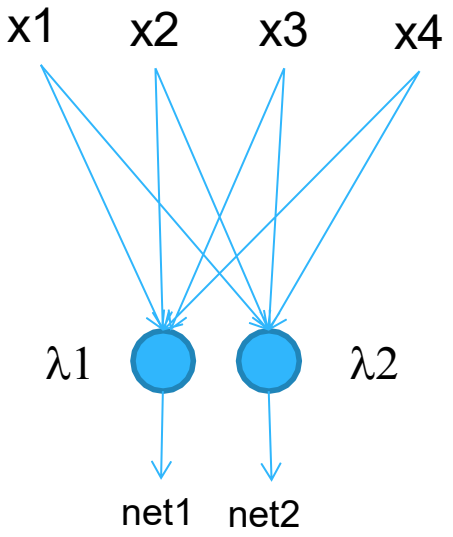
Hamming Network Example

Pattern 1 λ_1 =[1 1 1 1]

Pattern 2 λ_2 = [-1 -1 -1 -1]

For any input, calculate the Hamming distance and select the smallest:

$$net_m = \frac{1}{2} x^T \lambda_m - \frac{n}{2}$$



Let input be x = [1 1 1 1]

$$net_1 = \frac{1}{2}[1\ 1\ 1\ 1]^T[1\ 1\ 1\ 1] - \frac{4}{2} = 0$$

$$net_2 = \frac{1}{2}[1\ 1\ 1\ 1]^T[-1\ -1\ -1\ -1] - \frac{4}{2} = -4$$

| net_1 | < | net_2 | => x input provides pattern λ_1

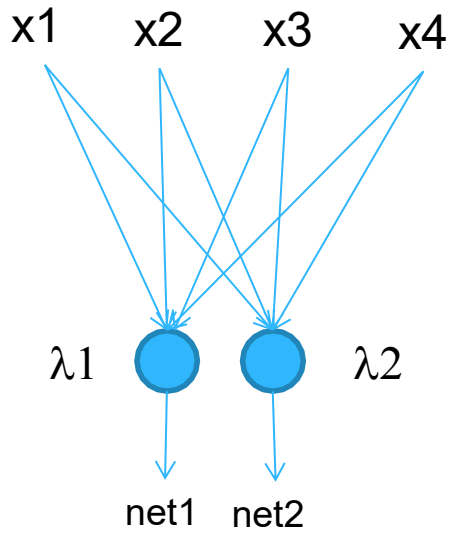
Hamming Network Example

Pattern 1 λ_1 =[1 1 1 1]

Pattern 2 λ_2 = [-1 -1 -1 -1]

$$net_m = \frac{1}{2} x^T \lambda_m - \frac{n}{2}$$

Let input be x = [-1 -1 -1 -1]



$$net_1 = \frac{1}{2}[-1\ -1\ -1\ -1]^T[1\ 1\ 1\ 1] - \frac{4}{2} = -4$$

$$net_2 = \frac{1}{2}[-1\ -1\ -1\ -1]^T[-1\ -1\ -1\ -1] - \frac{4}{2} = 0$$

$| net_2 | < | net_1 |$ => x input provides pattern λ_2

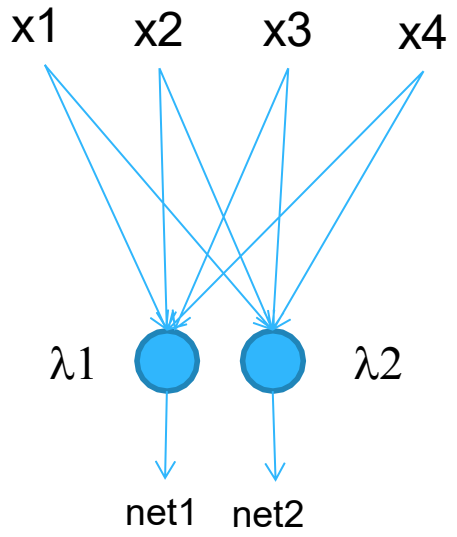
Hamming Network Example

Pattern 1 λ_1 =[1 1 1 1]

Pattern 2 λ_2 = [-1 -1 -1 -1]

$$net_m = \frac{1}{2} x^T \lambda_m - \frac{n}{2}$$

Let input be x = [-1 1 1 1]



$$net_1 = \frac{1}{2}[-1 \;\; 1 \; 1 \; 1]^T [1 \; 1 \; 1 \; 1] - \frac{4}{2} = -1$$

$$net_2 = \frac{1}{2}[-1 \; 1 \; 1 \; 1]^T [-1 \; -1 \; -1 \; -1] - \frac{4}{2} = -3$$

| net1 | < | net2 | => x input provides pattern λ1

Conclusions

- A Gaussian function is a normally distributed function that is used in probability theory.

- Radial Basis Functions use Gaussian-like functions

- RBF Network is composed of three layers: input, hidden, and output
- The input layer is the input signals
- hidden layer transforms the input-space to a hidden-space
- The output layer uses a linear activation function

Conclusion

- PNN is a network based on probability theory and statistical principles
- PNN is composed on an input layer, pattern (hidden) layer, and output layer.
- Hamming network is a two-layer feedforward network: The first layer is the input layer, and The second layer is the memory layer.

Conclusion

- Hamming network is used to store *p* patterns
- The Hamming distance is calculated between the input vector and stored patterns.
- The minimum distance provides the appropriate classification for the inputs

References

- Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks, and Evolutionary Computing (Chapter 4), by N. Siddique and H. Adeli. Wiley Publication 2013

- Neural Networks and Learning Machine (Chapter 5) by Simon Haykin 3rd Edition. Pearson 2009