هوش محاسباتی:
شبکه های عصبی مصنوعی

mozayani@iust.ac.ir

RBF

Radial Basis Functions

Introduction

- BP & MLP can be viewed as optimization under supervision
- Generalization means interpolation of new points
- Problem of function approximation or curve fitting

➔ RBF is another solution for this problem

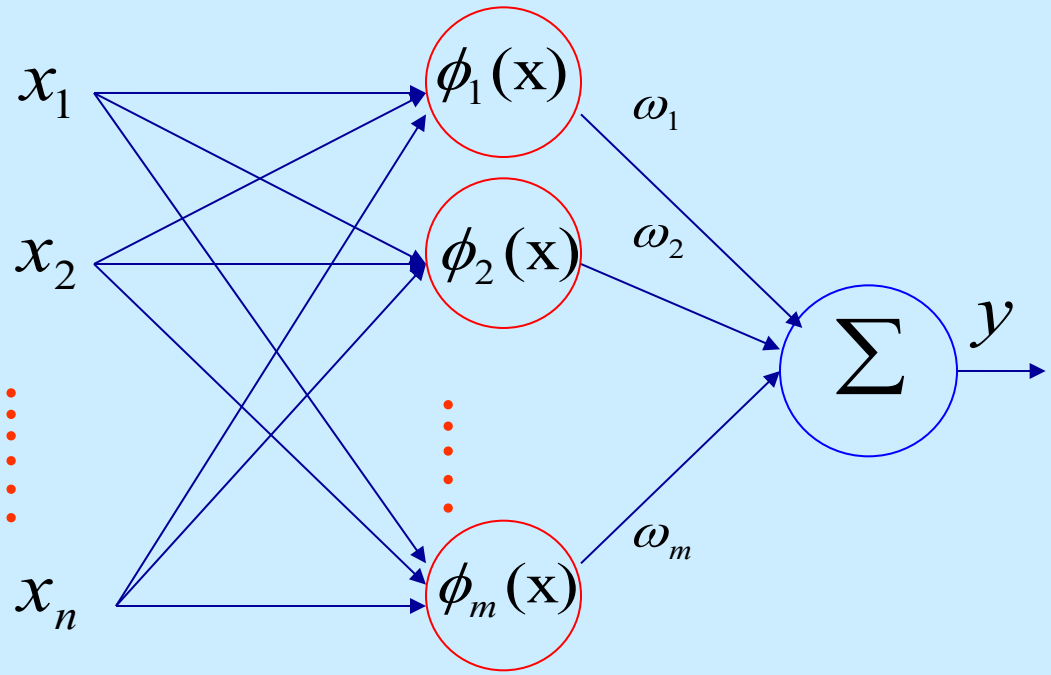
RBF's history

-first introduced in interpolation problem in numerical analysis (Powell 1985)

-using RBF in ANN (Broomhead & Lowe 1988, Moody & Darken 1988,...)

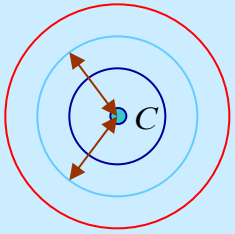
-some basis functions are defined for input patterns

Architecture



$$y = \sum_{i=1}^{m} \omega_i \phi_i(\mathrm{x})$$

-Weights of the input layer are all “1”,
-All basis functions are “radial”:



RBF (cont'd)

-Each hidden neuron in an RBF is tuned to respond to a rather local region of feature space by means of a radially symmetric function

-Mapping:

- -nonlinear mapping form input to hidden neurons
- -linear mapping from hidden neurons to outputs

Cover's Theorem

Cover's Theorem(Cover, 1965):

A complex pattern classification problem cast in a high dimensional space nonlinearly is more likely to be linearly separable than in a low dimensional space

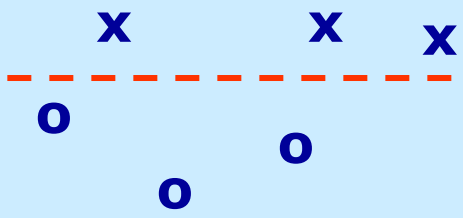
Cover's Theorem in Practice:

- A non-linear hidden layer
- Large dimensionality for the hidden layer compared to the input space

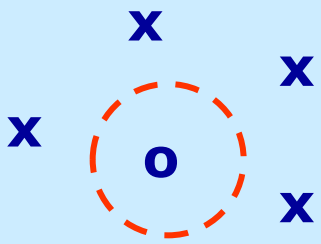
Separability

Functional Separability:

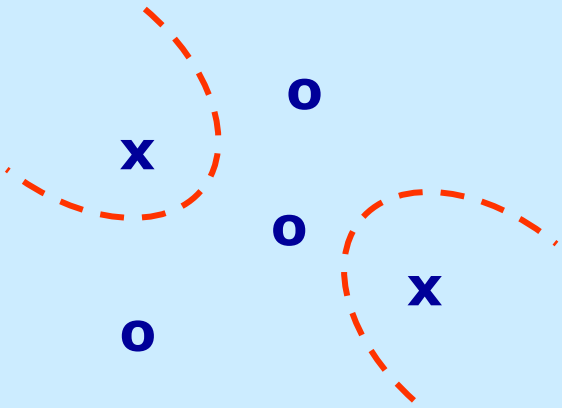
- Linearly Separable



- Spherically Separable



- Quadrically Separable



Polynomial separability is a natural extension of linear separability

Separability (cont'd)

$X^1, X^2, \ldots, X^P$ assigned to two classes C^1 C^2

φ–separability:

$$\exists W \Big| \begin{cases} W^T \varphi(x) > 0 & x \in C^1 \\ W^T \varphi(x) < 0 & x \in C^2 \end{cases}$$

Cover's Corollary:

The expected maximum number randomly assigned patterns that are linearly separable in an input space is double the dimensionality of that space

By displacing center of radial functions, we may reduce the dimensionality of hidden neurons

Activation

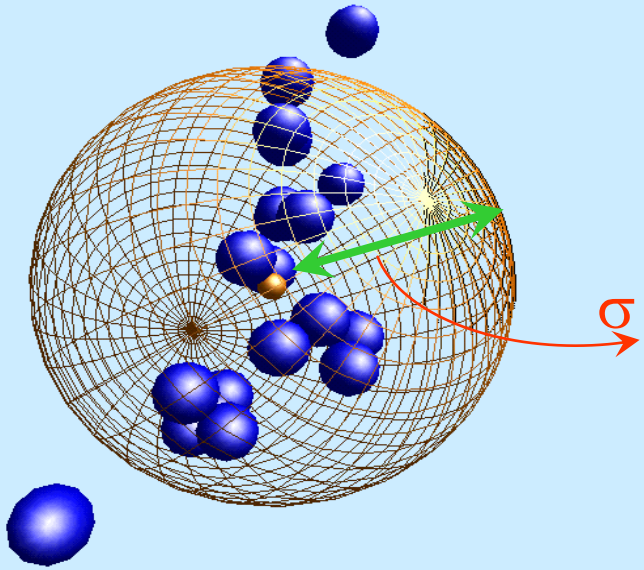
Activation of a hidden unit is determined by the DISTANCE between the input vector *x* and a prototype vector t

$$\varphi_j(x)=f(|x- t_j|)$$

-x is a multidimensional input vector

-t_j is a vector with same dimension as x

-$\varphi_j(.)$ is the j^{th} radial basis function with a single maximum at the origin



Activation (cont'd)

Choice of radial basis:

Although several forms of radial basis may be used (uniform, etc.), Gaussian kernels are most commonly used:

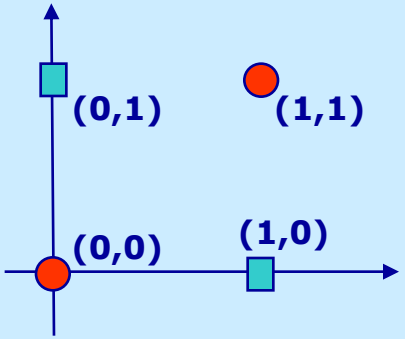
$$\phi_i(\mathrm{x}) = \exp\left(-\frac{\|\mathrm{x} - t_i\|^2}{\sigma_i^2}\right)$$

Coordinates of center: t_i

Width parameter: σ_i

Example : XOR

We need a classifier to respond 1 for (1,0) and (0,1) samples and respond 0 for (1,1) and (0,0).



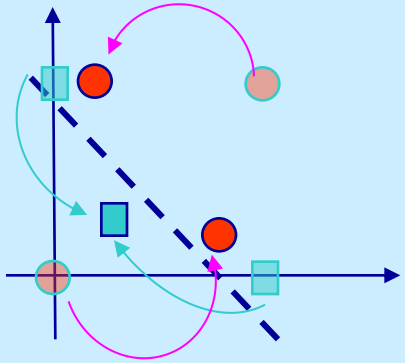
We define:

$$\phi_1(\mathrm{x}) = \exp\left(-\|\mathrm{x}-t_1\|^2\right), \qquad t_1 = [1,1]$$

$$\phi_2(\mathrm{x}) = \exp\left(-\|\mathrm{x}-t_2\|^2\right), \qquad t_2 = [0,0]$$

σ is considered 1 in this example.
We obtain:

Input x	$\Phi_1(x)$	$\Phi_2(x)$
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678



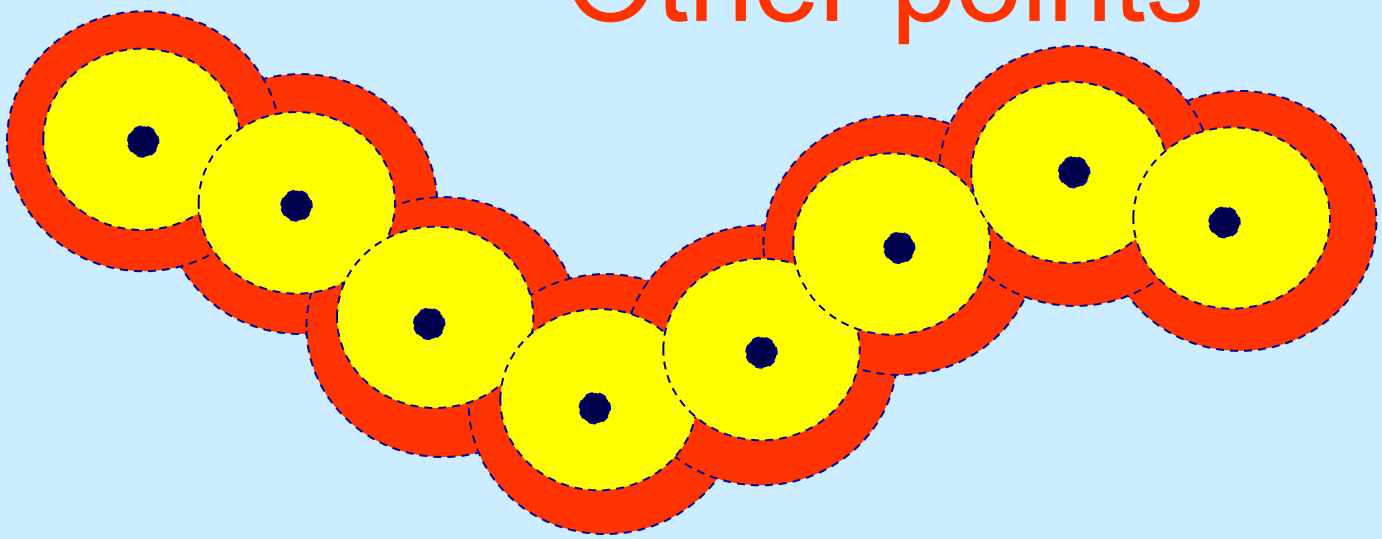
Other points

- Learning is finding surface in multidimensional space best fit to training data
- Approximate function with linear combination of Radial basis functions

$$F(x) = \Sigma \; w_i \, G(||x - t_j||) \quad i = 1, 2, \ldots, N$$

- $G(||x - t_j||)$ is also called Green function
- When N= number of samples, we call it regularization network
- When N < number of samples, Radial-basis function network
- It can be shown that we need at least: N=1/2 no of samples

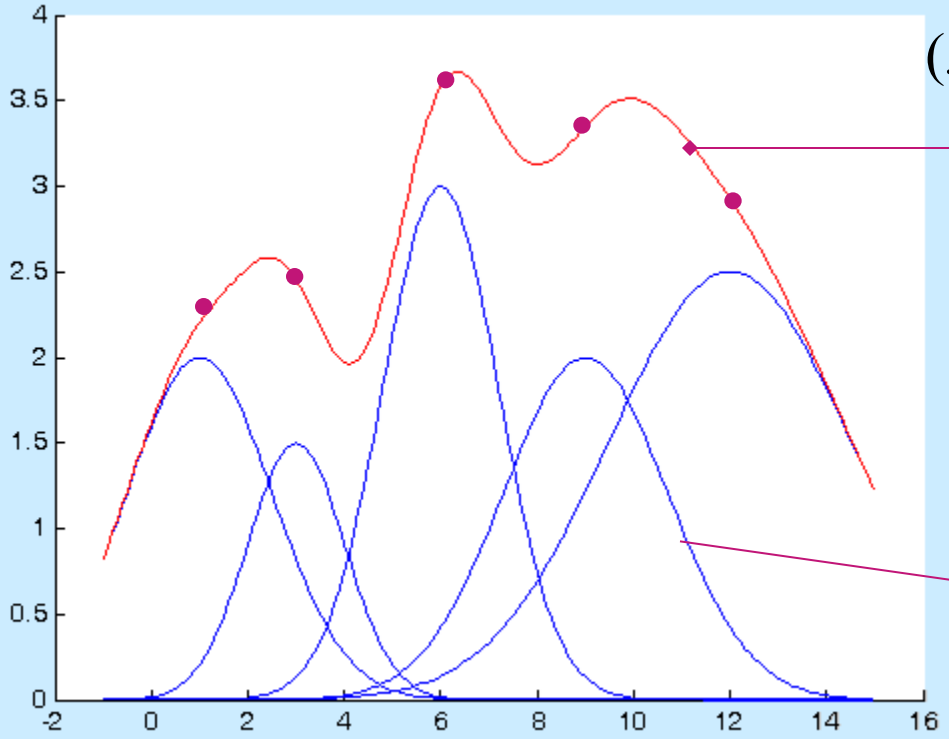
Other points



- The degree of generalization is controlled by σ
- It requires holding out a validation sample,
- training networks with various values of σ and measuring the validation error of each network to see which generalizes best
- Make number of parameters small as possible
  - Principles of Dimensionality

representation

A 1-D graphical representation of Kernel Smoothing with the Spherical Gaussian Functions



$$(x) = \sum_{i=1}^{N} w_i . \exp\left(-\frac{\left\| \mathrm{x} - t_i \right\|^2}{\sigma_i^{\,2}} \right) \cong F(x)$$

$$w_i . \exp\left(-\frac{\left\| \mathrm{x} - t_i \right\|^2}{\sigma_i^{\,2}} \right)$$

Learning Strategies

- Two levels of Learning
  - Center and spread learning (or determination)
  - Output layer Weights Learning

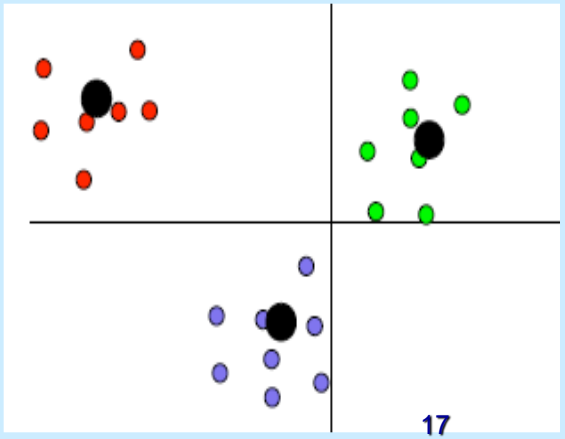
- Theoretically we can obtain output weights by:

$$\vec{\phi}.\vec{W} = \vec{d} \qquad \Rightarrow W = \phi^{-1} d$$

- But in practice, it is usually a difficult task

Selection of centers

- Fixed Centers
  - A quick solution is to place one of the hidden units at the location of every J^{th} example (randomly selected), where J is the number of hidden units
  - Another approach is to add nodes successively, until the approximation is good
  - In the limit, this might be one node per training pattern
- Self-organizing selection of Centers
  - clustering first : k-nearest neighbors



Selection of spreads

- Spreads
  - One way is to set the width parameter for all units to the same value , which is the average of the distances from each center to its nearest neighbor
- Supervised selection of Centers with weights
  - Error correction learning

RBF supervised learning

- Linear weights (output layer)

$$\frac{\partial \mathrm{E}(n)}{\partial w_i(n)} = \sum_{j=1}^{N} e_j(n) G(\| \mathbf{x}_j - \mathbf{t}_i(n) \|_{C_i})$$

$$w_i(n+1) = w_i(n) - \eta_1 \frac{\partial E(n)}{\partial w_i(n)}, \qquad i = 1,2,\ldots, M$$

- Positions of centers (hidden layer)

$$\frac{\partial E(n)}{\partial \mathbf{t}_i(n)} = 2 w_i(n) \sum_{j=1}^{N} e_j(n) G'(\| \mathbf{x}_j - \mathbf{t}_i(n) \|_{C_i}) \Sigma_i^{-1} [\mathbf{x}_j - \mathbf{t}_i(n)]$$

$$\mathbf{t}_i(n+1) = \mathbf{t}_i(n) - \eta_2 \frac{\partial E(n)}{\partial \mathbf{t}_i(n)}, \qquad i = 1,2,\ldots, M$$

- Spreads of centers (hidden layer)

$$\frac{\partial E(n)}{\partial \Sigma_i^{-1}(n)} = -w_i(n) \sum_{j=1}^{N} e_j(n) G'(\| \mathbf{x}_j - \mathbf{t}_i(n) \|_{C_i}) \mathbf{Q}_{ji}(n)$$

$$\mathbf{Q}_{ji}(n) = [\mathbf{x}_j - \mathbf{t}_i(n)][\mathbf{x}_j - \mathbf{t}_i(n)]^T$$

$$\Sigma_i^{-1}(n+1) = \Sigma_i^{-1}(n) - \eta_3 \frac{\partial E(n)}{\partial \Sigma_i^{-1}(n)}$$

Training with Noise

- Noise in the training set can be good; it can make the resulting network, which has learned to "average" noise in, more **robust**

- However, with too many neurons, a network can **over-train** to "learn the noise"

Regularization or Weight Decay

- **Weight Decay** method can be used:
- **prune** an RBF:
  - At each update, a small amount is **deducted** from each weight
  - Weights that are constantly being updated will end up with a non-0 value, while others will go to 0 and can be **eliminated**
  - The resulting network is less trained to the noise
- Weight Decay is also known as "**regularization**"

MLP vs RBF

- Network structure
  - MLP : shares common neuron model
  - RBF : output and hidden layer different function
- Hidden node computation
  - RBF : *Euclidean norm*
  - MLP : *inner product*
- Approximation to non-linear input-output mapping
  - MLP : *Global*
  - RBF : *Local*
    - *fast, insensitive of order of sample data*
    - *need large data to get smooth surface*