Introduction to Artificial Neural Networks

Lecture 7:

Radial-Basis Function (RBF)

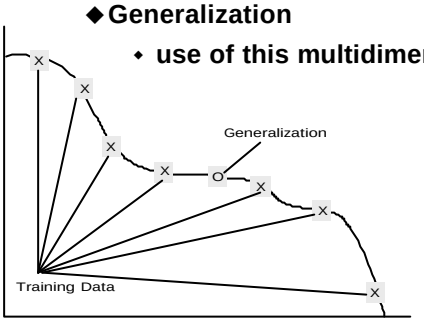
By: Ali Motie Nasrabadi

Outline

- Introduction
- Structure of RBF Networks
- Cover's Theorem on Separability
- Types of RBF function
- Regularization Networks
- Generalized Radial-Basis Function Networks
- Learning Strategies
  - Fixed centers selected at random
  - Self-organized selection of centers
  - Supervised selection of centers
- Problems of RBFs
- Comparison of RBF Networks with MLPs
- Real World Application – EEG Analysis
- MATLAB Toolbox

Introduction

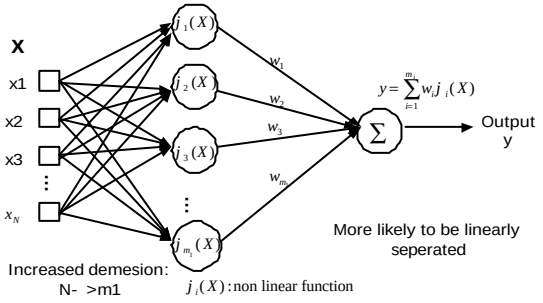
- **What is RBF networks?**
  - **A kind of supervised neural networks**
  - **Design of Neural Networks as curve fitting (approximation) problem**
  - **Learning**
    - **Find a surface that best fits to given training data**
  - **Generalization**
    - **use of this multidimensional surface to interpolate test data**



Lecture 7-3

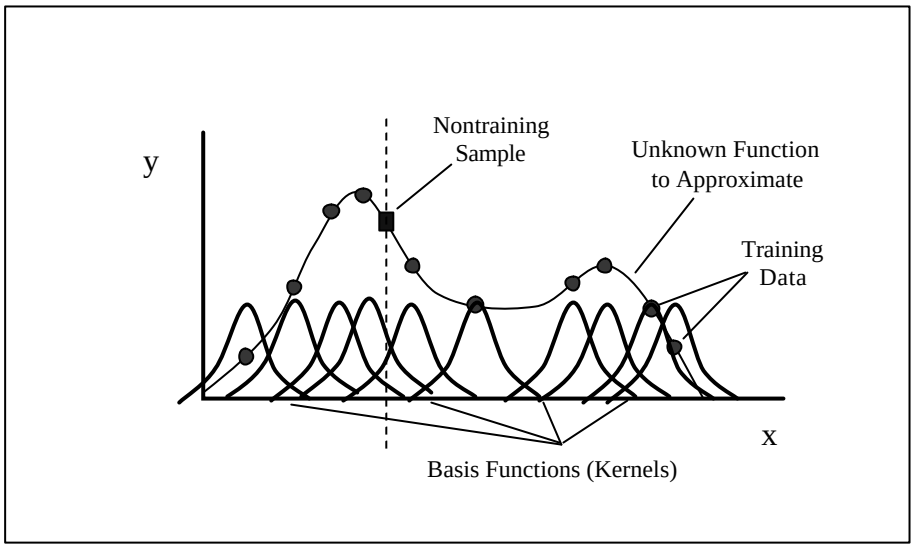
Structure of RBF Networks

- **Input layer**
- **Hidden layer**
  - **Hidden units provide a set of basis function**
  - **High dimension: more linearly separable [Cover's Theorem]**
- **Output layer**
  - **Linear combination of hidden functions**



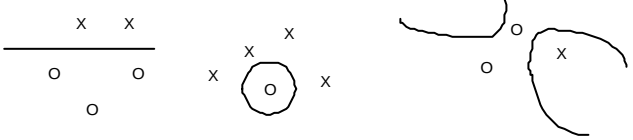
Lecture 7-4

The idea



Cover's Theorem on Separability(1)

- **A pattern classification cast in high dimensional space nonlinearly is more likely to be linearly separable than in a low dimension space**



$X = (x_i, x_2)$

$j(X) = [x_1, x_2]$

$a_1 x_1 + a_2 x_2 + a_0 = 0$

(# of basis function: 2)

$j(X) = [x_1^2, x_2^2, x_1, x_2]$

$a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_0 = 0$

(# of basis function: 4)

$j(X) = [x_1^2, x_2^2, x_1, x_2, x_1 x_2]$

$a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_5 x_1 x_2 + a_0 = 0$

(# of basis function : 5)

Cover's Theorem on Separability(2)

- **Cover's theorem in case of polynomial functions**
  - **Let** $j(X) = [j_1(X), j_2(X), ..., j_{m_1}(X)]$
  - $j - separable$: $\begin{cases} W^T j(X) > 0, X \in \aleph_1 \\ W^T j(X) < 0, X \in \aleph_2 \end{cases}$
  - **Polynomial hidden function**
    - **r-th order rational varieties** $\sum_{0 \le i_1 \le i_2 ... \le i_r \le m_1} a_{i_1 i_2 ... i_N} x_{i_1} x_{i_2} ... x_{i_r} = 0$
  - **Probability that particular dichotomy picked at random is**
    - **N: # of data points**
    - **Separating surface:** m_1 **degree of freedom**
    - **If** m_1 **is increasing, the probability of separability is increasing**

$$P(N, m_1) = (\frac{1}{2})^{N-1} \sum_{m=0}^{m_1 - 1} \binom{N-1}{m}$$

Example: XOR Problem

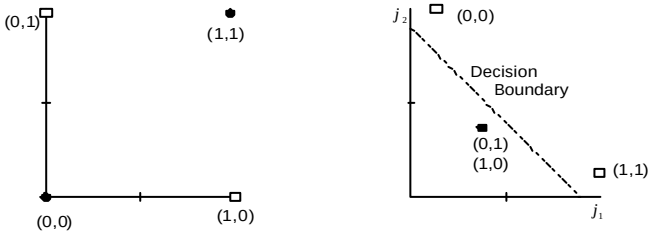
- **XOR data are not linearly separable**
- **Nonlinear transformation**

$$j_1(x) = e^{-\|x - t_1\|^2}, t_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$j_2(x) = e^{-\|x - t_2\|^2}, t_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Input pattern X	First Hidden Function $j_1(x)$	Second Hidden Function $j_2(x)$
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678

- **Linearly separable in** $(j_1(x), j_2(x))$ **space**

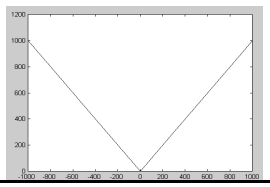
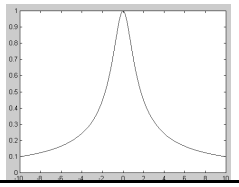
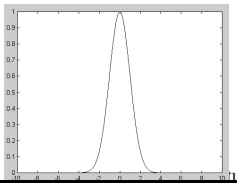


Types of RBF function

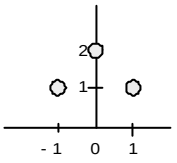
$$j(r) = \frac{1}{(r^2 + c^2)^{1/2}} \text{ for some } r > 0,\ r \in \mathbf{R}$$

- **Multiquadrics**
- **Inverse multiquadrics** $j(r) = (r^2 + c^2)^{1/2}$ for some $r > 0,\ r \in \mathbf{R}$
- **Gaussian functions** $j(r) = \exp(-\frac{r^2}{2s^2})$ for some $r > 0, r \in \mathbf{R}$

• Inverse multiquadircs and Gaussian RBFs are both examples of 'localized' functions
• Multiquadrics RBFs are 'nonlocalized' functions
• 'Localized': as distance from the centre increases the output of the RBF decreases
• 'Nonlocalized': as distance from the centre increases the output of the RBF increases

- **Example**
  - **{(x,d)} ={ (-1,1),(0,2),(1,1) }**



$$j(r) = \sqrt{(r^2 + 0.5)}$$

$$F(X) = w_1 j(|X - X_1|) + w_2 j(|X - X_2|) + w_3 j(|X - X_3|)$$
$$= w_1\sqrt{(x+1)^2 + 0.5} + w_2\sqrt{x^2 + 0.5} + w_3\sqrt{(x-1)^2 + 0.5}$$

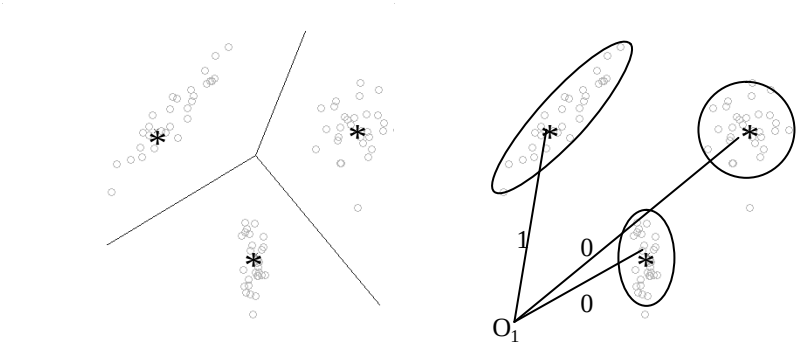
$$\begin{bmatrix} \sqrt{0.5} & \sqrt{1.5} & \sqrt{4.5} \\ \sqrt{1.5} & \sqrt{0.5} & \sqrt{1.5} \\ \sqrt{4.5} & \sqrt{1.5} & \sqrt{0.5} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} \sqrt{0.5} & \sqrt{1.5} & \sqrt{4.5} \\ \sqrt{1.5} & \sqrt{0.5} & \sqrt{1.5} \\ \sqrt{4.5} & \sqrt{1.5} & \sqrt{0.5} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

RBF of Summary

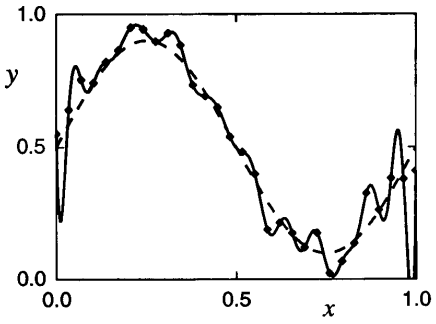
- **RBF Networks**
  - **Three layer : Input layer, hidden layer, output layer**
  - **Hidden units provide set of basis functions for input vectors**
  - **Dimension of hidden layer is much larger than that of input layer**
  - **Output is obtained from linear combination of hidden functions**
- **Cover's theorem**
  - **A pattern classification cast in high dimensional space nonlinearly is more likely to be linearly separable than in a low dimension space**

•Idea is to use a weighted sum of the outputs from the basis functions which for e.g. classification, density estimation etc.
•Theory can be motivated by many things (regularization, Bayesian classification, kernel density estimation, noisy interpolation etc), but all suggest that basis functions are set so as to represent the data.
•Thus centers can be thought of as prototypes of input data.

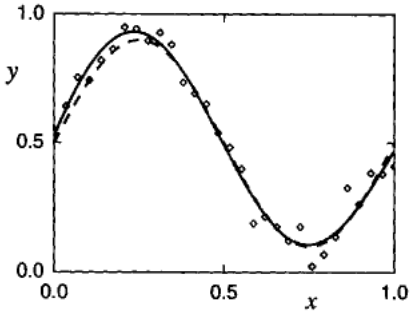


MLP distributed vs RBF local

Problems with exact interpolation



All Data Points



5 Basis functions

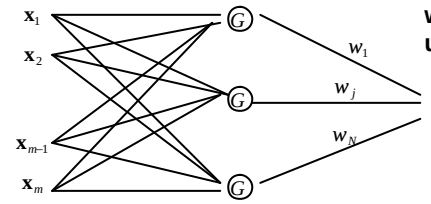
Lecture 7-13

Regularization Networks(1)

- **The expansion of the regularized approximating function**

$$F_l(\mathbf{x}) = \sum_{i=1}^{N} ?_i \mathbf{G}(\mathbf{x},\mathbf{x}_i)$$

- **Regularization Network**



Input Layer

Hidden layer of N Green's functions

Output layer

the i th hidden unit : $G(\mathbf{x},\mathbf{x}_i)$

the output of the network is a linearly weighted sum of the outputs of hidden units

$$\mathbf{w} = (\mathbf{G} + l\mathbf{I})^{-1}\mathbf{d}$$

lamda : positive real number (regularization parameter)

Lecture 7-14

Regularization Networks(2)

- Three desirable properties of the regulation network(Poggie and Girosi, 1990a)
  - universal approximator - it can approximate arbitrary well any multivariate continuous function on a compact subset of $\mathbf{R}^{m_0}$
  - Best-approximation property - given an unknown nonlinear function f , there always exists a choice of coefficients that approximates f better than all other possible choices
  - Optimal - the solution computed by the regularization network is optimal

Generalized Radial-Basis Function Networks(1)

- One-to-one correspondence between the training input $\mathbf{x}_i$ data and the Green's function $G(\mathbf{x}, \mathbf{x}_i)$ make regularization network prohibitively expensive to implement in computational terms for large N
  - ex) the computation of the linear weights of the network requires the inversion of an N-by-N matrix : grows polynomially with N
- To overcome
  - complexity of the network would have to be reduced
  - requires an approximation to the regularized solution
- Searching for suboptimal solution in a lower-dimensional space that approximates the regularized solution

Generalized Radial-Basis Function Networks(2)

- **Using Galerkin's method, the approximated solution $F^*(\mathbf{x})$ is expanded as shown by(Poggio and Girosi, 1990a)**

$$F^*(\mathbf{x}) = \sum_{i=1}^{m_1} ?_i j_i(\mathbf{x})$$

where

$$j_i(\mathbf{x}) = G(\|\mathbf{x} - \mathbf{t}_i\|) \quad , \quad i = 1,2,..,m_1 \quad , \quad m_1 < N$$

the set of centers $\{t_i \mid i = 1,2,.., m_1\}$ is to be determined

thus

$$F^*(\mathbf{x}) = \sum_{i=1}^{m_1} ?_i G(\|\mathbf{x} - \mathbf{t}_i\|)$$

Lecture 7-17

Generalized Radial-Basis Function Networks(3)

- **New cost functional $E(F^*)$ is defined by**

$$E(F^*) = \sum_{i=1}^{N} \left(d_i - \sum_{j=1}^{m_1} \mathbf{w}_j G(\|\mathbf{x}_i - \mathbf{t}_j\|) \right)^2 + l \|\mathbf{D}F^*\|^2$$

- **The minimization of new cost function with respect to the weight vector ? yields the result**

$$(\mathbf{G}^T\mathbf{G} + l\mathbf{G}_0)? = \mathbf{G}^T\mathbf{d}$$

$$G_0 = \begin{bmatrix} G(\mathbf{t}_1,\mathbf{t}_1) & G(\mathbf{t}_1,\mathbf{t}_2) & ... & G(\mathbf{t}_1,\mathbf{t}_{m_1}) \\ G(\mathbf{t}_2,\mathbf{t}_1) & G(\mathbf{t}_2,\mathbf{t}_2) & ... & G(\mathbf{t}_2,\mathbf{t}_{m_1}) \\ .: & :. & : & : \\ G(\mathbf{t}_{m_1},\mathbf{t}_1) & G(\mathbf{t}_{m_1},\mathbf{t}_1) & ... & G(\mathbf{t}_{m_1},\mathbf{t}_{m_1}) \end{bmatrix}$$

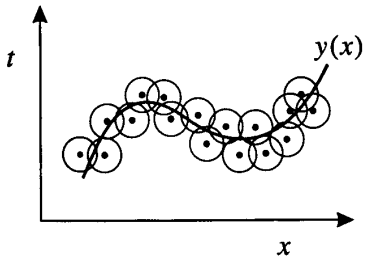
Lecture 7-18

Generalized Radial-Basis Function Networks(4)

- **As the regularization parameter l approaches zero, the weight vector w converges to the pseudoinverse solution to the overdetermined least-squares data-fitting problem for**

, $m_1 < N$ as shown by(Broomhead and Lowe, 1988)

$$? = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{d}, \; l = 0$$



All data is used for Interpolation

Lecture 7-19

Generalized Radial-Basis Function Networks(5)

- **Weighted Norm**
  - **previous norm is ordinarily intended to be a Euclidean norm**
  - **individual elements of the input vector $\mathbf{x}$ belong to different classes , it is appropriate to consider a general weighted norm**

the squared form

$$\|\mathbf{x}\|_C^2 = (\mathbf{C}\mathbf{x})^T(\mathbf{C}\mathbf{x}) = \mathbf{x}^T\mathbf{C}^T\mathbf{C}\mathbf{x}$$

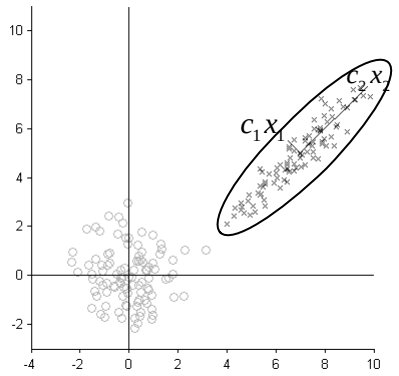
where $\mathbf{C}$ is $m_0 - by - m_0$ norm weighting matrix, and m_0 is the dimension of the input vector x

$$F^*(\mathbf{x}) = \sum_{i=1}^{m_1} ?_i \mathbf{G}(\|\mathbf{x} - \mathbf{t}_i\|_\mathbf{C})$$

$$\mathbf{G}(\|\mathbf{x} - \mathbf{t}_i\|_\mathbf{C}) = \exp(-(\mathbf{x} - \mathbf{t}_i)^T\mathbf{C}^T\mathbf{C}(\mathbf{x} - \mathbf{t}_i))$$
$$= \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{t}_i)^T \sum^{-1} (\mathbf{x} - \mathbf{t}_i))$$

Lecture 7-20

general weighted norm (Mahalanobis)



Lecture 7-21

Learning Strategies(1)

- **Learning process of RBF network**
  - **Hidden layer's activation function evolve slowly with some nonlinear optimization strategy.**
  - **Output layer's weight is adjusted rapidly through linear optimization strategy.**
  - **It is reasonable to separate the optimization of the hidden and output layers of the network by using different techniques, and perhaps on different time scales. (Lowe)**

Lecture 7-22

Learning Strategies(2)

- **Various learning strategies**
  - **According to the way how the centers of the radial-basis functions of the network are specified.**
  - **Interpolation theory**
    - **Fixed centers selected at random**
    - **Self-organized selection of centers**
    - **Supervised selection of centers**
  - **Regularization theory + kernel regression estimation theory**
    - **Strict interpolation with regularization**

Fixed centers selected at random(1)

- **The locations of the centers may be chosen randomly from the training data set.**
- **A radial basis function**

$$G(\|x - t_i\|^2) = \exp\left(-\frac{m_1}{d_{max}^2}\|x - t_i\|^2\right), \quad i = 1,2,..., m_1$$

  - m_1 **; number of centers**
  - d_{max} **; maximum distance between the chosen centers**
  - **standard deviation is fixed at** $\sigma = \frac{d_{max}}{\sqrt{2m_1}}$
- **We can use different values of centers and widths for each radial basis function -> experimentation with training data is needed.**

Fixed centers selected at random(2)

- Only output layer weight is need to be learned.
- Obtain the value of the output layer weight by pseudo-inverse method; $w = G^{+}d$
  - where G^{+} is pseudo-inverse matrix of the matrix $G = \{g_{ji}\}$

$$g_{ji} = \exp\left(-\frac{m_1}{d^2}\left\|x_j - t_i\right\|^2\right), \quad j = 1,2,...,N;\ i = 1,2,...,m_1$$

- Computation of pseudo-inverse matrix ; SVD decomposition
  - if G is a real N-by-M matrix, there exist orthogonal matrices
  - $U = \{u_1, u_2, ..., u_N\}$ and $V = \{v_1, v_2, \ldots, v_M\}$
  - such that $U^T G V = \mathrm{diag}(s_1, s_2, ..., s_K), K = \min(M, N)$
  - Then, pseudo inverse of matrix G is
  - $G^{+} = V \Sigma^{+} U^T$ where $\Sigma^{+} = \mathrm{diag}\left(\frac{1}{s_1}, \frac{1}{s_2}, ..., \frac{1}{s_K}, 0, ..., 0\right)$

Lecture 7-25

Self-organized selection of centers(1)

- Main problem of fixed centers method
  - it may require a large training set for a satisfactory level of performance
- Hybrid learning
  - self-organized learning to estimate the centers of RBFs in hidden layer
  - supervised learning to estimate the linear weights of the output layer
- Self-organized learning of centers by means of clustering (KNN)
- Supervised learning of output weights by LMS algorithm

Lecture 7-26

Self-organized selection of centers(2)

- k-means clustering
  - 1. Initialization - choose initial centers randomly
  - 2. Sampling - draw a sample vector x from input space
  - 3. Similarity matching - k(x) is index of the best matching center for input vector x

$$k(x) = \arg\min_k \|x(n) - t_k(n)\|, \quad k = 1,2,...,m_1$$

  - 4. Updating

$$t_k(n+1) = \begin{cases} t_k(n) + \eta\,[x(n) - t_k(n)], & k = k(x) \\ t_k(n), & \text{otherwise} \end{cases} \qquad 0 < \eta < 1$$

  - 5. Continuation - increment n by 1 and go back to step 2

Supervised selection of centers(1)

- All free parameters of the network are changed by supervised learning process.
- Error-correction learning using LMS algorithm.
- Cost function

$$E = \frac{1}{2}\sum_{j=1}^{N} e_j^2$$

- Error-signal

$$e_j = d_j - F^*(x_j)$$
$$= d_j - \sum_{i=1}^{M} w_i G(\|x_j - t_i\|_{C_i})$$

Supervised selection of centers(2)

- Find the free parameters so as to minimize E.
- linear weights

$$\frac{\partial E(n)}{\partial w_i(n)} = \sum_{j=1}^{N} e_j(n) G(\|x_j - t_i(n)\|_{C_i})$$

$$w_i(n+1) = w_i(n) - \eta_1 \frac{\partial E(n)}{\partial w_i(n)}, \quad i = 1,2,..., m_1$$

- position of centers

$$\frac{\partial E(n)}{\partial t_i(n)} = 2 w_i(n) \sum_{j=1}^{N} e_j(n) G'(\|x_j - t_i(n)\|_{C_i}) \Sigma_i^{-1} [x_j - t_i(n)]$$

$$t_i(n+1) = t_i(n) - \eta_2 \frac{\partial E(n)}{\partial t_i(n)}, \quad i = 1,2,..., m_1$$

- spreads of centers

$$\frac{\partial E(n)}{\partial \Sigma_i^{-1}(n)} = -w_i(n) \sum_{j=1}^{N} e_j(n) G'(\|x_j - t_i(n)\|_{C_i}) Q_{ji}(n)$$

$$\Sigma_i^{-1}(n+1) = \Sigma_i^{-1}(n) - \eta_3 \frac{\partial E(n)}{\partial \Sigma_i^{-1}(n)} \qquad Q_{ji}(n) = [x_j - t_i(n)][x_j - t_i(n)]^T$$

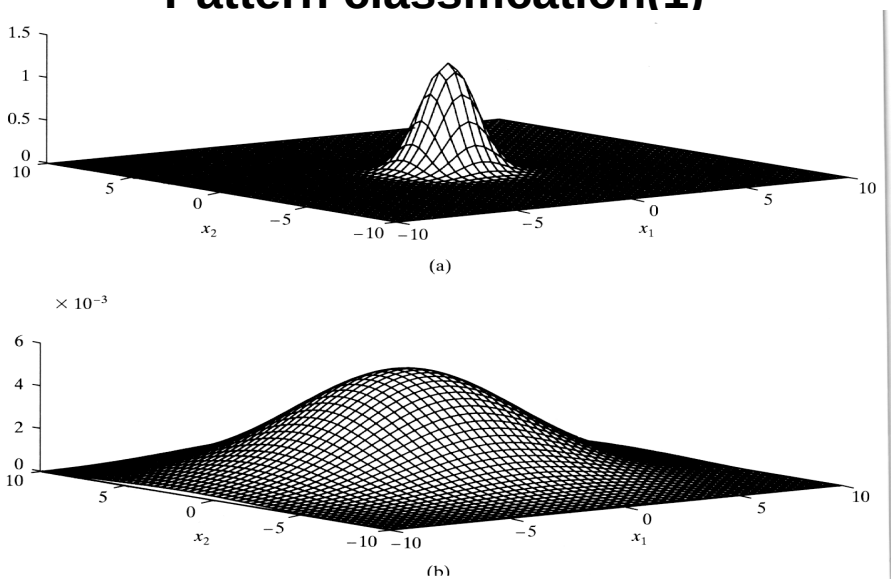
Lecture 7-29

Supervised selection of centers(3)

- Notable points
  - The cost function E is convex w.r.t linear parameter w_i
    The cost function E is not convex w.r.t t_i and Σ_i^{-1}
    -> search may get stuck in a local minimum in parameter space
  - Different learning-rate parameter for each parameter's update eqn. η_1, η_2, η_3 respectively.
  - The gradient-descent procedure in RBF does not involve error back-propagation.
  - The gradient vector $\partial E(n)/\partial t_i(n)$ has an effect similar to a clustering effect that is task-dependent.

Lecture 7-30

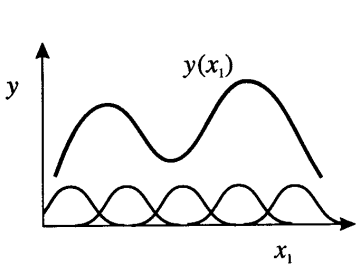
Computer experiment : Pattern classification(1)



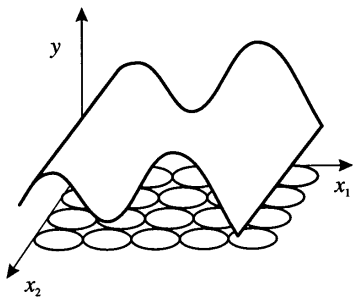
Problems of RBFs (1)

1. Need to choose number of basis functions

2. Due to local nature of basis functions has problems in ignoring 'noisy' input dimensions unlike MLPs (helps to use dimensionality reduction such as PCA)



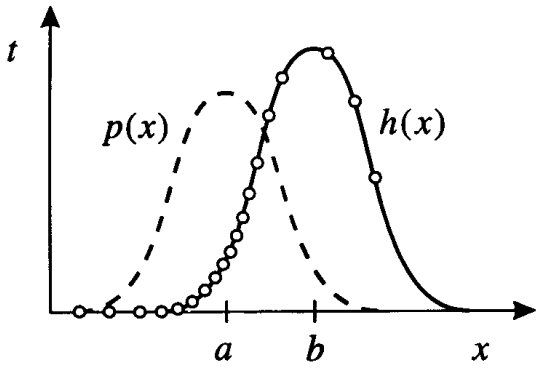
1D data, M rbfs



Same data with uncorrelated noise, M^2 rbfs

Problems of RBFs (2)

3. Optimal choice of basis function parameters may not be optimal for the output task



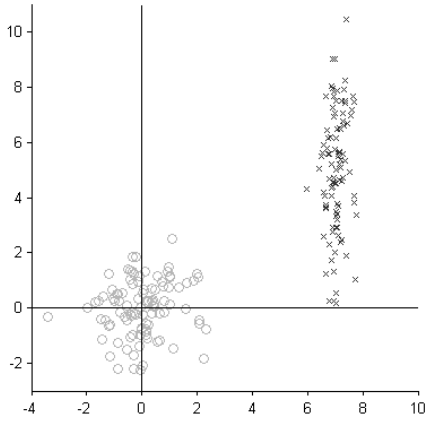
Data from h => RBF at a, but gives a bad representation of h. In contrast, one centered at b would be perfect

Problems of RBFs (3)

4. Because of dependence on distance, if variation in one parameter is small with respect to the others it will contribute very little to the outcome $(l + e)^2 \sim l^2$. Therefore, preprocess data to give zero mean and unit variance via simple transformation:

$$\underline{x}^* = \frac{(\underline{x} - \underline{m})}{s}$$

(Could achieve the same using general covariance matrices but this is simpler)



Comparison of RBF Networks with MLPs

- Similarities: **It is not surprising, then, to find that there always exists an RBF network capable of accurately mimicking a specified MLP, or vice versa.**
  - **1. They are both non-linear feed-forward networks.**
  - **2. They are both universal approximators.**
  - **3. They are used in similar application areas.**

Comparison of RBF Networks with MLPs

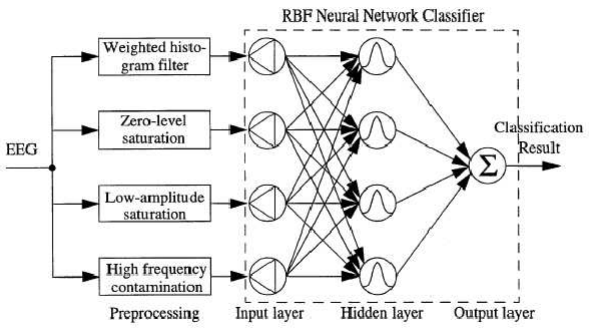
- Differences: **Although, for approximating non-linear input-output mappings, the RBF networks can be trained much faster, MLPs may require a smaller number of parameters.**
  - **1. An RBF network (in its natural form) has a single hidden layer, whereas MLPs can have any number of hidden layers.**
  - **2. RBF networks are usually fully connected, whereas it is common for MLPs to be only partially connected.**
  - **3. In MLPs the computation nodes (processing units) in different layers share a common neuronal model, though not necessarily the same activation function. In RBF networks the hidden nodes (basis functions) operate very differently, and have a very different purpose, to the output nodes.**

Comparison of RBF Networks with MLPs

- **4. In RBF networks, the argument of each hidden unit activation function is the** *distance* **between the input and the "weights" (RBF centers), whereas in MLPs it is the** *inner product* **of the input and the weights.**
- **5. MLPs are usually trained with a single global supervised algorithm, whereas RBF networks are usually trained one layer at a time with the first layer unsupervised.**
- **6. MLPs construct** *global* **approximations to non-linear input-output mappings with** *distributed* **hidden representations, whereas RBF networks tend to use** *localized* **non-linearity (Gaussians) at the hidden layer to construct** *local* **approximations.**

Real World Application – EEG Analysis

- **One successful RBF network detects epileptic form artifacts in EEG recordings:**



For full details see the original paper by: A. Saastamoinen, T. Pietilä, A. Värri, M. Lehtokangas, & J. Saarinen, (1998). Waveform detection with RBF network – Application to automated EEG analysis. *Neurocomputing,* vol. **20**, pp. 1-13

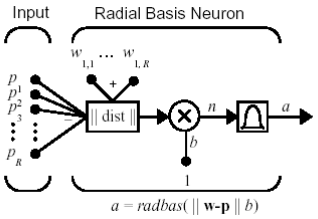
Summary

- **The structure of RBF network**
  - **hidden units are entirely different from output units.**
- **For a given data set containing N points *(xi,di)*, i=1,…,N**
  - **Choose a RBF function G**
  - **Calculate** $G(\|x_j - t_i\|)$
  - **Obtain the matrix G**
  - **Solve the linear equation G W = d**
  - **Get the unique solution**
  - **Done**

MATLAB Toolbox

- **net = newrbe(P,T,SPREAD)**
  - **Description of function**
    - **Radial basis networks can be used to approximate functions. NEWRBE very quickly designs a radial basis network with zero error on the design vectors.**
  - **NEWRBE(P,T,SPREAD) takes two or three arguments,**
    - **P - RxQ matrix of Q input vectors.**
    - **T - SxQ matrix of Q target class vectors.**
    - **SPREAD - of radial basis functions, default = 1.0.**
    - **and returns a new exact radial basis network.**
    - **The larger that SPREAD, is the smoother the function approximation will be. Too large a spread can cause numerical problems.**

- **Here is a radial basis network with *R* inputs**



Lecture 7-41

- **Example: function approximation sin(2*pi*x):x[0 1]**
- **See the M_file**

E:\Matlab6.5\work\rbfsin.m

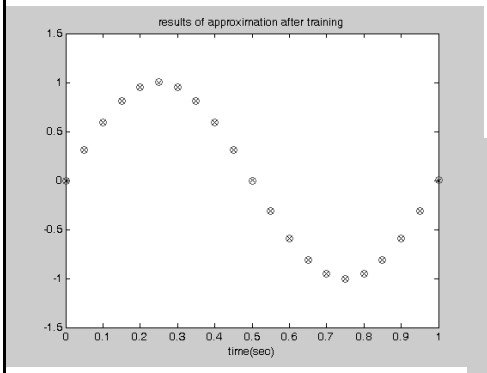
File Edit View Text Debug Breakpoints Web Window Help

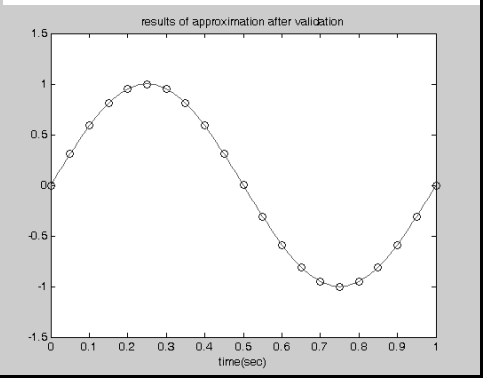
```
%function approximation :sin(2*pi*x) in [0 1] with RBF
clear
close all
x=0:0.05:1;
y=sin(2*pi*x);
plot(x,y,'o');xlabel('time(sec)');ylabel('sin(x)');
title('N-point of function')
P=x;
T=y;
%learnig phase
net = newrbe(P,T,1);
Y = sim(net,P);
figure;plot(P,T,'x',P,Y,'o');xlabel('time(sec)');
title('results of approximation after training')
%validation phase
t=0:.01:1;
Y = sim(net,t);
figure;plot(P,T,'o',t,Y)
xlabel('time(sec)');
title('results of approximation after validation')
```

script Ln 21 Col 1

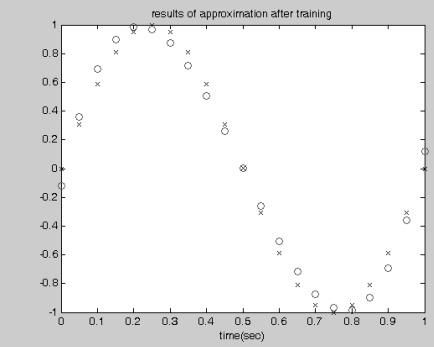
42

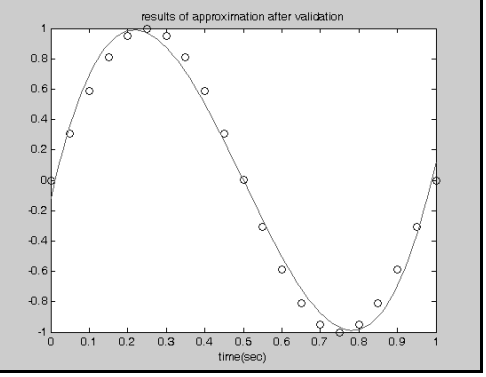
■ **For spread=1**



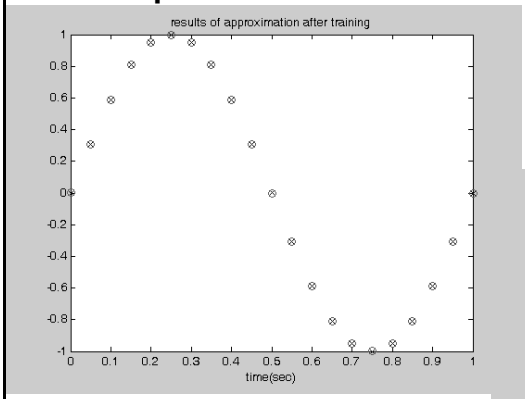


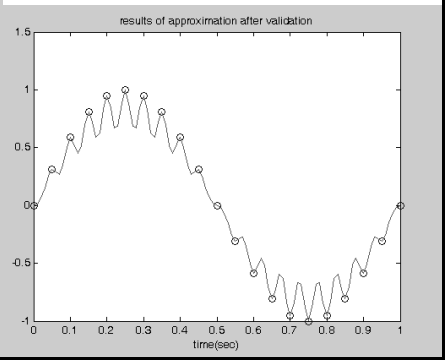
■ **For spread=10**





■ **For spread=0.02**





■ **N=11 and spread=1;**