Part 4 Machine Learning and Nonlinear System Modeling

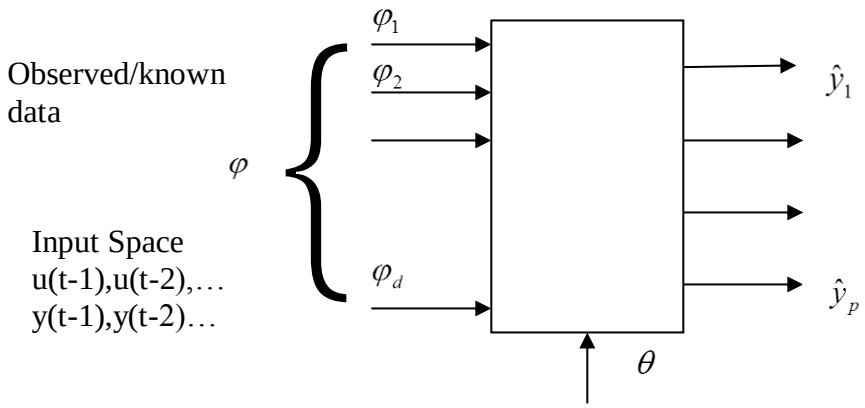
2.160 IDENTIFICATION, ESTIMATION, AND LEARNING

LECTURE NOTES NO. 15

In this part of the course, we will consider representation of nonlinear functions. In engineering problems we often need to deal with functional relationships that are nonlinear. Estimation and identification of nonlinear dynamical systems are challenging. It is often more tractable if we can divide the system into a nonlinear algebraic function and a linear dynamical subsystem, like Hammerstein model and Wiener model. In this chapter we will address how such a nonlinear algebraic function can be represented in a structure that is convenient for identification from sample data.

16. Nonlinear Models

16.1 Nonlinear Black-Box Models



Suppose that there is a nonlinear functional relationship between input vector φ and output y (without loss of generality, we assume the output is scalar in this section). Let $g_k(\varphi)$, $k = 1,\ldots, m$, be a set of nonlinear functions, e.g. sine and cosine functions.

Consider to approximate a nonlinear function $y(\phi)$ by a linear combination of nonlinear functions:

$$\hat{y} = \sum_{k=1}^{m} \alpha_k g_k(\varphi) \tag{1}$$

Functions $g_k(\phi)$'s can be viewed as "bases" in m dimensional vector space and α_k's can be treated as coordinates associated with the bases.

There are a number of Basis Functions that can be used for (1). They are classified into:

- Global basis functions
  - Fourier series
  - Volterra series

  } Varying over a large area in the variable space; Representing global features
- Local basis functions
  - Neural networks

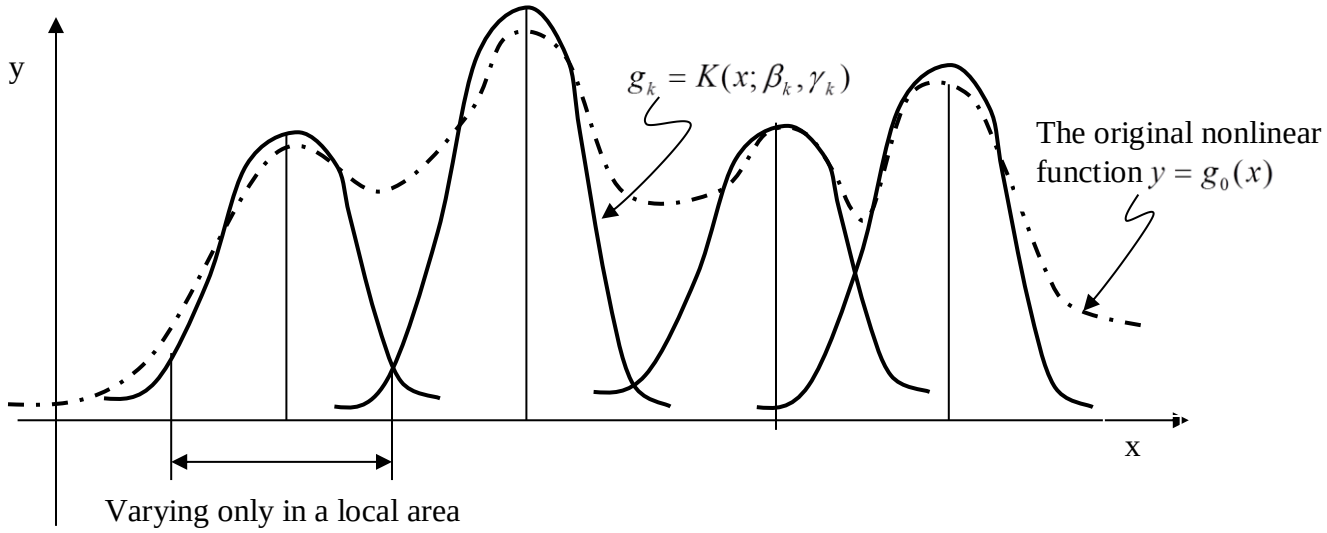
  Significant variation only in a local area

- Radial basis functions
- Wavelets

Local basis functions are powerful tools for capturing local features and representing a nonlinear function with locally-tunable resolution and accuracy. Over the last few decades, local basis functions have been investigated extensively. They have been applied to a number of system identification, learning, and control problems. We will focus on local basis functions in this chapter.

16.2 Local Basis Functions

We begin with a problem to approximate a scalar nonlinear function, $y = g_0(x), y \in R, x \in R$, with a group of basis functions, $g_k = K(x; \beta_k, \gamma_k)$, each of which covers only a local interval of axis x with parameters β_k and γ_k. See the figure below.



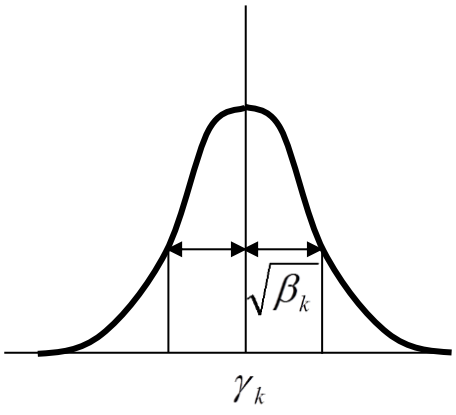
All the basis functions $g_k(\varphi)$, $k = 1,\ldots,m$ are generated from a single mother function of a single input variable, i.e. univariate: $K(x; \beta_k, \gamma_k)$.

Example:

Gaussian Bell $K(x; \beta_k, \gamma_k) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\gamma_k)^2}{2\beta_k}}$ (2)

Scale, dilation

where parameter γ_k determines the center position, and parameter $\sqrt{\beta_k}$ determines the scale of the local basis function.



Placing these local basis functions at *m* different points along the *x* axis with appropriate scale factors, we want to approximate the original nonlinear function to the following form:

$$g_0(x) \cong \sum_{k=1}^{m} \alpha_k g_k(x;\beta_k,\gamma_k) \qquad (3)$$

A simple case is to use the same Gaussian bell functions, i.e. $\beta_k = \overline{\beta}$, and place it at *m* equally-spaced points between *x* = *a* and *x* = *b*, where the given nonlinear function is defined: $a \le x \le b$. It can be shown based on Function Approximation Theory that a large class of nonlinear functions can be approximated to *any* accuracy with this group of Gaussian bell functions. Not only Gaussian bell functions, but also many other basis functions satisfying mild conditions can be used for the local basis functions.

Function Approximation Theory

For $\forall \varepsilon > 0$, there exists $m < \infty$ such that

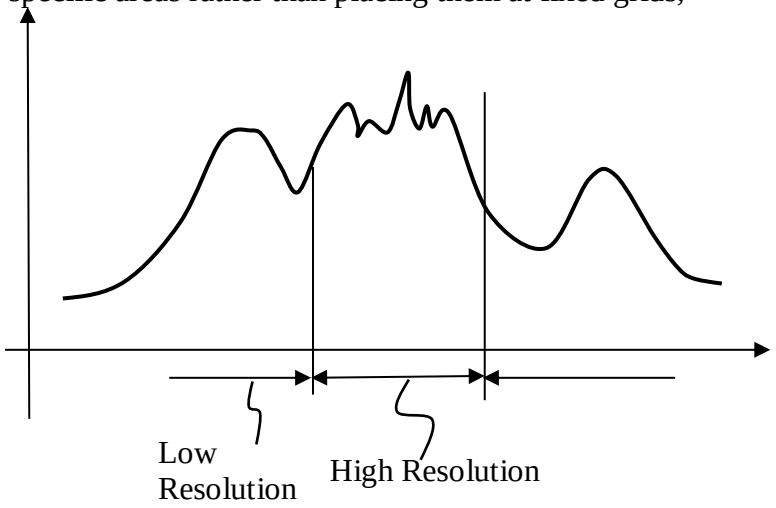
$$\left| g_0(x) - \sum_{k=1}^{m} \alpha_k g_k(x) \right| < \varepsilon \qquad (4)$$

Many mathematicians, including Kolmogorov, have worked on this problem and have extended the approximation throry to a large class of nonlinera functions, $g_0(x)$, and a large class of basis functions. Professor Tomaso Poggio has written an excellent survey paper on this topic.[1] His paper is available at the instructor's office.

The challenge of this function approximation is to minimize the number of basis functions while approximating a given nonlinear function to the same accuracy. It is interesting to know that the number of basis functions reduces quite significantly when they are placed more effectively at specific areas rather than placing them at fixed grids,

Features of local basis functions

- Multi-resolution
- Locally tunable
- (More stable in tuning)



[1] Poggio, T. and F. Girosi. Notes on PCA, Regularization, Sparsity and Support Vector Machines, *CBCL Paper #161/AI Memo #1632*, Massachusetts Institute of Technology, Cambridge, MA, April 1998.

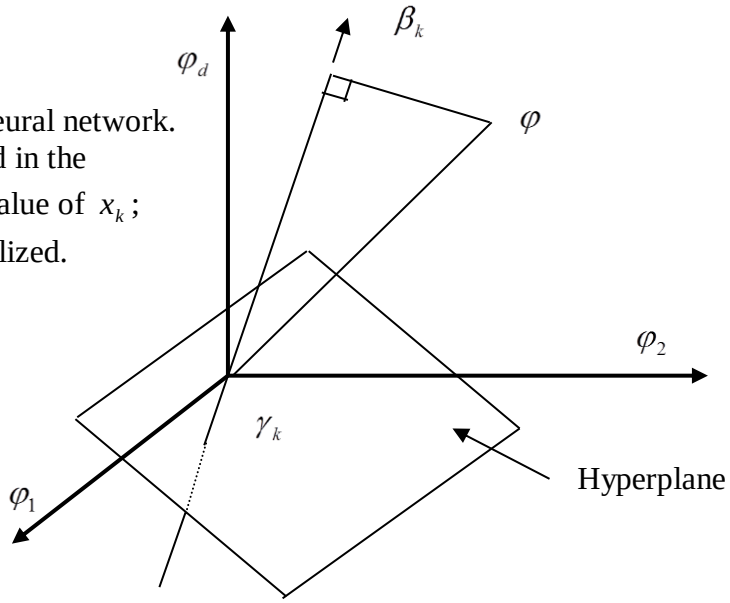
There are a number of local basis functions that have been used for diverse applications. They can be classified into three types.

1) Linear Combination type

$x_k = \beta_k^T \varphi + \gamma_k$

$\beta_k \in R^{d\times 1}, \gamma_k \in R^1$

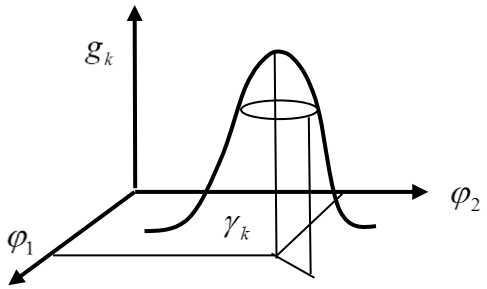
used for the perceptron multi-layer neural network. All the regression vectors φ involved in the hyperplane are mapped to the same value of x_k; The bases function $g_k(x_k)$ is not localized.



2) Distance type

$$x_k = \frac{|\varphi - \gamma_k|}{\beta_k} \qquad (6)$$

Locallized. Radial Basis Functions



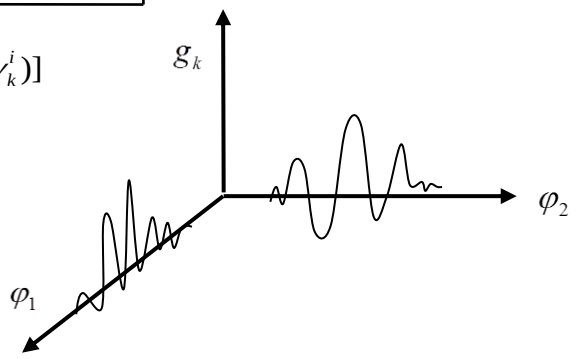
3) Product type

$g_k = K[\beta_k^1(\phi_1 - \gamma_k^1)]K[\beta_k^2(\phi_2 - \gamma_k^2)]\ldots K[\beta_k^d(\phi_d - \gamma_k^d)]$

d products of $K[\beta_k^i(\phi_i - \gamma_k^i)]$

Wavelets

Need a large number of basis functions



16.3 Non-Adaptive Tuning of Local Basis Function Networks

In the neural network community, parameter tuning or parameter estimation is called learning or training. (They tend to use colorful English!)

Parameters to tune

$$\theta = \begin{cases} \alpha_k & coordinates \\ \beta_k & scale \quad or \quad dilution \ parameters \\ \gamma_k & location \quad parameters \end{cases}$$

Data (Training Data)

Regression and corresponding true outputs

$\varphi(1)\ldots\varphi(N)$ $y(1)\ldots y(N)$

In the non-adaptive tuning problem under consideration, the scale parameters β_k and the location parameters γ_k are fixed; only coordinates α_k are learned from the input-output data

$$\hat{y} = \sum_{k=1}^{m} \alpha_k g_k(\varphi; \bar{\beta}_k, \bar{\gamma}_k) \tag{8}$$

Pre-determined A type of linear regression

This is a linear problem. The Least Squares estimate is applicable.

$$\begin{aligned} \hat{y} &= g_1(\varphi)\alpha_1 + \ldots + g_m(\varphi)\alpha_m \\ &= \begin{bmatrix} g_1(\varphi) & \ldots & g_m(\varphi) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix} \end{aligned} \tag{9}$$

Collectively arranging this vector $g(\varphi)$ for all the training data,

$$\begin{bmatrix} \hat{y}(1) \\ \hat{y}(2) \\ \\ \hat{y}(N) \end{bmatrix} = \underbrace{\begin{bmatrix} \leftarrow & g(\varphi(1)) & \rightarrow \\ \leftarrow & g(\varphi(2)) & \rightarrow \\ & \vdots & \\ \leftarrow & g(\varphi(N)) & \rightarrow \end{bmatrix}}_{\Phi^T \in R^{N\times m}} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \\ \alpha_m \end{bmatrix}}_{\alpha \in R^{m\times 1}} \tag{9'}$$

Φ has been used for $\Phi = [\varphi(1)\cdots\varphi(N)]$ in linear systems.

We use this for the above nonlinear problem, since there is no fundamental difference between the two.
The problem is to find α that minimizes the squared error of the above predictor compared with the true values (training data) $Y = [y(1) \cdots y(N)]^T$.

$$\hat{\alpha} = \arg\min_{\alpha} \left| Y - \Phi^T \alpha \right|^2 \tag{10}$$

The solution is

$$\hat{\alpha} = \left(\Phi \Phi^T\right)^{-1} \Phi Y \tag{11}$$

The Recursive Least Square (RLS) algorithm is also applicable. RLS is particularly useful for on-line learning as well as for dealing with a large number of training data.
Substituting (11) into (9) yields

$$\hat{y}(\varphi) = g(\varphi)\left(\Phi\Phi^T\right)^{-1}\Phi Y \tag{12}$$

$$\hat{y}(\varphi) = \sum_{i=1}^{N} S(\varphi, \varphi(i)) y(i) \tag{13}$$

$$\sum_{i=1}^{N} \begin{bmatrix} g_1(\varphi(i)) \\ g_2(\varphi(i)) \\ \vdots \\ g_m(\varphi(i)) \end{bmatrix} y(i) = \sum_{i=1}^{N} g^T(\varphi(i)) y(i)$$

where $S(\varphi, \varphi(i)) = g(\varphi)\left(\Phi \Phi^T\right)^{-1} g^T(\varphi(i))$ called the equivalent Kernel of the basis function.

16.4 Adaptive Tuning Methods for Radial Basis Function networks

Now we consider adaptive tuning methods that

- Allow to tune β_k and γ_k (scale and location parameters) together with the coordinator α_k by using both $\varphi(i)$ and $y(i)$, the training data; and
- Allow to allocate local basis functions more effectively to areas needing higher resolution.

Since α_k, β_k and γ_k are non-linearly involved in (9), adaptive methods are highly non-linear.

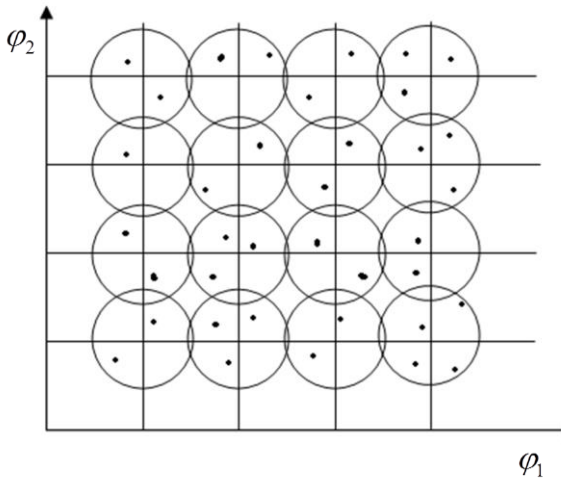
The following is an example of adaptive method: Radial Basis Function (RBF) Networks

The formula of RBF network is given by:

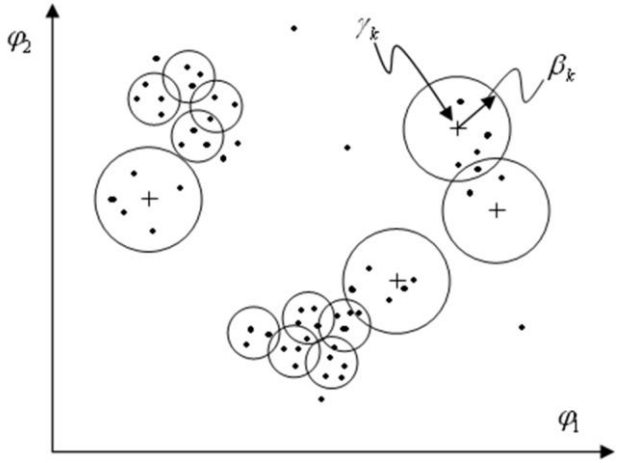
$$\hat{y}(\varphi; \alpha, \beta, \gamma) = \sum_{k=1}^{m} \alpha_k g\left(\frac{|\varphi - \gamma_k|}{\beta_k}\right) + \alpha_0 \tag{14}$$

where the mother basis function is the Gaussian bell, multi-quadratic function. The bias term α_0 can be treated as a special case of $\beta = \infty$.

Question: How can we determine β_k and γ_k ?



At fixed grid points



More basis functions at more densely populated areas

Allocation of the basis functions is a type of clustering problem or a vector quantization problem.

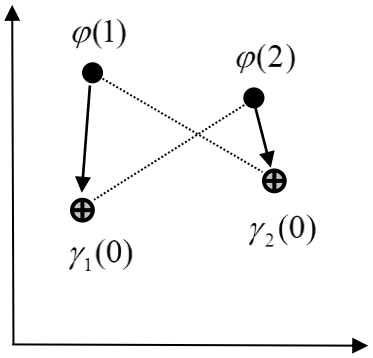
The *k*-Means Clustering algorithm is a well-known technique.

Problem:

Given the number of clusters (basis functions), *m*; initial locations of the *m* center points, $\gamma_1[0] \cdots \gamma_m[0]$; and data $\varphi(1) \cdots \varphi(N)$; Find optimal center points that minimize the mean squared distance between each center point γ_k and individual data points $\varphi(i)$ involved in the same cluster, *k*.

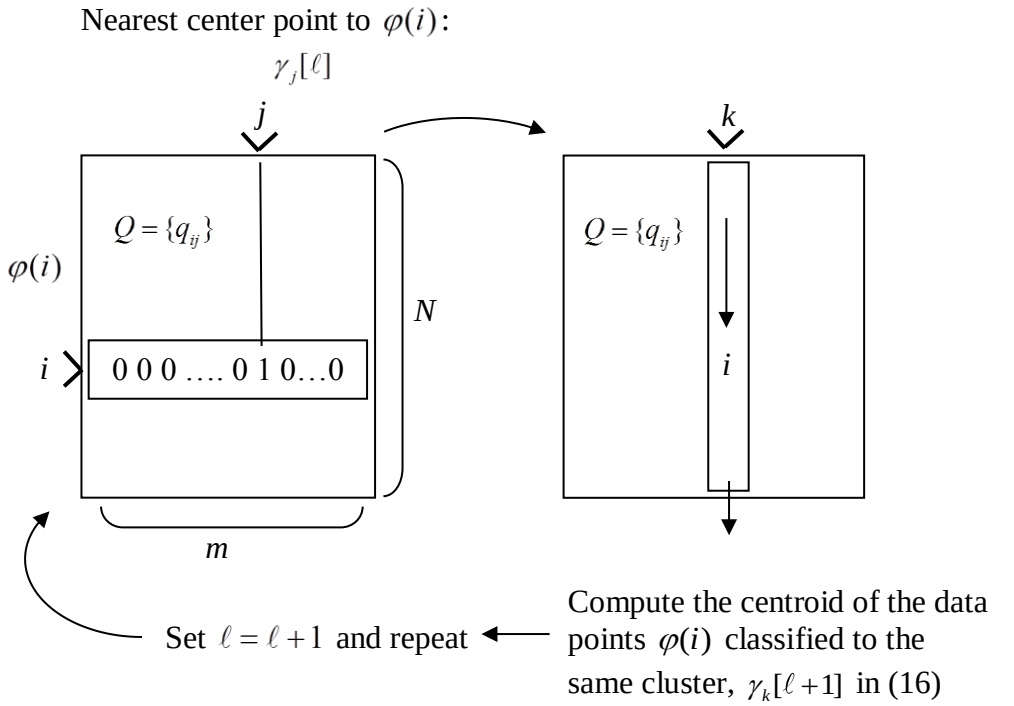
Algorithm

Set iteration number *l* to 1. Given initial center points,`



Step 1. Find the nearest center for each data point $\varphi(i)$ and store the results in an $N \times m$ matrix $Q = \{q_{ij}\}$, whose element is defined by

$$q_{ij} = \begin{cases} 1 & if \quad j = \arg\min_{1 \le k \le m} |\phi(i) - \gamma_k[l]| \\ 0 & elsewhere \end{cases} \tag{15}$$



Step 2. Compute the centroid of the data points $\varphi(i)$ classified to the same cluster

$$\gamma_k[l+1] = \frac{\sum_{i=1}^{N} \phi(i) q_{ik}[\ell]}{\sum_{i=1}^{N} q_{ik}[\ell]} \qquad k = 1, \ldots, m \tag{16}$$

Step 3. Set $\ell = \ell + 1$ and repeat steps 1-2 until the mean squared distance converges to a local minimum.

$$\frac{1}{N} \sum_{k=1}^{m} \sum_{i=1}^{N} |\phi(i) - \gamma_k[\ell]|^2 q_{ik}[\ell] \tag{17}$$

The scale (dilation) parameter β_k, called a receptive width, determines the smoothness of the approximation function as well as data fitting accuracy.

A heuristic method for determining the receptive width (variance) is given by

$$\beta_k = \frac{\sum_{i=1}^{N} |\varphi(i) - \gamma_k|^2 q_{ik}}{\sum_{i=1}^{N} q_{ik}} \tag{18}$$

Selection of β_k is a trade-off problem between fitting accuracy and smoothness. It is interpreted as the degree of generalization.