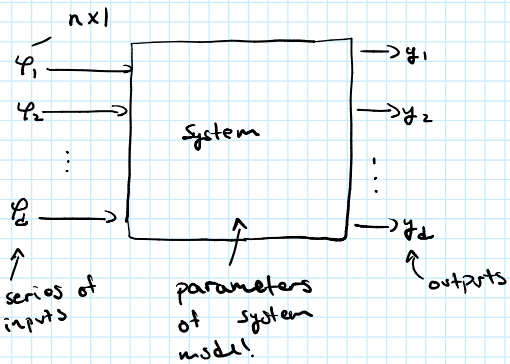
Lecture 16

Monday, March 14, 2022 10:04 AM

Nonlinear Modeling - I

Admin
① HWØ3, problem 3 → shift to HWØ4.
② HWØ3 due by friday.
③ Midterm exam Mar 30th.
↳ Cover through HWØ3.

Nonlinear Black Box Modeling



To generate a non-linear map from $\varphi \to y$, consider the functional expansion:

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

where $g_k(\varphi)$ are basis funcns & α_k is the magnitude of each one.

Types of Basis Funcns:

Global

- Fourier
- Volterre

} – vary over input output space
– represent global features of data.

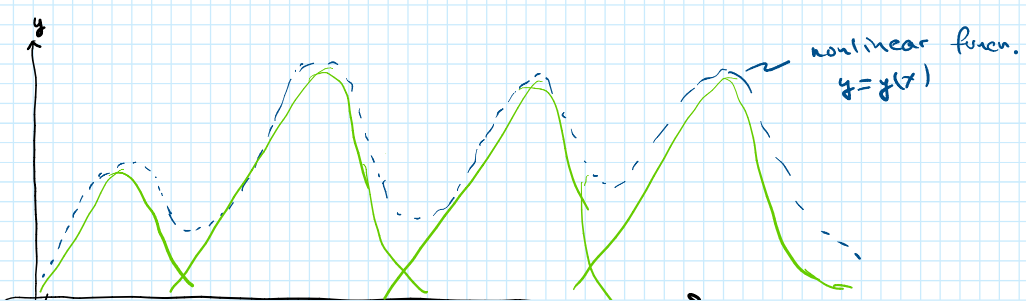
Local

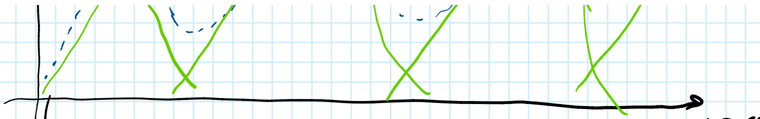
- Neural networks.
- Bayesian networks
- Wave lets
- Radial basis funcns.

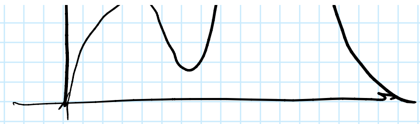
} – Represent features in local area of data.

Radial Basis Funcns:

- Consider non-linear input – output data







$x = \varphi$

estimate the data using a combination of Gaussian curves

$$\hat{y} = g(x)$$

$$= \sum_{k=1}^{d} \alpha_k\, g_k(x, \beta_k, \gamma_k) \; ; \quad g_k = \frac{1}{\sqrt{2\pi}} e^{-(x-\gamma_k)^2 / 2\beta_k}$$

scaling. center dilation

Methods. for Fitting Radial Basis Funcns. to Data

- Dilation, β_k, location, γ_k are already fixed.

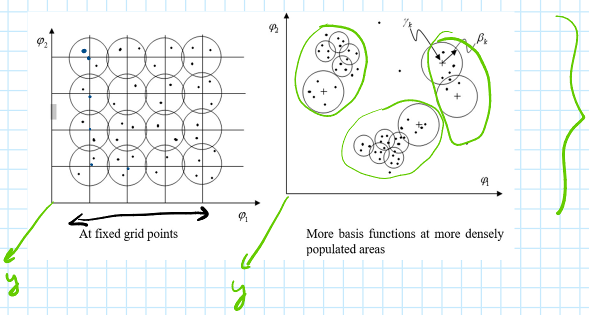
$$\hat{y} = \sum_{k=1}^{d} \hat{\alpha}_k\, g_k(\varphi, \hat{\beta}_k, \hat{\gamma}_k)$$

pre-determined.

$$\hat{y} = \left[g_1(\varphi) \quad g_2(\varphi), \dots \right] \begin{bmatrix} \hat{\alpha}_1 \\ \vdots \\ \hat{\alpha}_d \end{bmatrix}$$

$\leftarrow$ can solve using ordinary least squares.

- Identification of dilation & location: β_k, γ_k.



At fixed grid points

More basis functions at more densely populated areas

2D inputs.

clustering.

$\to$ Identification of β_k, γ_k is a nonlinear estimation problem.

$\to$ One soln. is the generalized Lloyd algorithm

Algorithm

Step 0:
- Initialize RBF into a grid.
- Set iteration number $l = 1$

Step 1: Find the nearest center for each data point $p(i)$ and store the results in a matrix. # of centers m.

$Q = \{ g_{ij} \}$ where $Q \leftarrow$ no. of datapoints N $\big[\quad \big]$

$$g_{ij} = \begin{cases} 1 & \text{if } j = \operatorname{argmin} |\varphi(i) - \gamma_k(j)| \\ 0 & \text{otherwise.} \end{cases}$$

$$\xi_{ij} = \begin{cases} 1 & \text{if} \quad j = \arg\min |\varphi(i) - \gamma_k(j)| \\ 0 & \text{otherwise} \end{cases}$$

<u>Step 2:</u> Compute the centroid of datapoints $\varphi(i)$ classified to the same cluster

$$\gamma_k(\ell+1) = \frac{\sum_{i=1}^{N} \varphi(i)\,\xi_{ik}}{\sum_{i=1}^{N} \xi_{ik}}, \quad k = 1 \ldots m$$

m ← # of clusters.

<u>Step 3:</u> Set $\ell = \ell+1$ and repeat until until the mean squared distance converges to a local minimum.

$$\frac{1}{N} \sum_{k=1}^{m} \sum_{i=1}^{N} |\varphi(i) - \gamma_k(\ell)|^2 \xi_{ik} \longrightarrow \text{minimum}$$

<u>Step 4:</u>
- Compute the dilation using heuristic:

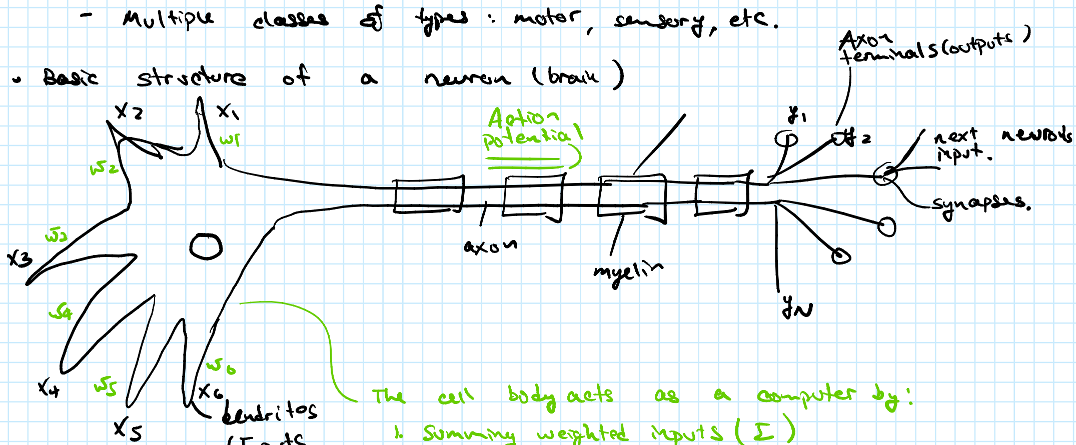
$$\beta_k^2 = \frac{\sum_{i=1}^{N} |\varphi(i) - \gamma_k|^2 \xi_{ik}}{\sum_{i=1}^{N} \xi_{ik}}$$

- Use least squares to compute scaling:

$$\hat{y}(\varphi, \alpha, \beta, \gamma) = \sum_{k=1}^{m} \alpha_k \, g\left(\frac{|\varphi - \gamma_k|}{\beta_k}\right) + \alpha_0 \leftarrow \text{least squares.}$$

<u>Neural Networks</u>

- Foundation of Artificial Neural Networks (ANNs)
  - ANN's inspired by the neurophysiology of the human brain.
    - The human brain has ~100 billion neurons.
    - Multiple classes of types: motor, sensory, etc.
  - Basic structure of a neuron (brain)



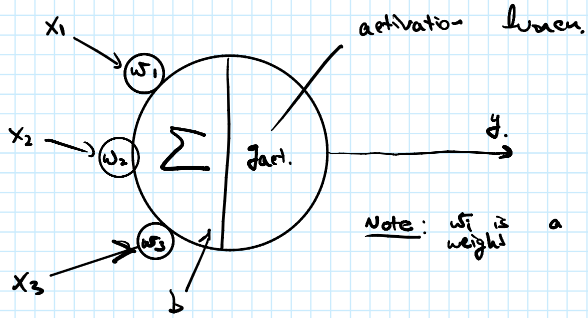
The cell body acts as a computer by:
1. Summing weighted inputs (Σ)
2. Firing an "action potential" base on an activation func. (g_{act}).

- Despite its simple structure, the neuron, when implemented in large networks, is capable of supporting massively parallel, distributed processing.

- Well suited for modern computing, popular in machine learning since 1980s.

- Well suited for modern computing, popular in machine learning since 1980s.
- ANNs often used for nonlinear system ID
  - Examples: Facial recognition, stock market prediction.
  - useful when underlying model is unknown.

<u>Artificial Neural Network Anatomy</u>



<u>Note</u>: w_i is a tunable input weight

<u>Key principle</u>: - Given some dataset $\{x_1, \ldots x_n\}$ and a desired output y, the weights w_i and bias b can be chosen to create a "perceptron" that produces the same input-out behavior.

- The perceptron works well as a classifier for linearly seperable data.
- The bias "b" allows the input threshold for the activation funcn. to be adjusted to better fit the data.