KIOS Research and Innovation Center of Excellence

MSc on Intelligent Critical Infrastructure Systems

Machine Learning

Lecture 4

Marios Polycarpou

Director, KIOS Research and Innovation Center of Excellence

Professor, Electrical and Computer Engineering

University of Cyprus

University of Cyprus

Imperial College London

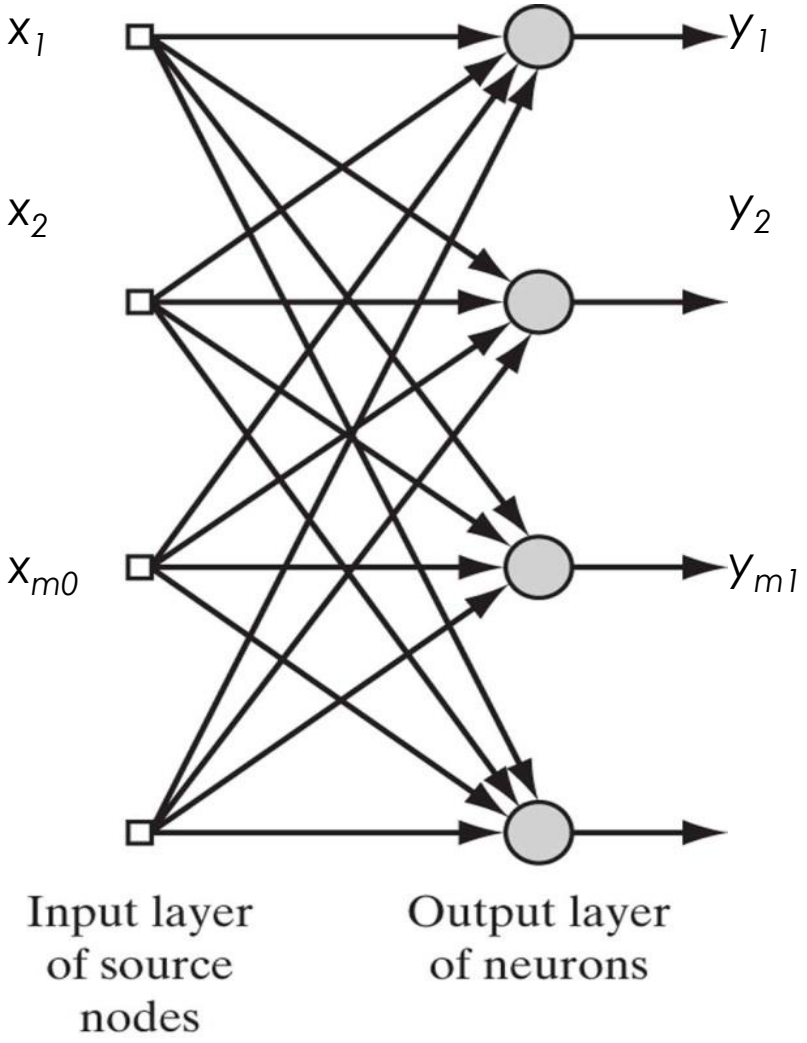
funded by:

Supervised learning in single layer ANNs

- One input layer with m_0 inputs and an output layer with m_1 output neurons; No hidden neurons
- The equations of the network are described by:

$$y_j = \varphi\left(\sum_{i=1}^{m_0} w_{ji} x_i\right), \quad j = 1, \ldots, m_1$$

- w_{ji}: weight associated with the i-th input to the j-th output neuron.
- The weights are the unknown parameters (the variables) that we have to adapt in order for an ANN to learn to perform a given task.
- The output of each neuron are independent of each other, so we can compute the weights associated with output neuron *j* separately (w_{ji}, for all *i*)

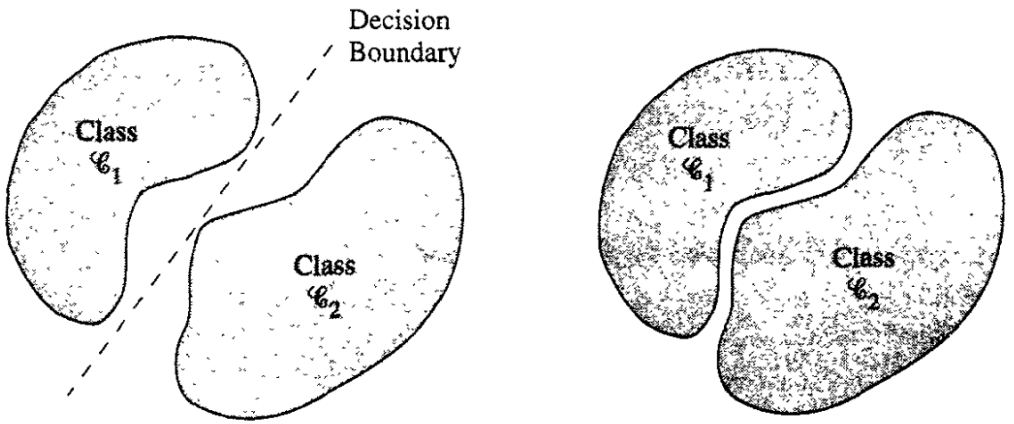


Classification with Rosenblatt's perceptron

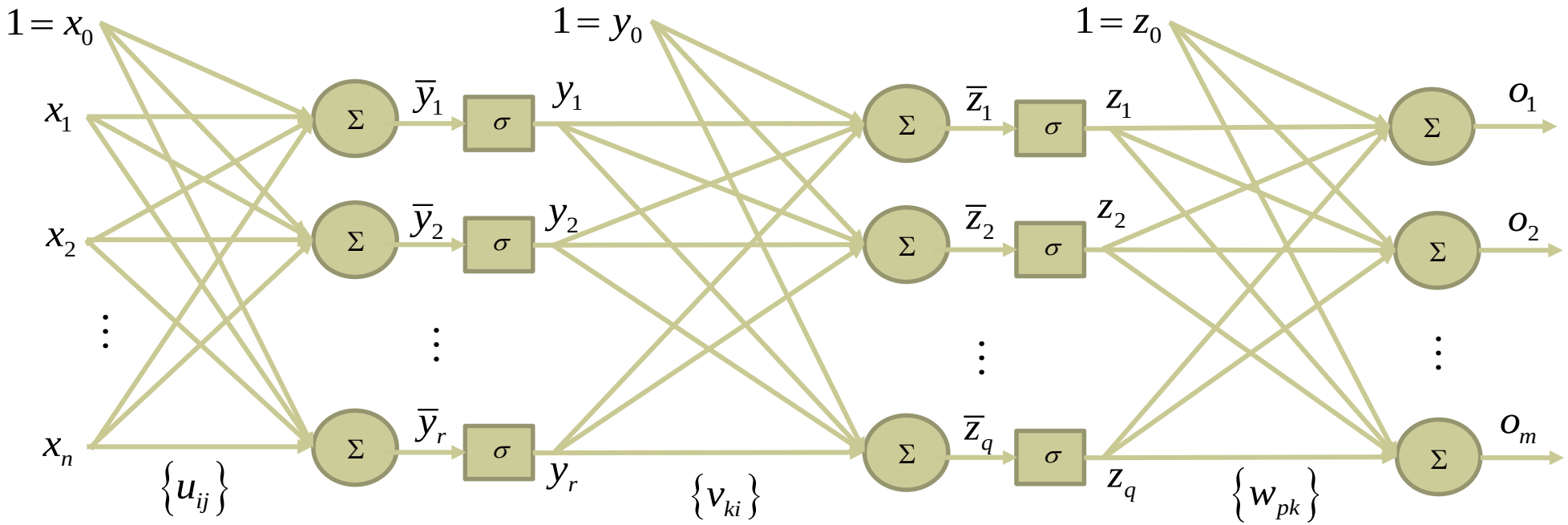
- Hard-limiter activation function: $$y = \varphi(z) = \mathrm{sgn}(z) = \begin{cases} 1, & z > 0 \\ -1, & z \le 0 \end{cases}$$
- Suitable for classifying two linearly separable classification classes C_1 and C_2.

$$\underline{w}^T\underline{x} > 0 \rightarrow C_1, \qquad \underline{w}^T\underline{x} \le 0 \rightarrow C_2$$

- $\underline{x}_1(i)$ are training examples that belong to class C_1 *and* $\underline{x}_2(i)$ are examples that belong to class C_2.
- If the two classes are linearly separable, it can be proven that there exists a vector $\underline{w}$ for which samples belonging to the two classes can be fully separated using Rosenblatt's perceptron neuron.



Multilayer Neural Networks



$$O = WZ$$
$$[m \times 1] = [m \times (q+1)] \times [(q+1) \times 1]$$

$$z_i = \sigma(\bar{z}_i),\ i = 1, \ldots, q \quad , \quad \bar{Z} = VY$$
$$[(q+1) \times 1] = [q \times (r+1)] \times [(r+1) \times 1]$$

$$y_j = \sigma(\bar{y}_j),\ j = 1, \ldots, r \quad , \quad \bar{Y} = UX$$
$$[(r+1) \times 1] = [r \times (n+1)] \times [(n+1) \times 1]$$

Weights: W, V, U

Input: X

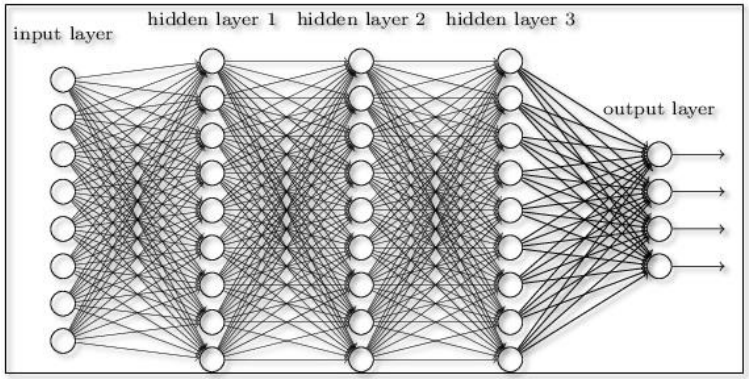
Output: O

Activation or squashing function: $\sigma(.) : \mathbb{R} \mapsto \mathbb{R}$

Multilayer Neural Networks – Properties

- Universal approximators
- Inherent potential for parallel computation
- Nonlinearly parameterized approximators
- Do not provide intuitive procedures for selecting the network structure.
- Has been utilized extensively in pattern recognition, image processing, classification, etc.

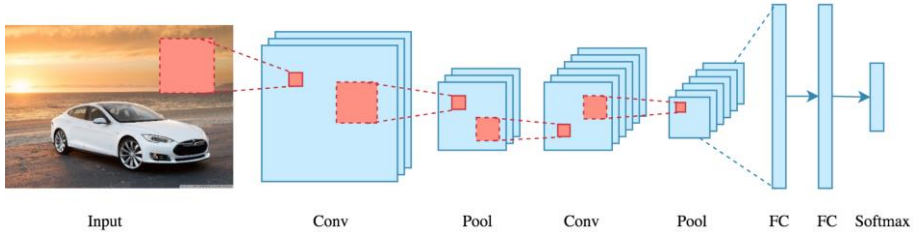
Deep Neural Networks and Deep Learning



- Typically, uses a large number of layers
- Each layer learns to transform its input data into a slightly more abstract and composite representation.
- Has been shown to work well in several applications, including automatic speech recognition, image processing, bioinformatics, board game program, etc)
- Requires a large number of adjustable parameters (weights)
- Lack of theory regarding the methods that are used.

Convolutional Neural Networks (CNN)

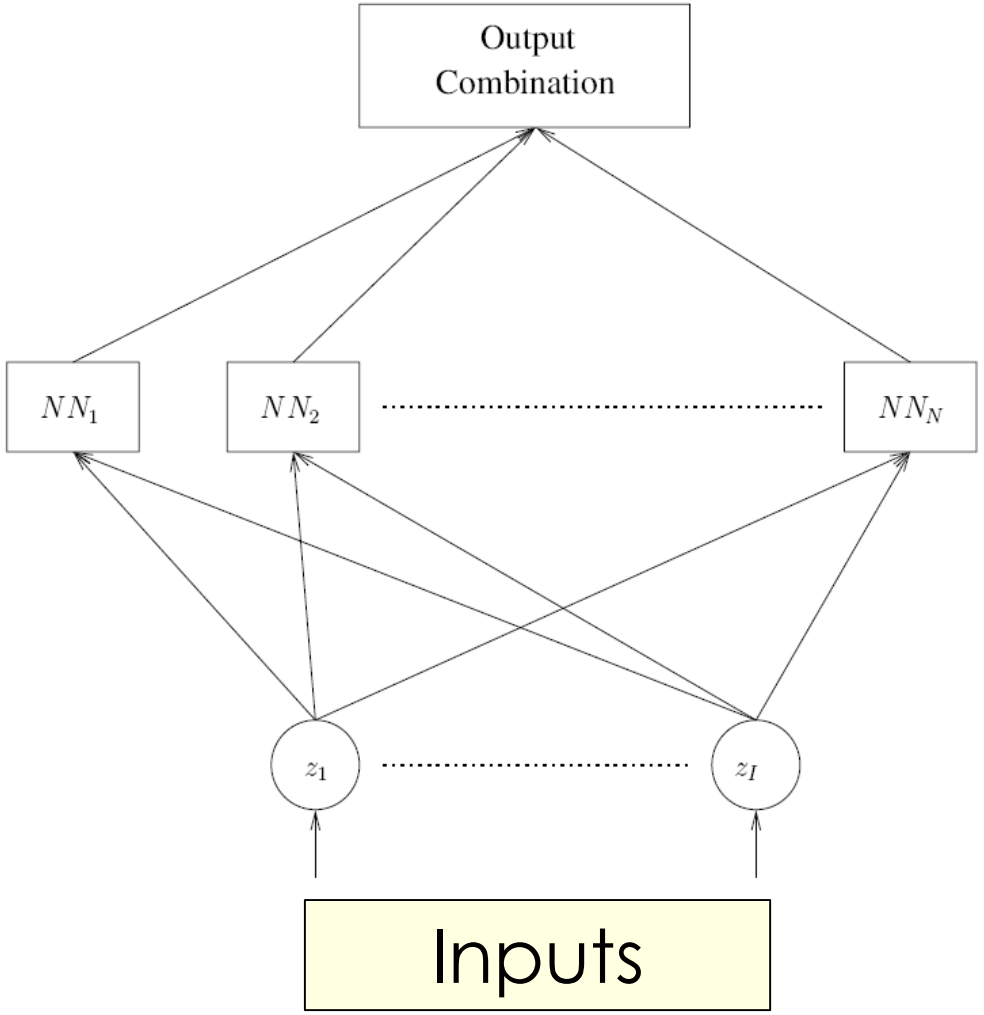
- CNN is a class of deep neural networks
- Typically, used for image processing



- Avoids the fully connectness between layers
- Avoids the overfitting problem and requires significantly fewer adjustable parameters compared to the full-connected multilayer neural network
- Convolution layer – convolve the input and pass it to the next layer. Instead of the inner product operation, the parameters of the hidden layer are sharing all the input pixels
- Pooling layer – reduce the dimension of the data by combining the outputs of a cluster of neurons to a single neuron (e.g., max pooling, average pooling)

Ensemble Neural Networks

- We can train each ANN of the ensemble either with the same or different training data
- It is not necessary for the ANNs to have the same characteristics
- Training can occur either in parallel or in a cooperative manner



Radial Basis Function (RBF) Networks

Motivation for RBF networks

- There exists strong connections to a number of scientific disciplines
  - function approximation
  - regularization theory
  - interpolation in the presence of noise
- RBFs allow for a straightforward interpretation of the internal representation produced by the hidden layer
- Training algorithms for RBFs are significantly faster than those for multilayer perceptrons

Radial Basis Function (RBF) Networks

Exact function interpolation

- RBFs have their origins in techniques for performing exact function interpolation
- Procedure:
  1. Assume a function $f(x)$ such that: $f(x^{(n)})=t^{(n)}$
  2. Place a basis function $\varphi(z)$ at each training example

$$h(x) = \sum_{k=1}^{N} w_k \varphi\left(\left\|x - x^{(n}\right\|\right) = \Phi w$$

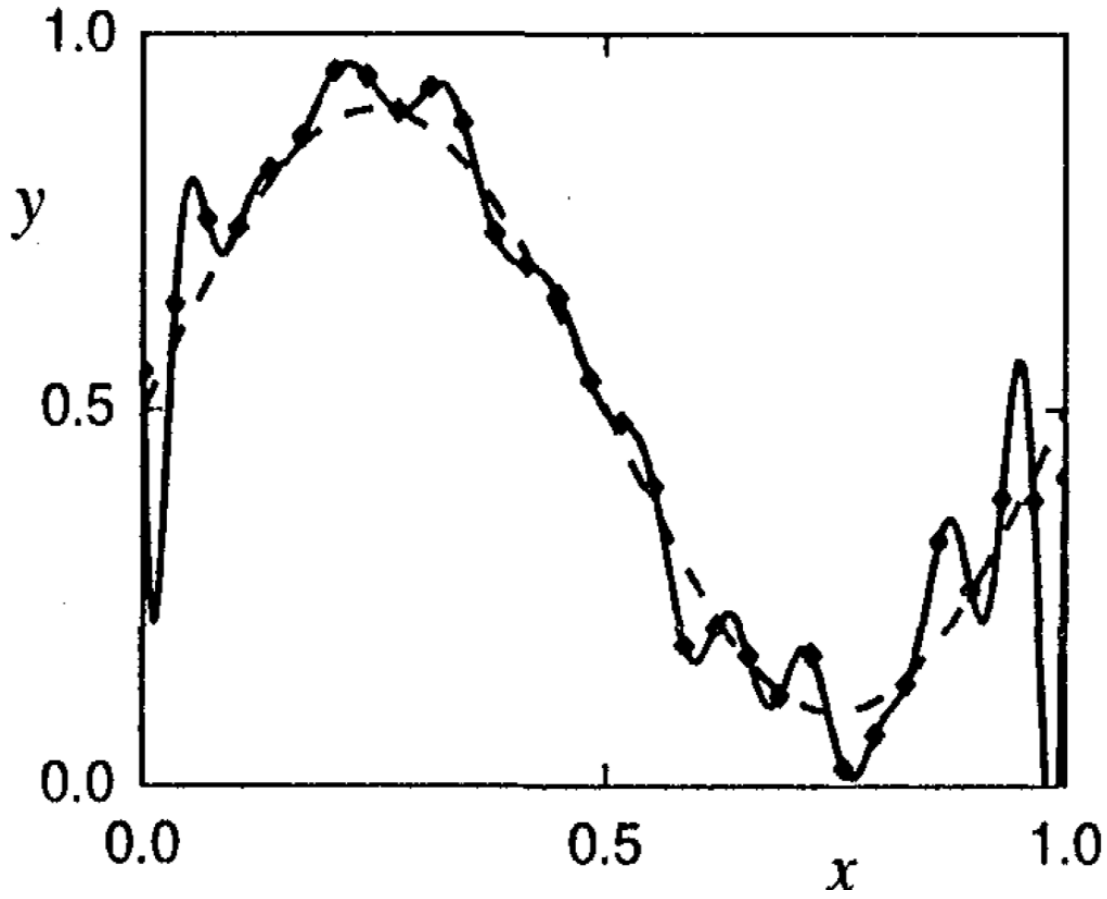
  3. Compute the coefficients so that the "mixture model" has zero error at those examples

$$h\left(x^{(i}\right) = \sum_{k=1}^{N} w_k \varphi\left(\left\|x^{(i} - x^{(n}\right\|\right) = t^{(i} \Leftrightarrow w = \Phi^{-1} t$$

Radial Basis Function (RBF) Networks

- Interpolated function:

  $y = 0.5 + 0.4\sin(2\pi x)$
- 30 data points
- Gaussian RBF.
- Drawbacks:
  - Approximating function is highly oscillatory
  - One RBF for each data points hence very costly for large data sets



Radial Basis Function (RBF) Networks

- **RBF networks are feed-forward networks consisting of**
  - A hidden layer of radial kernels
  - An output layer of linear neurons
- **Hidden layer:**
  - performs a non-linear transformation of input space
  - The resulting hidden space is typically of higher dimensionality than the input space
- **Output layer:**
  - performs linear regression to predict the desired targets
- **Why use a non-linear transformation followed by a linear one?**

  Cover's theorem on the separability of patterns: "*A complex pattern-classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space*"

Radial Basis Functions (RBFs)

$$\hat{f}(x) = \sum_{i=1}^{N} \theta_i g\left(\|x - c_i\|\right) + \sum_{i=1}^{\bar{k}} b_i p_i(x)$$

$\{c_i\}_{i=1}^{N}$ Set of center locations

$\|x - c_i\|$ Distance of the evaluation point to the i-th center

$g(\bullet): \mathbb{R}^+ \rightarrow \mathbb{R}^1$ radial function

$\{p_i(x)\}_{i=1}^{\bar{k}}$ Polynomial basis (allows RBF approximator to have polynomial precision). This is typically omitted.

Forms of Radial Basis Functions

Gaussian: $g_1(\rho) = \exp\left(-\frac{1}{2}\frac{\rho^2}{\gamma^2}\right)$

Multi-quadratic: $g_2(\rho) = \left(\rho^2 + \gamma^2\right)^{\beta}, \beta \in (0,1)$

Inverse multi-quadratic: $g_3(\rho) = \left(\rho^2 + \gamma^2\right)^{-\alpha}, \alpha > 0$

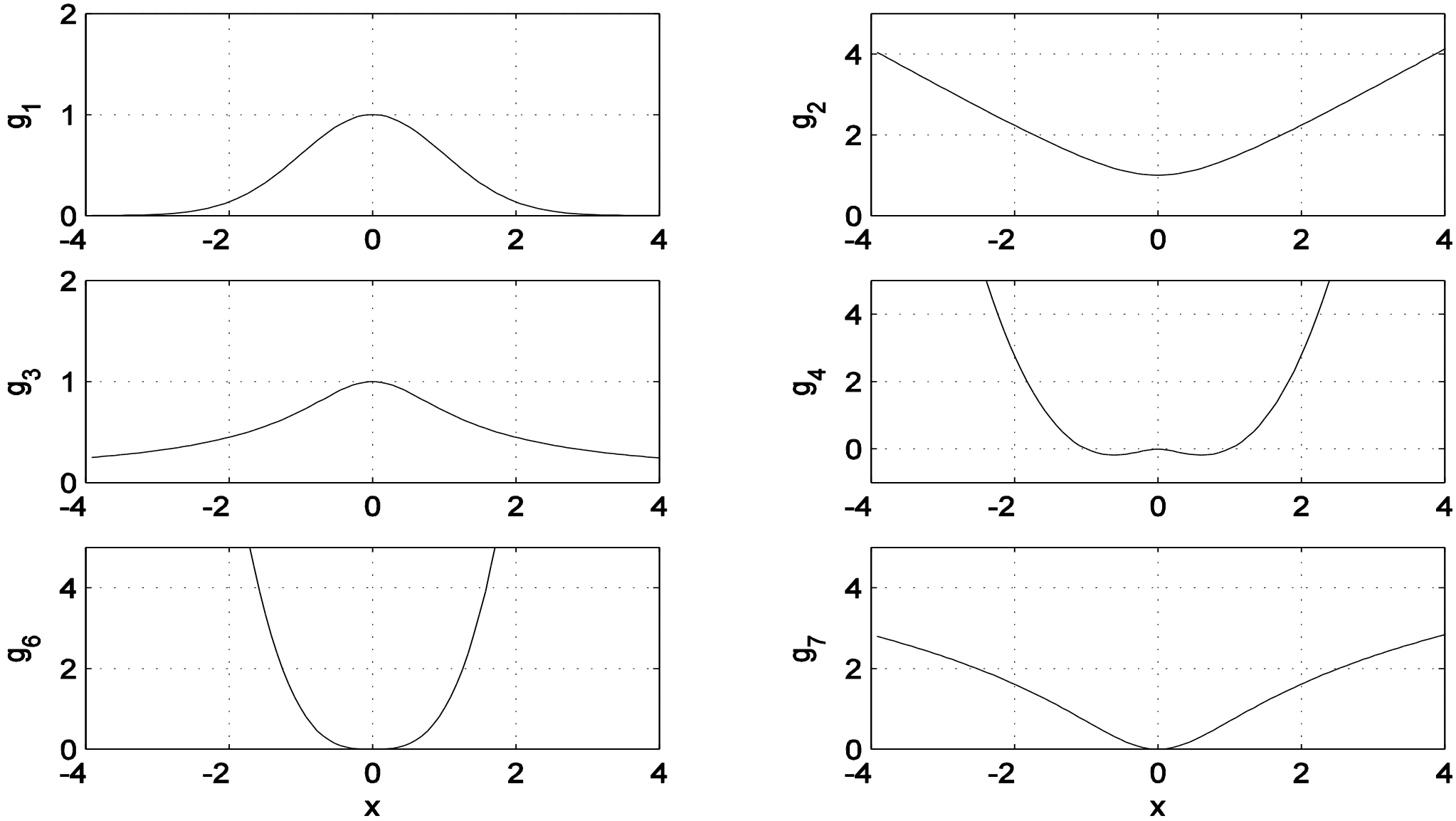
Thin plate spline: $g_4(\rho) = \rho^2 \log(\rho + \gamma)$

Linear: $g_5(\rho) = \rho$

Cubic: $g_6(\rho) = \rho^3$

Shifted Logarithm: $g_7(\rho) = \log\left(\rho^2 + \gamma^2\right)$

Forms of Radial Functions



Radial Basis Functions (RBFs)

→ **Selection of the radial function g** – in the case that g is selected to be the Gaussian function or the inverse multi-quadratic function, then the radial basis function approximator will have localization properties determined by the parameter γ.

→ **Approaches for the selection of the centers:**

- For a fixed batch of data, if the objective is interpolation of the data set, then the centers are set equal to the location of the sample data.
- The centers are specified on a lattice covering the region of interested D. May have to deal with the “curse of dimensionality”.
- The centers are estimated online as training data is accumulated. In this case, we have to deal with nonlinearly parameterized approximation.