PMR5406 Redes Neurais e Lógica Fuzzy

Aula 4
Radial Basis Function Networks

Baseado em:

Neural Networks, Simon Haykin, Prentice-Hall, 2^{nd} edition

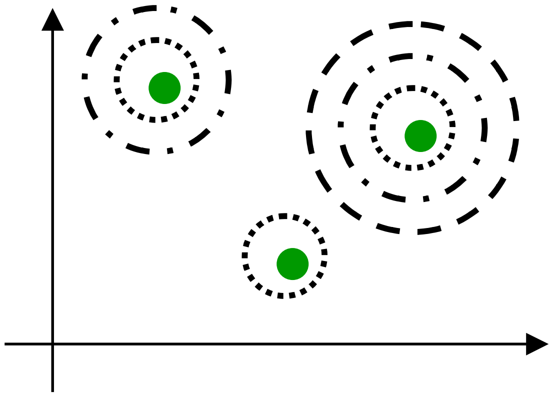
Slides do curso por Elena Marchiori, Vrije University

Radial-Basis Function Networks

- A function is approximated as a linear combination of radial basis functions (RBF). RBF's capture local behaviour of functions.

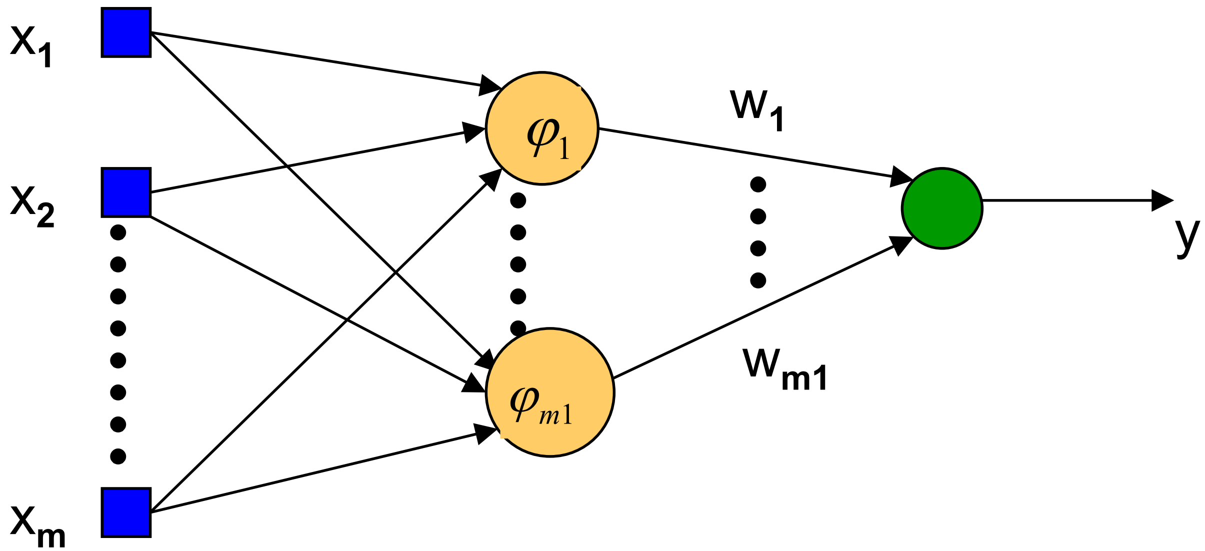
Biological motivation:

- RBF's represent local receptors:



ARCHITECTURE

- **Input layer**: source of nodes that connect the NN with its environment.



- **Hidden layer**: applies a non-linear transformation from the input space to the hidden space.
- **Output layer**: applies a linear transformation from the hidden space to the output space.

φ-separability of patterns

$$\varphi(x) = < \varphi_1(x), \ldots, \varphi_{m_1}(x) >$$

φ_i Hidden function

$\{\varphi_i(x)\}_{i=1}^{m_1}$ Hidden space

A (binary) partition, also called *dichotomy*, (C_1, C_2) of the training set C is φ-separable if there is a vector w of dimension m_1 such that:

$$w^T \varphi(x) > 0 \qquad x \in C_1$$

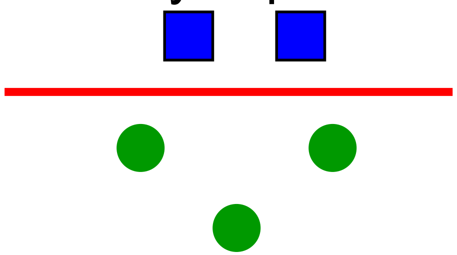
$$w^T \varphi(x) < 0 \qquad x \in C_2$$

Examples of φ-separability

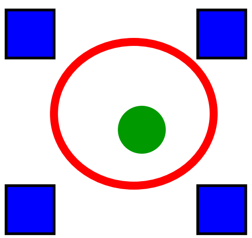
Separating surface: $\text{all } x \text{ such that } w^T \varphi(x) = 0$

Examples of separable partitions (C1,C2):

Linearly separable:

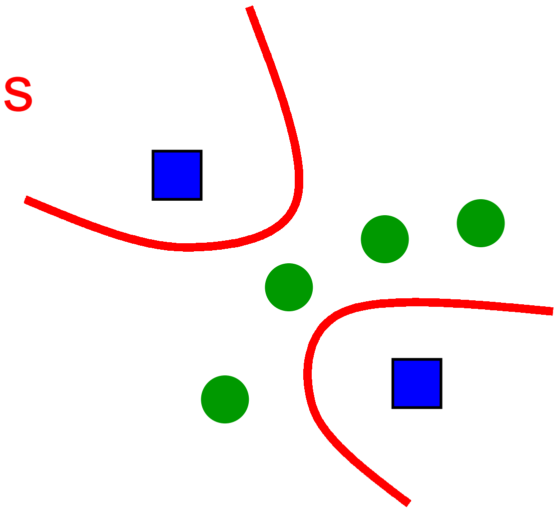


Polynomial type functions



Spherically separable:

Quadratically separable:



Cover's Theorem (1)

size of feature space $\varphi = <\varphi_1, \ldots, \varphi_{m1}>$

$P(N, m_1)$ - Probability that a particular partition (C1,C2) of the training set C picked at random is φ-separable

- **Cover's theorem. Under suitable assumptions on $C = \{x_1, \ldots, x_N\}$ and on the partitions (C1,C2) of *C*:**

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

Cover's Theorem (2)

- Essentially $P(N,m_1)$ is a cumulative binomial distribution that corresponds to the probability of picking N points $C = \{x_1, \ldots, x_N\}$ (each one has a probability $P(C_1)=P(C_2)=1/2$) which are φ-separable using m_1-1 or fewer degrees of freedom.

$$P(N,m_1) = \left(\frac{1}{2}\right)^{N-1}\left[\binom{N-1}{0} + \ldots + \binom{N-1}{m_1-1}\right]$$

Cover's Theorem (3)

- $P(N,m_1)$ tends to 1 with the increase of m_1 (size of feature space $\varphi = \langle \varphi 1, \ldots, \varphi m1 \rangle$).
- More flexibility with more functions in the feature space $\varphi = \langle \varphi 1, \ldots, \varphi m1 \rangle$

Cover's Theorem (4)

- 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.

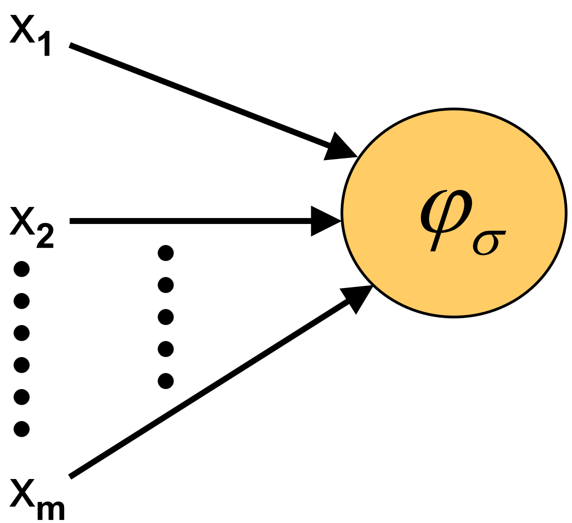
Corollary:

The expected maximum number of randomly assigned patterns that are linearly separable in a space of dimension m_1 is equal to $2m_1$

HIDDEN NEURON MODEL

- Hidden units: use a radial basis function

$\varphi_\sigma(\|x - t\|^2)$ the output depends on the distance of the input x from the center t



$\varphi_\sigma(\|x - t\|^2)$

t is called center
σ is called spread
center and spread are parameters

Hidden Neurons

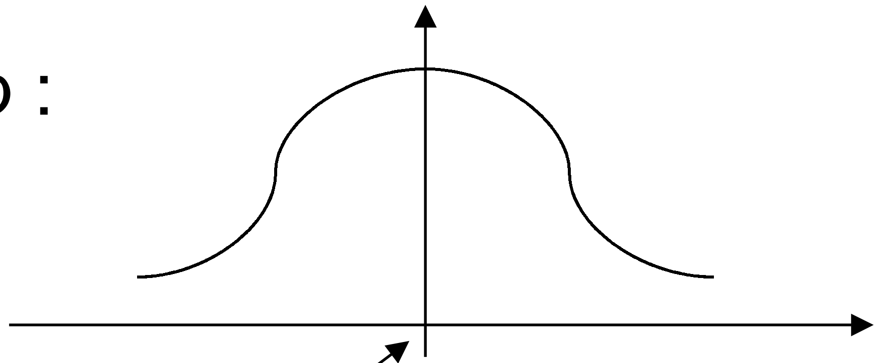
- A hidden neuron is more sensitive to data points near its center. This sensitivity may be tuned by adjusting the spread σ.

- Larger spread $\Rightarrow$ less sensitivity

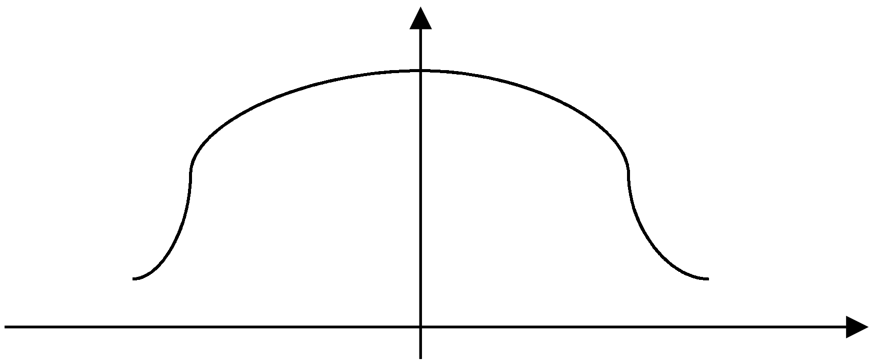
- Biological example: cochlear stereocilia cells have locally tuned frequency responses.

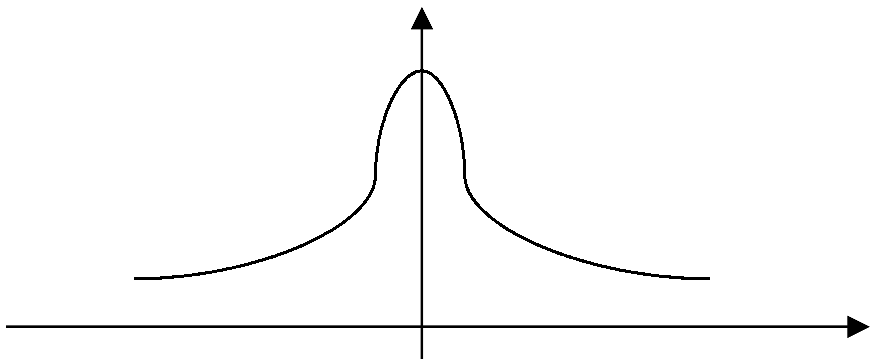
Gaussian Radial Basis Function φ

φ :



σ is a measure of how spread the curve is:





Types of φ

- Multiquadrics:

$$\varphi(r) = (r^2 + c^2)^{1/2} \qquad c > 0$$

$$r = \| x - t \|$$

- Inverse multiquadrics:

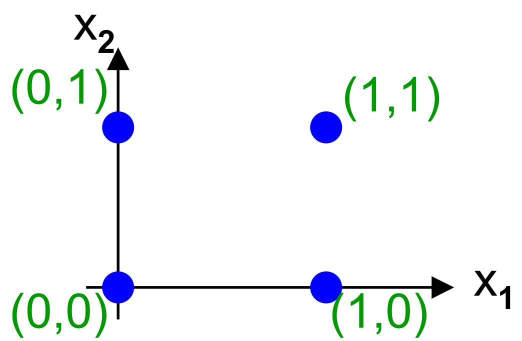
$$\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \qquad c > 0$$

- Gaussian functions:

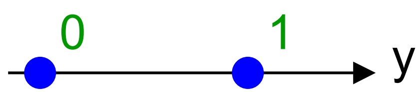
$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad \sigma > 0$$

Example: the XOR problem

- Input space:



- Output space:



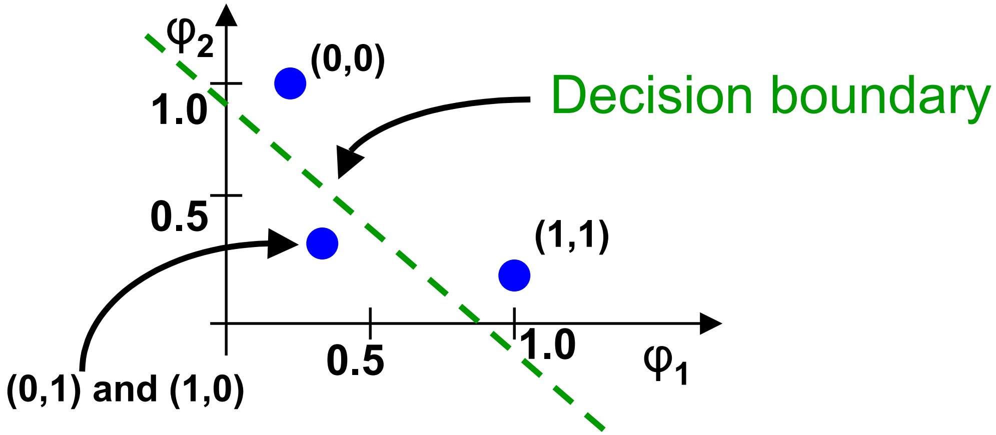
- Construct an RBF pattern classifier such that:
  (0,0) and (1,1) are mapped to 0, class C1
  (1,0) and (0,1) are mapped to 1, class C2

Example: the XOR problem

- In the feature (hidden) space:

$$\varphi_1(x_1, x_2) = e^{-\|x-t_1\|^2} \qquad t_1 = [1,1]^T$$

$$\varphi_2(x_1, x_2) = e^{-\|x-t_2\|^2} \qquad t_2 = [0,0]^T$$



- When mapped into the feature space < φ_1 , φ_2 >, C1 and C2 become *linearly separable.* So a linear classifier with φ_1(x) and φ_2(x) as inputs can be used to solve the XOR problem.

Learning Algorithms

- **Parameters** to be learnt are:
  - centers
  - spreads
  - weights
- Different learning algorithms

Learning Algorithm 1

- **Centers are selected at random**
  - **center locations** are chosen randomly from the training set
- **Spreads** are chosen by **normalization**:

$$\sigma = \frac{\text{Maximum distance between any 2 centers}}{\sqrt{\text{number of centers}}} = \frac{d_{max}}{\sqrt{m_1}}$$

$$\varphi_i\left(\|x - t_i\|^2\right) = \exp\left(-\frac{m_1}{d_{max}^2}\|x - t_i\|^2\right)$$

$$i \in [1, m_1]$$

Learning Algorithm 1

- Weights are found by means of **pseudo-inverse method**

$$w = \varphi^{+} d$$

Desired response

Pseudo-inverse of φ

$$\varphi = \{\varphi_{ji}\} \qquad \varphi_{ji} = \exp\left(-\frac{m_1}{d_{max}^2} \left\| x_j - t_i \right\|^2 \right)$$

$$j = 1, 2, \ldots, N \;, i = 1, 2, \ldots, m_1$$

Learning Algorithm 2

- ***Hybrid Learning Process:***
  - **Self-organized learning stage** for finding the centers
  - Spreads chosen by normalization
  - **Supervised learning stage** for finding the weights, using LMS algorithm

Learning Algorithm 2: Centers

- **K-means clustering algorithm for centers**

  1 **Initialization**: $t_k(0)$ random $k = 1, \ldots, m_1$

  2 **Sampling**: draw x from input space C

  3 **Similarity matching**: find index of best center

  $$\boxed{k(x) = \arg\min_k \left\| x(n) - t_k(n) \right\|}$$

  4 **Updating**: adjust centers

  $$t_k(n+1) = \begin{cases} t_k(n) + \eta\left[x(n) - t_k(n)\right] & \text{if } k = k(x) \\ t_k(n) & \text{otherwise} \end{cases}$$

  5. **Continuation**: increment *n* by *1*, goto *2* and continue until no noticeable changes of centers occur

Learning Algorithm 3

- **Supervised learning of all the parameters using the gradient descent method**
  - **Modify centers**

Instantaneous error function $\frac{1}{2}\sum_{i=1}^{N} e_i^2$

$$\Delta t_j = -\eta_{t_j} \frac{\partial E}{\partial t_j}$$

Learning rate for t_j

Depending on the specific function can be computed using the chain rule of calculus

Learning Algorithm 3

- **Modify spreads**

$$\Delta\sigma_{\mathrm{j}} = -\eta_{\sigma_j} \frac{\partial E}{\partial \sigma_{\mathrm{j}}}$$

- **Modify output weights**

$$\Delta \mathrm{w}_{\mathrm{ij}} = -\eta_{\mathrm{ij}} \frac{\partial E}{\partial \mathrm{w}_{\mathrm{ij}}}$$

Comparison with multilayer NN

RBF-Networks are used to perform complex (non-linear) pattern classification tasks.

Comparison between RBF networks and multilayer perceptrons:

- Both are examples of *non-linear layered feed-forward* networks.
- Both are *universal approximators*.
- Hidden layers:
  - RBF networks have one *single* hidden layer.
  - MLP networks may have *more* hidden layers.

Comparison with multilayer NN

- Neuron Models:
  - The computation nodes in the hidden layer of a RBF network are *different*. They serve a different purpose from those in the output layer.
  - Typically computation nodes of MLP in a hidden or output layer share a *common neuron model*.
- Linearity:
  - The hidden layer of RBF is *non-linear*, the output layer of RBF is *linear*.
  - Hidden and output layers of MLP are usually *non-linear*.

Comparison with multilayer NN

- Activation functions:
  - The argument of activation function of each hidden unit in a RBF NN computes the *Euclidean distance* between input vector and the center of that unit.
  - The argument of the activation function of each hidden unit in a MLP computes the *inner product* of input vector and the synaptic weight vector of that unit.
- Approximations:
  - RBF NN using Gaussian functions construct *local* approximations to non-linear I/O mapping.
  - MLP NN construct *global* approximations to non-linear I/O mapping.