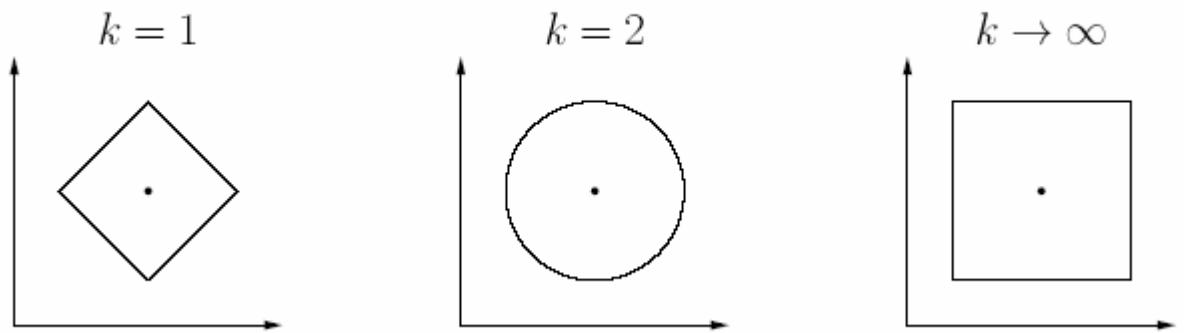
Distance Functions

Illustration of distance functions

$$d_k(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^k \right)^{\frac{1}{k}}$$

Well-known special cases from this family are:

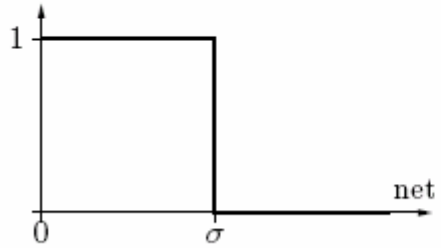
$k = 1$: Manhattan or city block distance,
$k = 2$: Euclidean distance,
$k \to \infty$: maximum distance, i.e. $d_\infty(\vec{x}, \vec{y}) = \max_{i=1}^{n}(x_i - y_i)$.



Radial Activation Functions

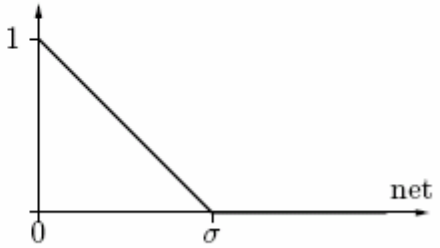
rectangle function:

$$f_{\text{act}}(\text{net}, \sigma) = \begin{cases} 0, & \text{if net} > \sigma, \\ 1, & \text{otherwise.} \end{cases}$$



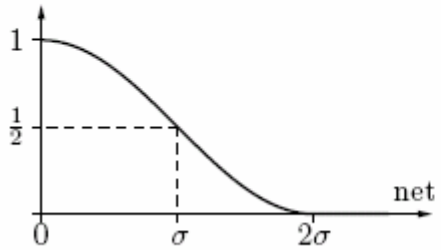
triangle function:

$$f_{\text{act}}(\text{net}, \sigma) = \begin{cases} 0, & \text{if net} > \sigma, \\ 1 - \frac{\text{net}}{\sigma}, & \text{otherwise.} \end{cases}$$



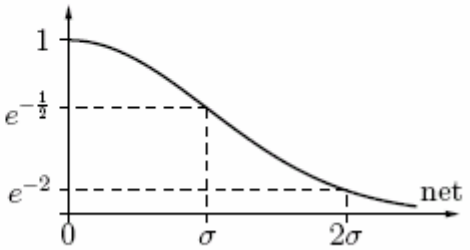
cosine until zero:

$$f_{\text{act}}(\text{net}, \sigma) = \begin{cases} 0, & \text{if net} > 2\sigma, \\ \frac{\cos(\frac{\pi}{2\sigma}\,\text{net})+1}{2}, & \text{otherwise.} \end{cases}$$



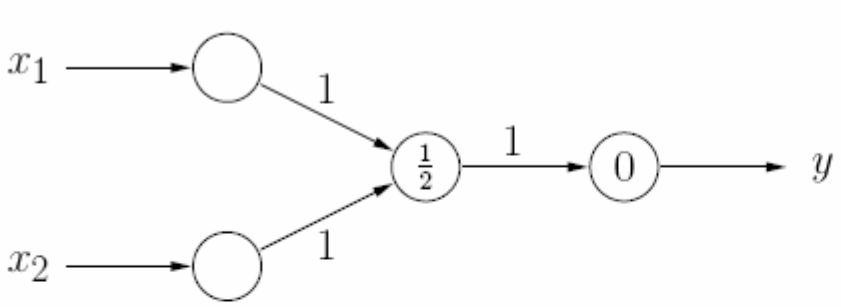
Gaussian function:

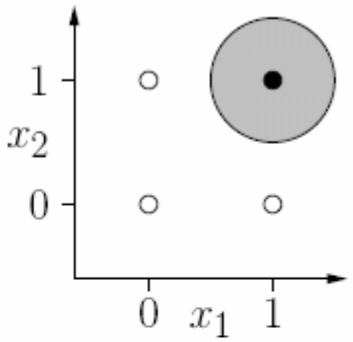
$$f_{\text{act}}(\text{net}, \sigma) = e^{-\frac{\text{net}^2}{2\sigma^2}}$$

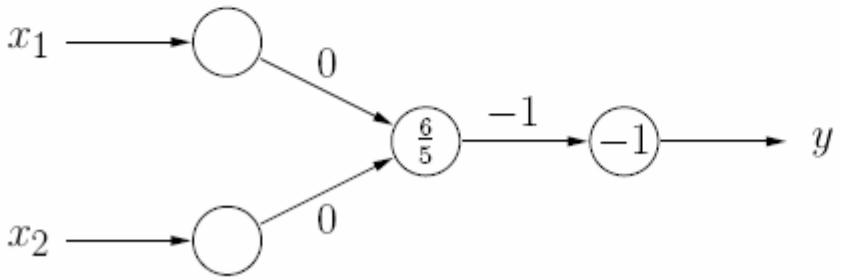


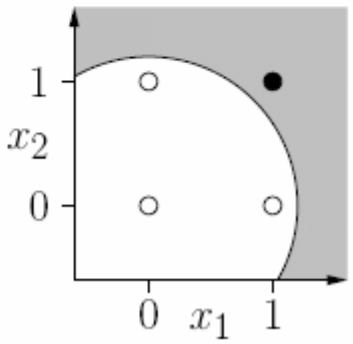
Radial Basis Function Networks: Examples

Radial basis function networks for the conjunction $x_1 \wedge x_2$







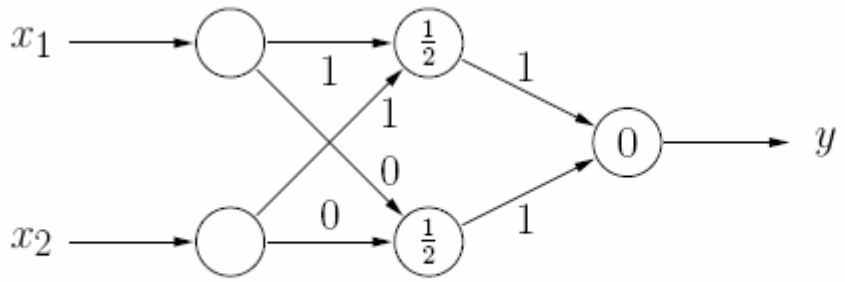


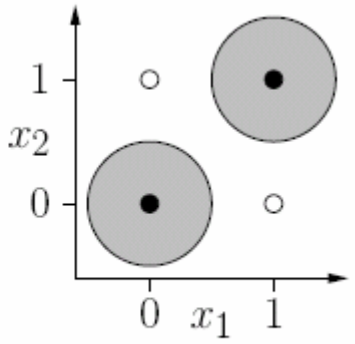
Radial Basis Function Networks: Examples

Radial basis function networks for the biimplication $x_1 \leftrightarrow x_2$

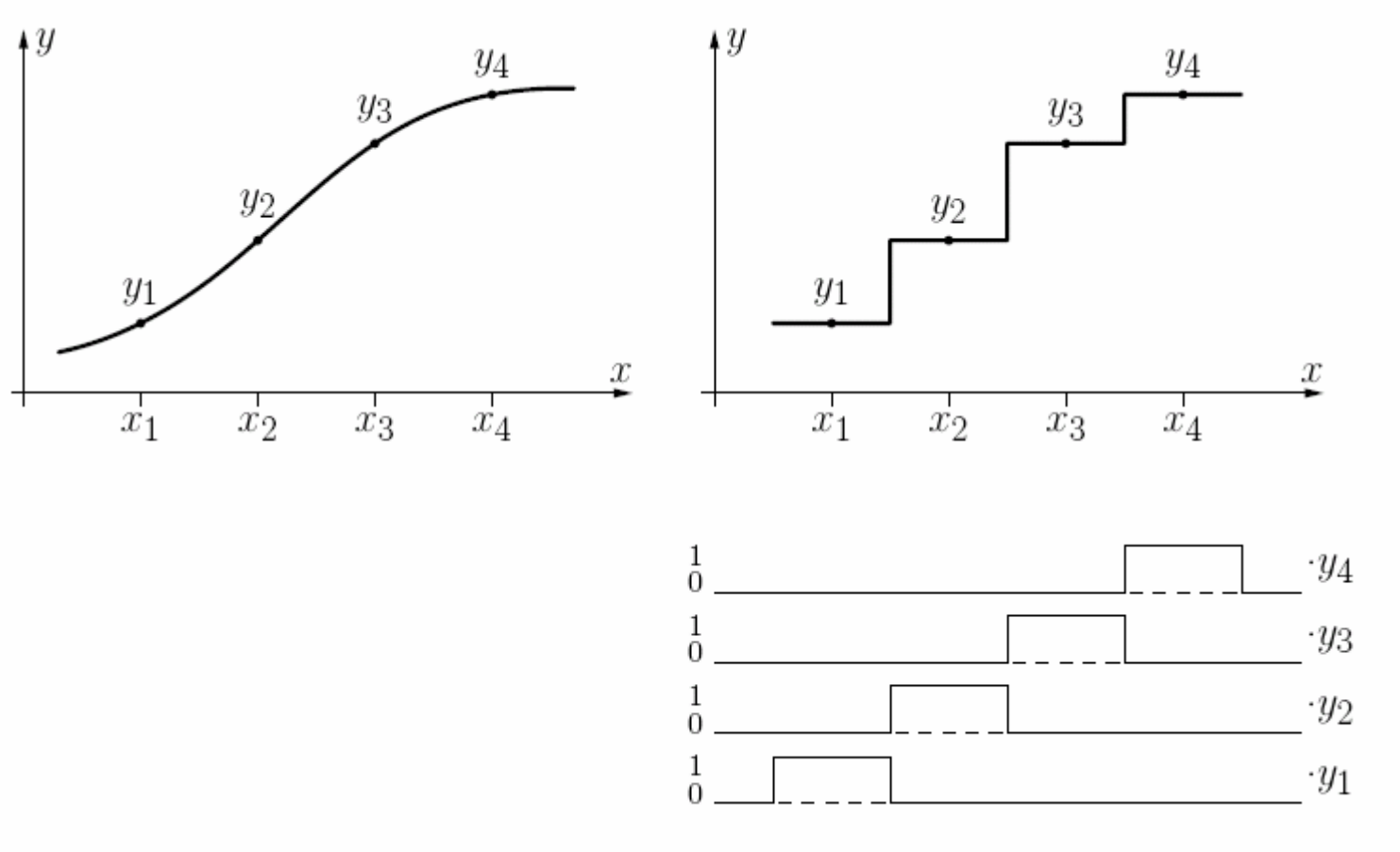
Idea: Logical decomposition

$$x_1 \leftrightarrow x_2 \quad \equiv \quad (x_1 \wedge x_2) \vee \neg(x_1 \vee x_2)$$

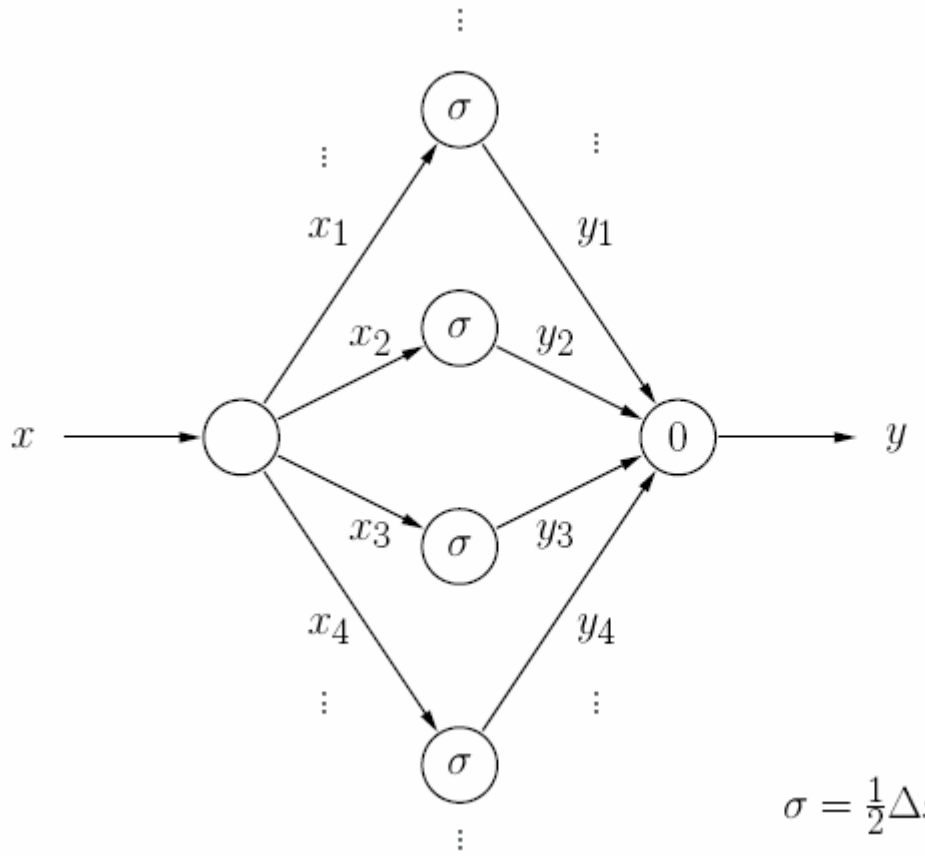




Radial Basis Function Networks: Function Approximation

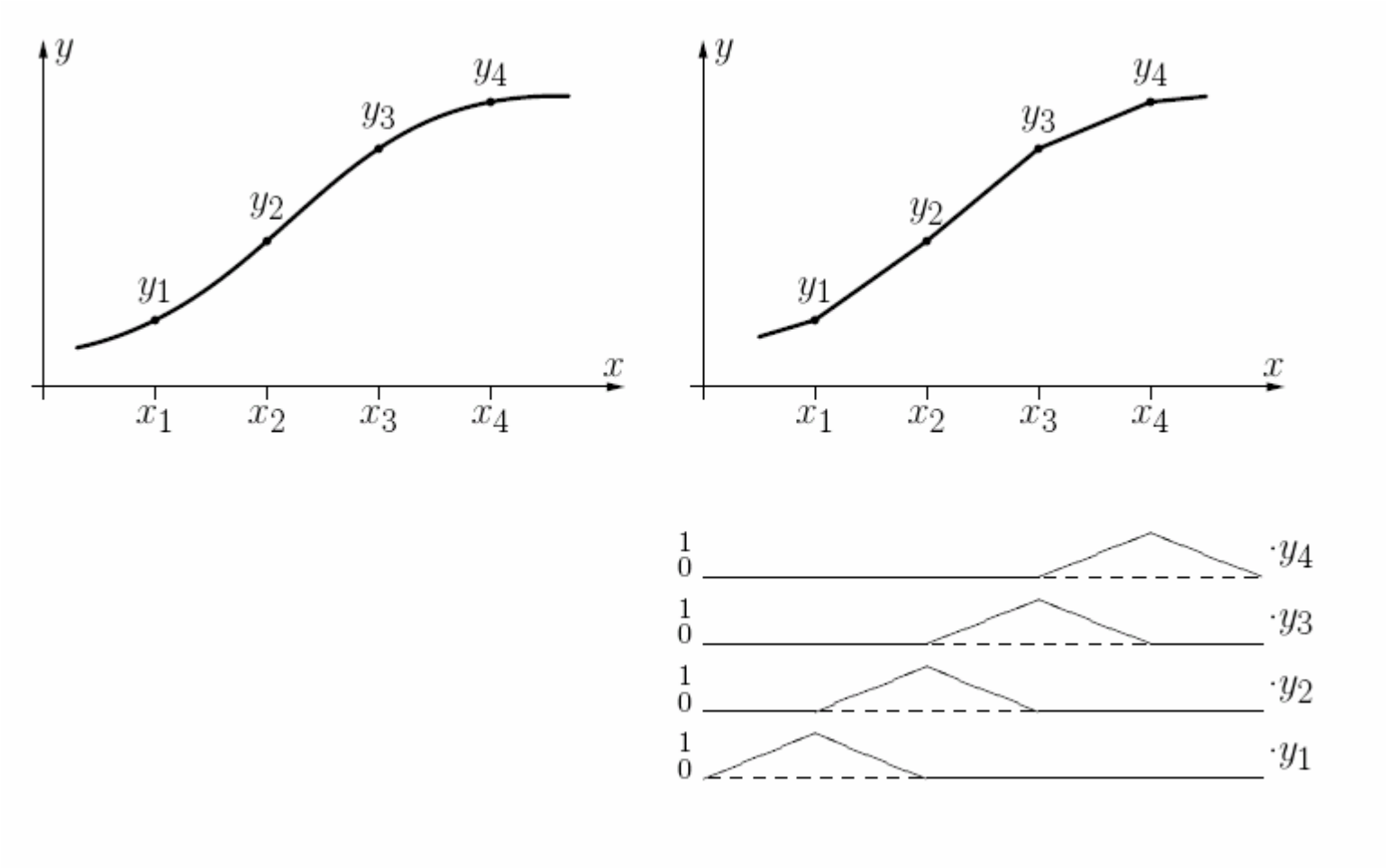


Radial Basis Function Networks: Function Approximation

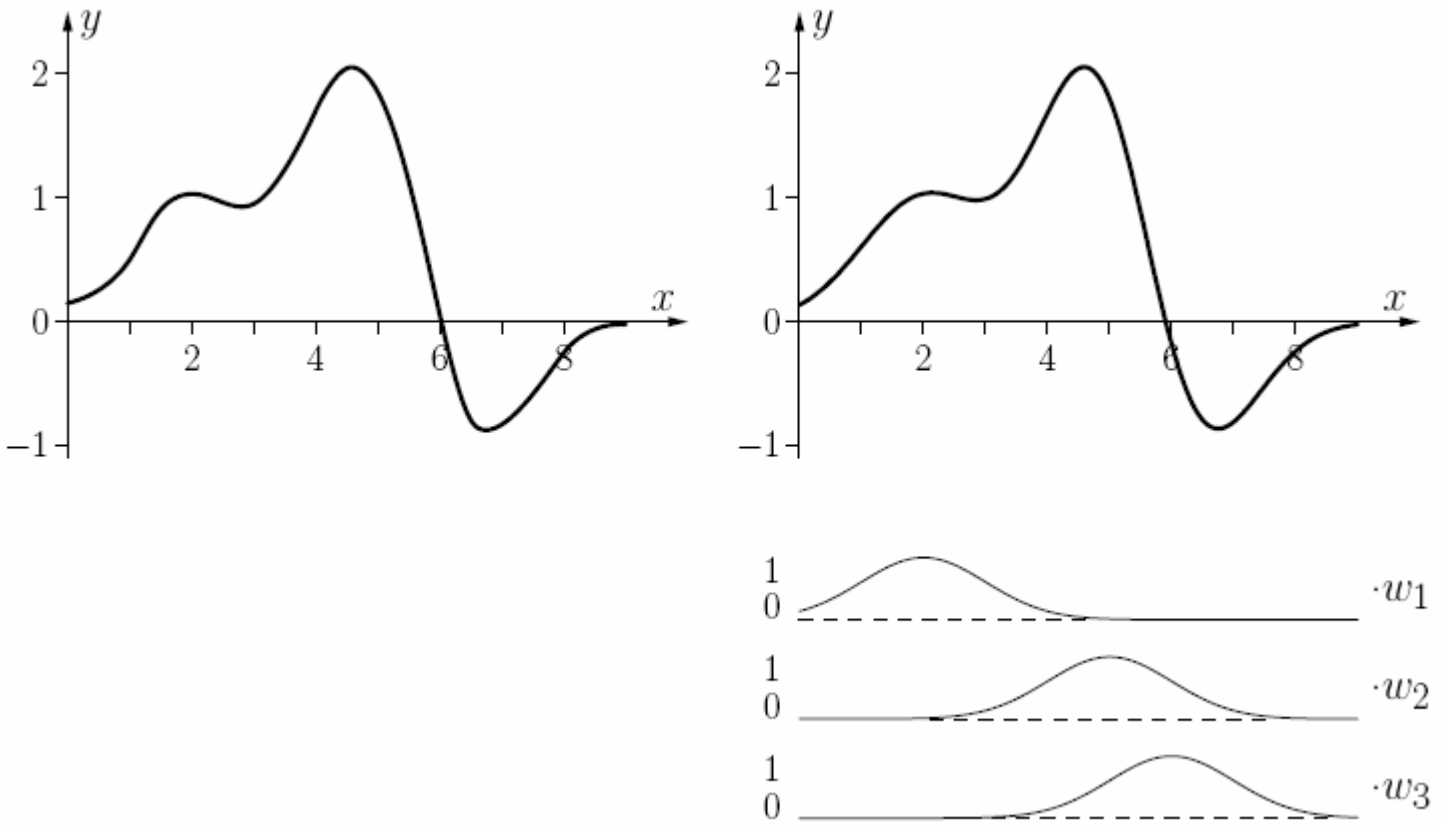


$$\sigma = \frac{1}{2}\Delta x = \frac{1}{2}(x_{i+1} - x_i)$$

Radial Basis Function Networks: Function Approximation

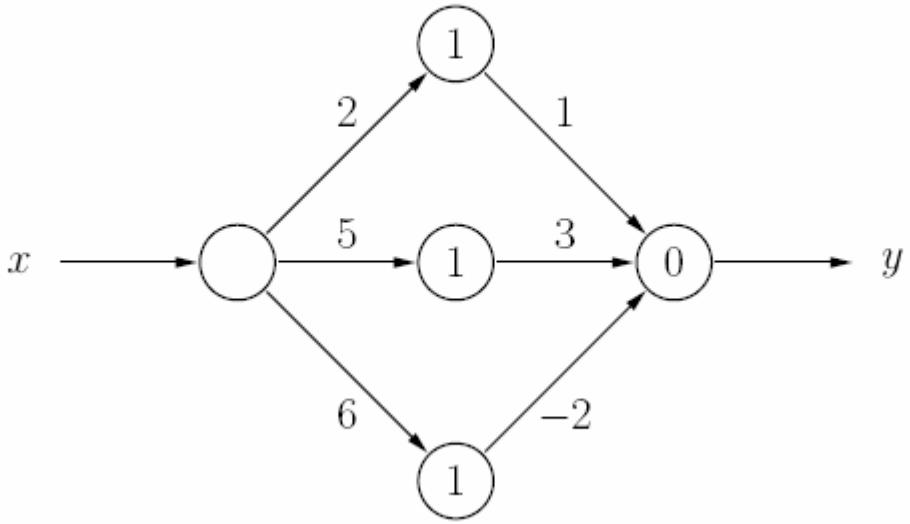


Radial Basis Function Networks: Function Approximation



Radial Basis Function Networks: Function Approximation

Radial basis function network for a sum of three Gaussian functions



Radial Basis Function Networks: Initialization

Let $L_{\text{fixed}} = \{l_1, \ldots, l_m\}$ be a fixed learning task,
consisting of m training patterns $l = (\vec{\imath}^{\,(l)}, \vec{o}^{\,(l)})$.

Simple radial basis function network:
One hidden neuron v_k, $k = 1, \ldots, m$, for each training pattern:

$$\forall k \in \{1, \ldots, m\}: \qquad \vec{w}_{v_k} = \vec{\imath}^{\,(l_k)}.$$

If the activation function is the Gaussian function,
the radii σ_k are chosen heuristically

$$\forall k \in \{1, \ldots, m\}: \qquad \sigma_k = \frac{d_{\max}}{\sqrt{2m}},$$

where

$$d_{\max} = \max_{l_j, l_k \in L_{\text{fixed}}} d\left(\vec{\imath}^{\,(l_j)}, \vec{\imath}^{\,(l_k)}\right).$$

Radial Basis Function Networks: Initialization

Initializing the connections from the hidden to the output neurons

$$\forall u : \sum_{k=1}^{m} w_{uv_m} \mathrm{out}_{v_m}^{(l)} - \theta_u = o_u^{(l)} \qquad \text{or abbreviated} \qquad \mathbf{A} \cdot \vec{w}_u = \vec{o}_u,$$

where $\vec{o}_u = (o_u^{(l_1)}, \ldots, o_u^{(l_m)})^T$ is the vector of desired outputs, $\theta_u = 0$, and

$$\mathbf{A} = \begin{pmatrix} \mathrm{out}_{v_1}^{(l_1)} & \mathrm{out}_{v_2}^{(l_1)} & \ldots & \mathrm{out}_{v_m}^{(l_1)} \\ \mathrm{out}_{v_1}^{(l_2)} & \mathrm{out}_{v_2}^{(l_2)} & \ldots & \mathrm{out}_{v_m}^{(l_2)} \\ \vdots & \vdots & & \vdots \\ \mathrm{out}_{v_1}^{(l_m)} & \mathrm{out}_{v_2}^{(l_m)} & \ldots & \mathrm{out}_{v_m}^{(l_m)} \end{pmatrix}.$$

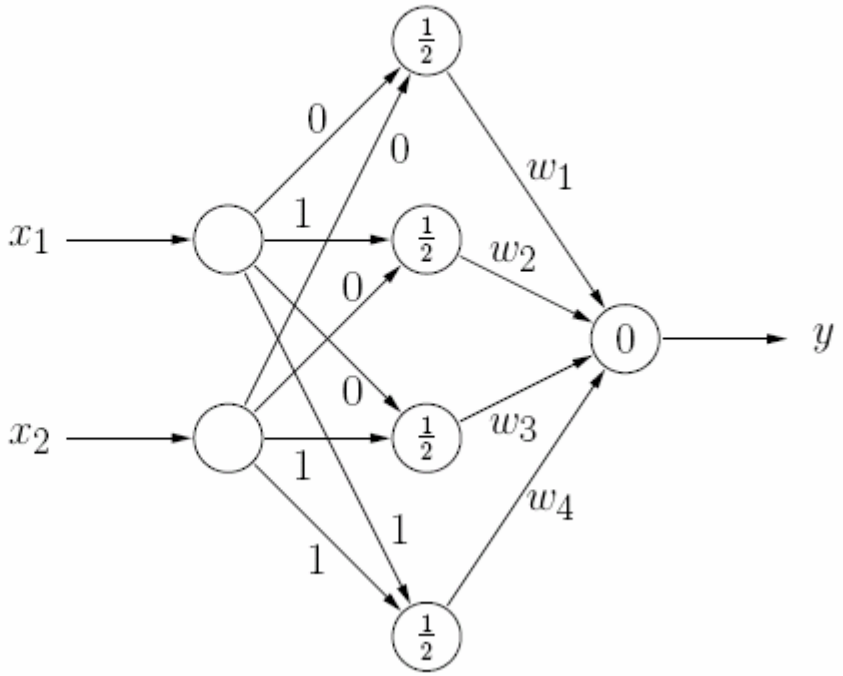
This is a linear equation system, that can be solved by inverting the matrix $\mathbf{A}$:

$$\vec{w}_u = \mathbf{A}^{-1} \cdot \vec{o}_u.$$

RBFN Initialization: Example

Simple radial basis function network for the biimplication $x_1 \leftrightarrow x_2$

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	1



RBFN Initialization: Example

Simple radial basis function network for the biimplication $x_1 \leftrightarrow x_2$

$$\mathbf{A} = \begin{pmatrix} 1 & e^{-2} & e^{-2} & e^{-4} \\ e^{-2} & 1 & e^{-4} & e^{-2} \\ e^{-2} & e^{-4} & 1 & e^{-2} \\ e^{-4} & e^{-2} & e^{-2} & 1 \end{pmatrix} \qquad \mathbf{A}^{-1} = \begin{pmatrix} \frac{a}{D} & \frac{b}{D} & \frac{b}{D} & \frac{c}{D} \\ \frac{b}{D} & \frac{a}{D} & \frac{c}{D} & \frac{b}{D} \\ \frac{b}{D} & \frac{c}{D} & \frac{a}{D} & \frac{b}{D} \\ \frac{c}{D} & \frac{b}{D} & \frac{b}{D} & \frac{a}{D} \end{pmatrix}$$

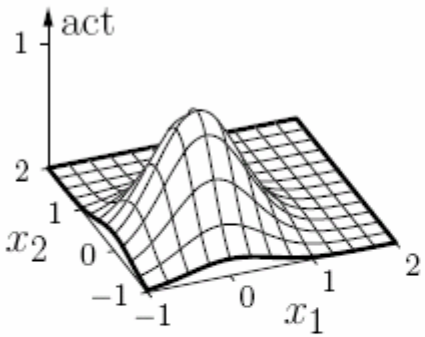
where

$$\begin{aligned} D &= 1 - 4e^{-4} + 6e^{-8} - 4e^{-12} + e^{-16} &&\approx 0.9287 \\ a &= 1 - 2e^{-4} + e^{-8} &&\approx 0.9637 \\ b &= -e^{-2} + 2e^{-6} - e^{-10} &&\approx -0.1304 \\ c &= e^{-4} - 2e^{-8} + e^{-12} &&\approx 0.0177 \end{aligned}$$

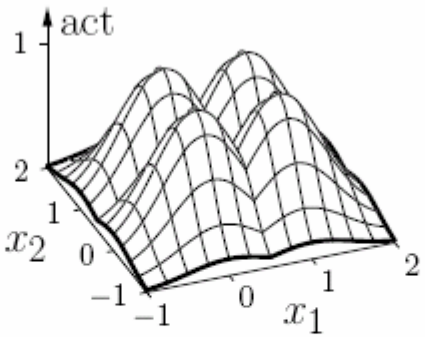
$$\vec{w}_u = \mathbf{A}^{-1} \cdot \vec{o}_u = \frac{1}{D} \begin{pmatrix} a+c \\ 2b \\ 2b \\ a+c \end{pmatrix} \approx \begin{pmatrix} 1.0567 \\ -0.2809 \\ -0.2809 \\ 1.0567 \end{pmatrix}$$

RBFN Initialization: Example

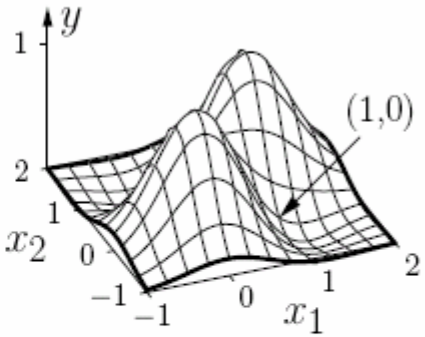
Simple radial basis function network for the biimplication $x_1 \leftrightarrow x_2$



single basis function



all basis functions



output

- Initialization leads already to a perfect solution of the learning task.
- Subsequent training is not necessary.

Radial Basis Function Networks: Initialization

Normal radial basis function networks:

Select subset of k training patterns as centers.

$$\mathbf{A} = \begin{pmatrix} 1 & \mathrm{out}_{v_1}^{(l_1)} & \mathrm{out}_{v_2}^{(l_1)} & \ldots & \mathrm{out}_{v_k}^{(l_1)} \\ 1 & \mathrm{out}_{v_1}^{(l_2)} & \mathrm{out}_{v_2}^{(l_2)} & \ldots & \mathrm{out}_{v_k}^{(l_2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \mathrm{out}_{v_1}^{(l_m)} & \mathrm{out}_{v_2}^{(l_m)} & \ldots & \mathrm{out}_{v_k}^{(l_m)} \end{pmatrix} \qquad \mathbf{A} \cdot \vec{w}_u = \vec{o}_u$$

Compute (Moore–Penrose) pseudo inverse:

$$\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T.$$

The weights can then be computed by

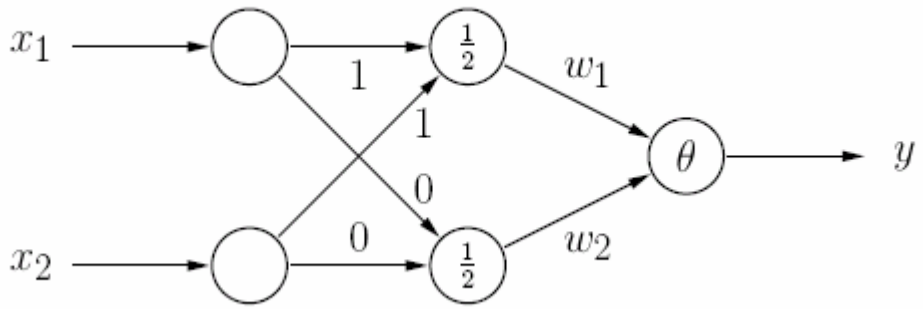
$$\vec{w}_u = \mathbf{A}^+ \cdot \vec{o}_u = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \cdot \vec{o}_u$$

RBFN Initialization: Example

Normal radial basis function network for the biimplication $x_1 \leftrightarrow x_2$

Select two training patterns:

- $l_1 = (\vec{\imath}^{(l_1)}, \vec{o}^{(l_1)}) = ((0,0),(1))$
- $l_4 = (\vec{\imath}^{(l_4)}, \vec{o}^{(l_4)}) = ((1,1),(1))$



RBFN Initialization: Example

Normal radial basis function network for the biimplication $x_1 \leftrightarrow x_2$

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & e^{-4} \\ 1 & e^{-2} & e^{-2} \\ 1 & e^{-2} & e^{-2} \\ 1 & e^{-4} & 1 \end{pmatrix} \qquad \mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T = \begin{pmatrix} a & b & b & a \\ c & d & d & e \\ e & d & d & c \end{pmatrix}$$

where

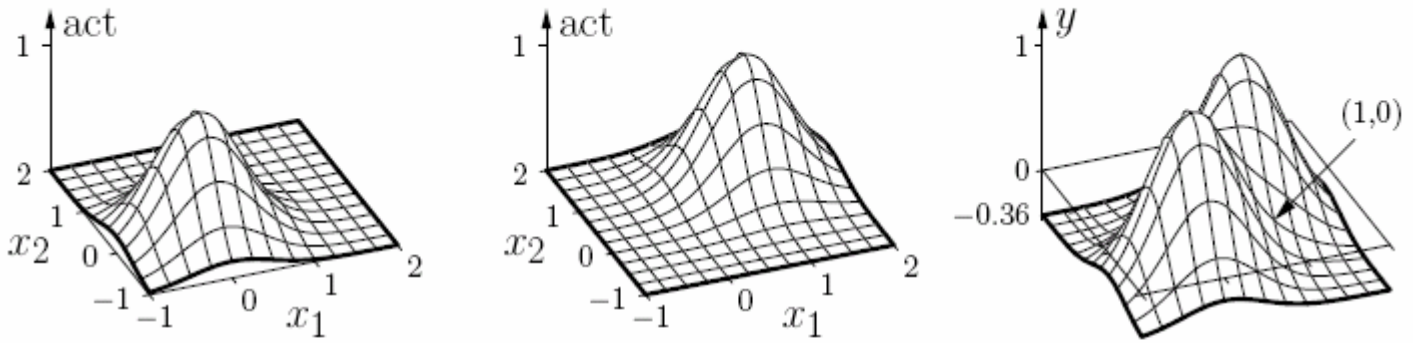
$$a \approx -0.1810, \quad b \approx 0.6810,$$
$$c \approx 1.1781, \quad d \approx -0.6688, \quad e \approx 0.1594.$$

Resulting weights:

$$\vec{w}_u = \begin{pmatrix} -\theta \\ w_1 \\ w_2 \end{pmatrix} = \mathbf{A}^+ \cdot \vec{o}_u \approx \begin{pmatrix} -0.3620 \\ 1.3375 \\ 1.3375 \end{pmatrix}.$$

RBFN Initialization: Example

Normal radial basis function network for the biimplication $x_1 \leftrightarrow x_2$



basis function (0,0) basis function (1,1) output

- Initialization leads already to a perfect solution of the learning task.
- This is an accident, because the linear equation system is not over-determined, due to linearly dependent equations.

Radial Basis Function Networks: Initialization

Finding appropriate centers for the radial basis functions

One approach: **k-means clustering**

- Select randomly k training patterns as centers.
- Assign to each center those training patterns that are closest to it.
- Compute new centers as the center of gravity of the assigned training patterns
- Repeat previous two steps until convergence,
  i.e., until the centers do not change anymore.
- Use resulting centers for the weight vectors of the hidden neurons.

Alternative approach: **learning vector quantization**

Radial Basis Function Networks: Training

Training radial basis function networks:
Derivation of update rules is analogous to that of multilayer perceptrons.

Weights from the hidden to the output neurons.

Gradient:

$$\vec{\nabla}_{\vec{w}_u} e_u^{(l)} = \frac{\partial e_u^{(l)}}{\partial \vec{w}_u} = -2(o_u^{(l)} - \mathrm{out}_u^{(l)})\, \vec{\mathrm{in}}_u^{(l)},$$

Weight update rule:

$$\Delta \vec{w}_u^{(l)} = -\frac{\eta_3}{2} \vec{\nabla}_{\vec{w}_u} e_u^{(l)} = \eta_3 (o_u^{(l)} - \mathrm{out}_u^{(l)})\, \vec{\mathrm{in}}_u^{(l)}$$

(Two more learning rates are needed for the center coordinates and the radii.)

Radial Basis Function Networks: Training

Training radial basis function networks:
Center coordinates (weights from the input to the hidden neurons).

Gradient:

$$\vec{\nabla}_{\vec{w}_v} e^{(l)} = \frac{\partial e^{(l)}}{\partial \vec{w}_v} = -2 \sum_{s \in \mathrm{succ}(v)} (o_s^{(l)} - \mathrm{out}_s^{(l)}) w_{su} \frac{\partial \mathrm{out}_v^{(l)}}{\partial \mathrm{net}_v^{(l)}} \frac{\partial \mathrm{net}_v^{(l)}}{\partial \vec{w}_v}$$

Weight update rule:

$$\Delta \vec{w}_v^{(l)} = -\frac{\eta_1}{2} \vec{\nabla}_{\vec{w}_v} e^{(l)} = \eta_1 \sum_{s \in \mathrm{succ}(v)} (o_s^{(l)} - \mathrm{out}_s^{(l)}) w_{sv} \frac{\partial \mathrm{out}_v^{(l)}}{\partial \mathrm{net}_v^{(l)}} \frac{\partial \mathrm{net}_v^{(l)}}{\partial \vec{w}_v}$$

Radial Basis Function Networks: Training

Training radial basis function networks:
Center coordinates (weights from the input to the hidden neurons).

Special case: **Euclidean distance**

$$\frac{\partial \operatorname{net}_v^{(l)}}{\partial \vec{w}_v} = \left(\sum_{i=1}^{n} (w_{vp_i} - \operatorname{out}_{p_i}^{(l)})^2 \right)^{-\frac{1}{2}} (\vec{w}_v - \vec{\operatorname{in}}_v^{(l)}).$$

Special case: **Gaussian activation function**

$$\frac{\partial \operatorname{out}_v^{(l)}}{\partial \operatorname{net}_v^{(l)}} = \frac{\partial f_{\text{act}}(\operatorname{net}_v^{(l)}, \sigma_v)}{\partial \operatorname{net}_v^{(l)}} = \frac{\partial}{\partial \operatorname{net}_v^{(l)}} e^{-\frac{\left(\operatorname{net}_v^{(l)}\right)^2}{2\sigma_v^2}} = -\frac{\operatorname{net}_v^{(l)}}{\sigma_v^2} e^{-\frac{\left(\operatorname{net}_v^{(l)}\right)^2}{2\sigma_v^2}}.$$

Radial Basis Function Networks: Training

Training radial basis function networks:
Radii of radial basis functions.

Gradient:

$$\frac{\partial e^{(l)}}{\partial \sigma_v} = -2 \sum_{s \in \mathrm{succ}(v)} (o_s^{(l)} - \mathrm{out}_s^{(l)}) w_{su} \frac{\partial \mathrm{out}_v^{(l)}}{\partial \sigma_v}.$$

Weight update rule:

$$\Delta \sigma_v^{(l)} = -\frac{\eta_2}{2} \frac{\partial e^{(l)}}{\partial \sigma_v} = \eta_2 \sum_{s \in \mathrm{succ}(v)} (o_s^{(l)} - \mathrm{out}_s^{(l)}) w_{sv} \frac{\partial \mathrm{out}_v^{(l)}}{\partial \sigma_v}.$$

Special case: **Gaussian activation function**

$$\frac{\partial \mathrm{out}_v^{(l)}}{\partial \sigma_v} = \frac{\partial}{\partial \sigma_v} e^{-\frac{\left(\mathrm{net}_v^{(l)}\right)^2}{2\sigma_v^2}} = \frac{\left(\mathrm{net}_v^{(l)}\right)^2}{\sigma_v^3} e^{-\frac{\left(\mathrm{net}_v^{(l)}\right)^2}{2\sigma_v^2}}.$$

Radial Basis Function Networks: Generalization

Generalization of the distance function

Idea: Use anisotropic distance function.

Example: **Mahalanobis distance**

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})}.$$

Example: **biimplication**