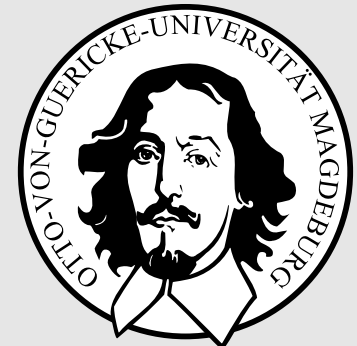


Neuronale Netze

Prof. Dr. Rudolf Kruse

Computational Intelligence
Institut für Wissens- und Sprachverarbeitung
Fakultät für Informatik
kruse@iws.cs.uni-magdeburg.de



Radiale-Basisfunktionen-Netze

Eigenschaften von Radiale-Basisfunktionen-Netzen (RBF-Netzen)

- RBF-Netze sind streng geschichtete, vorwärtsbetriebene neuronale Netze mit genau einer versteckten Schicht.
- Als Netzeingabe- und Aktivierungsfunktion werden radiale Basisfunktionen verwendet.
- Jedes Neuron erhält eine Art “Einzugsgebiet”.
- Die Gewichte der Verbindungen von der Eingabeschicht zu einem Neuron geben das Zentrum an.

Radiale-Basisfunktionen-Netze

Ein **radiale-Basisfunktionen-Netz (RBF-Netz)** ist ein neuronales Netz mit einem Graph $G = (U, C)$, das die folgenden Bedingungen erfüllt:

$$(i) \quad U_{\text{in}} \cap U_{\text{out}} = \emptyset,$$

$$(ii) \quad C = (U_{\text{in}} \times U_{\text{hidden}}) \cup C', \quad C' \subseteq (U_{\text{hidden}} \times U_{\text{out}})$$

Die Netzeingabefunktion jedes versteckten Neurons ist eine **Abstandsfunktion** zwischen dem Eingabevektor und dem Gewichtsvektor, d.h.

$$\forall u \in U_{\text{hidden}} : \quad f_{\text{net}}^{(u)}(\vec{w}_u, \vec{\text{in}}_u) = d(\vec{w}_u, \vec{\text{in}}_u),$$

wobei $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_0^+$ eine Funktion ist, die $\forall \vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^n$: erfüllt:

$$(i) \quad d(\vec{x}, \vec{y}) = 0 \quad \Leftrightarrow \quad \vec{x} = \vec{y},$$

$$(ii) \quad d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x}) \quad (\text{Symmetrie}),$$

$$(iii) \quad d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) \quad (\text{Dreiecksungleichung}).$$

Veranschaulichung von Abstandsfunktionen

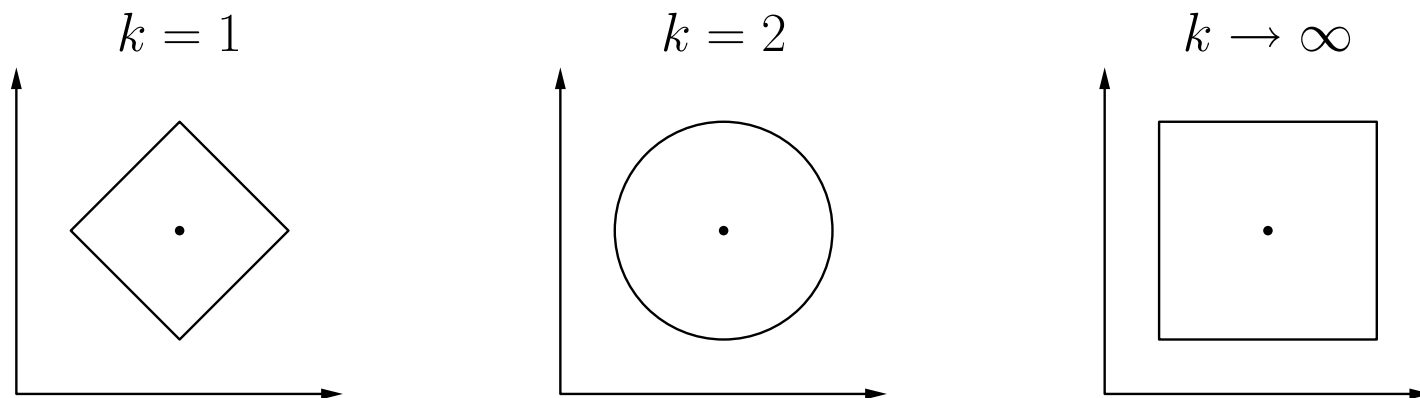
$$d_k(\vec{x}, \vec{y}) = \left(\sum_{i=1}^n (x_i - y_i)^k \right)^{\frac{1}{k}}$$

Bekannte Spezialfälle dieser Familie sind:

$k = 1$: Manhattan-Abstand ,

$k = 2$: Euklidischer Abstand,

$k \rightarrow \infty$: Maximum-Abstand, d.h. $d_\infty(\vec{x}, \vec{y}) = \max_{i=1}^n |x_i - y_i|$.



(alle Punkte auf dem Kreis bzw. den Vierecken haben denselben Abstand zum Mittelpunkt, entsprechend der jeweiligen Abstandsfunktion)

Radiale-Basisfunktionen-Netze

Die Netzeingabefunktion der Ausgabeneuronen ist die gewichtete Summe ihrer Eingaben, d.h.

$$\forall u \in U_{\text{out}} : f_{\text{net}}^{(u)}(\vec{w}_u, \vec{\text{in}}_u) = \vec{w}_u \vec{\text{in}}_u = \sum_{v \in \text{pred}(u)} w_{uv} \text{out}_v .$$

Die Aktivierungsfunktion jedes versteckten Neurons ist eine sogenannte **radiale Funktion**, d.h. eine monoton fallende Funktion

$$f : \mathbb{R}_0^+ \rightarrow [0, 1] \quad \text{with} \quad f(0) = 1 \quad \text{and} \quad \lim_{x \rightarrow \infty} f(x) = 0 .$$

Die Aktivierungsfunktion jedes Ausgabeneurons ist eine lineare Funktion

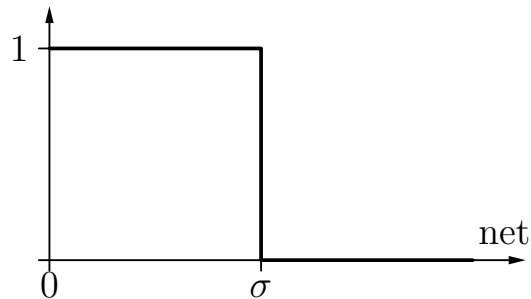
$$f_{\text{act}}^{(u)}(\text{net}_u, \theta_u) = \text{net}_u - \theta_u .$$

(Die lineare Aktivierungsfunktion ist wichtig für die Initialisierung.)

Radiale Aktivierungsfunktionen

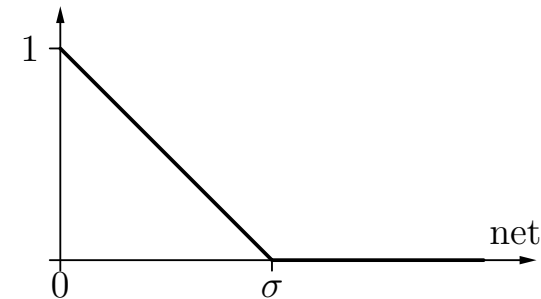
Rechteckfunktion:

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



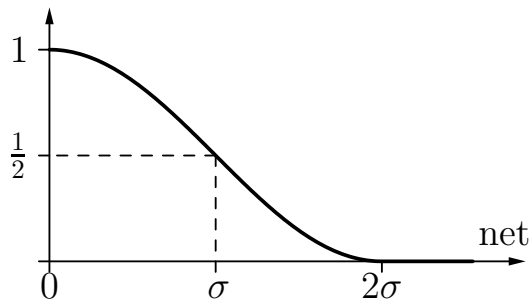
Dreiecksfunktion:

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



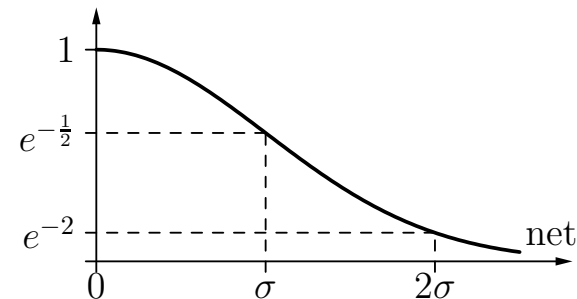
Kosinus bis Null:

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



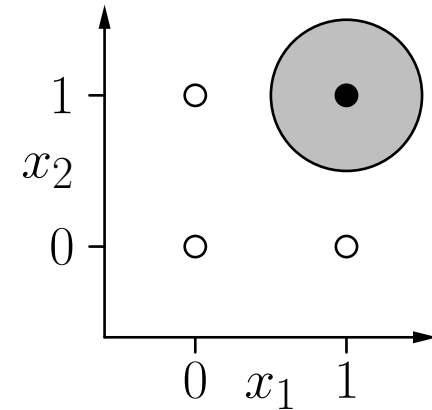
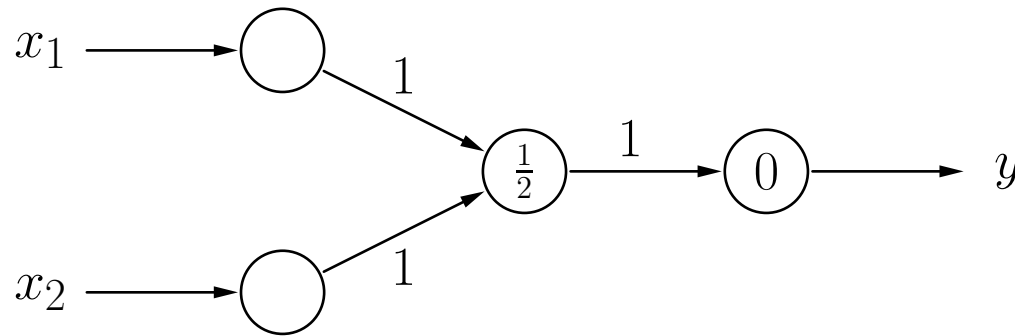
Gaußsche Funktion:

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



Radiale-Basisfunktionen-Netze: Beispiele

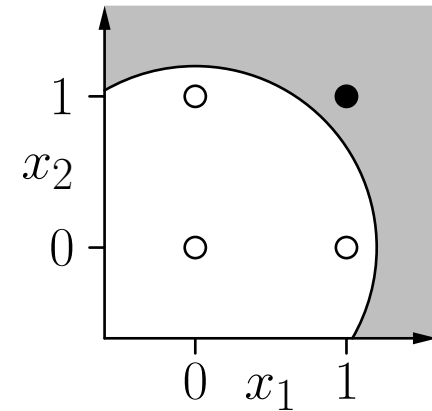
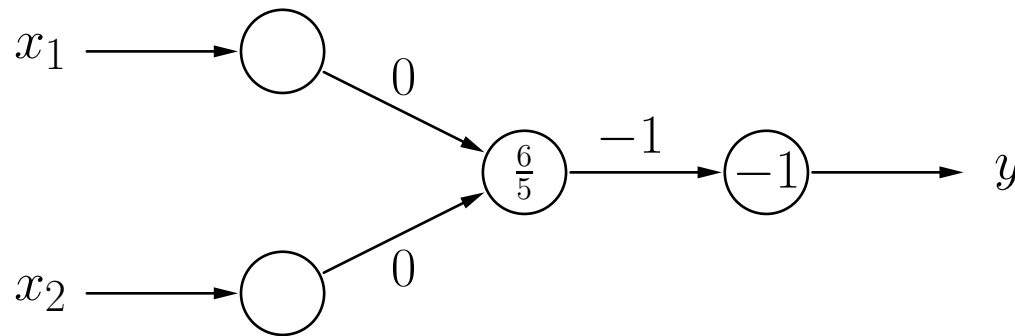
Radiale-Basisfunktionen-Netz für die Konjunktion $x_1 \wedge x_2$



- $(1,1)$ ist Zentrum
- Referenzradius ist $\frac{1}{2}$
- Euklidischer Abstand
- Rechteckfunktion als Aktivierung
- Biaswert 0 im Ausgabeneuron

Radiale-Basisfunktionen-Netze: Beispiele

Radiale-Basisfunktionen-Netz für die Konjunktion $x_1 \wedge x_2$



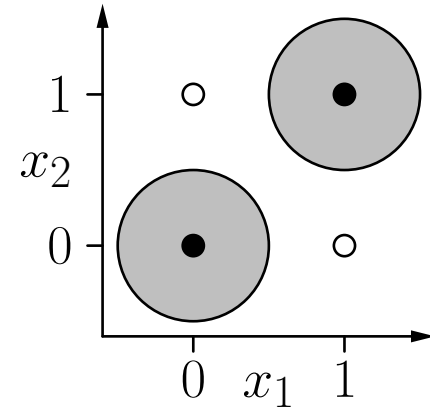
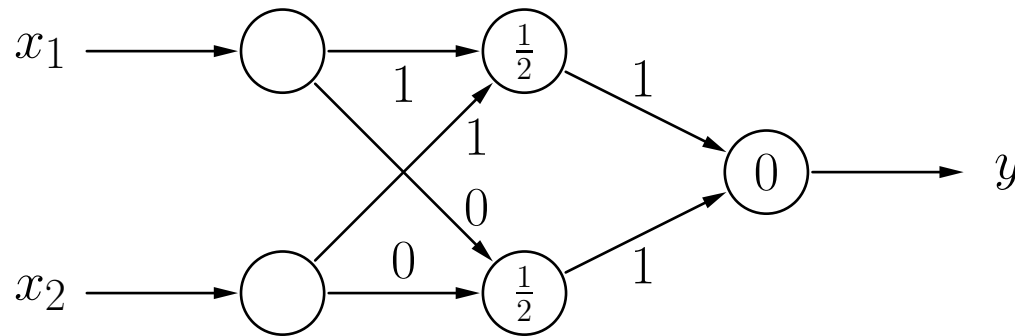
- $(0,0)$ ist Zentrum
- Referenzradius ist $\frac{6}{5}$
- Euklidischer Abstand
- Rechteckfunktion als Aktivierung
- Biaswert -1 im Ausgabeneuron

Radiale-Basisfunktionen-Netze: Beispiele

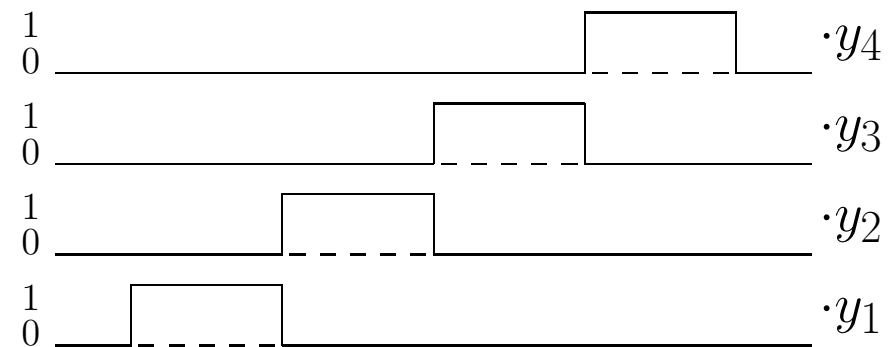
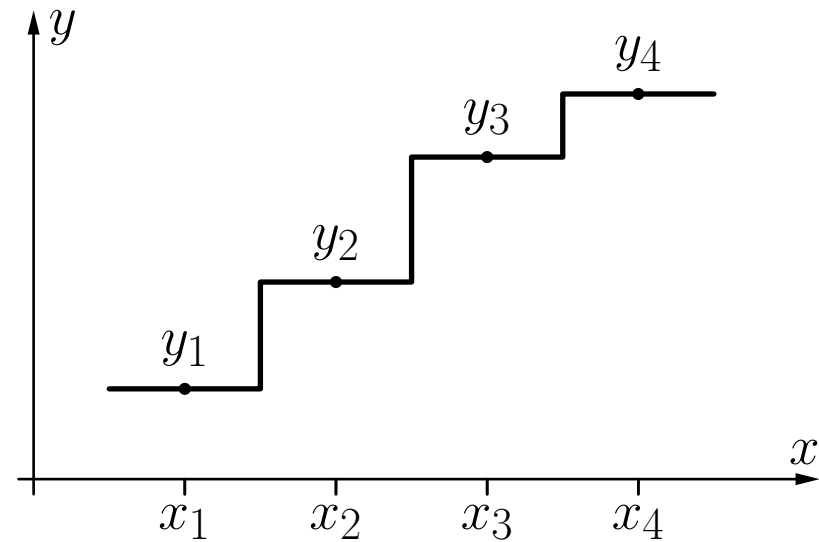
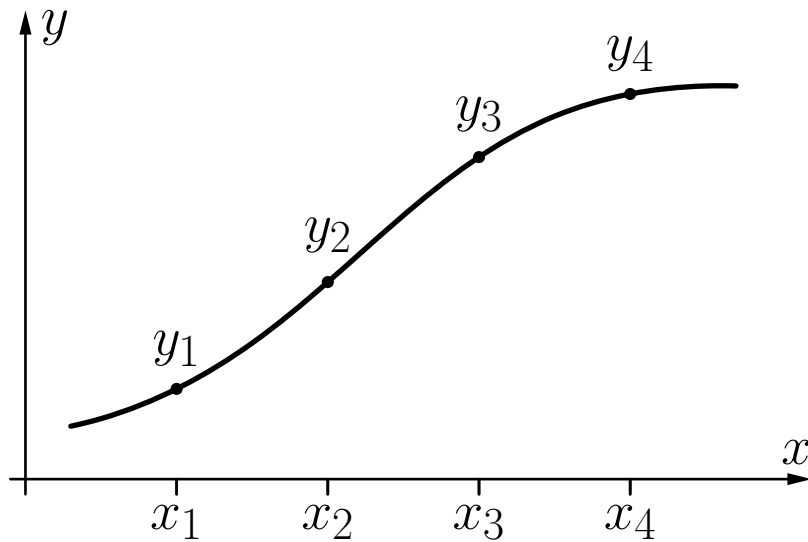
Radiale-Basisfunktionen-Netz für die Biimplikation $x_1 \leftrightarrow x_2$

Idee: logische Zerlegung

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

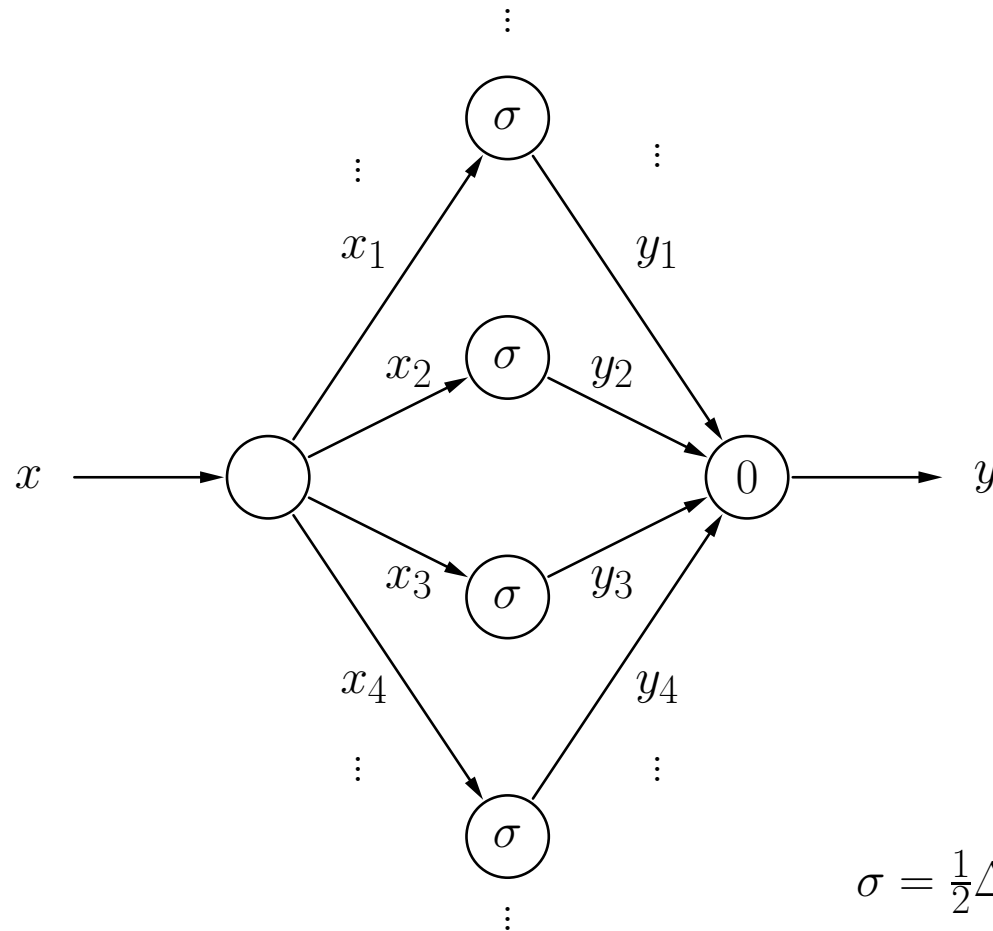


Radiale-Basisfunktionen-Netze: Funktionsapproximation



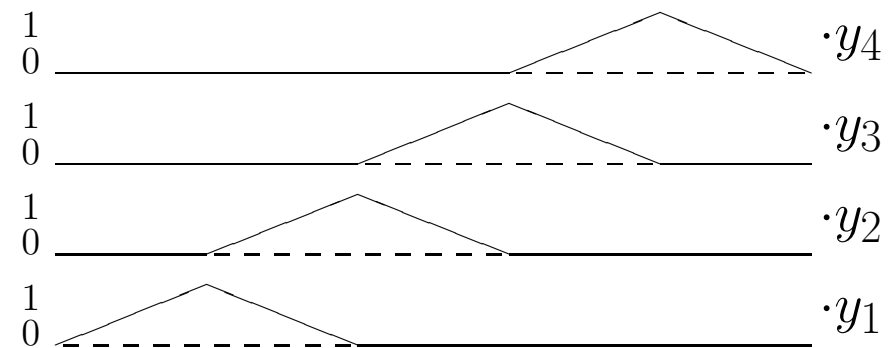
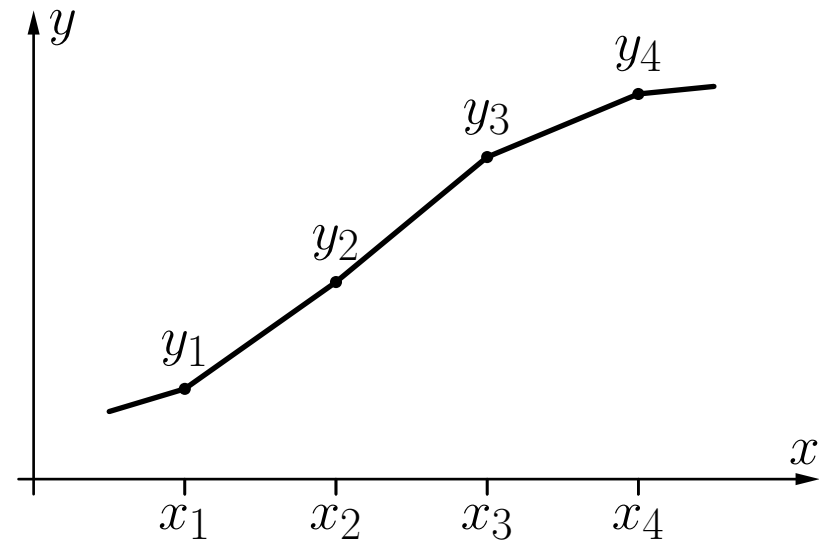
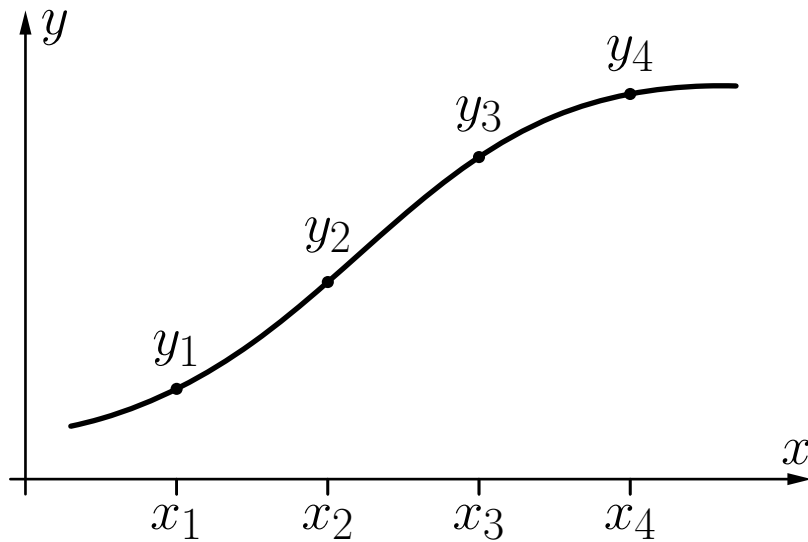
Annäherung der Originalfunktion durch Stufenfunktionen, deren Stufen durch einzelne Neuronen eines RBF-Netzes dargestellt werden können (vgl. MLPs).

Radiale-Basisfunktionen-Netze: Funktionsapproximation



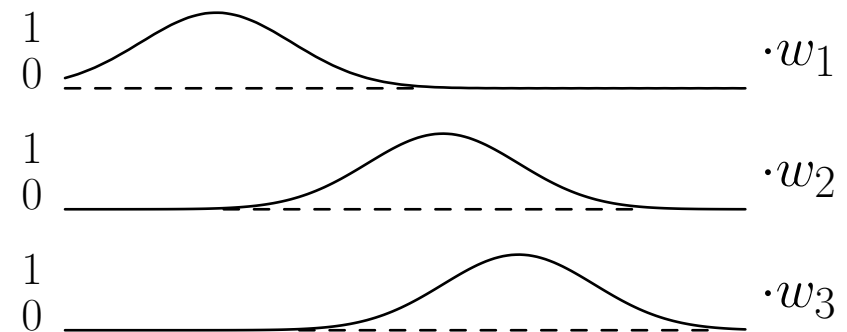
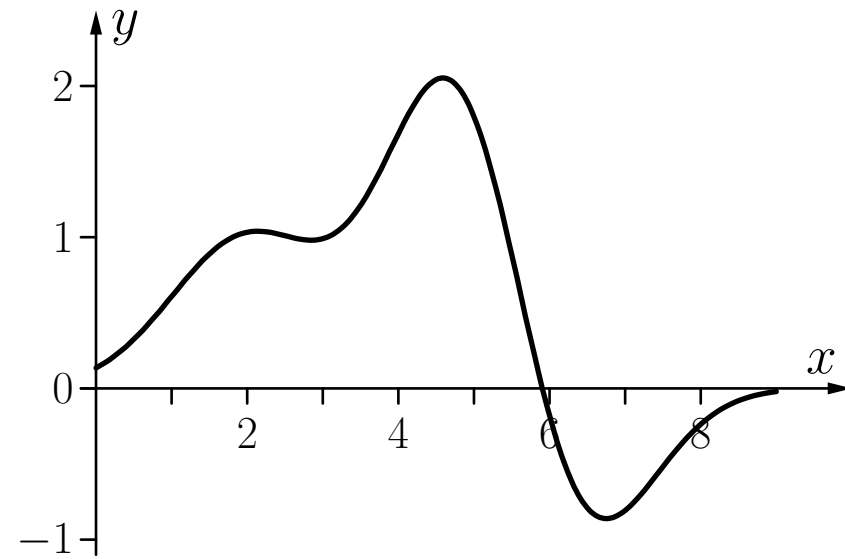
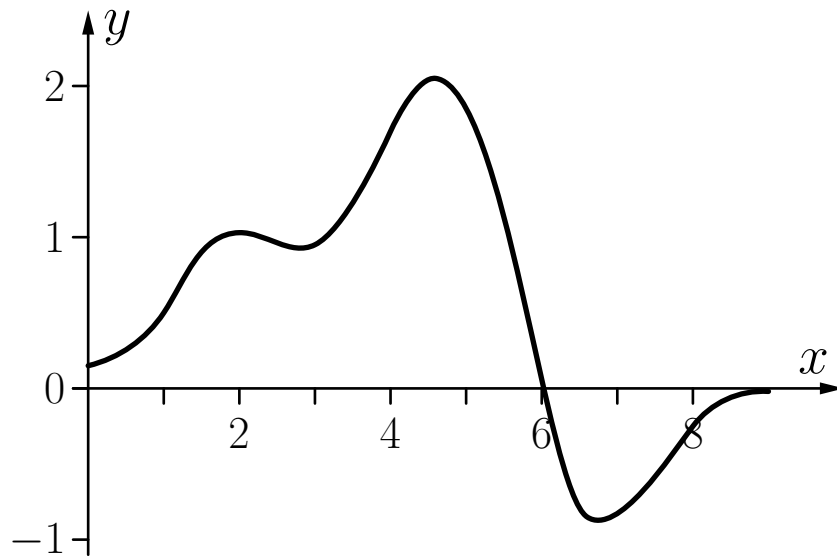
Ein RBF-Netz, das die Treppenfunktion von der vorherigen Folie bzw. die stückweise lineare Funktion der folgenden Folie berechnet (dazu muss nur die Aktivierungsfunktion der versteckten Neuronen geändert werden).

Radiale-Basisfunktionen-Netze: Funktionsapproximation



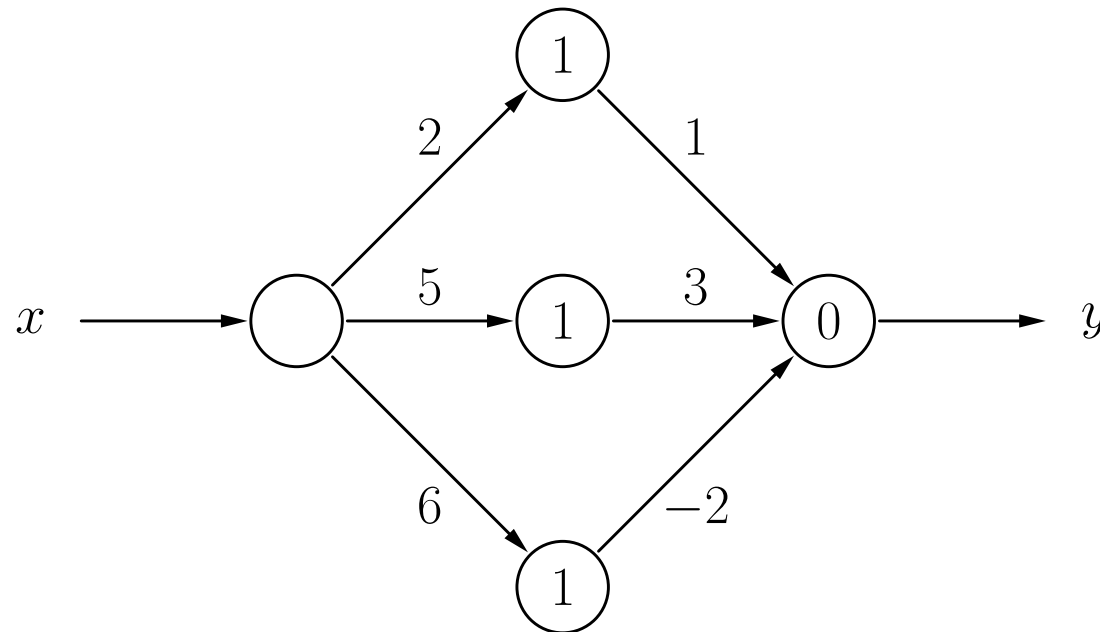
Darstellung einer stückweise linearen Funktion durch eine gewichtete Summe von Dreiecksfunktionen mit Zentren x_i .

Radiale-Basisfunktionen-Netze: Funktionsapproximation



Annäherung einer Funktion durch eine Summe von Gaußkurven mit Radius $\sigma = 1$.
Es ist $w_1 = 2$, $w_2 = 3$ und $w_3 = -2$.

RBF-Netz für eine Summe dreier Gaußfunktionen



Training von RBF-Netzen

Radiale-Basisfunktionen-Netze: Initialisierung

Sei $L_{\text{fixed}} = \{l_1, \dots, l_m\}$ eine feste Lernaufgabe, bestehend aus m Trainingsbeispielen $l = (\vec{i}^{(l)}, \vec{o}^{(l)})$.

Einfaches RBF-Netz:

Ein verstecktes Neuron v_k , $k = 1, \dots, m$, für jedes Trainingsbeispiel

$$\forall k \in \{1, \dots, m\} : \quad \vec{w}_{v_k} = \vec{i}^{(l_k)}.$$

Falls die Aktivierungsfunktion die Gaußfunktion ist, werden die Radien σ_k nach einer Heuristik gewählt

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

wobei

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

Radiale-Basisfunktionen-Netze: Initialisierung

Initialisieren der Verbindungen von den versteckten zu den Ausgabeneuronen

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

wobei $\vec{o}_u = (o_u^{(l_1)}, \dots, o_u^{(l_m)})^\top$ der Vektor der gewünschten Ausgaben ist, $\theta_u = 0$, und

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

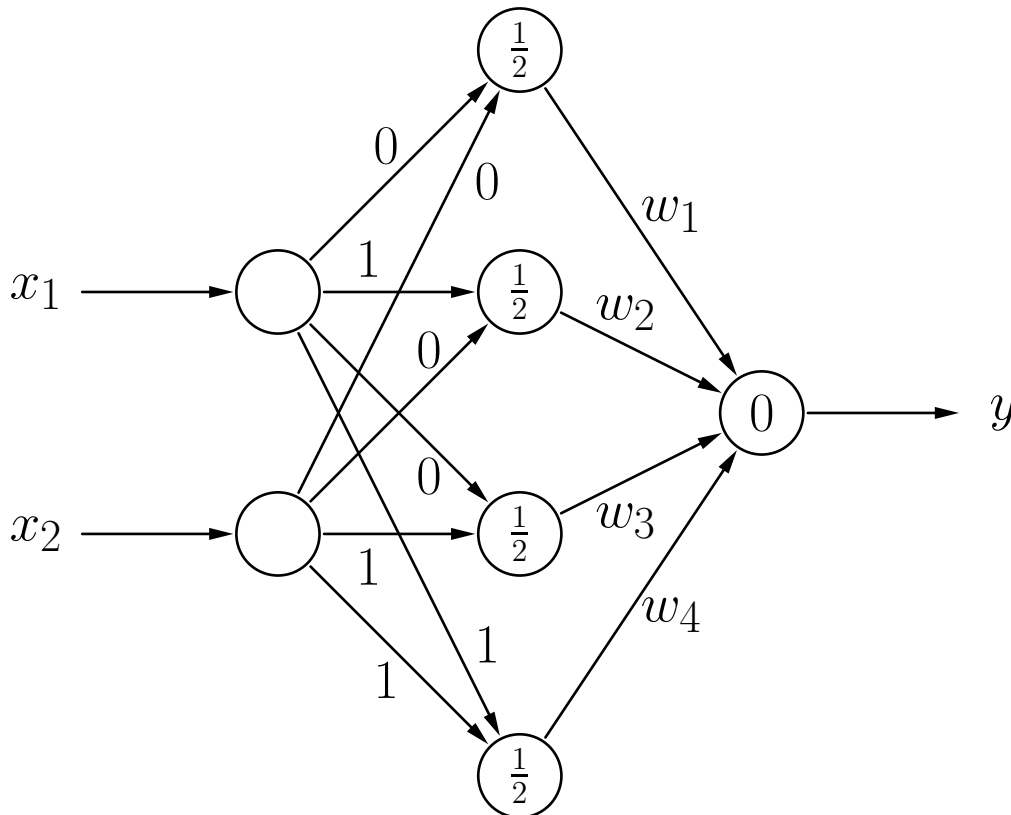
Ergebnis: Lineares Gleichungssystem, das durch Invertieren der Matrix \mathbf{A} gelöst werden kann:

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

RBF-Netz-Initialisierung: Beispiel

Einfaches RBF-Netz für die Biimplikation $x_1 \leftrightarrow x_2$

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



RBF-Netz-Initialisierung: Beispiel

Einfaches RBF-Netz für die Biimplikation $x_1 \leftrightarrow x_2$

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

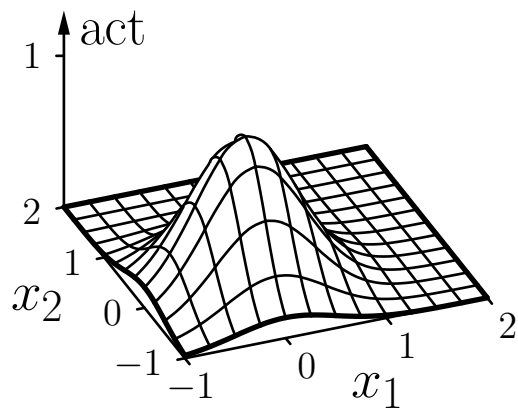
wobei

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

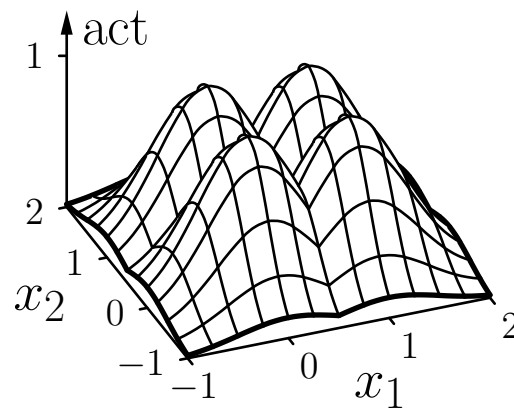
$$\vec{w}_u = \mathbf{A}^{-1} \cdot \vec{o}_u = \frac{1}{D} \begin{pmatrix} a + c \\ 2b \\ 2b \\ a + c \end{pmatrix} \approx \begin{pmatrix} 1.0567 \\ -0.2809 \\ -0.2809 \\ 1.0567 \end{pmatrix}$$

RBF-Netz-Initialisierung: Beispiel

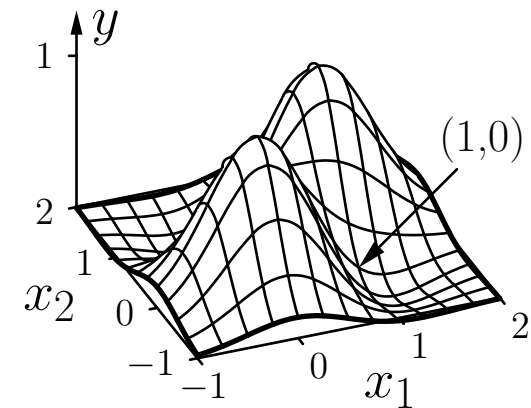
Einfaches RBF-Netz für die Biimplikation $x_1 \leftrightarrow x_2$



einzelne Basisfunktion



alle Basisfunktionen



Ausgabe

- Die Initialisierung führt bereits zu einer perfekten Lösung der Lernaufgabe.
- Weiteres Trainieren ist nicht notwendig.

Radiale-Basisfunktionen-Netze: Initialisierung

Normale Radiale-Basisfunktionen-Netze:

Wähle Teilmenge von k Trainingsbeispielen als Zentren aus.

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

Berechne (Moore–Penrose)-Pseudoinverse:

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

Die Gewichte können dann durch

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

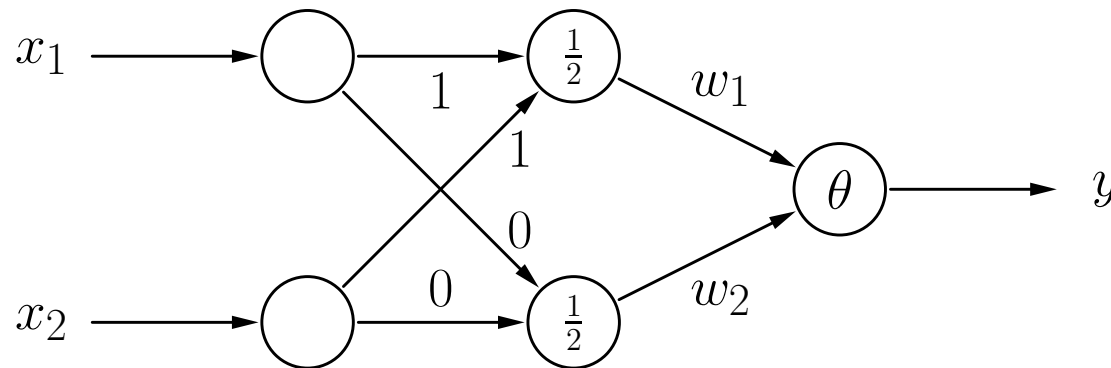
berechnet werden.

RBF-Netz-Initialisierung: Beispiel

Normales RBF-Netz für die Biimplikation $x_1 \leftrightarrow x_2$

Wähle zwei Trainingsbeispiele aus:

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



RBF-Netz-Initialisierung: Beispiel

Normales RBF-Netz für die Biimplikation $x_1 \leftrightarrow x_2$

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

wobei

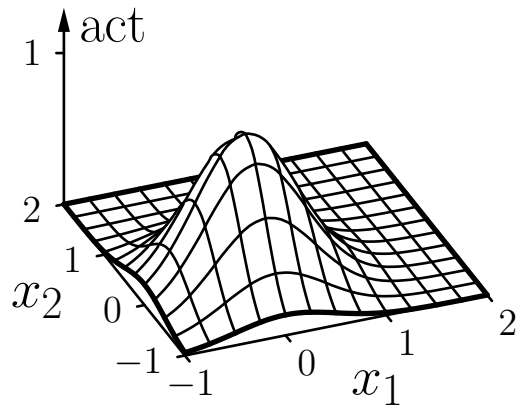
$$\begin{aligned} a &\approx -0.1810, & b &\approx 0.6810, \\ c &\approx 1.1781, & d &\approx -0.6688, & e &\approx 0.1594. \end{aligned}$$

Gewichte:

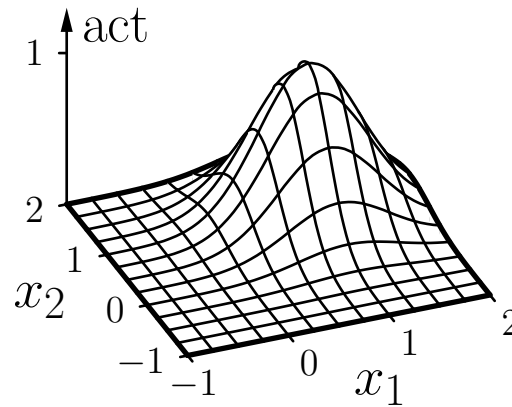
$$\vec{w}_u = \begin{pmatrix} -\theta \\ w_1 \\ w_2 \end{pmatrix} = \mathbf{A}^+ \cdot \vec{o}_u \approx \begin{pmatrix} -0.3620 \\ 1.3375 \\ 1.3375 \end{pmatrix}.$$

RBF-Netz-Initialisierung: Beispiel

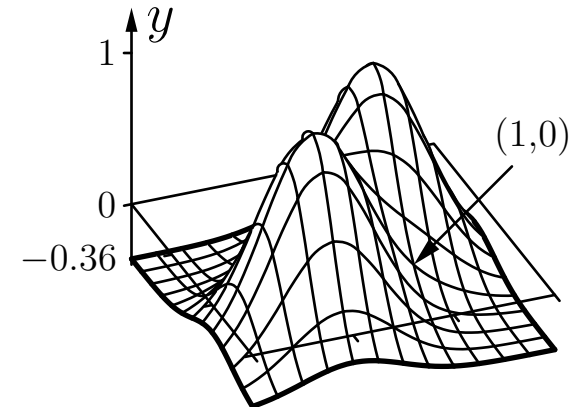
Normales RBF-Netz für die Biiimplikation $x_1 \leftrightarrow x_2$



Basisfunktion (0,0)



Basisfunktion (1,1)



Ausgabe

- Die Initialisierung führt bereits zu einer perfekten Lösung der Lernaufgabe.
- Dies ist Zufall, da das lineare Gleichungssystem wegen linear abhängiger Gleichungen nicht überbestimmt ist.

Bestimmung passender Zentren für die RBFs

Ein Ansatz: **k-means-Clustering**

- Wähle k zufällig ausgewählte Trainingsbeispiele als Zentren.
- Weise jedem Zentrum die am nächsten liegenden Trainingsbeispiele zu.
- Berechne neue Zentren als Schwerpunkt der dem Zentrum zugewiesenen Trainingsbeispiele.
- Wiederhole diese zwei Schritte bis zur Konvergenz, d.h. bis sich die Zentren nicht mehr ändern.
- Nutze die sich ergebenden Zentren für die Gewichtsvektoren der versteckten Neuronen.

Alternativer Ansatz: **Lernende Vektorquantisierung**

Radiale-Basisfunktionen-Netze: Training

Training von RBF-Netzen:

Herleitung der Update-Regeln ist analog zu der für MLPs.

Gewichte von den versteckten zu den Ausgabeneuronen.

Gradient:

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

Gewichtsänderungsregel:

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

(Zwei weitere Lernraten sind notwendig für die Positionen der Zentren und der Radien.)

Radiale-Basisfunktionen-Netze: Training

Training von RBF-Netzen:

Zentren: (Gewichte von Eingabe- zu versteckten Neuronen).

Gradient:

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

Gewichtsänderungsregel:

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

Radiale-Basisfunktionen-Netze: Training

Training von RBF-Netzen:

Zentren: (Gewichte von Eingabe- zu versteckten Neuronen).

Spezialfall: **Euklidischer Abstand**

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

Spezialfall: **Gaußsche Aktivierungsfunktion**

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

Radiale-Basisfunktionen-Netze: Training

Training von RBF-Netzen:

Radien der radialen Basisfunktionen.

Gradient:

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

Gewichtsänderungsregel:

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

Spezialfall: **Gaußsche Aktivierungsfunktion**

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

Radiale-Basisfunktionen-Netze: Verallgemeinerung

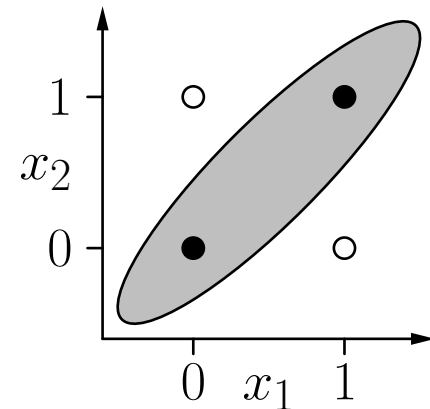
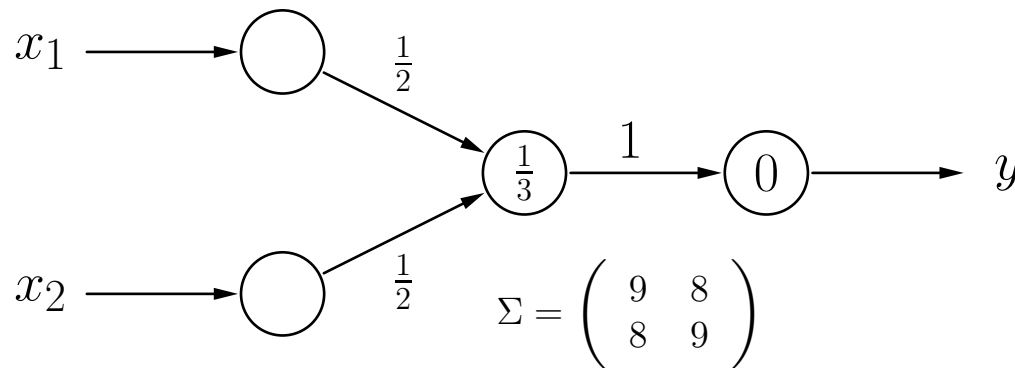
Verallgemeinerung der Abstandsfunktion

Idee: Benutze anisotrope (richtungsabhängige) Abstandsfunktion.

Beispiel: **Mahalanobis-Abstand**

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

Beispiel: **Biimplikation**



Radiale-Basisfunktionen-Netze: Anwendung

Vorteile

- einfache Feedforward-Architektur
- leichte Anpassbarkeit
- daher schnelle Optimierung und Berechnung

Anwendung

- kontinuierlich laufende Prozesse, die schnelle Anpassung erfordern
- Approximierung
- Mustererkennung
- Regelungstechnik