Training von RBF-Netzen

Radiale-Basisfunktionen-Netze: Initialisierung

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

Einfaches RBF-Netz:
Ein verstecktes Neuron v_k, $k = 1, \ldots, m$, für jedes Trainingsbeispiel

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

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

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

wobei

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

Initialisieren der Verbindungen von den versteckten zu den Ausgabeneuronen

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

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

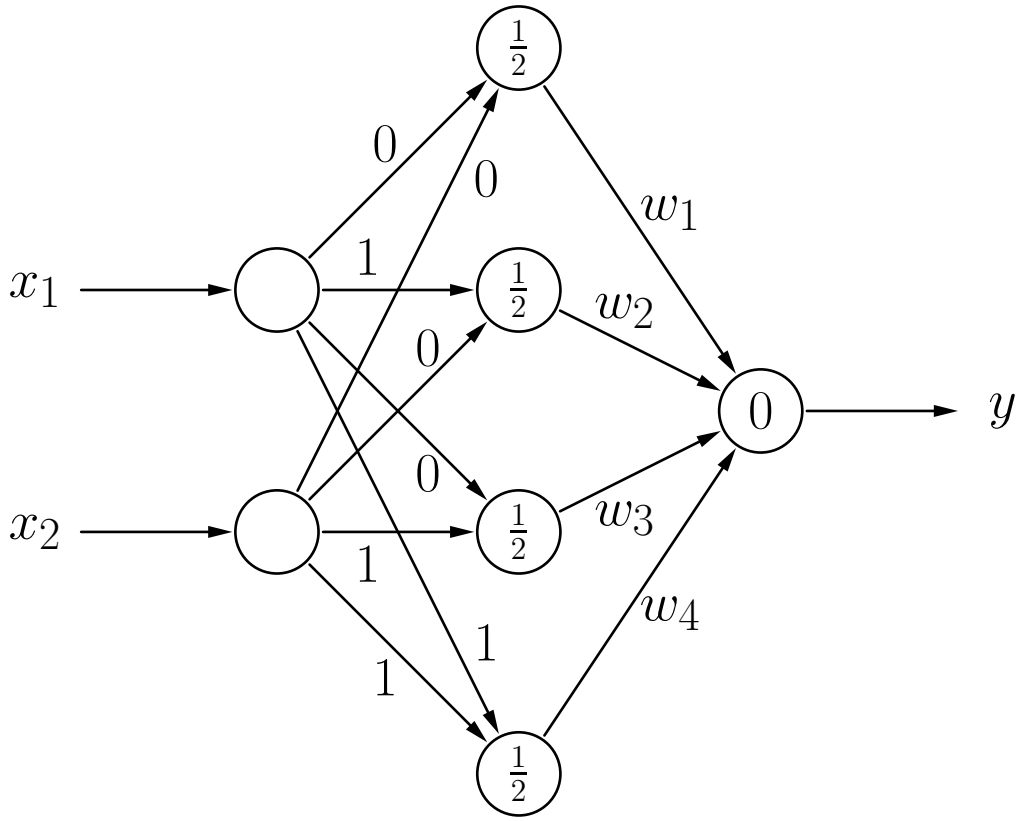
$$\mathbf{A} = \begin{pmatrix} \operatorname{out}_{v_1}^{(l_1)} & \operatorname{out}_{v_2}^{(l_1)} & \ldots & \operatorname{out}_{v_m}^{(l_1)} \\ \operatorname{out}_{v_1}^{(l_2)} & \operatorname{out}_{v_2}^{(l_2)} & \ldots & \operatorname{out}_{v_m}^{(l_2)} \\ \vdots & \vdots & & \vdots \\ \operatorname{out}_{v_1}^{(l_m)}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

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

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

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



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

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

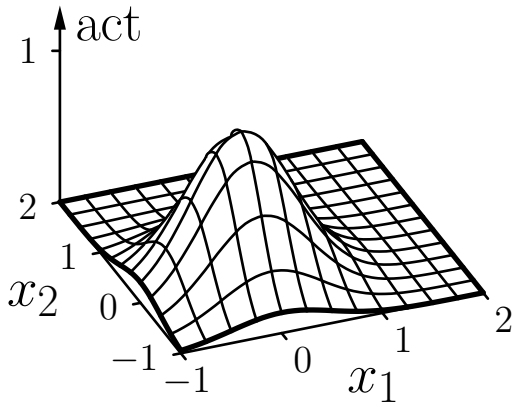
wobei

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

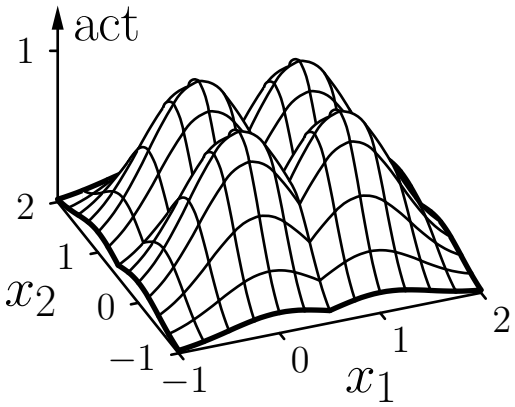
$$\boldsymbol{w}_u \;=\; \mathbf{A}^{-1} \cdot \boldsymbol{o}_u \;=\; \frac{1}{D} \begin{pmatrix} a+c \\ 2b \\ 2b \\ a+c \end{pmatrix} \approx \begin{pmatrix} 1.0567 \\ -0.2809 \\ -0.2809 \\ 1.0567 \end{pmatrix}$$

RBF-Netz-Initialisierung: Beispiel

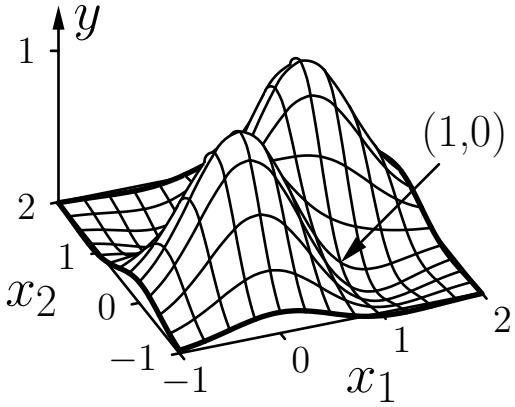
Einfaches RBF-Netz für die Biimplikation $x_1 \leftrightarrow x_2$



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.

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

Berechne (Moore–Penrose)-Pseudoinverse:

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

Die Gewichte können dann durch

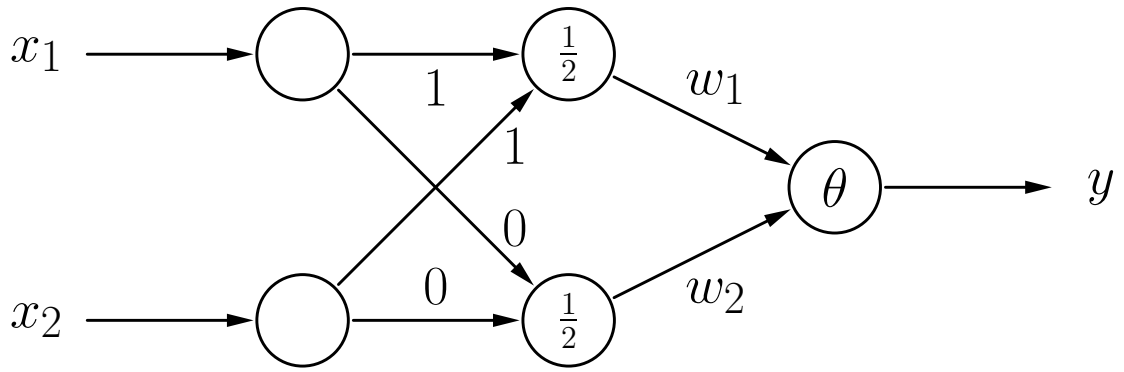
$$\boldsymbol{w}_u = \mathbf{A}^+ \cdot \boldsymbol{o}_u = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \cdot \boldsymbol{o}_u$$

berechnet werden.

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

Wähle zwei Trainingsbeispiele aus:

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



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

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

wobei

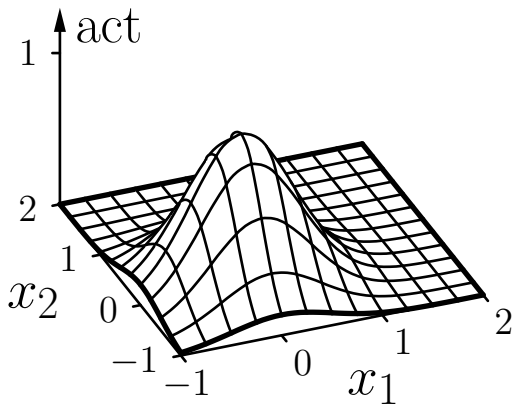
$$\begin{aligned} a &\approx -0.1810, & b &\approx 0.6810, \\ c &\approx 1.1781, & d &\approx -0.6688, & e &\approx 0.1594. \end{aligned}$$

Gewichte:

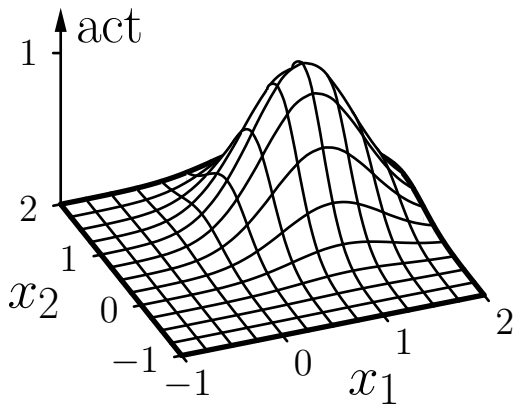
$$\boldsymbol{w}_u = \begin{pmatrix} -\theta \\ w_1 \\ w_2 \end{pmatrix} = \mathbf{A}^+ \cdot \boldsymbol{o}_u \approx \begin{pmatrix} -0.3620 \\ 1.3375 \\ 1.3375 \end{pmatrix}.$$

RBF-Netz-Initialisierung: Beispiel

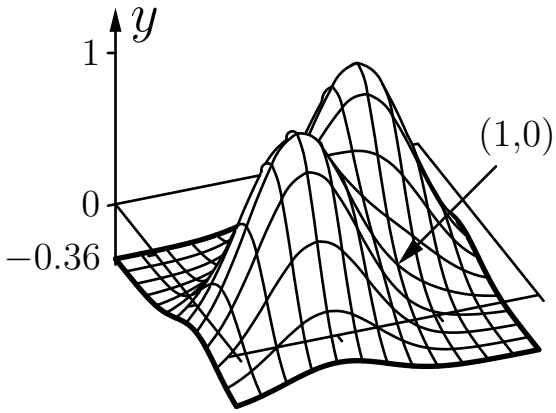
Normales RBF-Netz für die Biimplikation $x_1 \leftrightarrow x_2$



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**

Training von RBF-Netzen:
Herleitung der Update-Regeln ist analog zu der für MLPs.

Gewichte von den versteckten zu den Ausgabeneuronen.

Gradient:

$$\nabla_{\boldsymbol{w}_u} e_u^{(l)} = \frac{\partial e_u^{(l)}}{\partial \boldsymbol{w}_u} = -2\big(o_u^{(l)} - \mathrm{out}_u^{(l)}\big)\,\mathbf{in}_u^{(l)},$$

Gewichtsänderungsregel:

$$\Delta \boldsymbol{w}_u^{(l)} = -\frac{\eta_3}{2} \nabla_{\boldsymbol{w}_u} e_u^{(l)} = \eta_3\big(o_u^{(l)} - \mathrm{out}_u^{(l)}\big)\,\mathbf{in}_u^{(l)}$$

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

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

Gradient:

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

Gewichtsänderungsregel:

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

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

Spezialfall: **Euklidischer Abstand**

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

Spezialfall: **Gaußsche Aktivierungsfunktion**

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

Training von RBF-Netzen:
Radien der radialen Basisfunktionen.

Gradient:

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

Gewichtsänderungsregel:

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

Spezialfall: **Gaußsche Aktivierungsfunktion**

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

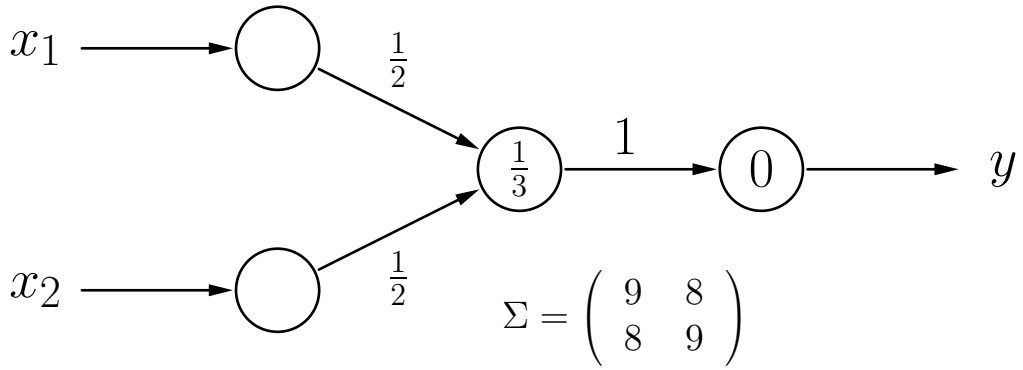
Verallgemeinerung der Abstandsfunktion

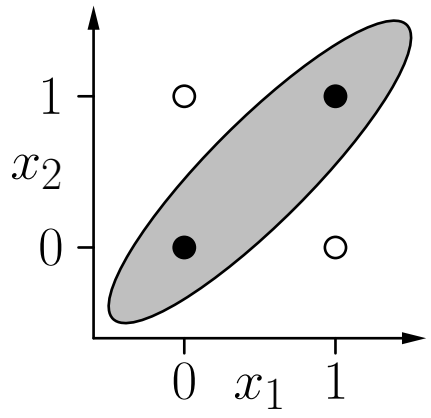
Idee: Benutze anisotrope (richtungsabhängige) Abstandsfunktion.

Beispiel: **Mahalanobis-Abstand**

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

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