

# Neuronale Netze

Prof. Dr. Rudolf Kruse

Computational Intelligence  
Institut für Wissens- und Sprachverarbeitung  
Fakultät für Informatik  
[kruse@iws.cs.uni-magdeburg.de](mailto:kruse@iws.cs.uni-magdeburg.de)



# Allgemeine (Künstliche) Neuronale Netze

## Graphentheoretische Grundlagen

Ein (gerichteter) **Graph** ist ein Tupel  $G = (V, E)$ , bestehend aus einer (endlichen) Menge  $V$  von **Knoten** oder **Ecken** und einer (endlichen) Menge  $E \subseteq V \times V$  von **Kanten**.

Wir nennen eine Kante  $e = (u, v) \in E$  **gerichtet** von Knoten  $u$  zu Knoten  $v$ .

Sei  $G = (V, E)$  ein (gerichteter) Graph und  $u \in V$  ein Knoten. Dann werden die Knoten der Menge

$$\text{pred}(u) = \{v \in V \mid (v, u) \in E\}$$

die **Vorgänger** des Knotens  $u$   
und die Knoten der Menge

$$\text{succ}(u) = \{v \in V \mid (u, v) \in E\}$$

die **Nachfolger** des Knotens  $u$  genannt.

## Allgemeine Definition eines neuronalen Netzes

Ein (künstliches) **neuronales Netz** ist ein (gerichteter) Graph  $G = (U, C)$ , dessen Knoten  $u \in U$  **Neuronen** oder **Einheiten** und dessen Kanten  $c \in C$  **Verbindungen** genannt werden.

Die Menge  $U$  der Knoten wird partitioniert in

- die Menge  $U_{\text{in}}$  der **Eingabeneuronen**,
- die Menge  $U_{\text{out}}$  der **Ausgabeneuronen**, und
- die Menge  $U_{\text{hidden}}$  der **versteckten Neuronen**.

It is

$$U = U_{\text{in}} \cup U_{\text{out}} \cup U_{\text{hidden}},$$

$$U_{\text{in}} \neq \emptyset, \quad U_{\text{out}} \neq \emptyset, \quad U_{\text{hidden}} \cap (U_{\text{in}} \cup U_{\text{out}}) = \emptyset.$$

# Allgemeine Neuronale Netze

Jede Verbindung  $(v, u) \in C$  besitzt ein **Gewicht**  $w_{uv}$  und jedes Neuron  $u \in U$  besitzt drei (reellwertige) Zustandsvariablen:

- die **Netzwerkeingabe**  $\text{net}_u$ ,
- die **Aktivierung**  $\text{act}_u$ , und
- die **Ausgabe**  $\text{out}_u$ .

Jedes Eingabeneuron  $u \in U_{\text{in}}$  besitzt weiterhin eine vierte reellwertige Zustandsvariable,

- die **externe Eingabe**  $\text{ex}_u$ .

Weiterhin besitzt jedes Neuron  $u \in U$  drei Funktionen:

- die **Netzwerkeingabefunktion**  $f_{\text{net}}^{(u)} : \mathbb{R}^{2|\text{pred}(u)|+\kappa_1(u)} \rightarrow \mathbb{R}$ ,
- die **Aktivierungsfunktion**  $f_{\text{act}}^{(u)} : \mathbb{R}^{\kappa_2(u)} \rightarrow \mathbb{R}$ , und
- die **Ausgabefunktion**  $f_{\text{out}}^{(u)} : \mathbb{R} \rightarrow \mathbb{R}$ ,

die benutzt werden, um die Werte der Zustandsvariablen zu berechnen.

## Typen (künstlicher) neuronaler Netze

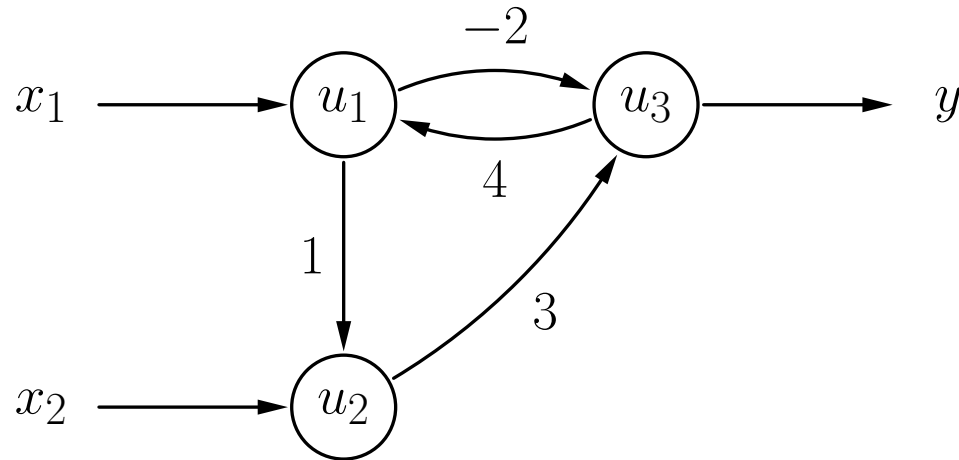
- Falls der Graph eines neuronalen Netzes **azyklisch** ist, wird das Netz **Feed-Forward-Netzwerk** genannt.
- Falls der Graph eines neuronalen Netzes **Zyklen** enthält, (rückwärtige Verbindungen), wird es **rekurrentes Netzwerk** genannt.

## Darstellung der Verbindungsgewichte als Matrix

$$\begin{array}{cccc} & u_1 & u_2 & \dots & u_r \\ \left( \begin{array}{cccc} w_{u_1 u_1} & w_{u_1 u_2} & \dots & w_{u_1 u_r} \\ w_{u_2 u_1} & w_{u_2 u_2} & & w_{u_2 u_r} \\ \vdots & & & \vdots \\ w_{u_r u_1} & w_{u_r u_2} & \dots & w_{u_r u_r} \end{array} \right) & \begin{array}{l} u_1 \\ u_2 \\ \vdots \\ u_r \end{array} \end{array}$$

# Allgemeine Neuronale Netze: Beispiel

Ein einfaches rekurrentes neuronales Netz:

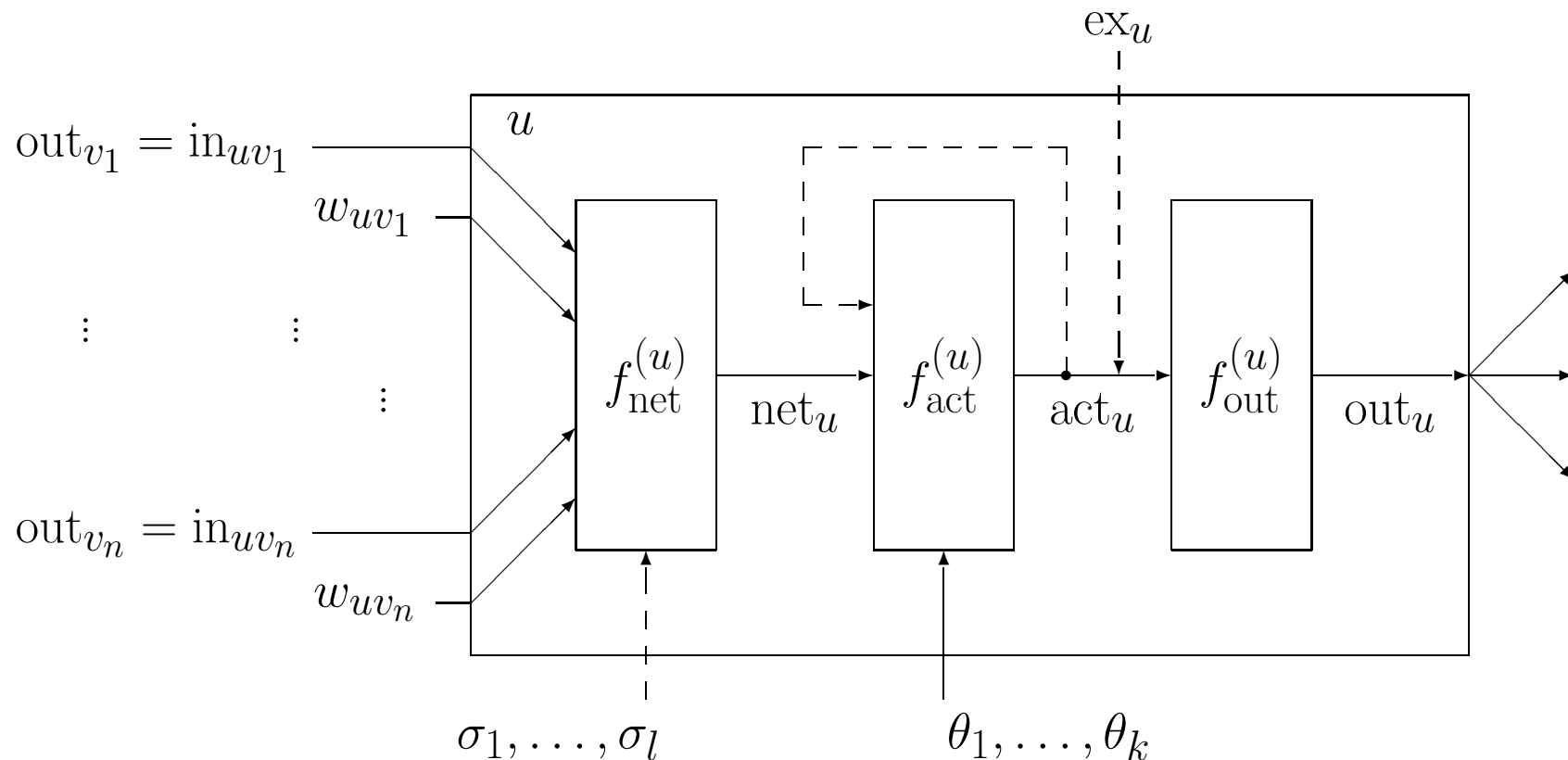


Gewichtsmatrix dieses Netzes:

$$\begin{array}{ccc} & u_1 & u_2 & u_3 \\ \left( \begin{array}{ccc} 0 & 1 & -2 \\ 0 & 0 & 3 \\ 4 & 0 & 0 \end{array} \right) & u_1 & u_2 & u_3 \end{array}$$

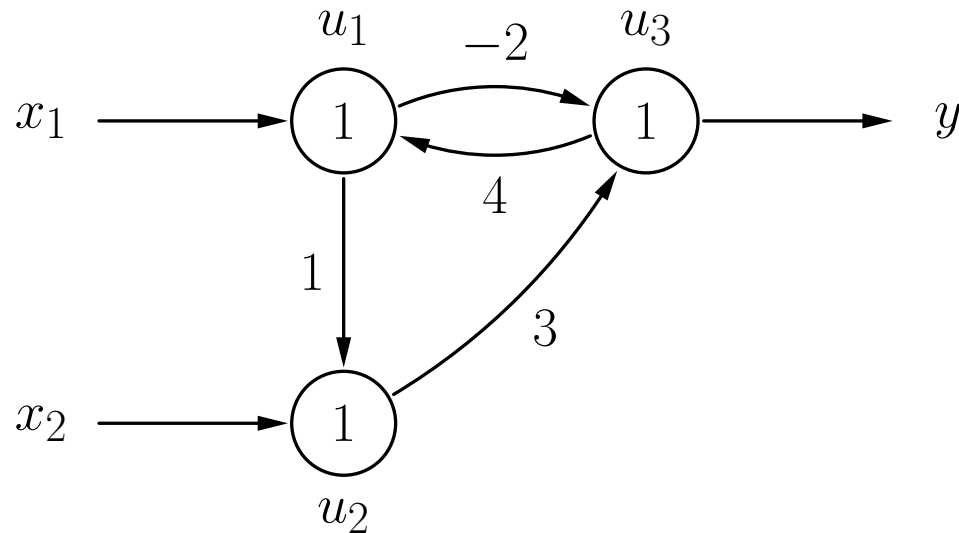
# Struktur eines verallgemeinerten Neurons

Ein verallgemeinertes Neuron verarbeitet numerische Werte





# Allgemeine Neuronale Netze: Beispiel



Merke: Die Gewichte der Knoten dieses Graphen sind die Schwellenwerte  $\theta_i$ .

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

$$f_{\text{act}}^{(u)}(\text{net}_u, \theta) = \begin{cases} 1, & \text{falls } \text{net}_u \geq \theta, \\ 0, & \text{sonst.} \end{cases}$$

$$f_{\text{out}}^{(u)}(\text{act}_u) = \text{act}_u$$

# Allgemeine Neuronale Netze: Beispiel

## Aktualisierung der Neuronenaktivierungen

	$u_1$	$u_2$	$u_3$	
Eingabephase	<b>1</b>	<b>0</b>	<b>0</b>	
Arbeitsphase	1	0	<b>0</b>	$\text{net}_{u_3} = -2$
	<b>0</b>	0	0	$\text{net}_{u_1} = 0$
	0	<b>0</b>	0	$\text{net}_{u_2} = 0$
	0	0	<b>0</b>	$\text{net}_{u_3} = 0$
	<b>0</b>	0	0	$\text{net}_{u_1} = 0$

- Aktualisierungsreihenfolge:  
 $u_3, u_1, u_2, u_3, u_1, u_2, u_3, \dots$
- Eingabephase: Aktivierungen/Ausgaben im Startzustand (erste Zeile)
- Die Aktivierung des gerade zu aktualisierenden Neurons (fettgedruckt) wird mit Hilfe der Ausgaben der anderen Neuronen und der Gewichte neu berechnet.
- Ein stabiler Zustand mit eindeutiger Ausgabe wird erreicht.

# Allgemeine Neuronale Netze: Beispiel

## Aktualisierung der Neuronenaktivierungen

	$u_1$	$u_2$	$u_3$	
Eingabephase	<b>1</b>	<b>0</b>	<b>0</b>	
Arbeitsphase	1	0	<b>0</b>	$\text{net}_{u_3} = -2$
	1	<b>1</b>	0	$\text{net}_{u_2} = 1$
	<b>0</b>	1	0	$\text{net}_{u_1} = 0$
	0	1	<b>1</b>	$\text{net}_{u_3} = 3$
	0	<b>0</b>	1	$\text{net}_{u_2} = 0$
	<b>1</b>	0	1	$\text{net}_{u_1} = 4$
	1	0	<b>0</b>	$\text{net}_{u_3} = -2$

- Aktualisierungsreihenfolge:  
 $u_3, u_2, u_1, u_3, u_2, u_1, u_3, \dots$
- Es wird kein stabiler Zustand erreicht (Oszillation der Ausgabe).

## Definition von Lernaufgaben für ein neuronales Netz

Eine  **feste Lernaufgabe**   $L_{\text{fixed}}$  für ein neuronales Netz mit

- $n$  Eingabeneuronen, d.h.  $U_{\text{in}} = \{u_1, \dots, u_n\}$ , and
- $m$  Ausgabeneuronen, d.h.  $U_{\text{out}} = \{v_1, \dots, v_m\}$ ,

ist eine Menge von  **Trainingsbeispielen**   $l = (\vec{i}^{(l)}, \vec{o}^{(l)})$ , bestehend aus

- einem  **Eingabevektor**   $\vec{i}^{(l)} = (\text{ex}_{u_1}^{(l)}, \dots, \text{ex}_{u_n}^{(l)})$  and
- einem  **Ausgabevektor**   $\vec{o}^{(l)} = (o_{v_1}^{(l)}, \dots, o_{v_m}^{(l)})$ .

Eine feste Lernaufgabe gilt als gelöst, wenn das NN für alle Trainingsbeispiele  $l \in L_{\text{fixed}}$  aus den externen Eingaben im Eingabevektor  $\vec{i}^{(l)}$  eines Trainingsmusters  $l$  die Ausgaben berechnet, die im entsprechenden Ausgabevektor  $\vec{o}^{(l)}$  enthalten sind.

## Lösen einer festen Lernaufgabe: Fehlerbestimmung

- Bestimme, wie gut ein neuronales Netz eine feste Lernaufgabe löst.
- Berechne Differenzen zwischen gewünschten und berechneten Ausgaben.
- Summiere Differenzen nicht einfach, da sich die Fehler gegenseitig aufheben könnten.
- Quadrieren liefert passende Eigenschaften, um die Anpassungsregeln abzuleiten.

$$e = \sum_{l \in L_{\text{fixed}}} e^{(l)} = \sum_{v \in U_{\text{out}}} e_v = \sum_{l \in L_{\text{fixed}}} \sum_{v \in U_{\text{out}}} e_v^{(l)},$$

$$\text{wobei } e_v^{(l)} = \left( o_v^{(l)} - \text{out}_v^{(l)} \right)^2$$

## Definition von Lernaufgaben für ein neuronales Netz

Eine **freie Lernaufgabe**  $L_{\text{free}}$  für ein neuronales Netz mit

- $n$  Eingabeneuronen, d.h.  $U_{\text{in}} = \{u_1, \dots, u_n\}$ ,

ist eine Menge von **Trainingsbeispielen**  $l = (\vec{v}^{(l)})$ , wobei jedes aus

- einem **Eingabevektor**  $\vec{v}^{(l)} = (\text{ex}_{u_1}^{(l)}, \dots, \text{ex}_{u_n}^{(l)})$  besteht.

Eigenschaften:

- Es gibt keine gewünschte Ausgabe für die Trainingsbeispiele.
- Ausgaben können von der Trainingsmethode frei gewählt werden.
- Lösungsidee: **Ähnliche Eingaben sollten zu ähnlichen Ausgaben führen.**  
(Clustering der Eingabevektoren)

## Normalisierung der Eingabevektoren

- Berechne Erwartungswert und Standardabweichung jeder Eingabe:

$$\mu_k = \frac{1}{|L|} \sum_{l \in L} \text{ex}_{u_k}^{(l)} \quad \text{and} \quad \sigma_k = \sqrt{\frac{1}{|L|} \sum_{l \in L} \left( \text{ex}_{u_k}^{(l)} - \mu_k \right)^2},$$

- Normalisiere die Eingabevektoren auf Erwartungswert 0 und Standardabweichung 1:

$$\text{ex}_{u_k}^{(l)(\text{neu})} = \frac{\text{ex}_{u_k}^{(l)(\text{alt})} - \mu_k}{\sigma_k}$$

- Vermeidet Skalierungsprobleme.