Neuronale Netze – Supervised Learning

Proseminar Kognitive Robotik (SS12)

Hannah Wester
Juan Jose Gonzalez

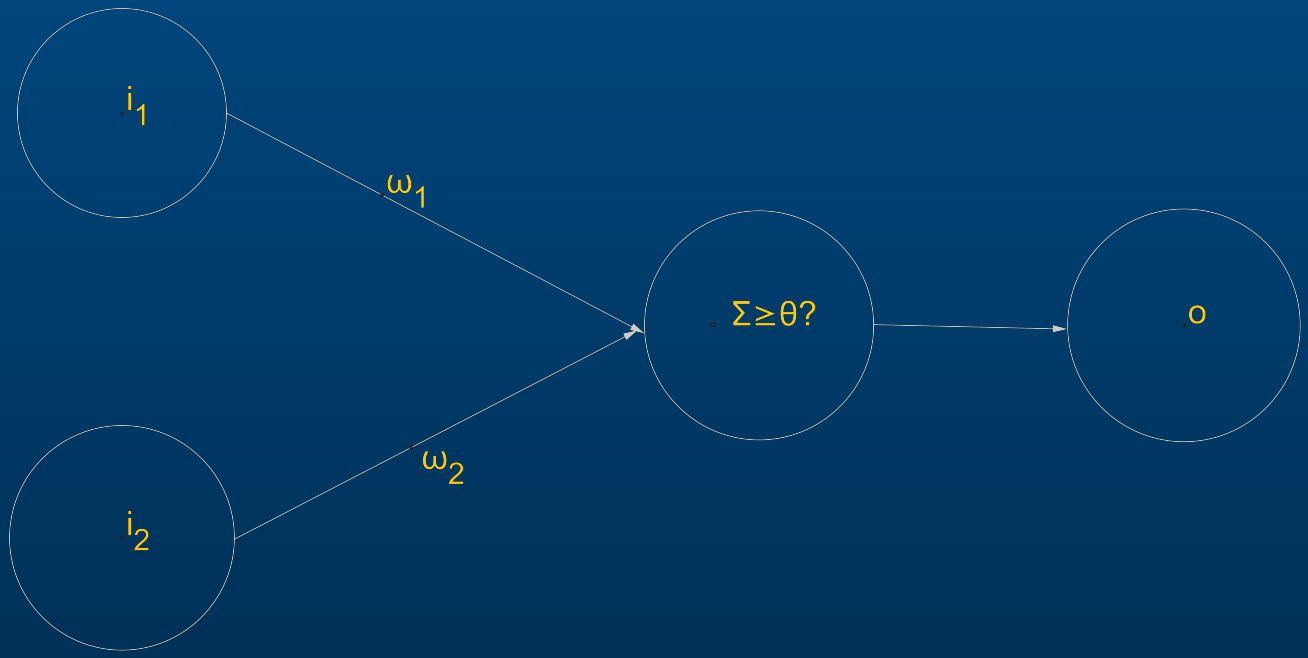
Kurze Einführung

Warum braucht man Neuronale Netze und insbesondere Supervised Learning?

Das Perzeptron

- Eingänge i_1 und i_2
- Gewichte ω_1 und ω_2
- Schwellwert θ (Bias)
- Binärer Ausgang $o = (\omega_1 i_1 + \omega_2 i_2 \geq \theta)$

Das Perzeptron



Das Perzeptron – Trainingsphase

- Training durch Änderung der Gewichte – θ wird oft als fester Eingang $i_0 = -1$ mit variablem Gewicht ω_0 definiert
- Überwachtes Lernen: Lernen durch Abarbeiten eines Trainingssets
- Aufbau eines Trainingssets: Festgelegte Eingänge mit gewünschten Ausgängen
- Ein Durchlauf durch ein Trainingsset nennt man Epoche

Das Perzeptron – Lernregeln

Hebb-Regel

- Verstärkendes Lernen
- Pfade zwischen aktiven Eingängen und aktiven Ausgängen werden verstärkt
- Lernrate ε: Grad der Verstärkung
- $\Delta\omega_i = \varepsilon\, i_i\, o$ bzw $\Delta\omega_{ij} = \varepsilon\, i_i\, o_j$

Das Perzeptron – Lernregeln

Delta-Regel

- Korrigierendes Lernen
- Die Gewichte werden proportional zur Differenz zwischen gewünschten und aktuellen Ausgängen verändert
- $\Delta\omega_i = \varepsilon\, i_i\, (\bar{o} - o)$ bzw $\Delta\omega_{ij} = \varepsilon\, i_i\, (\bar{o}_j - o_j)$

Das Perzeptron – Lernregeln

Delta-Regel – Beispiel

Eingang i1	Eingang i2	Ausgang o
0	0	0
0	1	1
1	0	0
1	1	1

- Lernrate ε: 0.2
- Gewichte zu Beginn: Alle 1.0
- Gewichte nach 2 Epochen: $\omega_0 = 1.0$, $\omega_1 = 0.8$, $\omega_2 = 1.2$

Das Perzeptron – Lernregeln

Epoche 1	$i_1=0/i_2=0$	$i_1=0/i_2=1$	$i_1=1/i_2=0$	$i_1=1/i_2=1$
ω_0	1.0	1.0	1.2	1.2
ω_1	1.0	1.0	0.8	0.8
ω_2	1.0	1.0	1.0	1.0
Epoche 2	$i_1=0/i_2=0$	$i_1=0/i_2=1$	$i_1=1/i_2=0$	$i_1=1/i_2=1$
ω_0	1.2	1.0	1.0	1.0
ω_1	0.8	0.8	0.8	0.8
ω_2	1.0	1.2	1.2	1.2

Das Perzeptron – Ausführungsphase

- Überprüfung: wurde das Trainingsmaterial erfasst?
- Überprüfung: kann das trainierte Netz auch weitere, ähnliche Problemstellungen lösen?
- Nicht gewünschte Ausgaben: neues Trainingsset, weiter trainieren

Das Perzeptron – XOR-Problem

Neues Beispiel für Delta-Regel:

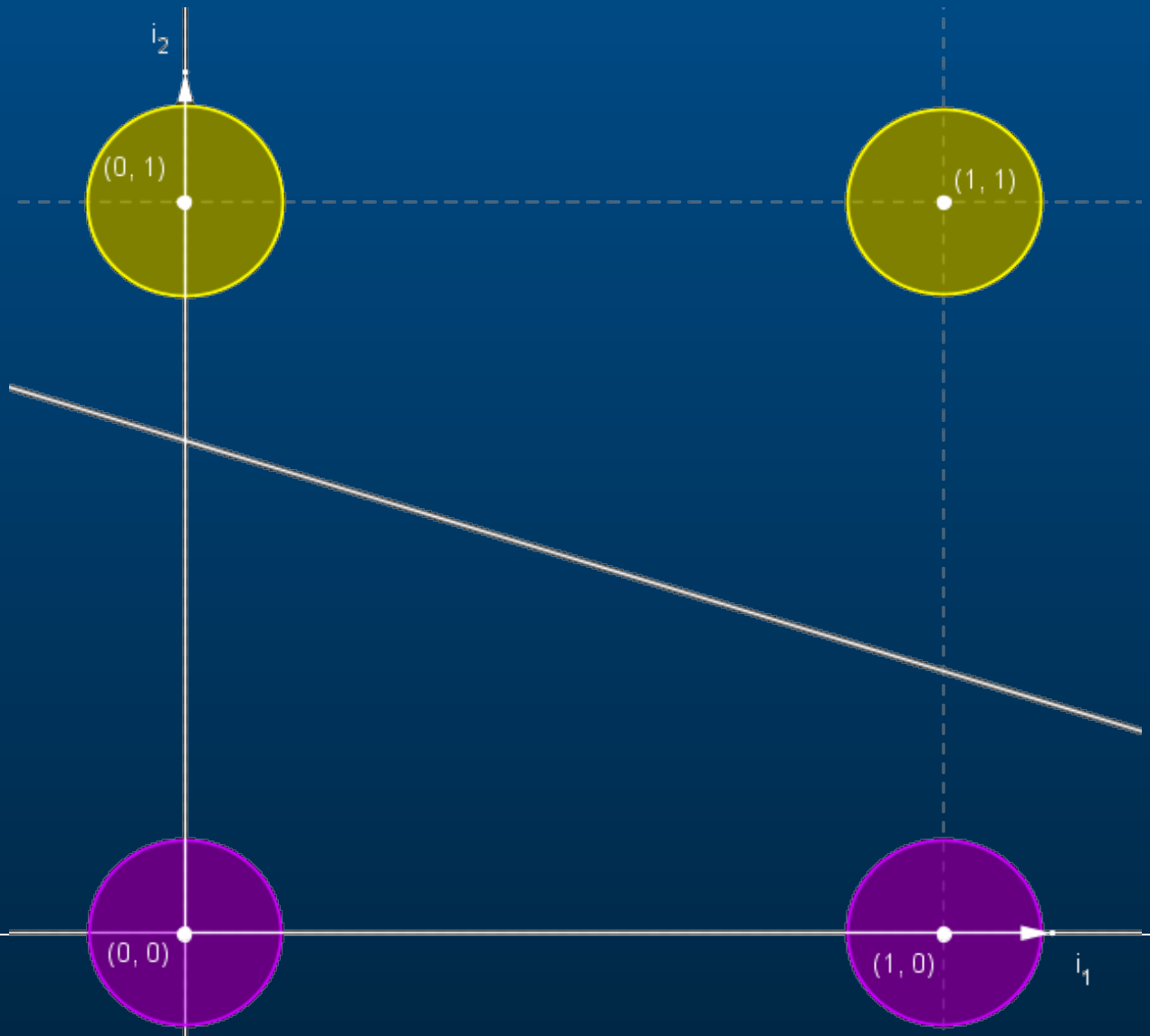
Eingang i1	Eingang i2	Ausgang o
0	0	0
0	1	1
1	0	1
1	1	0

Keine Lösungsfindung möglich!

Lineare Trennbarkeit

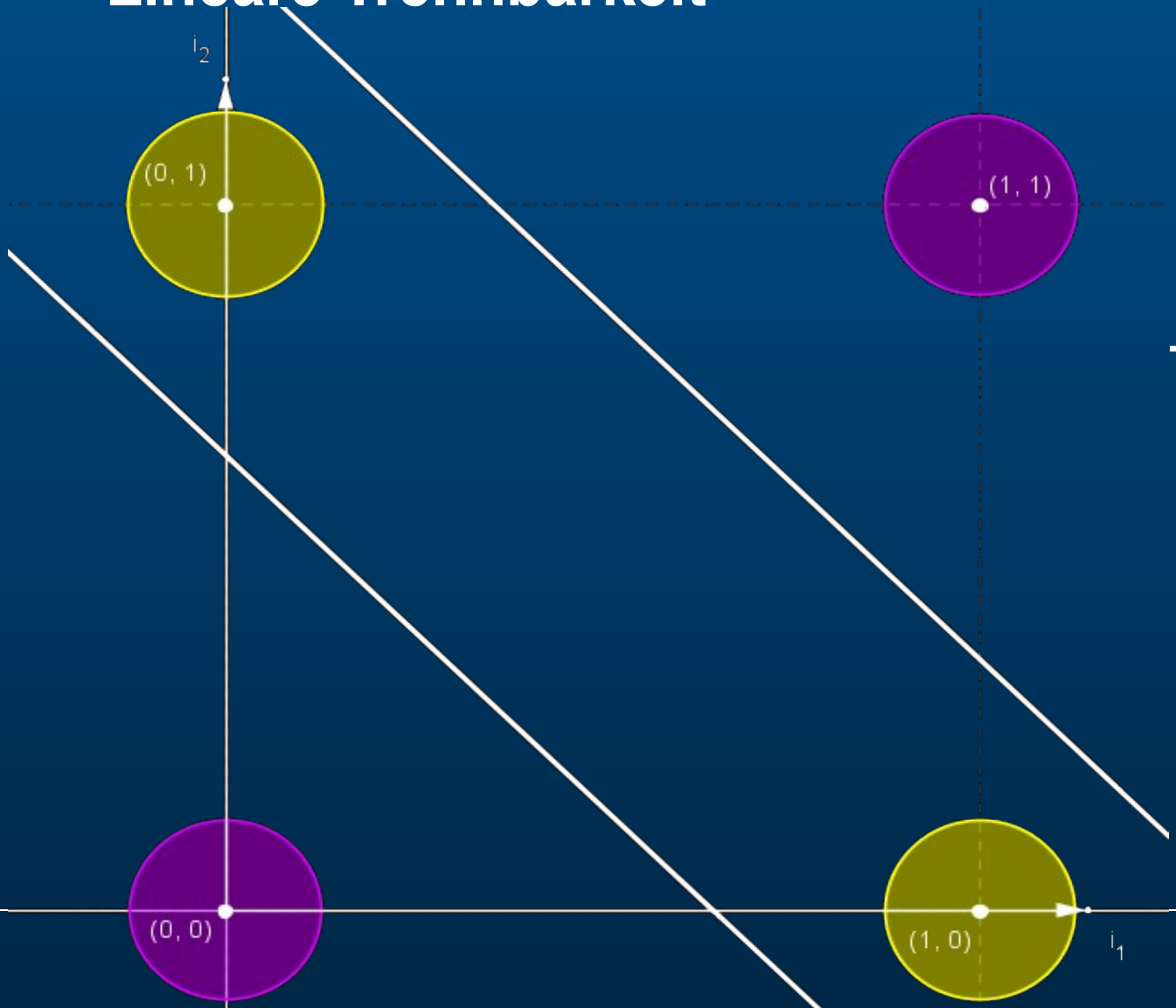
- sum = $\omega_1 i_1 + \omega_2 i_2 \geq \theta$
- $i_2 \geq \frac{\theta}{\omega_2} - \frac{\omega_1}{\omega_2} i_1$ → Geradengleichung
- Gerade trennt Eingaberaum in Eingaben die null liefern und Eingaben die eins liefern

Lineare Trennbarkeit



Trennbarkeit für Beispiel1 gezeigt

Lineare Trennbarkeit

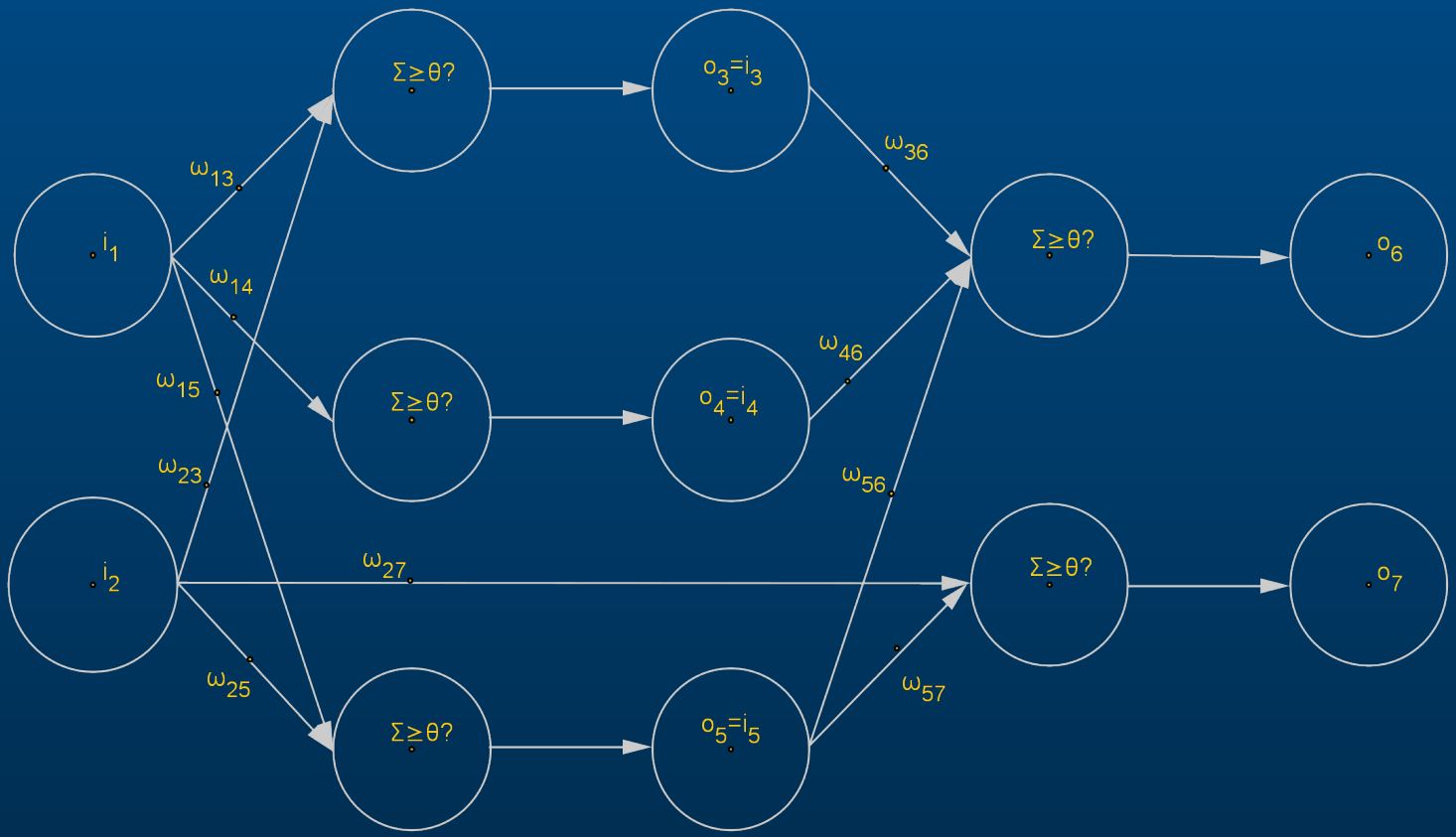


Trennbarkeit für Beispiel2 nicht möglich!

Mehrschichtige Netze

- Zusätzlich versteckte Schichten (Hidden Layers)
- Beliebig viele Ein- und Ausgänge
- Gerichtete Graphen nur von niedrigeren auf höhere Schichten: Feedforward-Netze
- Zyklische Feedback-Netze

Mehrschichtige Netze



Nichtlineare Netze

Inputfunktionen:

- Summe: $net_j = \sum_i \omega_{ij} i_i$
- Produkt: $net_j = \prod_i \omega_{ij} i_i$
- Maximum: $net_j = max(\omega_{ij} i_j)$
- Minimum: $net_j = min(\omega_{ij} i_j)$

Nichtlineare Netze

Aktivierungsfunktionen

$$f(net_i) = a_i$$

- Lineare Schwellwertfkt:

$$a_i = \begin{cases} 1 & \text{falls } net_i \leq \theta \\ 0 & \text{sonst} \end{cases}$$

- Fermi Funktion:

$$a_i = \frac{1}{1 + e^{-net_i}}$$

Nichtlineare Netze

Outputfunktion $f(a_i) = o_i$:

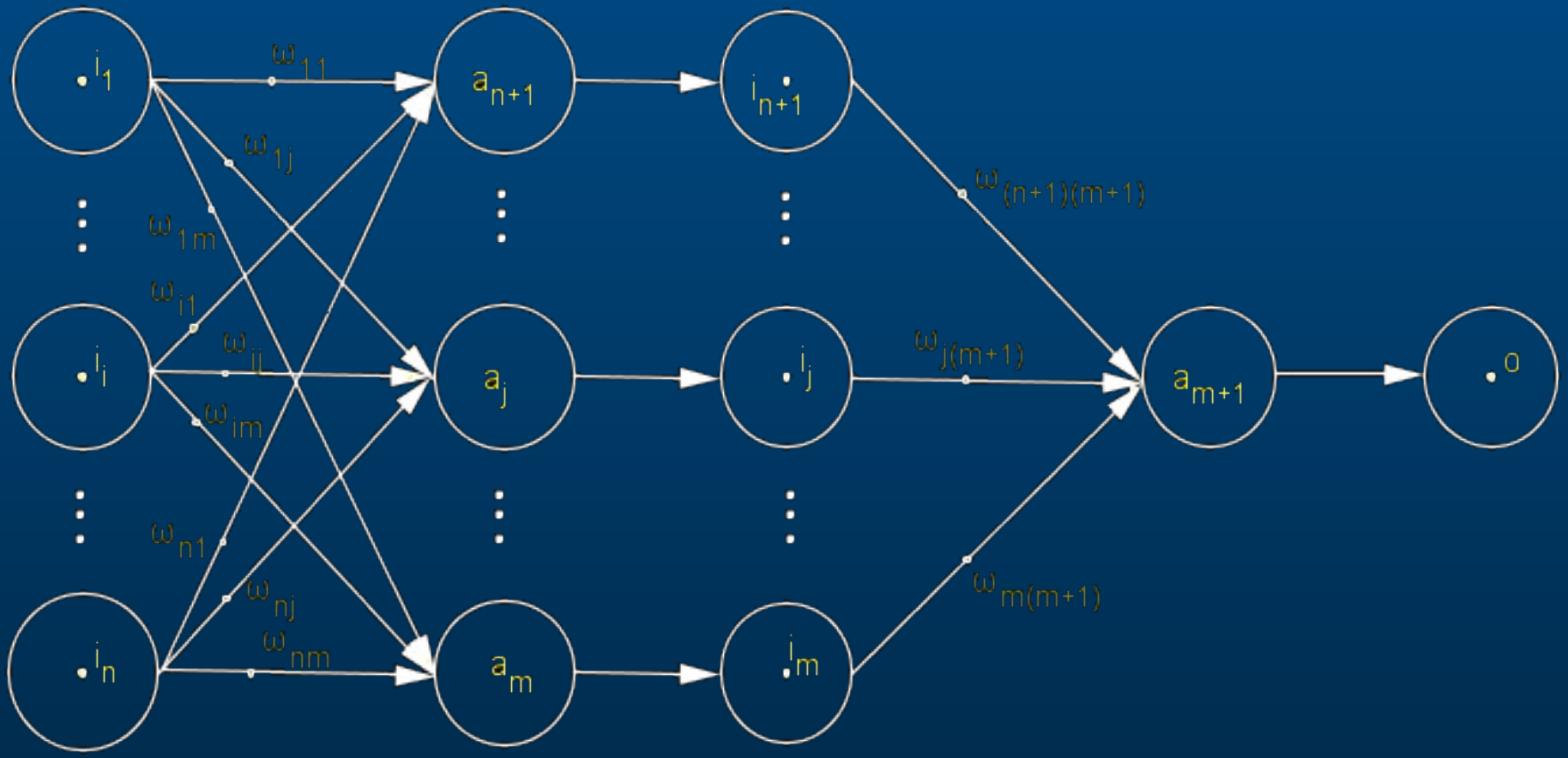
- Identität:

$$o_i = a_i$$

- Schwellwertfunktion:

$$o_i = \begin{cases} 1 & \text{falls } a_i \leq 0 \\ 0 & \text{sonst} \end{cases}$$

Nichtlineare Netze



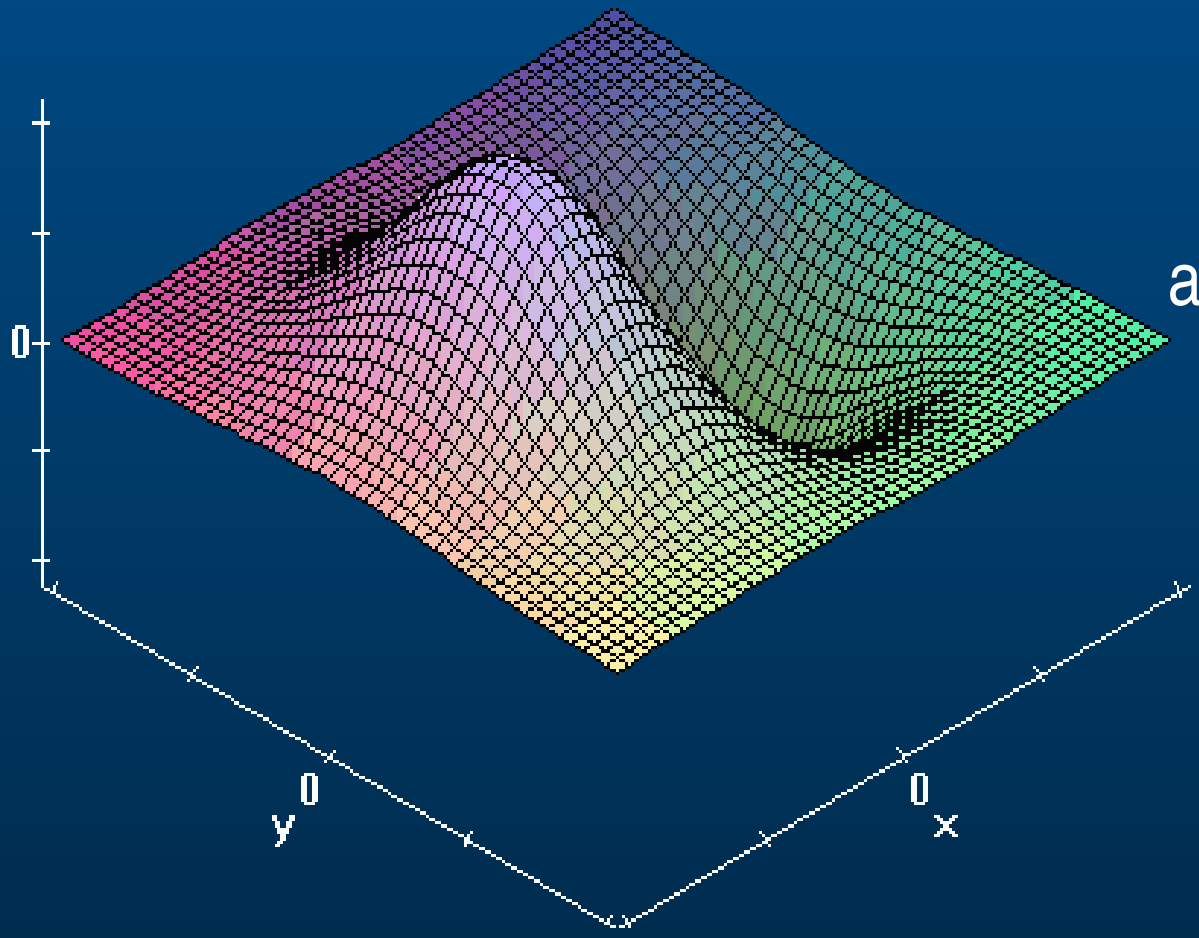
Backpropagation

- Algorithmus, um auch die Gewichte zwischen versteckten Schichten trainieren zu können
- Forward-Pass: Berechnen des Ausgangs
- Fehlerbestimmung: Ausgang = gewünschter Ausgang?
- Backward-Pass: Modifizieren der Gewichte

Backpropagation

- Fehlerfunktion $E = \frac{1}{2} \sum_j (\bar{o}_j - o_j)^2$
- Ableiten der Fehlerfunktion, um das Minimum zu finden
- Gradientenabstiegsverfahren

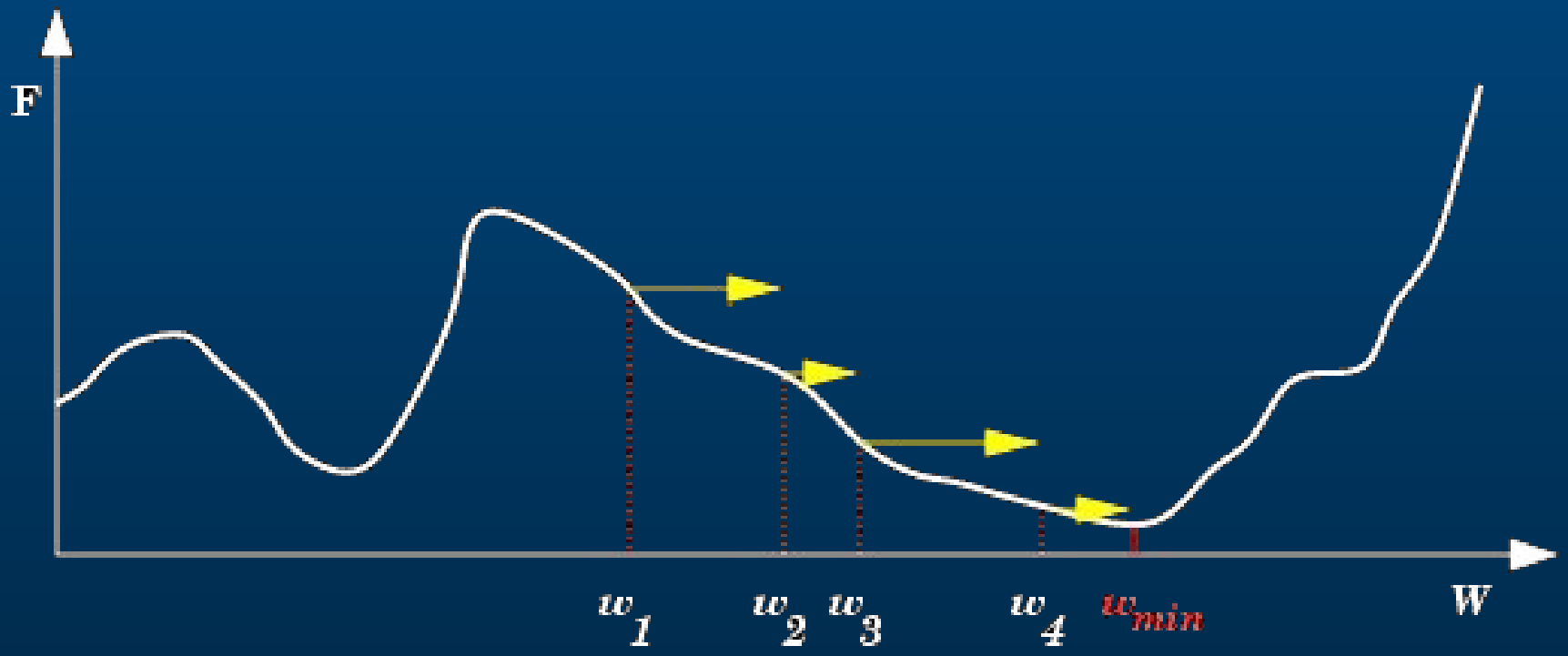
Backpropagation



Gradienten-abstiegsverfahren im dreidimensionalen Raum

Backpropagation

- Gradientenabstieg im zweidimensionalen Raum
- Vergrößern des Gewichtes, bis lokales Min. gefunden



Backpropagation

- Verallgemeinerung der Delta-Regel:

$$\Delta \omega_{ij} = -\varepsilon \frac{\partial E}{\partial \omega_{ij}}$$

- Umformen der Partiellen Ableitung:

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial \omega_{ij}}$$

Backpropagation

- Verallgemeinerung der Delta-Regel:

$$\Delta \omega_{ij} = -\varepsilon \frac{\partial E}{\partial \omega_{ij}}$$

- Umformen der Partiellen Ableitung:

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial \omega_{ij}}$$

Backpropagation

- Wegen $net_j = \sum_i \omega_{ij} i_i$ und $o_i = f(net_i)$ ergibt sich:

$$\frac{\partial net_j}{\partial \omega_{ij}} = i_i \text{ und } \frac{\partial o_i}{\partial net_i} = f'(net_i)$$

- o_j in Ausgabeschicht:

$$\frac{-\partial E}{\partial o_j} = \frac{-\partial}{\partial o_j}\left(\frac{1}{2}\sum_j (\bar{o}_j - o_j)^2\right) = -2\frac{1}{2}(\bar{o}_j - o_j)(-1) = (\bar{o}_j - o_j)$$

Backpropagation

- o_j in versteckter Schicht:

$$\frac{-\partial E}{\partial o_j} = -\sum_k \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial o_j}$$

$$= -\sum_k \frac{\partial E}{\partial net_k} \frac{\partial}{\partial o_j} \sum_i \omega_{ik} i_i = -\sum_k \frac{\partial E}{\partial net_k} \omega_{jk}$$

- $$-\frac{\partial E}{\partial net_j} = \delta_j \;\rightarrow\; \Delta \omega_{ij} = \varepsilon \delta_j i_i$$

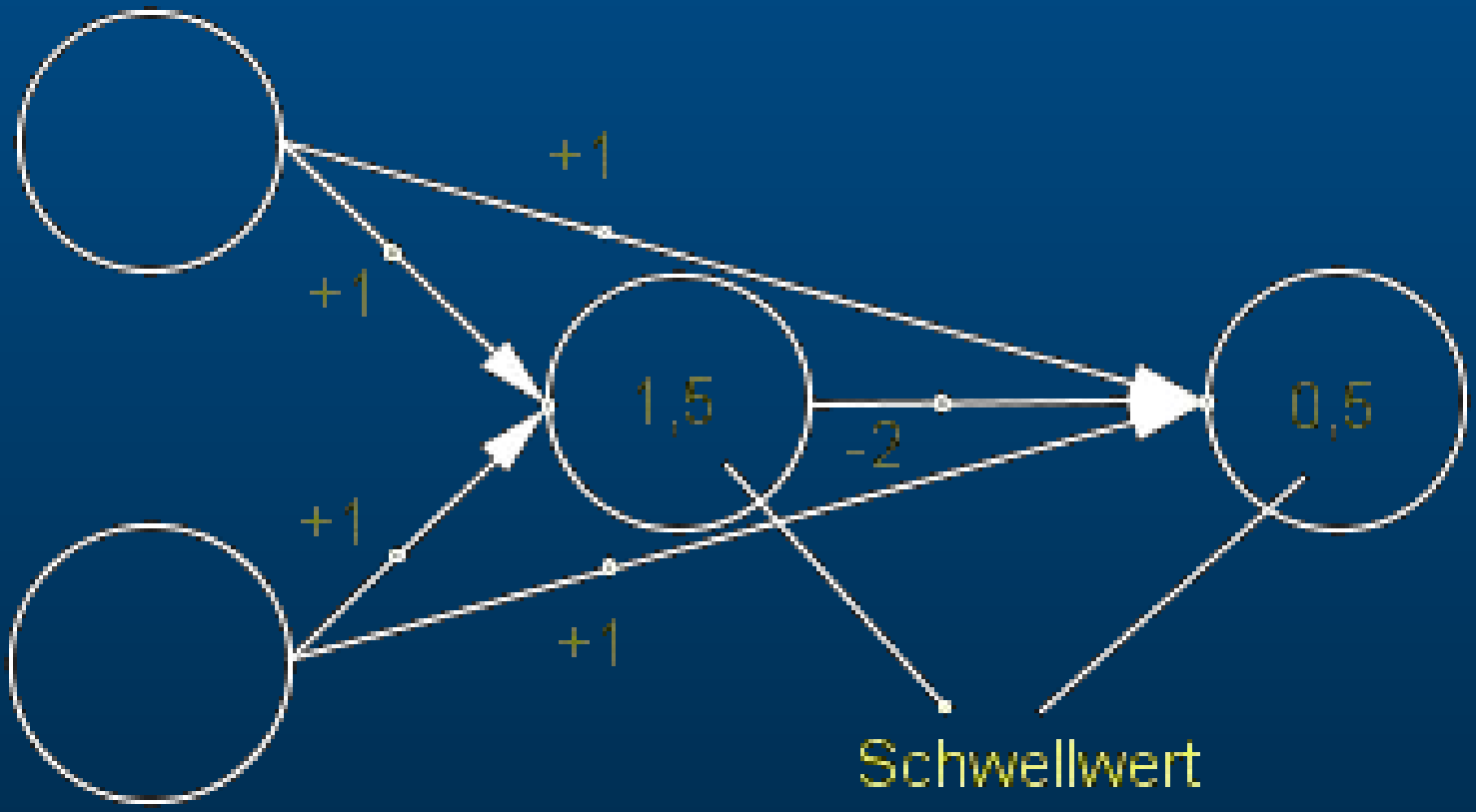
Backpropagation

- Verallgemeinerte Delta-Regel:

$$\Delta \omega_{ij} = \varepsilon \delta_j i_i \text{ mit}$$

$$\delta_j = \begin{cases} f'(net_j)(\bar{o}_j - o_j) & \text{falls } o_j \text{ in Ausgabeschicht} \\ f'(net_j) \sum_k \delta_k \omega_{jk} & \text{falls } o_j \text{ in versteckter Schicht} \end{cases}$$

Lösung XOR-Problem



Probleme von Backpropagation

- Die Struktur muss vorher definiert werden
- Sehr langsame Lernphase – Mehrere tausend Epochen sind üblicherweise nötig:
  - Gradientenabstiegsverfahren über $\partial E/\partial \omega$ → langsame Konvergenz beim Minimum
  - Alle Neuronen werden gleichzeitig trainiert → möglicherweise erkennen mehrere die gleiche Eigenschaft (keine homogene Verteilung)

Beschleunigte Backpropagation

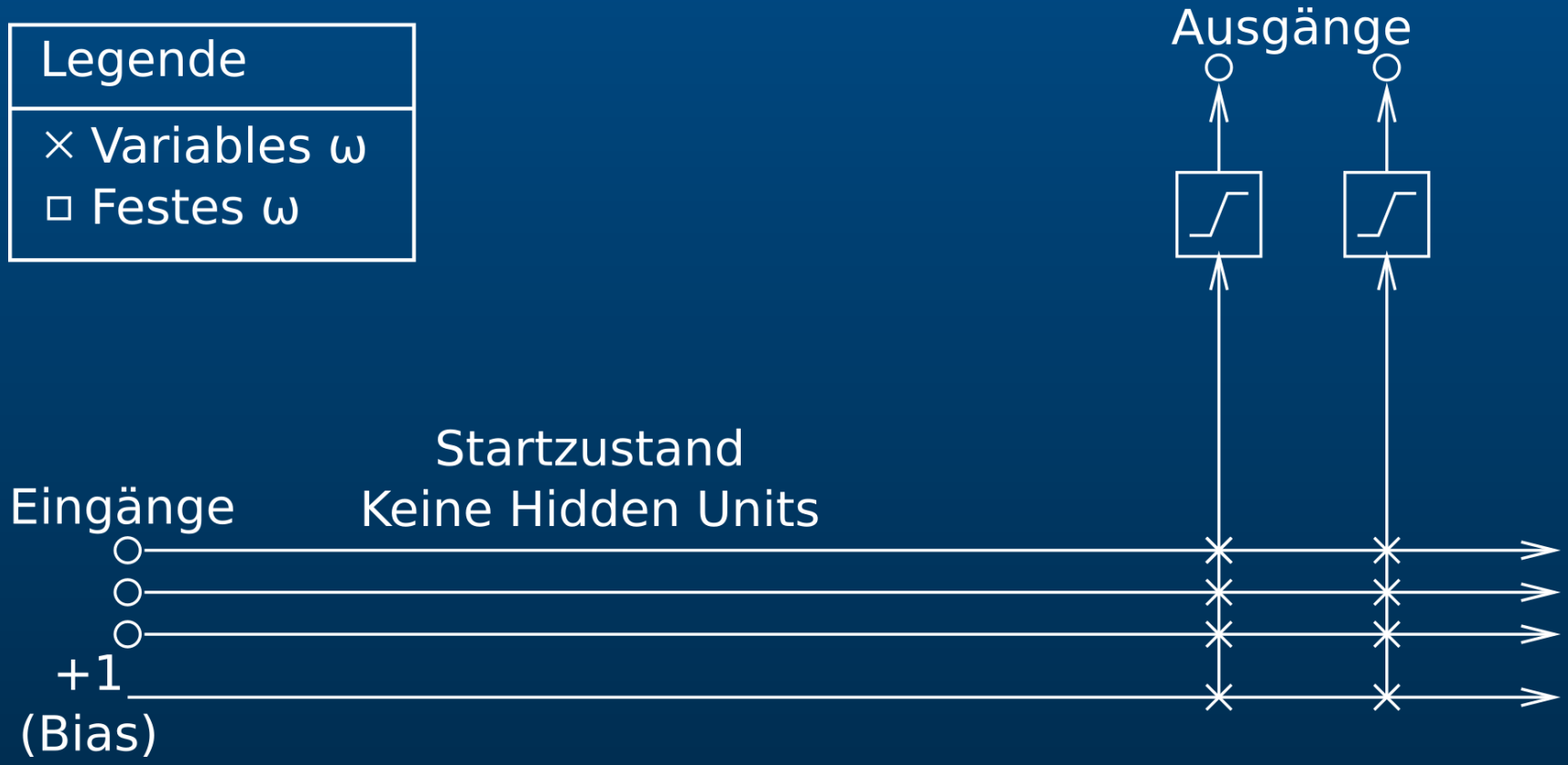
- Verringert die nötigen Schritte zur Konvergenz zum Minimum der Fehlerfunktion
- Beispiel: “Backpropagation with Momentum”
  - $\Delta\omega$ basiert nicht nur auf den aktuellen Werten, sondern auch auf vorherigen Änderungen
  - Neue Formel:

$$\Delta\omega_{ij}(t) = -\varepsilon(\delta_j o_i) + \alpha\Delta\omega_{ij}(t-1)$$

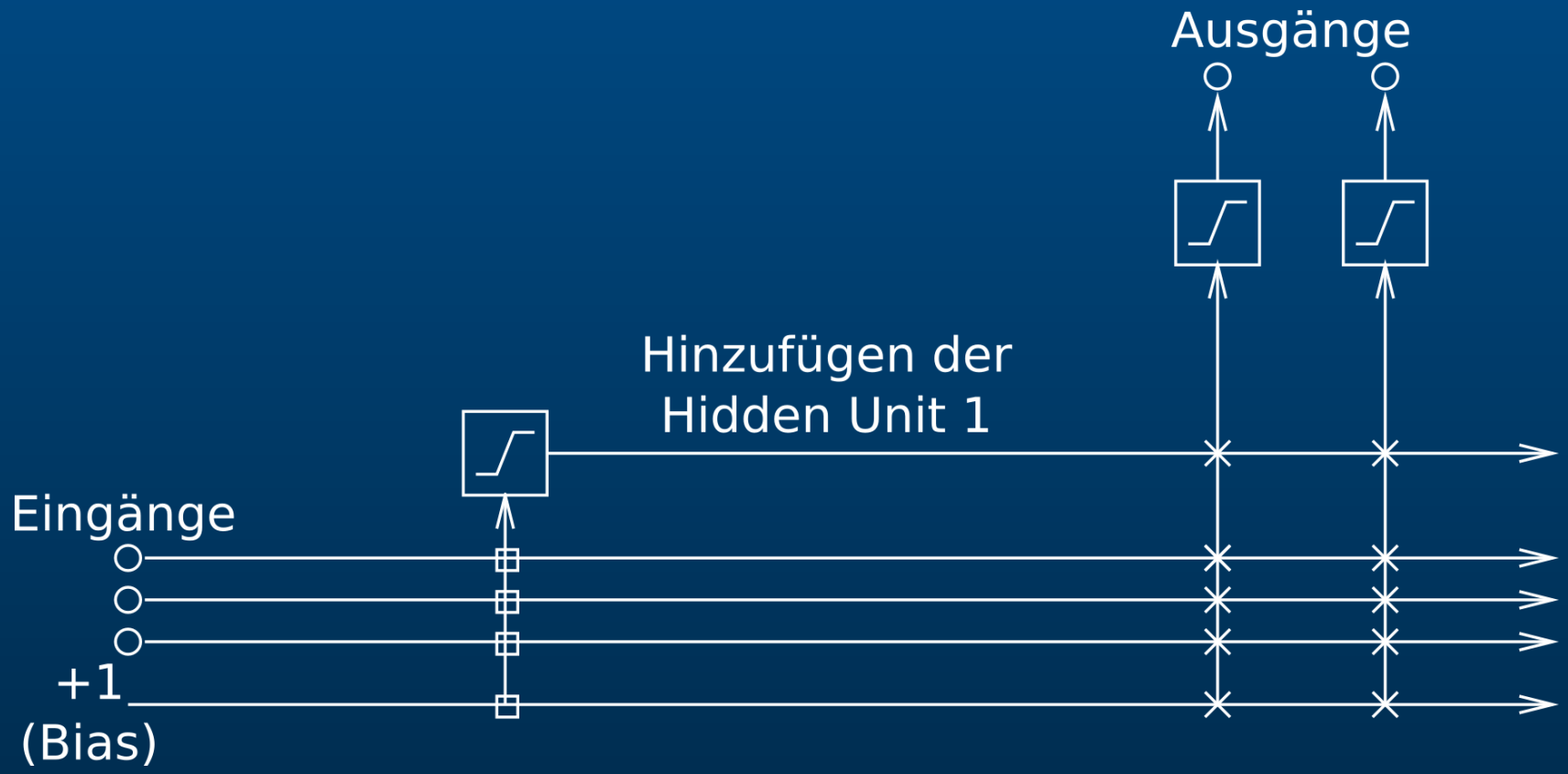
Cascade Correlation

- Das Netz wird “nach Bedarf” aufgebaut, es gibt keine vordefinierte Struktur
- Jedes Neuron wird einzeln trainiert und bezieht sich auf das Ergebniss aller vorigen → Bessere Erkennung der Eigenschaften des Inputs
- Bei den Trainingsschritten müssen nur die Outputwerte des neuen Neurons neu berechnet werden, alles andere ist konstant → Benutzung von Caches möglich

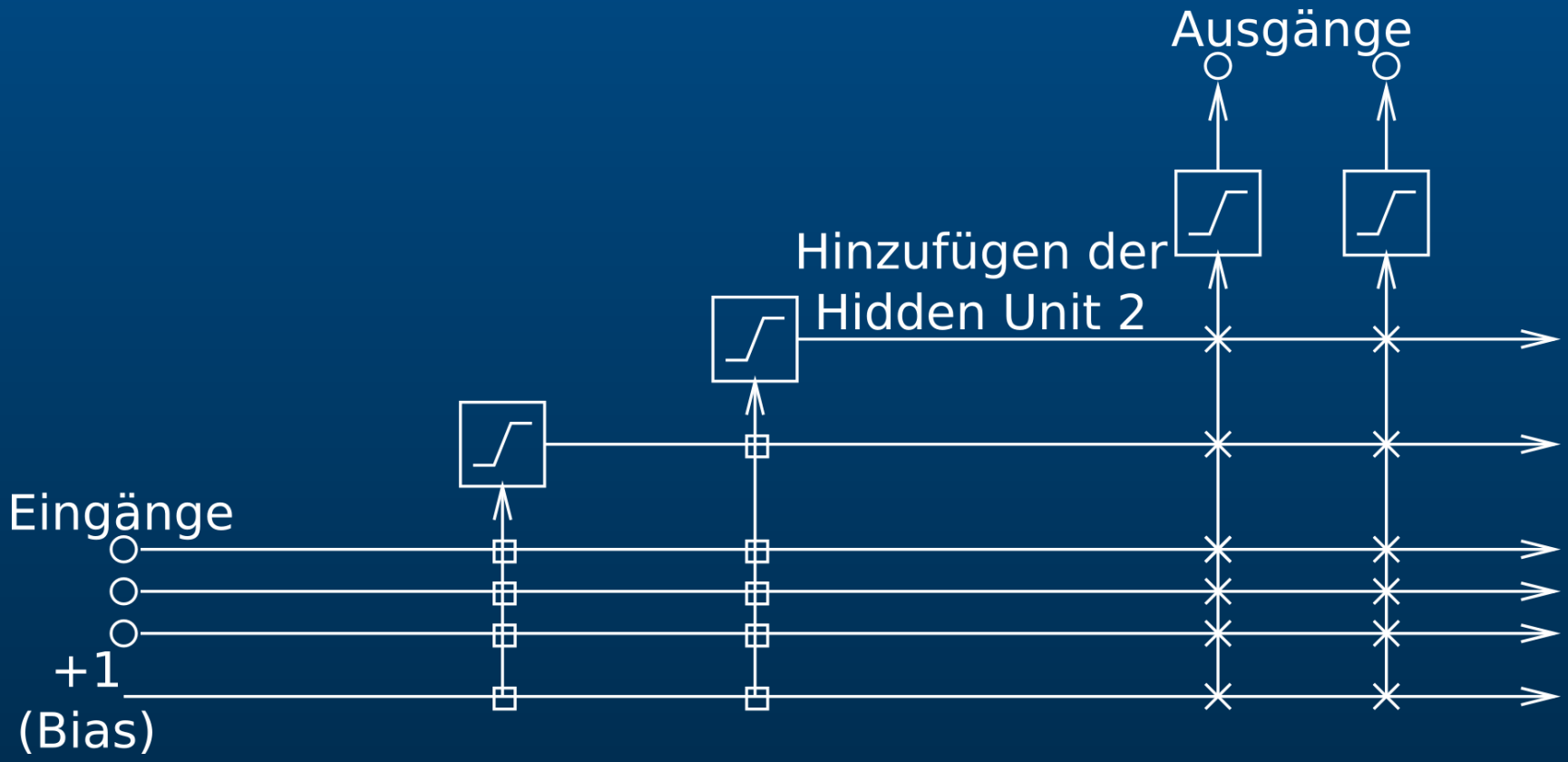
Cascade Correlation – Aufbau



Cascade Correlation – Aufbau



Cascade Correlation – Aufbau



Cascade Correlation – Training

- Ziel: Maximierung der Korrelation S zwischen den Ausgangswerten der Kandidaten und dem Fehler der Ausgangsneuronen

$$\text{Korrelation: } S = \sum_{\dot{o}} \left| \sum_{p} (o_{p,k} - avg(o_k))(\widehat{E}_{p,\dot{o}} - avg(\widehat{E}_{\dot{o}})) \right|$$

Änderung der Gewichte:

$$\Delta \omega_{ik} = \varepsilon \frac{\partial S}{\partial \omega_{ik}} = \varepsilon \sum_{\dot{o}} \sum_{p} \sigma_o (\widehat{E}_{p,\dot{o}} - avg(\widehat{E}_{\dot{o}})) f'_p(o_{p,i})$$

Cascade Correlation – Training

- Mehrere Kandidaten werden gleichzeitig trainiert → höhere Wahrscheinlichkeit, ein optimales Neuron zu finden
- Der beste Kandidat wird in das Netz eingefügt und die Gewichte der Ausgangsneuronen mit einem geeigneten Algorithmus (Delta-Regel, Backpropagation with Momentum, etc) angepasst

Cascade Correlation – Training

- Mögliche Optimierungen
  - Kandidaten mit verschiedenen Aktivierungsfunktionen
  - Anwendung von Verbesserungen wie z.B. bei “Backpropagation with Momentum” beim Trainieren der Kandidaten → Schnellere Maximierung von S

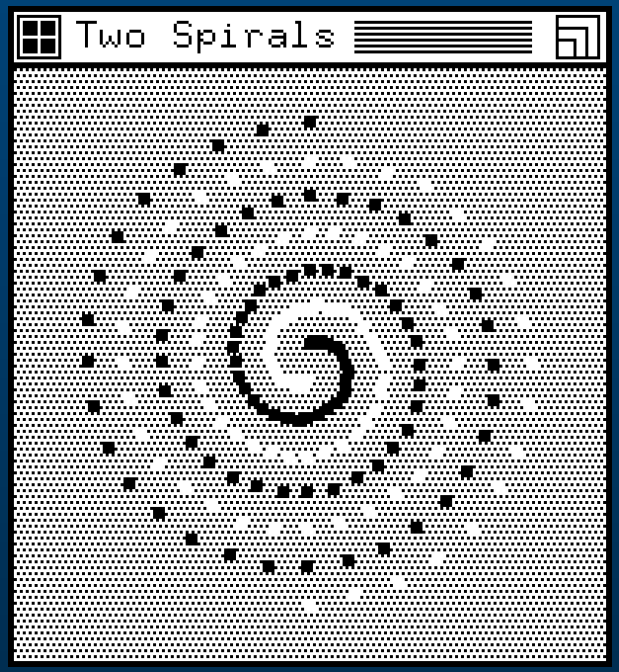
Cascade Correlation – Performanz

Das "Two Spirals"-Problem

- Benchmark für überwachte Lernalgorithmen
- Bestehend aus 194 (x,y)-Inputs mit dazugehörigen Outputs (1 oder -1)
- Komplex wegen der Nichtlinearität der Werte

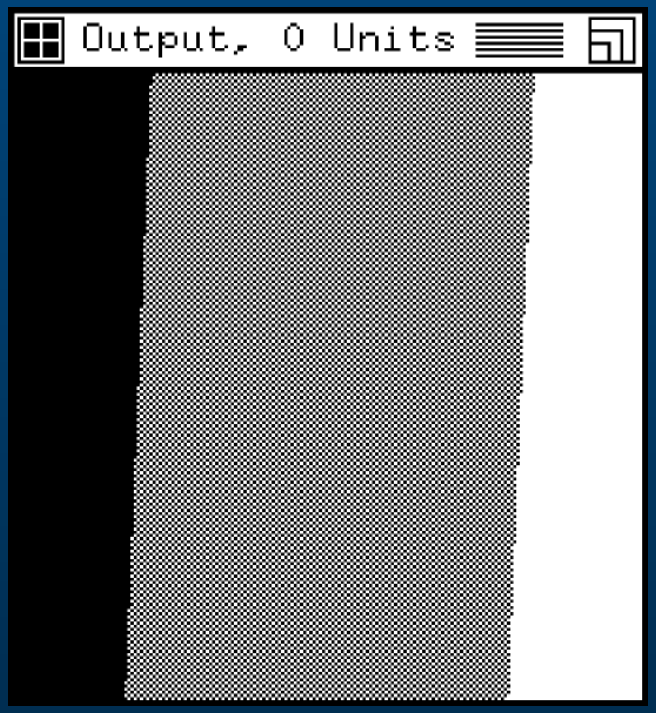
Cascade Correlation – Performanz

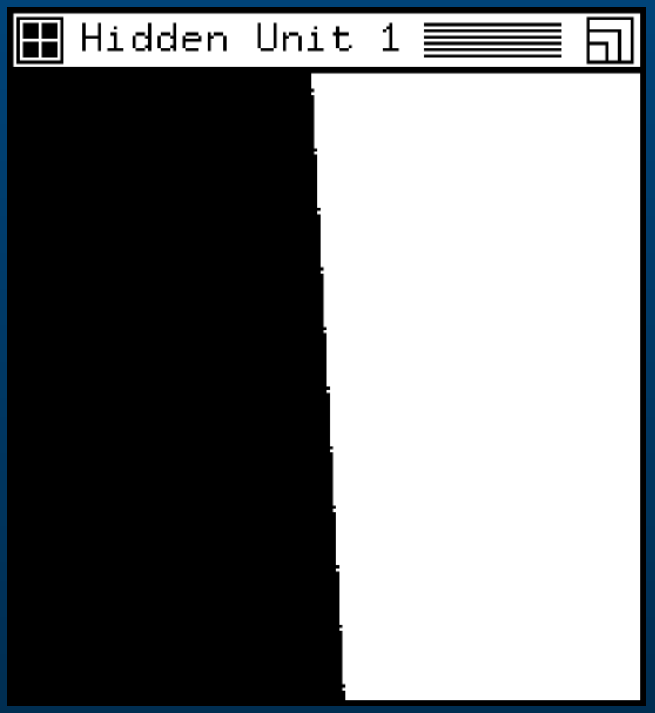
Das “Two Spirals”-Problem



Cascade Correlation – Performanz

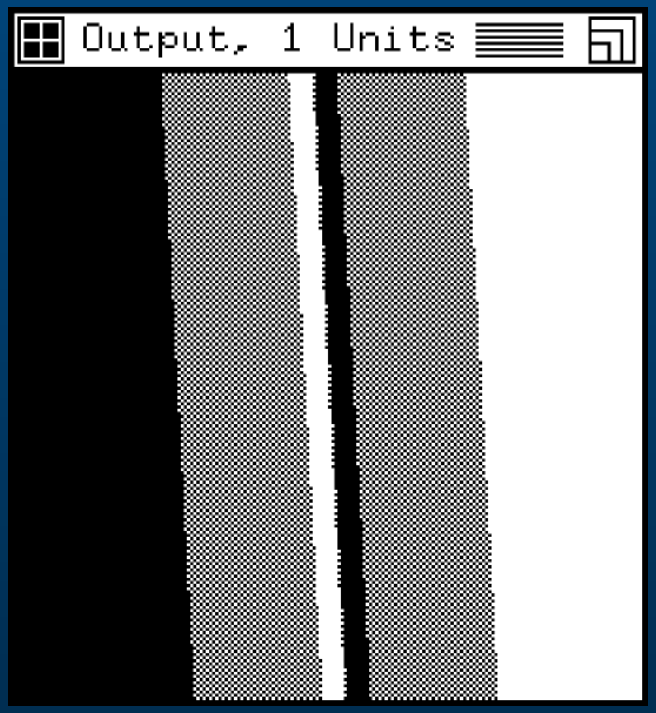
Das “Two Spirals”-Problem – Training

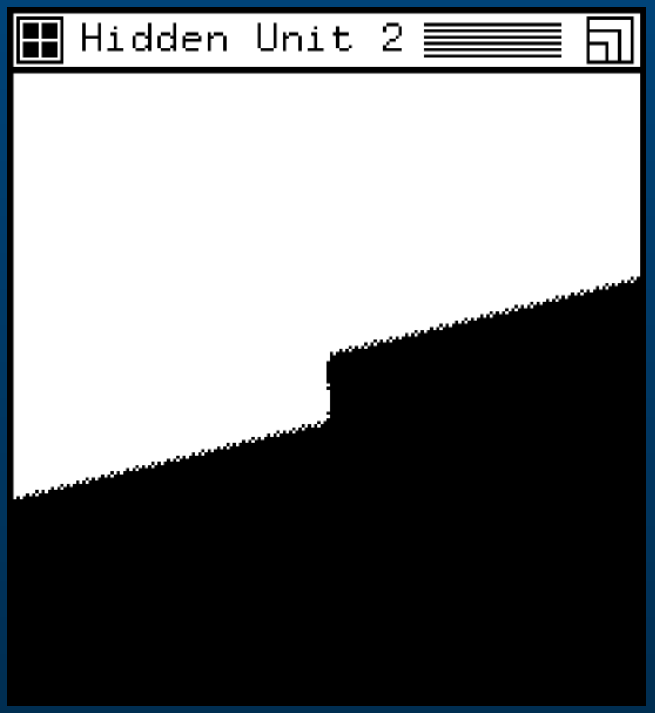




Cascade Correlation – Performanz

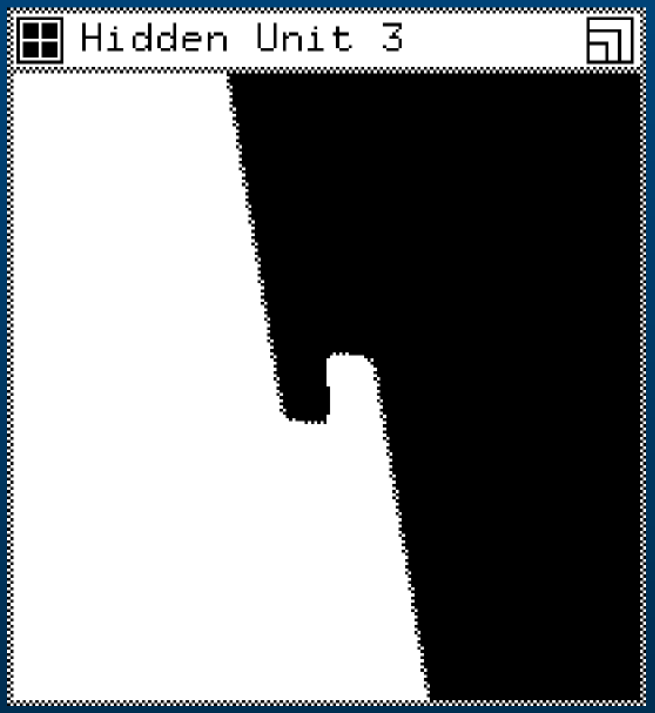
Das “Two Spirals”-Problem – Training





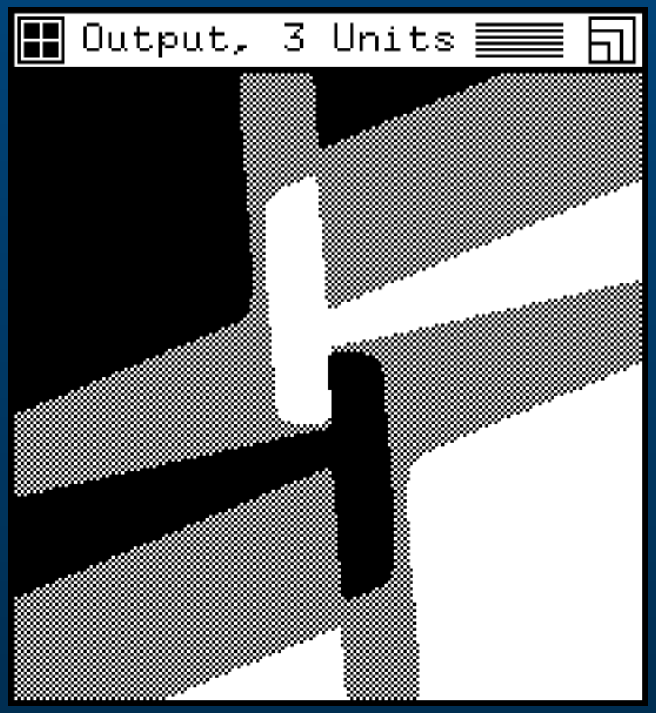
Cascade Correlation – Performanz

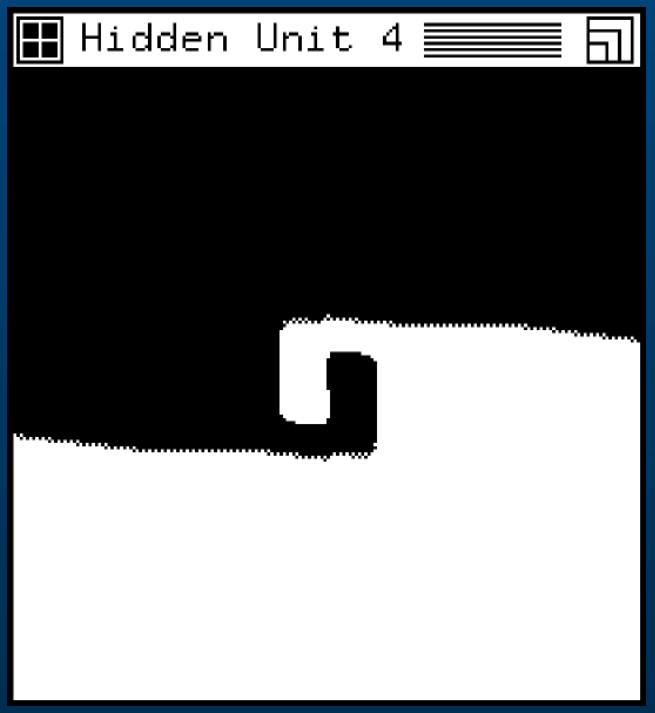
Das “Two Spirals”-Problem – Training



Cascade Correlation – Performanz

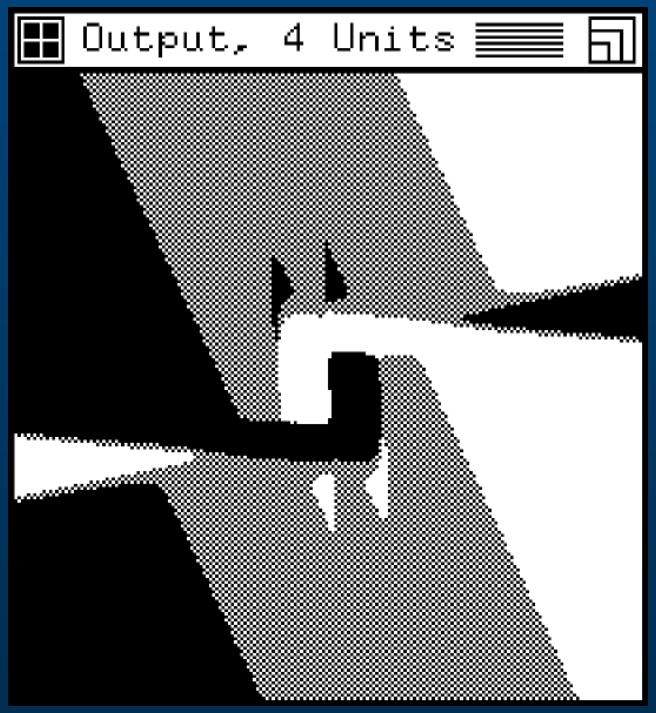
Das “Two Spirals”-Problem – Training

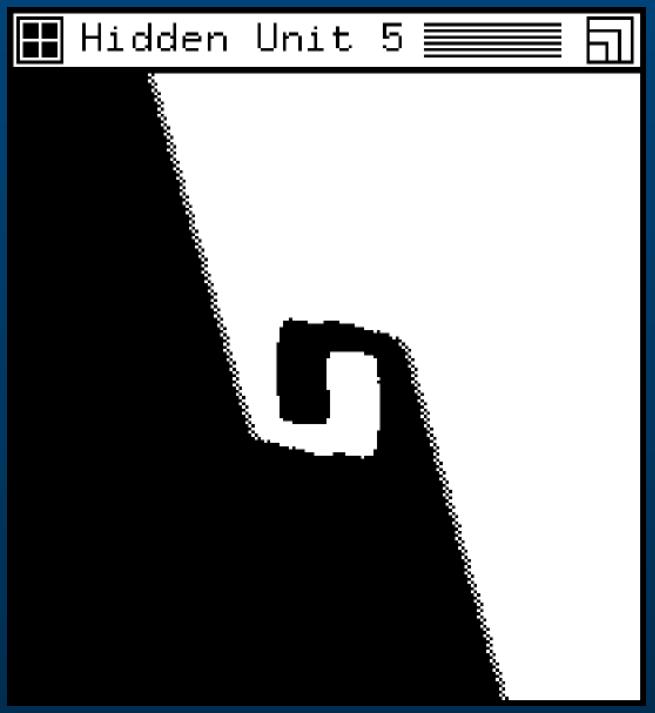




Cascade Correlation – Performanz

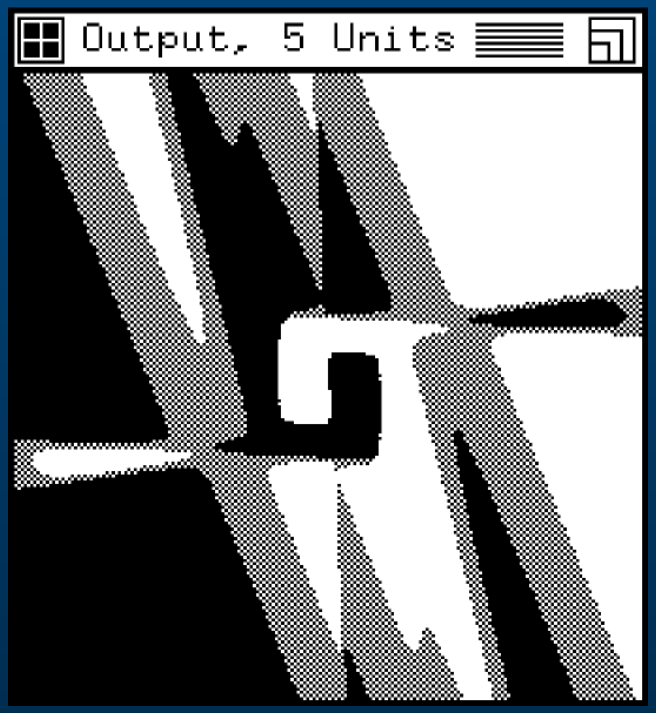
Das “Two Spirals”-Problem – Training





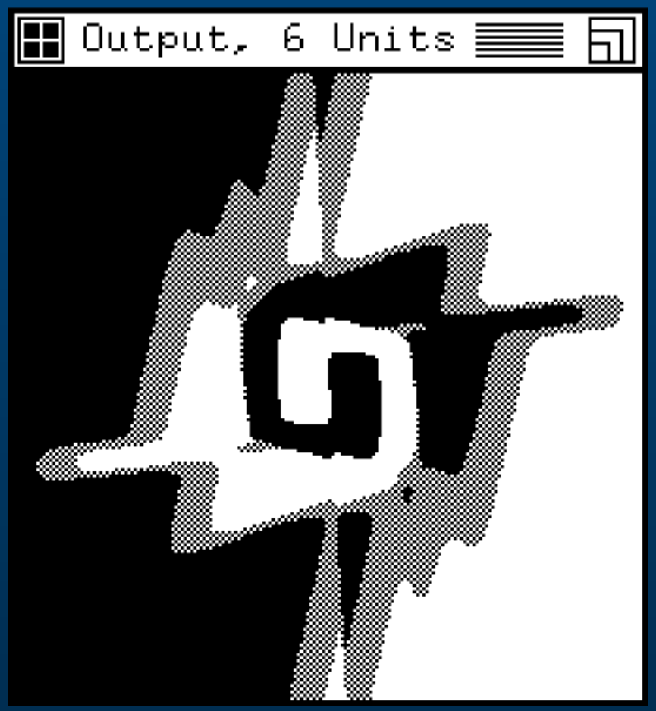
Cascade Correlation – Performanz

Das “Two Spirals”-Problem – Training



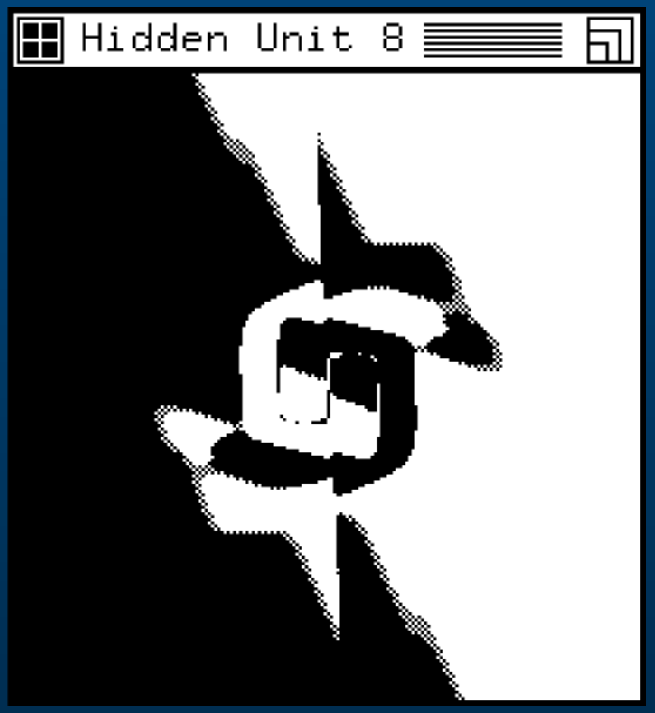
Cascade Correlation – Performanz

Das “Two Spirals”-Problem – Training



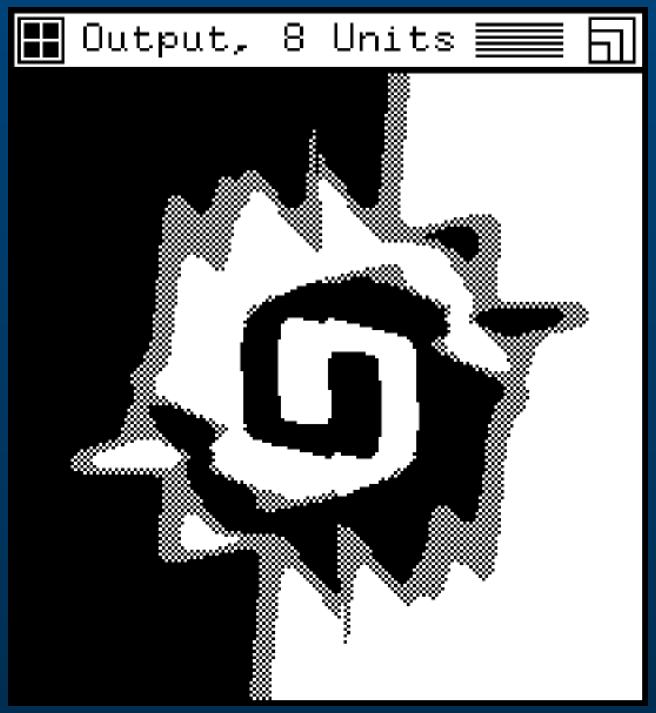
Cascade Correlation – Performanz

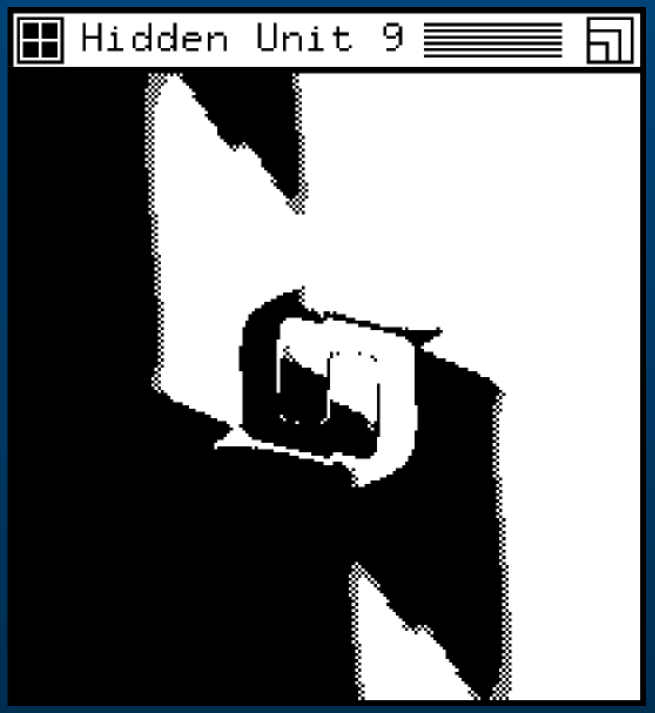
Das “Two Spirals”-Problem – Training



Cascade Correlation – Performanz

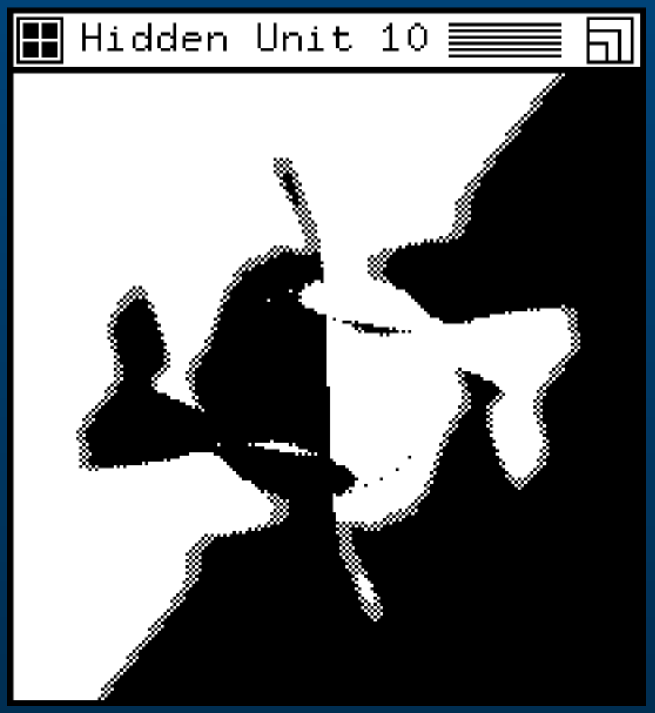
Das “Two Spirals”-Problem – Training





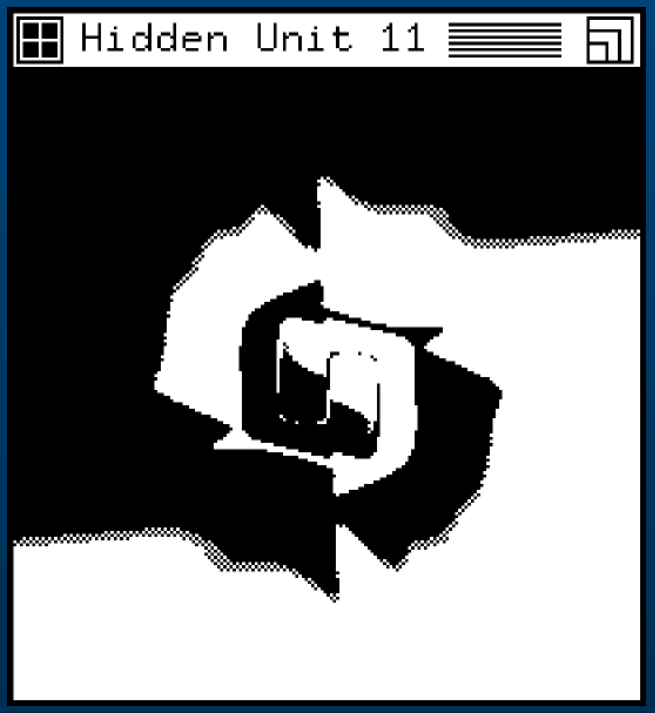
Cascade Correlation – Performanz

Das “Two Spirals”-Problem – Training



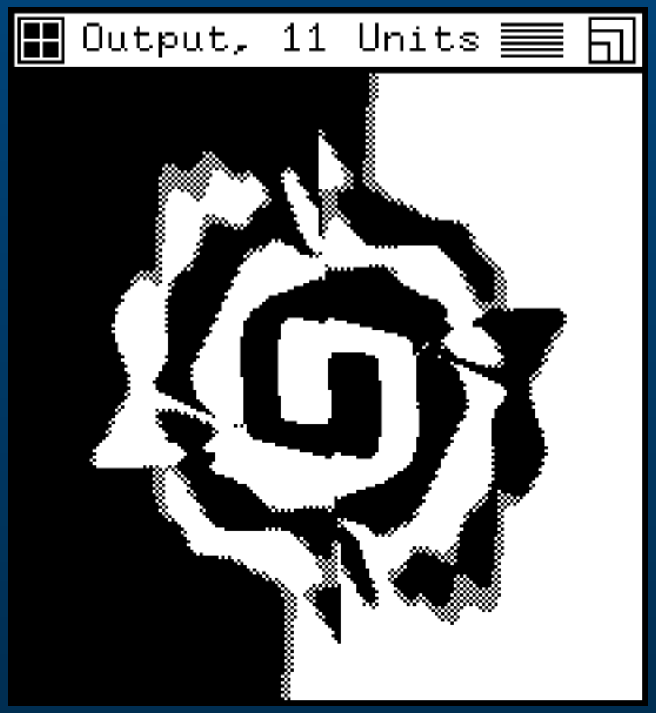
Cascade Correlation – Performanz

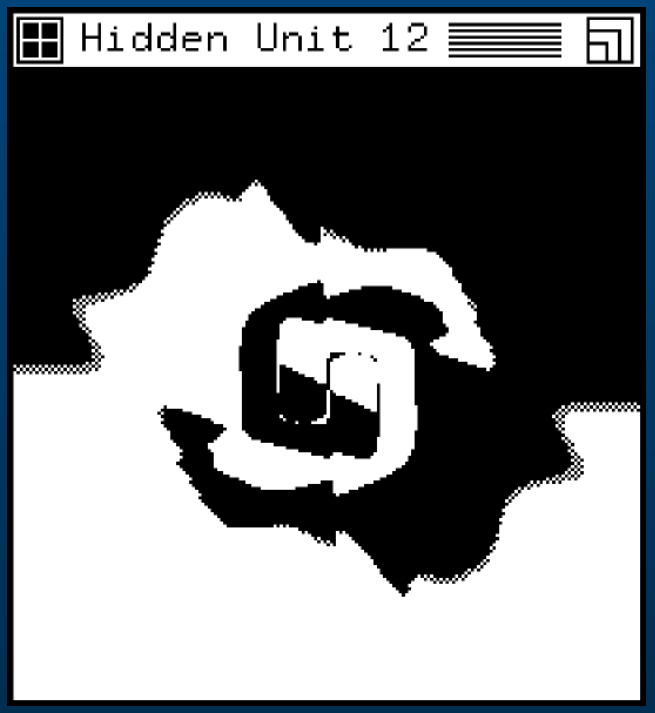
Das “Two Spirals”-Problem – Training



Cascade Correlation – Performanz

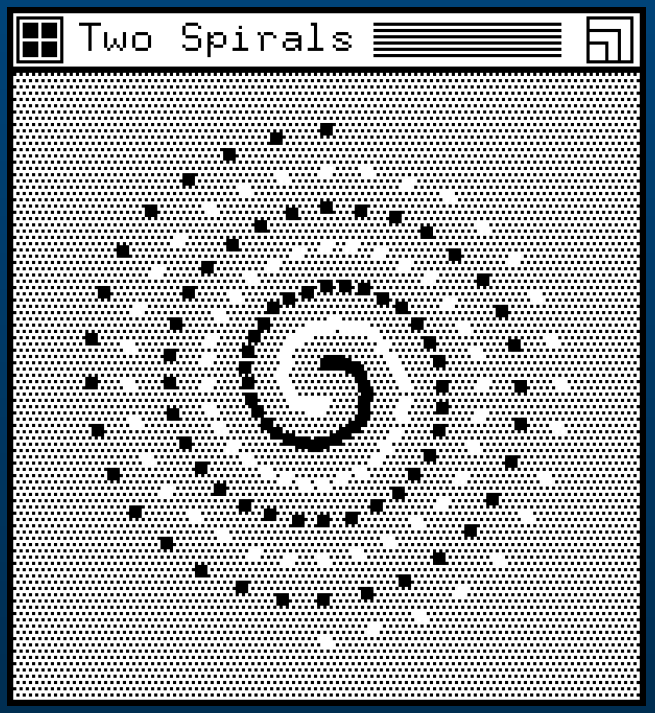
Das “Two Spirals”-Problem – Training





Cascade Correlation – Performanz

Das “Two Spirals”-Problem – Training



Cascade Correlation – Performanz

Das "Two Spirals"-Problem – Ergebnisse

- Trainingsepochen bis zum korrekten Ergebnis:
  - Cascade Correlation: durchschnittlich 1700 Epochen
  - Modifizierte Version von Backpropagation: 150000 bis 200000 Epochen
  - Quickprop-Algorithmus mit Verbindungen aus allen vorigen Layers: 8000 Epochen

Fragen?