Informatik IV - Tutorium XII & XIII (SR -120)

Tut Nr. 6 – Lernen von NN & Hopfield-Netze

David Münch

Universität Karlsruhe (TH)
Fakultät für Informatik
IBDS Prautzsch

5. Juni 2008

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

https://www.stud.uni-karlsruhe.de/~uhbro/

Tutorium 12: Donnerstags 8:00 Uhr - Raum -120
Tutorium 13: Donnerstags 9:45 Uhr - Raum -120

Übungsblattabgabe Donnerstag.

Schein / Übungsblätter

Nur die mit (**K**) gekennzeichneten Aufgaben sind abzugeben.
Diese werden korrigiert und bewertet.
66% der Punkte aller mit (**K**) gekennzeichneten Aufgaben aller Übungsblätter sind notwendig, um einen Schein zu erhalten.

Schein / Übungsblätter

Nur die mit (**K**) gekennzeichneten Aufgaben sind abzugeben.
Diese werden korrigiert und bewertet.
66% der Punkte aller mit (**K**) gekennzeichneten Aufgaben aller Übungsblätter sind notwendig, um einen Schein zu erhalten.

Abgabe meistens Donnerstags.

Schein / Übungsblätter

Nur die mit (**K**) gekennzeichneten Aufgaben sind abzugeben.
Diese werden korrigiert und bewertet.
66% der Punkte aller mit (**K**) gekennzeichneten Aufgaben aller Übungsblätter sind notwendig, um einen Schein zu erhalten.

Abgabe meistens Donnerstags.

Abgabe in Zweiergruppe erlaubt und ausdrücklich erwünscht!

Schein / Übungsblätter

Nur die mit (**K**) gekennzeichneten Aufgaben sind abzugeben. Diese werden korrigiert und bewertet.
66% der Punkte aller mit (**K**) gekennzeichneten Aufgaben aller Übungsblätter sind notwendig, um einen Schein zu erhalten.

Abgabe meistens Donnerstags.

Abgabe in Zweiergruppe erlaubt und ausdrücklich erwünscht!

Um das Übungsteam zu unterstützen bitte folgendes Deckblatt verwenden:
`http://www.stud.uni-karlsruhe.de/~unbdh/deckblatt/index.php?course=5`

Literatur

Boehm, Prautzsch: Numerical Methods. AK Peters 1993. ISBN 3-528-06350-5
http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&fbt=7319953&nd=3204657

Ash: Information Theory. Dover 1990. ISBN 0-486-66521-6
http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&nd=9866904

Goos: Vorlesungen über Informatik. Bd. 4, Springer 1998. ISBN 3-540-60650-5
http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&fbt=9316367&nd=6568301

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Hausaufgabenklausur besprechen

Was wollen wir heute erreichen?

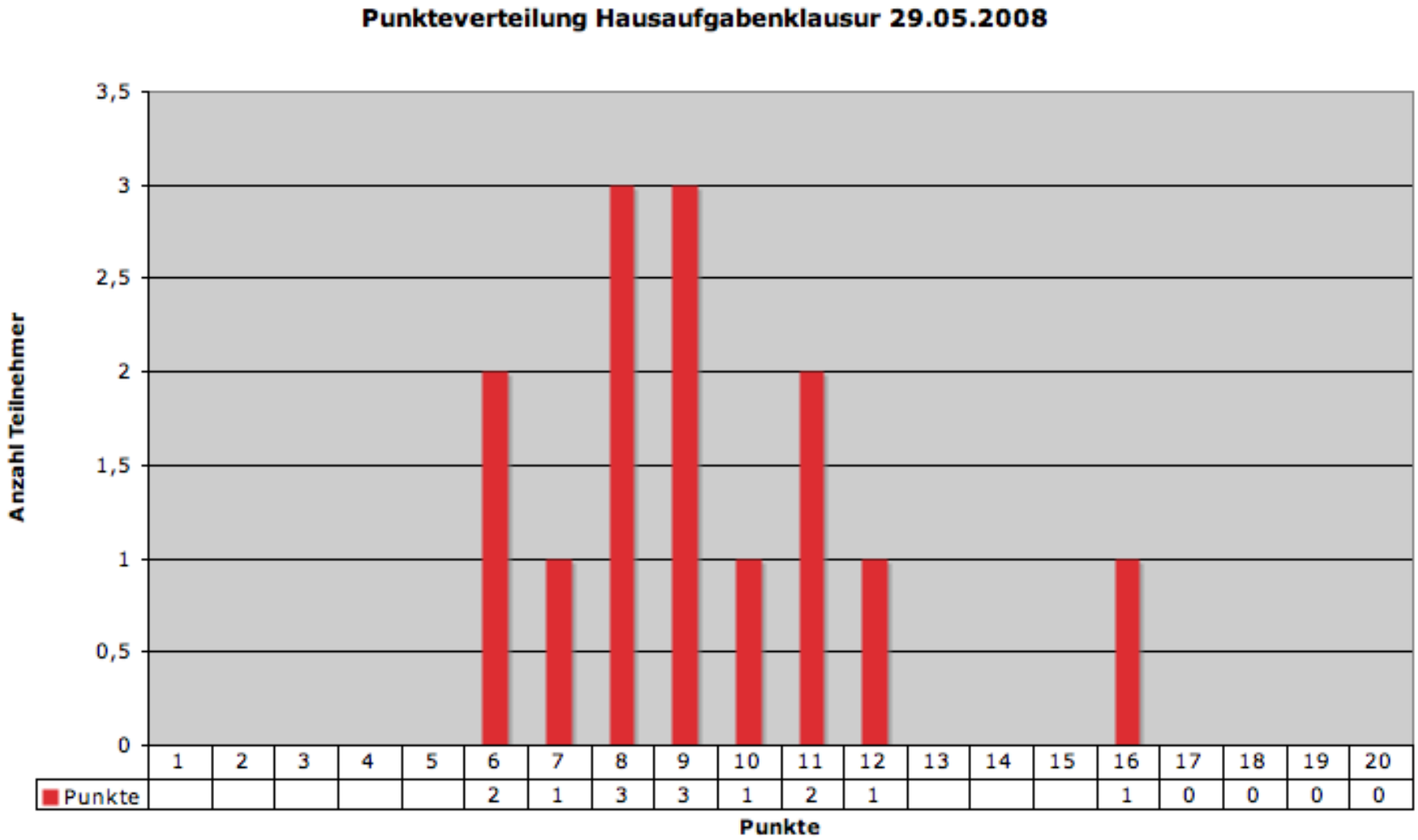
- Hausaufgabenklausur besprechen
- Lernen von einfachen Perzeptronen

Was wollen wir heute erreichen?

- Hausaufgabenklausur besprechen
- Lernen von einfachen Perzeptronen
- Weitere Lernkonzepte von Neuronalen Netzen

Was wollen wir heute erreichen?

- Hausaufgabenklausur besprechen
- Lernen von einfachen Perzeptronen
- Weitere Lernkonzepte von Neuronalen Netzen
- Hopfiled-Netze


Punkteverteilung Hausaufgabenklausur 29.05.2008
Anzahl Teilnehmer
3,5
3
2,5
2
1,5
1
0,5
0
Punkte
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Punkte 2 1 3 3 1 2 1 1 0 0 0 0

Aufgabe 1

Gegeben seien die Punkte

$$\mathbf{x}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{und} \quad \mathbf{x}_3 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

(1) Berechnen Sie mit dem Simplexverfahren die linerare Funktion $f(x) = ax + b$, die

$$\max_{i=1}^{3} |ax_i + b - y_i|$$

minimiert.

Aufgabe 1

(2) Berechnen Sie mit der Methode der kleinsten Quadrate die linerare Funktion $f(x) = ax + b$, die

$$\sum_{i=1}^{3}(ax_i + b - y_i)^2$$

minimiert.

Aufgabe 1

(3) Berechnen Sie Aufgabenteil (2) unter der Nebenbedingung $a + b = 1$.

Aufgabe 1

(4) Zeigen Sie, dass die durch $f(x)$ in (2) dargestellte Gerade nicht die Ausgleichsgerade zu $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ ist.
(*Hinweis:* Führen Sie die Berechnung der Ausgleichsgerade so weit durch, bis Sie zeigen können, dass sie eine andere Richtung hat.)

Aufgabe 2

Formulieren Sie die Akzeptanzbedingungen für die Maximierung einer Kostenfunktion $c(x)$ mit dem simulierten Tempern, dem Schwellwert-, dem Sintflut- und dem Rekordjagd-Algorithmus. Wie sind die Schwellwerte bzw. Temperaturen im Laufe des Verfahrens zu ändern?

Aufgabe 3

Formulieren Sie das Musterlernen für einfache Perzeptronen mit 0/1-Neuronen und beweisen Sie, dass das Verfahren konvergiert und eine Lösung gefunden wird, wenn die Lernmuster linear separierbar sind.
(*Hinweis:* Orientieren Sie sich am Verfahren für $-1/1$-Neuronen.)

Aufgabe 4

Konstruieren Sie für jede der folgenden boolschen Funktionen ein möglichst einfaches neuronales Netz, das diese berechnet. Benutzen Sie binäre Neuronen. Der Wert 1 soll dabei TRUE bedeuten; der Wert 0 bzw. -1 den Wahrheitswert FALSE.

(1) Modellieren Sie die OR-Funktion $f(A, B) = A \vee B$ mit 0/1-Neuronen.

Aufgabe 4

(2) Modellieren Sie die XOR-Funktion $f(A, B) = A \text{ XOR } B$ mit $-1/1$-Neuronen.

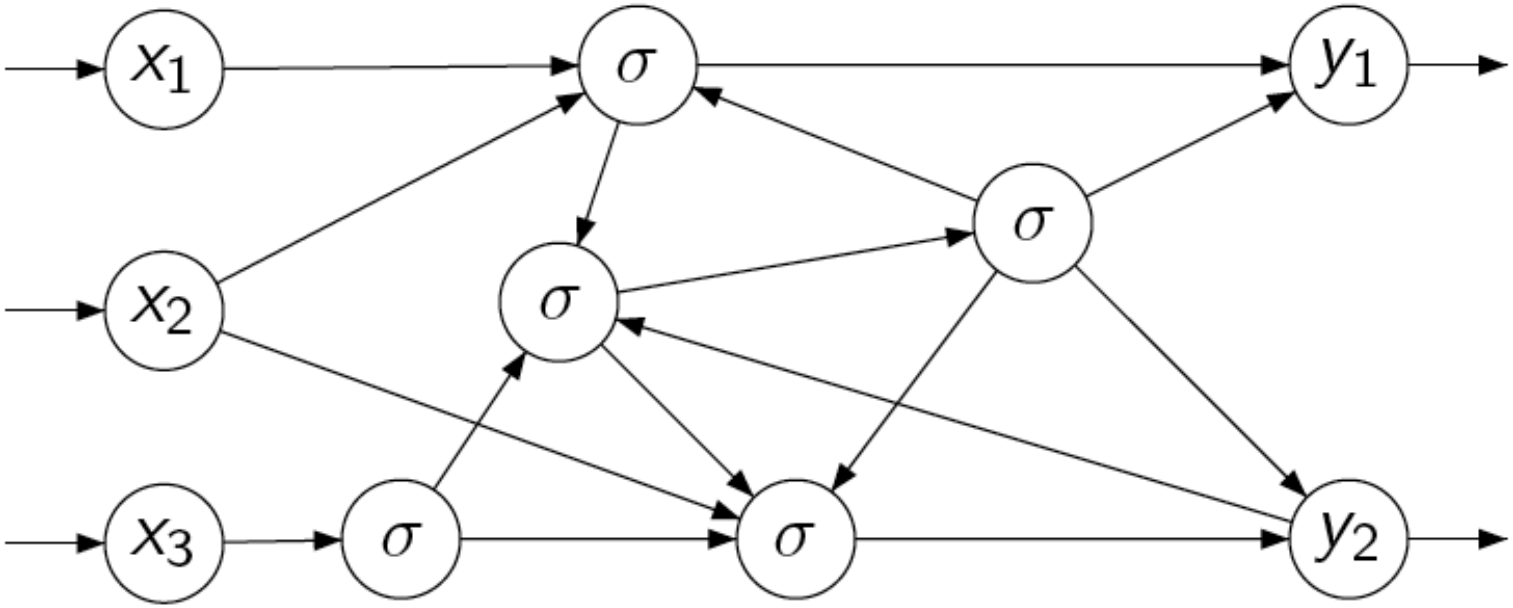
Aufgabe 4

(3) Modellieren Sie die Funktion $f(A, B) = (A \Rightarrow \neg B)$ mit $-1/1$-Neuronen.

Lernen von Neuronalen Netzen

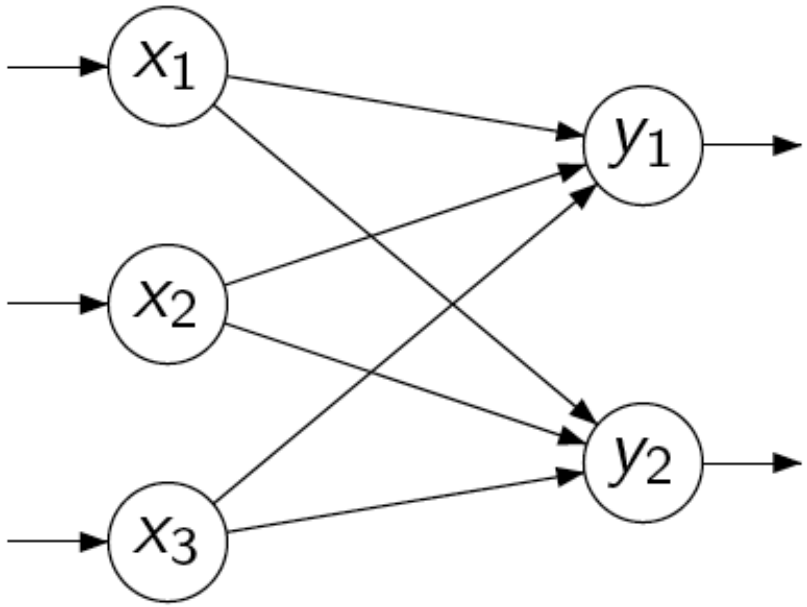
Definition: Neuronales Netz

Ein Neuronales Netz besteht aus einer Menge von Neuronen und einer Menge von gerichteten Kanten. Jeder Kante zwischen den Neuronen N_i und N_j ist ein Gewicht w_{ij} zugeordnet. Jedes Neuron N_j hat ausserdem einen Schwellwert σ_j.



Definition: Perzeptron

Das Perzeptron ist ein vorwärtsgerichtetes Neuronales Netz, bei dem die Neuronen schichtweise angeordnet sind. Verbindungen existieren nur zwischen benachbarten Schichten. Das einfache Perzeptron besitzt nur eine Eingabe- sowie eine Ausgabeschicht.



Notation

Die Gewichte der Kanten werden in einer Gewichtsmatrix $W = [w_{ij}]_{i,j=0}^{n}$ gespeichert.

Der Zustandvektor sei $q = \begin{bmatrix} q_0 \\ \vdots \\ q_n \end{bmatrix}$.

Dann berechnet sich der neue Zustand zum Zeitpunkt $t + 1$ so:

$$q(t+1) = \boldsymbol{h}(W^t q(t))$$

mit $h(x)$ bei $0/1$-Neuronen komponentenweise:

$$h(x) = \begin{cases} 1, & \text{falls } x \geq 0 \text{ oder } x > 0 \\ 0, & \text{sonst} \end{cases}$$

Einfaches Perzeptron

Ein einfaches Perzeptron besteht nur aus einer Schicht, der Eingabeschicht mit Neuronen $E_1, ..., E_m$ und der Ausgabeschicht mit Neuronen $A_1, ..., A_n$.

$\boldsymbol{e} = \begin{bmatrix} e_1 \\ \vdots \\ q_m \end{bmatrix}$ und $\boldsymbol{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$ seien der Vektor der Eingabezustände

und der Ausgabezustände.
Die Ausgabe eines Neurons A_j berechnet sich wie folgt:

$$a_j = \boldsymbol{h}(\begin{bmatrix} w_{1j} \\ \vdots \\ w_{mj} \end{bmatrix}^t \boldsymbol{e} - \sigma_j)$$

Musterlernen von einfachen Perzeptronen

Gegeben sind Lernmuster $h = (\boldsymbol{e}_\mu, \boldsymbol{a}_\mu)$.
Gesucht ist nun eine Gewichtsmatrix W, sodass $\boldsymbol{h}(W\boldsymbol{e}_\mu - \sigma) = \boldsymbol{a}_\mu$ für alle μ.

Statt Schwellwerten, verwenden wir ein zusätzliches Neuron N_0
Weil wir es hier mit nur einem einfachen Perzeptron zu tun haben, reicht ein Ausgabeneuron aus.

Wir wollen also $q = \boldsymbol{h}(We_\mu - \sigma) \stackrel{!}{=} a_\mu$ optimieren.
Durch geschicktes Umformen kommt man zu der Perzeptronlernregel:

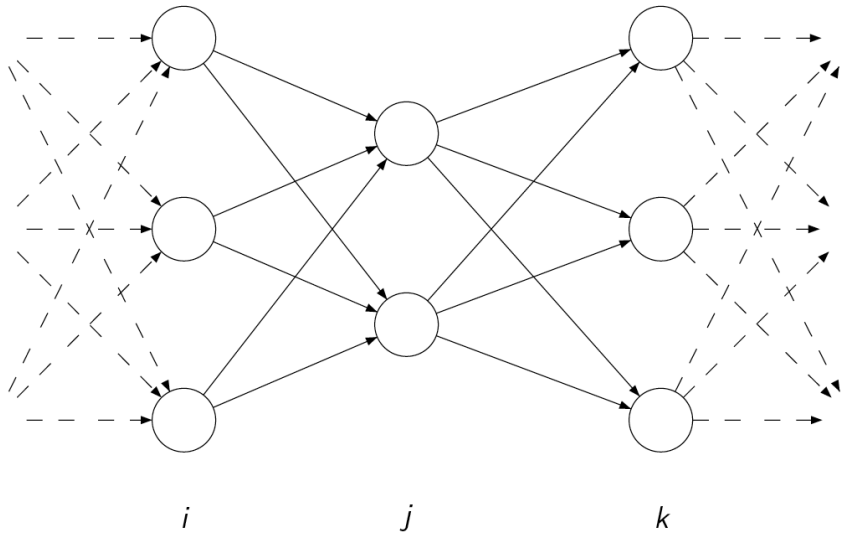
$$w := w - (q - a)\eta e$$

mit einer frei wählbaren Lernrate η.

Lernen für sigmoide Perzeptronen

Wie oben, aber h ist sigmoide Erregungsfunktion.
Wir verwenden hier eine Fehlerfunktion
$E(W) = \frac{1}{2} \sum_\mu ||\boldsymbol{q}_\mu - \boldsymbol{a}_\mu||^2$.
Je besser das Netz, umso kleiner $E(W)$.
Epochelernen: $W := W - \nabla E \eta \Leftrightarrow w_{ij} := w_{ij} - \frac{\partial E}{\partial w_{ij}} \eta$
Musterlernen: geht auch...

Mehrschichtige vorwärtsgerichtete Neuronale Netze



Satz

Mit zweischichtigen Perzeptronen kann jede berechenbare stetige Funktion beliebig gut angenähert werden, mit dreischichtigen Perzeptronen jede berechenbare Funktion.

Lernen für mehrschichtige Perzeptronen

Gegeben: Menge von Trainingsdaten und zugehöriger Klassifikation

Lernen für mehrschichtige Perzeptronen

Gegeben: Menge von Trainingsdaten und zugehöriger Klassifikation

Gesucht: Gewichte und Schwellwerte, dass das NN die Trainingsdaten richtig klassifiziert.

Lernen für mehrschichtige Perzeptronen

Gegeben: Menge von Trainingsdaten und zugehöriger Klassifikation
Gesucht: Gewichte und Schwellwerte, dass das NN die Trainingsdaten richtig klassifiziert.
Problem: Die gewünschte Ausgabe der versteckten Neronen ist unbekannt.

Lernen für mehrschichtige Perzeptronen

Gegeben: Menge von Trainingsdaten und zugehöriger Klassifikation
Gesucht: Gewichte und Schwellwerte, dass das NN die Trainingsdaten richtig klassifiziert.
Problem: Die gewünschte Ausgabe der versteckten Neronen ist unbekannt.
Lösung: Error-Backpropagation-Algorithmus

Lernen für mehrschichtige Perzeptronen

Gegeben: Menge von Trainingsdaten und zugehöriger Klassifikation
Gesucht: Gewichte und Schwellwerte, dass das NN die Trainingsdaten richtig klassifiziert.
Problem: Die gewünschte Ausgabe der versteckten Neronen ist unbekannt.
Lösung: Error-Backpropagation-Algorithmus
Idee: Definiere eine von den Gewichten abhängige Fehlerfunktion und minimiere deren Wert mittels Gradientenabstieg.

Lernen für mehrschichtige Perzeptronen

Gegeben: Menge von Trainingsdaten und zugehöriger Klassifikation
Gesucht: Gewichte und Schwellwerte, dass das NN die Trainingsdaten richtig klassifiziert.
Problem: Die gewünschte Ausgabe der versteckten Neronen ist unbekannt.
Lösung: Error-Backpropagation-Algorithmus
Idee: Definiere eine von den Gewichten abhängige Fehlerfunktion und minimiere deren Wert mittels Gradientenabstieg.
Algorithmus:
1. Forward-Pass (Schichtweises Durchreichen der Eingabe)
2. Bestimmen des Fehlers
3. Backpropagation (Rückrechnen des Fehlers)
4. Gewichte anpassen

Lernen von Neuronalen Netzen

- Erregungsfunktion für ein Neuron N_j

$$p_j = \sum_i w_{ij} q_i$$

- Erregungsfunktion für ein Neuron N_j und zu min. Fehler:

$$p_j = \sum_i w_{ij} q_i \qquad E(W) = \frac{1}{2} \sum_j (a_j - q_j)^2$$

- Erregungsfunktion für ein Neuron N_j und zu min. Fehler:

$$p_j = \sum_i w_{ij} q_i \qquad E(W) = \frac{1}{2} \sum_j (a_j - q_j)^2$$

- Gradientenabstieg:

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

- Erregungsfunktion für ein Neuron N_j und zu min. Fehler:

$$p_j = \sum_i w_{ij} q_i \qquad E(W) = \frac{1}{2} \sum_j (a_j - q_j)^2$$

- Gradientenabstieg:

$$w_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}}$$

- Erregungsfunktion für ein Neuron N_j und zu min. Fehler:

$$p_j = \sum_i w_{ij} q_i \qquad E(W) = \frac{1}{2} \sum_j (a_j - q_j)^2$$

- Gradientenabstieg:

$$w_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}}$$

- Fehler:

$$\delta_j := -\frac{\partial E}{\partial p_j}$$

- Erregungsfunktion für ein Neuron N_j und zu min. Fehler:

$$p_j = \sum_i w_{ij} q_i \qquad E(W) = \frac{1}{2} \sum_j (a_j - q_j)^2$$

- Gradientenabstieg:

$$w_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}} = w_{ij} + \underbrace{\eta \delta_j q_i}_{=\Delta w_{ij}}$$

- Fehler:

$$\delta_j := -\frac{\partial E}{\partial p_j}$$

- Erregungsfunktion für ein Neuron N_j und zu min. Fehler:

$$p_j = \sum_i w_{ij} q_i \qquad E(W) = \frac{1}{2} \sum_j (a_j - q_j)^2$$

- Gradientenabstieg:

$$w_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}} = w_{ij} + \underbrace{\eta \delta_j q_i}_{=\Delta w_{ij}}$$

- Fehler:

$$\delta_j := -\frac{\partial E}{\partial p_j} = -\frac{\partial E}{\partial q_j} \frac{\partial q_j}{\partial p_j}$$

- Erregungsfunktion für ein Neuron N_j und zu min. Fehler:

$$p_j = \sum_i w_{ij} q_i \qquad E(W) = \frac{1}{2} \sum_j (a_j - q_j)^2$$

- Gradientenabstieg:

$$w_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}} = w_{ij} + \underbrace{\eta \delta_j q_i}_{=\Delta w_{ij}}$$

- Fehler:

$$\delta_j := -\frac{\partial E}{\partial p_j} = -\frac{\partial E}{\partial q_j} \frac{\partial q_j}{\partial p_j}$$

- Berechnung von $-\partial E / \partial q_j$:

  Ausg.: $-\frac{\partial E}{\partial q_j} = (a_j - q_j)$

- Erregungsfunktion für ein Neuron N_j und zu min. Fehler:

$$p_j = \sum_i w_{ij} q_i \qquad E(W) = \frac{1}{2} \sum_j (a_j - q_j)^2$$

- Gradientenabstieg:

$$w_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} - \eta \cdot \frac{\partial E}{\partial w_{ij}} = w_{ij} + \underbrace{\eta \delta_j q_i}_{=\Delta w_{ij}}$$

- Fehler:

$$\delta_j := -\frac{\partial E}{\partial p_j} = -\frac{\partial E}{\partial q_j} \frac{\partial q_j}{\partial p_j}$$

- Berechnung von $-\partial E / \partial q_j$:

Ausg.: $-\frac{\partial E}{\partial q_j} = (a_j - q_j)$ Inn. Schicht: $-\frac{\partial E}{\partial q_j} = \sum_k \delta_k w_{jk}$

Hopfield-Netze

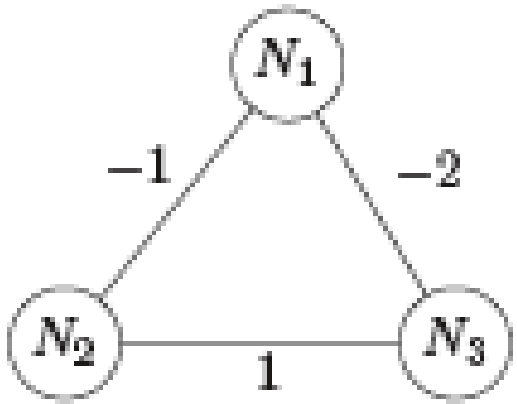
Ein Hopfield-Netz ist ein rekurrentes Neuronales Netz mit den folgenden Eigenschaften:

- Es besteht aus n Neuronen die paarweise miteinander verbunden sind
- Es gibt keine Schlingen (keine direkte Rückkopplung)
- Es gibt keine Eingabe- und Ausgabeneuronen
- Für die Gewichte gilt $w_{ij} = w_{ji}$

Ein Hopfield-Netz ist ein rekurrentes Neuronales Netz mit den folgenden Eigenschaften:

- Es besteht aus n Neuronen die paarweise miteinander verbunden sind
- Es gibt keine Schlingen (keine direkte Rückkopplung)
- Es gibt keine Eingabe- und Ausgabeneuronen
- Für die Gewichte gilt $w_{ij} = w_{ji}$

Beispiel:



Die Gewichte fassen wir als Matrix $W \in \mathbb{R}^{n \times n}$ und die Schwellwerte als Vektor $\boldsymbol{\sigma} \in \mathbb{R}^n$ auf. W ist eine symmetrische Matrix mit Nullen auf der Hauptdiagonalen.
Die Erregungsfunktion ist wie beim Perzeptron:

$$p_{ij} := \sum_{j=1}^{n} w_{ij} q_j - \sigma_i = W_i \boldsymbol{q} - \sigma_i$$

Mit $\boldsymbol{p} = (p_1, ..., p_n)^t$ gilt dann: $\boldsymbol{p} = W\boldsymbol{q} - \boldsymbol{\sigma}$.

Die Gewichte fassen wir als Matrix $W \in \mathbb{R}^{n \times n}$ und die Schwellwerte als Vektor $\boldsymbol{\sigma} \in \mathbb{R}^n$ auf. W ist eine symmetrische Matrix mit Nullen auf der Hauptdiagonalen.
Die Erregungsfunktion ist wie beim Perzeptron:

$$p_{ij} := \sum_{j=1}^{n} w_{ij} q_j - \sigma_i = W_i \boldsymbol{q} - \sigma_i$$

Mit $\boldsymbol{p} = (p_1, ..., p_n)^t$ gilt dann: $\boldsymbol{p} = W\boldsymbol{q} - \boldsymbol{\sigma}$.

Die Aktivierungsfunktion ist $q_i^+ = \begin{cases} 1 & \text{falls } p_i > 0 \\ -1 & \text{falls } p_i < 0 \\ q_i & \text{falls } p_i = 0 \end{cases}$

Das Netz ist stabil falls $q^+ = q$.

Varianten:

- synchron: Alle Neuronen schalten gleichzeitig
- asynchron: In einem Zeitpunkt schaltet ein zufälliges Neuron

Die Eingabe ist der Startzustand des Netzes.

Varianten:

- synchron: Alle Neuronen schalten gleichzeitig
- asynchron: In einem Zeitpunkt schaltet ein zufälliges Neuron

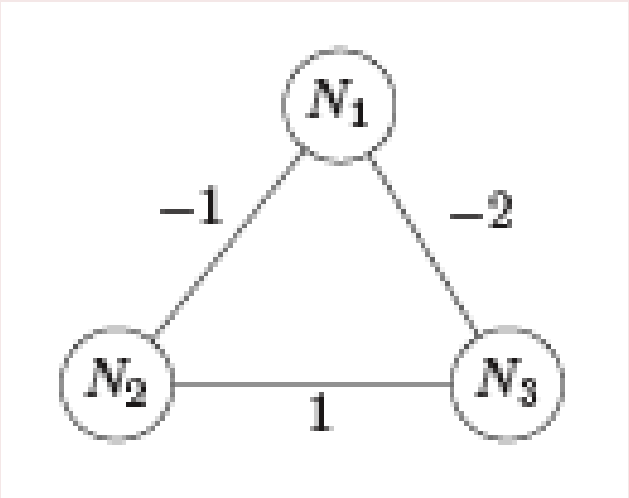
Die Eingabe ist der Startzustand des Netzes.

Satz

Ein Hopfield-Netz erreicht für jede Eingabe im asynchronen Betrieb nach endlich vielen Schritten einen stabilen Zustand.

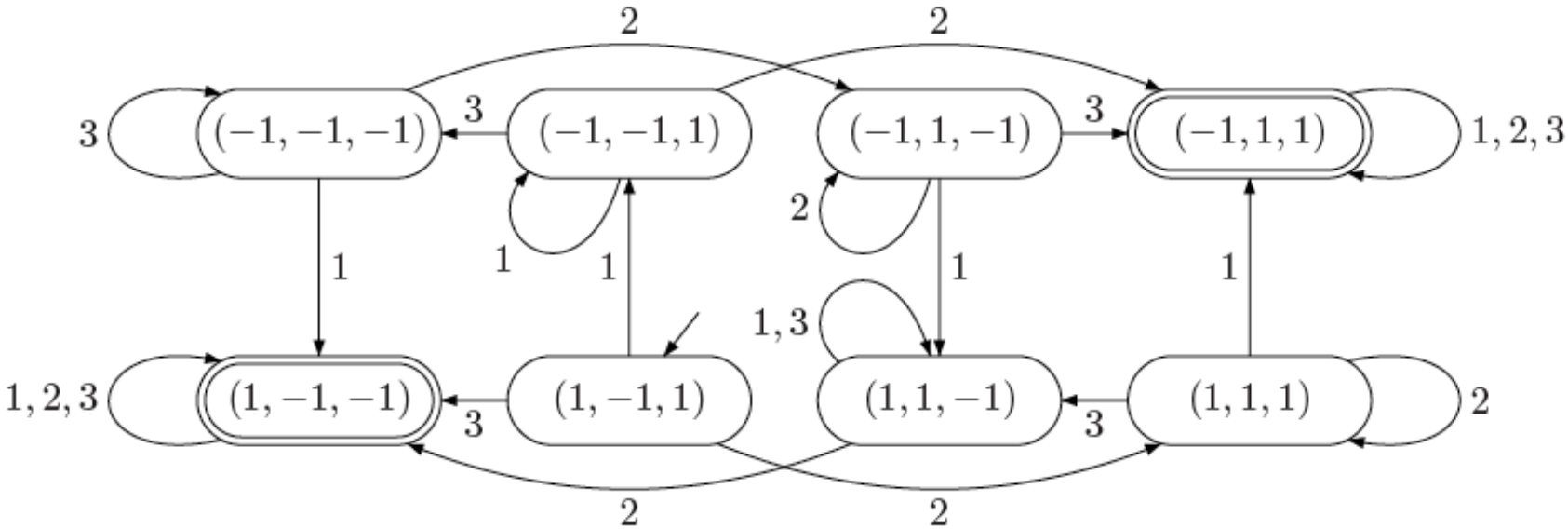
Aufgabe

Betrachte das hier abgebildete sehr einfache Hopfield-Netz



mit $\boldsymbol{\sigma} := (0, -1, 2)^t$.
Bestimme für dieses Netz ausgehend vom Anfangszustand $(q_1, q_2, q_3)^t = (1, -1, 1)^t$ den, bzw. die Endzustände, wenn das Netz asynchron arbeitet.
Hinweis: benutze einen Zustandsgraphen als Hilfe.

Lösung



Quellen

Pajor - Informatik 4 Tutorium SS2007
Prautzsch - Skript Informatik 4 SS2008
Goos, Vorlesungen über Informatik Band 4

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- Hausaufgabenklausur besprochen

Reflexion

Was haben wir heute gelernt?

- Hausaufgabenklausur besprochen
- Lernen von einfachen Perzeptronen verstanden

Reflexion

Was haben wir heute gelernt?

- Hausaufgabenklausur besprochen
- Lernen von einfachen Perzeptronen verstanden
- Hopfield-Netze

Reflexion

Was haben wir heute gelernt?

- Hausaufgabenklausur besprochen
- Lernen von einfachen Perzeptronen verstanden
- Hopfield-Netze
- Bis einschliesslich Kapitel 6 abgeschlossen

Noch Fragen?

Vorschau

Vorschau

- Informationsmass

Vorschau

- Informationsmass
- Codes

Bis zum nächsten Mal