AS1-Teil 4 Konkurrentes Lernen

Lernparadigmen

Das **Plastizität** vs**. Stabilitäts-Dilemma**

- **„weiche“, adaptierte Verbindungen**
  - ✓ **Schnelle Anpassung** auf veränderte Anforderungen bzw.Umgebung
  - ✓ **Bessere Reaktion** bei veränderter Umgebung
  - ▼ Bilden unspezifischer, generalisierter Mittelwertsreaktionen
  - ▼ **Keine** neue Reaktion (Klasse) möglich bei **stark veränd**. Umgebung
- **„harte“, inflexible, stabile Verbindungen**
  - ✓ einmal erlangte **gute Reaktion bleibt erhalten** trotz Ausreisser
  - ✓ Bei schlechter Passung (Test!) **Bildung neuer Klassen** möglich
  - ▼ **keine** Anpassung bei **leicht veränderter** Umgebung möglich

Idee: Lernen + neue Klassen durch Konkurrenzsituation: der Bessere übernimmt und lernt die Klasse

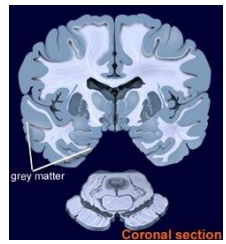
Somatotopische Karten

Selbstorganisierende Karten

Anwendung Sprachanalyse

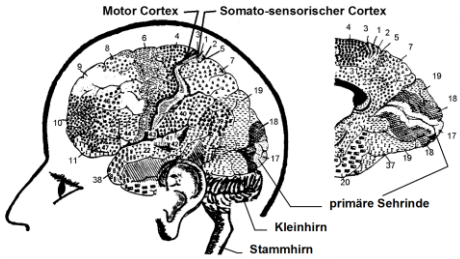
Anwendung Robotersteuerung

Vorbild Gehirn



Das Vorbild: Gehirnfunktionen

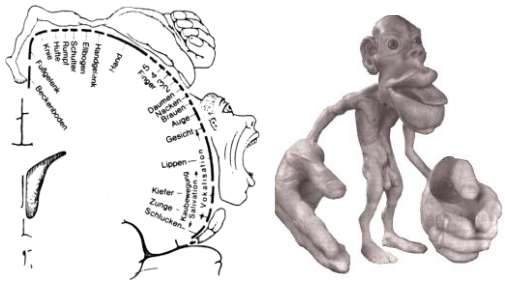
Unterteilung des Gehirns in funktionale Bereiche



Gehirn = 2-dim „Tuch“

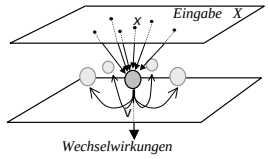
Somatotopie

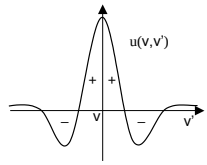
Beobachtung: Abbilder, Karten



Lokale Wechselwirkungen

Aktivität in kontinuierlichen Neuronenfeldern





$$\tau \frac{\partial}{\partial t} z(\mathbf{v},t) = -z(\mathbf{v},t) + \int_X w(\mathbf{v},\mathbf{x},t)\, x(\mathbf{x},t)\, d\mathbf{x} - s(\mathbf{v},t) + \int_V u(\mathbf{v}-\mathbf{v}')\, y(\mathbf{v}',t)\, d\mathbf{v}'$$

Eingabe für Neuron **v** – *Schwelle* + *laterale Kopplungen*

Cohen-Grossberg 1983:
Aktivität stabil bei symm. Gewichten u_{ij}, monoton wachs. y(.) und $\tau>0$

Lokale Wechselwirkungen

Training: Hebb'sches Lernen

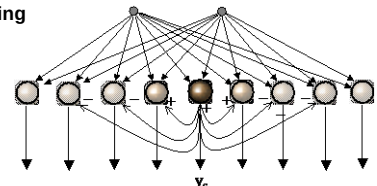
$$\tau_1 \frac{\partial}{\partial t} w(\mathbf{v},\mathbf{x},t) = -w(\mathbf{v},\mathbf{x},t) + \gamma_1\, y(\mathbf{v},t)\, x(\mathbf{x},t) \qquad \textit{Gewichte}$$

$$\tau_2 \frac{\partial}{\partial t} s(\mathbf{v},t) = -s(\mathbf{v},t) + \gamma_2\, y(\mathbf{v},t)\, x_0 \qquad \textit{Schwelle}$$

$\tau_1, \tau_2 >> \tau$

Lokale Wechselwirkungen

Training



$$\tau \frac{\partial}{\partial t} z(\mathbf{v}, t=t') \approx z_i(t) - z_i(t\text{-}1) = -z_i(t\text{-}1) + \sum_j w_{ij}(t)\, x_j(t) + \sum_j u_{ij} y_j(t)$$

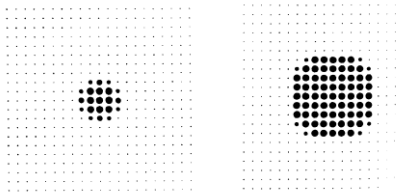
Abklingen der Veränderung, **diskrete** Strukturen

$y_i(t) = S(\sum_j w_{ij}(t)\, x_j(t) + \sum_j u_{ij} y_j(t))$

$w_{ij}(t) = w_{ij}(t\text{-}1) + \gamma[x_j y_i - B(y_i) w_{ij}]$ mit B(.) zum Vergessen

Lokale Wechselwirkungen

Simulation der Aktivität: Populationskodierung



hohe Schwelle

niedrige Schwelle

Abstraktion: Mittelpunkt einer „Blase“. Hier: B(y)=1

Somatotopische Karten

Selbstorganisierende Karten

Anwendung Sprachanalyse

Anwendung Robotersteuerung

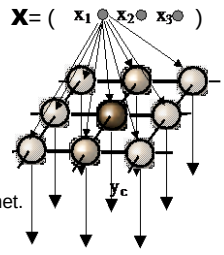
Kohonennetze: Topologie

Eingabedimension n = 3

Alle Neuronen erhalten die gesamte Eingabe.

Ausgabedimension d = 2

Die Ausgaben werden in einem Ausgaberaum angeordnet.



Def. „Nachbarschaft“ von Neuronen

Lernalgorithmus Kohonenkarten

1. **Suche Neuron (Gewichtsvektor) mit kleinstem Abstand zur Eingabe x** (größter Ausgabe)

$$|\mathbf{x} - \mathbf{w}_c| = \min_k |\mathbf{x} - \mathbf{w}_k| \qquad \textit{winner selection}$$

2. **Adaptiere die Gewichte**

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \gamma_{(t+1)} [\mathbf{x} - \mathbf{w}_k(t)]$$

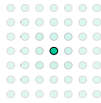
3. **Adaptiere auch die nächsten Nachbarn**

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \gamma_{(t+1)}\, h_{(t+1,c,k)} [\mathbf{x} - \mathbf{w}_k(t)]$$

z.B. mit $h(t+1,c,k) := \begin{cases} 1 & \text{wenn Neuron } k \text{ aus der Nachbarsch aft von } c \text{ ist} \\ 0 & \text{sonst} \end{cases}$ *Winner Take All*

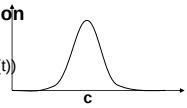
Nachbarschaftsfunktionen

- **Binäre Funktion:** zeitabhängige Einschränkung



- **Glockenfunktion**

h(k,c,t)= exp(k-c,σ(t))

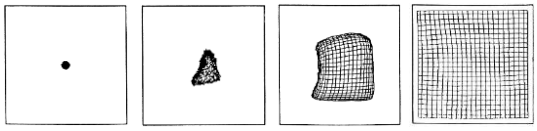


Topologie-Entwicklung

Eingabe: uniform-verteilte, 2-dim Zufallszahlen
Zeichnung: Verbindungsline vom Gewichtsvektorendpkt zum Nachbarn

Zeichnung :

t=0 t=20 t=100 t=10000

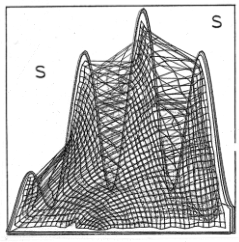


Simulation SOM

- **Simulation** einer uniformen, rechteckigen 2D-Eingabeverteilung, 2D-Ausgabeanordnung
  teacher demo/ reale Approximation
- **Simulation** einer ringförmigen 2D-Verteilung,
  Lupeneffekt

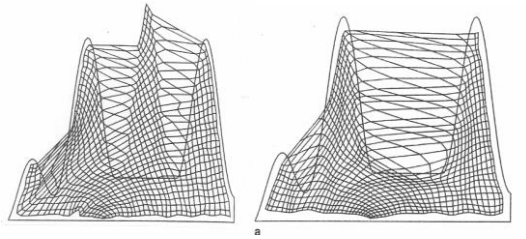
Topologie-Entwicklung SOM

Beispiel Finger-Sensorikentwicklung



Topologie-Entwicklung

Beispiel Finger-Amputation, Phantomschmerzen



Topologie-Entwicklung

Tendenz zur Selbstordnung

Iteration für zwei Gewichtsvektoren $\mathbf{w}_1$ und $\mathbf{w}_2$ bei gleicher Eingabe:

- Abstand verringert sich **(Rechnung)**
- Kein Nachbar überholt den Gewinner **(Rechnung)**

Pseudocode für Kohonenkarten

```
VAR x:   ARRAY [1..n] OF REAL; (* Muster*)
    w:   ARRAY[1..m,1..m] OF ARRAY [1..n] OF REAL; (* Gewichte*)
    vc:  RECORD i,j : INTEGER END; (* sel. Neuron *)
BEGIN
   σ := σmax ;                              (* initial: max. Nachbarschaft *)
  FOR t:=1TO tmax DO
    Read(PatternFile,x)                     (* Muster erzeugen oder einlesen *)
    (* Neuron selektieren *)
    Min:= ABS(x-w[1,1]);                    (* initialer Wert *)
    FOR i:=1 TO mDO                         (* In allen Spalten *)
      FOR j:=1 TO m DO                      (* und Zeilen *)
        Abstand:= ABS(x-w[i,j]);            (* suche Minimum *)
        IF Abstand < Min
          THEN Min:=Abstand; vc.i:=i; vc.j:=j;
        END
      END
    END
    (* Gewichte adaptieren *)
    FOR i:= vc.i-σ TO vc.i+σ DO             (* Evaluiere 2-dim. *)
     FOR j:= vc.j-σ TO vc.j+σ DO            (* Nachbarschaft um c*)
       IF i>0 AND i<= m AND j>0 AND j<=m THEN   (* Vorsicht am Rand *)
            w[i,j] = w[i,j] + 1.0/FLOAT(t) * (x -w[i,j])
       ENDIF
     END
    END (*i*)
    GrafikAnzeige(t,w)                      (* Visualisierung der Iteration *)
    upDate(σ,t)                             (* Verkleinerung des Nachbarschaftsradius σ *)
  END   (*t*)
```


Überwachte adaptive Vektorquantisierung

Klasse bekannt: Lerne die Klassengrenzen

$$\mathbf{w_k}(t+1) = \mathbf{w_k}(t) + \boldsymbol{\gamma}(t+1)\, h(t,c,k)[\mathbf{x} - \mathbf{w_k}(t)]$$

$$\text{mit } h(t,c,k) := \begin{cases} 1 & k = c,\ \text{Klasse}(\mathbf{w_c}) = \text{Klasse}(\mathbf{x}) \\ -1 & k = c,\ \text{Klasse}(\mathbf{w_c}) \neq \text{Klasse}(\mathbf{x}) \\ 0 & k \neq c \end{cases}$$

2. Nachbar c‘ ebenfalls

$$h(t,c,k) := \begin{cases} 1 & k = c, c' \text{ und } \text{Klasse}(\mathbf{w_k}) = \text{Klasse}(\mathbf{x}) \\ -1 & k = c, c' \text{ und } \text{Klasse}(\mathbf{w_k}) \neq \text{Klasse}(\mathbf{x}) \\ 0 & k \neq c, c' \end{cases}$$

Somatotopische Karten

Selbstorganisierende Karten

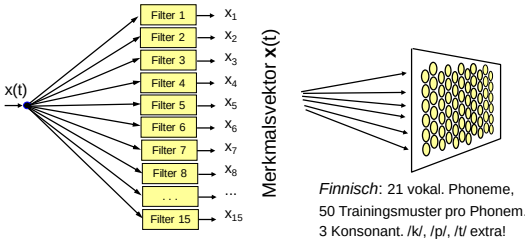
Anwendung Sprachanalyse

Anwendung Robotersteuerung

Spracherkennung

Beispiel: Neuronale Schreibmaschine

Fouriertranformation 200 Hz-5KHz

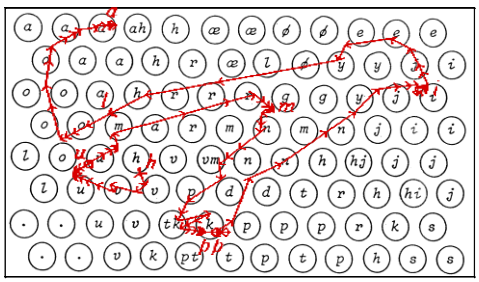


Finnisch: 21 vokal. Phoneme, 50 Trainingsmuster pro Phonem. 3 Konsonant. /k/, /p/, /t/ extra!

26 ms überlappende Intervalle, alle 10 ms

Spracherkennung

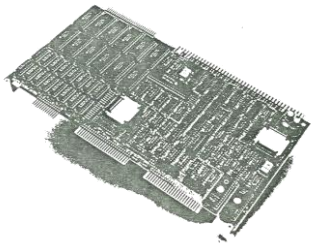
Erfolg: ähnliche Phoneme - ähnlicher Ort.
z.B. Dynamik des Wortes >>*humppila*<<



Spracherkennung

UNIVERSITÄT FRANKFURT AM MAIN

Implementierung: Hardware-Platine



Training: 10min/100Worte pro Sprecher

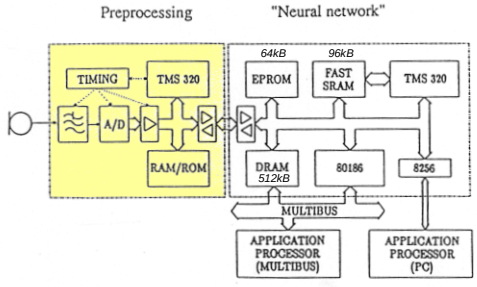
Erkennung: 250ms/Wort

Leistung
92%-97% richtig erkannt

Spracherkennung

UNIVERSITÄT FRANKFURT AM MAIN

Implementierung



UNIVERSITÄT FRANKFURT AM MAIN

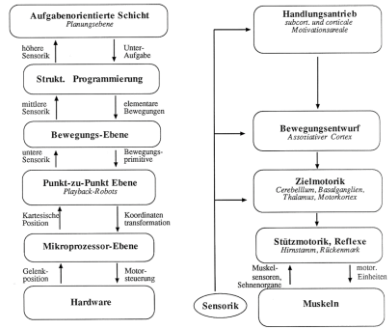
Somatotopische Karten

Selbstorganisierende Karten

Anwendung Sprachanalyse

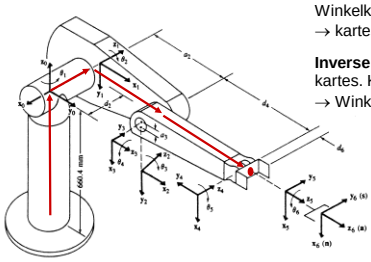
Anwendg Robotersteuerung

Robotersteuerung vs. Muskelkontrolle



Robotersteuerung

Beispiel Puma-Roboter



Probleme

Kinematik
Winkelkoord. Gelenke **θ**
→ kartes. Koord. **x**

Inverse Kinematik
kartes. Koord. **x**
→ Winkelkoord. **θ**

Robotersteuerung

- **Kinematik** *durch homogene Koord.transformationen*

Drehung

$$\begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$$

Skalierung

$$\begin{pmatrix} c_1 & 0 \\ 0 & c_2 \end{pmatrix}$$

Shift

$$\begin{pmatrix} 1 & 0 & s_1 \\ 0 & 1 & s_2 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{W} = \mathbf{W_{shift}} \cdot \mathbf{W_{rot}} \cdot \mathbf{W_{scal}} = \begin{pmatrix} c_1\cos\varphi & -c_2\sin\varphi & s_1 \\ c_1\sin\varphi & c_2\cos\varphi & s_2 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Inverse Kinematik** *Analytisch schwierig oder unmöglich!*

$$\theta_1 = \tan^{-1}(f_1(p_1,p_2,d_2)) := t_1(\mathbf{p}) \quad |\theta_1| \le \pi$$
$$\theta_2 = \tan^{-1}(f_2(p_1,p_2,p_3,a_2,a_3,d_2d_4)) := t_2(\mathbf{p}) \quad |\theta_2| \le \pi$$
$$\theta_3 = \tan^{-1}(f_3(p_1,p_2,p_3,a_2,a_3,d_2d_4)) := t_3(\mathbf{p}) \quad |\theta_3| \le \pi$$

Inverse Kinematik beim PUMA-Roboterarm

$$\theta_1 = \tan^{-1}\left(\frac{-\text{ARM}\, p_y\sqrt{p_x^2+p_y^2-d_2^2} - p_x d_2}{-\text{ARM}\, p_x\sqrt{p_x^2+p_y^2-d_2^2} + p_y d_2}\right) \qquad -\pi \le \theta_1 \le \pi$$

$$\theta_2 = \tan^{-1}\left[\frac{\sin\alpha\cos\beta + (\text{ARM}\cdot\text{ELBOW})\cos\alpha\sin\beta}{\cos\alpha\cos\beta - (\text{ARM}\cdot\text{ELBOW})\sin\alpha\sin\beta}\right] \qquad -\pi \le \theta_2 \le \pi$$

$$\cos\alpha = -\frac{\text{ARM}\cdot\sqrt{p_x^2+p_y^2-d_2^2}}{\sqrt{p_x^2+p_y^2+p_z^2-d_2^2}} \qquad \cos\beta = \frac{p_x^2+p_y^2+p_z^2+a_2^2-d_4^2-(d_4^2+a_3^2)}{2a_2\sqrt{p_x^2+p_y^2+p_z^2-d_2^2}}$$

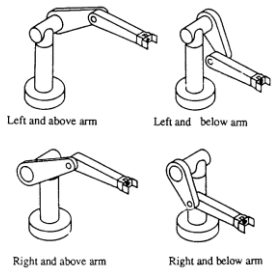
$$\sin\alpha = -\frac{p_z}{\sqrt{p_x^2+p_y^2+p_z^2-d_2^2}} \qquad \sin\beta = \sqrt{1-\cos^2\beta}$$

$$\theta_3 = \tan^{-1}\left(\frac{\sin\phi\cos\beta - \cos\phi\sin\beta}{\cos\phi\cos\beta + \sin\phi\sin\beta}\right) \qquad -\pi \le \theta_3 \le \pi$$

$$\sin\phi = \text{ARM}\cdot\text{ELBOW}\sqrt{1-\cos^2\phi} \qquad \cos\phi = \frac{a_2^2+(d_4^2+a_3^2)-R^2}{2a_2\sqrt{d_4^2+a_3^2}}$$

$$\sin\beta = \frac{d_4}{\sqrt{d_4^2+a_3^2}} \qquad \cos\beta = \frac{|a_3|}{\sqrt{d_4^2+a_3^2}} \qquad R = \sqrt{p_x^2+p_y^2+p_z^2-d_2^2}$$

Inverse Kinematik beim PUMA-Roboterarm



Arm configurations	ARM	ELBOW
LEFT and ABOVE arm	−1	+1
LEFT and BELOW arm	−1	−1
RIGHT and ABOVE arm	+1	+1
RIGHT and BELOW arm	+1	−1

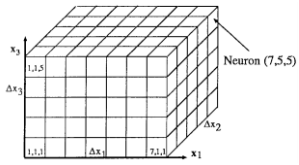
Besser:

Lernen der inversen Kinematik!

Probleme der Positionskontrolle

- Änderungen der Armgeometrie sind nicht möglich,
  sonst Neu-Analyse und Reprogrammierung der Transformation nötig
- Mehr Gelenke als Freiheitsgrade
  nur Teilbewegungen möglich
- Analytische Transformation Kart. Koord→Gelenk-Koord. langsam
  Echtzeitprobleme
- Keine bessere Auflösung an kritischen Stellen
- Keine automatische Bewegungsoptimierung (z.B. Energie)

Grobe Positionierung



Suche Neuron
mit *Winner-take-all*

$$| \mathbf{w}_k - \mathbf{x} | = \max_i | \mathbf{w}_i - \mathbf{x} |$$

Assoziiert ist $\Theta_k = (\theta_1, \theta_2, \theta_3)$

Lernen von $\mathbf{w}_k$ z.B. mit Kohonen-Netz,
oder fest initialisiert

Feine Positionierung

Aktivität

grobe Pos. *feine Pos.*

$$\mathbf{\Theta}(\mathbf{x}) = \mathbf{\Theta}_c + \Delta\mathbf{\Theta} = \mathbf{\Theta}_c + \mathbf{A}_c (\mathbf{x}-\mathbf{w}_c) = \mathbf{U}_c \Delta\mathbf{x}$$

mit $\mathbf{U}_c = (\mathbf{A}_c, \Theta_c)$, $\Delta\mathbf{x} = (\mathbf{x}-\mathbf{w}_c, 1)$

Lernen

$$\mathbf{U}_c(t+1) = \mathbf{U}_c(t) + h(.)\gamma(t+1)[\mathbf{U}_c^* - \mathbf{U}_c(t)]$$ *SOM Lernregel*

Widrow-Hoff Lernregel für Schätzung $\mathbf{U}^*$

Lehrervorgaben $\mathbf{x}_I$: beob. Position durch $\mathbf{w}_c$ bzw. Θ_c

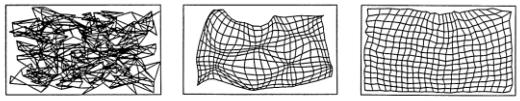
$\mathbf{x}_F$: beob. finale Position

$$\mathbf{\Theta}_c^* = \mathbf{\Theta}_c + \mathbf{A}_c(\mathbf{x}-\mathbf{x}_F)$$

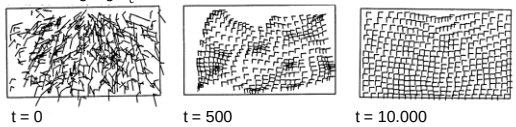
$$\mathbf{A}_c^* = \mathbf{A}_c + \mathbf{A}_c[\,(\mathbf{x}-\mathbf{w}_c) - (\mathbf{x}_F-\mathbf{x}_I)\,]\,(\mathbf{x}_F-\mathbf{x}_I)^T / |(\mathbf{x}_F-\mathbf{x}_I)|^2$$

Konvergenzverhalten

Grobbewegung $\mathbf{\Theta}_c$



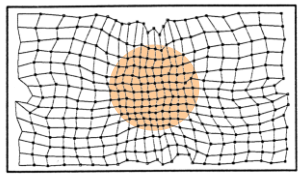
Feinbewegung $\mathbf{A}_c$



t = 0 t = 500 t = 10.000

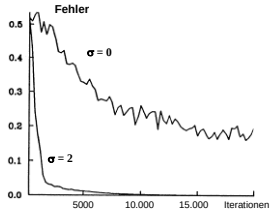
Vorteile

- **Schnelle Kontrolle**, da tabellierte Funktion
- SOMs steuern auch sehr **unkonventioneller Architekturen** (Vielgelenksarme, benutzer-definierte Geometrie)
- **Verbesserte Auflösung** (Neuronendichte) der topologie-erhaltenden Abbildung in oft benutzten Gebieten : bedarfsabhängige Auflösung!



Vorteile

- **Nebenbedingungen** sind einfacher zu integrieren (z.B. Energie)
- **Lerngeschwindigkeit** ist durch die Nachbarschaft deutlich beschleunigt (Fluch der Dimensionen gemildert!)



Nachteile

- Die erlernten Transformationstabellen sind *fest*
- *Zeitsequenzen* können so nicht erlernt werden (Trajektorien)
- Es gibt **keine** *Generalisierung* oder Abstraktion einer Bewegung; Skalierung der Bewegung ist **nicht** möglich!