Konkurrentes Lernen AS2-4

www.uni-frankfurt.de

Quantisierung & Klassen

Competive Learning

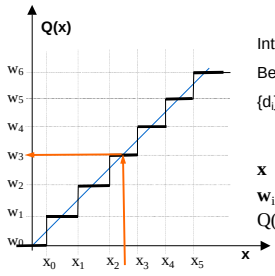
Selbstorganisierende Karten

Traveling Salesman-Problem

Neuronale Gase

Klassifikation und Quantisierung

- Quantisierung



Intervall $[x_i, x_{i+1}]$ → Prototyp w_i

Bezeichnung mit Codewort d_i z.B. 010

$\{d_i\}$ = Codebuch

$\mathbf{x} \in \Re^n$:

$\mathbf{w}_i$ = Codebuchvektor

$Q(\mathbf{x})$: Vektorquantisierung

Klassifikation und Quantisierung

- **p(x) = const**: Gleiche Intervallbreiten, *Uniforme Quantisierung*
- MSE = min ⇔ Prototyp in Intervallmitte

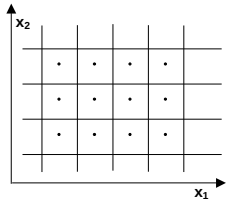
p(x) nicht const. *Nicht-uniforme Quantisierung*

- die Prototypen x_i sind die Schwerpunkte (*centroids*) der p(**x**) im Intervall der Klasse
- für die Intervalldichte (Klassendichte M(x)) gilt $M(x) \sim p(x)^{1/3}$
- die Klassengrenzen sind in der Mitte zwischen den Prototypen x_i
- die Varianzen aller Intervalle (Fehlerabweichungen) sind gleich

Klassifikation und Quantisierung

- Forderung $\langle(\mathbf{x}\text{-}\mathbf{w}_i)^2\rangle = \min$
- $\Rightarrow$ Klassen $\Omega_i = \{x \mid |x\text{-}w_i| \leq |x\text{-}w_j| \; \forall j \}$ *Voronoi-Teilung*

2-dim Fall: min.Fehler



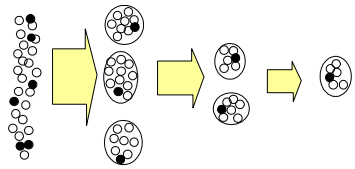
Lernparadigmen

- 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

Winner-take-all Grundmechanismus

- **Beschränkung** der Konkurrenz auf die Clustermitglieder
- **Zusammenfassung** der Ausgaben zu neuem Cluster
- **Erneute Anwendung** des Winner-take-all



Eingabe 1. Cluster -Ebene 2. Cluster -Ebene Klassifi-zierung

DEF Cluster = ?

z.B. ähnliches y

Winner-take-all: Grundmechanismus

- **Gegeben:** Neuronencluster
- **Auswahlregel:** Wähle das Neuron mit größter Aktivität

$$z_r = \max_i z_i \qquad \text{mit } z_i = \mathbf{w}^T\mathbf{x}$$

$$y_i = S(z_i) := \begin{cases} 1 & r = i \\ 0 & r \neq i \end{cases} \qquad \textit{winner-take-all}$$

$$y_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \qquad \textit{soft winner-take-all}$$

Winner-take-all Grundmechanismus

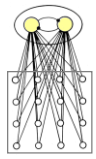
- **Forderung** $|\mathbf{x}-\mathbf{w}| \to 0$, d.h. $\Delta\mathbf{w} \sim (\mathbf{x}-\mathbf{w})$
- **Rechnung**
- **Lerngleichung** *winner-take-all*

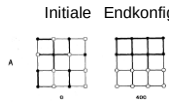
$$\mathbf{w}_r(t) = \mathbf{w}_r(t-1) + \gamma[\mathbf{x}/|\mathbf{x}|^2 - \mathbf{w}_r(t-1)] \qquad i = r$$
$$\mathbf{w}_i(t) = \mathbf{w}_i(t-1) \qquad \text{für } i \neq r\,.$$

für Probleme mit eindeutiger Lösung.

Competitive Learning

- **Beispiel:** Dipol-Korrelationen

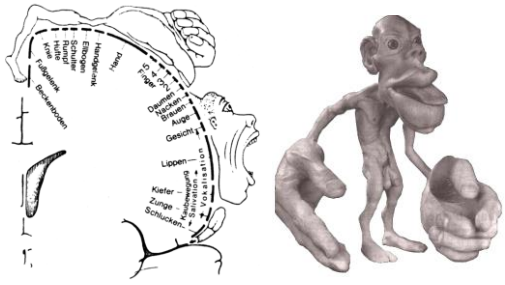




Erklärung: normierte Eingabeaktivität verwenden

Somatotopie

- Beobachtung: Abbilder, Karten

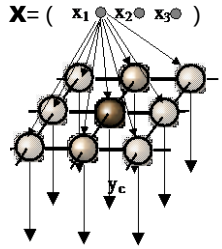


Kohonennetze: Topologie

Eingabedimension n = 3

Ausgabedimension d = 2

Def. „Nachbarschaft“ von Neuronen



Lernalgorithmus Kohonenkarten

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

$$|\mathbf{x} - \mathbf{w}_c| = \min_k |\mathbf{x} - \mathbf{w}_k|$$

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**

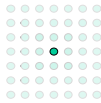
$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \gamma(t+1)\,h(t+1,c,k)\,[\mathbf{x} - \mathbf{w}_k(t)]$$

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

Winner Take All

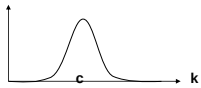
Nachbarschaftsfunktionen

- **Binäre Funktion:** zeitabhängig



- **Glockenfunktion**

h(k,c,t)= exp(k-c,σ(t))



- **Soft-winner-take-all (RBF-Neuronen)**

$$S_{RBF}(\mathbf{x}-\mathbf{w}_c) = \max_k S_{RBF}(\mathbf{x}-\mathbf{w}_k)$$

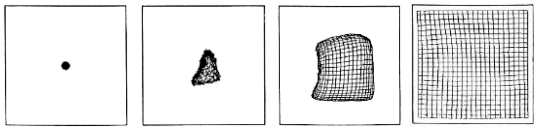
Topologie-Entwicklung

- **Eingabe:** uniform-verteilte, 2-dim Zufallszahlen

Zeichnung: Verbindungsline vom Gewichtsvektorendpkt zum Nachbarn

Zeichnung :

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



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)

Topologie-Entwicklung

- **Netzformation für maximale Information**
- *M* Klassen: maximale Information bei uniformer Verteilung

$$P(\omega_i) = P(\omega_j) \overset{!}{=} 1/M$$

Sehr viele Klassen (Kontinuum): *Raumelement ΔA*

Klassendichte $M(x) := \lim_{\Delta A \to 0} \frac{\text{Klassenzahl k}}{\Delta A}$ *magnification factor*

$$k = \frac{P(\mathbf{x} \in \Delta A)}{P(\omega)} = \frac{P(\mathbf{x} \in \Delta A)}{1/M}$$

$$M(\mathbf{x}) := \lim_{\Delta A \to 0} M \frac{P(\mathbf{x} \in \Delta A)}{\Delta A} = M\, p(\mathbf{x})$$

Aber: 1-dim: $M(\mathbf{x}) \sim p(\mathbf{x})^{2/3}$

Besser: $|\mathbf{x} - \mathbf{w}_c| P_c = \min_k |\mathbf{x} - \mathbf{w}_k| P_k$

Überwachte adaptive Vektorquantisierung

- **Klasse bekannt: Lerne die Klassengrenzen**

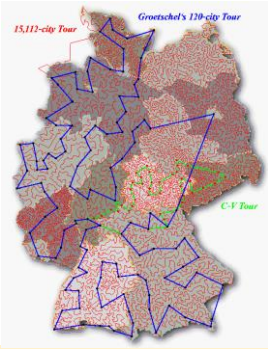
$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \boldsymbol{\gamma}(t+1)\, h(t,c,k)[\mathbf{x} - \mathbf{w}_k(t)]$$

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

2. Nachbar c' ebenfalls

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

Problem des Handlungsreisenden



Zahl der möglichen Reisen

$M = (N-1)(N-2)(N-3)... = (N-1)!$

Stirling:

$N! \approx (2\pi N)^{1/2} N^N e^{-N}$

Also

$M = (N-1)! \approx (2\pi)^{1/2} e^{(N-1/2)\ln N - N}$

Exponentielle Zunahme in N:

N=120 $\Rightarrow$ $M = 6 \cdot 10^{196}$ mögl. Reisen

Suche **suboptimale** Lösungen!

Problem des Handlungsreisenden

- **Lösung** mit *Kohonen map* *B.Angéniol et al., 1988*

  Rundreise = Kette von *m* Neuronen mit 1-dim Nachbarschaft

- **Algorithmus**

1. Start mit *m=1* Neuron, Tourenzähler *t=1*,Rundenzähler *r=1*
2. Gebe Koord. einer Trainingsstadt ein
3. Suche Neuron *i* dazu mit min. Abstand.
4. Hat das Neuron schon eine Stadt, DANN erzeuge Kopie des Neurons.
   SONST: adaptiere Gewichtsvektor mit
   $\mathbf{w}_i(r) = \mathbf{w}_i(r-1) + h(\sigma,d)\,(\mathbf{x}-\mathbf{w}_i)$ $\quad d := \inf(|i-i_c|, m-|i-i_c|)$
5. r = m **?** DANN r=1, $\sigma(t+1) = (1-\alpha)\sigma(t)$, t=t+1;
   SONST r=r+1
6. Wenn $t < t_{max}$ dann t=t+1, r=1, gehe zu 2.

Ergebnis: **mittl. Abstand** im Einzuggebiet pro Neuron **minimal**

Problem des Handlungsreisenden

- **Simulation 30 Städte** Entwicklung der Lösung

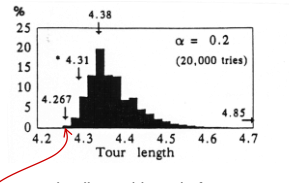
Zusätzlicher Löschmechanismus

Neuron in 3 Runden nicht ausgewählt: zu weit weg, löschen!

Problem des Handlungsreisenden

UNIVERSITÄT FRANKFURT AM MAIN

- Ergebnisgenauigkeit



schnelle Nachbarschafts-Verkleinerung, viele Versuche

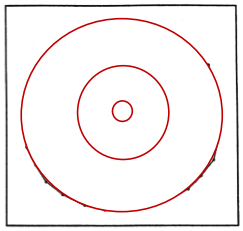
Optimum: 31 mal erreicht in 20.000 Versuchen

Problem des Handlungsreisenden

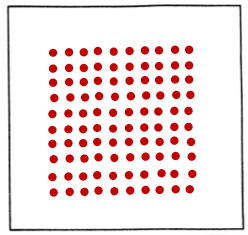
UNIVERSITÄT FRANKFURT AM MAIN

- Andere Punktverteilungen

Konzentrische Kreise



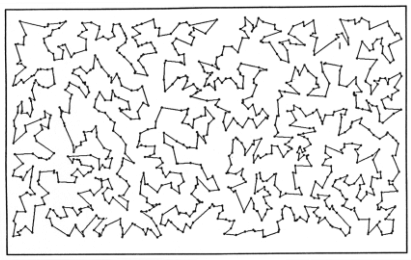
Gleichmäßige, symmetrische Verteilung



Problem des Handlungsreisenden

UNIVERSITÄT FRANKFURT AM MAIN

- Sehr viele Städte (1000) α=0.1 bei 12 Std ~ α=0.2, 20 min



Quantisierung & Klassen

Competive Learning

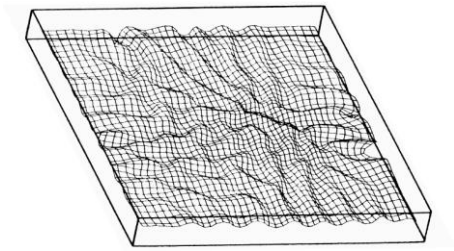
Selbstorganisierende Karten

Traveling Salesman-Problem

Neuronale Gase

Einbettung und intrinsische Dimension

- **Beispiel** 3-dim Punktmenge in 2-dim Ausgabedimension



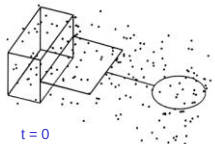
Neuronale Gase

- **Grundidee:** Nachbarschaftsverbindungen adaptiv formen
- **Graph des Netzes**:
  Ziel: Klassenprototypen sind Nachbarn $\Leftrightarrow$ Adjazenzmatrix $A_{ij}=1$
- **Selektion**
  Wähle Neuron *k* mit $y_k = \max_i y_i$ *winner take all*
- **Nachbarschaft**
  Wähle Nachbarn *r* mit $y_k y_r = \max_j y_k y_j$
- **Lernregel** Nachbarschaft
  $B_{kr}(t+1) = B_{kr}(t) + \gamma\, y_k y_r$
- **Lernregel** Neuron
  $\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \boldsymbol{\gamma}(t+1)\,[\mathbf{x} - \mathbf{w}_k(t)]$

Neuronale Gase

UNIVERSITÄT FRANKFURT AM MAIN

- **Anpassung an interne Dimensionalität**



t = 0

NN-Gas Varianten

UNIVERSITÄT FRANKFURT AM MAIN

- **Einbeziehung der Nachbarschaft**
  - statt feste Nachbarschaft h(.) wie bei SOM dynamische durch Aktivität y_k
  - Abstand von Nachbarn verkleinern

Elastikanteil

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \alpha y_k[\mathbf{x} - \mathbf{w}_k(t)] + \beta \sum_{i \in N_k} (\mathbf{w}_k - \mathbf{w}_i)$$

„Elastische Netz“:

$y_k(\sigma_x) = S_k(\mathbf{x}) / \sum_i S_i(\mathbf{x})$ *"soft" winner-take-all Auswahl*

und $S_i = \exp(-(\mathbf{x}-\mathbf{w}_i)^2/\sigma_y^2)$

Statt Elastikanteil Nachbarschaftsfunktion möglich

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \gamma(t+1) \left\langle \sum_i y_i(\sigma_x) h(i,k,\sigma_y) [\mathbf{x} - \mathbf{w}_k(t)] \right\rangle_x$$