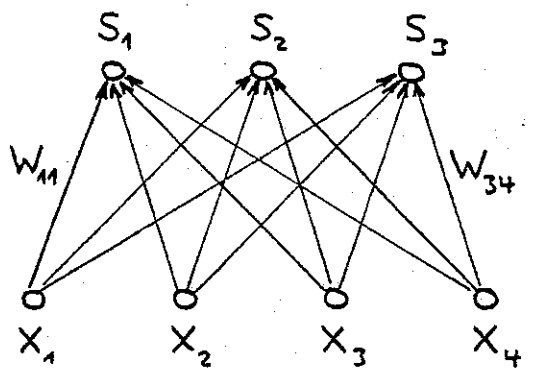
IV. KOMPETITIVES LERNEN

IV.1 "Winner-Take-All" Netzwerke



Output: S_i , $i=1,\dots,M$

Gewichte: W_{ij}

Input: X_j , $j=1,\dots,N$

- Die Inputs X_j können diskret oder kontinuierlich sein.
- Die Output-Neuronen sind sogenannte "Winner-Take-All" Neuronen ("Grandmother Cells"), d.h. dass immer nur eines der S_i den Wert 1 hat und alle übrigen den Wert 0.

- *Realisierungsmöglichkeiten:*

A) Berechne $d_i = \|\underline{X} - \underline{W}_i\| = \sum_j (X_j - W_{ij})^2$ $\quad i=1,\dots,M$

Bestimme $i^* = \operatorname{argmin}_i d_i$

[d.h. $d_{i^*} = \min_i d_i$]

$\longrightarrow$ $S_{i^*} = 1$, alle übrigen $S_i = 0$

B) Analog zu A), aber $d_i = \|\underline{X} - \underline{W}_i\|$ durch $\underline{X} \cdot \underline{W}_i = \sum_j X_j W_{ij}$ ersetzt.

→ B) ist äquivalent zu A), wenn die Gewichtsvektoren $\underline{W}_i$ normiert sind:

$$\|\underline{W}_i\| = \sum_j W_{ij}^2 = 1$$

C) Durch inhibitorische symmetrische Verbindungen $W_{ii'} = W_{i'i} < 0$ zwischen den Outputneuronen (und exzitatorischen Selbstverbindungen $W_{ii} > 0$).

→ Outputschicht wie ein Hopfield-Netzwerk relaxieren lassen (gegen Grenzzustand mit nur einem $S_i = 1$).

- Anwendungen:
  - Diskretisierung eines kontinuierlichen Inputraums
  - Clustering, Kategorisierung, Data Compression

  [→ Vorverarbeitung von Inputdaten!]

- Vorteile:
  - Vereinfachung eines nachfolgenden Lernprozesses.
  - Erhaltung von Nachbarschaftsrelationen (siehe "Feature Maps").

- Nachteile:
  - Für jede Kategorie wird ein separates Outputneuron benötigt.
  - Nicht robust gegenüber Degradation oder Ausfall von Komponenten.

IV.2 Einfaches kompetitives Lernen

→ LERNALGORITHMUS (UNSUPERVISED):

① Zufällige Startgewichte $W_{ij}(0)$ $\quad i = 1,\dots,M$, $j = 1,\dots,N$

② Wahl eines Inputbeispiels : $X_j(t)$, $j = 1,\dots,N$

③ Berechne $d_i = \sum_{j=1}^{N} [X_j(t) - W_{ij}(t)]^2$, $i = 1,\dots,M$

④ Bestimme $i^* = \operatorname{argmin}_i d_i$: $d_{i^*} = \min_i d_i$

⑤ Ändere Gewichte:

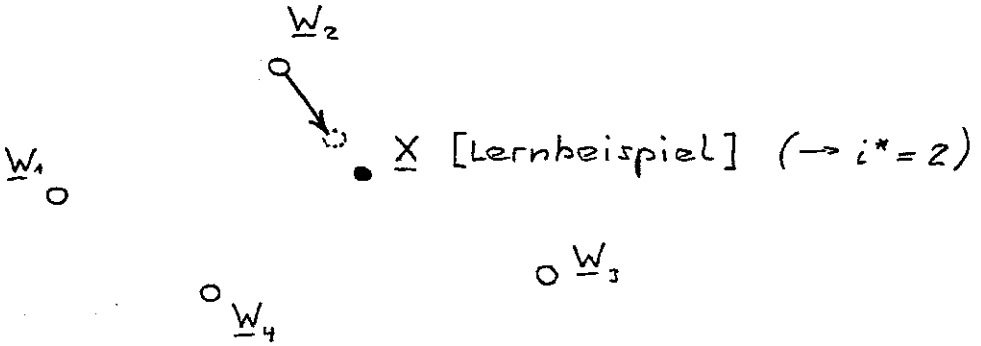
$$W_{i^*j}(t+1) = W_{i^*j}(t) + \eta(t)\left[X_j(t) - W_{i^*j}(t)\right]$$
$$W_{ij}(t+1) = W_{ij}(t) \quad , \quad i \neq i^* \qquad j = 1,\dots,N$$

⑥ $t \leftarrow t+1$, GO TO ②

Die Lernrate $\eta(t)$, $0 \leq \eta(t) < 1$, nimmt während des Lernprozesses ab:

z.B. $\eta(t) = \eta_0 e^{-\alpha t}$ oder $\eta(t) = \eta_0 (1-\alpha t)$

Graphisch:



IV.3 "Learning Vector Quantization" (LVQ)

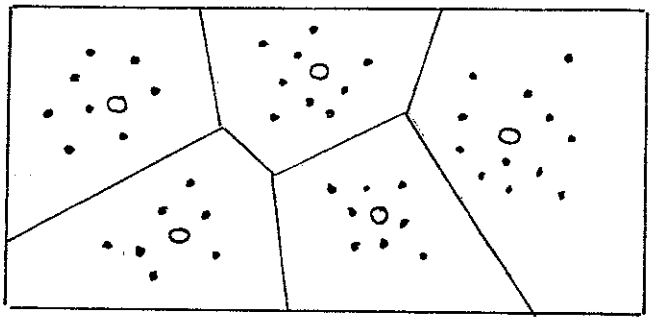
LVQ für Datenkompression ist wahrscheinlich die wichtigste Anwendung von kompetitiven Lernalgorithmen.

Idee: Kategorisierung eines vorgegebenen Satzes von Inputvektoren $\underline{X}^{\nu}$ in M Klassen, wobei diese durch Prototypvektoren $\underline{W}_i$ dargestellt werden:

Inputraum:

• : Inputvektoren $\underline{X}^{\nu}$

o : Prototypvektoren $\underline{W}_i$



⟶ Realisierung der $\underline{W}_i = (W_{i1}, W_{i2}, \dots, W_{iN})$ durch die Gewichtsvektoren von kompetitiven Neuronen (ev. mehrere Neuronen pro Klasse!):

A) Benutze einfachen Lernalgorithmus von IV.2
[+ nachträgliche Bestimmung der Klassen C(i)]

B) SUPERVISED LVQ:

[T. Kohonen, "Selforganization and Associative Memory", 3rd Edition, Springer 1989]

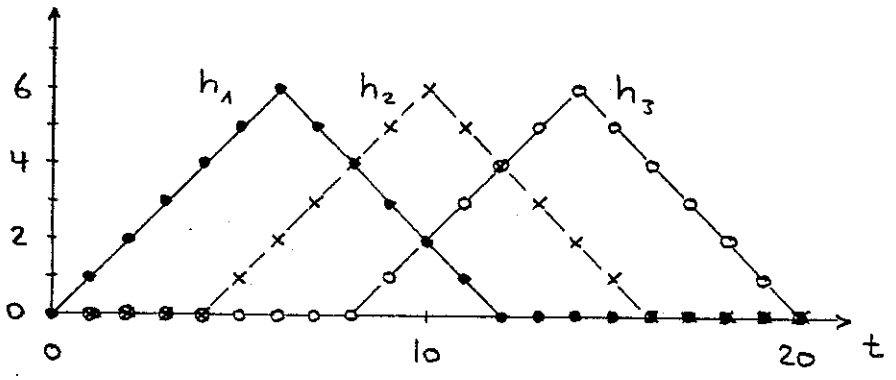
- Klasse $C(\underline{X})$ der Inputbeispiele bekannt
- $C(i)$ = Klasse des kompetitiven Neurons i

⑤
$$W_{i^*_j}(t+1) = W_{i^*_j}(t) + \eta(t)\left[X_j(t) - W_{i^*_j}(t)\right] \quad \text{if } C(i^*) = C(\underline{X}(t))$$

$$W_{i^*_j}(t+1) = W_{i^*_j}(t) - \eta(t)\left[X_j(t) - W_{i^*_j}(t)\right] \quad \text{if } C(i^*) \neq C(\underline{X}(t))$$

$$W_{ij}(t+1) = W_{ij}(t) \quad , \; i \neq i^*$$

Beispiel: [M. de Bollivier et al, IJCNN 1990]



3 Klassen: A) Superpositionen von h_1 und h_2

B) " " h_1 " h_3

C) " " h_2 " h_3

Problem 1: $x(t) = \alpha h_n(t) + (1-\alpha) h_m(t) + \varepsilon(t) \qquad 0 \le t \le 20$

α zufällig in $[0,1]$, $\varepsilon(t) \sim N(0,1)$

Problem 2:
$$x(t) = \begin{cases} \frac{1}{5}\left[\alpha h_n(t) + (1-\alpha) h_m(t)\right] + \varepsilon(t) & 0 \le t \le 20 \\ \varepsilon(t) & 21 \le t \le 40 \end{cases}$$

Trainingsset: 300 Beispiele

Testset: 5000 Beispiele

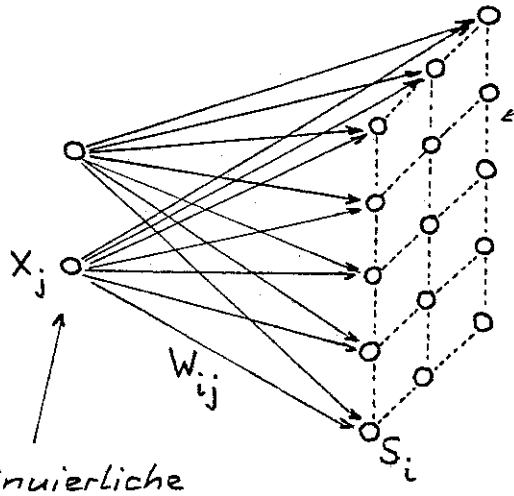
Resultate:

Methode	LVQ	Multilayer Perceptron
Test Set (Problem 1)	84.6 % korrekt	81.6 % korrekt
Test Set (Problem 2)	77.0 % korrekt	80.5 % korrekt

IV.4 Topologieerhaltende Abbildungen

(KOHONEN FEATURE MAPS)

→ Benachbarte Inputvektoren werden auch durch benachbarte Outputneuronen dargestellt
[ERHALTUNG VON NACHBARSCHAFTSRELATIONEN !]



1 oder 2-dim Gitter aus M kompetitiven Neuronen

$$d_{i^*} = \min_i d_i = \min_i \sum_{j=1}^{N} (X_j - W_{ij})^2$$

→ $S_{i^*} = 1$

übrige $S_i = 0$

N kontinuierliche Inputs (z.B. $0 \le X_j \le 1$)

→ LERNALGORITHMUS FÜR DIE W_{ij}:

① $W_{ij}(0) = 0.5 + \varepsilon \cdot RND$ $\quad i = 1,\dots,M$, $j = 1,\dots,N$ [RND zufällig $\in (-1,+1)$]

② $X_j = X_j(t)$, $j = 1,\dots,N$

③ $d_i = \sum_{j=1}^{N} [X_j(t) - W_{ij}(t)]^2$, $i = 1,\dots,M$

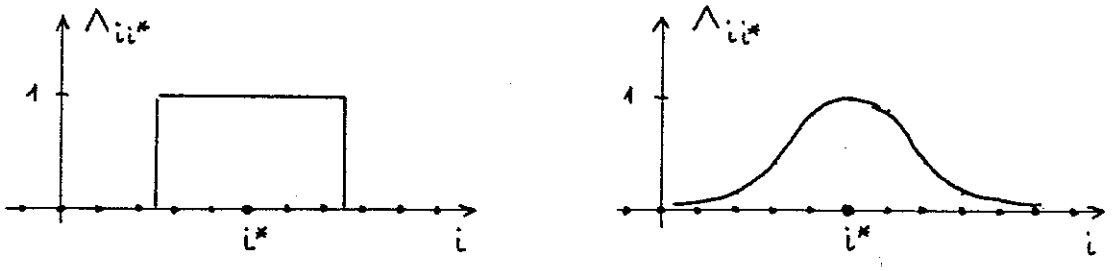
④ $i^* = \operatorname{argmin}_i d_i$: $d_{i^*} = \min_i d_i$

⑤ $W_{ij}(t+1) = W_{ij}(t) + \eta(t) \cdot \Lambda_{ii^*}(t) \, [X_j(t) - W_{ij}(t)]$ $\quad j = 1,\dots,N$

⑥ $t \leftarrow t+1$, GO TO ②

- Nachbarschaftsfunktion $\Lambda_{ii^*} = \begin{cases} 1 & \text{für } i = i^* \\ \text{abnehmend mit } |\underline{r}_i - \underline{r}_{i^*}| \end{cases}$

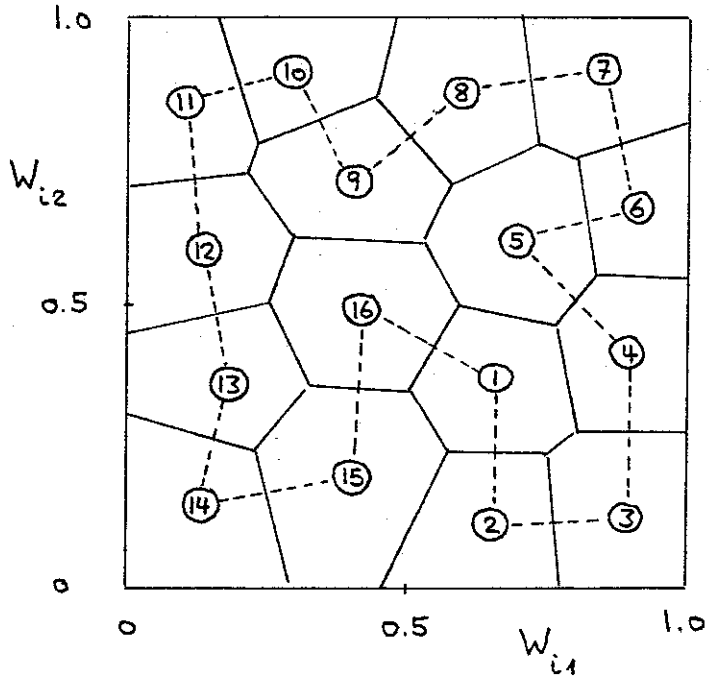
- *Beispiele für Nachbarschaftsfunktionen:*



- *Lernrate* $\eta(t)$, $0 \le \eta(t) < 1$, *und Reichweite von* $\Lambda_{ii^*}(t)$ *sind abnehmende Funktionen von* t.

BEISPIEL 1:

- 2-dim Inputraum: $X_1(t), X_2(t)$ zufällig in $(0,1)$
- 1-dim Ring mit 16 kompetitiven Neuronen, $i = 1, \ldots, 16$

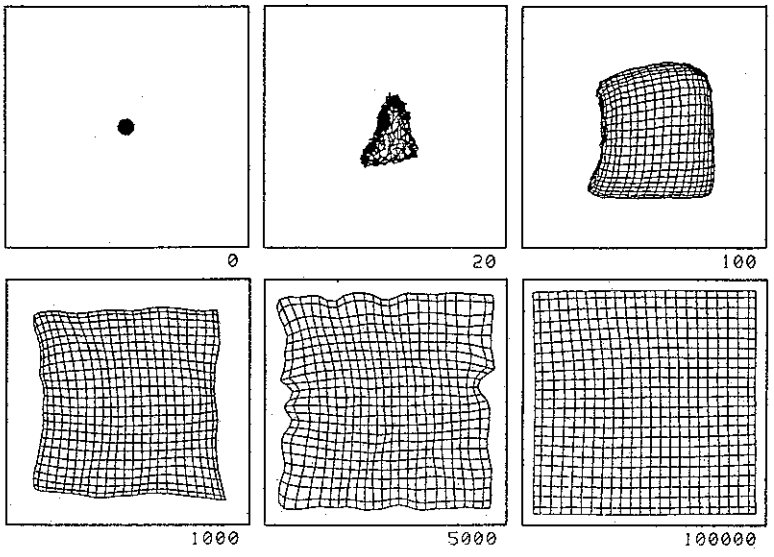


$t_{max} = 16'000$

BEISPIEL 2 : [Kohonen 1984]

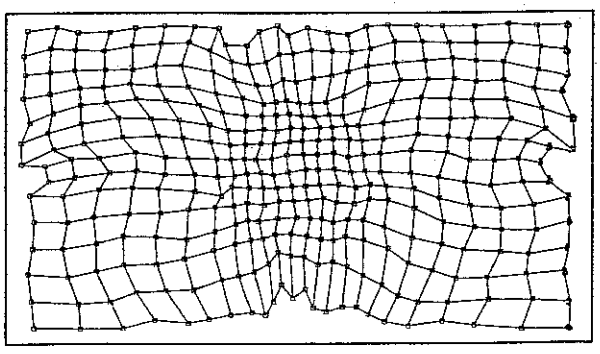
- 2-dim Inputraum : $X_1(t)$, $X_2(t)$ zufällig in $(0,1)$
- 2-dim Gitter mit 100 kompetitiven Neuronen

→ Zeitliche Entwicklung der Gewichte W_{i1} , W_{i2} :



BEISPIEL 3 : [Ritter et al , 1989]

- 2-dim Inputraum : $X_1(t)$, $X_2(t)$ zufällig , aber mit grösserer Wahrsch.keit in der Mitte des rechteckigen Gebietes
- 2-dim Gitter mit 15 x 24 kompetitiven Neuronen



→ Grössere Konzentration in der Mitte
(→ feinere Auflösung!)

Anwendungen von Kohonen Feature Maps

- **Spracherkennung:**

  → T. Kohonen:

  "The Neural Phonetic Typewriter",
  IEEE Computer 21, 11 (March 1988)

- **Klassifizierung von Zuständen eines elektrischen Verteilnetzes:**

  → D. Niebur and A.J. Germond:

  "Unsupervised Neural Net Classification of Power System Static Security States",
  Journal of Electrical Power and Energy Systems, Vol. 14, No. 2/3, pp. 233-242 (1992).

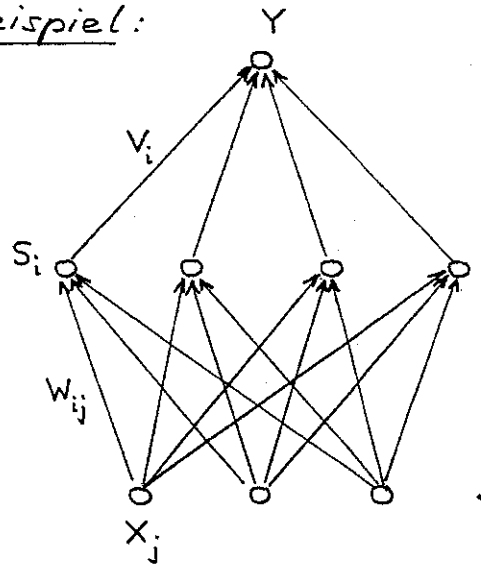
- **Lastvorhersage in elektrischen Verteilnetzen:**

  → A.J. Germond, T. Macabrey, and T. Baumann:

  "Application of Artificial Neural Networks to Load Forecasting",
  Proceedings Summer Workshop on Neural Network Computing for the Electric Power Industry, Stanford, pp. 24-35 (1992).

IV.5 Hybride Systeme

Beispiel:



← Output: $Y = V_{i^*}$

← Winner-take-all Neuronen:

$$S_i = \begin{cases} 1 & i = i^* \\ 0 & i \neq i^* \end{cases}$$

← Inputs

→ Feature Map (Vorverarbeitung) mit anschliessendem Feedforward-Netzwerk!

→ *Lernverfahren:*

$$W_{ij}(t+1) = W_{ij}(t) + \eta(t)\, \Lambda_{ii^*}(t) \left[X_j(t) - W_{ij}(t) \right]$$

$$V_i(t+1) = V_i(t) + \eta'(t)\, \Lambda'_{ii^*}(t) \left[Y^D - V_i(t) \right]$$

(Y^D = gewünschter Output)

ANWENDUNGEN:

- *ROBOTIK, REGELUNG*

  → H.J. Ritter, T.M. Martinetz, and K.J. Schulten:
  "Topology-Conserving Maps for Learning Visuo-Motor-Coordination", Neural Networks 2, 159 (1989).

- *VORHERSAGE*

  → T.M. Martinetz et al.:
  "Neural Gas Network for Time-Series Prediction"
  IEEE Trans. on Neural Networks, Vol. 4, No. 4, 558-569 (1993).