

4.4. Konstruktive Algorithmen

bisher: Anzahl der Schichten und Neuronen vorgegeben

→ Topologie vorgegeben u. Gewichte optimiert

⇒ Nachteile: • ungeeignete Topologie
• Verbesserung nur über „trial and error“

jetzt:

• Topologie wird während des Algorithmus verändert:

minimales Netz (nur Input u. Output)



Gewichte optimieren (pocket o.ä.)



graduell Neuronen u./o. Schichten hinzufügen
(versch. Methoden)



Gewichte optimieren



kein genug gelernt?

↓ via
evolve



- Vorteile: • optimalere Topologie
- möglicherweise schnellere Konvergenz (weniger Gewichts Anpassung bei einzelnen Schritten)

→ aber: endgültige Topologie nicht notwendig optimal

a) Tilling Algorithmus

- Netz mit Schwellwert-Funktionen $g(h) = \text{sgn}(h)$
- diskreter Input u. Output

hier:

$$f: \{-1, 1\}^n \rightarrow \{-1, 1\} \quad (\text{Boolsche Fkt.})$$

soll dargestellt werden

- $N_p \leq 2^n$ Anzahl Input-Muster
- N_p^L verschiedene Muster (interne Darstellungen) ^{"Klassen"}
in der Schicht L ; repr. mind. ein Input-Muster

$$N_p^0 = N_p$$

$$N_p^L \leq N_p$$

- alle N_p^L müssen wahrheitsgetreue Darstellungen (faithful representations) der input-Muster sein, d.h. zwei Muster mit versch. output müssen versch. in jeder Schicht dargestellt werden

Prinzip des Algorithmus:

jede Schicht ^{ist} besteht aus:

- einem Master-Neuron, wird trainiert (pocket-Alg.)
mögl. viel zu lernen $\Rightarrow N_e^L$ nicht gelernt, $\rightarrow 2$ Klassen
- weiteren Hilfs-Neuronen (ancillars), wird auf Unter-
menge der nicht korrekten intern. Mustern (Klassen)
trainiert \Rightarrow jeweils eine weitere Klasse, bis alle
Klassen wahrheitsgetreu
- nächste Schicht analog
- Ende: Master-Neuron der letzten Schicht hat alles
gelernt ($N_e^L = 0$) \rightarrow keine weiteren
ancillars nötig; Master-Neuron = output-Neuron

XOR mit Tilling-Algorithmus

XOR

A	B	C
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1

$N_{inp} = 2$

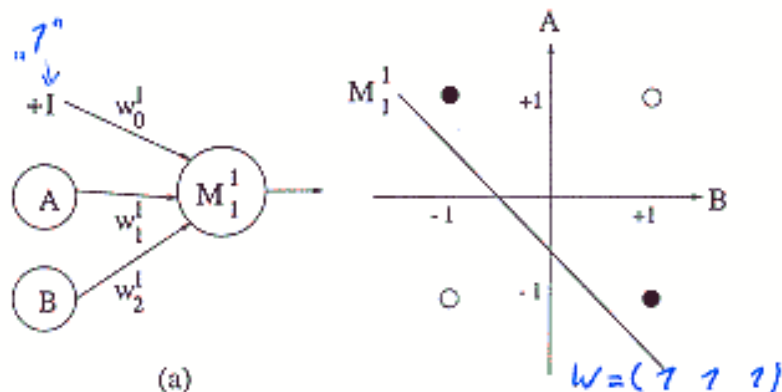
$N_p = 4$

$X_1 = (+1, -1, -1)$

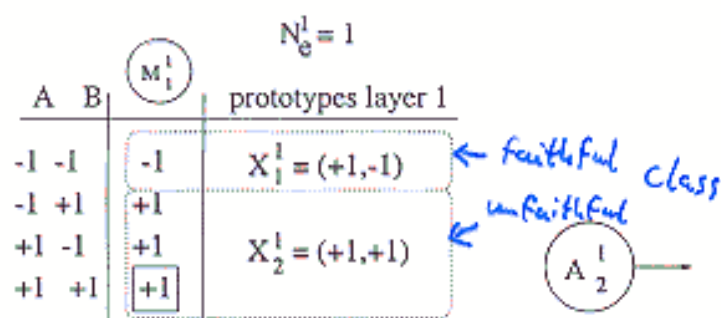
$X_2 = (+1, -1, +1)$

$X_3 = (+1, +1, -1)$

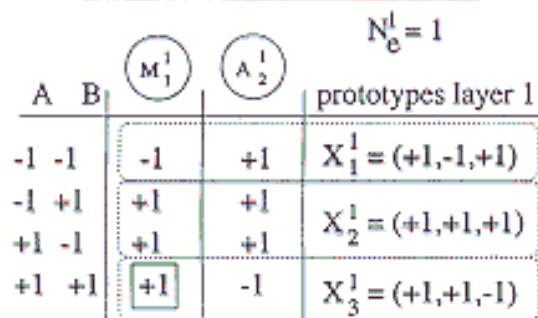
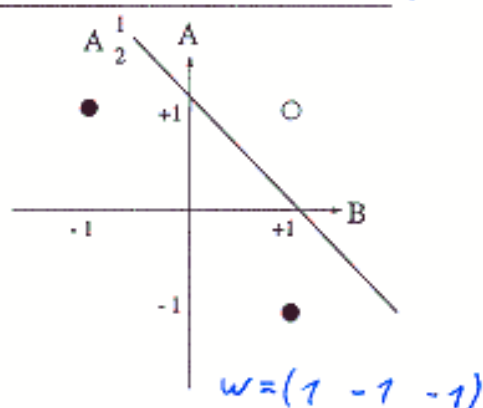
$X_4 = (+1, +1, +1)$



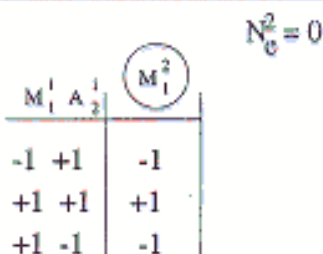
(a)



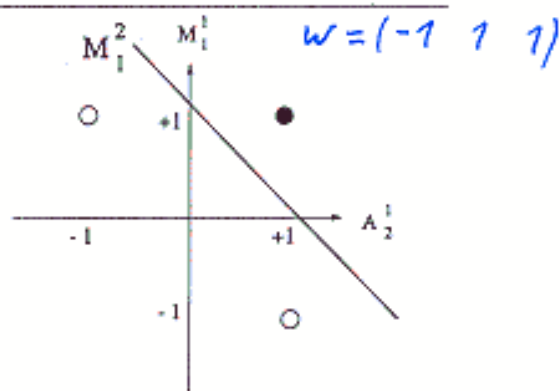
(b)



(c)



(d)

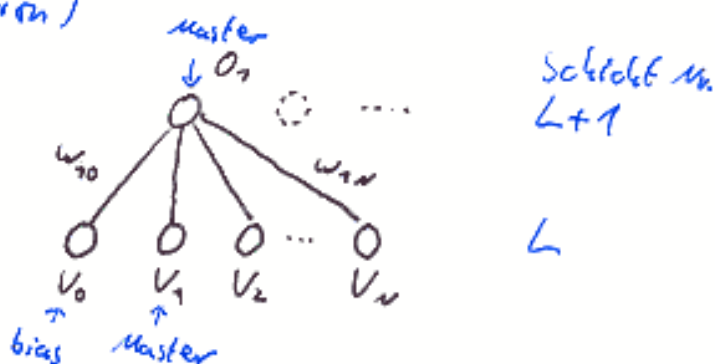


Konvergiert der Algorithmus immer?

ja, denn: Das Master-Neuron der Schicht $L+1$ klassifiziert wenigstens ein Muster mehr korrekt als das Master-Neuron der Schicht L ; also:

$$N_e^{L+1} \geq N_e^L + 1$$

Beweis: (durch Konstruktion)



- V_1 klassif. q Muster kor. ($V_1^M = Y^M$) $q < N_p$
- Muster v wird nicht kor. klassif. ($V_1^M = -Y^M$)

sei $w_{aj} = \begin{cases} 1 & \text{wenn } j=1 \\ \epsilon Y^v V_j^v & \text{sonst} \end{cases}$

mit $\frac{1}{N} < \epsilon < \frac{1}{N-2}$

$\Rightarrow v$ wird kor. klassif.

$$O_1^v = \text{sgn}\left(\sum_j w_{aj} V_j^v\right) = \text{sgn}(-Y^v + \epsilon Y^v N) = Y^v$$

(da $N\epsilon > 1$)

aber auch die Master q werden kor. klassif.:

$$O_1^M = \text{sgn}\left(\sum_j w_{aj} V_j^M\right) = \text{sgn}\left(Y^M + \underbrace{\epsilon Y^v \sum_{j \neq 1} V_j^M V_j^v}_{\text{Korrekturen}}$$

Korrekturen Vorzeichen von Y^M nicht ändern, da

$$|X| = \left|\sum_{j \neq 1} V_j^M V_j^v\right| \leq N-2 \quad \text{und} \quad (N-2)\epsilon < 1$$

- da $V_0^M = V_0^v = 1$ (bias)
- $V_j^M \neq V_j^v$ (Durst. ist faithful)

q.e.d.

Leistungsfähigkeit des Algorithmus

Zufällige boolesche Funktionen

$\langle L \rangle$ - mittlere Anzahl der Schichten (hidden)

$\langle N_a \rangle$ - mittl. Anz. von Neuronen in Schicht a

Anzahl input bits

Table 2. Learning random Boolean functions with $N_0 = 4, 6$ and 8 . For each N_0 we give the average number of layers (for 100 random functions) and the average number of hidden units in each layer. For each layer the average is over the trials for which the total number of layers is at least equal to L . The percentage of these trials is given in parentheses.

	$N_0 = 4$	$N_0 = 6$	$N_0 = 8$
$\langle L \rangle$	2.08 ± 0.05	3.75 ± 0.07	7.13 ± 0.08
$\langle N_1 \rangle$	2.68 ± 0.08 (100%)	8.55 ± 0.09 (100%)	15.57 ± 0.10 (100%)
$\langle N_2 \rangle$	1.15 ± 0.04 (95%)	4.68 ± 0.12 (100%)	11.55 ± 0.10 (100%)
$\langle N_3 \rangle$	1.0 (13%)	2.09 ± 0.10 (99%)	10.80 ± 0.12 (100%)
$\langle N_4 \rangle$		1.20 ± 0.06 (64%)	8.50 ± 0.14 (100%)
$\langle N_5 \rangle$		1.0 (12%)	5.84 ± 0.16 (100%)
$\langle N_6 \rangle$			3.01 ± 0.17 (100%)
$\langle N_7 \rangle$			1.73 ± 0.11 (75%)
$\langle N_8 \rangle$			1.06 ± 0.06 (36%)
$\langle N_9 \rangle$			2.0 (1%)
$\langle N_{10} \rangle$			1.0 (1%)

Generalisierungsfähigkeit - Anzahl von Domain-Grenzen

+++ --- ~ ++ -
Grenzen

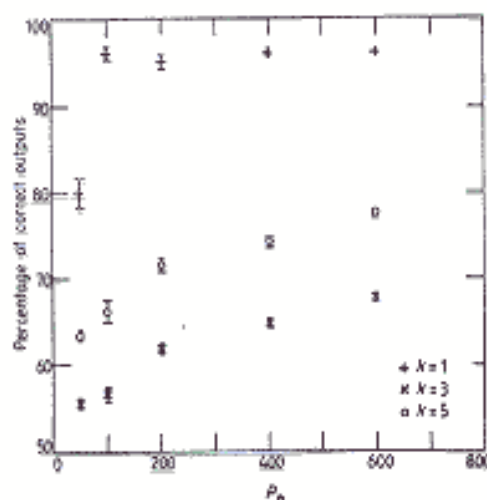


Figure 1. Learning to discriminate Ising chains by the number of domain walls (greater or smaller than three). Shown is the percentage of patterns of $N_0 = 25$ units with the correct output in the test set, as a function of the size p_0 of the training set (equal to the size of the test set). The average number of domain walls in the presented patterns is k . For each k and p_0 , data are averaged typically over 25 samples (250 samples for $p_0 = 50$). For $p_0 = 600$, the average number of layers of the network was 3.5, 3.0, 8.5 and 5.5 for $k = 1, 3$ and 5 .