VII. LERNEN MIT EINEM KRITIKER

VII.1 Lehrer oder Kritiker

- *Lehrer (Experte):*

  → Liefert für einen Satz von Inputs $\underline{I}^\mu$ (Lernbeispiele!) den zugehörigen gewünschten Output $\underline{D}^\mu = \underline{D}(\underline{I}^\mu)$

  → Lernen durch Optimierung einer Fehlerfunktion F:

  z.B. $$F = \sum_\mu \sum_i (D_i^\mu - O_i^\mu)^2 = \min$$

  wobei $\underline{O}^\mu = \underline{O}(\underline{I}^\mu, \underline{W})$ = Netzwerk-Output

- *Kritiker:*

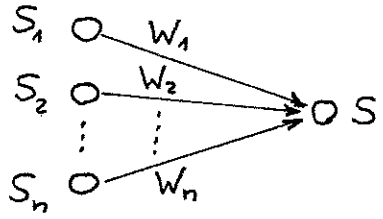
  → Liefert für jeden Input $\underline{I}^\mu$ eine Bewertung G^μ des Netzwerk-Outputs:

  $$G^\mu = G(\underline{O}^\mu) \quad \text{oder} \quad G(\underline{O}^\mu, \underline{I}^\mu)$$

  → Lernen durch Optimierung einer Bewertungsfunktion $\bar{G}$:

  z.B. $$\bar{G} = \sum_\mu G^\mu = \min \text{ oder } \max$$

VII.2 Deterministische oder stochastische Neuronen



$$S = f(V)$$

$$V = \sum_{i=1}^{n} W_i S_i$$

- Deterministische Neuronen:

$$S = \begin{cases} 1 & \text{falls } V \geq 0 \\ 0 & \text{" } V < 0 \end{cases} \quad \text{oder} \quad S = \frac{1}{1+e^{-V}}$$

bzw.

$$S = \begin{cases} 1 & \text{falls } V \geq 0 \\ -1 & \text{" } V < 0 \end{cases} \quad \text{oder} \quad S = \frac{1-e^{-V}}{1+e^{-V}}$$

- Stochastische Neuronen:

$$S = \begin{cases} 1 & \text{mit Wahrsch.keit } p(V) \\ 0 & \text{" " } 1-p(V) \end{cases}$$

bzw.

$$S = \begin{cases} 1 & \text{mit Wahrsch.keit } p(V) \\ -1 & \text{" " } 1-p(V) \end{cases}$$

wobei $$\boxed{p(V) = \frac{1}{1+e^{-\beta V}}}$$

[Standard Stoch. Neuron: $\beta = 1$]

[$\beta \to \infty$: → Übergang zu deterministischem Neuron:

$$S = \begin{cases} 1 \\ 0 \end{cases} \text{ bzw. } \begin{cases} 1 & \text{falls } V > 0 \\ -1 & \text{" } V < 0 \end{cases}$$]

VII.3 Optimierung einer Bewertungsfunktion

(deterministische Neuronen)

Neuronale Netzwerke mit deterministischen Neuronen können sowohl mit Hilfe eines Lehrers (→ Fehlerfunktion) als auch mit Hilfe eines Kritikers (→ Bewertungsfunktion) trainiert werden.

Wie beim Trainieren mit einem Lehrer (siehe "Error-Backpropagation", Kapitel VI) kann man zur Minimierung (oder Maximierung) der Bewertung ebenfalls ein Gradientenverfahren verwenden, vorausgesetzt, dass die Bewertung $G(\underline{O}, \underline{I})$ des Kritikers differenzierbar ist:

Lehrer: Fehlerfunktion $F = \frac{1}{2}(D-O)^2 = \min$

$$\rightarrow \Delta w_{ij} = -\eta \frac{\partial F}{\partial w_{ij}} = -\eta \frac{\partial F}{\partial O} \frac{\partial O}{\partial w_{ij}}$$

$$= \eta (D-O) \frac{\partial O}{\partial w_{ij}}$$

Kritiker: Bewertungsfunktion $G = G(O, I) = \min$

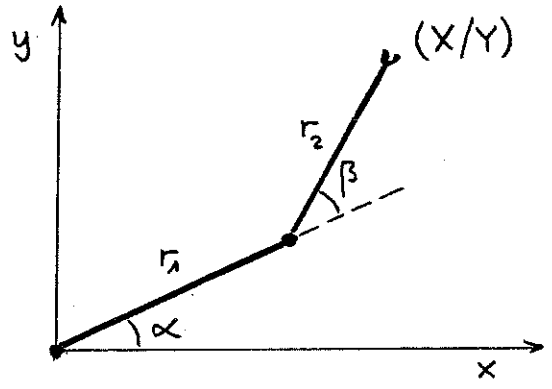
$$\rightarrow \Delta w_{ij} = -\eta \frac{\partial G}{\partial w_{ij}} = -\eta \frac{\partial G}{\partial O} \frac{\partial O}{\partial w_{ij}}$$

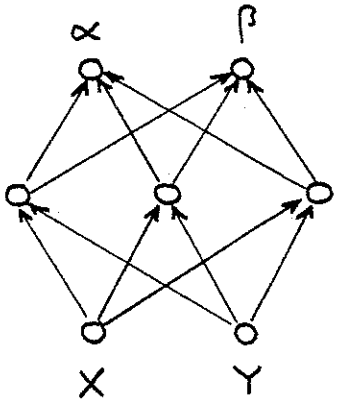
(siehe Beispiel "Roboter-Arm")

Bei komplizierten Bewertungsfunktion verwendet man besser ein stochastisches Lernverfahren (z.B. Simulated Annealing).

(siehe Beispiel "Mondlandung")

Beispiel: ROBOTER-ARM





- Lehrer: ⟶ Berechnung von $\alpha_D(X,Y)$, $\beta_D(X,Y)$ ["kompliziert"!]

⟶ Fehler:

$$F = (\alpha_D - \alpha)^2 + (\beta_D - \beta)^2 = \min$$

⟶ $$\Delta W_{ij} = \eta \left[(\alpha_D - \alpha) \frac{\partial \alpha}{\partial W_{ij}} + (\beta_D - \beta) \frac{\partial \beta}{\partial W_{ij}} \right]$$

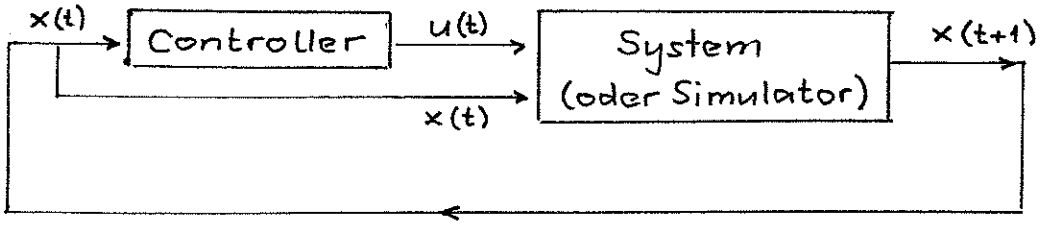
- Kritiker: ⟶ $\hat{X} = r_1 \cos\alpha + r_2 \cos(\alpha+\beta)$
$\hat{Y} = r_1 \sin\alpha + r_2 \sin(\alpha+\beta)$ ["einfach"!]

⟶ Bewertung:

$$G = (X-\hat{X})^2 + (Y-\hat{Y})^2 = \min$$

⟶ $$\Delta W_{ij} = \eta \left[(X-\hat{X}) \frac{\partial \hat{X}}{\partial \alpha} \frac{\partial \alpha}{\partial W_{ij}} + (Y-\hat{Y}) \frac{\partial \hat{Y}}{\partial \alpha} \frac{\partial \alpha}{\partial W_{ij}} + (X-\hat{X}) \frac{\partial \hat{X}}{\partial \beta} \frac{\partial \beta}{\partial W_{ij}} + (Y-\hat{Y}) \frac{\partial \hat{Y}}{\partial \beta} \frac{\partial \beta}{\partial W_{ij}} \right]$$

VII.4 Neurocontrol



Neurocontrol:

- Controller = Neuronales Netzwerk : $u(t) = u(x(t), \underline{W})$
  anpassbare Gewichte
- lokale Bewertung der Control-Entscheidung:
  $g(x(t), u(t))$

Problem der verzögerten Bewertung:

Bei vielen Control-Problemen kann die Effizienz des Controllers erst nach vielen Control-Aktionen bewertet werden
(oft erst am Ende eines ganzen Control-Zyklus)

→ Bewertungsfunktion:

$$G(t) = \sum_{\tau=t}^{T} \gamma^{\tau-t} g(x(\tau), u(\tau)) = min$$

(Abdiskontierte) Summe über zukünftige lokale Bewertungen!
$(0 < \gamma \leq 1)$

oder

$$G(T) = H(x(T)) = min$$

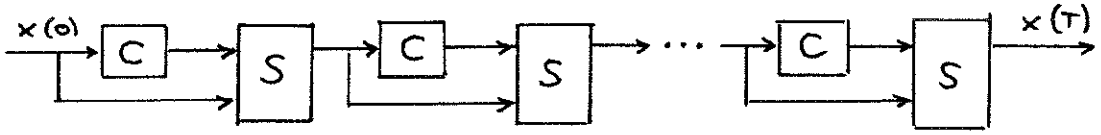
Bewertung des Endzustandes!

Beispiele:

- "Truck-Backer-Upper"
- Mondlandeproblem
- Spiele (z.B. Backgammon, etc)

Lernverfahren für Neurocontroller:

(i) "Backpropagation through Time"



C = Neurocontroller

S = Simulator (= zweites neuronales Netzwerk)

↑ feste Gewichte (im Voraus trainiert!)

- Ganzer Control-Zyklus $t = 0, \ldots, T$ mit gleichen Gewichten für C
- Error-Backpropagation durch die ganze Kette von neuronalen Netzwerken (nur Gewichte von C ändern)

[Beispiel: "Truck-Backer-Upper"]

(ii) Verwendung von stochastischen Lernverfahren (statt Backpropagation)

→ S muss nicht ein neuronales Netzwerk sein!

[Beispiel: Mondlandeproblem]

(iii) Verwendung eines zusätzlich neuronalen Netzwerks (="Kritiker"), das während des Control-Prozesses lernt, die zukünftige Bewertung G(t) immer genauer vorherzusagen

→ "TEMPORAL DIFFERENCE LEARNING" (TD-LEARNING)

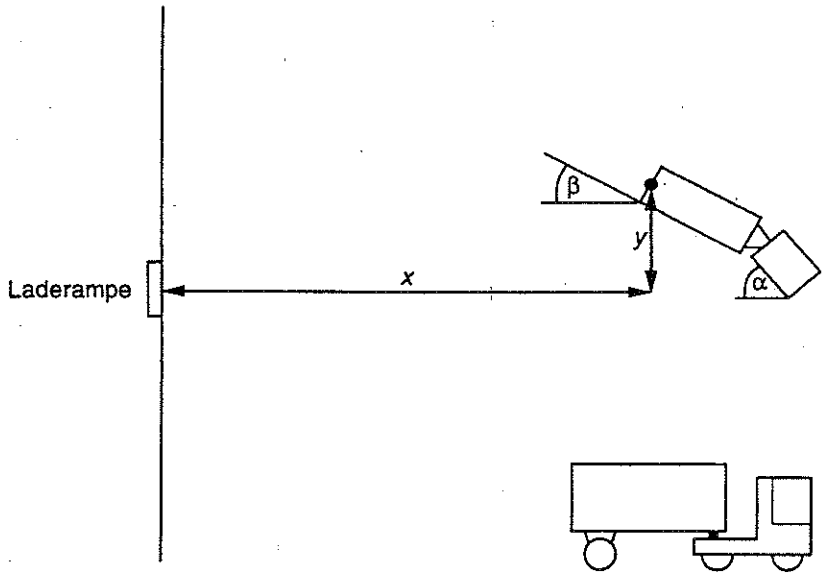
[siehe z.B.: R.S. Sutton and A.G. Barto
Reinforcement Learning
MIT Press, 1998
Chapters 6, 7, and 11]

Beispiel: DER "TRUCK-BACKER-UPPER"

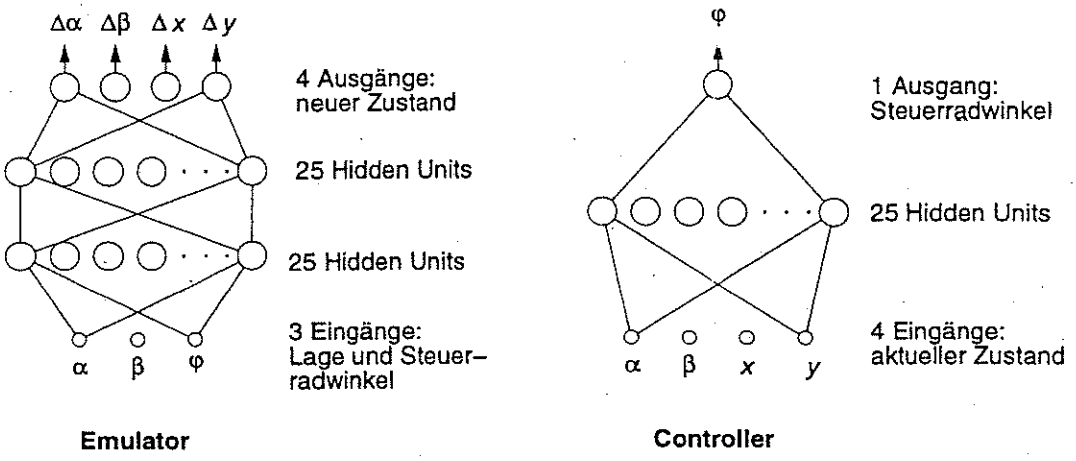
[D.H. Nguyen and B. Widrow
Neural Networks for Self-Learning Control Systems
IEEE Control Systems Magazine, April 1990, pp. 18-23]

→ Rückwärtsparkieren eines Lastwagens mit Anhänger an vorgegebene Stelle einer Laderampe

- Situationsplan des rückwärtsparkierenden Lastwagens und Definition der Positions-Parameter:



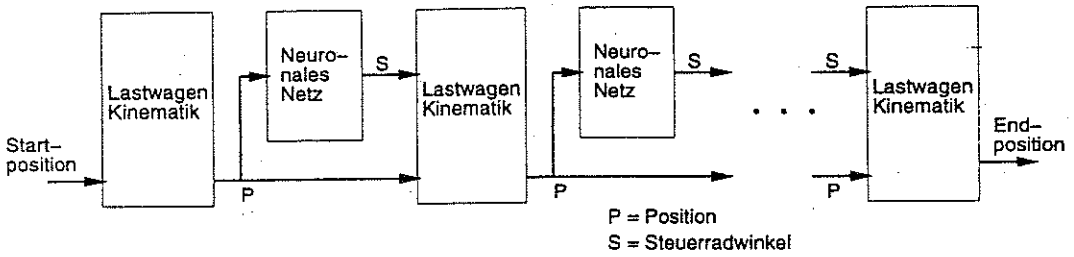
- Verwendete neuronale Netwerke für Lastwagen-Simulator (= Emulator) und Controller:



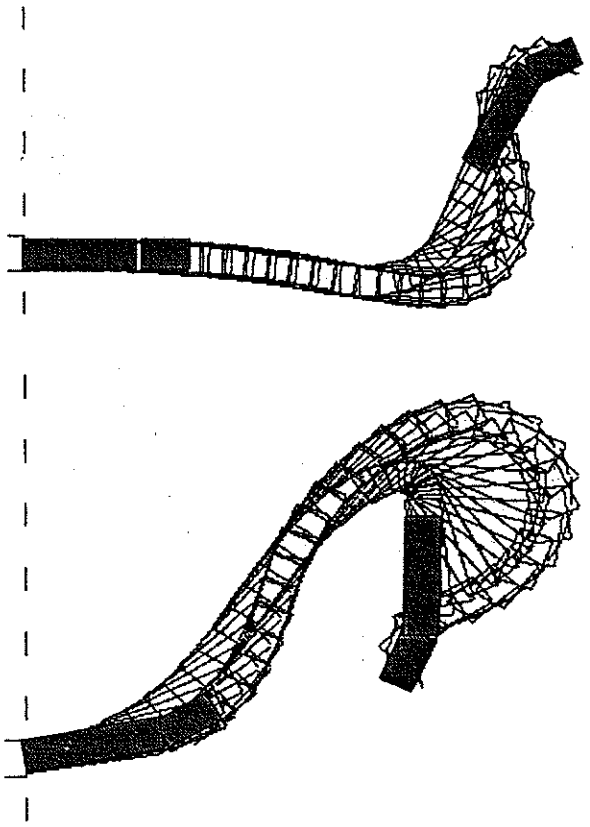
- Informationsfluss bei einer Rückwärtsfahrt des Lastwagens

  (= Vorwärtspfad durch die neuronalen Netzwerke)

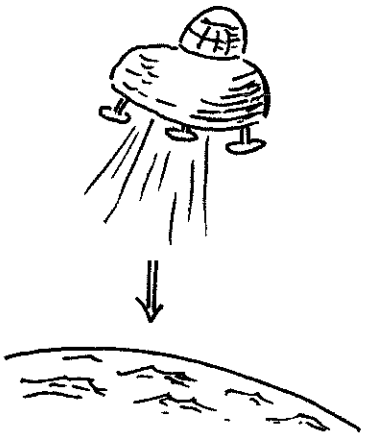
  [Lernprozess: Error-Backpropagation durch die ganze Kette von neuronalen Netzwerken]



- Beispiele von Fahrtrajektorien mit dem trainierten Neurocontroller:



Beispiel: DAS MONDLANDEPROBLEM



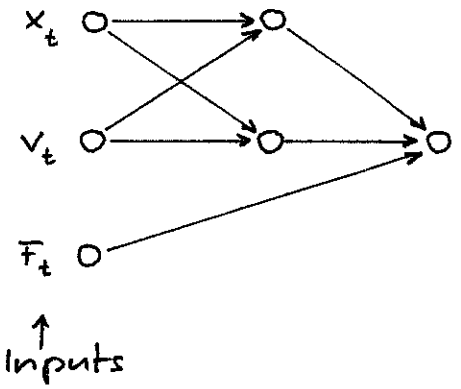
x_t : Höhe über Mondoberfläche

v_t : Sinkgeschwindigkeit

F_t : Treibstoffvorrat

f_t : Verzögerungskraft/Masseneinheit
(= Treibstoffverbrauch/sec)

- **Neurocontroller:**



$$f_t = f\left(x_t, v_t, F_t ; \{W_{ij}\}\right)$$

- **Bewertung:**

$$G = \langle v_c^2 + \lambda \cdot \Delta F \rangle = \min$$

wobei: v_c = Aufprallgeschwindigkeit

ΔF = totaler Treibstoffverbrauch

$\langle \ldots \rangle$ = Mittelung über verschiedene Anfangspositionen

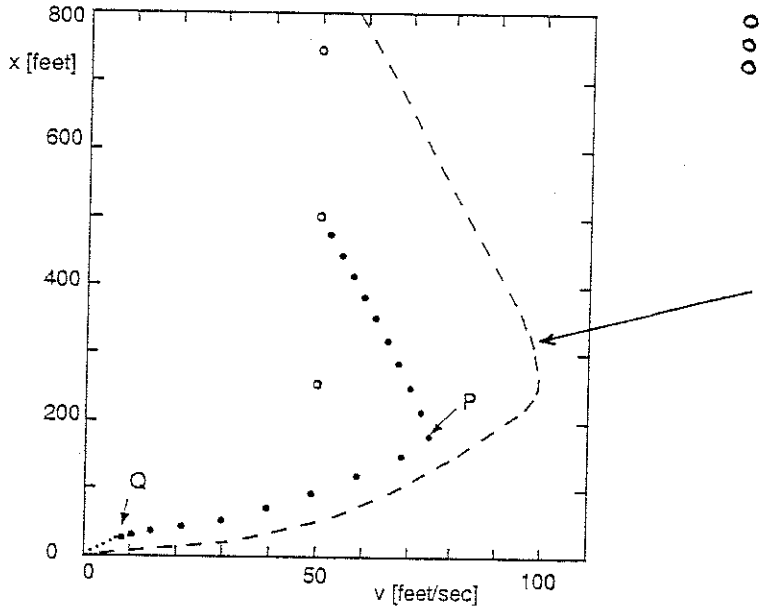
[→ Bewertung des Neurocontrollers erst am Ende eines Landevorgangs!]

- Training:

  - Landesimulation mit festgehaltenen W_{ij}
  - Stochastisches Lernverfahren (Simulated Annealing) zur Bestimmung von $\{W_{ij}\}$ mit $G = G(\{W_{ij}\}) = \min$

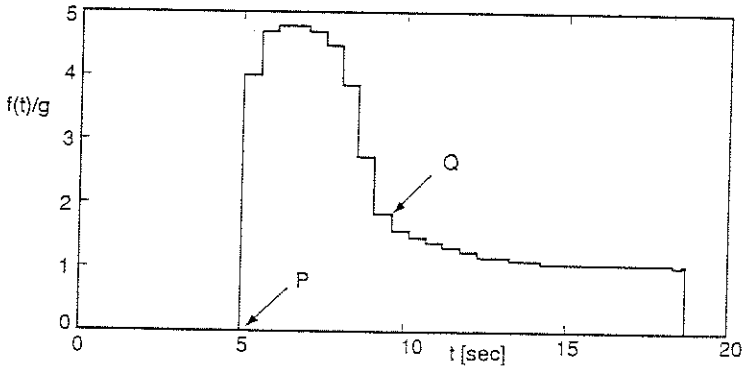
- Resultate:

  → Nahezu optimale Regelung nach ca. 500 Landesimulationen:



o o o : Anfangspositionen für Trainingsphase

Stabilitätsbereich ($v_c < 5$ feet/sec) für $F_0 = 150$



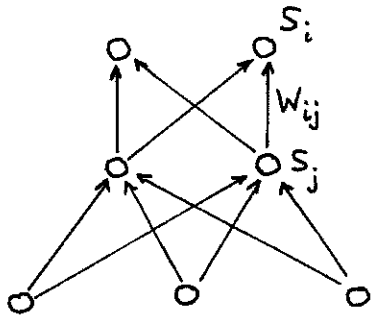
VII.5 Reinforcement Learning

[R.S. Sutton and A.G. Barto
Reinforcement Learning – An Introduction
MIT Press, 1998]

→ VARIANTE FÜR STOCHASTISCHE NEURONALE NETZWERKE:
A_{R-P} - LERNVERFAHREN (Associative Reward-Penalty)

[A.G. Barto, P. Anadan, IEEE Trans. Sys., Man, Cybern. 15, 360 (1985)]
[A.G. Barto, AIP Conf. Proc. 151 (1986), pp. 41-46]

· Stochastische Feedforward-Netzwerke:



$$V_i = \sum_j W_{ij} S_j$$

$$p(V_i) = \frac{1}{1 + e^{-V_i}}$$

[$\beta = 1$!]

$$S_i = \begin{cases} 1 & \text{mit Wahrsch.keit } p(V_i) \\ 0 & \text{- - } 1 - p(V_i) \end{cases}$$

Bewertung: $r = r(\text{Output}, \text{Input})$

z.B. $r = \begin{cases} \text{reward} & (\text{z.B. } r=1) \\ \text{penalty} & (\text{z.B. } r=0) \end{cases}$

oder $r \in [0,1]$ $(0 \le r \le 1)$

[skalare Bewertung]

- **A_{R-P} – Lernverfahren:**

  - Input vorgeben
  - Output bestimmen (stochastische Grösse!)
  - Bewertung r [alle Output- und Hidden Units erhalten dieselbe Bewertung!]
  - Gewichte W_{ij} ändern:

a) $$\Delta W_{ij} = \begin{cases} \eta \left[S_i - p(V_i) \right] S_j & \text{falls } r = \text{reward} \\ -\lambda\eta \left[S_i - (1 - p(V_i)) \right] S_j & \text{falls } r = \text{penalty} \end{cases}$$

$$[\, 0 \le \lambda \le 1 \,]$$

b) $$\Delta W_{ij} = \eta \left\{ r \left[S_i - p(V_i) \right] - \lambda (1-r) \left[S_i - (1 - p(V_i)) \right] \right\} S_j$$

falls $0 \le r \le 1$

[a) folgt aus b) wenn $r(\text{reward}) = 1$, $r(\text{penalty}) = 0$]

- **Erwartungswerte für $\lambda = 0$:**

$$E\{r\} = \text{prob}(S_i = 1) \cdot E\{r \mid S_i = 1\} + \text{prob}(S_i = 0) \cdot E\{r \mid S_i = 0\}$$

$$= p(V_i) \cdot E\{r \mid S_i = 1\} + (1 - p(V_i)) \cdot E\{r \mid S_i = 0\}$$

$$\rightarrow \quad \frac{\partial E\{r\}}{\partial W_{ij}} = p(V_i)\,(1 - p(V_i))\, S_j \left[E\{r \mid S_i = 1\} - E\{r \mid S_i = 0\} \right]$$

$$E\{\Delta w_{ij}\} = \eta\, s_j \left[p(v_i)\,(1-p(v_i))\, E\{r \mid s_i = 1\} + \right.$$
$$\left. + (1-p(v_i))\,(-p(v_i))\, E\{r \mid s_i = 0\} \right]$$

$$\longrightarrow \quad \boxed{E\{\Delta w_{ij}\} = \eta \frac{\partial E\{r\}}{\partial w_{ij}}} \qquad (\lambda = 0)$$

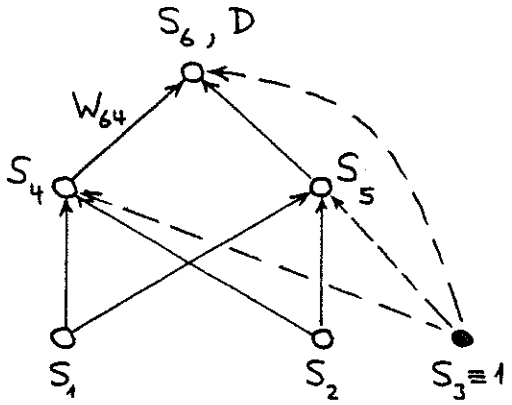
d.h. für $\lambda = 0$ und $p(v_i) = \frac{1}{1+e^{-v_i}}$ $[\beta = 1 \;!!]$ ist das A_{R-P} - Verfahren ein stochastisches Gradientenverfahren für $E\{r\} = \max$.

- Ein kleines $\lambda > 0$ [z.B. $\lambda = 0.01$] verhindert das Steckenbleiben in schlechten lokalen Optima.
- A_{R-P} ist im allgemeinen langsamer als Error-Backpropagation.

ABER:

- A_{R-P} ist einfacher als Backpropagation!
- A_{R-P} ist auch dann anwendbar, wenn nur eine grobe Bewertung möglich ist (z.B. gut/schlecht)!
- A_{R-P} ist auch anwendbar, wenn die Bewertung r eine Zufallsgrösse ist!

BEISPIEL A : DAS XOR-PROBLEM



	Input S_1	Input S_2	Output D
(a)	1	1	0
(b)	1	0	1
(c)	0	1	1
(d)	0	0	0

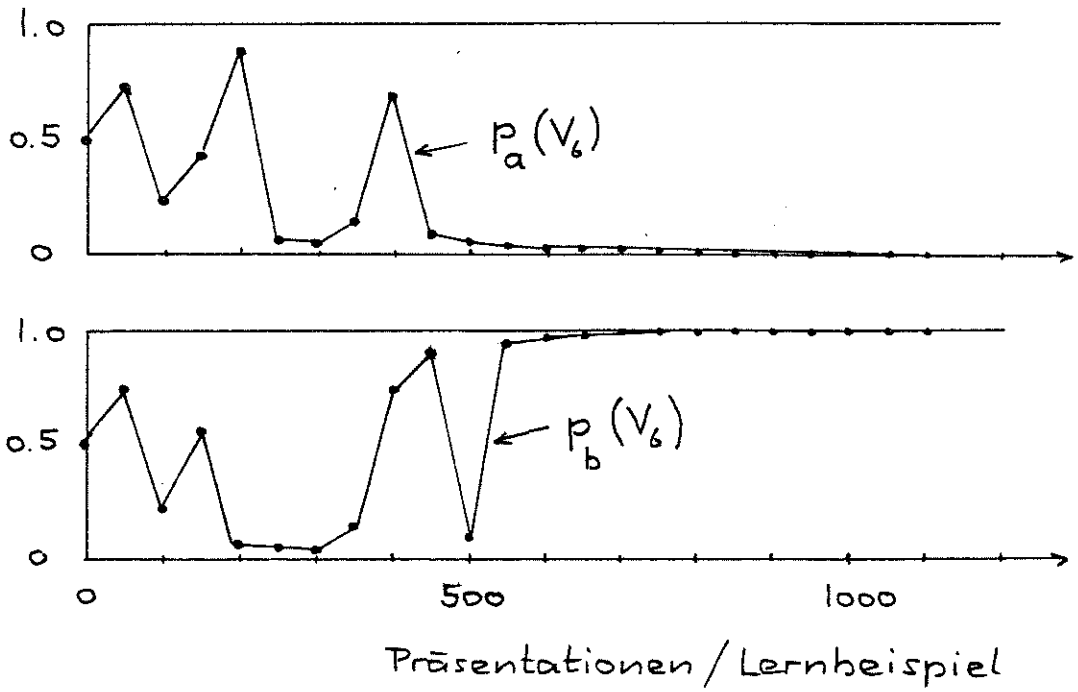
$$S_i = \begin{cases} 1 & \text{mit Wahrsch. } p(V_i) \\ 0 & \text{" " } 1-p(V_i) \end{cases} \qquad V_i = \sum_j W_{ij} S_j \qquad i = 4,5,6$$

Bewertung:

$$r = \begin{cases} 1 & \text{falls } S_6 = D \\ 0 & \text{" } S_6 \neq D \end{cases}$$

$$\rightarrow \Delta W_{ij} = \eta \{ r [S_i - p(V_i)] - \lambda (1-r) [S_i - (1-p(V_i))] \} S_j$$

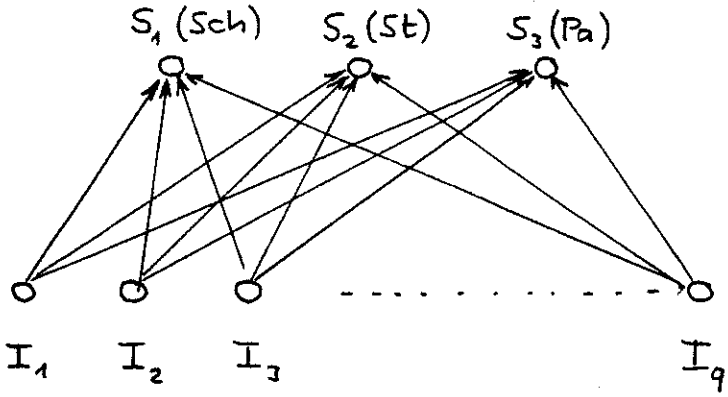
$$(\eta = 0.5 \,, \quad \lambda = 0.01)$$



BEISPIEL B : "SCHERE, STEIN, PAPIER"

"Schere (Sch)" gewinnt gegen "Papier (Pa)"
"Stein (St)" " " "Schere (Sch)"
"Papier (Pa)" " " "Stein (St)"

Adaptives stochastisches Neuronales Netzwerk spielt gegen einen menschlichen Gegner:



$$P_i = \frac{1}{1+\exp\left[-\sum_{j=1}^{q} W_{ij} I_j\right]}$$

$I_j = 0$ oder 1

$I_1 = 1$ iff letzte 2 Symbole des Gegners = (Sch, Sch)
$I_2 = 1$ " " " " " " = (Sch, St)
etc

Netzwerk wählt nächstes Symbol gemäss:

$$(S_1, S_2, S_3) = \begin{cases} (1,0,0) & \text{mit Wahrsch.keit } Q_1 \quad [\text{Sch}] \\ (0,1,0) & \text{" " } Q_2 \quad [\text{St}] \\ (0,0,1) & \text{" " } Q_3 \quad [\text{Pa}] \end{cases}$$

wobei: $Q_i = P_i / (P_1 + P_2 + P_3)$

Für die Änderung der Gewichte W_{ij} wird eine heuristische Erweiterung des A_{R-P}-Verfahrens verwendet:

$$\Delta W_{ij} = \eta \left\{ r \left[S_i - P_i \right] - \lambda (1-r) \left[S_i - (1-P_i) \right] \right\} I_j$$

wobei: $$r = \begin{cases} 1 & \text{bei Gewinn des Zuges} \\ 0 & \text{sonst} \end{cases}$$

Das Netzwerk adaptiert also seine Taktik ständig an die Strategie des Gegners.

Resultate für einfache (Sch, St, Pa, Sch, St, Pa, ...)-Strategie des Gegners: