

19.04.2002

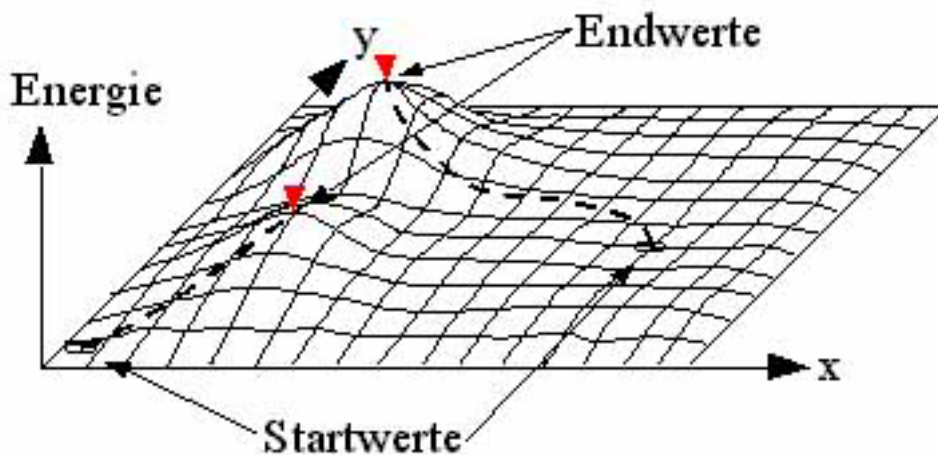
**Optimieren:** z.B.  $x^2 - 2x + 1 \rightarrow f' = 2x - 2 \rightarrow x = 1$   
Dies ist ein 1-dimensionales Problem

Bei n-dimensionalen Problemen kann man das Gradientenverfahren anwenden.

**Vorgehen:**  $x_{neu} \leftarrow x_{alt} \pm \eta \nabla f(x_{alt})$   
+ für max  
- für min

Es gibt Funktionen bei denen das Gradientenverfahren nicht funktioniert!

**Diese Energielandschaft ist von zwei Variablen, x und y, abhängig:**

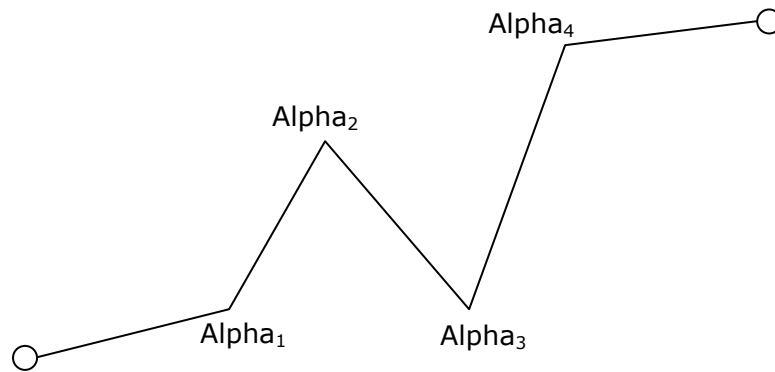


**In Abhängigkeit vom Startwert findet das Gradientenverfahren unterschiedliche Lösungen (in diesem Fall).**

## Evolutionäre Algorithmen

Versuch von Rechenberg/Schwefel (Bienert) 1961

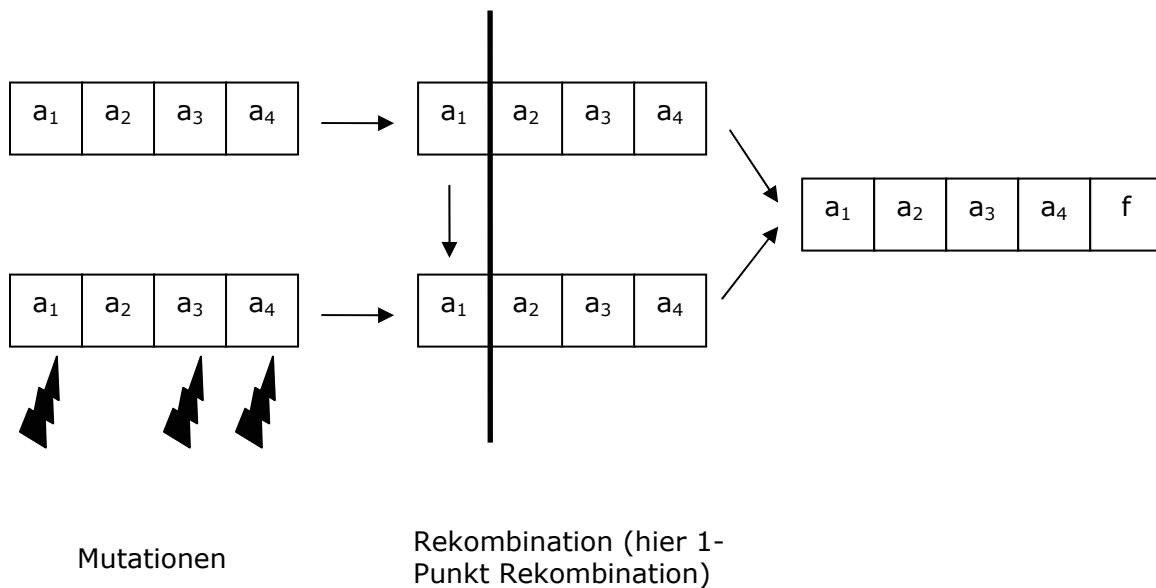
Luftwiderstand vom Punkt links zum Punkt rechts. Wie muss man die Winkel einstellen, damit es den geringsten Luftwiderstand hat?



Die Aufgabe war es, die Winkel  $\text{Alpha}_1 - \text{Alpha}_4$  so einzustellen, dass damit ein Optimum erreicht wurde.

$a_1$	$a_2$	$a_3$	$a_4$	fitness
-------	-------	-------	-------	---------

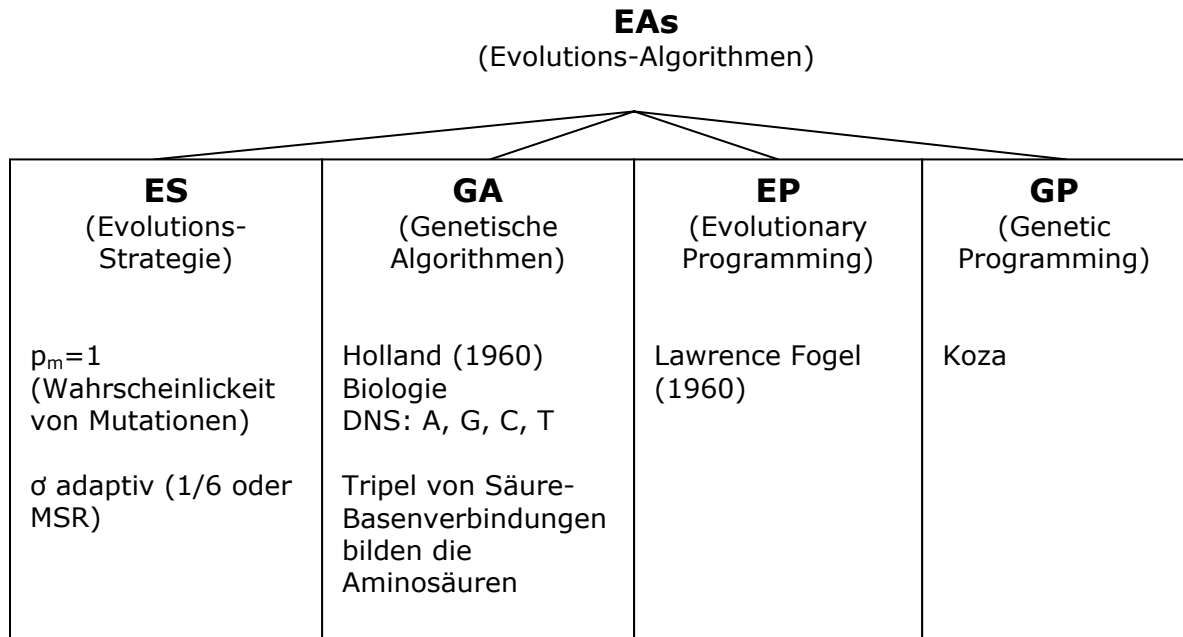
fitness = cost function, error function  
 $a_1 - a_4$  sind diskrete Werte, die man wählen kann



$\lambda$  Anzahl der Nachkommen ,N Anzahl der Eltern:

- (N, $\lambda$ )             $\rightarrow$  nur Kinder überleben
- (N+ $\lambda$ )            $\rightarrow$  Eltern werden auch ausgewertet, überleben also auch

Übersicht



Evolutions-Strategie

Mutationen: alle  $x_i$  werden mutiert

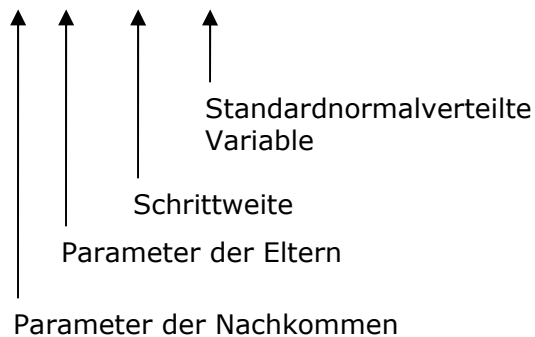
$N$  = Anzahl Eltern  $\rightarrow$  Mutationen  $\rightarrow$  Rekombination  $\rightarrow \lambda$  = Anzahl Nachkommen

Was ist die optimale Anzahl Eltern und Nachkommen?

Parallele Rechner: mehr Nachkommen, weniger Generationen

Serielle Rechner: weniger Nachkommen, mehr Generationen

$$x_i \leftarrow x_i + \sigma * N(0,1)$$



## 1/6 – Erfolgsregel: (nach Rechenberg)

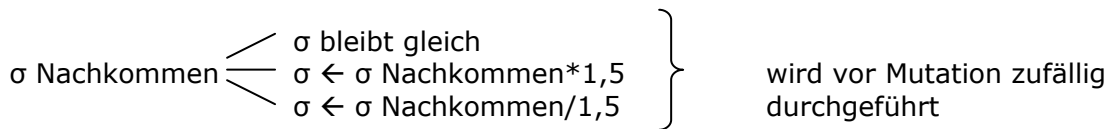
1/6 der Nachkommen besser als die Eltern  
 5/6 der Nachkommen schlechter als die Eltern

führt statistisch gesehen zum besten Fortschritt

falls = 1/6 →  $\sigma$  nicht verändern  
 falls < 1/6 →  $\sigma * 1,3$   
 falls > 1/6 →  $\sigma / 1,3$

## Mutative Schrittweitenregelung (MSR):

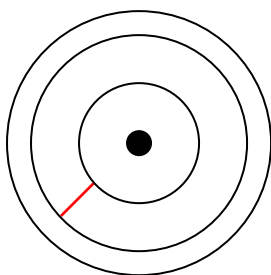
1. Kopieren  
 $\sigma$  Nachkommen →  $\sigma$  Eltern



→ pro Population gibt es dann 3 Kreise

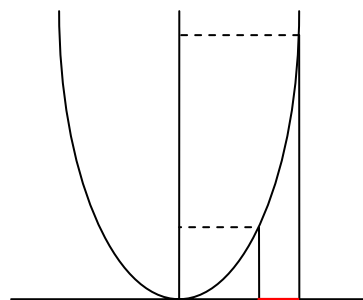
20.04.2002

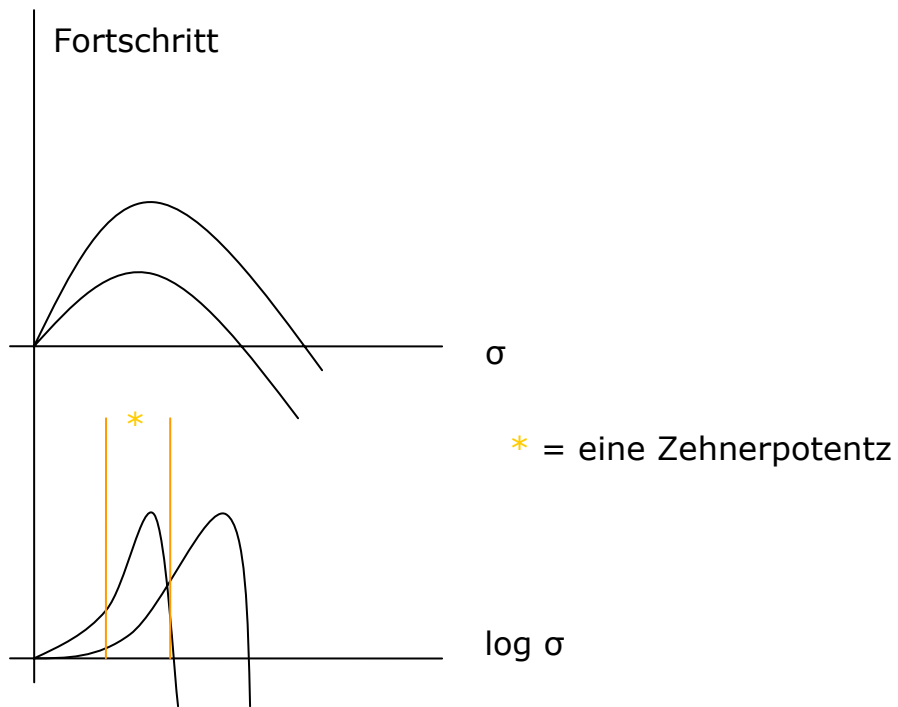
$$f(x_1 - x_n) = \sum_{i=1}^n x_i^2$$



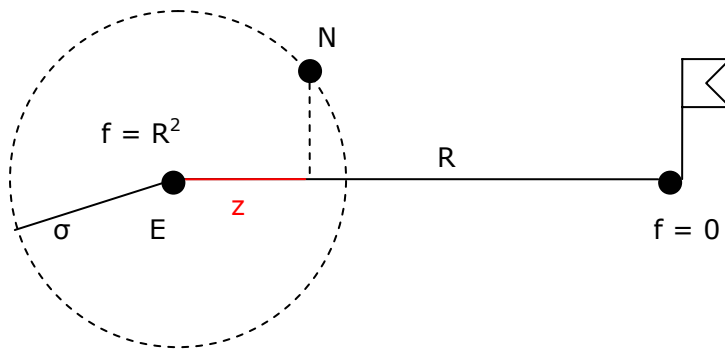
Fortschritt

(konzentrische Kreise als Höhenlinien)





relevante Grösse:  $\frac{ABSTAND}{SCHRITTWEITE}$



$z = \text{Fortschritt} = ?(R, f, \sigma, n)$

$f = R^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 = R^2$

$$E = \begin{pmatrix} 1 \\ 0,8 \\ 7 \\ 8 \\ 6 \\ 2,5 \end{pmatrix} \longrightarrow E = \begin{pmatrix} R \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \longrightarrow N = \begin{pmatrix} R-z \\ \sigma \\ \sigma \\ \sigma \\ \sigma \\ \sigma \end{pmatrix}$$

← Statistisches Mittel ergibt  $\sigma$ !  
 $\sigma$  sollte zu 50% positiv und zu 50% negativ sein (statistisches Mittel)

$$f(E) = (R-z)^2 + \sigma^2 + \sigma^2 + \sigma^2 + \sigma^2 + \sigma^2$$

$$f(N) = (R-z)^2 + (n-1) \sigma^2$$

$$f = R^2 - 2Rz + z^2 + (n-1) \sigma^2$$

$$\approx R^2 - 2Rz + z^2 + n\sigma^2$$

$$\varphi = f(E) - f(N) = R^2 - (R^2 - 2Rz + z^2 + n\sigma^2)$$

$$= 2Rz - z^2 - n\sigma^2$$

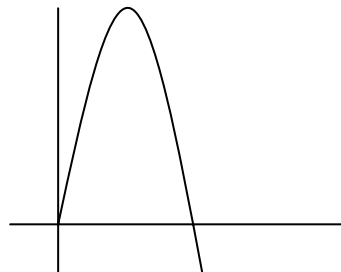
$$\approx 2Rz - n\sigma^2$$

$z^2$  ist kleiner als  $Rz$  und wird daher vernachlässigt

$$z = \sigma * C_{\mu+\lambda}$$

$$\varphi = 2R * C_{\mu+\lambda} * \sigma - n\sigma^2$$

$$\frac{\varphi}{R} = 2C * \sigma - \frac{n\sigma^2}{R}$$



Diese Berechnung gilt nur für die hier gewählte Fitnessfunktion, d.h. wir müssen für jede andere Fitnessfunktion den Fortschritt neu berechnen.

## Genetische Algorithmen

Mutation durch Bit Flip

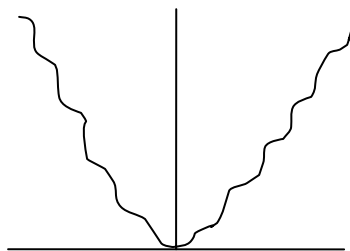
Algorithmen

Mutationswahrscheinlichkeit  $p_m = \frac{1}{n}$

0	0	0	0
1	1	1	1

Rekombinationswahrscheinlichkeit  $p_R = 0,8$  (möglichst hoch!)

$$\varphi(x_1, \dots, x_n) = \sum_i (x_i^2 - \cos 2x_i + 1)$$



$$t = O(n \ln n)$$

1. Eine Mutation reicht aus, um  $f_i(x_i)$  zu optimieren

2.  $p_m = \frac{1}{n}$

- Wahrscheinlichkeit  $x_i$  zu ziehen ist  $\frac{1}{n}$
- Wahrscheinlichkeit, dass  $x_i$  in  $g$  Generationen mindestens einmal gezogen wurde?

$$= 1 - \left(\frac{n-1}{n}\right)^g$$

Wahrscheinlichkeit, dass alle  $x_i$  gezogen wurden:

$$p = \left(1 - \left(\frac{n-1}{n}\right)^g\right)^n \quad \text{es gilt } g \geq n$$

$$p \approx 1 - n \left(\frac{n-1}{n}\right)^g$$

$$n \left(\frac{n-1}{n}\right)^g = 1 - p$$

$$\ln \left[ n \left( \frac{n-1}{n} \right)^g \right] = \ln(1-p)$$

$$\ln(n) + \ln \left( \frac{n-1}{n} \right)^g = \ln(1-p)$$

$$\ln(n) + g \ln \left( \frac{n-1}{n} \right) = \ln(1-p)$$

$$g = \frac{\ln(1-p) - \ln(n)}{\ln \left( \frac{n-1}{n} \right)}$$

$$\ln \left( \frac{n-1}{n} \right) = \ln \left( 1 - \frac{1}{n} \right)$$

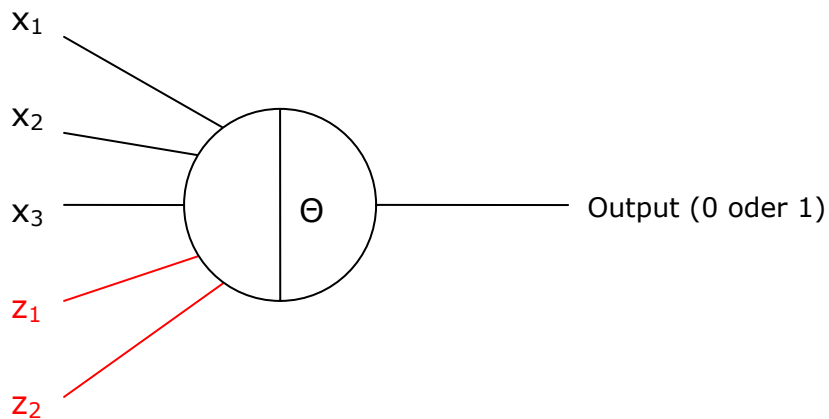
$$g = n \ln(n) - n \ln(1-p)$$

$$g = O(n \ln n)$$

21.04.2002

## Neuronale Netze

Mc Cullock & Pitts 1943/1947



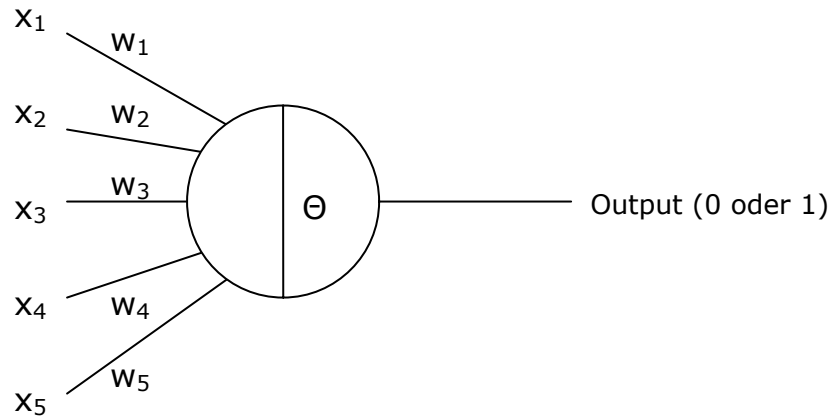
$$\sum x_i \geq \theta \begin{cases} \text{ja} \rightarrow 1 \\ \text{nein} \rightarrow 0 \end{cases}$$

Wenn ein  $z = 1$  ist, kommt als Output immer 0 heraus, sind alle  $z$  auf 0 gesetzt, so berücksichtigt das Netz die Enigabewerte  $x_i$ .

Man muss folgende Fälle unterscheiden:

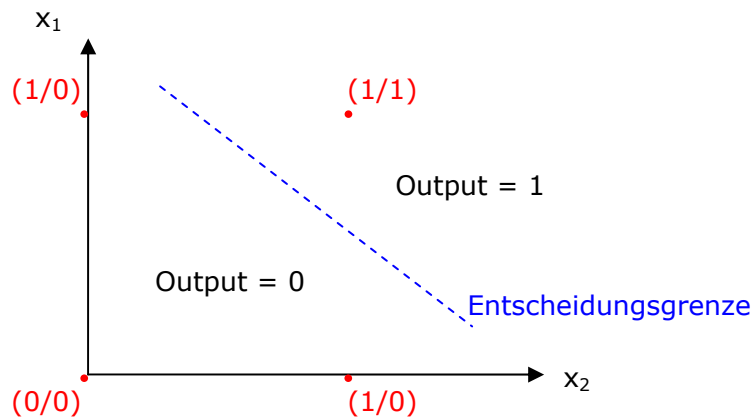
1. Modell selbst (nicht lernfähig)
2. Konfiguration des Netzes (Lernregeln etc.)

## Perzeptron



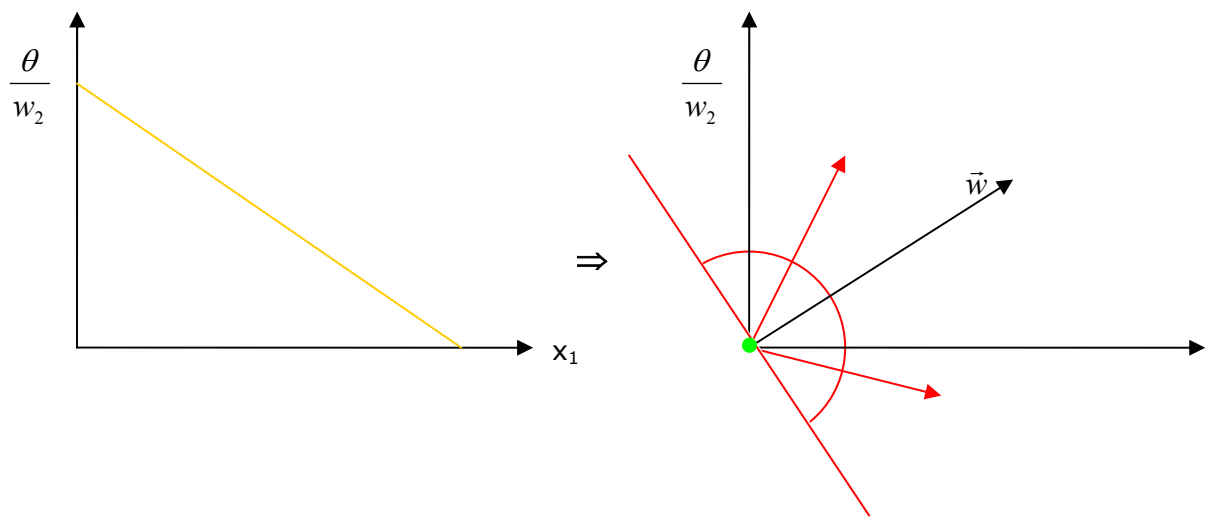
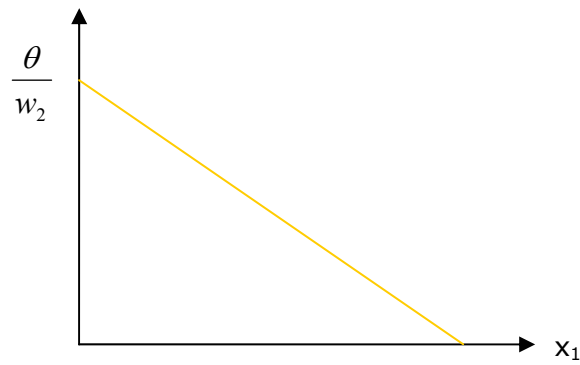
$$\begin{array}{ll} \sum x_i w_i \geq \theta & \rightarrow 1 \\ \text{sonst} & \rightarrow 0 \end{array}$$

Es gibt einen Algorithmus, der es erlaubt, die Gewichte der Schwellwerte mathematisch zu bestimmen. Dieser ist jedoch auf einschichtige Netze beschränkt.



$$\begin{array}{ll} x_1 w_1 + x_2 w_2 \geq \theta & (\geq \rightarrow =) \\ x_2 w_2 = -x_1 w_1 + \theta & \end{array}$$

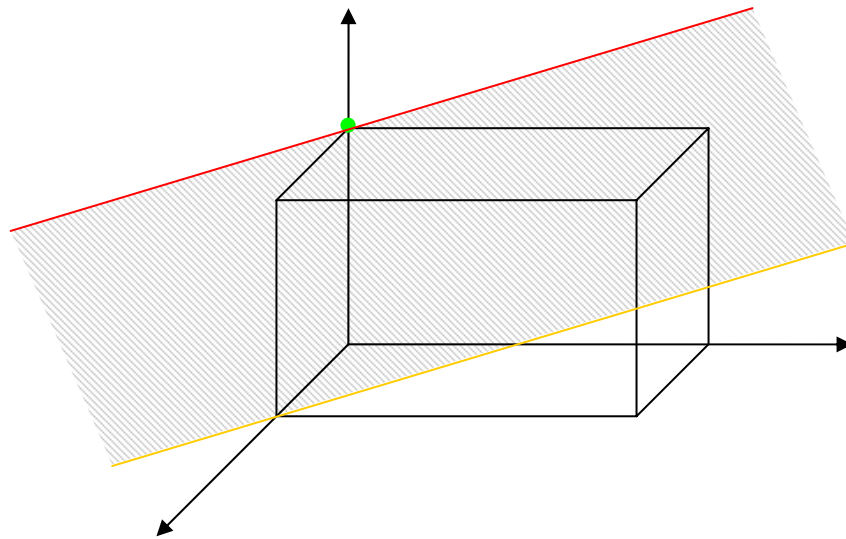
$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}$$



$$\sum_{i=1}^n x_i w_i \geq \theta$$

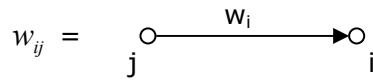
$$\sum_{i=0}^n x_i w_i \geq \theta$$

⇒ eine Dimension mehr

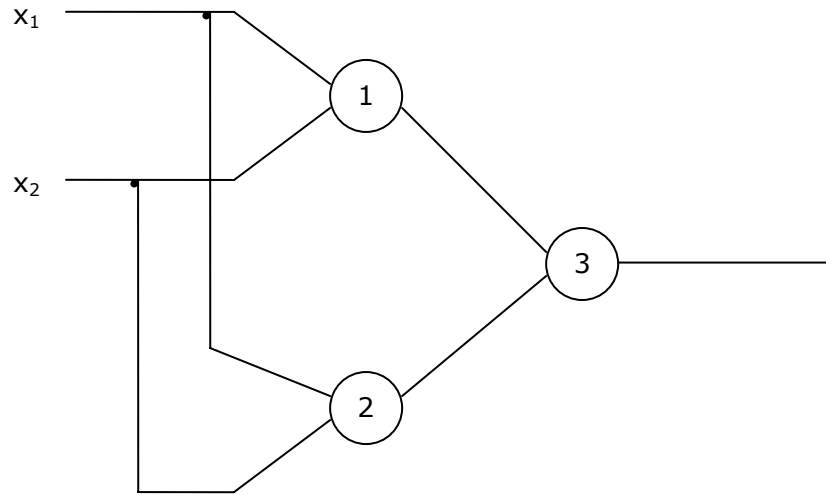


**Backpropagation**

Output:  $\sum_i x_i w_i$



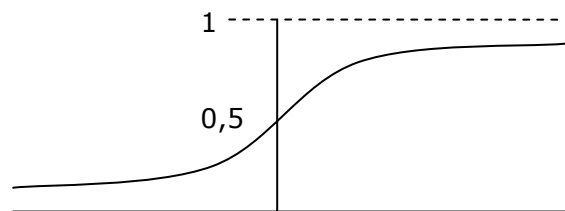
Aktivierung:  $o_i = \sum_j w_{ij} o_j$



$o_i = f(h_i)$

$h_i = \sum_{j=1}^n w_j o_j$  (= net input)

$f(h) = \frac{1}{1 + e^{-h}}$  (Sigmoidfunktion)



## Fehlerfunktion

Die Fehlerfunktion vergleicht den gewünschten mit dem effektiven Output.

$$f(w_{11}, w_{12}, \dots) = \frac{1}{2} \sum \sum (o_i - t_i)^2$$

Summiert wird über die Anzahl Patterns und über die Anzahl Output Neuronen, deshalb ergibt es die Doppelsumme in der Formel.

$o_i$  = effektiver Output

$t_i$  = gewünschter Output

Wie werden die Gewichte verändert?

1. Fitnessfunktion gegeben und Formelzusammenhang bekannt → Gradientenverfahren
2. Fitnessfunktion nicht gegeben Formelzusammenhang unbekannt → EA (ES, GA, ...)

→ Die Fitnessfunktion entspricht der Fehlerfunktion bei neuronalen Netzen und muss minimiert werden.

Gradientenverfahren:  $\vec{w}^{neu} = \vec{w}^{alt} - \eta \nabla f$        $\eta$  = Lernrate (anstatt Schrittweite)

Universalregel für alle neuronalen Netze:

$$w_{xy} \leftarrow w_{xy} + \underbrace{\eta o_i (1 - o_i)}_{f'} (t - o) o_j$$

(Delta-Regel usw. sind nur Spezialfälle davon!)

Design des neuronalen Netzes ist oft gegeben, hat aber starken Einfluss auf den Output.