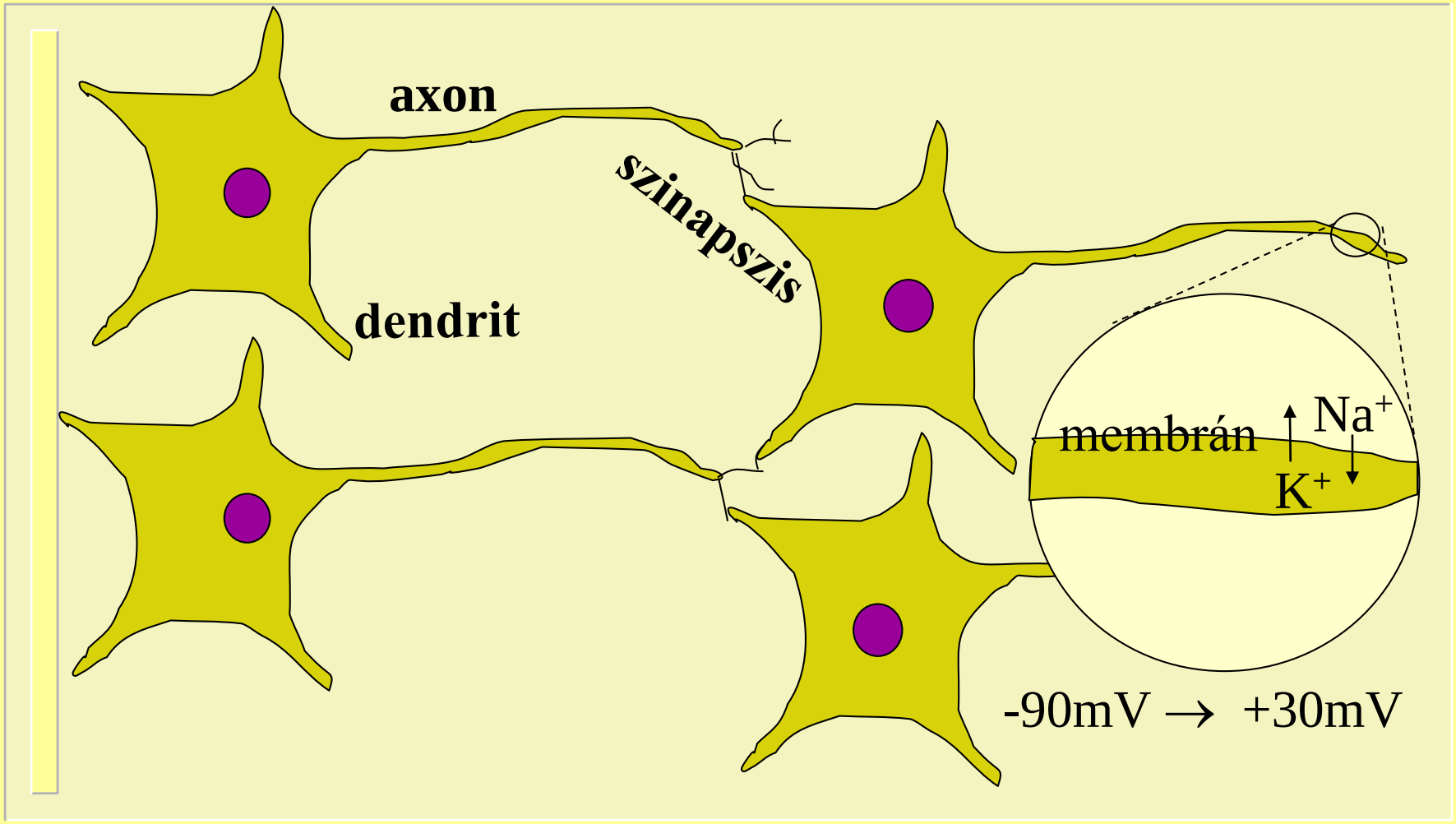
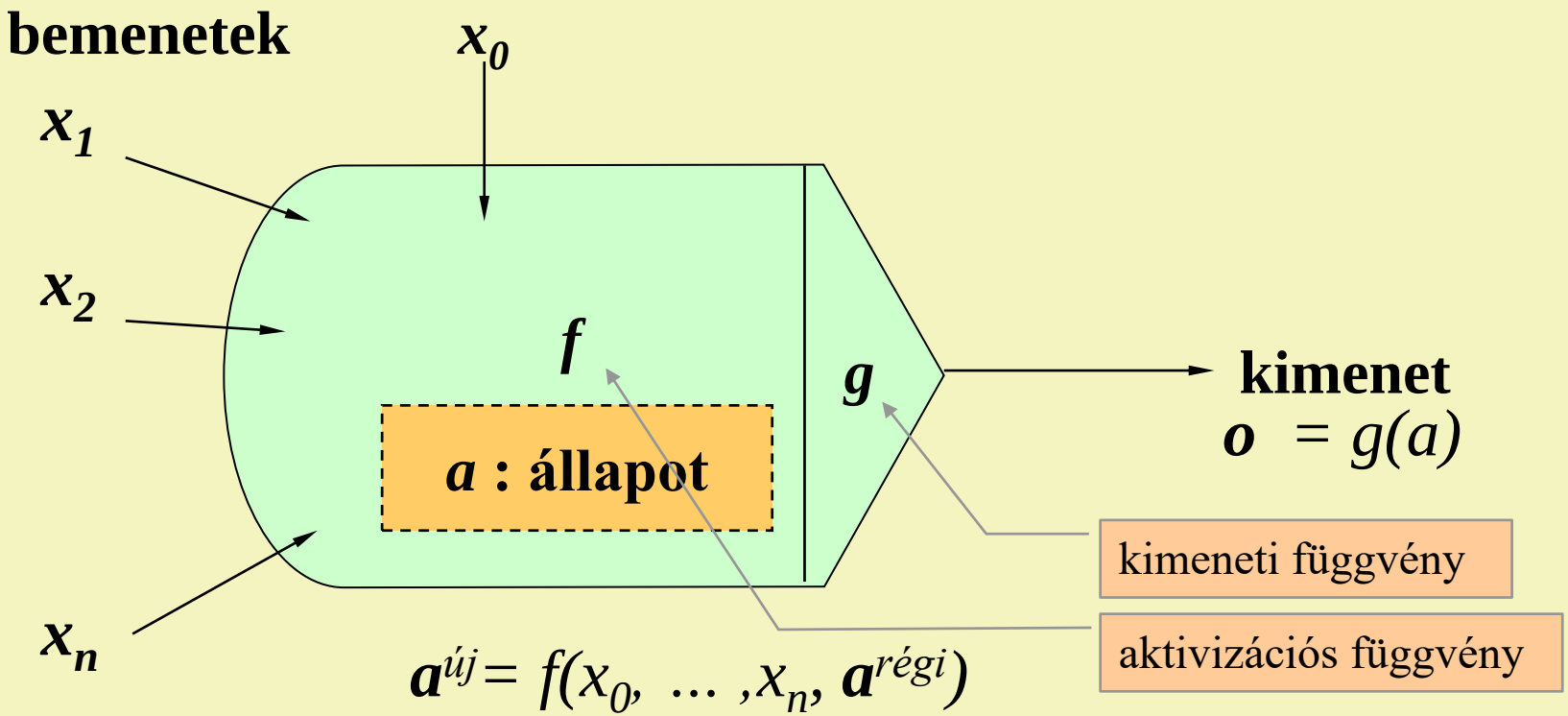
1.4. Mesterséges neuronhálók

- Mesterséges neuronhálók alkotóelemei
  - Mesterséges neuron
    - bemenő értékekből kimenő értéket számoló egység, amelynek számítási képlete változtatható, tanítható
  - Hálózati topológia
    - sok mesterséges neuron egymáshoz kapcsolva, ahol egyik neuron kimenete egy másik neuron bemenete lesz
    - bizonyos neuronok a bemenetüket a hálózaton kívülről kapják, mások kimeneteit pedig hálózat kimenetének tekintjük
  - Tanulási szabály
    - egy neuron számítási képletét meghatározó eljárás, amely lehet egy tanító példák alapján működő algoritmus is.
- Alkalmazás
  - osztályozás, approximáció, optimalizálás, asszociatív memória,

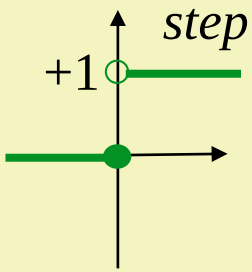
Természetes neuronhálók

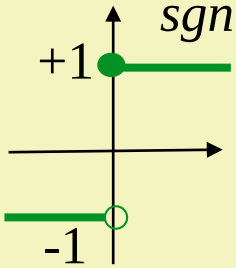


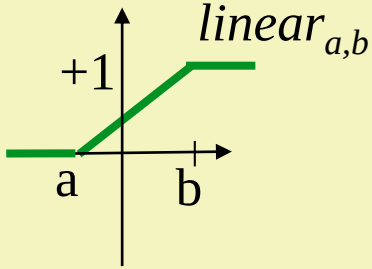
Általános mesterséges neuron

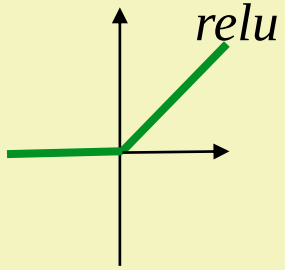


Kimeneti függvények

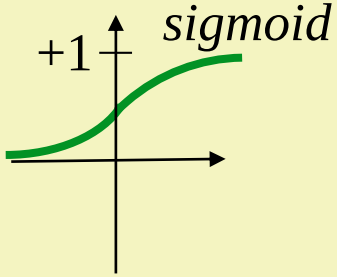




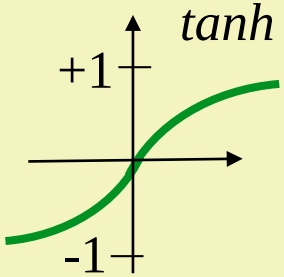




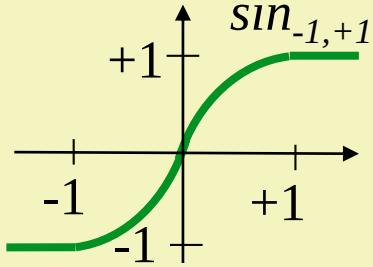
max(0, x)

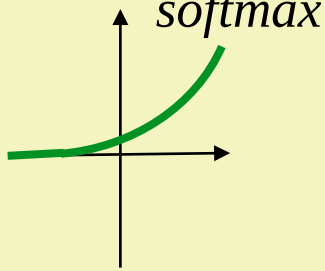


$$\frac{1}{1+e^{-x}}$$



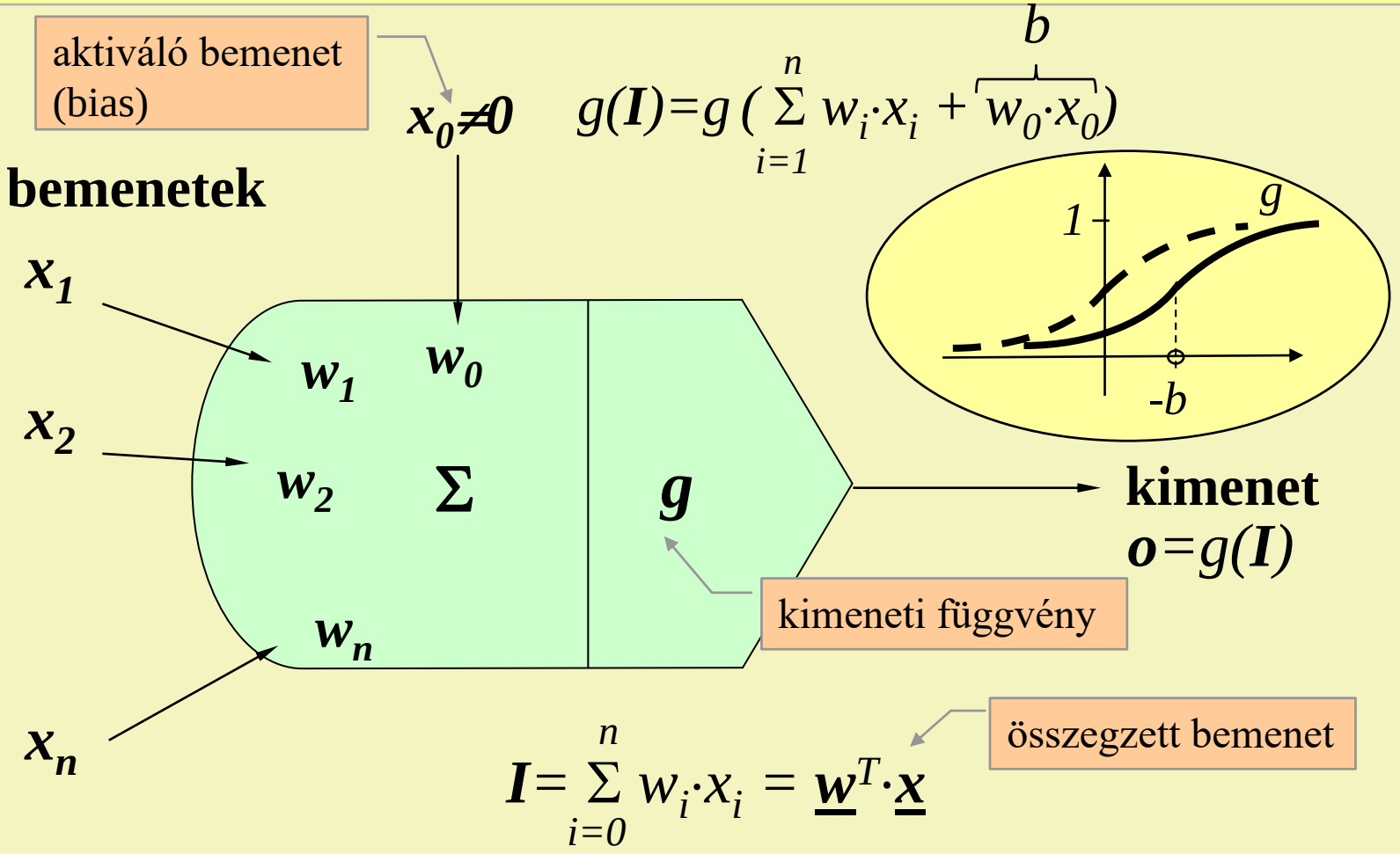
$$\frac{1-e^{-x}}{1+e^{-x}}$$





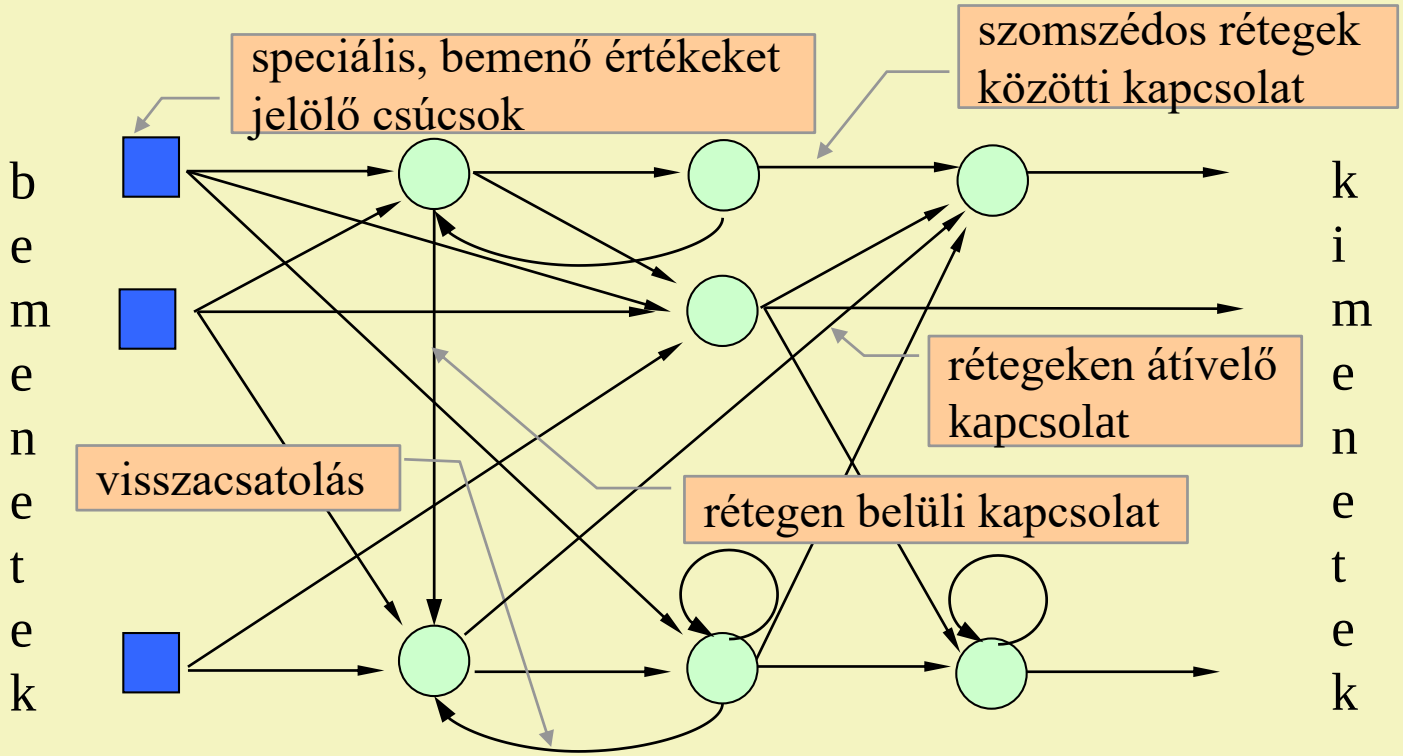
$$\frac{e^{x_j}}{\sum e^{x_k}}$$

Általánosított perceptron

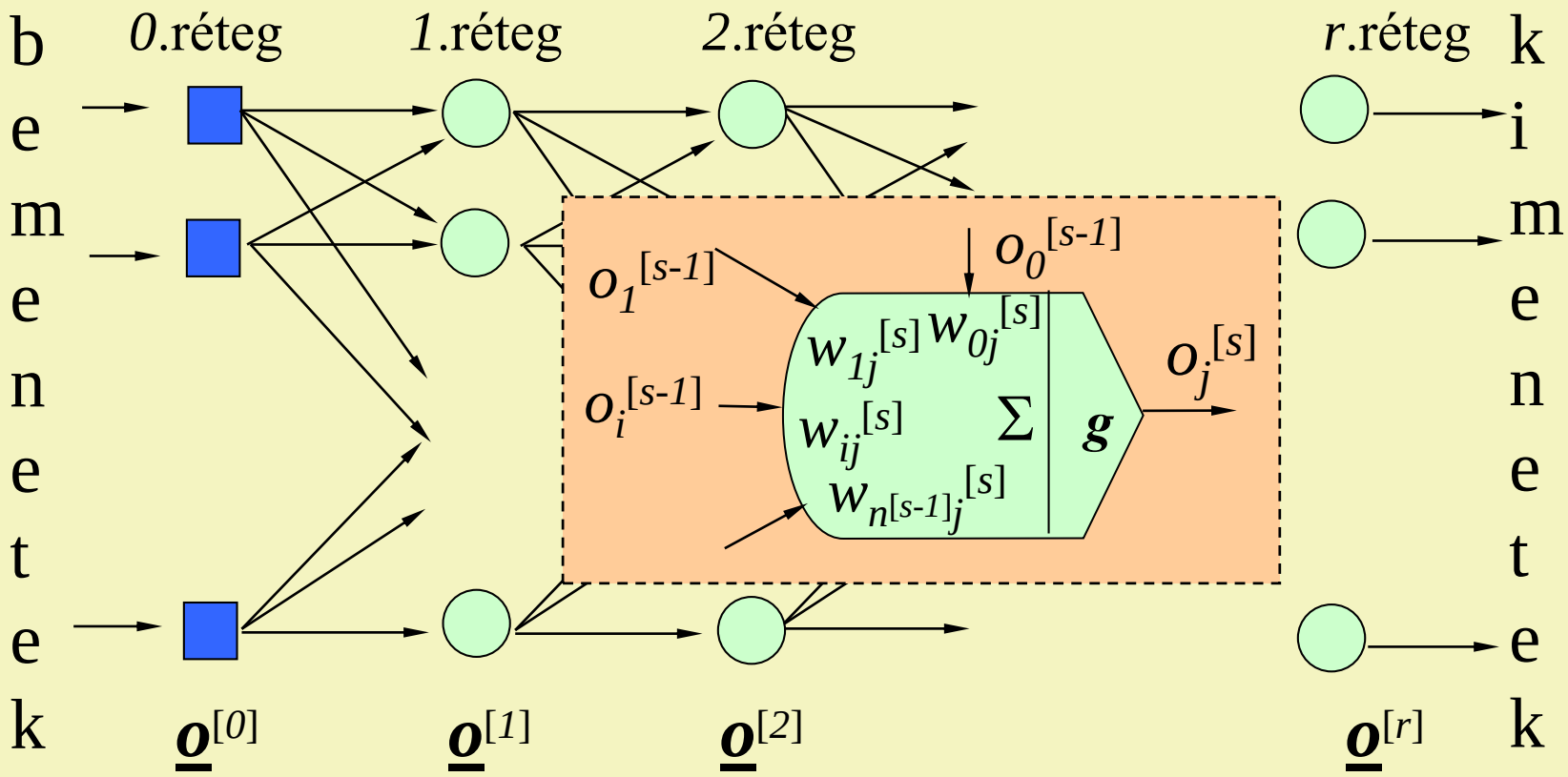


Hálózati topológia

Irányított gráf, amelynek csúcsai mesterséges neuronok, amelyek rétegekbe csoportosíthatók. Az irányított élek az adatáramlás irányát jelölik:
$a \rightarrow b$: az a neuron kimeneti értékét kapja meg a b neuron bemenetként.

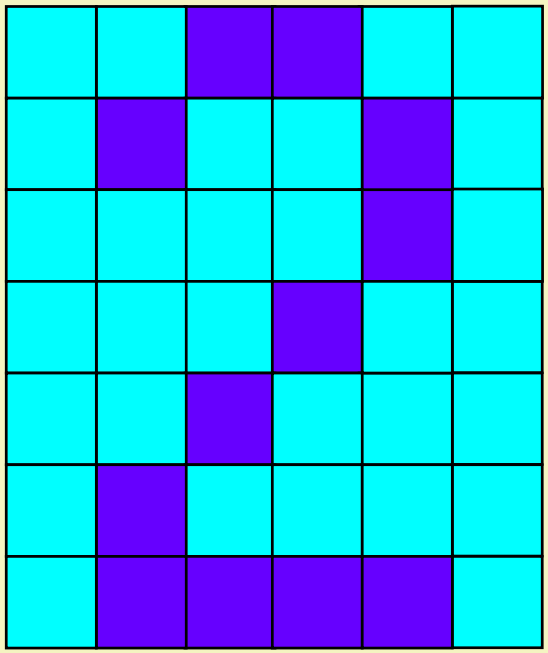


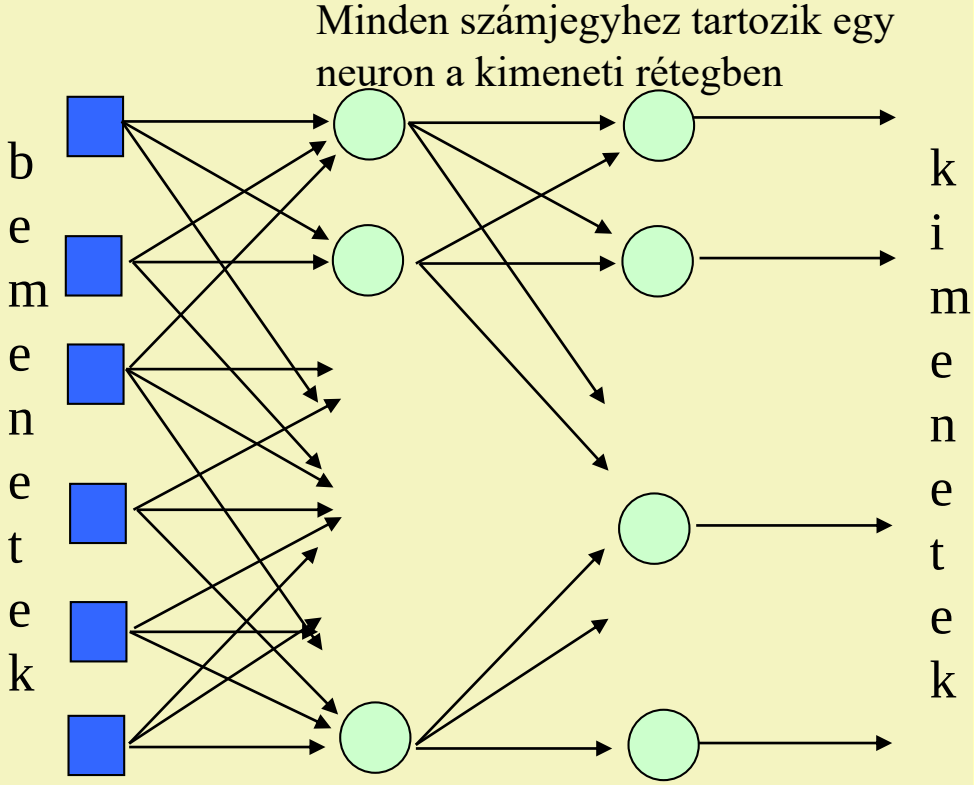
Többrétegű előrecsatolt hálózat (feed forward MLP)



Számjegy felismerés

Bemeneti értékek száma: 42 Kimenetek száma: 10



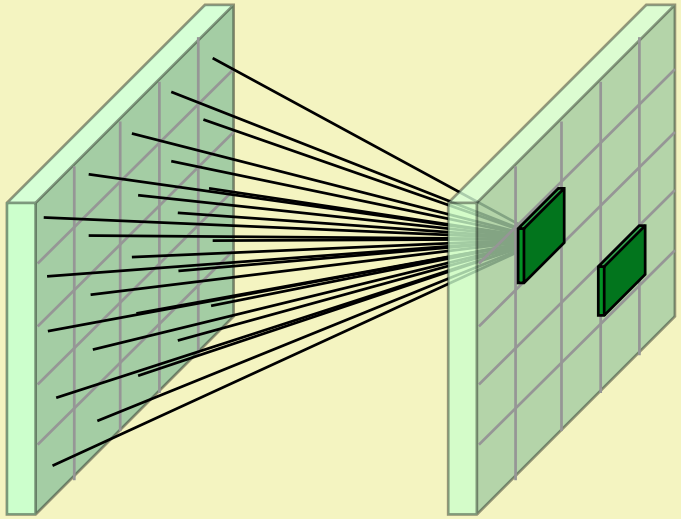


Beállítások: közbülső réteg neuronjainak száma: 11

$x_i \in \{0,1\}$, $f(x)=szigmoid(x)$, $o^s_i \in (0,1)$, $w^s_{ij}=rand(-0.1 , 0.1)$, $o^s_0=1.0, \eta=0.35$

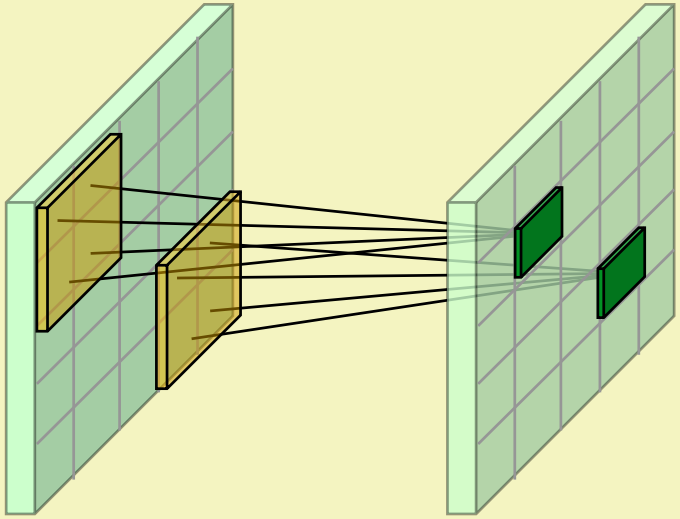
Konvolúciós neuron hálózat

Teljesen összekötött (sűrű) fully connected neural net



Pl.: két 1000×1000 réteg esetén
a második réteg egy neuronjában
10^6 darab súlyt kell tárolni.

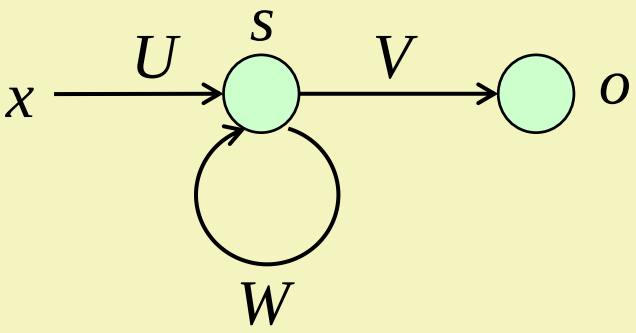
Lokálisan összekötött convolutional neural net

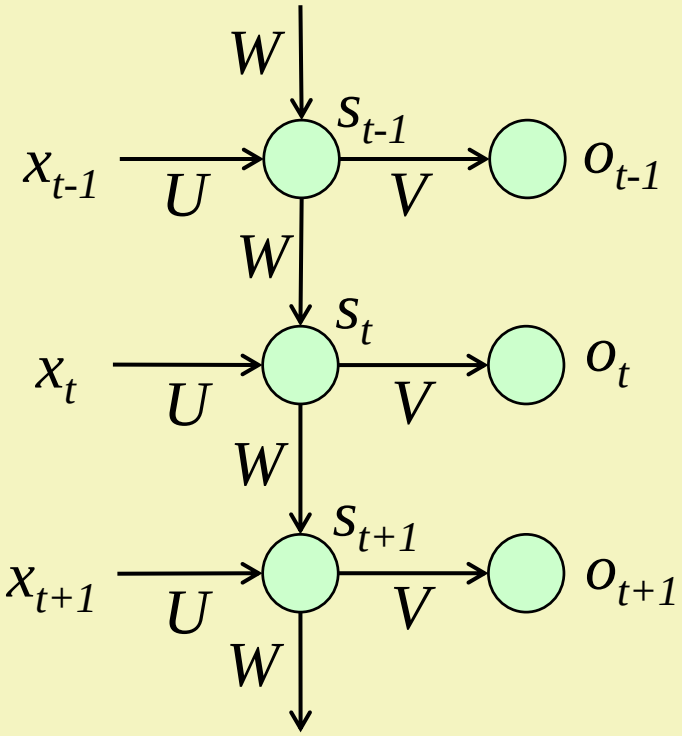


Pl.: két 1000×1000 réteg esetén
egy 2×2-es szűrőt használva
a második réteg egy neuronjában
csak 4 súlyt kell tárolni.

Rekurrens neurális hálózat

- Az input és/vagy az output változó hosszúságú sorozat.
- A számítás a belső memória segítségével emlékezik a megelőző inputokra.

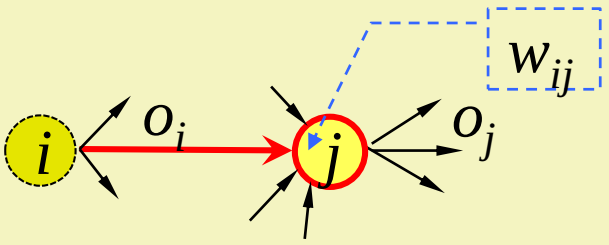




Általánosított perceptron tanulása

- Egy neuron számítási képletét a neuron *w* súlyai határozzák meg.
- A súlyok implicit módon a hálózat topológiáját is kijelölik, hiszen neuronhoz vezető nulla értékű súllyal ellátott él lényegében az él figyelmen kívül hagyását (törlését) jelenti.
- Tanulás során a súlyokat fokozatosan módosítjuk ($w:=w+\Delta w$).
- A Δw a neuron bemeneti értékeitől és a neuron által kiszámított kimeneti értéktől függ.
  - Felügyelt tanulás esetén felhasználjuk az elvárt kimenetet.
  - Felügyelet nélküli tanulás esetén az elvárt kimenetre nincs szükség.

Példák egy neuron súlyát módosító tanuló szabályokra



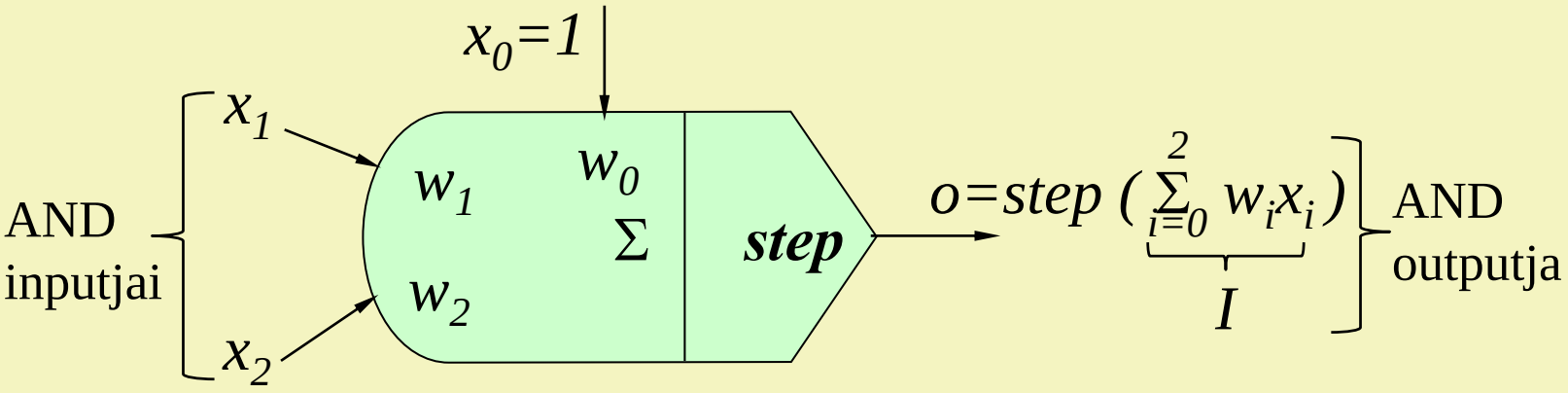
o_i az i neuron számított kimenete és egyben a j neuron i-dik bemenete is
w_{ij} a j neuron i-dik bemenetének súlya
o_j a j neuron számított kimenete

Felügyelt tanulás	**Felügyelet nélküli tanulás**
Pl: Delta szabály: $\Delta w_{ij} = \eta \cdot o_i \cdot (y_j - o_j)$ y_j a j-dik neuron várt kimenete	Pl: Hebb szabály: $\Delta w_{ij} = \eta \cdot o_i \cdot o_j$

η tanulási együttható a tanulás sebességét befolyásolja

Példa felügyelt tanulásra

Tanítsuk meg egy egyszerű számoló egységnek (egyetlen mesterséges neuronnak) a logikai AND művelet működését!



x_i a neuron i-dik bemenete ($x_i \in \{0,1\}$, $x_0 = 1$)
w_i a neuron i-dik bemenetének súlya ($w_0, w_1, w_2 \in \mathbb{R}$)
I a neuron összegzett bemenete
o a neuron számított kimenete ($o \in \{0,1\}$)

i-edik súly változása

felügyelt tanulási szabály: $\Delta w_i = \eta \cdot x_i \cdot (y\text{-}o)$

hiba (e)
y ~ várt
o ~ számított

$\eta = 0.1$

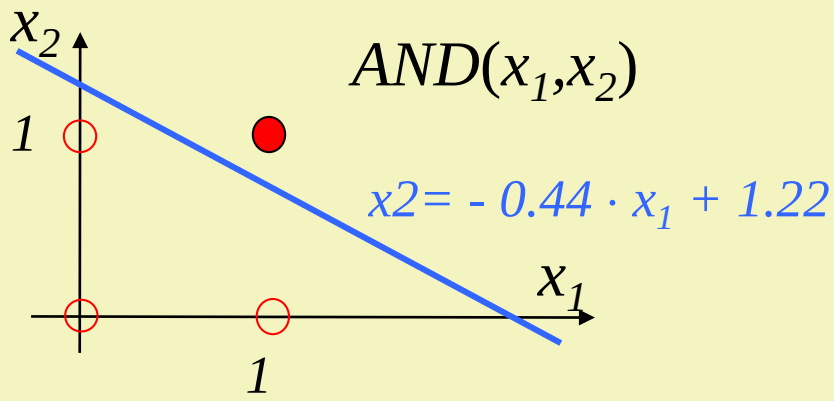
	x_1	x_2	y	w_0	w_1	w_2	I	o	e
				0.08	**0.08**	**0.08**			
1.	1	0	0	0.08	+ 0.08	+ 0	= 0.160	1	-1
				-0.02	**-0.02**	**0.08**			
2.	0	1	0	-0.02	+ 0	+ 0.08	= 0.06	1	-1
				-0.12	**-0.02**	**-0.02**			
3.	1	1	1	-0.12	+ -0.02	+ -0.02	=-0.16	0	1
				-0.02	**0.08**	**0.08**			
4.	1	0	0	-0.02	+ 0.08	+ 0	= 0.06	1	-1
				-0.12	**-0.02**	**0.08**			
5.	0	1	0	-0.12	+ 0	+ 0.08	= -0.04	0	0
				-0.12	**-0.02**	**0.08**			
6.	1	1	1	-0.12	+ -0.02	+ 0.08	= -0.06	0	1
...									
13.				**-0.22**	**0.08**	**0.18**			
14.	1	0	0	-0.22	+ 0.08	+ 0	= -0.14	0	0
15.	0	1	0	-0.22	+ 0	+ 0.18	= -0.04	0	0
16.	1	1	1	-0.22	+ 0.08	+ 0.18	= 0.04	1	0
17.	**0**	**0**	**0**	-0.22	+ 0	+ 0	= -0.22	**0**	0

első epoch

második epoch

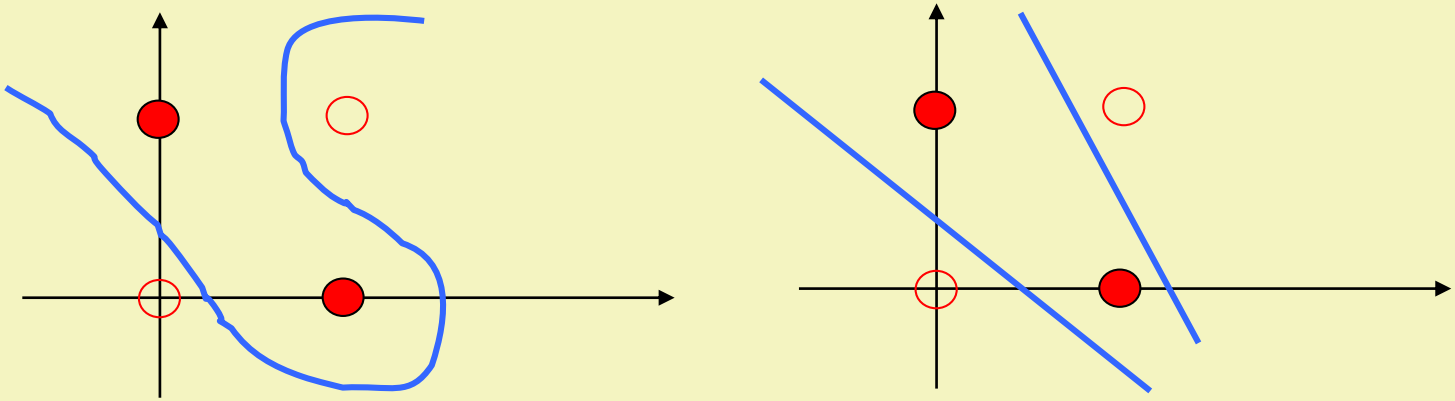
Mit tanultunk meg?

- A példában a lehetséges bemenet-párok alkotta sík egy egyenesét, pontosabban az egyenes $I(x_1,x_2)=w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$ képletében szereplő *w* együtthatókat tanultuk meg. Ez az egyenes a lehetséges bemenet-párokat (sík bizonyos pontjait) két csoportra vágja.
  - $I(x_1,x_2) \leqq 0$ esetén a $step(I(x_1,x_2)) = 0 = AND(x_1,x_2)$
  - $I(x_1,x_2) > 0$ esetén a $step(I(x_1,x_2)) = 1 = AND(x_1,x_2)$



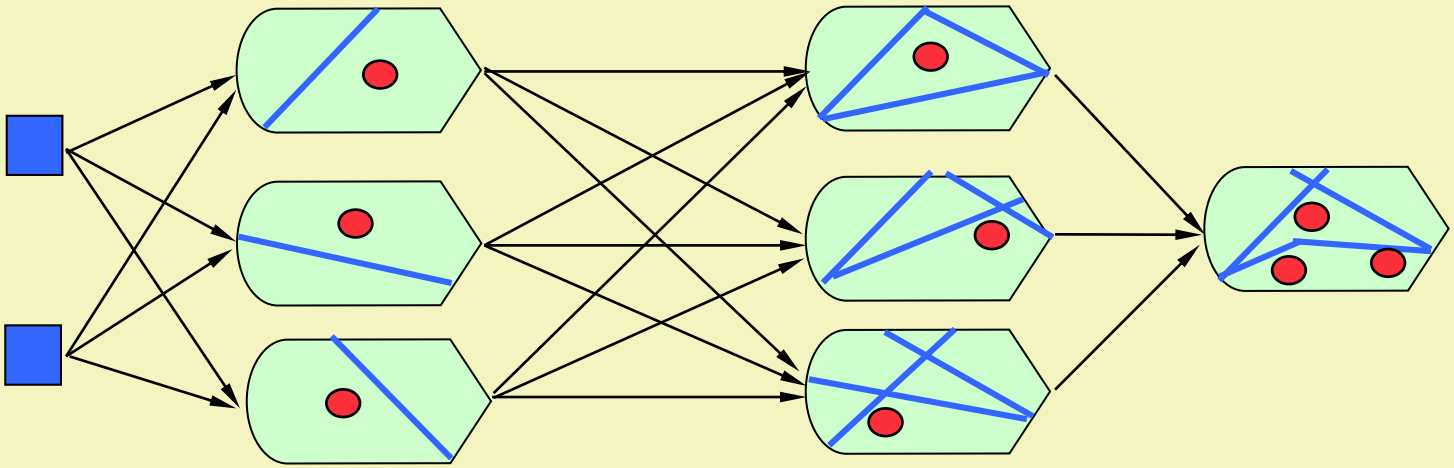
Lineáris szeparálhatóság

- Egyetlen perceptronnal olyan bonyolultságú feladatot vagyunk képesek megoldani, ahol az eredményük alapján a bemeneteket úgy lehet két csoportba sorolni, hogy azokat egy hipersík választja szét, azaz lineárisan szeparálhatók.
- Nem lehet például a XOR műveletet egyetlen perceptronnal megvalósítani, mert ez a feladat lineárisan nem szeparálható.



Rétegek „tudása”

- Egy egyrétegű perceptron modell neuronjai tehát csak féltereket képesek felismerni (osztályozni), de egy kétrétegű már ezek kombinációit, azaz konvex poliédereket is, egy három rétegű pedig tetszőleges poliédereket.



Homogén MLP háló számítási modellje

- Egy réteg számítási modellje

s–1-edik réteg neuronjainak száma = s-dik réteg neuronjainak bemenet száma

az *s*-edik réteg *j*-edik neuronjában az *i*-edik bemenethez tartozó súly

$$o_j^{[s]} = g(\underline{\underline{w}}_j^{[s]} \cdot \underline{o}^{[s-1]}) = g(\sum_{i=0}^{n^{[s-1]}} w_{ij}^{[s]} \cdot o_i^{[s-1]})$$

s–1-edik réteg *i*-dik neuronjának kimenete = az *s*-edik réteg neuronjainak *i*-dik bemenete is

- A teljes háló számítási modellje:

$$f(\Theta, \underline{x}) = g(\underline{\underline{w}}^{[r]} \cdot \ldots \cdot g(\underline{\underline{w}}^{[s]} \cdot \ldots \cdot g(\underline{\underline{w}}^{[2]} \cdot g(\underline{\underline{w}}^{[1]} \cdot \underline{x})) \ldots) \ldots)$$

a Θ paraméter tehát a $(w_{ij}^{[s]})$ súlyok összessége

Hiba visszaterjesztés módszere (error backpropagation)

A modell hibafüggvénye: $L(\Theta) = \frac{1}{2}\sum_{j=1}^{n^{[r]}} (y_j - o_j^{[r]})^2$

$f(\Theta, \underline{x})$

Az $L(\Theta)$ egy olyan több változós függvény, amely a $(w_{ij}^{[s]})$ súlyoktól függ. Ennek a függvénynek keressük a minimum helyét gradiens módszerrel:

$$w_{ij}^{[s]} := w_{ij}^{[s]} - \underbrace{\eta \cdot \frac{\partial L}{\partial w_{ij}^{[s]}}}_{\Delta w_{ij}^{[s]}}$$

$$\frac{\partial L}{\partial w_{ij}^{[s]}} = \eta \cdot \frac{\partial L}{\partial I_j^{[s]}} \cdot \frac{\partial I_j^{[s]}}{\partial w_{ij}^{[s]}} = \eta \cdot \underbrace{\frac{\partial L}{\partial I_j^{[s]}}}_{e_j^{[s]}} \cdot o_i^{[s-1]}$$

hiszen $I_j^{[s]} = \sum_{i=0}^{n^{[s-1]}} w_{ij}^{[s]} o_i^{[s-1]}$

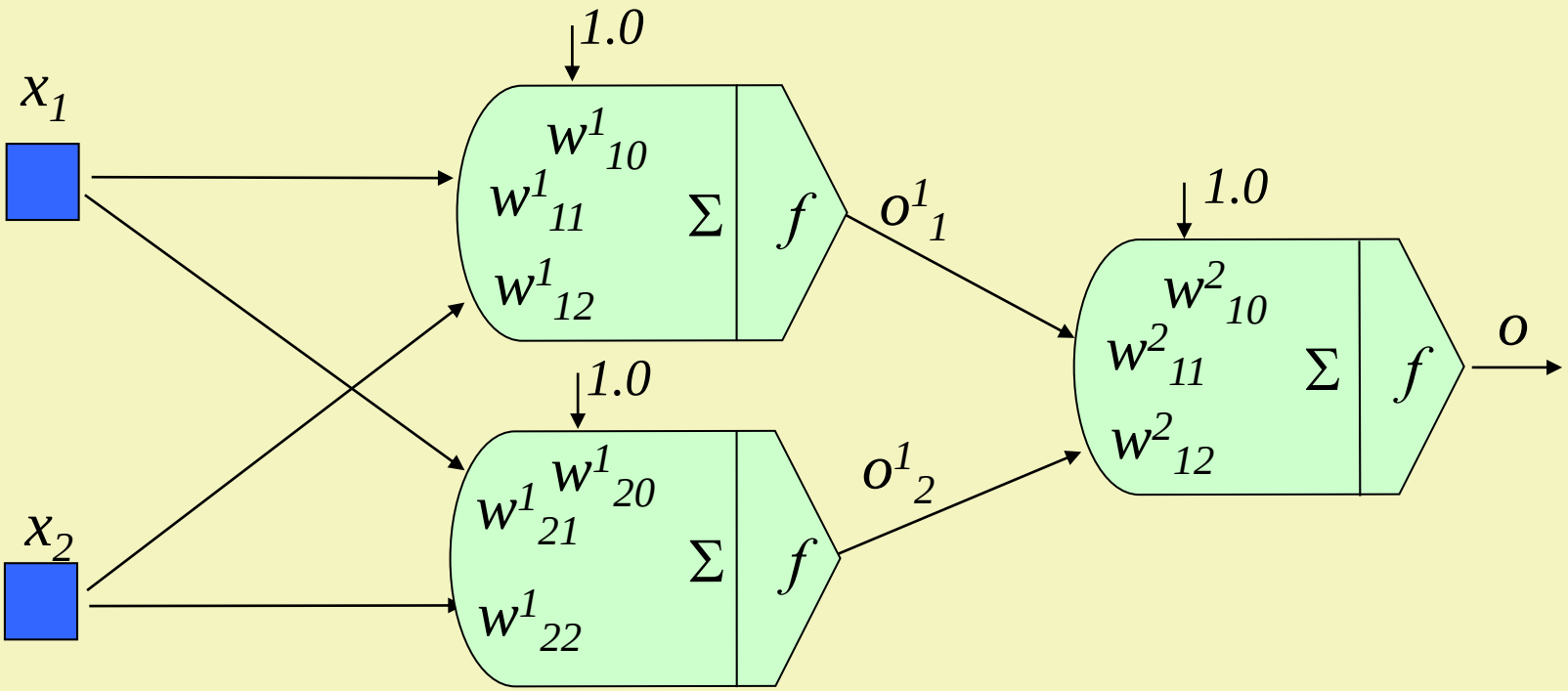
a számítási hibának az *s*-edik réteg *j*-edik neuronjára jutó hányada

Backpropagation tanuló algoritmus

1. Az $\underline{x}$ bemeneti vektorból kiindulva rétegenként kiszámoljuk a neuronok kimeneteit: $o_j^{[s]}$, így eljutunk a kimeneti réteg kimeneteihez $o_j^{[r]}$ is.
2. A kimeneti réteg minden neuronjára kiszámoljuk a lokális hibát: $e_j^{[r]} = o_j^{[r]} \cdot (1 - o_j^{[r]}) \cdot (t_j - o_j^{[r]})$ ha g a szigmoid függvény
3. Rétegenként hátulról előre haladva számoljuk a belső neuronok hibáit:
$$e_j^{[s]} = o_j^{[s]} \cdot (1 - o_j^{[s]}) \cdot \left(\sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} \cdot w_{jk}^{[s+1]} \right)$$
ha g a szigmoid függvény
4. Végül módosítjuk a hálózat súlyait: $w_{ij}^{[s]} := w_{ij}^{[s]} + \Delta w_{ij}^{[s]}$ ahol a súlytényező-változás: $\Delta w_{ij}^{[s]} = \eta \cdot e_j^{[s]} \cdot o_i^{[s-1]}$

XOR művelet példája

Beállítások: $x_1, x_2 \in \{0,1\}$ $f(x)=logisztikus(x)$ $o^s_i \in (0,1)$
$w^s_{ij} = rand(-0.1 , 0.1)$ $o^s_0 = 1.0$ $\eta = 1.0$



Rétegenként eltérő számítási képlet

- Egy neuronháló számítási modellje:
  $f : P \times X \rightarrow Y$
  $f(\Theta, \underline{x}) = g_r(\underline{\underline{W}}^{[r]}, g_{r-1}(\ldots g_2(\underline{\underline{W}}^{[2]}, g_1(\underline{\underline{W}}^{[1]}, \underline{x})) \ldots))$
  - $\Theta = \{w^{[1]}, \ldots, w^{[r]}\}$ azaz a paraméterek a súlyok
  - $g_s : P \times X^{s-1} \rightarrow X^s$ s-edik réteg kimeneti függvénye
    $(X = X^0, Y = X^r)$
- Hibafüggvény:

$$L(\Theta) = \frac{1}{N} \sum_{n=1}^{N} \ell (\underbrace{f(\Theta, x_n)}_{t_n}, y_n)$$

$\ell : Y \times Y \rightarrow \mathbb{R}$ hibafüggvény: $\ell(t_n, y_n)$ lehet például $\|t_n - y_n\|_1$, $\|t_n - y_n\|_2^2$, vagy $-\Sigma_i y_{n\,i} \cdot \log t_{n\,i}$

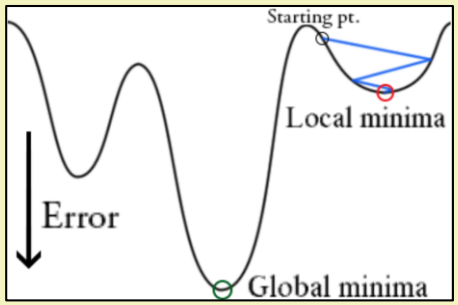
Gradiens módszer

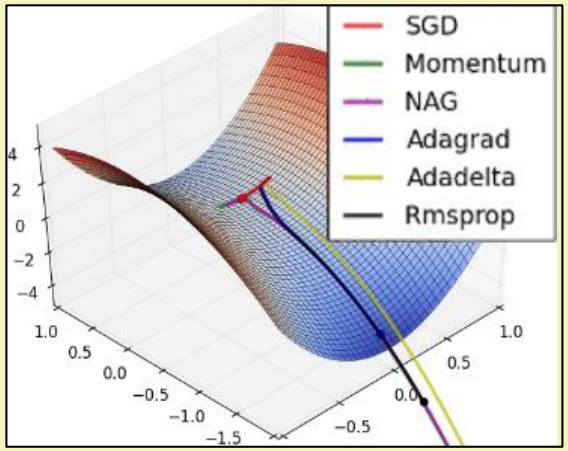
- A gradiens elméleti kiszámítása nehéz:

$$\Theta_{új} := \Theta_{régi} - \eta \cdot \frac{\partial L(\Theta)}{\partial \Theta}\Bigg|_{\Theta=\Theta_{régi}}$$

  ezért helyette numerikus módszereket használnak.

- A lokális minimum-, illetve a nyeregpontok problémája:





- A tanító minták kiválasztása nehéz (homogén minták, overfitting)
- A hiperparaméterek (N, η) megtanulásának kérdése

A mélytanulás módszerei ezeken a hátrányokon igyekeznek javítani.