Alkalmazott mesterséges intelligencia (AMI)

http://www.mit.bme.hu/oktatas/targyak/vimibb01

5. ea. (2023 ősz)

Neurális hálók

http://http://mialmanach.mit.bme.hu/aima/ch20s05 20.5. fejezet
http://mialmanach.mit.bme.hu/neuralis/ch04 4. fejezet

Előadó: Pataki Béla

a fóliák
Dobrowiecki Tadeusz és
Hullám Gábor anyagainak
felhasználásával készültek

https://www.esrcheck.com/2023/06/05/artificial-intelligence-ai-experts-sign-statement-on-ai-risk/

BME I.E. 414, 463-26-79
pataki@mit.bme.hu,
http://www.mit.bme.hu/general/staff/pataki

Tanulás alapvető fajtái:

felügyelt (ellenőrzött) tanulás a tanítani kívánt rendszernél az összes ismert bemenetnél tudjuk, hogy az egyes bemeneti mintákra milyen kimenetet várunk (tanítóminta: bemeneti minta + kívánt válasz)

megerősítéses tanulás az ágens az általa végrehajtott tevékenység csak bizonyos értékelését kapja meg, esetleg nem is minden lépésben (jutalom, büntetés, megerősítés)

felügyelet nélküli (nem ellenőrzött) tanulás semmilyen információ sem áll rendelkezésünkre a helyes kimenetről (az észlelések közötti összefüggések tanulása)

féligellenőrzött tanulás a tanításra használt esetek egy részénél mind a bemenetet, mind a kimenetet észlelni tudjuk (bemeneti minta + kívánt válasz), a másik – tipikusan nagyobb – részénél csak a bemeneti leírás ismert

Induktív (ellenőrzött) tanulás (konkrét, egyedi példákból általánosít)

A tanítás folyamata:

Kiinduló (tanító) mintahalmaz

$\{(\mathbf{x}_n, d_n)\}$, $n=1,\ldots,N$

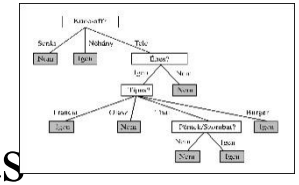
Például:

$\mathbf{x}_n=[x_{n1}=1{,}5 \; ; \; x_{n2}=1; \; x_{n2}=$‚SZÉP’, ...]

$d_n=$‚IGAZ’

h(x) hipotézis, mintákból leszűrendő általános szabály/tudás

például:



Tanítási algoritmus (hogyan építsük be a mintákban hordozott tudást az eszközbe)

Például: döntési fa kialakítása, növesztése

Induktív tanulás

A megtanított eszköz felhasználása

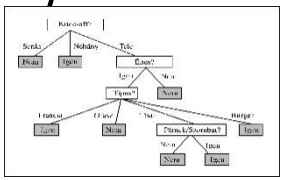
Új, ismeretlen szituáció leírása: $\mathbf{x}_{új}$

Például:

$\mathbf{x}_{új}$=[$x_{új1}$=2,7 ; $x_{új2}$=0; $x_{új3}$=‚CSÚNYA’, ...]

$h(\mathbf{x})$ hipotézis, mintákból leszűrt általános szabály/tudás

például:



Az új, ismeretlen szituációra ($\mathbf{x}_{új}$) javasolt válasz

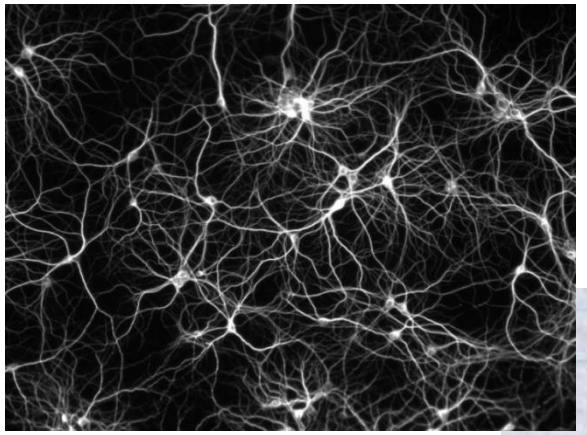
Például:

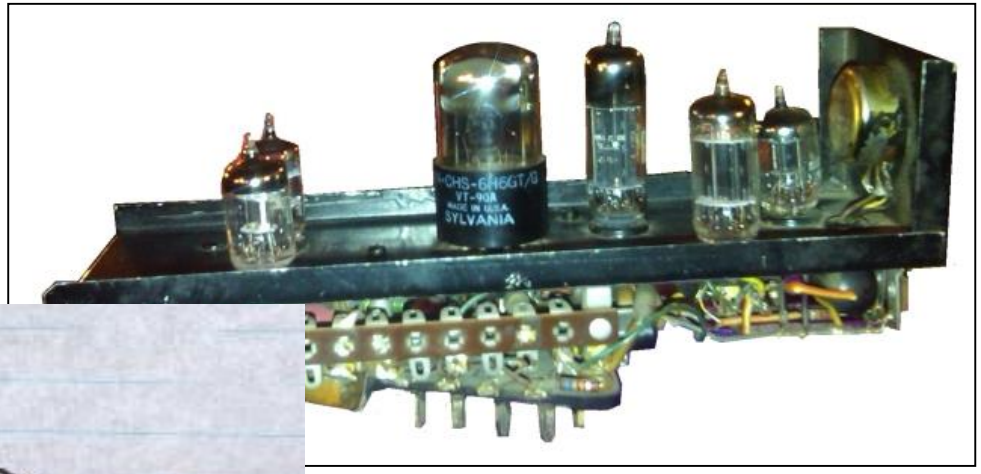
$h(\mathbf{x}_{új})$=‚NEM’

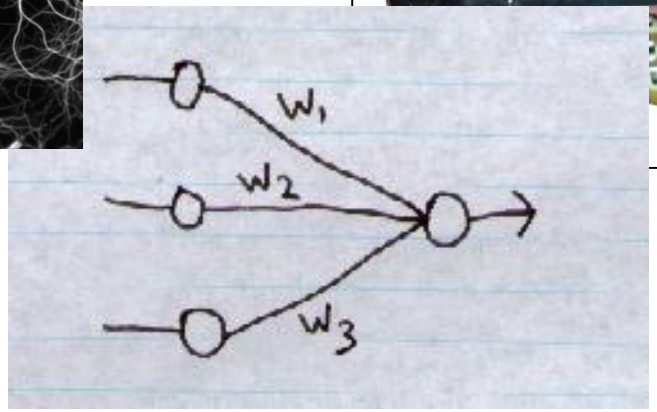
Sokféle tanítható eszközt kitaláltak:

- **neurális hálók**
- Bayes-hálók
- kernelgépek
- döntési fák
- legközelebbi szomszéd osztályozók
- stb. stb.

A következőkben először a neurális hálókkal foglalkozunk – mostanában már a harmadik hullámban nőtt meg a népszerűségük. Működésük az emberi gondolkodás számára jóval kevésbé áttekinthető, mint pl. a döntési fáké.



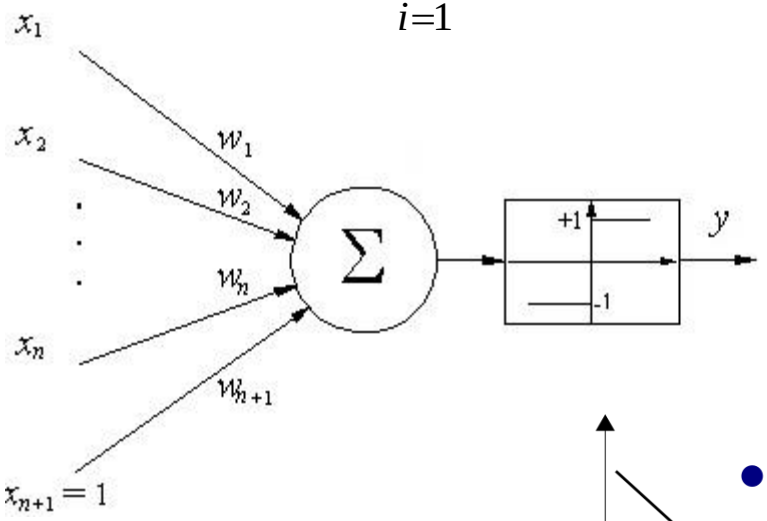




Neuron doktrina: S. Ramón y Cajal (1852-1934)
Mesterséges neuron: W. McCulloch and W. Pitts, 1943
Tanulás: D. Hebb, 1949
SNARC (Stochastic Neural Analog Reinforcement Calculator): M. Minsky, 1951
Perceptron: F. Rosenblatt, 1957
…
Mély neurális hálók, stb., 2006

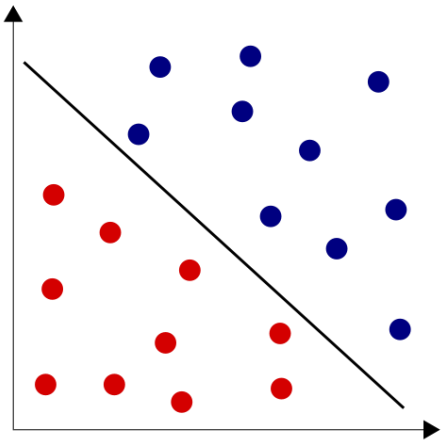
Perceptron

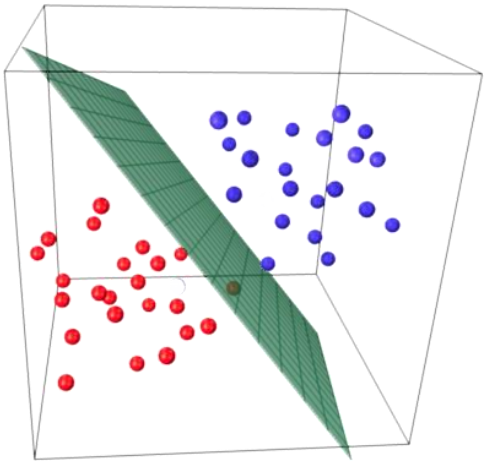
$$y = \mathrm{sgn}(\sum_{i=1}^{n+1} w_i \; x_i) = \mathrm{sgn}(\sum_{i=1}^{n} w_i \; x_i + w_{n+1}) = \mathrm{sgn}(\mathbf{x}^T \mathbf{w} + b)$$



$$\mathbf{x}^T \mathbf{w} + b > 0$$

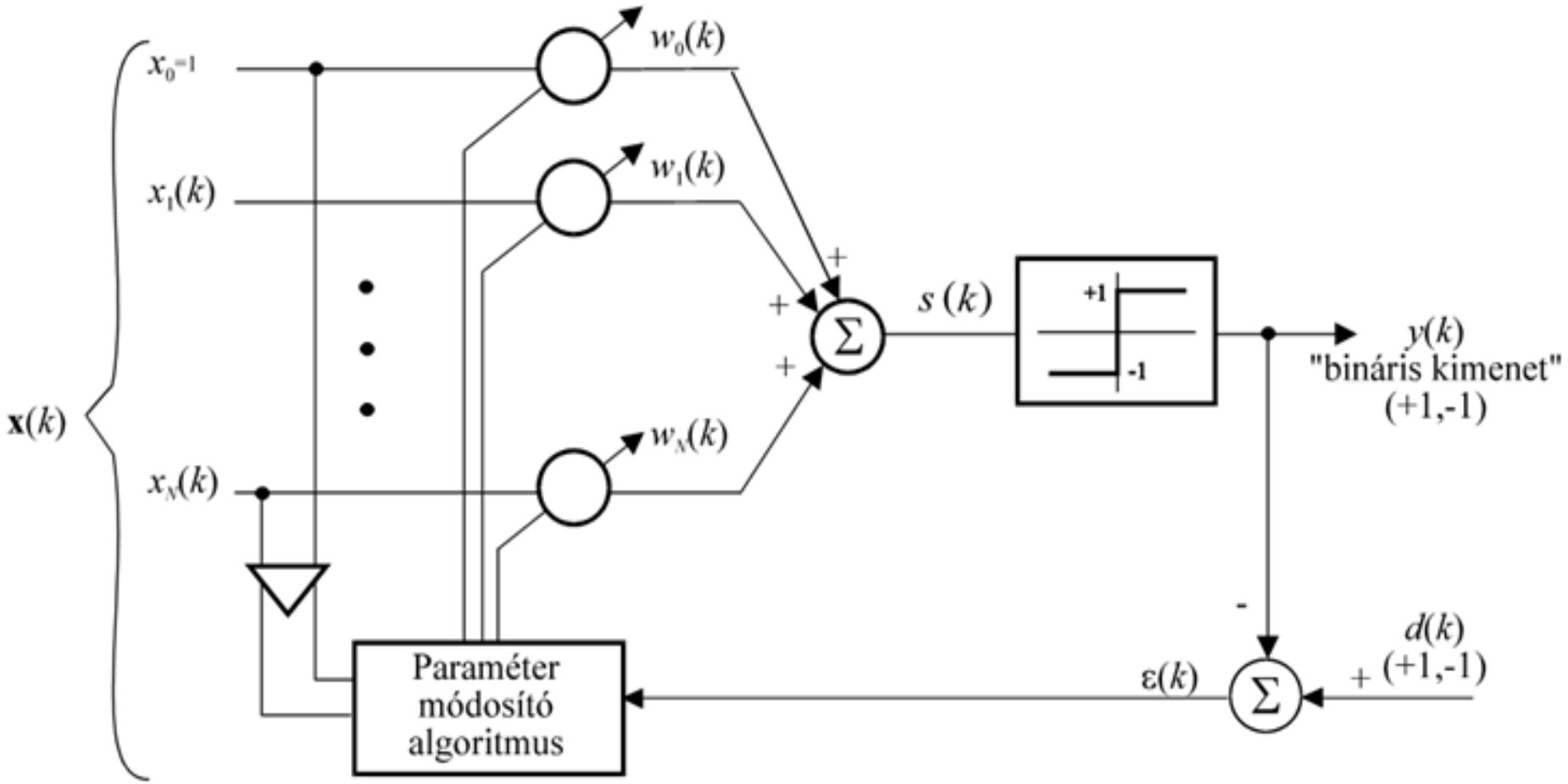
$$\mathbf{x}^T \mathbf{w} + b < 0$$





lineárisan (hipersíkkal) szeparáló függvényt képes megvalósítani

Perceptron tanítása



$$s(k) = \mathbf{w}^T\mathbf{x}(k) \qquad y(k) = \mathrm{sgn}(s(k)) \qquad \varepsilon(k) = d(k) - y(k)$$

$$\mathbf{w}_{új} \leftarrow \mathbf{w}_{régi} + \alpha \cdot \varepsilon \cdot \mathbf{x}$$

α - **bátorsági faktor,**
tanulási tényező

Perceptron tanítása

Első lelkesedés: ilyen egyszerű eszköz, és tanul!!! 1950-1960 körül.

A tanítás konvergens, ha

- a tanító példák lineárisan szeparálhatók
- a bátorsági tényező elegendően kicsi

Például: egy tanítási lépés a táblázatban adott minta alapján.

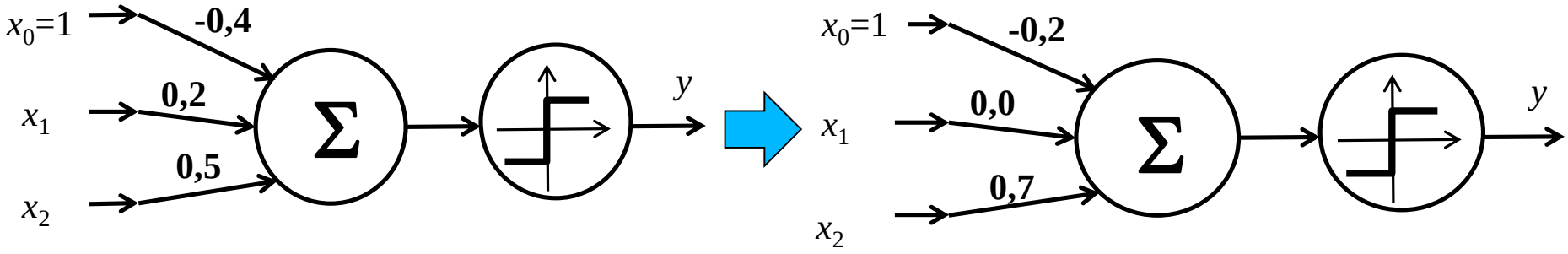
(x_1, x_2, *d*) a tanítóminta, *y* a tanítási lépés előtt kapott válasz.

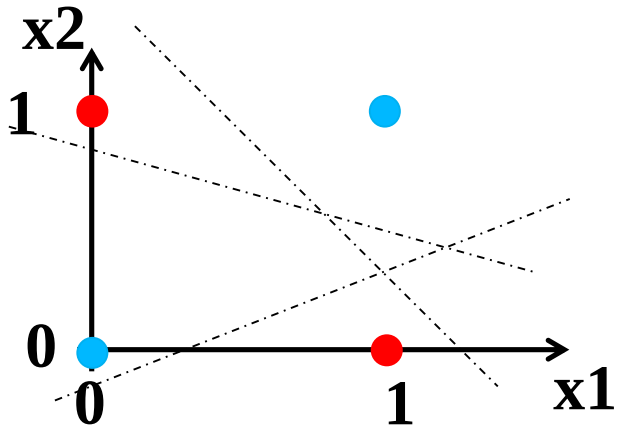
x_1	x_2	*d*	*y*
-1	1	1	-1

$$\alpha = 0{,}1 \qquad \varepsilon = d - y = 2$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \cdot \varepsilon \cdot \mathbf{x}$$

$$\begin{bmatrix} -0{,}2 \\ 0{,}0 \\ 0{,}7 \end{bmatrix} = \begin{bmatrix} -0{,}4 \\ 0{,}2 \\ 0{,}5 \end{bmatrix} + 0{,}1 \cdot 2 \cdot \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$



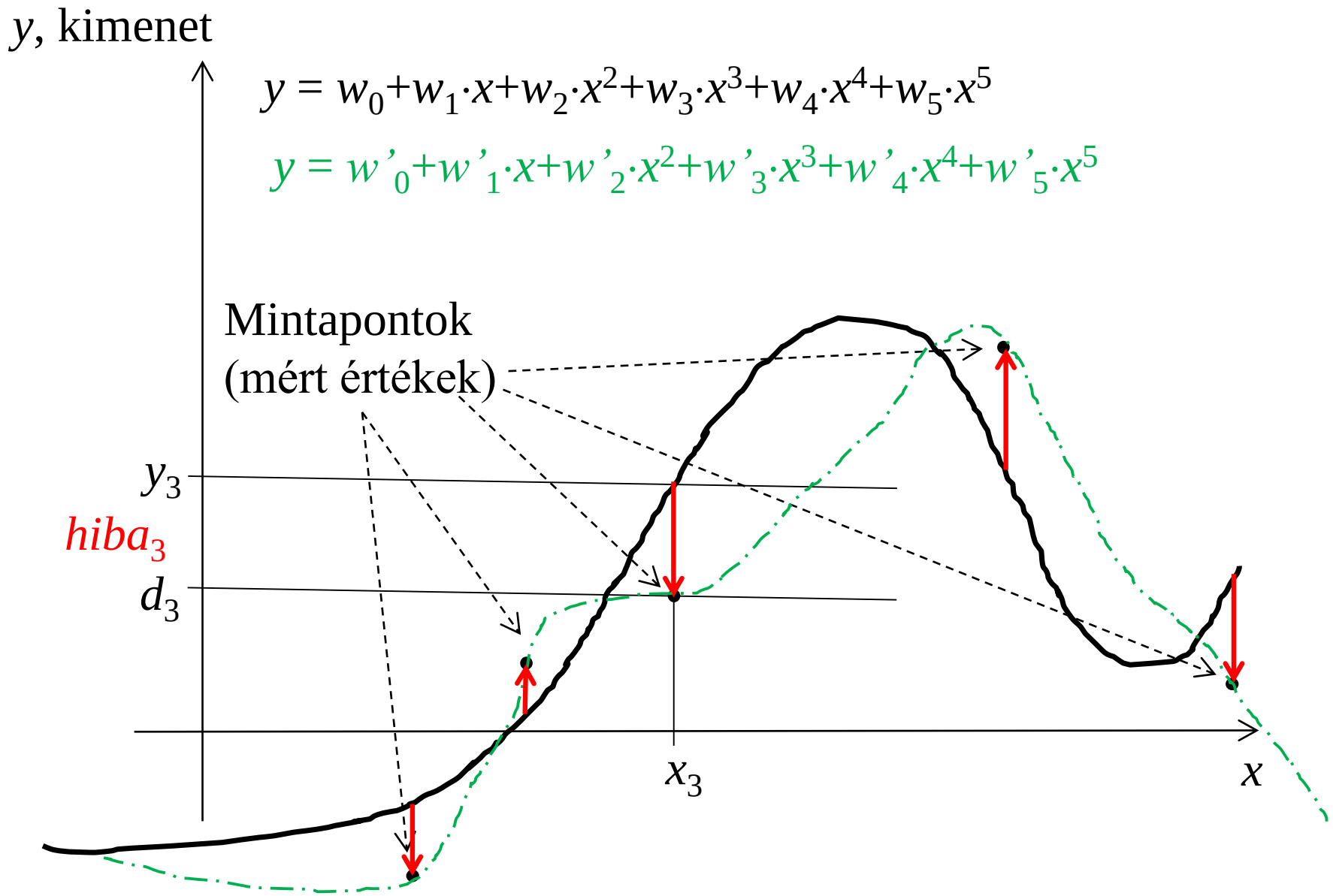


x1	x2	x1 XOR x2
0	0	0
0	1	1
1	0	1
1	1	0

XOR (és hasonlóan lineárisan nem szeparálhatók) problémák:

- egy lineáris szeparáló réteggel nem oldható meg
- több réteg kell
- DE a perceptrontanulás csak 1 rétegnél működik
  (*hiba értelmezése csak a kimeneten, nem tudható, hogy a több réteg több neuronjából melyik okozta a hibát*)

Tanítás gradiens alapon, egy példán keresztül mutatjuk be



Tanítás gradiens alapon

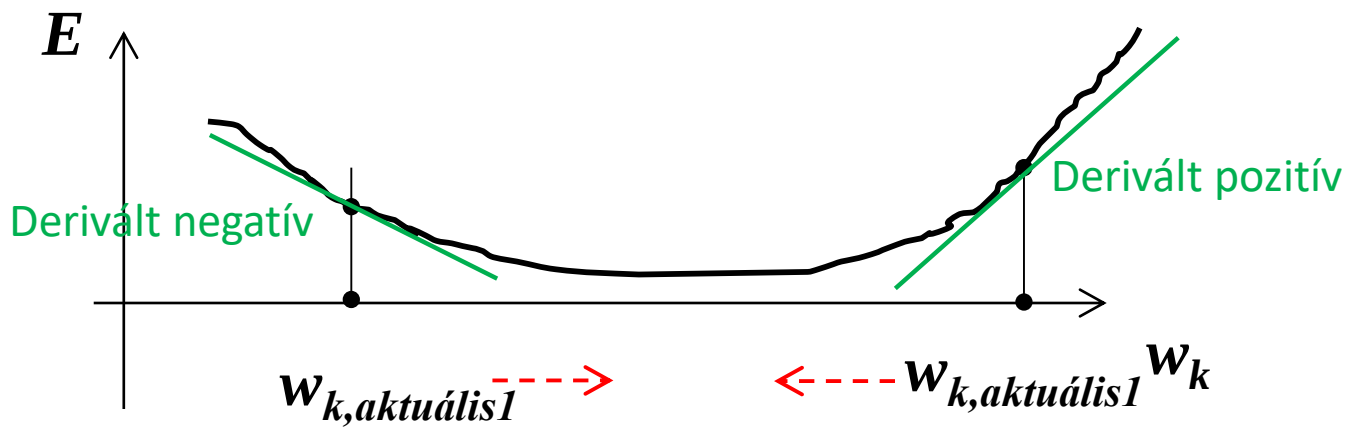
Csökkentsük az x_3,d_3 pontnál fellépő hibát, a paraméterek változtatásával

$$y_3 = w_0+w_1\cdot x_3+w_2\cdot x_3^2+w_3\cdot x_3^3+w_4\cdot x_3^4+w_5\cdot x_3^5$$

A hiba, amit csökkenteni szeretnénk: $E_3 = (d_3 - y_3)\wedge 2$

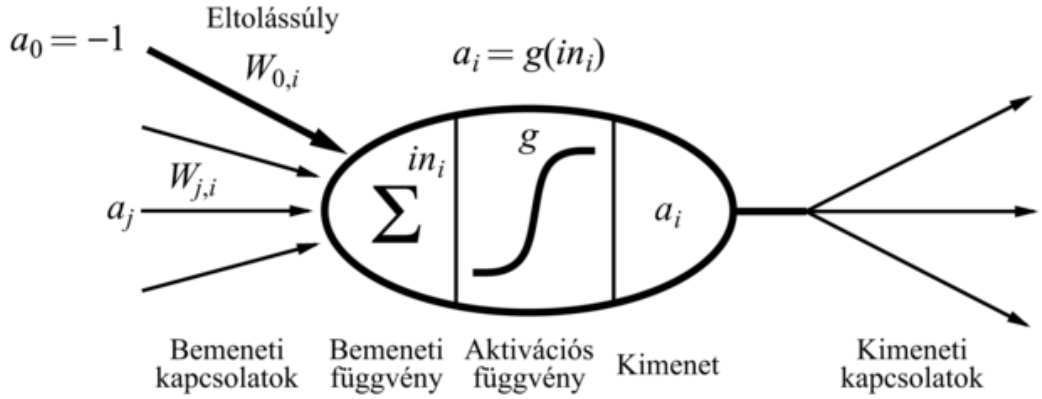
$$E_3 = (d_3-w_0-w_1\cdot x_3-w_2\cdot x_3^2-w_3\cdot x_3^3-w_4\cdot x_3^4-w_5\cdot x_3^5)^2$$

Vegyük észre, hogy itt most x_3 és d_3 állandó, és w_0,w_1,…,w_5 az, amelyeket változtatunk! Milyen irányban kell változtatnunk? Csökkentsük vagy növeljük az egyes paraméterek értékeit?

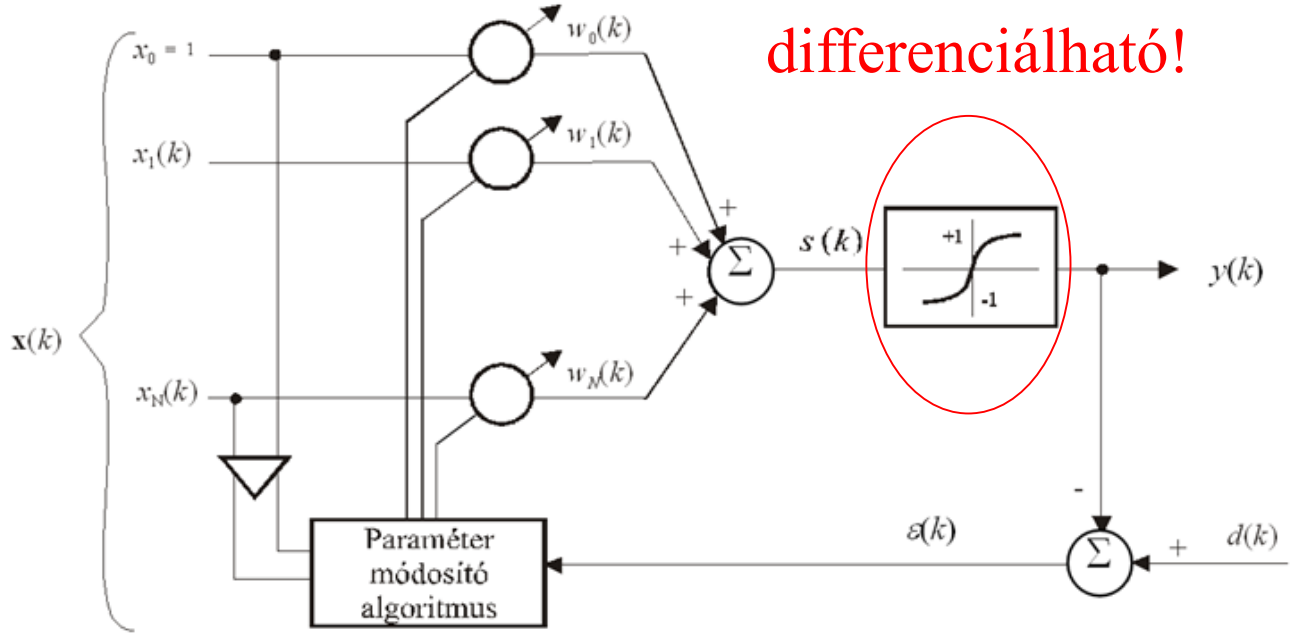


Módosítás: Egyszerű perceptron

→ **Mesterséges neuron**



differenciálható!



$s(k) = \mathbf{w}^T\mathbf{x}(k)$ $\quad y(k) = \mathrm{sgm}(s(k))$ $\quad \varepsilon(k) = d(k) - y(k)$

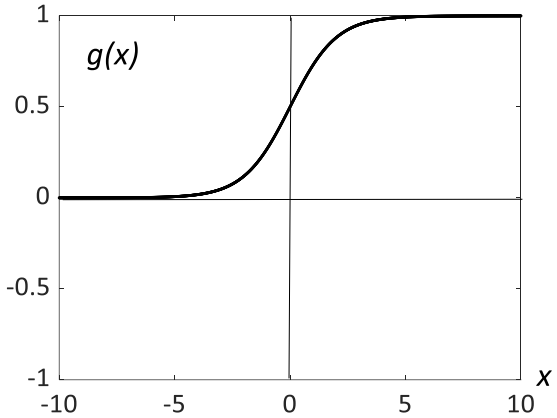
A többrétegű preceptronhálókban (Multilayer Perceptron: MLP) általában a szigmoid függvényt vagy annak valamelyik változatát használjuk. A derivált is nagyon egyszerűen számítható.

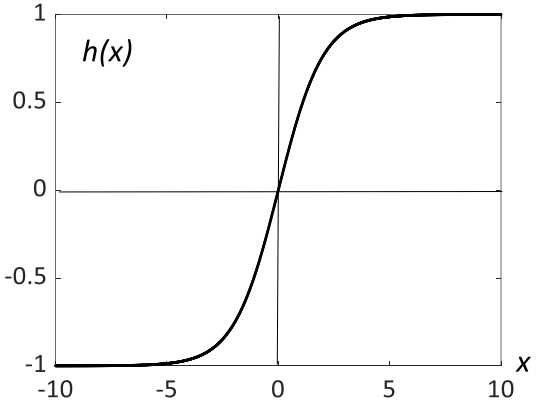
$$szigm(x) = g(x) = \frac{1}{1+e^{-x}}$$

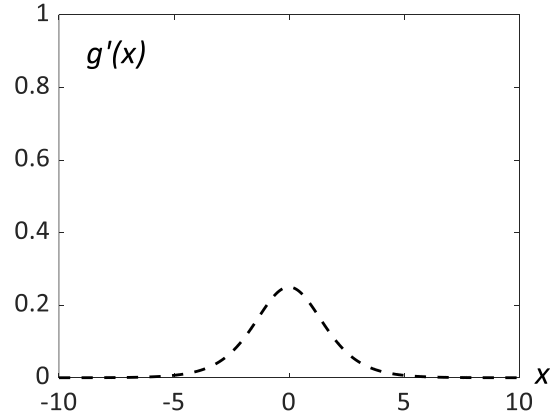
$$h(x) = 2 \cdot g(x) - 1 = \tanh(x/2) = \frac{1-e^{-x}}{1+e^{-x}}$$

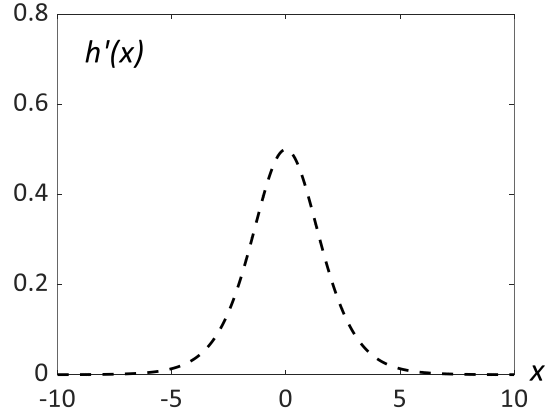
$$g'(x) = \frac{dg(x)}{dx} = g(x) \cdot (1 - g(x))$$

$$h'(x) = \frac{dh(x)}{dx} = \frac{1}{2} \cdot (1 + h(x)) \cdot (1 - h(x))$$

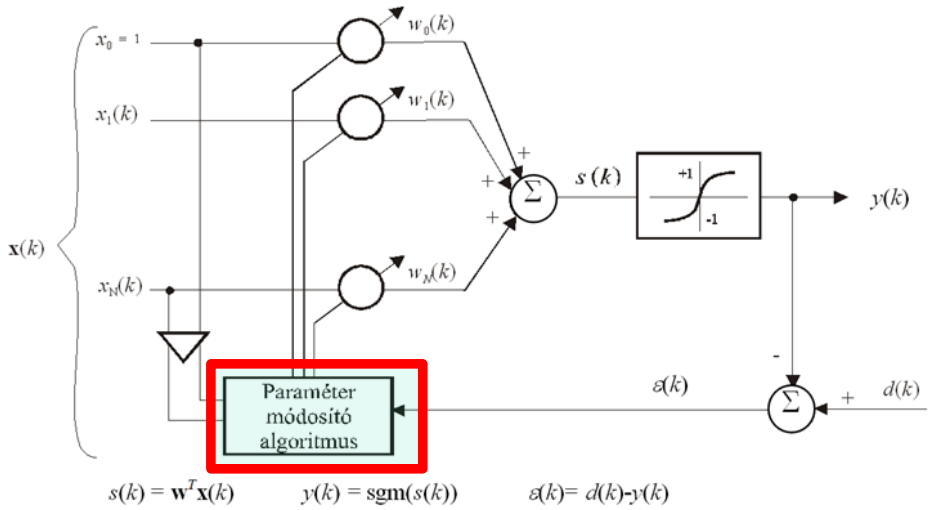








Egyetlen nemlineáris – szigmoid – neuron tanítása:



$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \frac{\partial E}{\partial W} \quad \text{ahol}$$

$$E = \varepsilon^2 = (d - y)^2$$

$$\varepsilon = d - y \qquad y = f(s) = szigm(s) \qquad s = \mathbf{w}^T\mathbf{x}$$

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial \varepsilon} \cdot \frac{d\varepsilon}{dy} \cdot \frac{dy}{ds} \cdot \frac{\partial s}{\partial \mathbf{W}} = \frac{\mathrm{d}E}{\mathrm{d}\varepsilon} \cdot \frac{d}{dy} \cdot \frac{d\, szigm(s)}{ds} \cdot \frac{\partial s}{\partial \mathbf{W}}$$

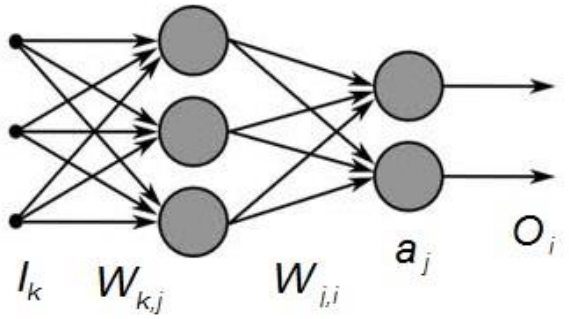
$$2\varepsilon \qquad -1 \qquad (1-y)\cdot y \qquad \mathbf{X}$$

$$\mathbf{W}_{\text{új}} = \mathbf{W}_{régi} - \alpha \cdot \frac{\partial \mathrm{E}}{\partial \mathbf{W}} = \mathbf{W}_{régi} - 2 \cdot \alpha \cdot \varepsilon \cdot (-1) \cdot (1-y) \cdot y \cdot \mathbf{X}$$
$$= \mathbf{W}_{régi} + 2 \cdot \alpha \cdot \varepsilon \cdot (1-y) \cdot y \cdot \mathbf{X}$$

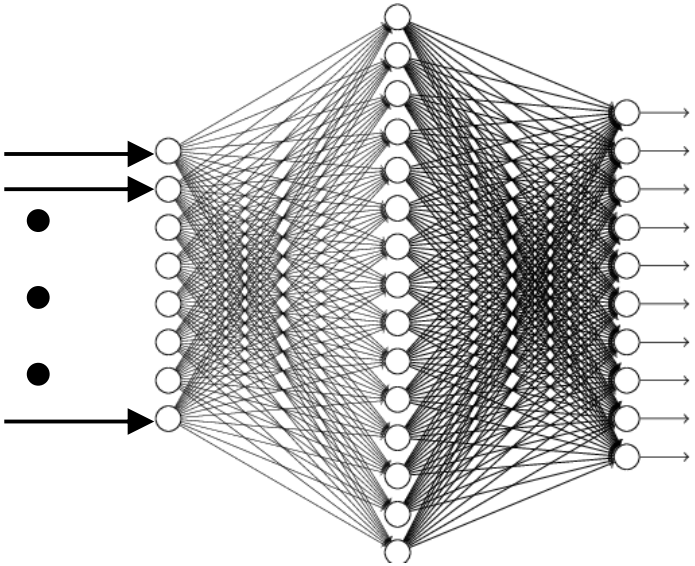
Mesterséges Neurális Háló (Artificial Neural Network)

Nemlineáris approximációt megvalósító, induktív tanulási algoritmussal tanítható matematikai struktú

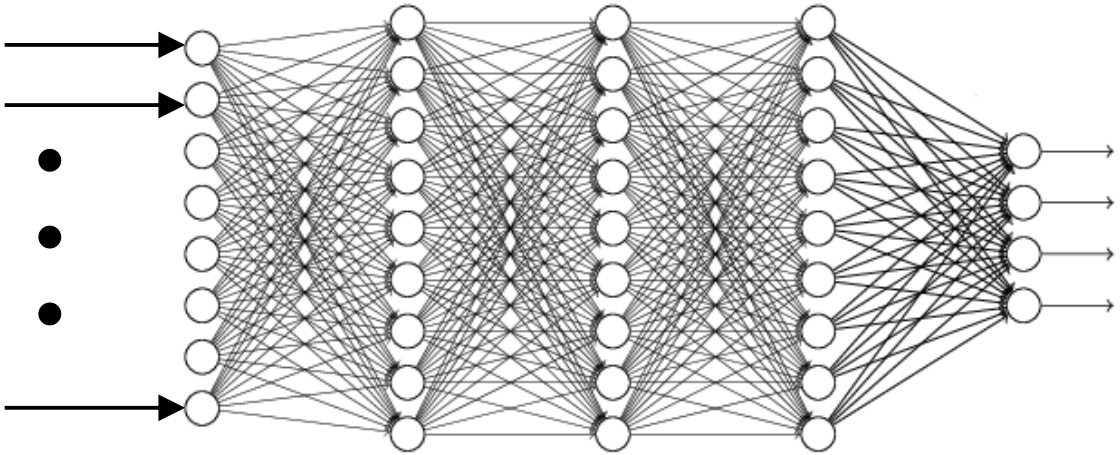
Az egyik legismertebb, az MLP:
Előrecsatolt, rétegek között teljesen összekötött



Egy rejtett réteg



Több rejtett réteg



Mesterséges Neurális Háló

Többrétegű előrecsatolt háló tanítása (elemi alapok)
Tanítás: Hibavisszaterjesztés (*backpropagation, BP*) gradiens módszerrel

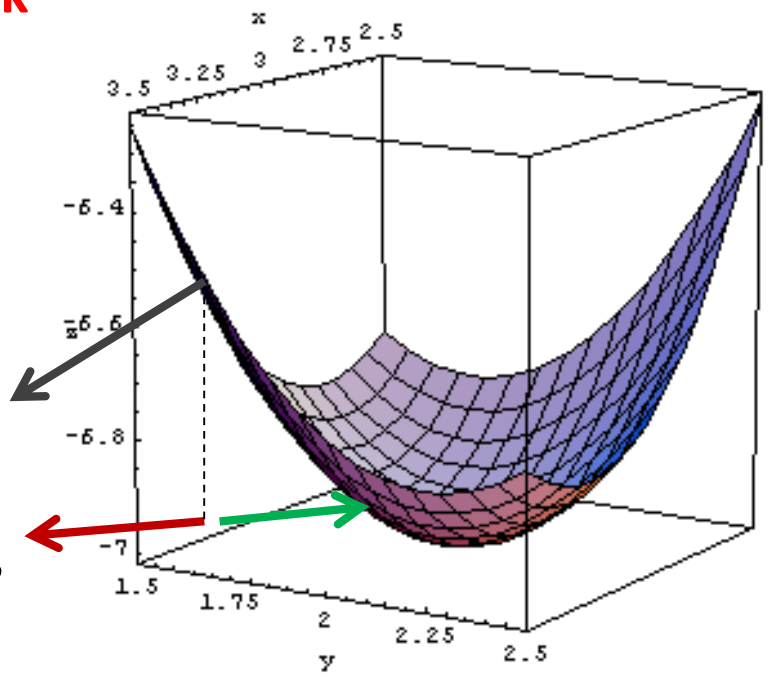
- példa: bemeneti mintákat és hozzájuk rendelt kívánt kimeneteket mutatunk a hálónak,
- ha hiba lép fel (a kimenet és a kívánt érték eltér), a súlyokat úgy módosítjuk, hogy a hiba csökkenjen.

A trükk a hiba megállapítása, és a hibának a hibát okozó súlyok közti szétosztása.

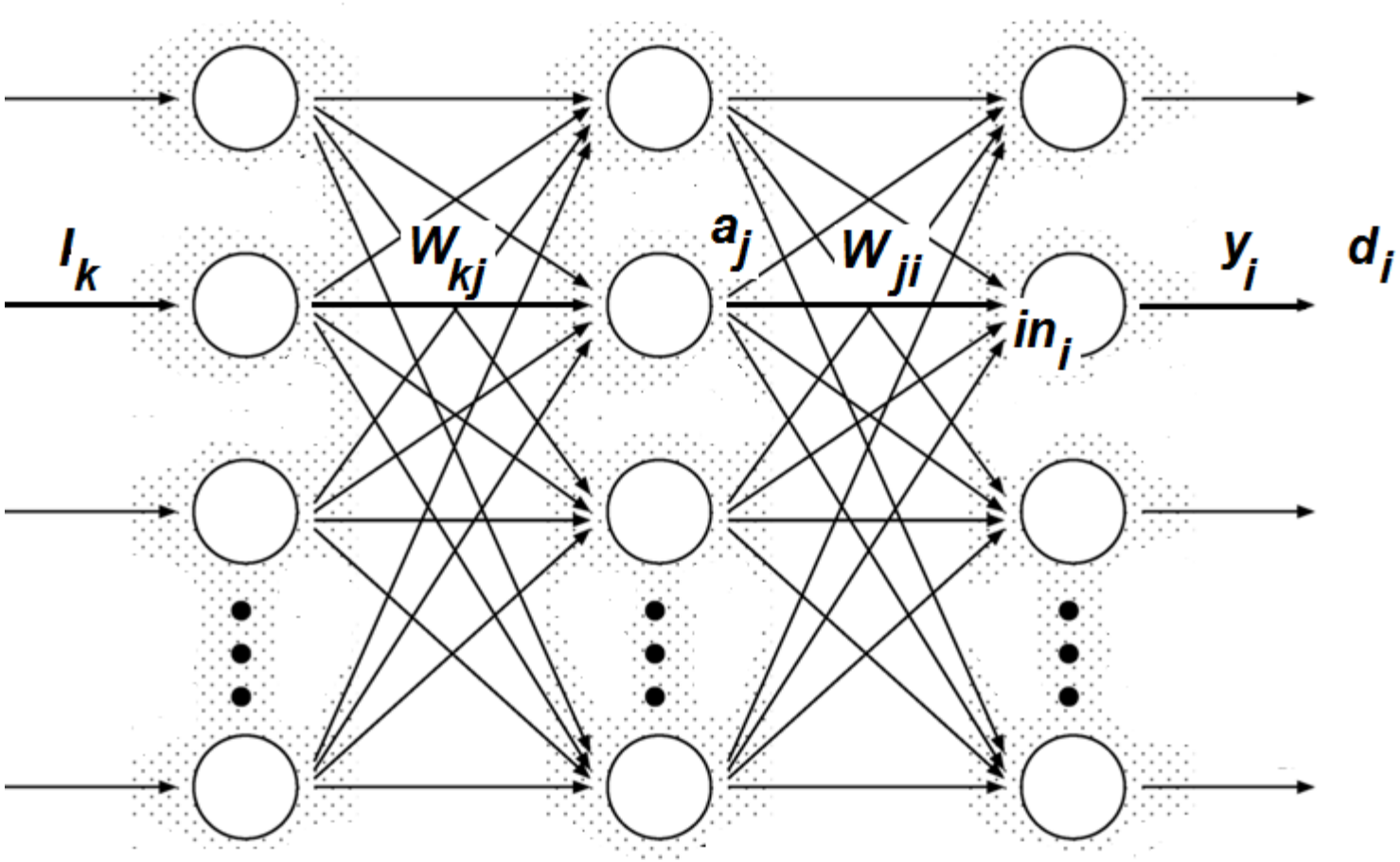
Gradiens módszer

$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \frac{\partial E}{\partial W}$$

A gradiens abba az irányba mutat, amelyik irányban a leggyorsabban nő a függvény (meredeken fel a hegyre…). Negatív gradiens: a meredek csökkenés iránya!



Mesterséges Neurális Háló



$$W_{k,j} \leftarrow W_{k,j} - \alpha \frac{\partial E}{\partial W_{k,j}}$$

$$W_{j,i} \leftarrow W_{j,i} - \alpha \frac{\partial E}{\partial W_{j,i}}$$

$$E = \frac{1}{2} \sum_i (d_i - y_i)^2$$

A hibavisszaterjesztés (backpropagation, BP) tanítási algoritmus bemutatása a Neurális hálózatok (Altrichter-Horváth-Pataki-Strausz-Takács-Valyon, Panem 2005) könyv alapján részletesen megismerhető.

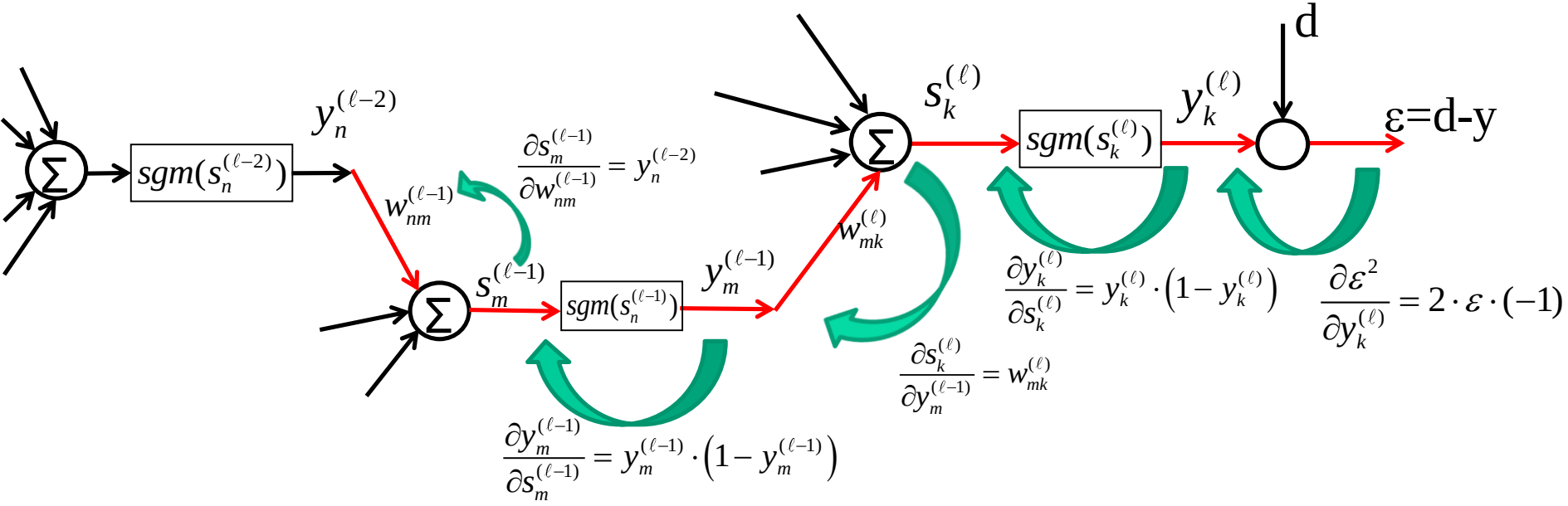
A következő dián azt mutatjuk be, hogy a kimeneti hibát hogyan lehet visszavezetni egy belső súlyig.
Cél annak kiszámítása, hogy a vizsgált súly hogyan hat a kimeneti hibára. (differenciálhányados!)

$$\frac{\partial E}{\partial w_{nm}^{(\ell-1)}} = \frac{\partial \varepsilon^2}{\partial w_{nm}^{(\ell-1)}} = \frac{\partial \varepsilon^2}{\partial y_k^{(\ell)}} \cdot \frac{\partial y_k^{(\ell)}}{\partial s_k^{(\ell)}} \cdot \frac{\partial s_k^{(\ell)}}{\partial y_n^{(\ell-1)}} \cdot \frac{\partial y_n^{(\ell-1)}}{\partial s_n^{(\ell-1)}} \cdot \frac{\partial s_n^{(\ell-1)}}{\partial w_{nm}^{(\ell-1)}}$$

Tanítás: hibavisszaterjesztés (backpropagation) algoritmus

A deriválás láncszabálya:

$$\frac{\partial \varepsilon^2}{\partial w_{nm}^{(\ell-1)}} = \frac{\partial \varepsilon^2}{\partial y_k^{(\ell)}} \cdot \frac{\partial y_k^{(\ell)}}{\partial s_k^{(\ell)}} \cdot \frac{\partial s_k^{(\ell)}}{\partial y_n^{(\ell-1)}} \cdot \frac{\partial y_n^{(\ell-1)}}{\partial s_n^{(\ell-1)}} \cdot \frac{\partial s_n^{(\ell-1)}}{\partial w_{nm}^{(\ell-1)}}$$



$$E = \varepsilon^2 = \left(d - y_k^{(\ell)}\right)^2$$

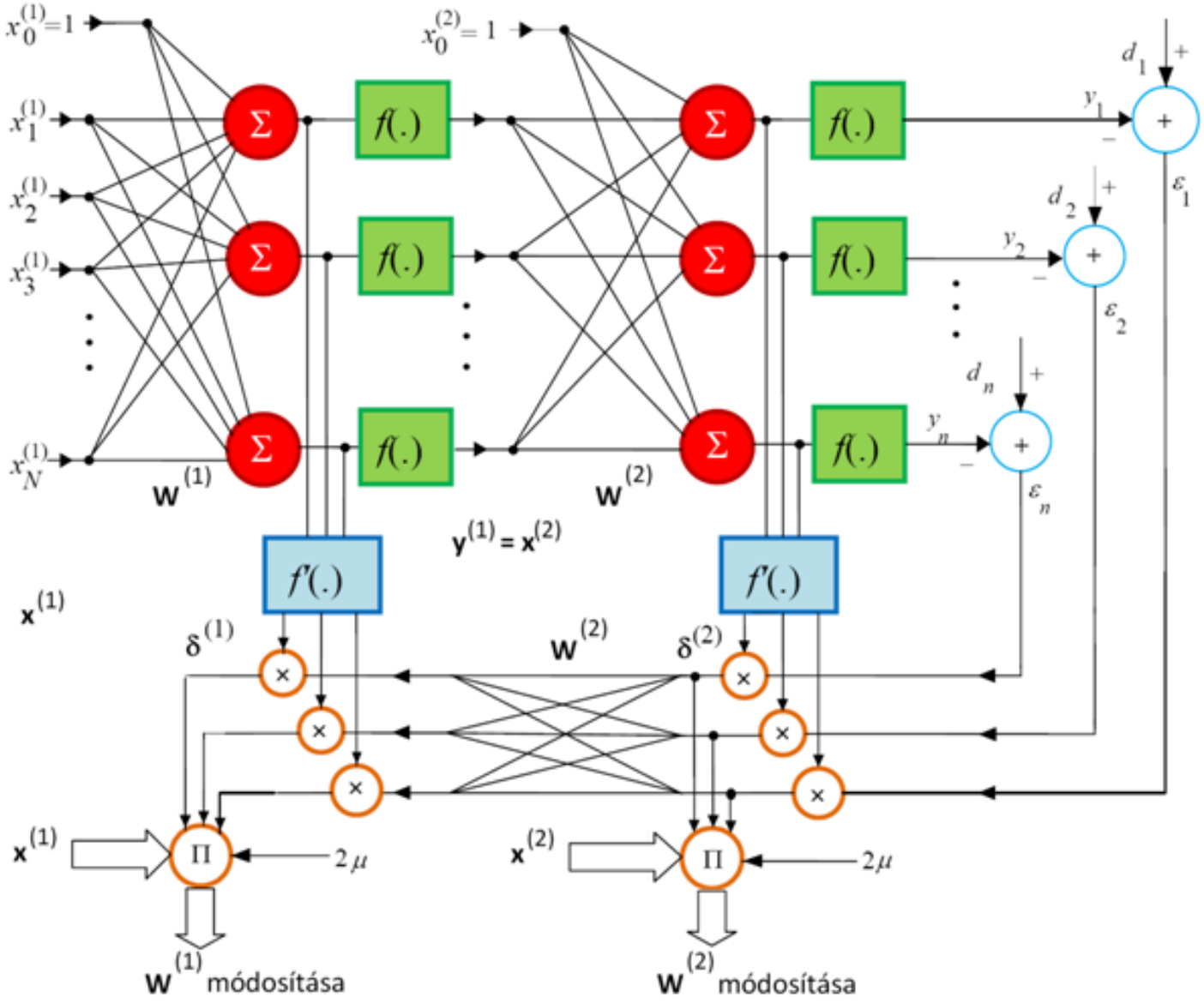
$$y_n^{(\ell-1)} = sigmoid\left(s_n^{(\ell-1)}\right)$$

$$y_k^{(\ell)} = sigmoid\left(s_k^{(\ell)}\right)$$

$$s_n^{(\ell-1)} = w_{0n}^{(\ell-1)} \cdot 1 + w_{1n}^{(\ell-1)} y_1^{(\ell-2)} + \ldots + w_{nm}^{(\ell-1)} y_n^{(\ell-2)} + \ldots.$$

$$s_k^{(\ell)} = w_{0k}^{(\ell)} \cdot 1 + w_{1k}^{(\ell)} y_1^{(\ell-1)} + \ldots + w_{mk}^{(\ell)} y_m^{(\ell-1)} + \ldots. + w_{Mk}^{(\ell)} y_m^{(\ell-1)}$$

MLP Mesterséges Neurális Háló



http://mialmanach.mit.bme.hu/neuralis/index

Hibavisszaterjesztés (backpropagation)

A *t*-dik iterációs lépésben használt iteratív összefüggéspár az *l*-dik réteg *i*-dik neuron *j*-dik súlyának tanításánál
(A bátorsági faktor: α)

Látható, ahogy a δ általánosított hibát terjesztjük visszafele az *l*+1-dik réteg minden neuronjáról az *l*-dik réteg *i*-dik neuronjához a kettőt összekötő ($w_{ri}^{(\ell+1)}$) súlyon keresztül

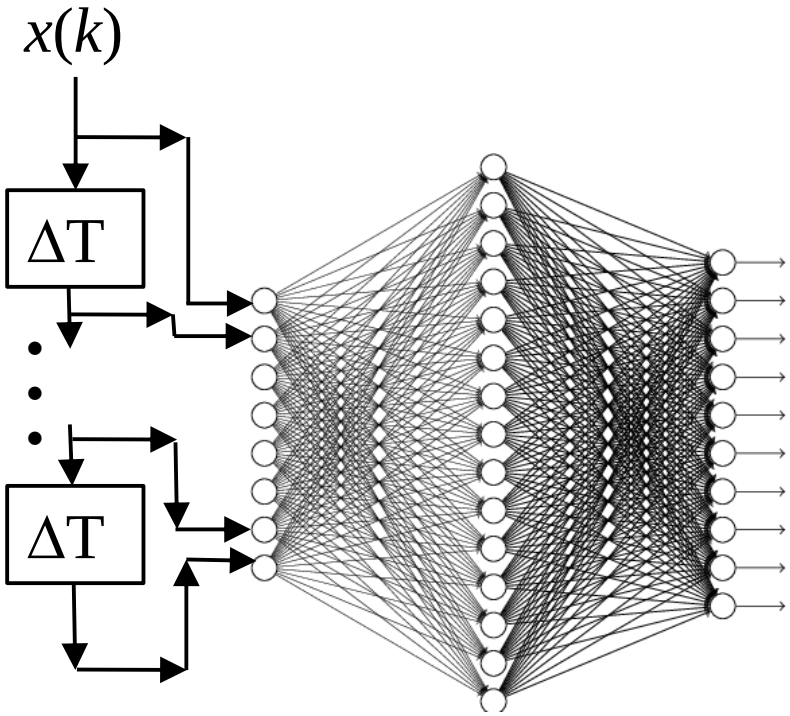
$$\delta_i^{(\ell)}(t) = \sum_{r=1}^{N_{\ell+1}} \delta_r^{(\ell+1)}(t) \cdot w_{ri}^{(\ell+1)}(t) \cdot f'(s_i^{(\ell)}(t))$$

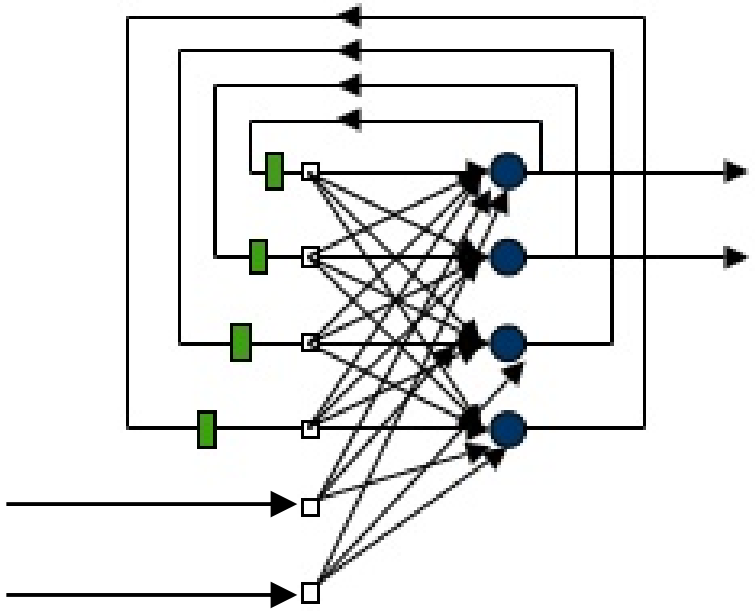
$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) + 2 \cdot \alpha \cdot \delta_i^{(\ell)}(t) \cdot x_j^{(\ell)}(t)$$

Időfüggő neurálisháló-variánsok

Idősorok modellezése, jóslása (predikció) (FIR-IIR)

- kiegészítés tárolóelemmel
- visszacsatolás





Mesterséges Neurális Hálók approximációs képessége

D. Hilbert (1900) 23 matematikai problémája/sejtése:
13. probléma: „Bizonyítsuk be, hogy az $x^7+ax^3+bx^2+cx+1=0$ hetedfokú egyenlet nem oldható meg pusztán kétváltozós függvények segítségével!” ?☺

…

„Mutassuk meg, hogy van olyan háromváltozós folytonos függvény, mely nem írható fel véges számú kétváltozós folytonos függvény segítségével!”

…

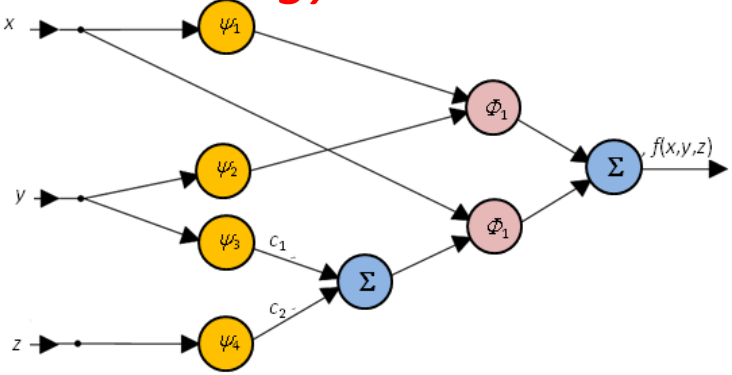
A. Kolmogorov (1957): nem csupán minden háromváltozós függvény, hanem tetszőleges *N*-változós folytonos függvény felírható csupán **egy**változós függvények és az összeadás segítségével.

…

A neuronháló bizonyos feltételekkel bármilyen függvényt tud tetszőleges pontossággal approximálni! (*második nagy lelkesedés 1970-1980 körül*)

http://mialmanach.mit.bme.hu/neuralis/ch01s04

$$f(x_1,x_2,\ldots,x_N)=\sum_{q=1}^{2N+1}\Phi_q\left(\sum_{p=1}^{N}\psi_{pq}(x_p)\right)$$

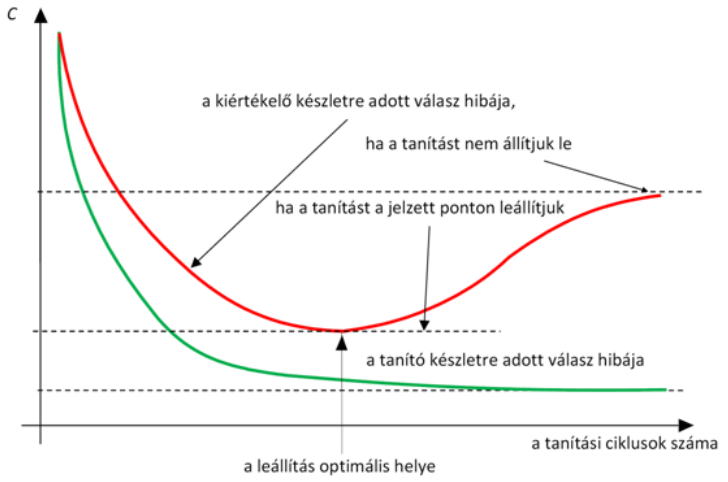


Mesterséges Neurális Háló

Kérdések, problémák:

- mekkora (hány réteg, rétegenként hány processzáló elem) hálózatot válasszunk?
- hogyan válasszuk meg a tanulási tényező (α) értékét?
- milyen kezdeti súlyértékeket állítsunk be?
- hogyan válasszuk meg a tanító és a tesztelő minta készletet?
- hogyan használjuk fel a tanítópontokat, milyen gyakorisággal módosítsuk a hálózat súlyait?
- meddig tanítsuk a hálózatot, stb?
- hogyan védekezzünk a túltanulással szemben?
- (hogyan gyorsítsuk meg a tanulást?)

http://mialmanach.mit.bme.hu/neuralis/index



Elemzés

Kifejezőképesség: A többrétegű hálók osztálya egészében mint osztály a bemenetek (attribútumok) bármely függvényének reprezentációjára képes.

Számítási hatékonyság: legrosszabb esetben a tanításhoz szükséges epochok (tanítási menetek: az összes mintát felhasználjuk egy-egy epochban) száma a bemenetek számának, *n*-nek, még *exponenciális függvénye* is lehet! A hibafelület *lokális minimumai* problémát jelenthetnek.

Általánosító képesség: jó általánosításra képesek.

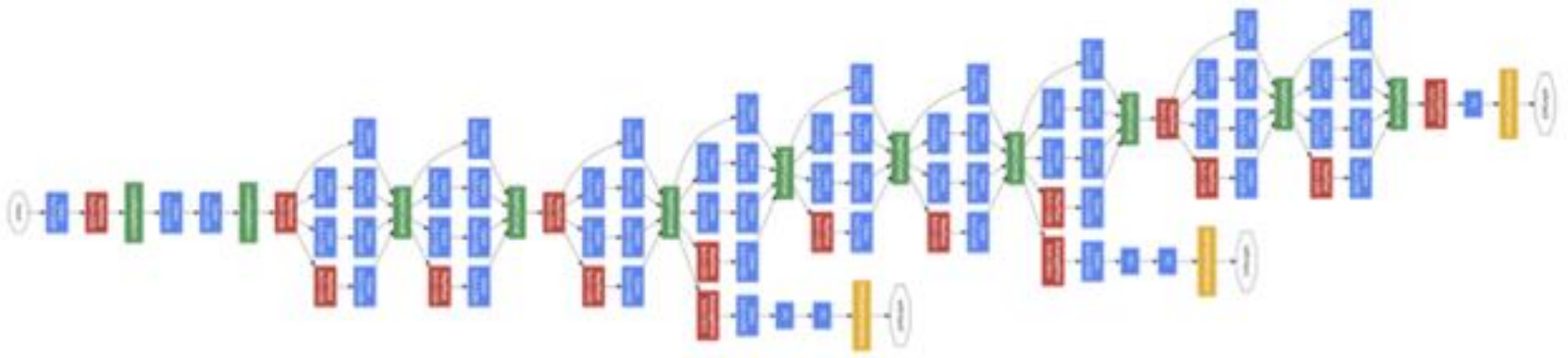
Zajérzékenység: alapvetően nemlineáris regresszió, nagyon jól tolerálják a bemeneti adatok zajosságát (*ha elegendő mintánk van*!).

Átláthatóság: alapvetően fekete doboz, a működése nem átlátható.

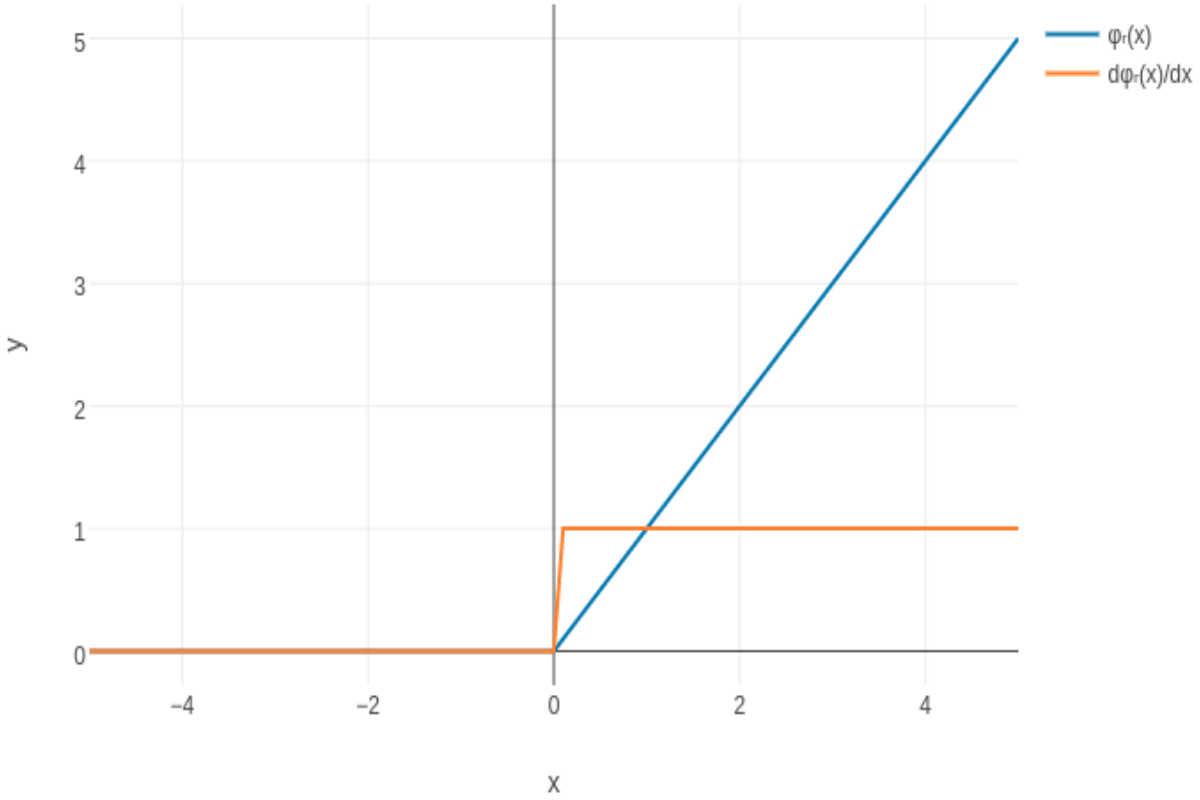
Mesterséges neurális hálók 1990-es évek fellendülés – majd a problémák miatt apály. 2005-2010-től újabb lelkesedés ⇒ **mély neurális hálók**. (Elsősorban a képfeldolgozás területén hozott áttörést, azóta más területeken is erősen kutatják.)

Néhány karakterisztikus változás

1. Jóval nagyobb rétegszám! (1990-es évekig 3-4-5 ritkán megy fel 10 rétegig.) Most **150-200-300-…** rétegű hálók! (Ezért is nevezzük mélynek.) – Nagyrészt a hardver fejlődés tette lehetővé (memória, GPU-k megjelenése stb.). Számítási gráf:



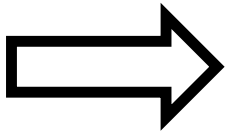
1. Mivel a gradiens (derivált) a szigmoidnál 1-nél kisebb, a hibavisszaterjesztés során a sok rétegen visszaterjesztve 0-hoz tart (összeszorzódnak). → új nemlinearitást vezettek be a szigmoid helyett. ReLu (Rectified Linear).



(Kisebb továbbfejlesztések: a negatív szakasz jóval kisebb meredekséggel, de nem nulla, az origóban a derivált nulla stb.)

2. Dimenzióredukció – (max, min, avg) pooling

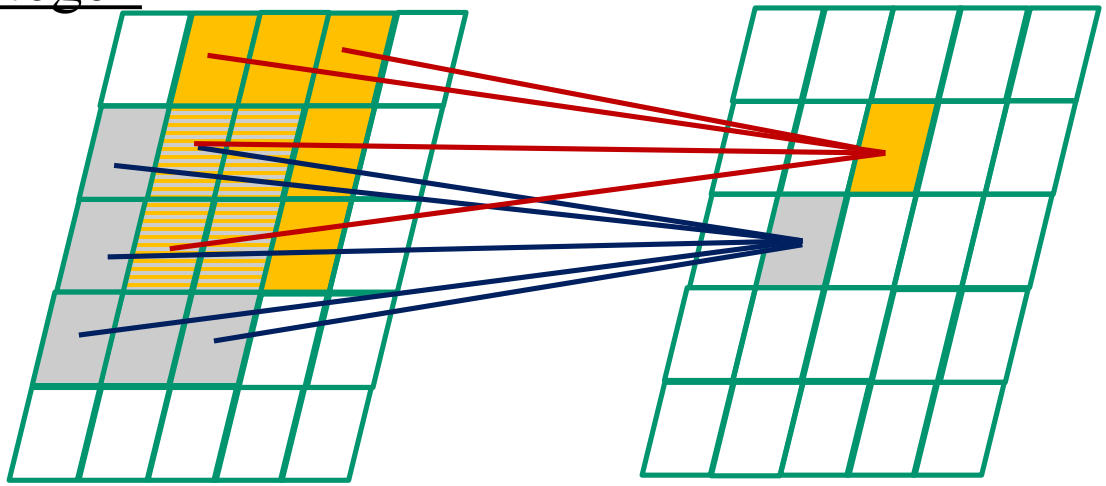
2	5	7	1
1	3	2	2
9	3	3	2
3	1	8	5



5	7
9	8

(itt az ablakok nem fednek át, 2*2-es ablak és lépésköz (stride) =2)

3. Konvolúciós rétegek



Az összes konvolúció súlyai (az ábrán 9) azonosak, együtt változnak a tanítás során! Hibavisszaterjesztés mindegyiknél, de az összes kialakuló gradiens eredőjét használjuk.

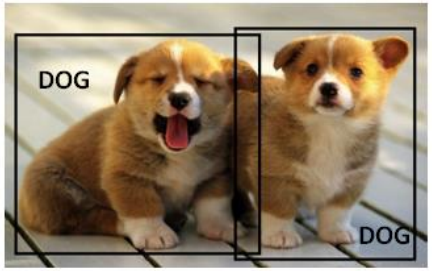
4. Dropout

Alapvetően a túltanulás ellen szolgál, nagyon érdekes módon. A tanítás során egyes aktivációkat valamilyen valószínűséggel nullára állít egy-egy tanítási menetben. A háló redundanciára kényszerül, és egyik dropout konfigurációban (néhány véletlenszerűen választott aktivitás nulla) sem tud nagyon rátanulni a mintákra.

5. Augmentáció (nagyobbítás)

Az óriási hálókhoz rengeteg minősített kép/képrészlet kell! (Túltanulás!) → Augmentáció: az alapképből eltolással, elforgatással, aszimmetrikus torzítással stb., újabb tanítómintákat hozunk létre. Nekünk pl. egy 1 pixellel eltolt kép észrevehetetlen változást jelent. A számítógépnek egészen más kép! Ezen (is) segít az augmentáció.

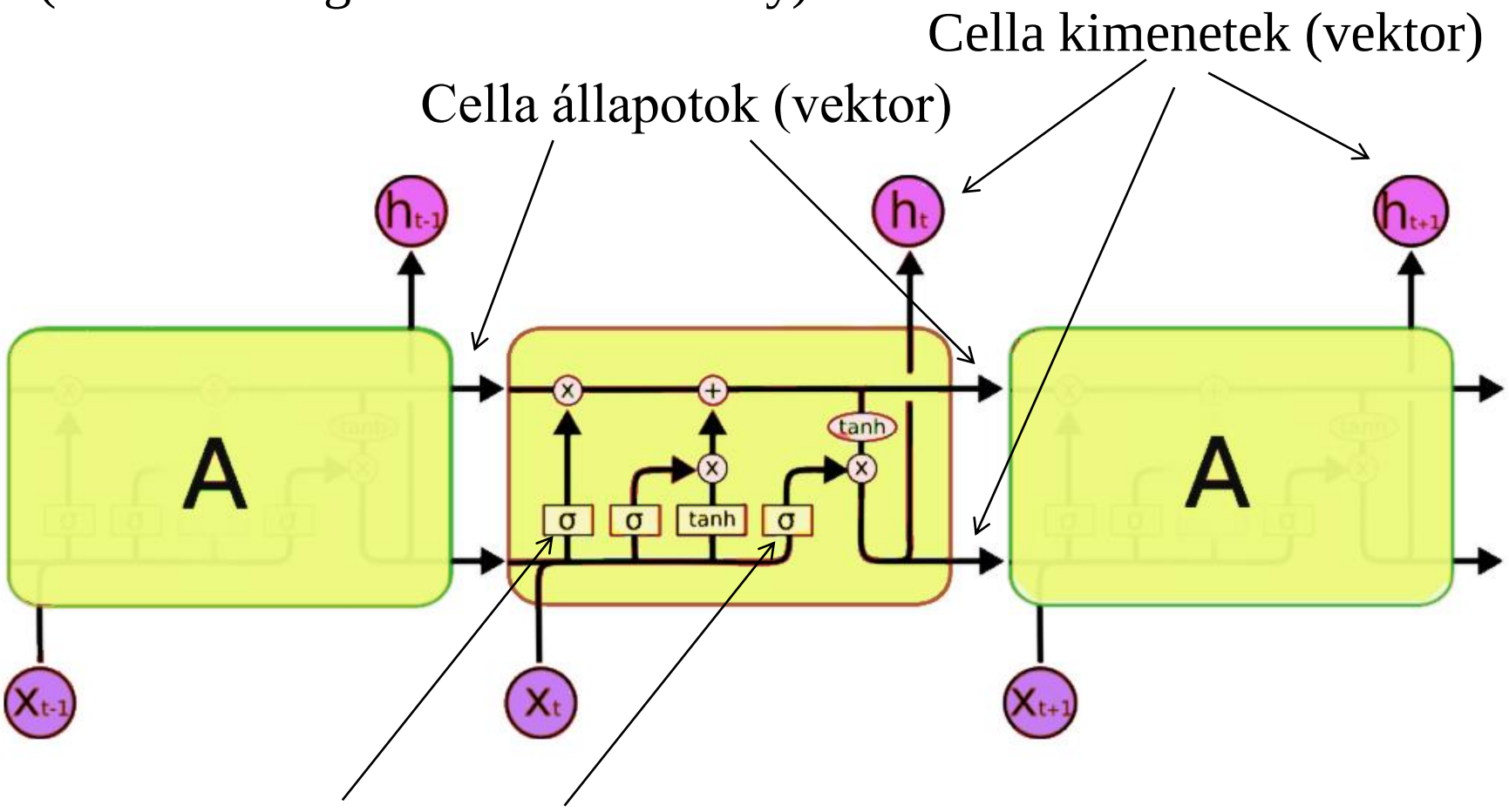
(A google képfelismerőjét kb. 10 millió minősített képpel tanították.)



Object Detection involves localization of multiple objects (doesn't have to be the same class).

Object Segmentation involves the class label as well as an outline of the object in interest.

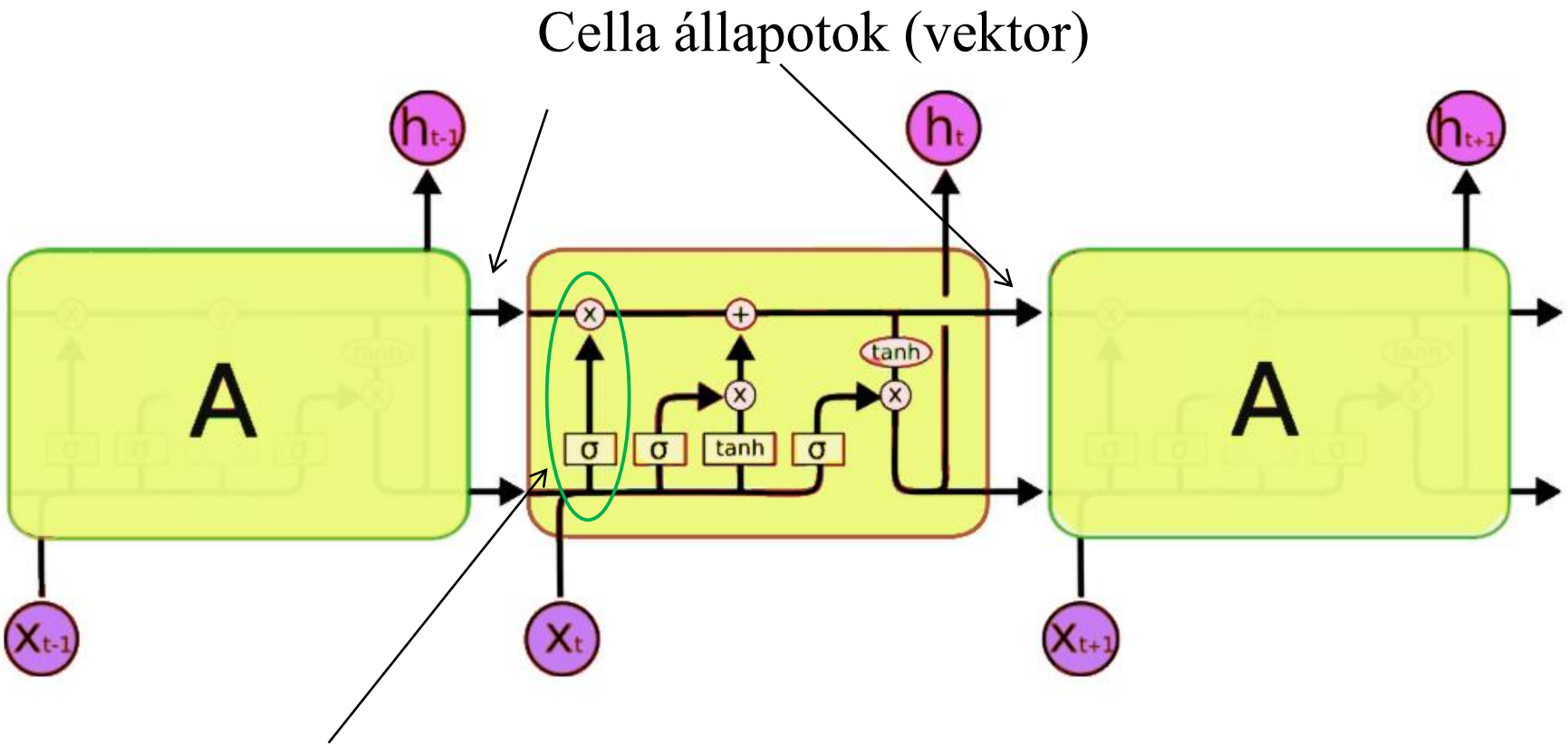
6. Időfüggő problémákra kidolgozott neuronstruktúra
(LSTM: Long short term memory)



Szigmoid neurális rétegek
0…1 kimenet: súlyozza (kapuzza)
az információ továbbítást

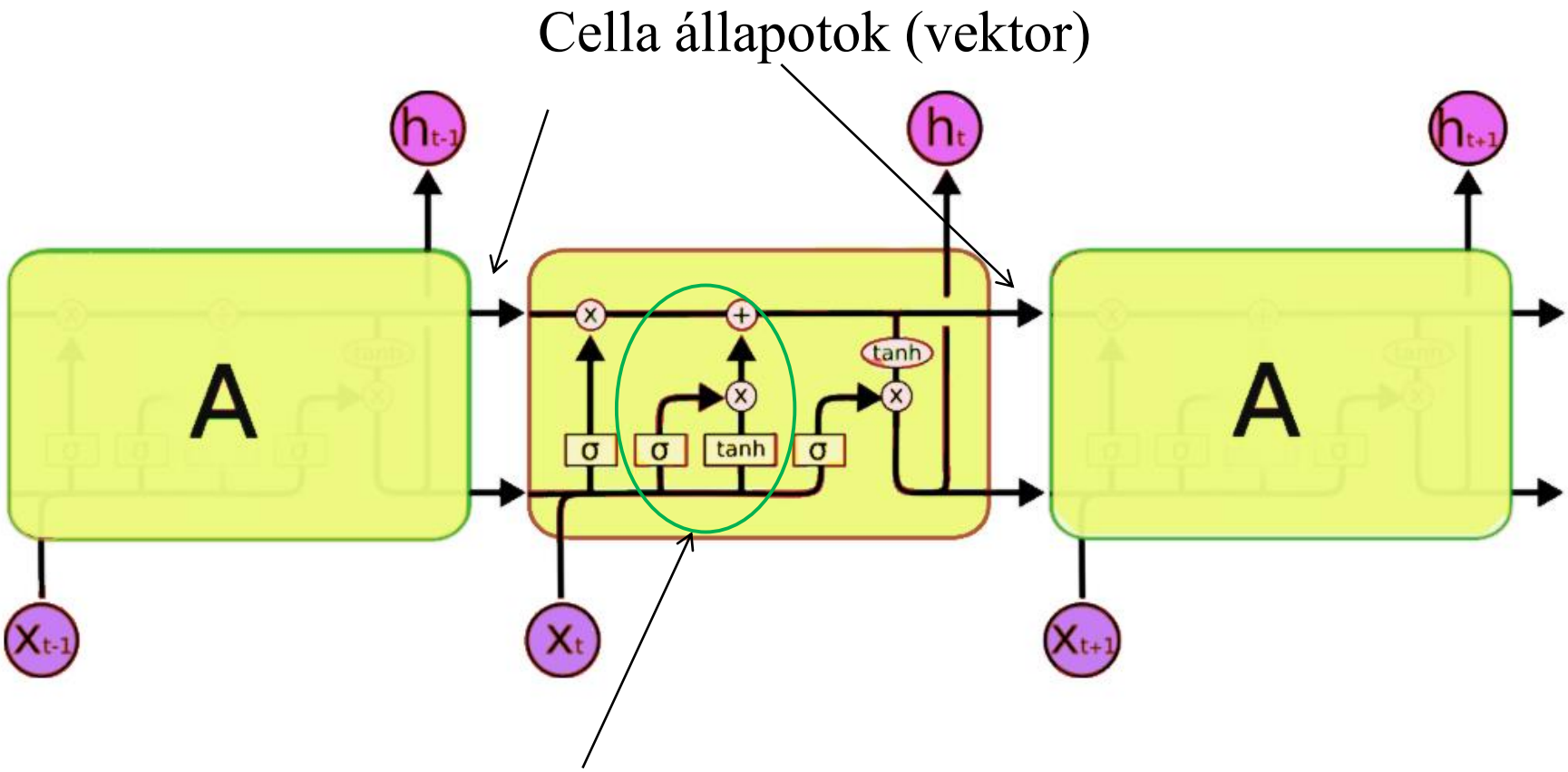
x_t és h_{t-1} konkatenálva,
szorzás és összeadás
pontonkénti

6. LSTM (Long Short Term Memory)



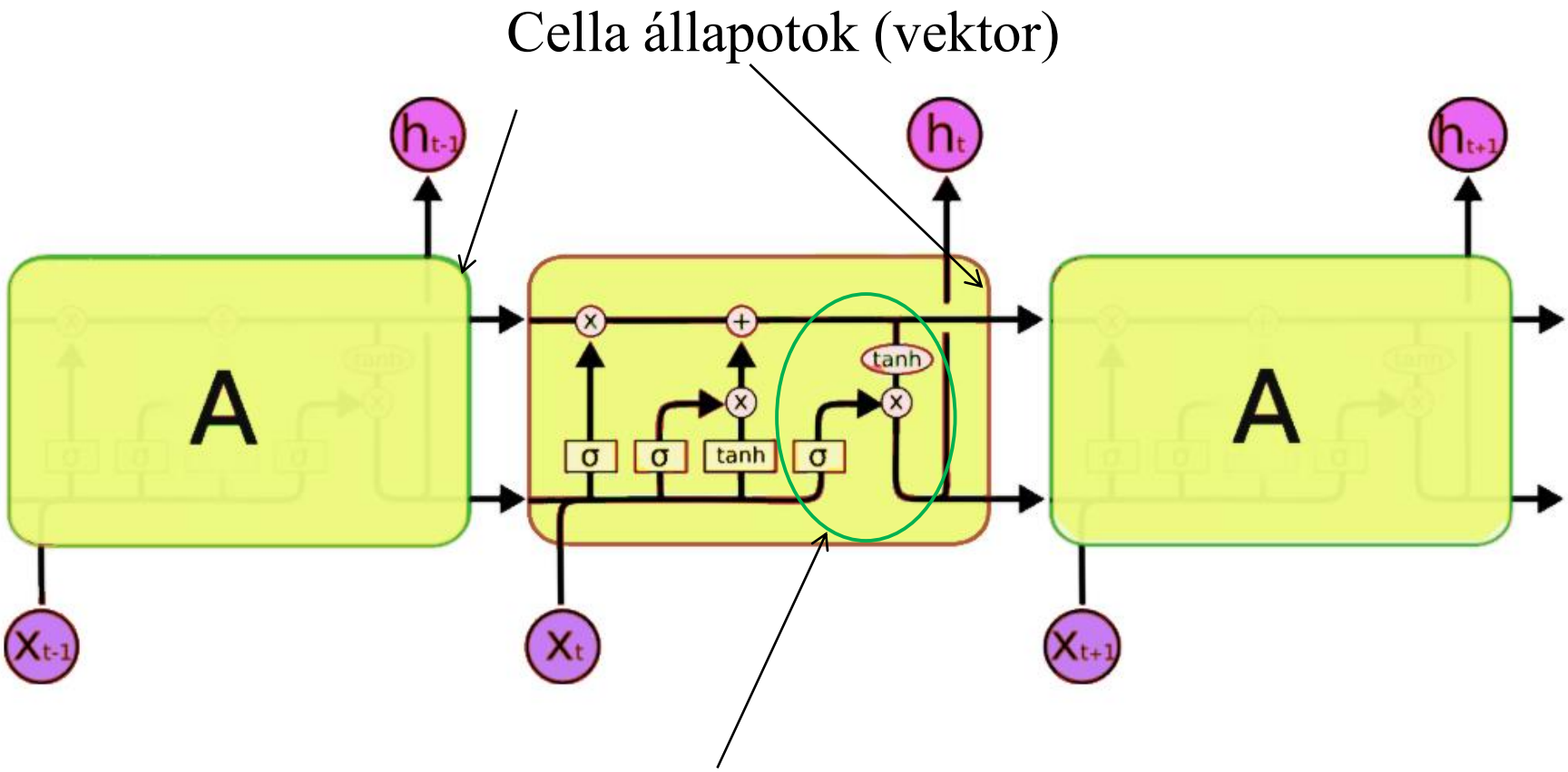
Felejtés:
az előző cellaállapotból mit és mennyire akarunk elfelejteni

6. LSTM (Long Short Term Memory)



Az új cellainformáció kialakítása (tanh) és annak súlyozása (szigm) az új cellainformációba

6. LSTM (Long Short Term Memory)



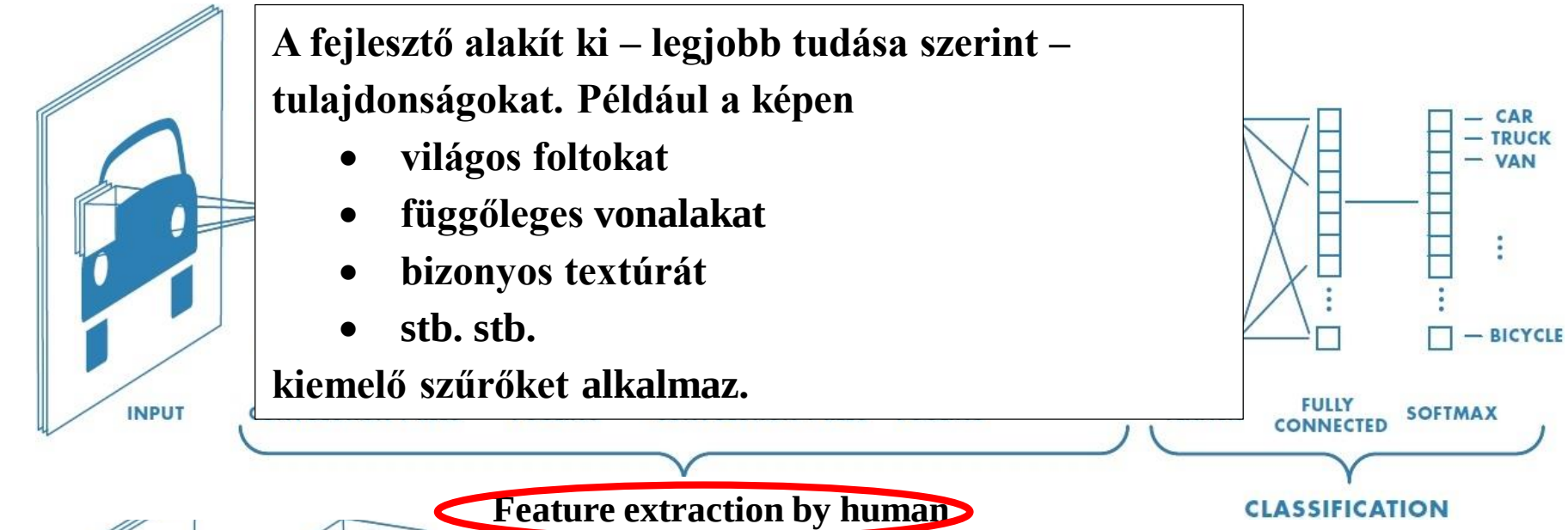
Az adott cella kimenetének kialakítása (tanh) és annak súlyozása (szigm) a kimenetre

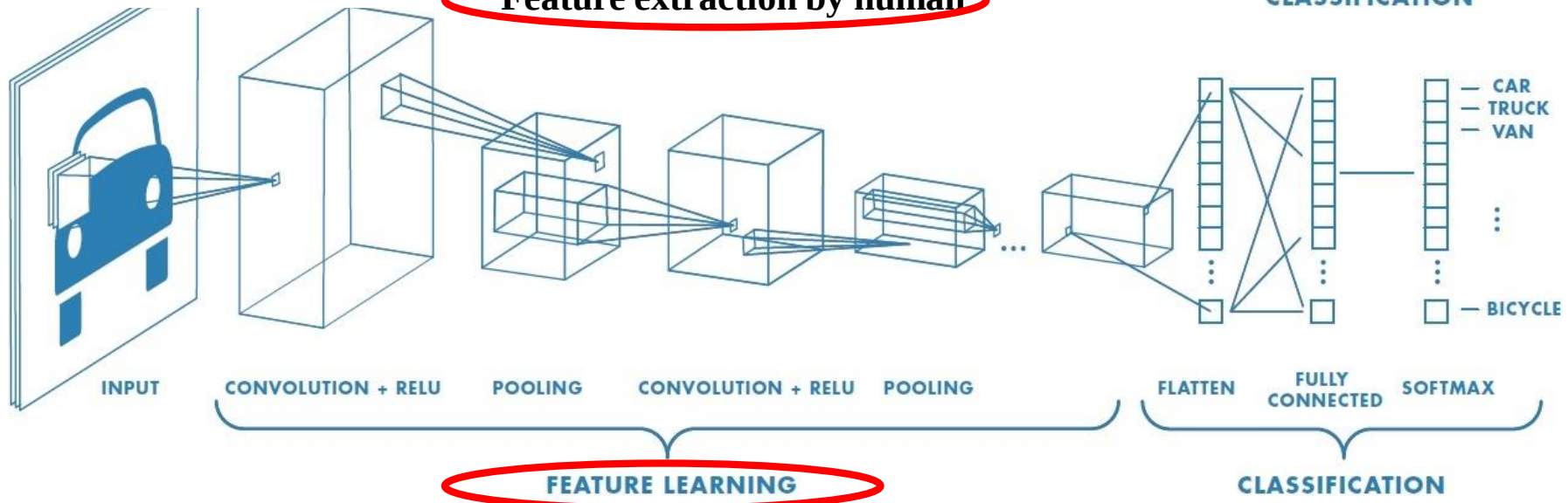
A „klasszikus” és a „mély” neuronhálók közti koncepcionális különbség:

A fejlesztő alakít ki – legjobb tudása szerint – tulajdonságokat. Például a képen

- **világos foltokat**
- **függőleges vonalakat**
- **bizonyos textúrát**
- **stb. stb.**

kiemelő szűrőket alkalmaz.





Az irgalmatlan bonyolult háló tanításához irgalmatlan sok minta kell! (Emlékeztető: a túltanulás veszélye miatt a szükséges mintaszám és az eszköz komplexitása összefügg. Minél komplexebb egy eszköz, általában annál több minta kell, hogy ne tanuljon túl.)

7. Transzfer tanulás (transfer learning)

Egy már – rengeteg mintával – valamilyen célra megtanított hálót átveszünk, és csak egy kis részét tanítjuk újra a mi hasonló, de egyedi problémánkból származó mintákkal!