A mesterséges intelligencia alapjai

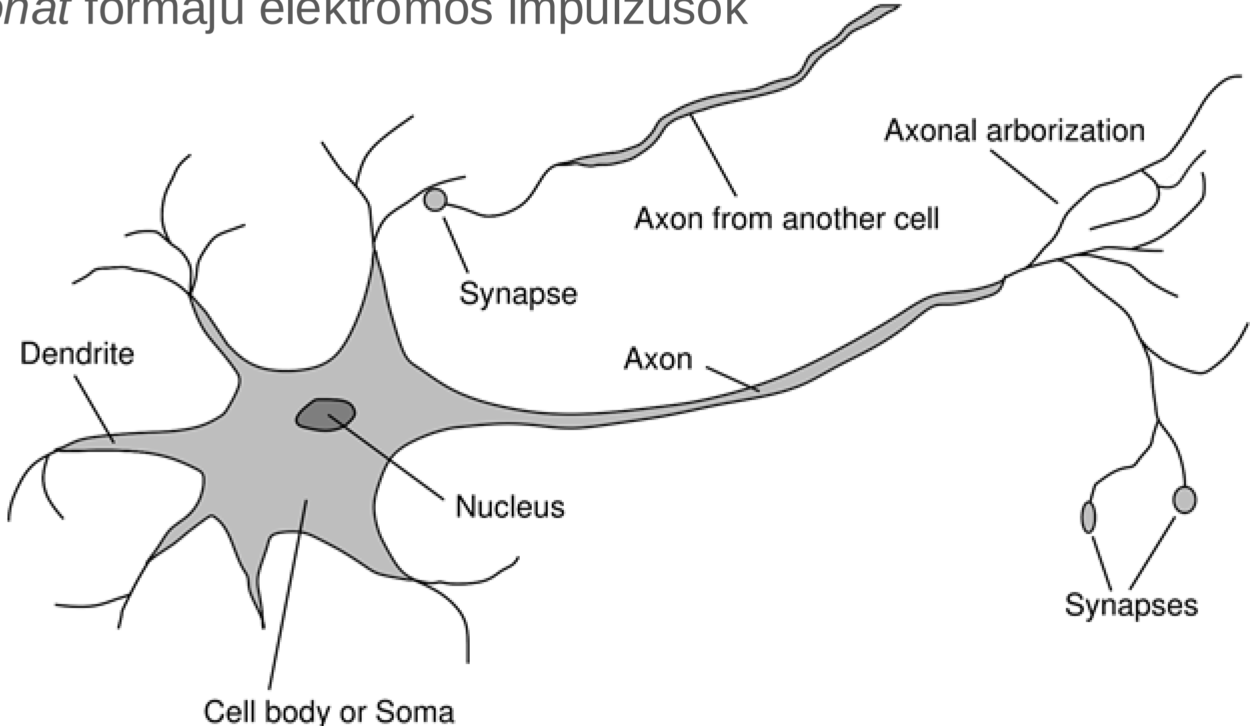
neurális hálók

tartalom

- az agy szerkezete
- neurális hálók
- perceptron
- többrétegű perceptron
- alkalmazások

neuronok és kapcsolataik

- 10^{11} neuron (több mint 20 típus), 10^{14} szinapszis, 1-10 ms pontos időzítéssel
- a jelek zajos, *tüskés vonat* formájú elektromos impulzusok

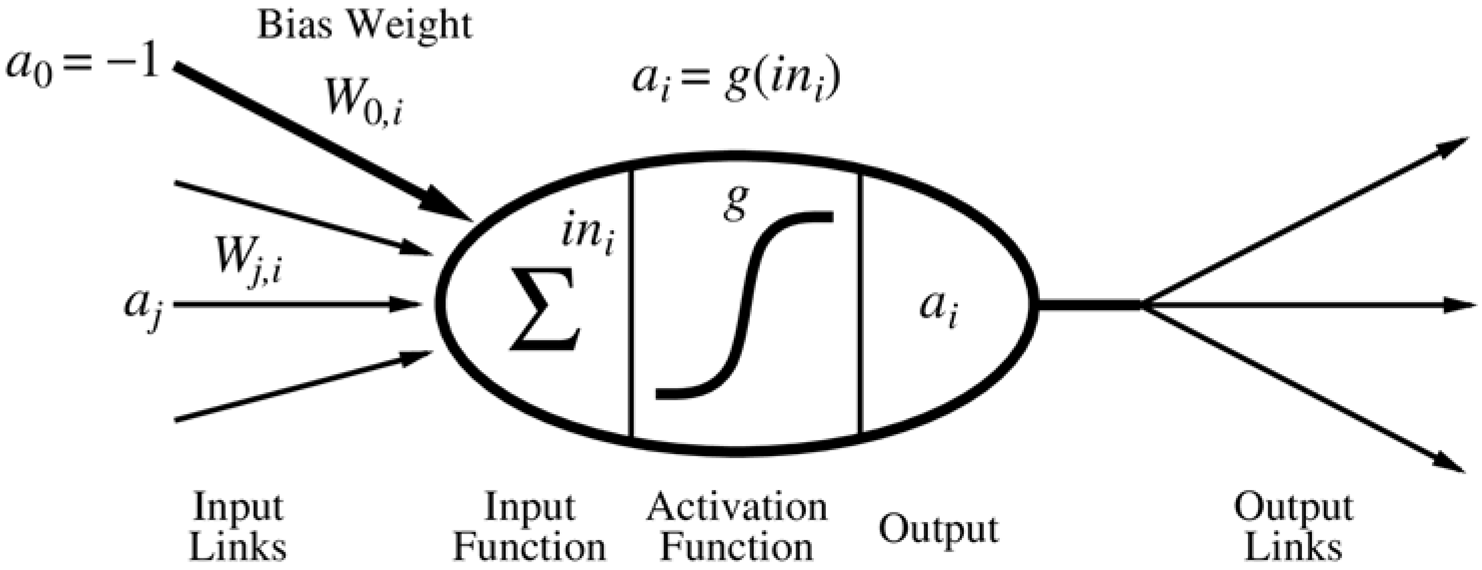


mesterséges modell

McCulloch-Pitts egység

mesterséges modell 1943-ból - a neuron akkor tüzel, ha a bemeneti értékek összege meghalad egy küszöböt (nagyon elnagyolt modell, de a lényeget tartalmazza)

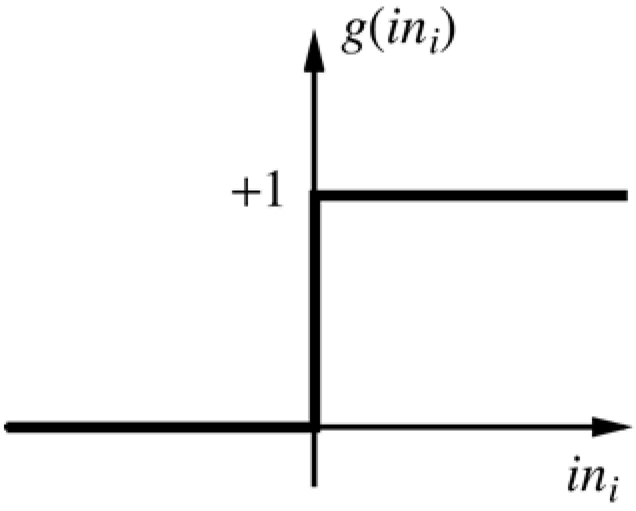
$a_i = g(\sum_j w_{j,i} a_j)$



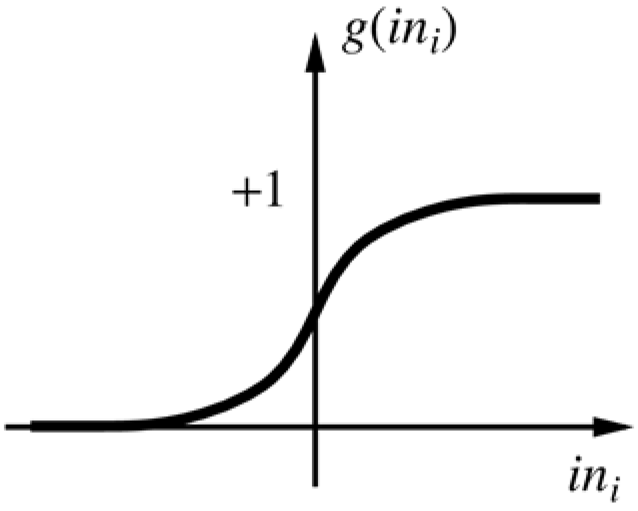
aktivációs függvény

a) küszöb aktivációs függvény
b) szigmoid függvény

$W_{0,i}$ eltolássúly módosításával mozgatható a küszöb



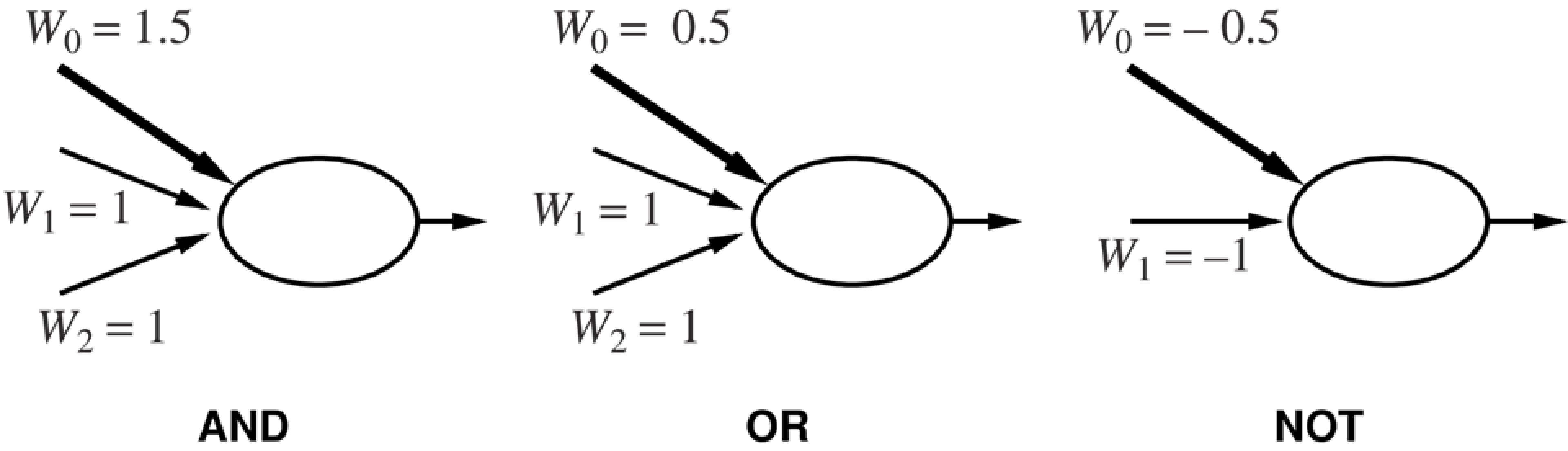
(a)



(b)

logikai függvények implementációja

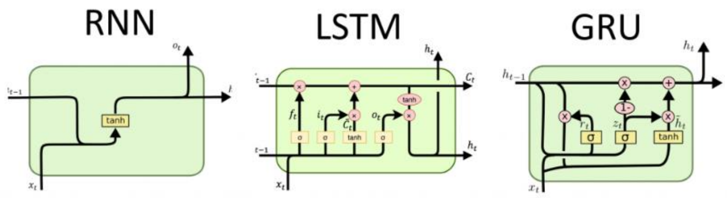
McCulloch és Pitts: minden logikai függvény implementálható



hálóstruktúrák

hálóstruktúrák

- előrecsatolt hálók
  - függvényt definiál, nincs memóriája
  - a pillanatnyi bemenet függvénye a kimenet
  - jellemzően rétegekbe szervezzük, egy egység csak a szomszédos rétegekkel áll kapcsolatban
  - egyrétegű/többbrétegű perceptronok
- rekurrens (visszacsatolt) hálók (link)
  - Hopfield hálózat (1982), bináris küszöb, szimmetrikus súlyok ($W_{ij}=W_{ji}$), $g(x)=sign(x)$, $a_i=\mp1$
  - Boltzman gépek (1986), sztochasztikus aktivációs függvény, ≅MCMC Bayes-hálóknál
  - rekurrens hálók irányított, késleltetett ciklusokkal
    - belső állapot, oszcilláció, memória

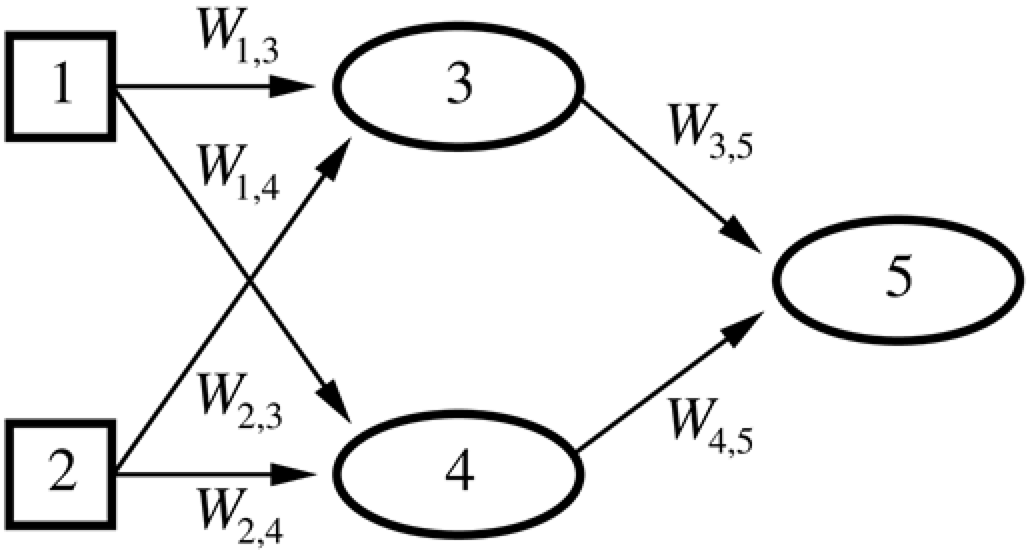


egyszerű előrecsatolt háló, egy rejtett réteggel

paraméterezett nemlineáris függvények családja

- a súlyok megváltoztatásával a függvény is változik, ez a tanulás módja

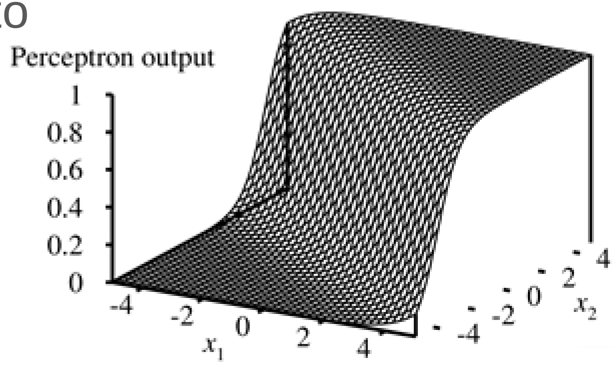
$a_5 = g(W_{3,5}a_3+W_{4,5}a_4) = g(W_{3,5}g(W_{1,3}a_1+W_{2,3}a_2)+W_{4,5}g(W_{1,4}a_1+W_{2,4}a_2))$

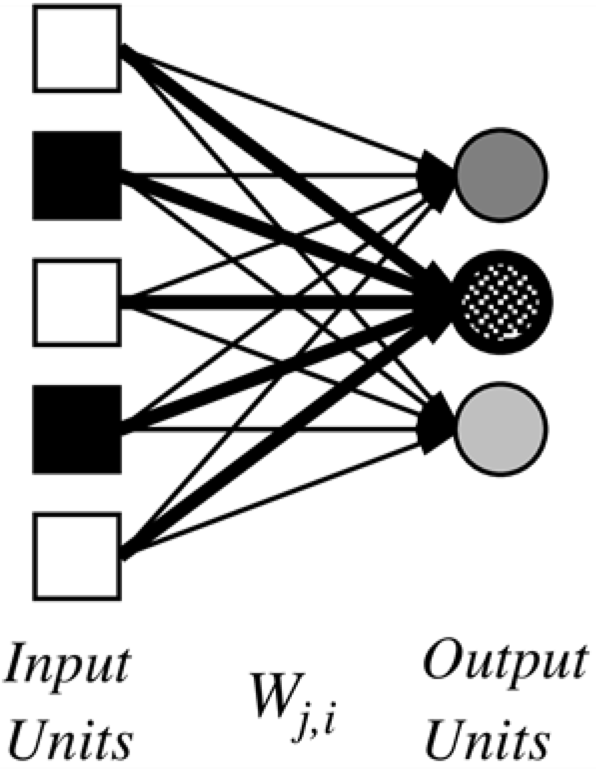


egyrétegű előrecsatolt neurális hálók (perceptronok)

az összes bemenet közvetlenül a kimenetre kapcsolódik

- minden kimeneti egység független a többitől, így elég egyenként vizsgálni
- a súlyok módosításával a output-függvény helye, alakja, iránya változtatható

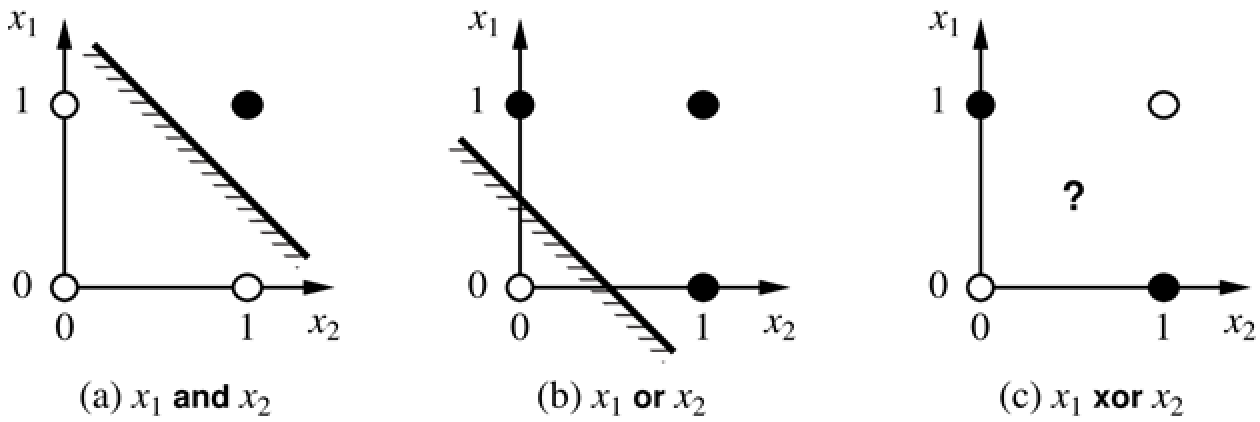




a perceptron kifejezőképessége

Tekintsük, amikor g = küszöb (Rosenblatt 1957, 1960)

- képes az AND, OR, NOT, a többségi függvényt ábrázolni, a XOR-t nem
- lineáris szeparátor függvény $\sum_j W_j x_j > 0$, vagy $\mathbf{W}\mathbf{x} > 0$
- Minsky és Papert könyve (1969) problémák, hiányosságok bemutatása; kutatások visszavetése



perceptron tanulás

Tanulás a tanulóhalmazon mért hiba csökkentése a súlyok módosításával

x input, *y* output esetén a négyzetes hiba:

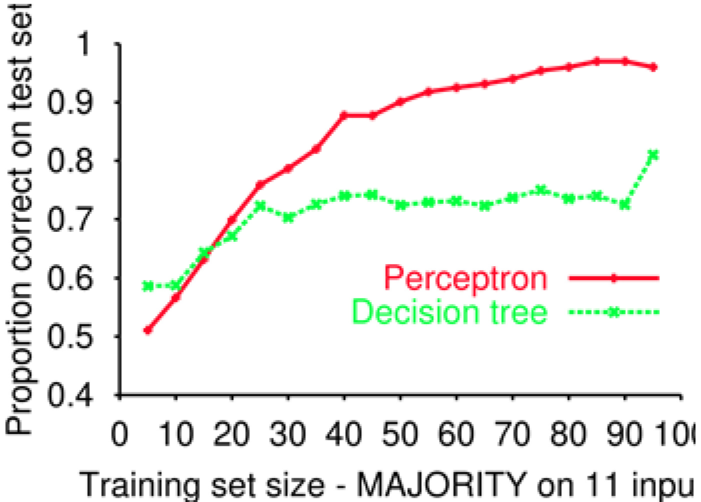
- $E = ½Err^2 = ½(y-h_w(x))^2$, ahol $h_w(x)$ a perceptron kimeneti értéke
- gradiensalapú optimalizálás, minden súlyra vonatkozó parciális derivált kell

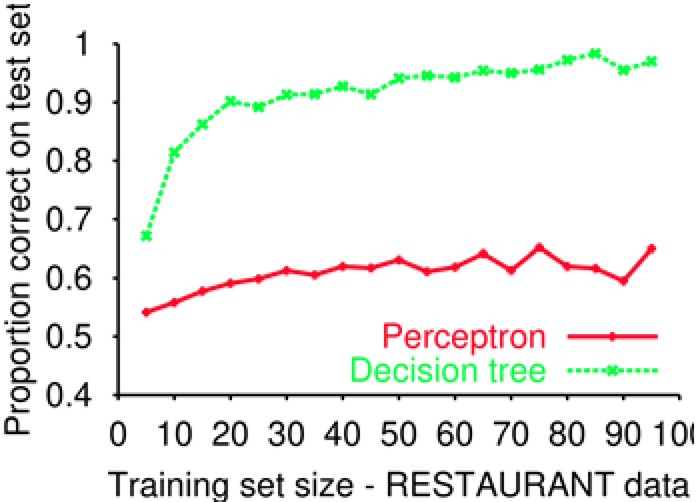
$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j}\left(y - g\left(\sum_{j=0}^{n} W_j x_j\right)\right) = -Err \times g'(in) \times x_j$$

- súlyfrissítés
  - $W_j \leftarrow W_j + \alpha Err \times g'(in) \times x_j$ (**α** - tanulási faktor)
- ha a hiba pozitív (a háló kimenete kicsi), pozitív bemenet súlyait növelni, negatív bemenet súlyait csökkenteni kell (ha negatív a hiba, akkor fordítva)

perceptron tanulási görbe

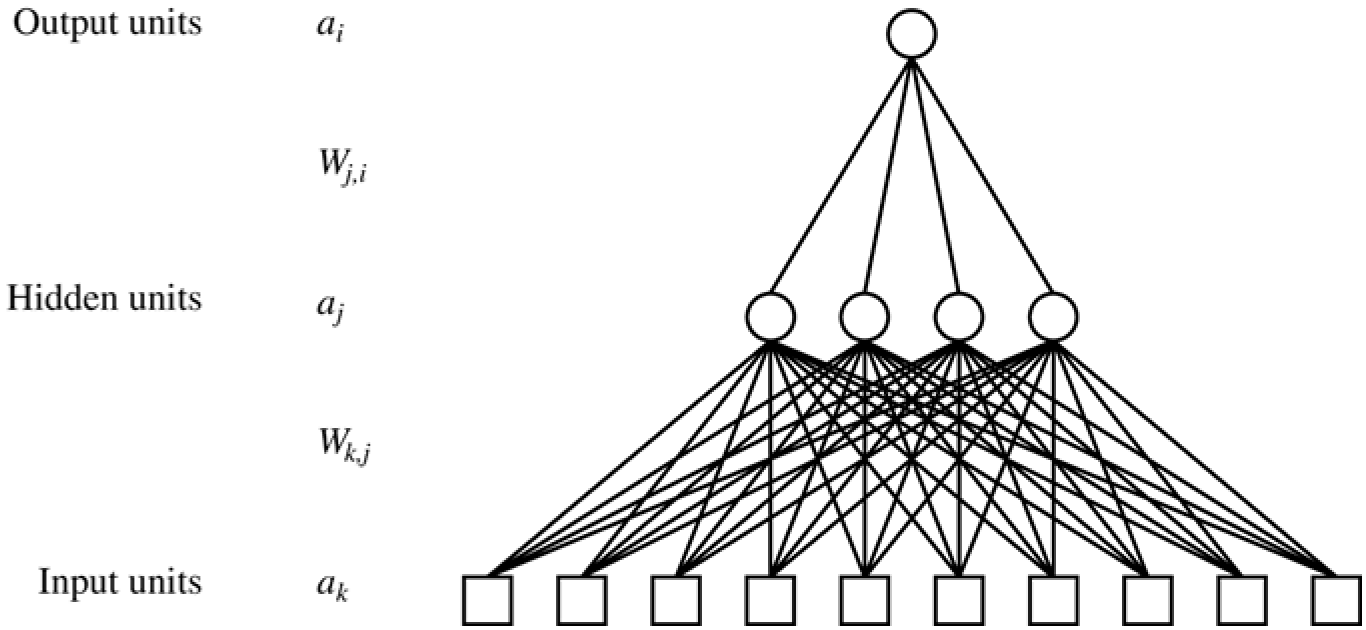
- a perceptron tanulása egy konzisztens függvényhez konvergál
  - minden lineárisan szeparálható adathalmaz esetén
- a perceptron a *többségi* függvényt gyorsan megtanulja, a döntési fa képtelen
- a vendéglői függvényt a döntési fa jól tanulja, a perceptron nem képes ábrázolni





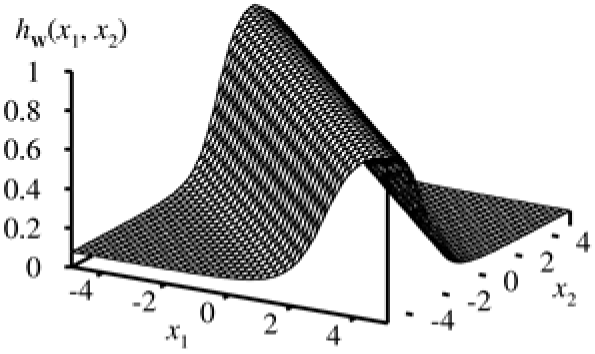
többrétegű előrecsatolt neurális hálók (TEN)

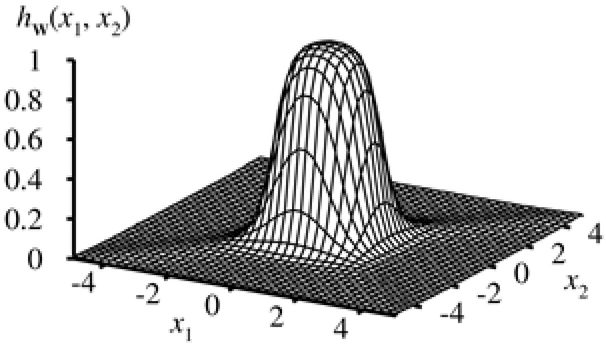
- a rétegek rendszerint teljesen összekötöttek
- a rejtett rétegek és egységek számának meghatározására nincs módszer



TEN kifejezőképessége

- két, egymással szemben álló lágy küszöbfüggvény -> hegygerinc
- két hegygerinc kombinációja -> dudor
- sok dudorral minden függvény közelíthető
  - egy, megfelelően nagy (akár exponenciális) rejtett réteggel bármely folytonos függvény tetszőlegesen közelíthető
  - két rejtett réteggel nem folytonos függvények is közelíthetőek





hiba-visszaterjesztéses tanulás (back-propagation)

- kimeneti réteg - mint korábban
  - $W_{j,i} \leftarrow W_{j,i} + \alpha a_j \times \Delta_i$ $\Delta_i = Err_i \times g\prime(in_i)$ ahol
- rejtett réteg: terjesszük vissza a hibát a kimeneti rétegből
  - $\Delta_j = g\prime(in_j) \sum_i W_{j,i} \Delta_i$
- frissítsük a rejtett réteg súlyait:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \Delta_j$$

(az agykutatók szerint az agyunkban nincs ilyen hiba-visszaterjesztés)

hiba-visszaterjesztés számítása (kimeneti réteg)

- egyetlen mintára a négyzetes hiba
  - $E = \frac{1}{2}\sum_i (y_i - a_i)^2$ ahol a kimeneti réteg csomópontjaira összegzünk

$$\frac{\partial E}{\partial W_{j,i}} = -(y_i - a_i)\frac{\partial a_i}{\partial W_{j,i}} = -(y_i - a_i)\frac{\partial g(in_i)}{\partial W_{j,i}}$$

$$= -(y_i - a_i)g'(in_i)\frac{\partial in_i}{\partial W_{j,i}} = -(y_i - a_i)g'(in_i)\frac{\partial}{\partial W_{j,i}}\left(\sum_j W_{j,i}a_j\right)$$

$$= -(y_i - a_i)g'(in_i)a_j = -a_j\Delta_j$$

hiba-visszaterjesztés számítása (rejtett réteg)

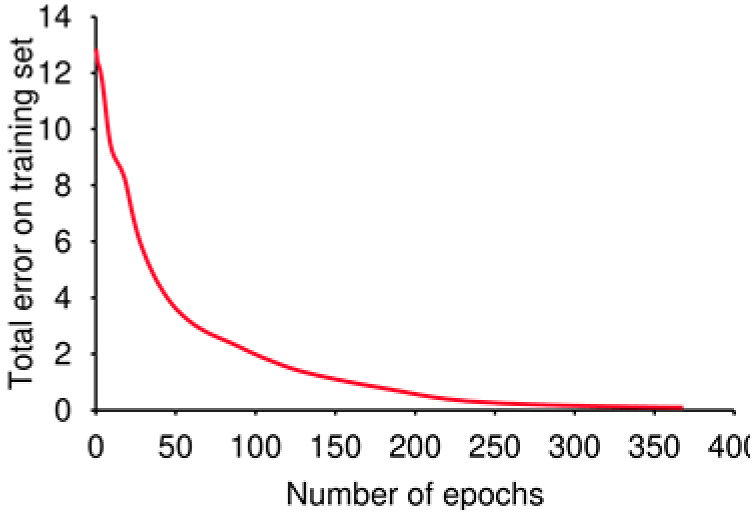
$$
\begin{aligned}
\frac{\partial E}{\partial W_{k,j}} &= -\sum_i (y_i - a_i)\frac{\partial a_i}{\partial W_{k,j}} = -\sum_i (y_i - a_i)\frac{\partial g(in_i)}{\partial W_{k,j}} \\
&= -\sum_i (y_i - a_i) g\prime(in_i)\frac{\partial in_i}{\partial W_{k,j}} = -\sum_i \Delta_i \frac{\partial}{\partial W_{k,j}}\left(\sum_j W_{j,i} a_j\right) \\
&= -\sum_i \Delta_i W_{j,i} \frac{\partial a_j}{\partial W_{k,j}} = -\sum_i \Delta_i W_{j,i} \frac{\partial g(in_j)}{\partial W_{k,j}} \\
&= -\sum_i \Delta_i W_{j,i} g\prime(in_j) \frac{\partial in_j}{\partial W_{k,j}} \\
&= -\sum_i \Delta_i W_{j,i} g\prime(in_j) \frac{\partial}{\partial W_{k,j}}\left(\sum_k W_{k,j} a_k\right) \\
&= -\sum_i \Delta_i W_{j,i} g\prime(in_j) a_k = -a_k \Delta_j
\end{aligned}
$$

tanulási görbe

a tanuló algoritmus minden egyes mintára kiszámolja a hálót, és kissé módosítja a súlyokat, a mintahalmaz végigfuttatását epochnak nevezzük

100 étterem feladata esetén - végül teljes egyezés ér el.

jellemző hibák: lassú konvergencia, lokális minimumba érkezés



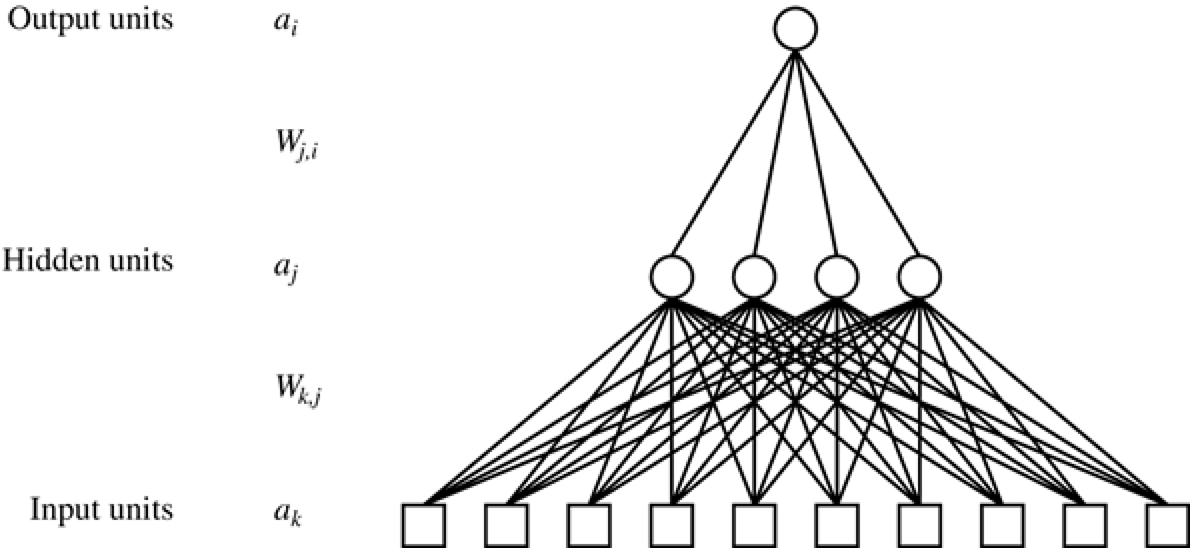
tanulási görbék

4 rejtett egységet tartalmazó TEN

A TEN rendszerint nagyon jó komplex felismerési feladatokra (valós alkalmazások)

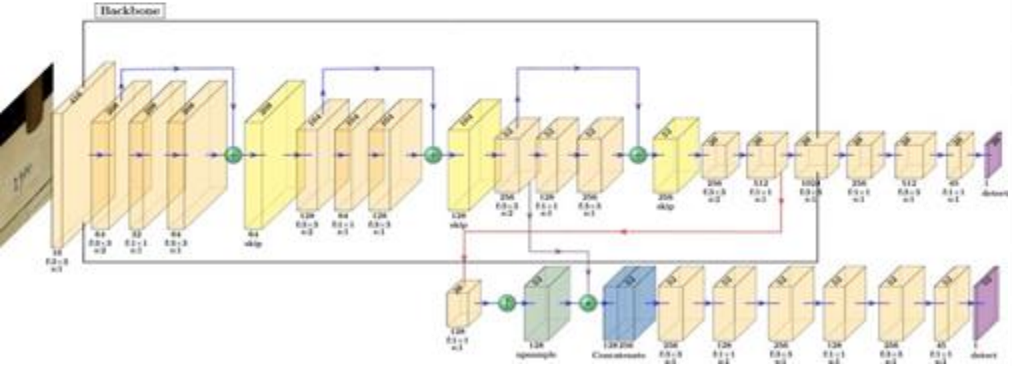
A megoldást nem értjük.

Hogyan verjük át a TEN-t?



kézzel írt számjegyek felismerése

- 3 legközelebbi szomszéd 2,4%-os hiba
- 400-300-10 egységes TEN 1,6%-os hiba
- LeNet: 768-192-30-10 TEN 0,9%-os hiba (1998, konvolúciós)
- 2003 legjobb módszerei 0,6%
- 2020 - LeNet variáns (10 r.) 0,6%



összefoglalás

- agyunkban elképesztően sok neuron található
    - minden neuron egy lineáris küszöb egység?
- a perceptron önmagában nem elég kifejező
- a többrétegű előrecsatolt neurális háló már eléggé kifejező, gradiens módszerrel tanítható, hiba-visszaterjesztéssel
- napjaink *csodafegyvere*
    - főleg a mélytanulás
- a tervezés, kognitív modellezés, neurális rendszerek modellezése
    - jelentősen elvált tudományterületek