

# Perceptron, neurális hálózatok

Csima Judit

BME, VIK,  
Számítástudományi és Információelméleti Tanszék

2015. április 1. és 8.

# Mesterséges neurális hálózatok

- most osztályozásra használjuk
- mindenféle függvény közelítésére jó
- ötlete az idegsejtek összehangolt működését utánozza
- régebben népszerű volt, aztán nem, de mostanában újra igen

## Motiváció: idegsejt

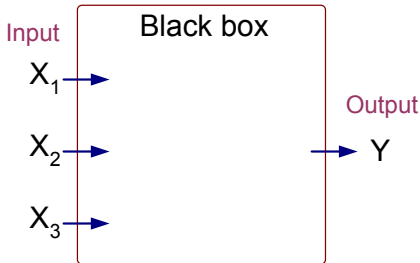
- a dendriteken (ezekből több van) keresztül impulzusokat gyűjt be, ha ezek összértéke egy küszöböt meghalad, akkor az idegsejt kisül és az axon-ján (ebből egy van) keresztül az ingert továbbadja a hozzá kapcsolódó másik idegsejteknek
- van tehát egy központi egység, amibe beérkezik az infó több helyről, itt történik vele valami és van egy kimenet, amin keresztül új infó hagyja el az idegsejtet
- a tanulás abból áll, hogy a bejövő kapcsolatok erőssége változik: a sokat használt kapcsolatokban erősödik, a ritkán használtakban gyengül

## Perceptron: mesterséges neuron

- vannak input csúcsok, itt jön be az input
- van egy központi csúcs, itt számolunk
- van egy kimenet, itt adjuk ki a választ
- az input csúcsokból bejövő kapcsolatok súlyozva vannak
- a számoló, központi csúcsban a súlyozott input alapján valami aktivációs függvénnyel kiszámoljuk a kimenetet

# Artificial Neural Networks (ANN)

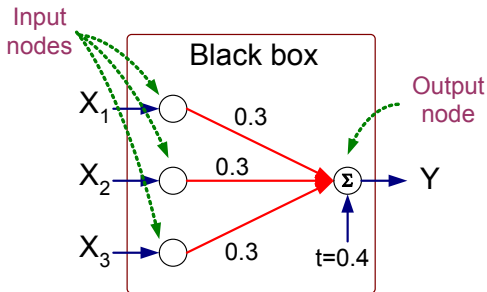
$X_1$	$X_2$	$X_3$	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

# Artificial Neural Networks (ANN)

$X_1$	$X_2$	$X_3$	$Y$
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

- Hogy jön ez az osztályozáshoz?
  - az inputok az egyes attribútumok
  - a kimenet az osztálycímke (ha bináris osztályozás van, akkor egy kimenet van, értéke 0 vagy 1 lehet)
  - bonyolultabb feladatokhoz nem elég egy perceptron, hálózat kell (erről majd)
- Mi az aktivációs függvény?
  - aktivációs függvény sok minden lehet
  - általában sigmoid függvényt használunk
  - ha bináris osztályozás van, akkor olyan függvény kell, aminek 0-1 kimenete van (főleg)
- Hogyan lesznek súlyok a kapcsolatokhoz?
  - erről majd később
  - backpropagation algo
  - egyenlőre nézzük, hogy hogyan működik, ha már vannak súlyok valahonnan

# AND, OR

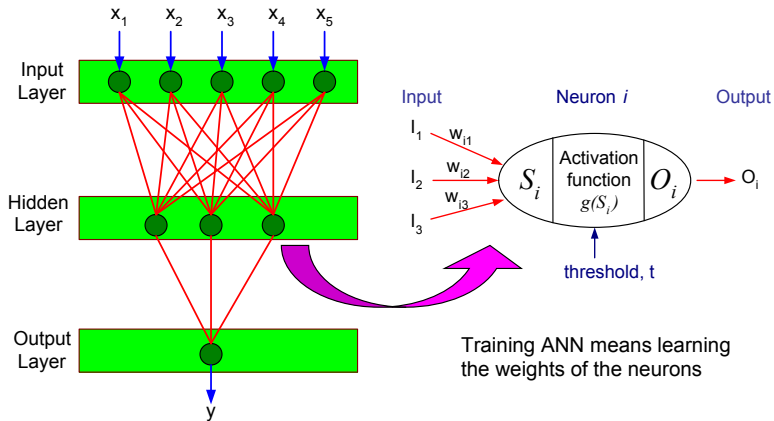
- ezekre lehet perceptront csinálni
- három input csúcs lesz, egy kimenet
- mindkét esetben lineárisan szeparálhatóak a két címkéhez tartozó halmazok
- ez azért fontos, mert egy perceptron csak lineárisan tud szeparálni
- XOR nem lineárisan szeparálható: erre már hálózatot kell építenünk



# Artificial Neural Network, multilayer perceptron

- rétegekbe szervezve több perceptron
- első szinten az input csúcsok vannak
- utolsó szinten a kimeneti csúcsok vannak (több is lehet)
- belső szinteken (hidden layer) szintén perceptronok
- szomszédos szintek perceptronjai össze vannak kötve
- minden perceptron az egyel korábbi szinten levőkből kapja az inputot és az egyel későbbi szintre továbbítja
- a kapcsolatok súlyai egymástól függetlenül állíthatók

# General Structure of ANN



## Több osztálycímke

- több kimeneti csúcs van: annyi, ahány lehetséges címke van
- az  $i$ . címkét kapja egy input  $n$ -es, ha az  $i$ . outputon van 1, máshol 0
- vagy ha nem csak 0 és 1 jöhet ki az outputra, akkor ahol a legnagyobb értéket kapjuk
- pl. karakterfelismerésnél annyi kimeneti csúcs van, ahány lehetséges karakter

## Mi a hálózat struktúrája?

- input csúcsok száma a problémából adódik
- pl. karakterfelismerésnél annyi csúcs, ahány pixeles a kép
- output csúcsok száma is adódik a problémából (előbb néztük)
- minden más viszont szabadon választható
- könnyű túlzásba esni: overfitting
- önmérséklet: legyen csak egy hidden layer
- vagy legalább legyen minden hidden layer-en ugyanannyi csúcs

## Paraméterek tanulása, egy perceptron

- vannak tanító példáink, minden példa egy  $x$  input-vektorból és egy  $y$  értékből áll ( $x$  esetén  $y$  kimenetet várunk)
- vannak  $\Theta$  értékeink a kapcsolatokhoz (kezdetben random választva, később az eggyel korábbi iteráció eredménye)
- kiszámoljuk az aktuális  $\Theta$  súlyozással a kimenetet (ez  $\hat{y}$ ), a hálózat aktuális hibája  $\hat{y} - y$
- az új  $\Theta$  értékek ezután:  $\Theta_j = \Theta_j + \lambda(\hat{y} - y)x_j$
- ezt iteráljuk több tanító példán addig, amíg nem konvergál

## Perceptron tanulás, intuíció

- $\Theta_j = \Theta_j + \lambda(\hat{y} - y)x_j$
- $\lambda$  a tanulási paraméter (learning rate), ez mutatja, hogy mennyire befolyásolja az aktuális hiba  $\Theta$  új értékét
- arról van szó, hogy a régi  $\Theta$  értékeket akkor módosítjuk csak, ha  $\hat{y} - y$  nem nulla, azaz amikor nem az elvárt kimenetet kapjuk
- minél nagyobb volt az adott élen jövő input (minél jobban hozzájárult az outputhoz), annál nagyobb a módosítás
- $\lambda$  értéke változhat a tanítás során (elején nagyobb, aztán egyre kisebb)

# Paraméterek tanulása hálózatban

- egyesével veszem a tanító példákat
- minden példa alapján állítok a paramétereken
- egy paraméter-állítás így néz ki vázlatosan:
  - forward propagation: az aktuális  $\Theta$  és  $x$  értékekkel minden belső és output csúcsra kiszámolom az ottani kimenetet
  - az output csúcsokon kapott kimenetet összevetem az ott elvárt kimenettel ( $y$ ) és ez alapján kiszámolom a kimeneti csúcsok hibáját
  - backpropagation: az aktuális  $\Theta$  értékeket használva visszaszámolom a hibát a korábbi szintek csúcsaira (kivéve input szint)
  - így minden (belső és output) csúcsra, mint perceptronra van egy input, van egy kiszámolt output és van egy kiszámolt hiba
  - ez alapján  $\Theta$  korrekciója, ahogy a perceptronnál volt

## Paraméterek tanulása, általános jellemzők

- igazából egy sokdimenziós térben levő felület minimumát keresem
- a felület a hálózat hibáját leíró függvény
- a dimenzió a  $\Theta$  paraméterek darabszámából jön
- a módszer ezen felület egy lokális minimumába konvergál, ha  $\lambda$  jól van megválasztva
- lokális vs. globális minimum: több véletlen indítás
- a paraméterek módosítása annak felel meg, hogy ezen felület mentén mozdulok el a  $\Theta_j$  szerinti parciális derivált irányába (ezt nem lehet látni ennyiből, amit tanultunk, de ez van mögötte)



Az alábbi két függvényről döntse el, hogy lehet-e rájuk perceptront szerkeszteni vagy csak hálózatot és adjon is meg perceptront vagy hálózatot hozzájuk:

- $(\text{not } A) \text{ AND } B$
- $(A \text{ OR } B) \text{ AND } (A \text{ OR } C)$