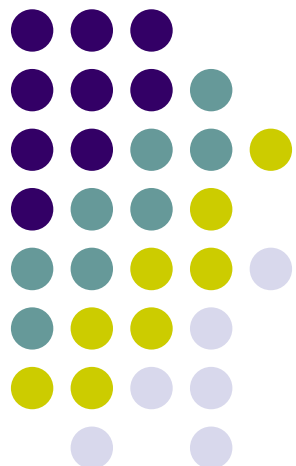
Pannon Egyetem
Villamosmérnöki és Információs Tanszék

Bevezetés a neurális számításokba – Analóg processzor-tömbök, neurális hálózatok



Előadó: dr. Tömördi Katalin

Neurális áramkörök (ismétlés)

A „neurális” hálózatokban a neuronok egy szabályos geometriai rácson helyezkednek el, melyek egy processzor réteget alkotnak (1 v 2 dimenziós). Több réteg összekapcsolásával kapjuk a teljes neurális hálózatot.

A programot a rétegek közötti és a rétegen belüli összeköttetések súlytényezői jelentik.

Tervezési irányok:

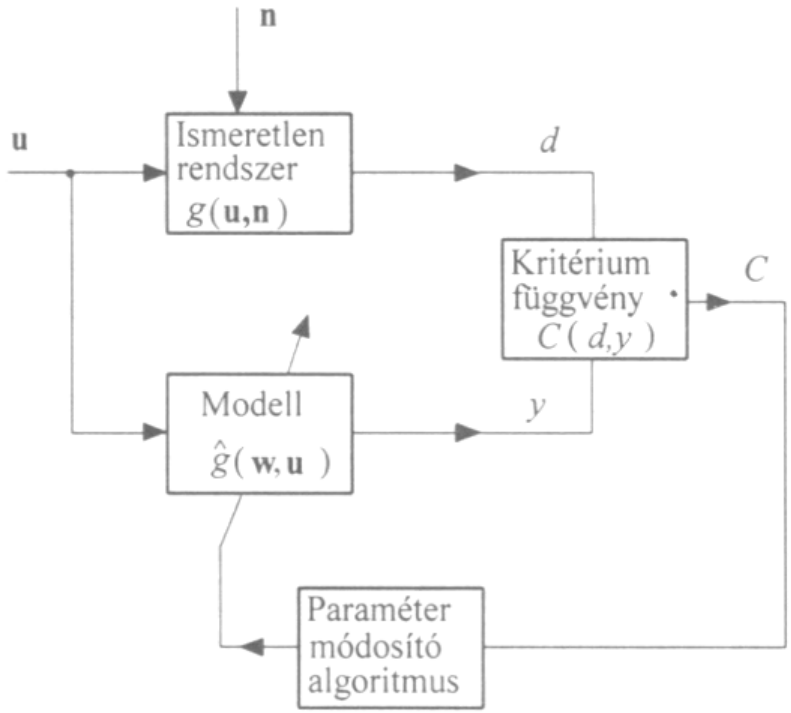
- élő szervezetek részletes modelljeinek elektronikai utánzása
- idegrendszeri működés struktúrájának és az elemi neuron-működés modelljének utánzása
- tipikus kanonikus optimálási eljárások, komplex fizikai, kémiai effektusok áramköri megoldásba transzformálása

A tanulási szabályok főbb típusai

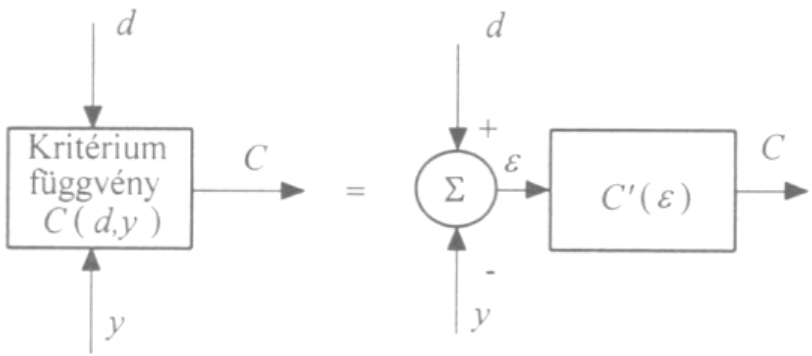
- tanítással/tanítóval tanulás (Ellenőrzött vagy felügyelt tanítás)
  - amikor a bemenetekhez a kívánt kimeneteket is megadjuk, ismerjük. A kívánt és számított válaszok alapján módosítjuk a súlytényezőket, cél, hogy a különbség csökkenjen.
  - Megerősítő tanulás – amikor csak azt tudjuk, hogy a válasz helyes vagy nem
- tanítás/tanító nélkül (Nem ellenőrzött vagy felügyelet nélküli tanítás)
  - Nem ismerjük a kívánt válaszokat. Valamilyen adaptív úton a bemenetek sorozata alapján a beépített tanuló algoritmus valamilyen szempont szerint osztályokat alkot (és egyben módosítja saját magát - azaz az összeköttetések súlytényezőit) ezért önszervező hálózatnak is hívják
- Analitikus tanítás - direkt eljárással
  - a súlytényezőket a feladatból elméleti úton, zárt formában kapjuk meg.

Ellenőrzött tanulás

- Összetartozó be-ki meneti tanító mintapárokkal
- Neurális hálónk, vagyis a megtanított rendszer a vizsgált rendszer valamilyen modelljeként jelenjen meg, működése a rendszer működésének feleljen meg, - modell illesztési feladat (ábra)
- **d**=g(**u**,**n**) , **n** zaj-hatások
  y= ĝ(**u**) modell illesztés, itt a paramétereken keresztül – **y**= ĝ(**w**,**u**) kifejezőbb.
- Az illesztési feladat ekkor minimalizálási, szélsőérték keresési feladat, ahol C(**d**,**y**) kritérium fv. minimumát keressük **w** függvényében



(a)



(b)

3.1 ábra (a) A modell-illesztés általános sémája, (b) a kritériumfüggvény tipikus felépítése

Ellenőrzött tanulás

Kritérium függvény

- C=f(d,y)
- Leggyakrabb valamilyen négyzetes hibakritérium:

$$C(\mathbf{d},\mathbf{y}) = E\left\{(\mathbf{d}-\mathbf{y})^T(\mathbf{d}-\mathbf{y})\right\} = E\left\{\sum_j (d_j - y_j)^2\right\}$$

Lehet még pl.:

abszolút érték fv., annak hatványai, lin. Kombinációi, igen-nem fv.

Ellenőrzött tanulás
szélsőérték keresés

- A kritérium fv minimumát, szélsőértékét kereső eljárások leginkább gradiens alapúak:

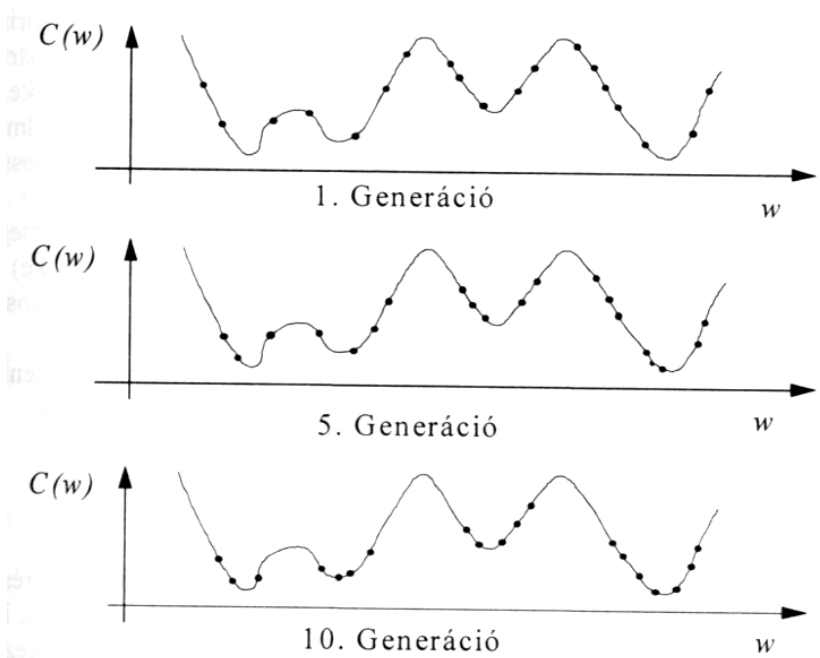
$$\nabla\left[C(\mathbf{w})\right] = \frac{\partial C}{\partial \mathbf{w}} = 0$$

- Megerősítő tanulásnál csak azt tudjuk,helyes-e a válasz (lehet több is), csak „kritika” van (learning with a critic) nincs hiba mérték, nincs gradiens. Ez a módszer akkor lehet jó, ha több jó válasz van ugyanarra a bementre.
- Sztochasztikus eljárások
  Gradiens módszerek hátránya – lokális minimumokban benn ragadhat, ezért véletlen zaj segítségét használják

Ellenőrzött tanulás
szélsőérték keresés

- Véletlen keresés: (Ha nem ismerjük a kritérium felületet)
  Pl. ha a próbált paraméterhez tartozó kritérium érték az előzőnél kisebb, akkor ez lesz a következő lépés kiinduló pontja, egyébként a régi, és úgy keresünk tovább
- Előzőek kombinációi
- Genetikus algoritmus:
  itt nem egy pont mozog a vizsgálat szempontjából a kritérium felületen, hanem egyszerre több ponton értékelünk.

  jellemző: kereszteződés, mutáció, reprodukció



3.13 ábra A genetikus algoritmus szélsőérték-kereső mechanizmusa

Nem ellenőrzött tanulás

Nincsenek kívánt kimenetek megadva

- Hebb tanulás (biológiai eredetű)
  két elem közti w az elemek aktivitásának szorzatával arányosan nő: $w_{ij}(k+1) = w_{ij}(k) + \mu x_i y_j$

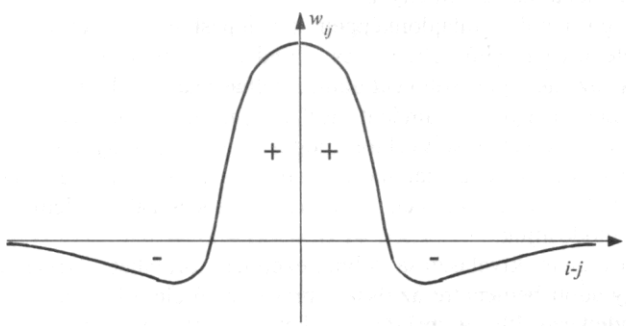
- Anti Hebb: $w_{ij}(k+1) = w_{ij}(k) - \mu x_i y_j$
  A különböző változatainál normalizáló eljárások használatosak – súlyok ne növekedjenek minden határon túl

Nem ellenőrzött tanulás

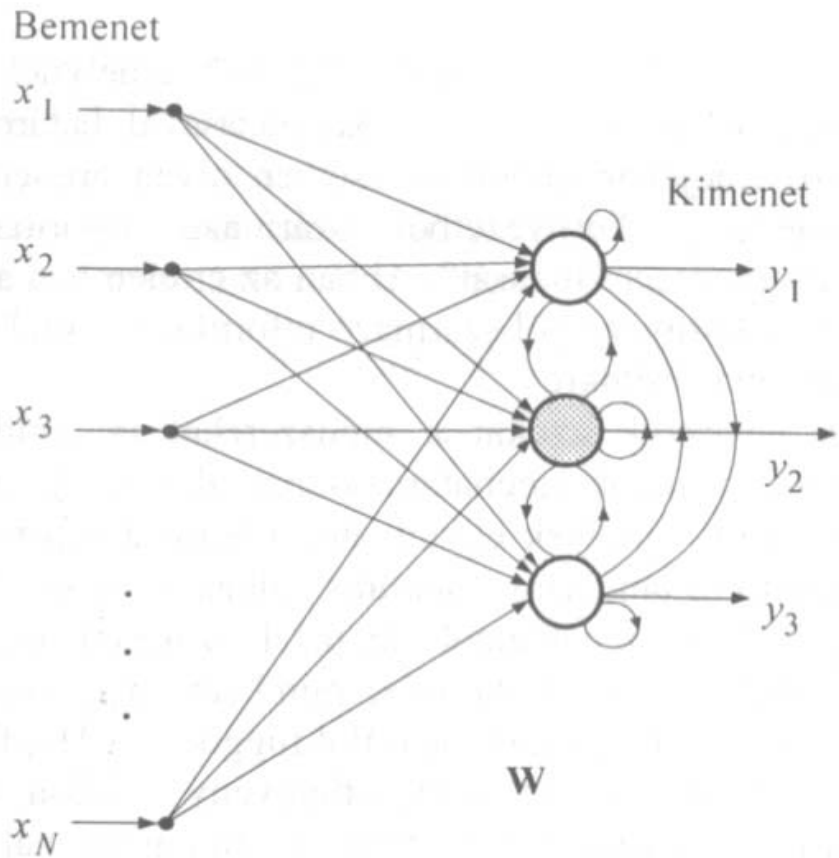
- Versengő
  A processzáló elem-elrendezésben egy győztest válasszunk ki. A győztes PE kimenete aktiv 1 , a többié 0. A cél a bemeneti mintatér olyan tartományokra osztása, hogy minden egyes tartományba tartozó bemenet hatására egy, és csakis egy elem aktivizálódjon.
  Két lépés van: 1. kiszámoljuk minden elem kimenetét, kiválasztjuk a győztest, amely a győztes mindent visz elv alapján működik. 2. csak az ő vagy még a környezete súlytényezőit is módosítjuk

Nem ellenőrzött tanulás

Győztes PE beállítása pl. az oldalirányú súlytényezők beállításával: közeli PE-k között gerjesztő, távoliknál gátló kapcsolat: Mexikói kalap fv.



3.17 ábra Az oldalirányú kapcsolatokat meghatározó mexikói kalap függvény



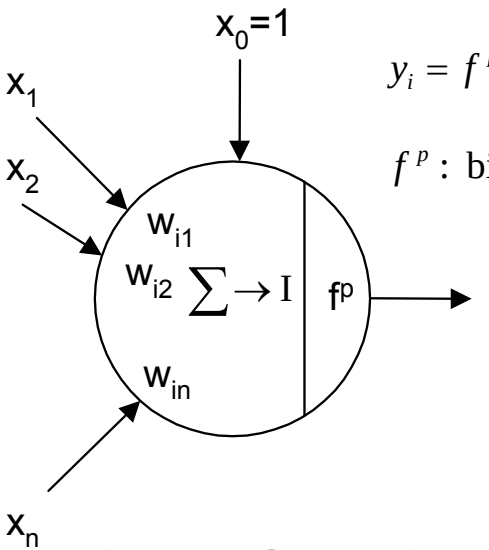
3.16 ábra Versengő tanulással tanított hálózat

Analitikus tanulás

Energia fv. pl.:

- Hopfield
- CNN (Cellular Neural Network)

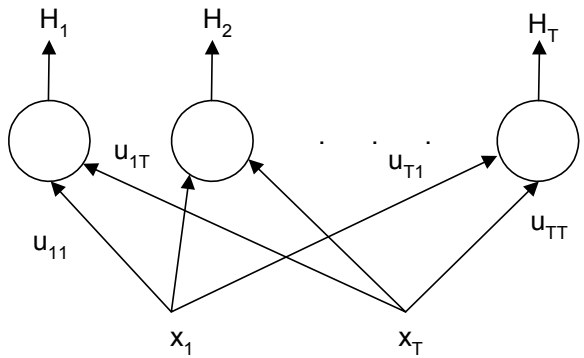
Ellenőrzött tanulás
A perceptron (Rosenblatt) tanulása



$$y_i = f^p\left(\sum_{j=0}^{n} w_{ij}x_j\right) = f^p(I) = \begin{cases} 1.0 \text{ ha } I > 0 \\ 0.0 \text{ ha } I \le 0 \end{cases}$$

f^p : bináris küszöbelem

$$I = \sum_{j=0}^{n} w_{ij}x_j = w_{i0} + \sum_{j=1}^{n} w_{ij}x_j$$



A tanulás szakaszosan történik. Egy tanulási szakasz (epoch) alatt egy adott bemenet(sorozat)ból kiszámít a neurális hálózat egy kimenetet (sorozatot) és ennek értékétől függően megváltoztatjuk a súlytényezősorozat (a „program”) értékét.

A perceptron tanulása

A tanulás szabályai:

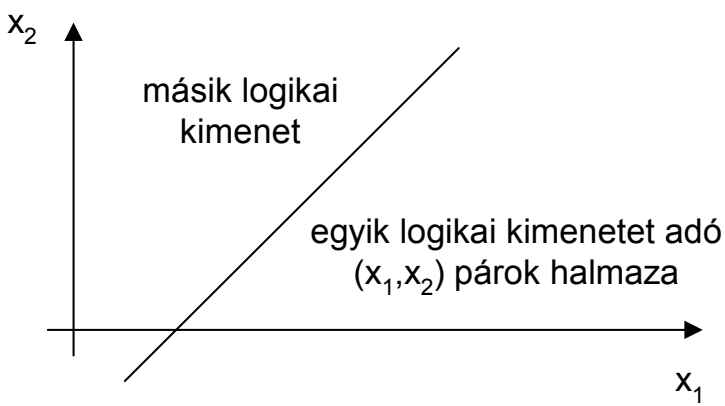
- ha az adott bemenet mellett számított kimenet jó ($y_i = d_i$), akkor az ezen PE-hez tartozó súlytényezőket (w_{ij}) nem változtatjuk;
- ha $y_i = 0$, de $d_i = 1$, akkor növeljük az "aktív" inputok (melyek zérusnál nagyobbak) súlytényezőit;
- ha $y_i = 1$, de $d_i = 0$, akkor csökkentjük az "aktív" inputok súlytényezőit.

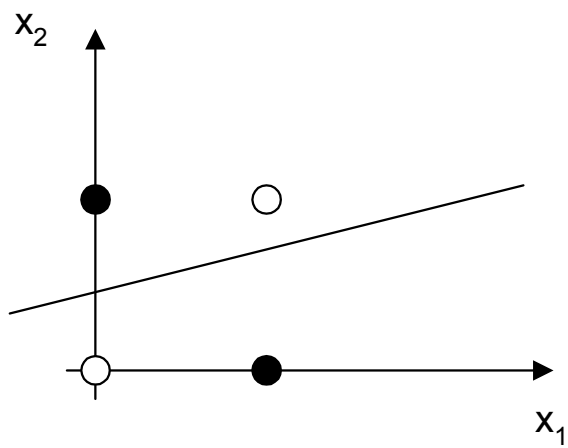
A fenti növelések és csökkentések mértéke sokféle lehet, tipikusan:

- rögzített növekmény,
- a ($d_i - y_i$) -vel arányos növekmény vagy
- az előző kettő kombinációja.

A perceptron tanulása és a lineáris szeparálhatóság

A kétbemenetű perceptron $I=w_1x_1+w_2x_2+w_0$ egyenletének megoldása az (x_1, x_2) kétdimenziós síkon egy egyenest ad. Ez az egyenes tehát lineárisan szeparálja azon (x_1, x_2) értékek halmazát, amelyek 0 kimenetet adnak azoktól, melyek 1-et adnak. Az XOR (exclusive or) logikai függvény igazságtáblázata azonban az alábbi ábrán látható ebben a koordináta rendszerben.





lineáris szeparálhatóság

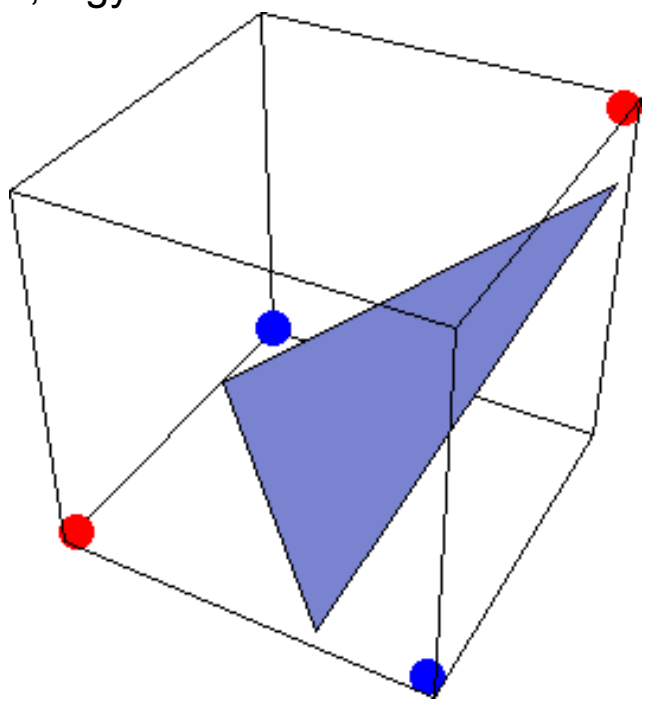
Megoldás:

- rejtett réteget vezetünk be, vagy
- magasabb dimenzióban probálkozunk lineáris szétválasztással.

Az utóbbi esetben egy új input változót alkalmazunk, legyen
x3 = x1 x2
azaz

x1	x2	x3	y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

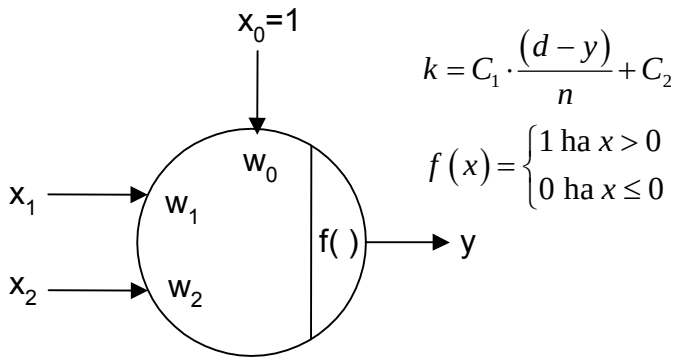
Ekkor az bemeneti állapotokat térben ábrázolhatjuk és ezeket a pontokat már egy síkkal szét tudjuk választani:



Térben lineárisan szeparálható XOR függvény

http://geo2.fmt.bme.hu/palancz/neural/Chapt01/HTMLLinks/index_8.html

Perceptron tanulása - Példa



$$k = C_1 \cdot \frac{(d-y)}{n} + C_2$$

$$f(x) = \begin{cases} 1 \text{ ha } x > 0 \\ 0 \text{ ha } x \le 0 \end{cases}$$

Az ábrán látható perceptron ki- ill. bementének lehetséges értékei a 0 (inaktív) és az 1 (aktív). A tanulási szabály kimondja, hogy ha a kívánt és a számolt kimenet megegyezik, akkor a súlytényezők értékei

nem változnak, ellenkező esetben pedig, ha a kimenet aktív kellene legyen, de nem az, akkor növeljük, míg ha a kimenet passzív kellene legyen, de nem az, akkor csökkentjük az aktív bemenetekhez tartozó súlytényezőket. A hálózatnak az ÉS műveletet fogjuk megtanítani, mely az alábbi 4 szabályt jelenti:

x_1	0	1	0	1
x_2	0	0	1	1
d	0	0	0	1

Perceptron tanulása - Példa

A két tanulási együtthatót C_1=0 ill. C_2=0.1-nek választjuk, azaz fix növekményt (k=C_2) használunk. Mindhárom súlytényező értékét kezdetben 0.08-nak választva a táblázatban nyomonkövethetjük a tanulás folyamatát. Ennek során először kiszámoljuk a súlyozott összeget $\left(I = \sum w_i x_i\right)$ majd a kimenetet (y=f(I)) , és végül hiba (e≠0) esetén ±k-val módosítjuk az aktív bemenetekhez (x_i=1) tartozó súlytényezőket.

Perceptron tanulása - Példa

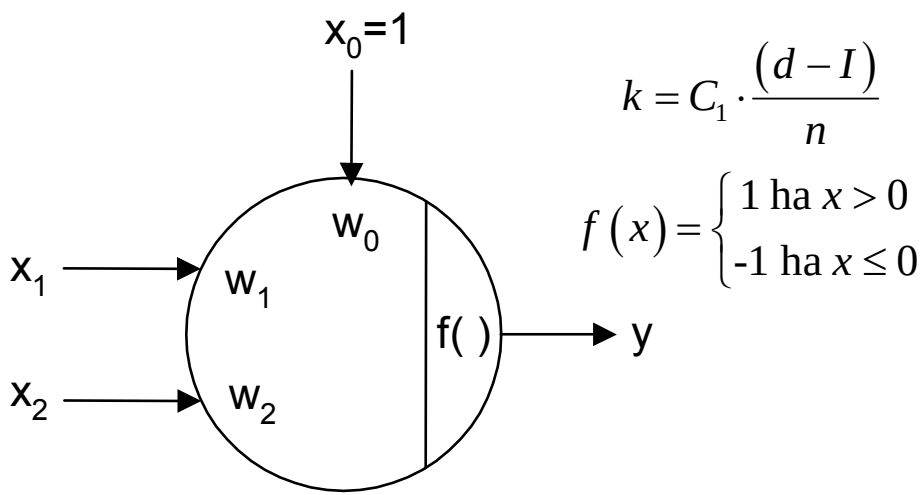
w_0	w_1	w_2	x_1	x_2	d	i	y	e
0.08	0.08	0.08	1	0	0	0.16	1	1
-0.02	-0.02	0.08	0	1	0	0.06	1	1
-0.12	-0.02	-0.02	1	1	1	-0.16	0	1
-0.02	0.08	0.08	1	0	0	0.06	1	1
-0.12	-0.02	0.08	0	1	0	-0.04	0	0
-0.12	-0.02	0.08	1	1	1	-0.06	1	1
-0.02	0.08	0.18	1	0	0	0.06	1	1
-0.12	-0.02	0.18	0	1	0	0.06	1	1
-0.22	-0.02	0.08	1	1	1	-0.16	0	1
-0.12	0.08	0.18	1	0	0	-0.04	0	0
-0.12	0.08	0.18	0	1	0	0.06	1	1
-0.22	0.08	0.08	1	1	1	-0.06	0	1
-0.12	0.18	0.18	1	0	0	0.06	1	1
-0.22	0.08	0.18	0	1	0	-0.04	0	0
-0.22	0.08	0.18	1	1	1	0.04	1	0
-0.22	0.08	0.18	1	0	0	-0.14	0	0
-0.22	0.08	0.18	0	0	0	-0.22	0	0

Adaline - Példa

Az Adaline felépítése lényegében megegyezik a perceptronéval, azzal a különbséggel, hogy itt a be- és kimenetek értéke {-1,+1 } és a következő Widrow-Hoff tanulási szabályt alkalmazzuk:

$$w_i^{'} = w_i + \frac{C}{n} \cdot \left(d - I\right) \cdot x_i$$

Látható, hogy itt a hiba valamekkora részét (C-től függően) egyenletesen elosztjuk az összes súlytényező között és minden lépésben mindegyik értékét ennek megfelelően változtatjuk.



$$k = C_1 \cdot \frac{\left(d - I\right)}{n}$$

$$f\left(x\right) = \begin{cases} 1 \text{ ha } x > 0 \\ -1 \text{ ha } x \le 0 \end{cases}$$

$$k = C_1 \cdot \frac{(d - I)}{n}$$

$$f(x) = \begin{cases} 1 \text{ ha } x > 0 \\ -1 \text{ ha } x \le 0 \end{cases}$$

Adaline - Példa

C = 0.5

e = d – I

e’ = d – y

A C értékét egynél kisebbre (C=0.5) választva könnyedén megtaníthatjuk az ÉS műveletet

w_0	w_1	w_2	x_1	x_2	d	I	y	e	e'
0.20	-0.30	-0.10	-1	1	-1	0.40	1	-1.40	-2
-0.03	-0.07	-0.33	-1	-1	-1	0.37	1	-1.37	-2
-0.26	0.16	-0.11	1	-1	-1	0.01	1	-1.01	-2
-0.43	-0.01	0.06	1	1	1	-0.37	-1	1.37	2
-0.20	0.22	0.29	-1	1	-1	-0.13	-1	-0.87	0
-0.34	0.37	0.15	-1	-1	-1	-0.86	-1	-0.14	0
-0.37	0.39	0.17	1	-1	-1	-0.15	-1	-0.85	0
-0.51	0.25	0.31	1	1	1	0.05	1	0.95	0
-0.35	0.41	0.47	-1	1	-1	-0.29	-1	-0.71	0

Ellenőrzött tanulás

Tanulási szabályok: Backpropagation (hibavisszaterjesztés)

A backpropagation a többrétegű hálózatoknál alkalmazott hibavisszaterjesztés módszere. Az elemi processzálóegység a perceptronhoz hasonló felépítésű, azzal a különbséggel, hogy a kimenet a rejtett rétegeken

$$y_j^{[s]} = f\left(I_j^{[s]}\right) = \frac{1}{1+e^{\left(-I_j^{[s]}\right)}}$$

szigmoid karakterisztika adja meg, ahol s-sel a réteg számát jelöljük. A tanítás algoritmusa a következő:

1. Az x bemeneti vektorból kiindulva, rétegenként kiszámoljuk a kimeneteket, s végül eljutunk a hálózat y kimenetéhez.

2. A kimeneti réteg minden processzáló elemére előbb kiszámítjuk a lokális hibát: $e_j=d_j - y_j$, majd ennek felhasználásával a hozzátartozó súlytényezők megváltozását: $\Delta w_{ij}^{[s]} = C \cdot e_j^{[s]} \cdot y_i^{[s-1]}$ (1)

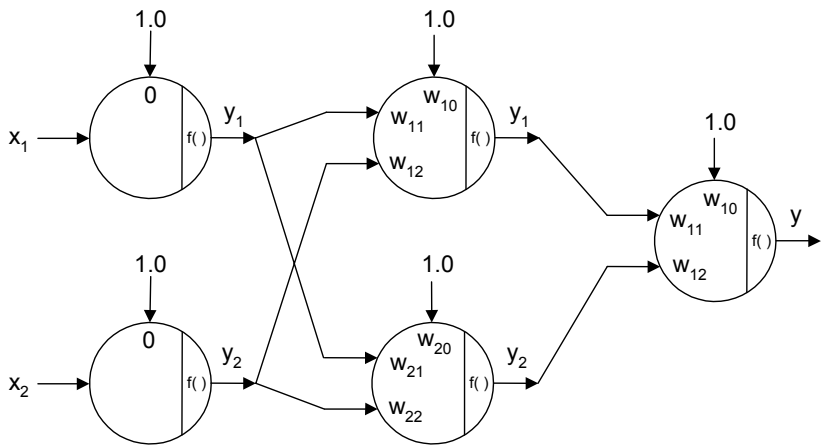
Tanulási szabályok: Backpropagation (hibavisszaterjesztés)

3. Az összes rejtett réteg minden processzáló elemére hátulról előre haladva számoljuk ki előbb a lokális hibákat:

$$e_j^{[s]} = y_j^{[s]}\left(1 - y_j^{[s]}\right) \cdot \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} w_{kj}^{[s+1]}$$

majd pedig a $\Delta w_{ij}^{[s]} = C \cdot e_j^{[s]} \cdot y_i^{[s-1]}$ egyenlettel a súlytényező-változást.

4. Módosítjuk a hálózat összes súlytényezőjét a kapott értékekkel.



A hibavisszaterjesztés módszere (részletesebben)

Adott az 1-3 rejtett rétegű előrecsatolt struktúra, $f(z)=\frac{1}{1+e^{-z}}$ szigmoid karakterisztikájú, folytonos működésű elemekkel.

A leíró egyenletek :

$$I_j^{[s]}=\sum_{i=1}^{n} w_{ij}^{[s]} y_i^{[s-1]};\quad y_j^{[s]}=f\left(I_j^{[s]}\right)=\frac{1}{1+e^{-I_j^{[s]}}};\quad s=1,2,...,r\quad (r=\max\ 4)$$

Feltesszük, hogy:

- Létezik egy globális hibafüggvény E. Csak ennyit, a formáját sem írjuk elő, csak azt, hogy: $E=\hat{E}\left(I_j^{[1]},I_j^{[2]},...,I_j^{[r]}\right);\ j=1,2,...,n^{[s]}$,

amely tehát az $I_j^{[1]}$-ek függvénye, továbbá

- A lokális hiba legyen $e_j^{[s]}=-\frac{\partial E}{\partial I_j^{[s]}}$ (gradiens tipus) valamint
- A kimeneti (r-edik) rétegen a lokális hiba $e_j^{[r]}$ adott. Pl. lehet $d_j-y_j^{[r]}$.

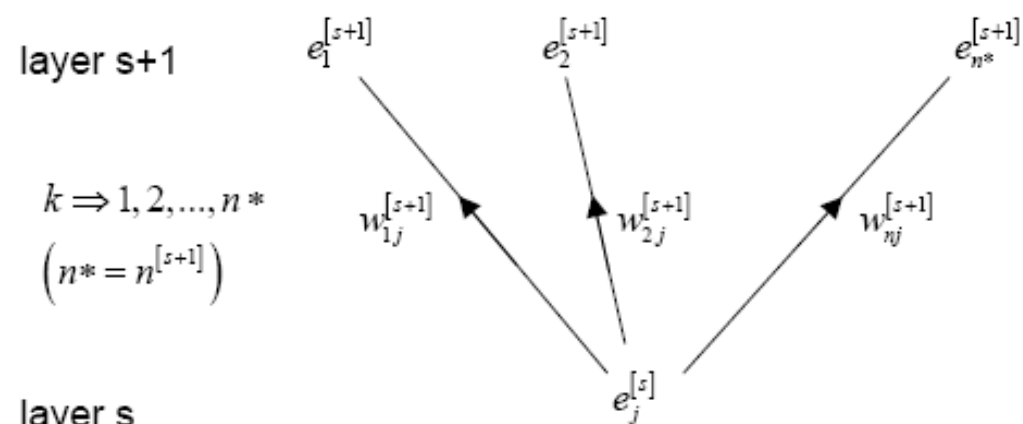
A hibavisszaterjesztés módszere

- Alaplépés:
  Ha adottak a lokális hibák az (s+1)-edik rétegen, akkor kiszámítjuk a lokális hibákat az s-edik rétegen:

$$e_j^{[s]} = -\frac{\partial E}{\partial I_j^{[s]}} = \sum_k -\frac{\partial E}{\partial I_k^{[s+1]}} \frac{\partial I_k^{[s+1]}}{\partial I_j^{[s]}} = \sum_k -\frac{\partial E}{\partial I_k^{[s+1]}} \frac{\partial I_k^{[s+1]}}{\partial y_j^{[s]}} \frac{\partial y_j^{[s]}}{\partial I_j^{[s]}} = f'\left(I_j^{[s]}\right) \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} w_{kj}^{[s+1]}$$

tehát a hibavisszaszámító egyenlet:

$$e_j^{[s]} = y_j^{[s]}\left(1 - y_j^{[s]}\right) \cdot \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} w_{kj}^{[s+1]}$$



A hibavisszaterjesztés módszere

A további levezetés ebből ismét a láncszabály alkalmazásával történik:

Standard legmeredekebb csökkenés módszerével (c: tanuló együttható):

$$w_{ij}^{[s]} = -c\frac{\partial E}{\partial w_{ji}^{[s]}}$$

Ahol:

$$\frac{\partial E}{\partial w_{ji}^{[s]}} = \frac{\partial E}{\partial I_j^{[s]}}\frac{\partial I_j^{[s]}}{\partial w_{ji}^{[s]}} = -e_j^{[s]} \cdot y_i^{[s-1]}$$

A tanulási szabály tehát:

$$\Delta w_{ji}^{[s]} = c \cdot e_j^{[s]} \cdot y_i^{[s-1]} \quad \left(+m_{ij}\right)$$

Legyen a globális hibafüggvény: $E = \frac{1}{2}\sum_k \left(d_k - y_k^{[o]}\right)^2$, ahol o a kimeneti réteg, itt f(z)=z, ezért

$$\downarrow \qquad \downarrow$$

$$e_k^{[o]} = -\frac{\partial E}{\partial I_k^{[o]}} = -\frac{\partial E}{\partial y_k^{[o]}} = --\left(d_k - y_k^{[o]}\right)$$

A hibavisszaterjesztés módszere, számítás

Számítás:

1. Adottak az input (u) → kívánt output (d) párok és a $w_{jk}^{[s]}$ kezdeti súlytényező értékek. u alapján kiszámítjuk az y_j-ket minden réteg minden PE-jére.

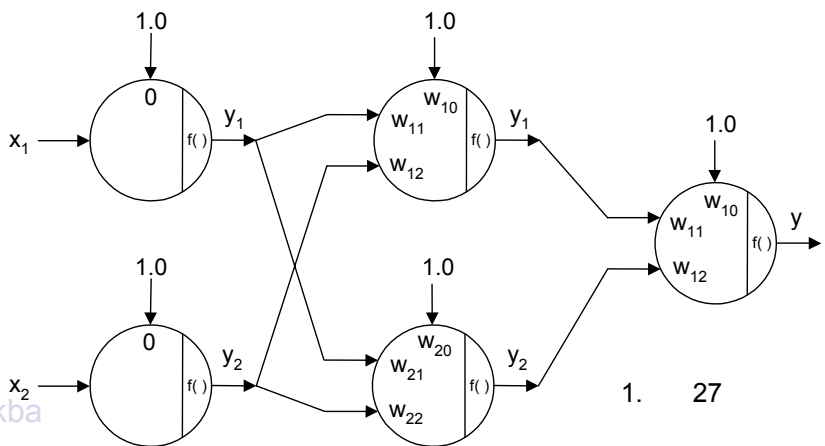
Tanulás:

2. A kimeneti rétegen f(z)=z, így $e_j^{[o]} := \left(d_j - y_j^{[o]}\right)$ és $\Delta w_{ji}^{[o]} = c \cdot e_j^{[o]} \cdot y_i$.

3. Minden rejtett rétegben kiszámítjuk a hibát a hibavisszaszámító egyenlettel $e_j^{[s]} = y_j^{[s]}\left(1 - y_j^{[s]}\right) \cdot \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} w_{kj}^{[s+1]}$

lépésenként az első rejtett rétegig és kiszámítjuk ugyanilyen sorrendben a súlytényező változásokat (illetve ezekből az új súlytényező értékeket):

$$\Delta w_{ji}^{[s]} = c \cdot e_j^{[s]} \cdot y_i^{[s-1]} \quad \left(+m_{ij}\right)$$



A hibavisszaterjesztés módszere, számítás

$w_{10}^{[1]}$	$w_{11}^{[1]}$	$w_{12}^{[1]}$	$w_{20}^{[1]}$	$w_{21}^{[1]}$	$w_{22}^{[1]}$	$w_{10}^{[2]}$	$w_{11}^{[2]}$	$w_{12}^{[2]}$	x_1	x_2	d	i	y	e	$e_1^{[1]}$	$e_2^{[1]}$
-2.00	0.00	2.00	1.00	5.00	3.00	2.00	4.00	-1.00	0	1	0	3.02	0.95	0.95	-0.95	0.02
-2.95	0.00	1.05	1.02	5.00	3.02	1.05	3.52	-1.94	0	0	1	-0.20	0.45	0.55	0.09	-0.21
-2.86	0.00	1.05	0.81	5.00	3.02	1.60	3.55	-1.53	1	0	0	0.26	0.56	-0.56	-0.10	0.00
-2.96	-0.10	1.05	0.81	5.00	3.02	1.03	3.52	-2.10	1	1	1	-0.65	0.34	0.66	0.24	0.00
-2.73	0.14	1.29	0.81	5.00	3.02	1.69	3.60	-1.44	0	1	0	0.97	0.73	-0.73	-0.40	0.02
-3.13	0.14	0.88	0.83	5.00	3.04	0.96	3.46	-2.15	0	0	1	-0.39	0.40	0.60	0.08	-0.27
-3.05	0.14	0.88	0.56	5.00	3.04	1.56	3.48	-1.73	1	0	0	0.01	0.50	-0.50	-0.09	0.00
-3.13	0.05	0.88	0.57	5.01	3.04	1.06	3.46	-2.23	1	1	1	-0.83	0.30	0.70	0.22	0.00
-2.92	0.27	1.10	0.57	5.01	3.04	1.75	3.53	-1.54	0	1	0	0.75	0.68	-0.68	-0.29	0.03
-3.20	0.27	0.81	0.59	5.01	3.07	1.07	3.43	-2.20	0	0	1	-0.21	0.45	0.55	0.07	-0.28
-3.13	0.27	0.81	0.31	5.01	3.07	1.63	3.45	-1.84	1	0	0	-0.02	0.49	-0.49	- 0.09	0.00
-3.22	0.18	0.81	0.32	5.01	3.07	1.13	3.45	-2.34	1	1	1	-0.87	0.30	0.70	0.21	0.00
-5.96	4.11	3.87	-2.27	5.83	5.32	2.89	6.85	-6.26	0	1	0	-2.34	0.09	-0.09	-0.06	0.02
-6.02	4.11	3.81	-2.25	5.83	5.34	2.80	6.84	-6.35	0	0	1	2.21	0.90	0.10	0.00	-0.05
-6.02	4.11	3.81	-2.30	5.83	5.34	2.90	6.84	-6.35	1	0	0	-2.38	0.08	-0.08	-0.06	0.01
-6.09	4.04	3.81	-2.29	5.85	5.34	2.82	6.83	-6.43	1	1	1	2.22	0.90	0.10	0.08	0.00
-6.00	4.13	3.89	-2.29	5.85	5.34	2.91	6.91	-6.33	0	1	0	-2.38	0.08	-0.08	-0.06	0.02
-6.06	4.13	3.84	-2.27	5.85	5.36	2.83	6.90	-6.41	0	0	1	2.24	0.90	0.10	0.00	-0.05
-6.06	4.13	3.84	-2.32	5.85	5.36	2.93	6.90	-6.40	1	0	0	-2.42	0.08	-0.08	-0.06	0.01
-6.12	4.06	3.84	-2.30	5.86	5.36	2.84	6.89	-6.48	1	1	1	2.26	0.91	0.09	0.08	0.00
-6.04	4.14	3.92	-2.30	5.86	5.36	2.94	6.97	-6.39	0	1	0	-2.42	0.08	-0.08	-0.05	0.02
-6.09	4.14	3.86	-2.28	5.86	5.39	2.95	6.96	-6.46	0	0	0	2.27	0.91	0.09	0.00	-0.05
-6.09	4.14	3.86	-2.33	5.86	5.39	2.95	6.96	-6.46	1	0	0	-2.45	0.08	-0.08	-0.06	0.01
-6.15	4.08	3.86	-2.32	5.88	5.39	2.87	6.95	-6.53	1	1	1	2.30	0.91	0.09	0.08	0.00

4-4. táblázat Back propagation tanulás C=1 tanulási együtthatóval. A tanulás kezdete, az üres sor után pedig a vége, amikor az e hiba abszolútértéke 0.1 alá csökken