Neurális hálózatok elméleti alapjai

TULICS MIKLÓS GÁBRIEL

TULICS@TMIT.BME.HU

Példa

X (tanult órák száma, aludt órák száma)	y (dolgozaton elért pontszám)
(5, 8)	80
(3, 5)	78
(5, 1)	82
(10, 2)	93
(4, 4)	75
(6, 6)	67
(1, 7)	90
…	

- Felügyelt tanulás
  - A modellt úgy tanítjuk, hogy a bemeneti értékekre ismerjük a kimeneti címkéket
- Regressziós probléma
  - A kimenet folyamatos
- ->Supervised regression

0-30 pont -> 1

30-50 pont-> 2

50-70 pont -> 3

70 – 90 pont -> 4

90 -100 pont -> 100

Osztályozási probléma

Lehetséges csoportosítások

A célváltozó címkéje alapján:

- ellenőrzött (felügyelt) osztályozók (supervised classifier): ahol a művelet végrehajtásában ismert tematikájú tananyagot használunk fel a döntést reprezentáló eljárás paramétereinek meghatározásához
- ellenőrizetlen (nem felügyelt) osztályozók (unsupervised classifier): ahol az eljárások az adatok belső sajátosságai alapján képezik a csoportokat és sorolják be az ismeretlen elemeket.

VAGY

- egyszerű osztályozók (pl. legkisebb távolság-módszer, Box-módszer)
- statisztikai osztályozók (pl. maximum likelihood eljárás, Bayes-döntés)
- mesterséges intelligencián alapuló (pl. neurális hálózatos) osztályozók.

Gépi tanulást, osztályozás

- Neurális hálózatok (artificial neural network, ANN)
- Rejtett Markov-modellek (hidden Markov model, HMM)
- Szupport vektor gépek (Support Vector Machines, SVM)

A neurális hálózatok előnyei közé tartozik, hogy jobban modelleznek nemlineáris feladatokat.

„Definíció”

Neurális hálónak nevezzük azt a hardver vagy szoftver alapú, párhuzamos elosztott működésre képes információfeldolgozó eszközt, amely azonos, vagy hasonló típusú, lokális feldolgozást végző műveleti elemek összekapcsolt rendszeréből áll, rendelkezik tanulási algoritmussal és előhívási algoritmussal

A neurális hálók működése tipikusan két fázisból áll.

1. Tanulási fázis: a hálózatba eltároljuk a megkívánt információfeldolgozó eljárást (tanulási fázis)
   - lassú, sikertelen tanulási szakaszokat is hordozhat
2. Előhívási fázis: a tárolt eljárás felhasználásával végezzük el az információfeldolgozást.
   - gyors feldolgozás

A két fázis rendszerint időben szétválik, bizonyos adaptív rendszerek az információ előhívási ciklusban is módosíthatják paramétereiket, tanulhatnak.

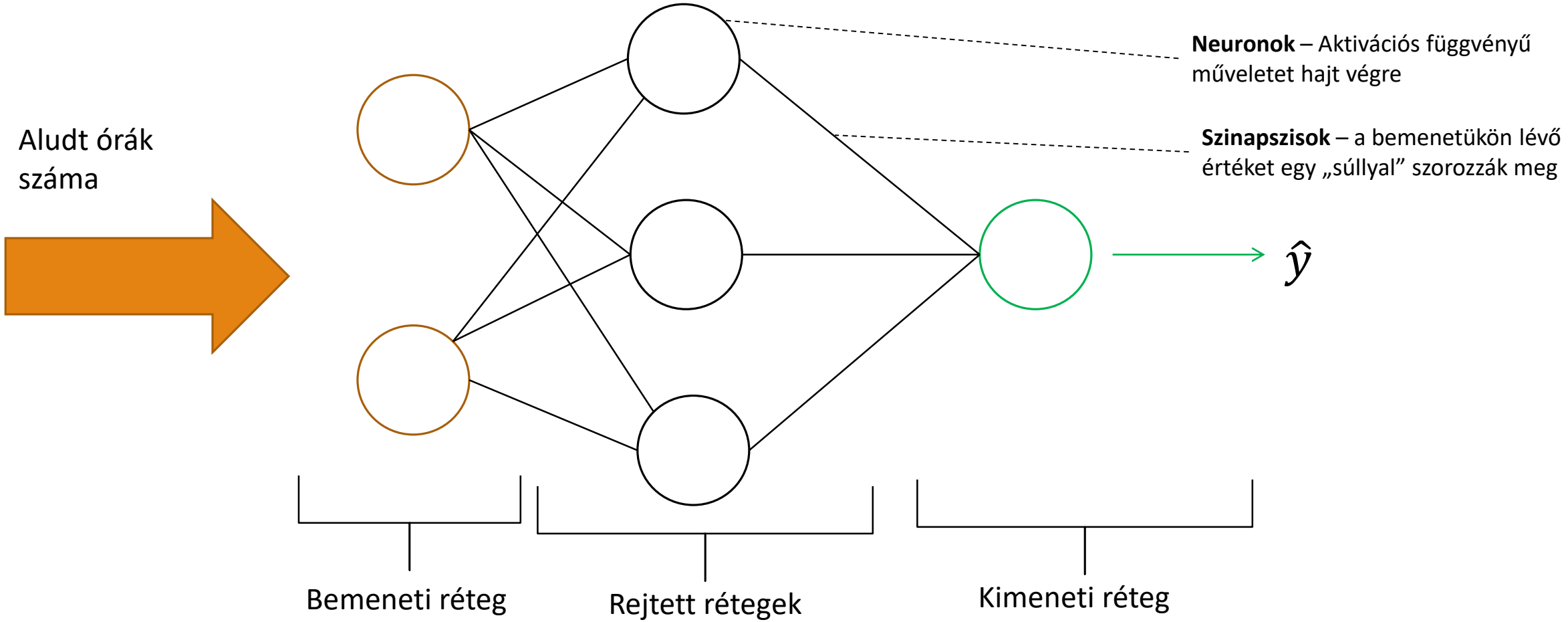
Nézzük meg az adatokat!

X (tanult órák száma, aludt órák száma)	y (dolgozaton elért pontszám)
(5, 8)	80
(3, 5)	78
(5, 1)	82
(10, 2)	93
(4, 4)	75
(6, 6)	67
(1, 7)	90
…	

- Bemeneteket órában adjuk meg, kimenetünk egy pontszám (1-től 100-ig)
- Az adatainkat ezért skálázni kell
  - [0,1] közé vagy standard normális eloszlásúra 0 várható értékkel, 1 szórásnégyzettel
- Skálázás:
  - $x^i_{norm} = \frac{x^i - \min(X)}{\max(X) - \min(X)}$;
  - $X_{scaled} = \frac{X}{\max(X)}; y_{scaled} = \frac{y}{\max(y)}$
- Standardizálás
  - $x^i_{norm} = \frac{x^i - \mu_x}{\sigma_x}$

Input	Stand	Norm
0	-1,3361	0
1	-0,801	0,2
2	-0,267	0,4
3	0,2672	0,6
4	0,801	0,8
5	1,336	1

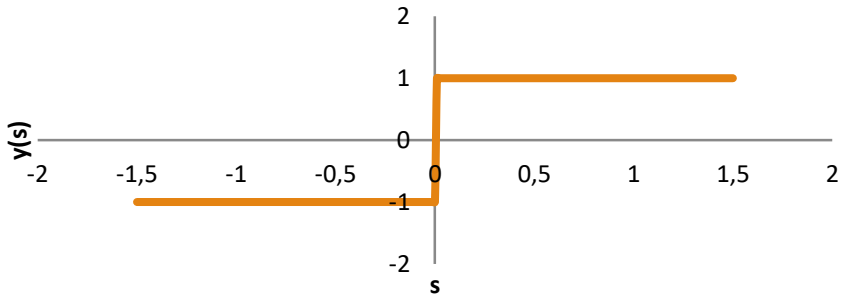
Egyszerű neurális háló



Aktivációs függvények /1

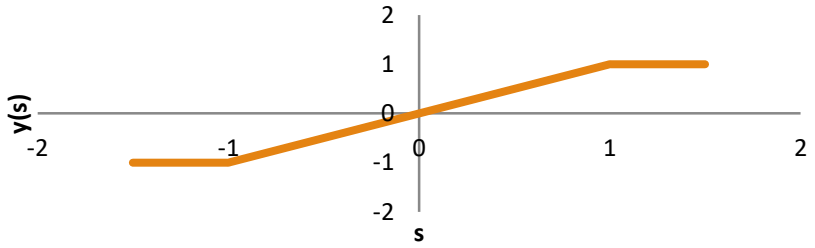
- Lépcsőfüggvény

$$y = \begin{cases} 1, & s > 0 \\ -1, & s \leq 0 \end{cases}$$



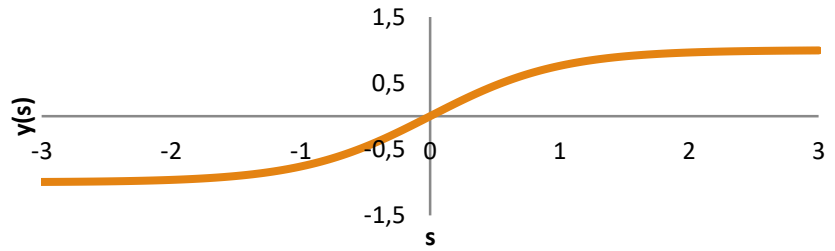
- Telítéses lineáris függvény

$$y = \begin{cases} 1, & s > 1 \\ s, & -1 \leq s \leq 1 \\ -1, & s \leq -1 \end{cases}$$



- Tangens hiberbolikusz függvény (K=2)

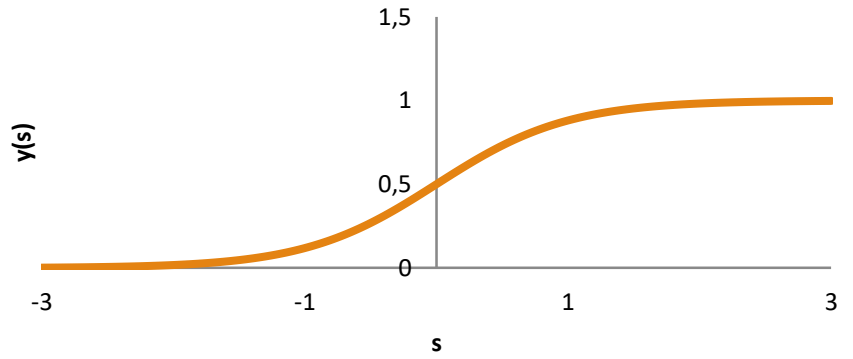
$$y = \frac{1 - e^{-Ks}}{1 + e^{-Ks}};\ K > 0$$



Aktivációs függvények /2

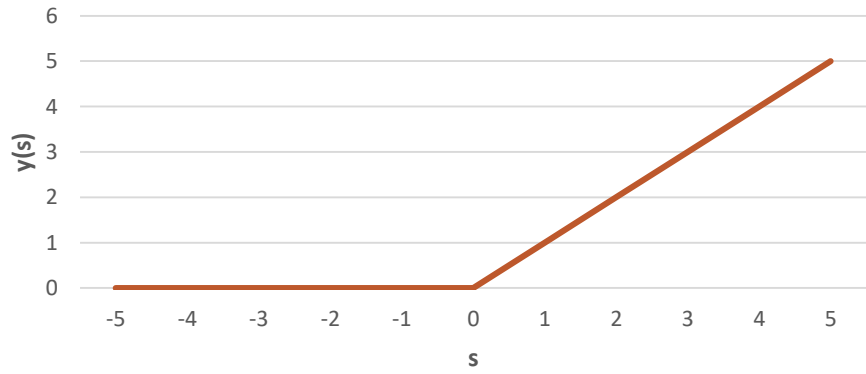
- Szigmoid függvény

$$y = \frac{1}{1 + e^{-Ks}};\ K > 0$$

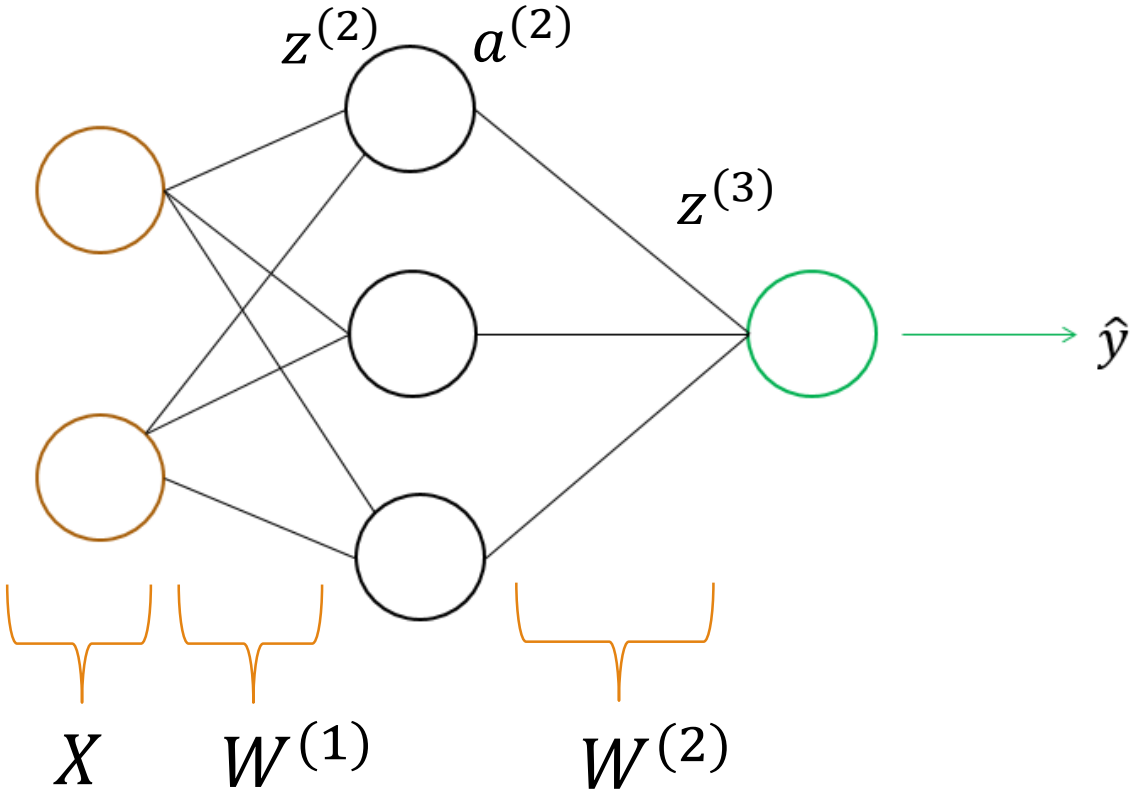


- ReLu (Rectifier)

$$y = \max(0, s)$$



Forward propagation



$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

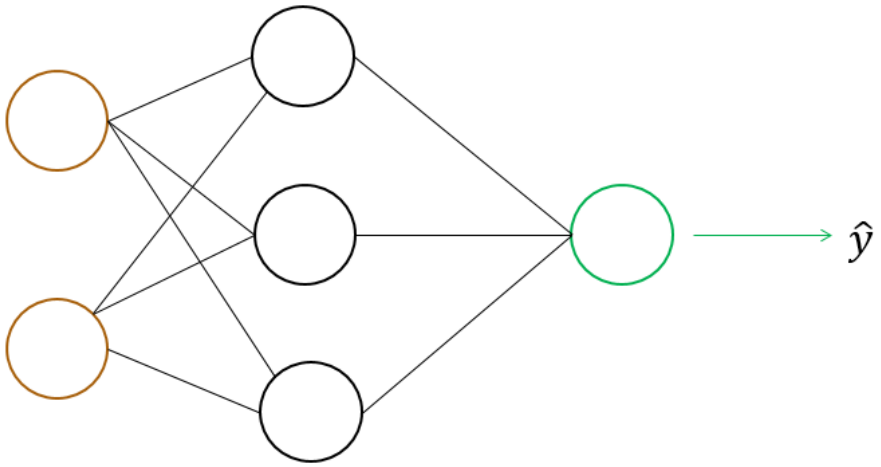
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

Nem fog nekünk jó pontszámokat jósolni a tanult/aludt órák száma alapján. Tanítani kell a hálózatot!

(Tanítás == költségfüggvény minimalizálás)

Gradient Descent



Meg kell határoznunk, hogy az előrejelzett eredményeink „mennyire rosszak” -> létrehozunk egy költségfüggvényt. Ez a költségfüggvény minél kisebb, annál jobb nekünk.

$$J = \sum (y - \hat{y})^2$$

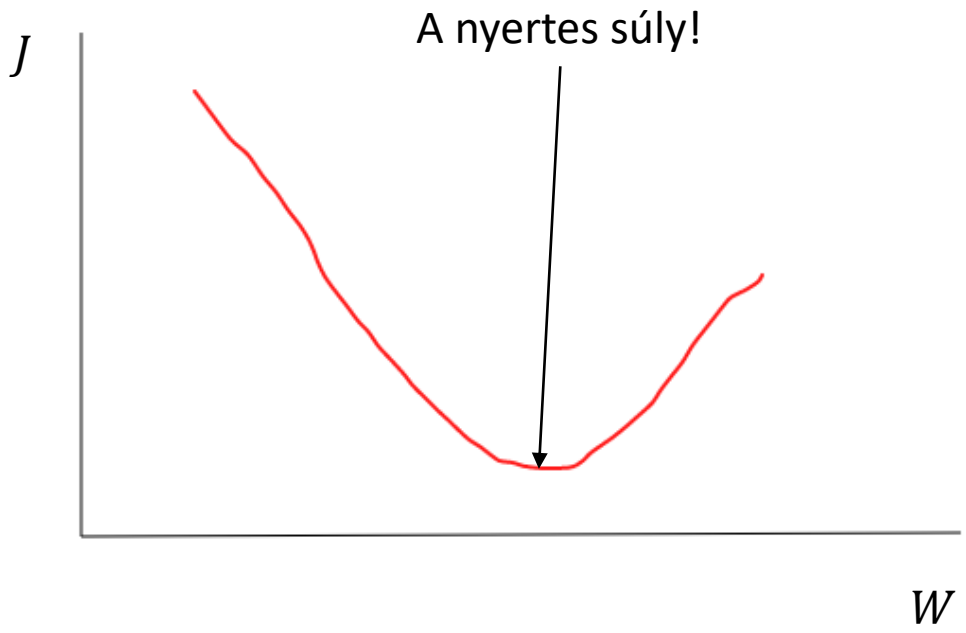
A rejtett rétegeket számát és a neuronjaink számát menet közben nem tudjuk változtatni.

A súlyaink az elején véletlen számok, utána azokat változtathatjuk!

Dimenzionalizás problémája

A hipotézis: A súlyoknak van olyan kombinációja amely minimalizálja a költségfüggvényt.

Miért nem próbáljuk ki az összes lehetséges súly kombinációját (brute force) az optimális kombináció megtalálásához?



Ha egy súlyérték meghatározása (1000 különböző súly kipróbálása) ~ 0,5 mp

Két súlyérték párhuzamos megtalálása (1000*1000 különböző súly kipróbálása) ~ 50 mp

Három súlyérték párhuzamos megtalálása (1000*1000*1000 különböző súly kipróbálása) ~ 11 óra

…

Jó lenne tudni, hogy merre csökken a költségfüggvény és csak arra keresni (növelni vagy csökkenteni) a súlyértéket

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum (y - \hat{y})^2 \quad (5)$$

$$J = \sum (y - f(z^{(3)}))^2$$

$$J = \sum (y - f(a^{(2)}W^{(2)}))^2$$

$$J = \sum (y - f(f(z^{(2)})\, W^{(2)}))^2$$

$$J = \sum (y - f(f(XW^{(1)})\, W^{(2)}))^2$$

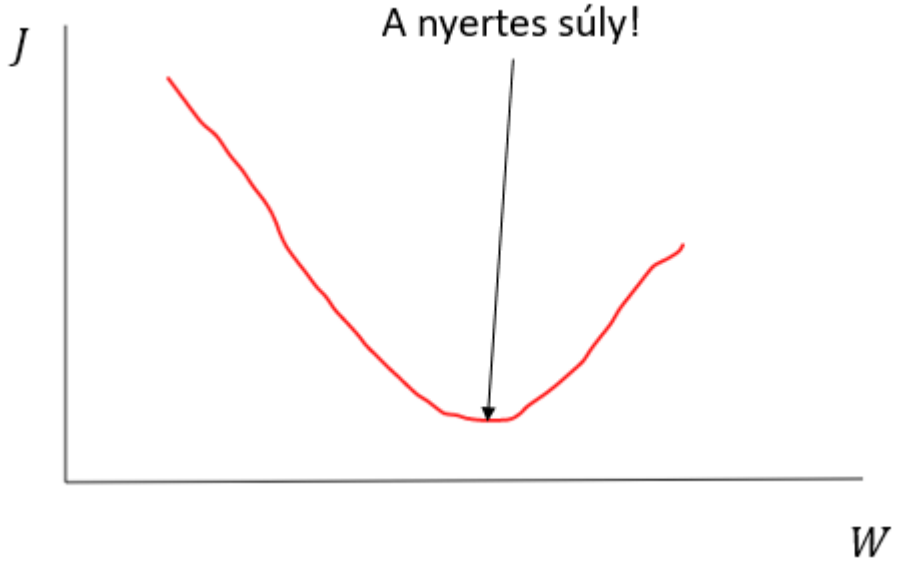
Nagyobb „lépéseket” is tehetünk lefelé és megállunk, amikor költségfüggvény nem csökken tovább.
Ezt nevezzük **Gradient Descent**-nek.

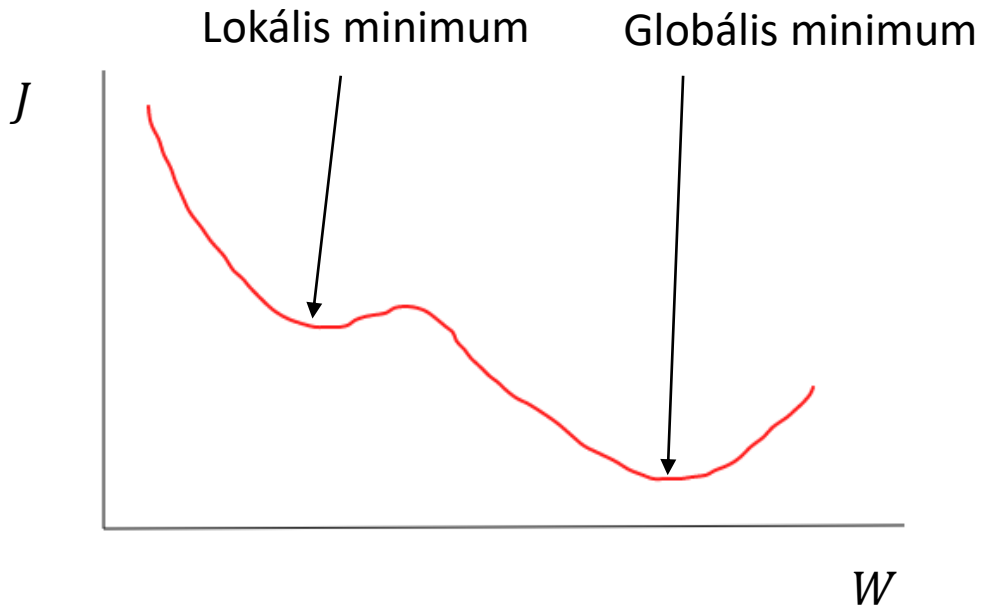
$$\frac{\partial J}{\partial W}$$

Azért parciális derivált, mert egyszerre csak egy irányban mentén nézzük a változást.

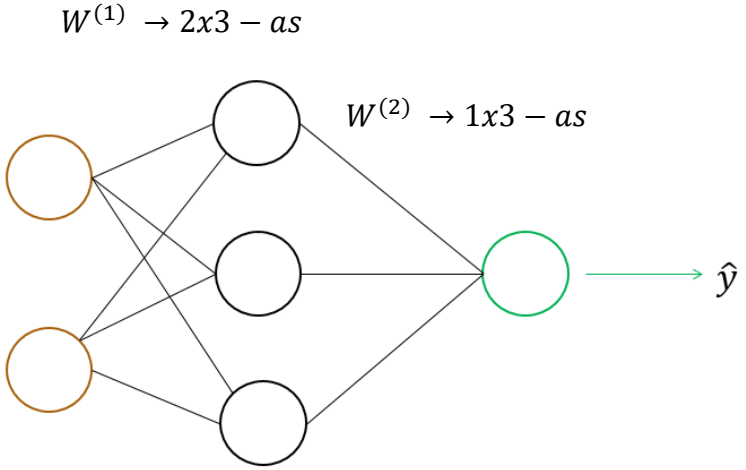
Ha a derivált értéke pozitív akkor a költségfüggvény felfelé halad, ha negatív, akkor lefelé.

Probléma lehet





Backpropagation



$$\frac{\partial J}{\partial W} = ?$$

$$W^{(1)} = \begin{matrix} W^{(1)}_{11} & W^{(1)}_{12} & W^{(1)}_{13} \\ W^{(1)}_{21} & W^{(1)}_{22} & W^{(1)}_{33} \end{matrix}$$

$$W^{(2)} = \begin{matrix} W^{(2)}_{11} \\ W^{(2)}_{21} \\ W^{(2)}_{31} \end{matrix}$$

$$\frac{\partial J}{\partial W^{(1)}} = \begin{matrix} \frac{\partial J}{\partial W^{(1)}_{11}} & \frac{\partial J}{\partial W^{(1)}_{12}} & \frac{\partial J}{\partial W^{(1)}_{13}} \\ \frac{\partial J}{\partial W^{(1)}_{21}} & \frac{\partial J}{\partial W^{(1)}_{22}} & \frac{\partial J}{\partial W^{(1)}_{23}} \end{matrix}$$

$$\frac{\partial J}{\partial W^{(2)}} = \begin{matrix} \frac{\partial J}{\partial W^{(2)}_{11}} \\ \frac{\partial J}{\partial W^{(2)}_{21}} \\ \frac{\partial J}{\partial W^{(2)}_{31}} \end{matrix}$$

$$J = \sum (y - f(f(XW^{(1)})\, W^{(2)}))^2 \qquad J = \sum (y - \hat{y})^2$$

$W^{(2)}$: $$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \sum (y - \hat{y})^2}{\partial W^{(2)}} \qquad \frac{\partial J}{\partial W^{(2)}} = \sum \frac{\partial (y - \hat{y})^2}{\partial W^{(2)}}$$

Deriválási szabályok

$$(cf)' = cf', \quad (f \pm g)' = f' \pm g', \quad (fg)' = f'g + fg', \quad \left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2}$$

$$(f \circ g)'(x) = f'(g(x))g'(x), \quad f^{-1\prime}(x) = \frac{1}{f'(f^{-1}(x))}$$

$$\frac{\partial J}{\partial W^{(2)}} = -2(y - \hat{y}) \frac{\partial \hat{y}}{\partial W^{(2)}}$$

$$\hat{y} = f(z^{(3)}) \qquad (4)$$

$$\frac{\partial J}{\partial W^{(2)}} = -2(y - \hat{y}) \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(2)}}$$

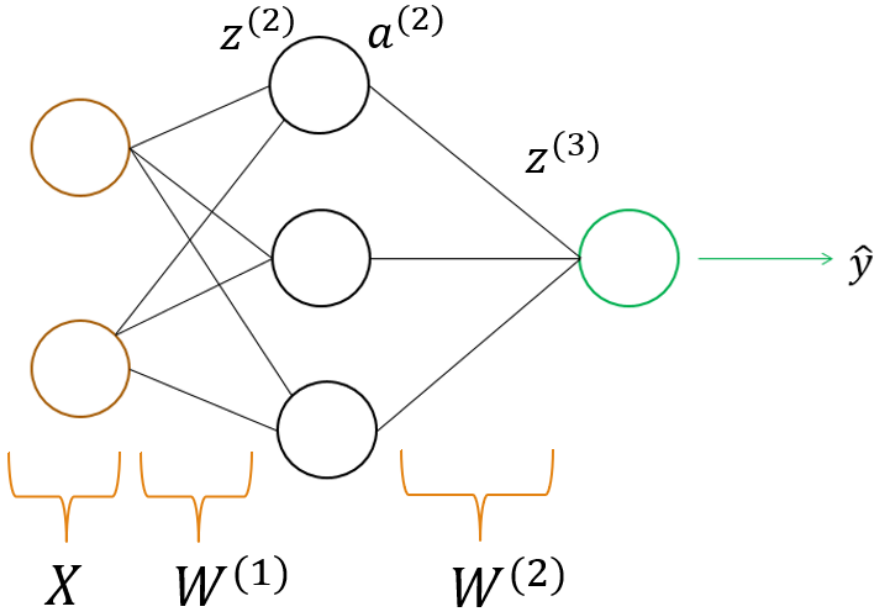
$$\frac{\partial J}{\partial W^{(2)}} = -2(y - \hat{y}) \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(2)}}$$

↓ Aktivációs függvény deriváltja

$$\frac{\partial J}{\partial W^{(2)}} = -2(y - \hat{y}) f'(z^{(3)}) \frac{\partial z^{(3)}}{\partial W^{(2)}}$$

↓ „A szinapszisok aktivitása”

$$\frac{\partial J}{\partial W^{(2)}} = -2(y - \hat{y}) f'(z^{(3)})\, a^{(2)}$$



$$z^{(3)} = a^{(2)} W^{(2)} \qquad (3)$$

$W^{(1)}$:

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \sum (y - \hat{y})^2}{\partial W^{(1)}} \qquad \frac{\partial J}{\partial W^{(1)}} = \sum \frac{\partial (y - \hat{y})^2}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -2(y - \hat{y}) \frac{\partial \hat{y}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -2(y - \hat{y}) \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -2(y - \hat{y}) f'(z^{(3)}) \frac{\partial z^{(3)}}{\partial W^{(1)}}$$

Oké, de hogyan változtassuk a két súlymátrixot?

Ha a derivált értékeket hozzáadjuk az eredeti súlyértékekhez, a költségfüggvényünk növekedni fog.

Ha viszont kivonjuk őket, akkor a költségfüggvény csökken!

$$W^{(1)} = W^{(1)} - skalár \cdot \frac{\partial J}{\partial W^{(1)}}$$

$$W^{(2)} = W^{(2)} - skalár \cdot \frac{\partial J}{\partial W^{(1)}}$$

Learning rate

Kitekintés / Mitől deep learning?

- Gépi tanuló algoritmusok struktúrált összessége
- Rendszerint több rejtett réteg
- Az adatok különböző szintű absztrakciója

Néhány „tipikus” háló réteg:

- Előrecsatolt rétegek (Fully Connected, FC)
  - Osztályozási és regressziós feladatokra
- Konvolúciós rétegek (Convolutional Neural Net, CNN)
  - Jellemző tanulás: feature extraction vs feature learning
  - 1D, 2D és 3D konvolúció
- Rekurrens rétegek (Long Short-Term Memory, LSTM)
  - Időbeliség modellezésére

Kitekintés / Legismesmertebb kutatók

Geoffrey Hinton

Emeritus Professor of Computer Science, University of Toronto & Engineering Fellow, Google Inc.

machine learning, neural networks, artificial intelligence, cognitive science, computer science

„Learning internal representations by error-propagation” (1986) David E Rumelhart, Geoffrey E Hinton, Ronald J Williams

Yann LeCun

Director of AI Research at Facebook & Silver Professor at the Courant Institute, New York University
AI, machine learning, computer vision, robotics, image compression
„Gradient-based learning applied to document recognition” (1998) Yann LeCun, Léon Bottou, **Yoshua Bengio**, Patrick Haffner

Yosua Bengio

Professor, U. Montreal (Computer Sc. & Op. Res.), MILA, CIFAR, CRM, REPARTI, GRSNC
Machine learning, deep learning, artificial intelligence

Kitekintés / Keretrendszerek

Keras

- Python alapú, Theano és TensorFlow backend, könnyen olvasható kód, gyors fejlesztés, nem olyan mély szintű. (Google)

Tensorflow

- Elosztott tanításokhoz, mélyebben bele lehet nyúlni az architektúrába amit a Keras elfed, erős marketing és Google támogatás.

Torch7

- LUA, binárisra fordítható, leggyorsabb, Jól olvasható kód, Tudományos életben standard. (Facebook, Twitter, NVidia, IBM, IDIAP, stb.)

Köszönöm a figyelmet!

Tulics Miklós Gábriel
tulics@tmit.bme.hu