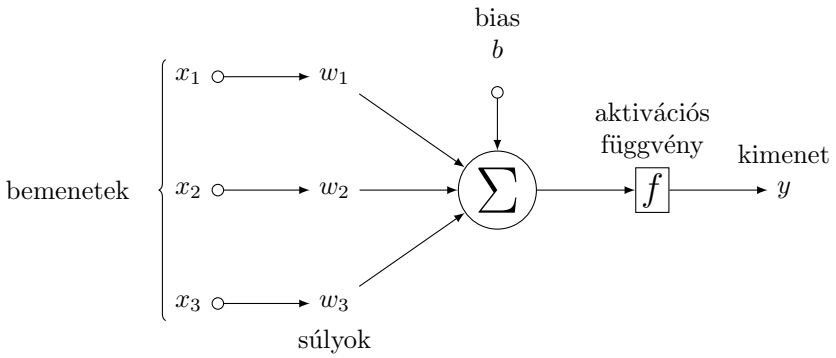
Neurális hálózatok képletgyűjtemény

oktatási segédlet a Mesterséges intelligencia c. tárgyhoz

Engedy István
BME-MIT

2016-09-01

1 Elemi neuron

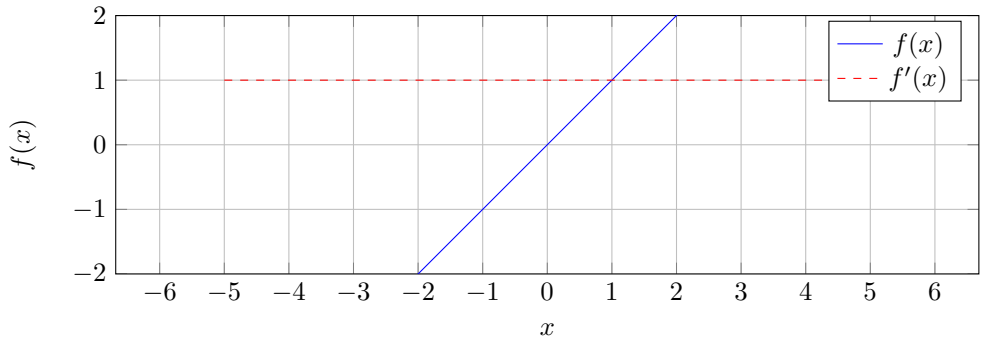


$$y = f(b + \sum_{i=1}^{N} w_i x_i)$$
$$y = f(w^T x + b)$$

1.1 Aktivációs függvények és deriváltjaik

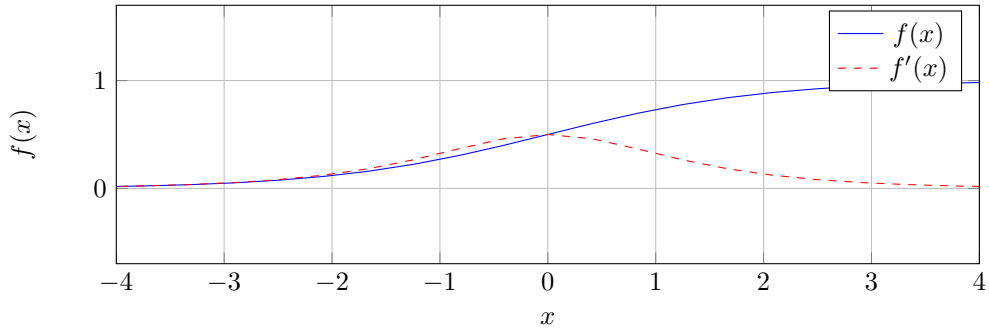
Kimeneti neuronok esetén tipikusan lineáris aktivációs függvényt használunk. Rejtett rétegbeli neuronok esetén a manapság a ReLU aktivációs függvényt szokás használni, a korábban használt szigmoid, illetve tanh aktivációs függvényekkel szemben, mivel a ReLU számos pozitív tulajdonsággal rendelkezik a másik két függvénnyel szemben.

1.1.1 Lineáris



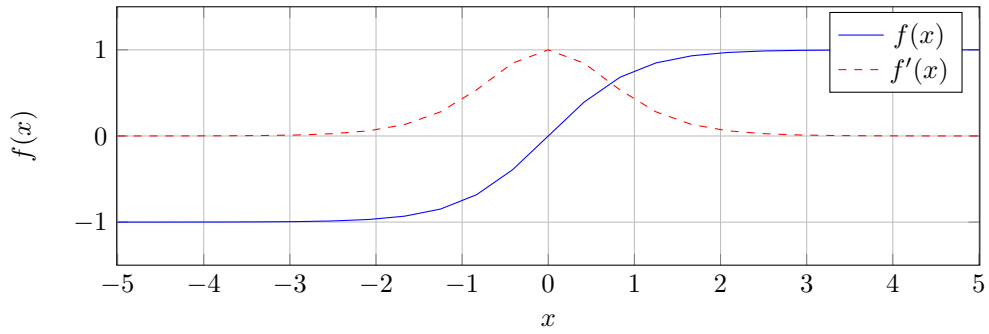
$$f(x) = x$$
$$f'(x) = 1$$

1.1.2 Szigmoid



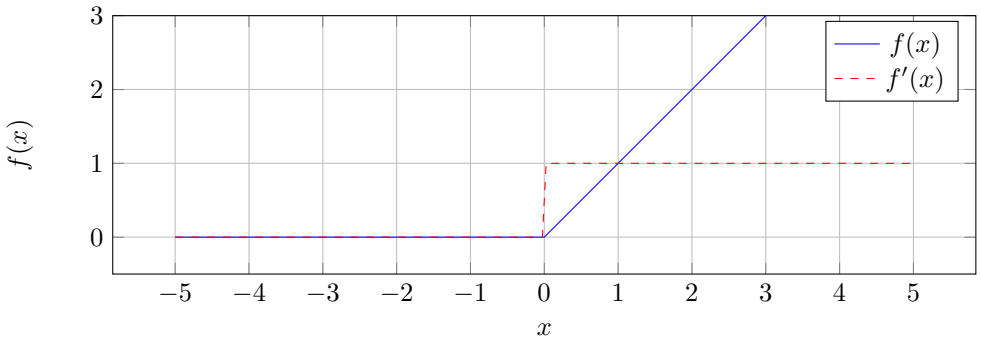
$$f(x) = S(x) = \frac{1}{1 + e^{-x}}$$
$$f'(x) = S(x)(1 - S(x)) = \frac{e^{-x}}{(1 + e^{-x})^2}$$

1.1.3 Hiperbolikus tangens



$$f(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
$$f'(x) = 1 - tanh^2(x)$$

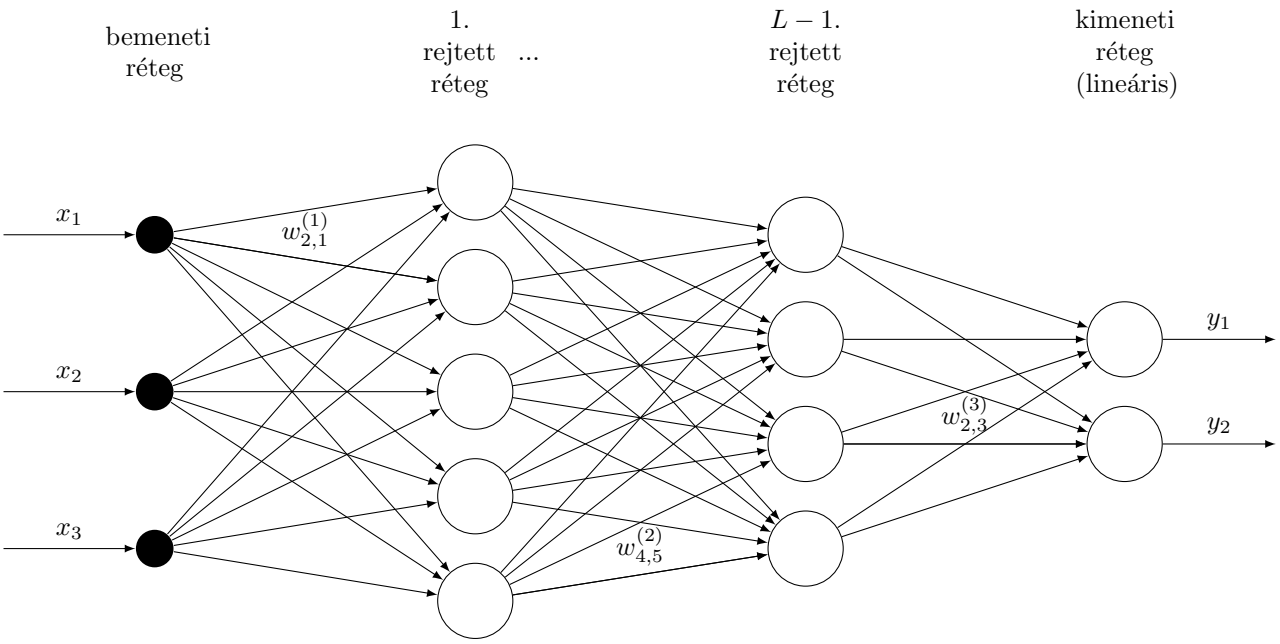
1.1.4 Rectified linear unit (ReLU)



$$f(x) = max(0, x) = \begin{cases} x, & \text{ha } x > 0 \\ 0 & \text{egyébként} \end{cases}$$

$$f'(x) = \begin{cases} 1, & \text{ha } x > 0 \\ 0 & \text{egyébként} \end{cases}$$

2 Neurális hálózat



2.1 Kimenet kiszámítása

Legyen $w_{i,j}^{(l)}$ az l. réteg i. neuronjának az a súlya, mely az előző réteg j. kimenetét súlyozza. Jelölje továbbá $b_i^{(l)}$ az l. réteg i. neuronjának a bias értékét. Legyen $s_i^{(l)}$ az l. réteg i. neuronjában a súlyozott összeg és a bias összege, míg $y_i^{(l)}$ ugyanennek a neuronnak a kimenete. A hálózat l. rejtett rétegében a neuronok számát jelölje N_l. Az i. bemenetet jelölje x_i, a háló j. kimenetét pedig y_j. Legyen $f()$ a rejtett rétegekben alkalmazott aktivációs függvény, a kimeneti rétegben ez legyen lineáris. A kimeneti réteg indexe legyen L.

$$
\begin{aligned}
s_i^{(l)} &= b_i^{(l)} + \sum_{j=1}^{N_l} w_{i,j}^{(l)} x_j^{(l)} \\
y_i^{(l)} &= f(s_i^{(l)}) \\
x_i^{(l)} &= \begin{cases} x_i, & \text{ha } l = 1 \\ y_i^{(l-1)} & \text{egyébként} \end{cases} \\
y_i &= s_i^{(L)}
\end{aligned}
$$

A fentiek leírhatóak mátrixos alakban is. Legyen $W^{(l)} = \begin{bmatrix} w_{1,1}^{(l)} & w_{1,2}^{(l)} & \cdots \\ w_{2,1}^{(l)} & w_{2,2}^{(l)} & \\ \vdots & & \ddots \end{bmatrix}$, $b^{(l)} = \begin{bmatrix} b_1^{(l)} \\ b_2^{(l)} \\ \vdots \end{bmatrix}$, $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix}$

$$
\begin{aligned}
s^{(l)} &= b^{(l)} + W^{(l)} x^{(l)} \\
y^{(l)} &= f(s^{(l)}) \\
x^{(l)} &= \begin{cases} x, & \text{ha } l = 1 \\ y^{(l-1)} & \text{egyébként} \end{cases} \\
y &= s^{(L)}
\end{aligned}
$$

Tehát a fenti neurális hálózat kimenetének kiszámítása:

$$y = b^{(3)} + W^{(3)} f(b^{(2)} + W^{(2)} f(b^{(1)} + W^{(1)} x))$$

$$y = \begin{bmatrix} b_1^{(3)} \\ b_2^{(3)} \end{bmatrix} + \begin{bmatrix} w_{1,1}^{(3)} & \cdots & w_{1,4}^{(3)} \\ w_{2,1}^{(3)} & \cdots & w_{2,4}^{(3)} \end{bmatrix} f\left(\begin{bmatrix} b_1^{(2)} \\ \vdots \\ b_4^{(2)} \end{bmatrix} + \begin{bmatrix} w_{1,1}^{(2)} & \cdots & w_{1,5}^{(2)} \\ \vdots & \ddots & \vdots \\ w_{4,1}^{(2)} & \cdots & w_{4,5}^{(2)} \end{bmatrix} f\left(\begin{bmatrix} b_1^{(1)} \\ \vdots \\ b_5^{(1)} \end{bmatrix} + \begin{bmatrix} w_{1,1}^{(1)} & w_{1,2}^{(1)} & w_{1,3}^{(1)} \\ \vdots & \ddots & \vdots \\ w_{5,1}^{(1)} & \cdots & w_{5,3}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) \right)$$

2.2 Tanítás

A neurális hálózat tanítása tanítómintákkal történik. Egy tanítóminta két részből áll: x bemenet, és d kívánt kimenet. A tanítás menete a következő:

1. A következő tanítóminta kiválasztása, (x, d)
2. A neurális hálózat kimenetének kiszámítása az aktuális súlyaival, $y = MLP(x, w)$

3. A hiba kiszámítása: $\varepsilon = d - y$
4. A költségfüggvény kiszámítása: $C = C(\varepsilon) = \varepsilon^2 = (d-y)^2$
5. Az x helyen a költségfüggvény paraméterek (súlyok, biasok) szerinti érzékenységének kiszámítása a hibavisszaterjesztés algoritmusával: $\frac{\partial C}{\partial w_{i,j}^{(l)}}\Big|_x$
6. A paraméterek módosítása a delta szabállyal: $\Delta w_{i,j}^{(l)} = -\mu \frac{\partial C}{\partial w_{i,j}^{(l)}}\Big|_x$

2.2.1 Hibavisszaterjesztés algoritmusa

A hibavisszaterjesztés során az éppen adott tanítóminta bemenetének megfelelő helyen a neurális hálózat kimenetéből és a kívánt válaszból származtatott költségfüggvénynek a paraméterek szerinti érzékenységét számítjuk ki, vagyis a költség paraméterek szerinti parciális deriváltjait. A backpropagation algoritmus egy hatékony módszer ezeknek a kiszámítására, mely a deriválás láncszabályát használja ki.

$$\frac{\partial C}{\partial w_{i,j}^{(l)}} = \frac{\partial C}{\partial \varepsilon}\frac{\partial \varepsilon}{\partial w_{i,j}^{(l)}} = \frac{\partial C}{\partial \varepsilon}\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w_{i,j}^{(l)}} = -2\varepsilon^T \frac{\partial y}{\partial w_{i,j}^{(l)}}$$

$$\frac{\partial y}{\partial w_{i,j}^{(l)}} = \underbrace{\frac{\partial y}{\partial s^{(L)}}\frac{\partial s^{(L)}}{\partial y^{(L-1)}}}_{\text{kimeneti réteg}} \underbrace{\frac{\partial y^{(L-1)}}{\partial s^{(L-1)}}\frac{\partial s^{(L-1)}}{\partial y^{(L-2)}}}_{\text{rejtett réteg}} \cdots \underbrace{\frac{\partial y^{(l)}}{\partial s^{(l)}}\frac{\partial s^{(l)}}{\partial w_{i,j}^{(l)}}}_{\text{a súlyt tartalmazó réteg}}$$

$$\frac{\partial y}{\partial w_{i,j}^{(l)}} = \underbrace{W^{(L)}}_{\text{kimeneti réteg}} \underbrace{f'(s^{(L-1)})W^{(L-1)}}_{\text{rejtett réteg}} \cdots \underbrace{f'(s^{(l)})x_j^{(l)}}_{\text{a súlyt tartalmazó réteg}}$$

Például:

$$\frac{\partial C}{\partial w_{2,1}^{(1)}} = -2\begin{bmatrix}\varepsilon_1 & \varepsilon_2\end{bmatrix}\begin{bmatrix} w_{1,1}^{(3)} & \cdots & w_{1,4}^{(3)} \\ w_{2,1}^{(3)} & \cdots & w_{2,4}^{(3)}\end{bmatrix} f'\left(\begin{bmatrix} s_1^{(2)} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_4^{(2)}\end{bmatrix}\right)\begin{bmatrix} w_{1,1}^{(2)} & \cdots & w_{1,5}^{(2)} \\ \vdots & \ddots & \vdots \\ w_{4,1}^{(2)} & \cdots & w_{4,5}^{(2)}\end{bmatrix} f'\left(\begin{bmatrix} 0 \\ s_2^{(1)} \\ 0 \\ \vdots\end{bmatrix}\right) x_1$$

A fenti összefüggést önmagában alkalmazni nem érdemes, mivel nagyon pazarlóan végeznénk számításokat: sokszor kiszámolnánk ugyanazokat a mátrix szorzatokat. Éppen ezt oldja meg a hibavisszaterjesztés algoritmusa, bizonyos részszámításokat eltárolunk, hogy azokat ne kelljen újra és újra kiszámolni az egyes súlyok szerinti parciális deriváltak számításakor.

Legyen

$$\delta_i^{(l)} = \begin{cases} \varepsilon_i, & \text{ha } l = L \\ \left(\sum_{j=1}^{N_{l+1}} \delta_j^{(l+1)} w_{j,i}^{(l+1)}\right) f'(s_i^{(l)}) & \text{egyébként.}\end{cases}$$

Ekkor az előző összefüggések felírhatóak a következő zárt alakban:

$$\frac{\partial C}{\partial w_{i,j}^{(l)}} = -2\delta_i^{(l)} x_j^{(l)}$$

$$\frac{\partial C}{\partial b_i^{(l)}} = -2\delta_i^{(l)}$$

Mindez felírható mátrixos formában is:

$$\delta^{(l)} = \begin{cases} \varepsilon^T, & \text{ha } l = L \\ \delta^{(l+1)} W^{(l+1)} f'(s^{(l)}) & \text{egyébként.} \end{cases}$$

$$\frac{\partial C}{\partial W^{(l)}} = -2{\delta^{(l)}}^T {x^{(l)}}^T$$

$$\frac{\partial C}{\partial b^{(l)}} = -2{\delta^{(l)}}^T$$

2.2.2 Súlymódosítások, a delta szabály

A legegyszerűbb, gradiens módszer során a súlyokat a kimeneti hiba súlyok szerinti deriváltjának μ bátorsági faktor-szorosával (tanulási tényező) módosítjuk. Léteznek ennél bonyolultabb súlymódosító eljárások, melyek gyorsabb tanulást eredményeznek, mint például a momentum módszer, vagy az ADAM módszer.

A súlyok tehát a $k+1$. tanítási iterációban a következő módon módosulnak:

$$w_{i,j}^{(l)}(k+1) = w_{i,j}^{(l)}(k) - \mu \frac{\partial C}{\partial w_{i,j}^{(l)}} = w_{i,j}^{(l)}(k) + 2\mu\delta_i^{(l)} x_j^{(l)}$$

$$b_i^{(l)}(k+1) = b_i^{(l)}(k) - \mu \frac{\partial C}{\partial b_i^{(l)}} = b_i^{(l)}(k) + 2\mu\delta_i^{(l)}$$

Vagy mátrixos alakban:

$$W^{(l)}(k+1) = W^{(l)}(k) - \mu \frac{\partial C}{\partial W^{(l)}} = W^{(l)}(k) + 2\mu{\delta^{(l)}}^T {x^{(l)}}^T$$

$$b^{(l)}(k+1) = b^{(l)}(k) - \mu \frac{\partial C}{\partial b^{(l)}} = b^{(l)}(k) + 2\mu{\delta^{(l)}}^T$$