Backpropagation, avagy hogyan tanulnak a neurális hálók

Learning Lunch 2024.03.20.

A mesterséges intelligencia töretlen fejlődése kapcsán egyre több embert érdekel az, hogy hogyan tanulnak a gépek.

A gépi tanulás egyik módja egy mesterséges intelligencia modellen, a multilayer perceptron-on alapul.

Multi-layer perceptron

Attól függetlenül, hogy ez a legegyszerűbb neurális háló, ennek is sokféle felhasználási területe van. Pl: Karakterfelismerés, szöveg nyelvének kitalálása

Az előadás anyagából kód is készült!

https://github.com/gkalocsai/MultiLayerPerceptron

A modellről

Biológiai ihletésű modell.

A modell legkisebb egységei a nodeok vagy neuronok. A nodeok rétegekbe szerveződnek és minden egyes node pontosan egy rétegnek része.

Minden nodenak van sorszáma, ami a rétegen belüli elhelyezkedést írja le, a kódban és az egyenletekben ez a sorszám indexként jelenik meg.

Háromféle réteg van ebben a modellben, bemeneti, kimeneti és rejtett rétegek. Bemeneti és kimeneti rétegből egy-egy van, rejtett rétegből lehet több.

A rétegek sorba vannak rendezve. Minden réteg ismeri a megelőzőjét és a rákövetkezőjét, ha van.

A bemeneti rétegben a node-ok egyszerűen számokat tartalmaznak. Ezek a számok azt jelzik, hogy az adott ponton (indexen) mennyire erős a bemeneti aktiváció.

Amikor nem tanítani szeretnénk a hálót, csak használni, akkor a működtetéshez csak ezeket a bemeneti aktivációkat kell megadni.

A rejtett rétegekben és a kimeneti rétegben a node-ok ugyanúgy épülnek fel.

Aktivációt ezek is tárolnak. Ezen felül ezek a node-ok kapcsolatban vannak a megelőző réteg összes nodejával és döntenek arról, hogy az előző réteg nodejainak kimenetét(aktivációját) milyen súllyal veszik figyelembe. Ezeket a súlyokat szintén a node-ban tároljuk. Ezeken a súlyokon felül a modellünk még egy extra súlyt tárol, egy eltolási értéket (bias), ami nincs kapcsolatban az

előző réteg egyik nodejának aktivációjával sem, de úgy tekintjük, mintha az előző rétegnek lenne egy olyan nulladik node-ja, aminek az aktivációja mindig pontosan 1 és ezzel szoroznánk fel ezt az extra súlyt. A neuron aktívabb/passzívabb lesz miatta.

A modellben a node-ok súlyainak véletlenszerűen adunk kezdőértéket.

A működés

Rétegenként haladunk a bemeneti rétegtől a kimeneti rétegig és rétegenként meghatározzuk az egyes nodeok aktivációját.

Egy node aktivációjának meghatározása két lépésből áll:

1. Kiszámolunk egy súlyozott összeget: az előző réteg aktivációit egyesével összeszorozzuk a node-ban tárolt súlyokkal.

Ezeket a szorzatokat összegezzük. Egy súlyhoz egy aktiváció tartozik.

2. Erre a súlyozott összegre alkalmazunk egy nemlineáris függvényt. Azt a modell létrehozásakor eldöntjük, hogy pontosan melyik függvényt használjuk.

Ez a függvény azért kell, hogy a modell bonyolultabb összefüggéseket is meg tudjon találni.

A kimeneti réteg aktivációit tekintjük a neurális háló válaszának.

Azt, hogy a neurális hálónk jól vagy rosszul működik, végső soron a súlyok nagysága határozza meg. A tanítás során csak a súlyok értékeit módosítjuk.

Hogyan alkossunk konkrét modellt konkrét problémákra?

Általánosságban igaz az, hogy a modellünk csak számokat fogad el és számokat ad vissza. A bemeneti adatot számsorozattá kell alakítanunk, a kimeneti számsorozathoz pedig jelentést kell rendelnünk.

1. Példa - Modell a kizáró vagy (eXclusive OR - XOR) logikai műveletre:

Az XOR művelet bemeneti és kimeneti értékei logikai típusúak. Két bemeneti érték van és egy kimeneti. A kimeneti érték akkor lesz igaz, ha a bemeneti értékek közül pontosan egy igaz. Ha két hamis, vagy két igaz értéket kap, akkor az eredmény hamis lesz.

A modell építésénél a két bemeneti értéknek megfeleltetünk egy-egy nodeot a bemeneti rétegben. A kimeneti rétegben egy node lesz.

A logikai igaz értéket az 1-es fogja jelenteni, a hamis pedig a 0 lesz. Legyen két rejtett réteg is 12 és 10 db. node-al.

2. Példa - kézzel írott számjegyek felismerése. Minden számjegy egy-egy szürkeárnyalatos képen van ábrázolva. A képek minden pixeléhez hozzá van rendelve egy 0 és 1 közötti szám, aszerint, hogy mennyire erősen világít az adott pixel. A képek 28*28 pixel méretűek, ezért

a bemeneti réteg 28*28 = 784 darab node-ot tartalmaz az aktivációjuk pedig rendre a hozzájuk rendelt n. pixel fényességét jelző érték. A kimeneti réteg 10 node-ot tartalmaz.

0-tól 9-ig minden számjegynek megfeleltetünk egy-egy node-ot. Azt szeretnénk ha a beadott képen lévő számjegyhez tartozó node értéke egyes lenne, a többi pedig 0.

Van két rejtett rétegünk is. A rejtett rétegekben a node-ok száma 64 és 32.

A tanítás

A tanítás példákkal, ún. tanító mintákkal történik. A tanító minta két részből áll: a bemeneti értékekből, és a bemeneti értékhez tartozó elvárt eredményből.

Ha megadjuk a bemeneti aktivációkat és működtetjük a neurális hálónkat, az fog adni valamilyen eredményt.

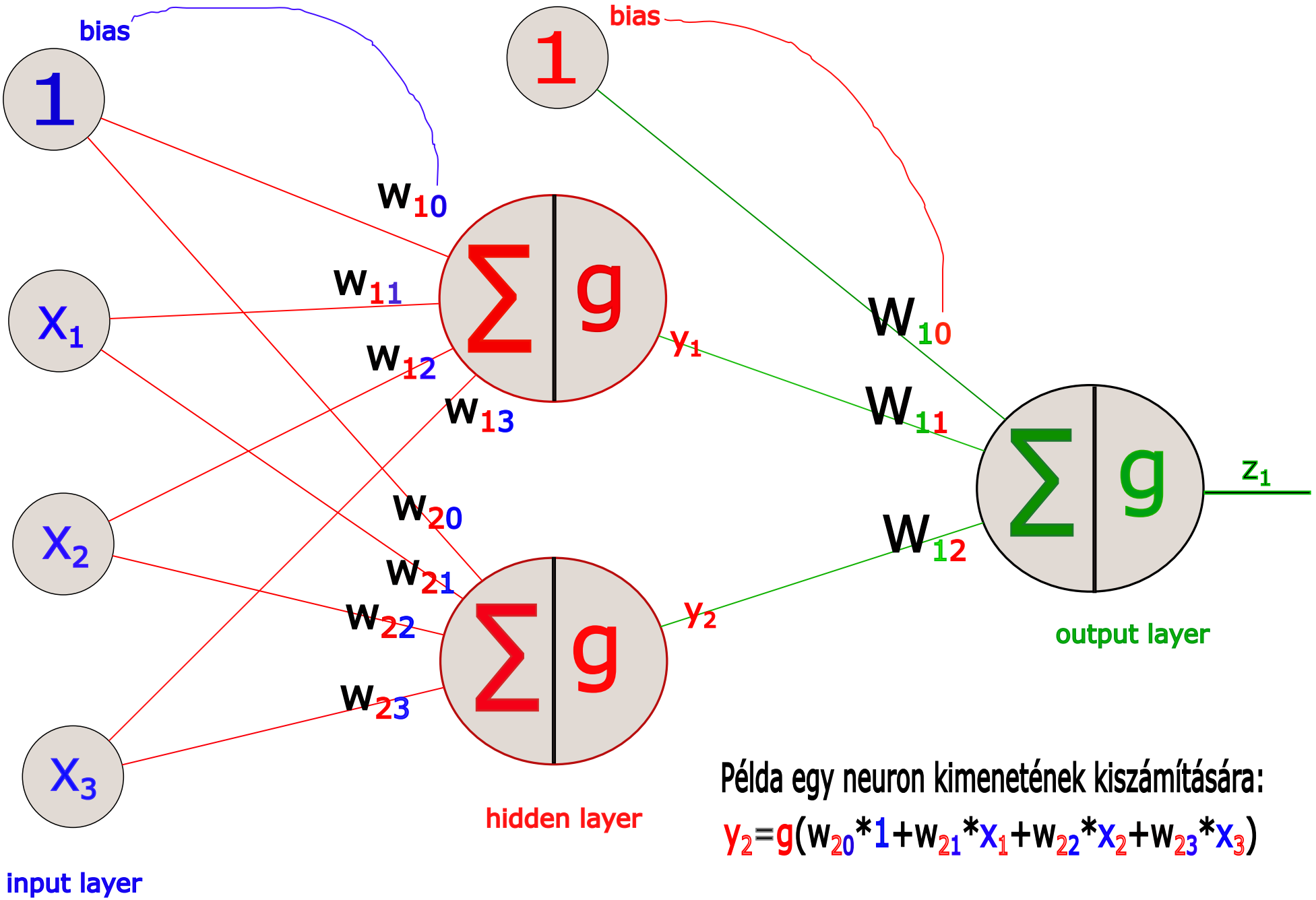
Ez az eredmény a tanítás előtt köszönőviszonyban sem lesz a vágyott eredménnyel.

A feltanított neurális hálónál az egyes rétegeket sorban használjuk az eredmény meghatározásához - az ábrán balról-jobbra - a tanítás viszont pont a másik irányból történik visszafelé - innen ered a backpropagation elnevezés.

Jobbról haladva első körben kiszámolhatjuk, hogy mekkora hibával dolgozik a neurális hálónk, ennek meghatározásához alkotunk egy hibafüggvényt.

Azt keressük, hogy a hibafüggvény értéke milyen súlyok mellett lesz minimális.

A súlyok értékét minden egyes tanító minta hatására módosítjuk egy kicsit, úgy hogy a hibafüggvényünk értékét az adott tanító mintánál csökkentsük!



Példa egy neuron kimenetének kiszámítására:

$y_2 = g(w_{20}*1 + w_{21}*x_1 + w_{22}*x_2 + w_{23}*x_3)$

Hiba/Költségfüggvény:

$$J = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2$$

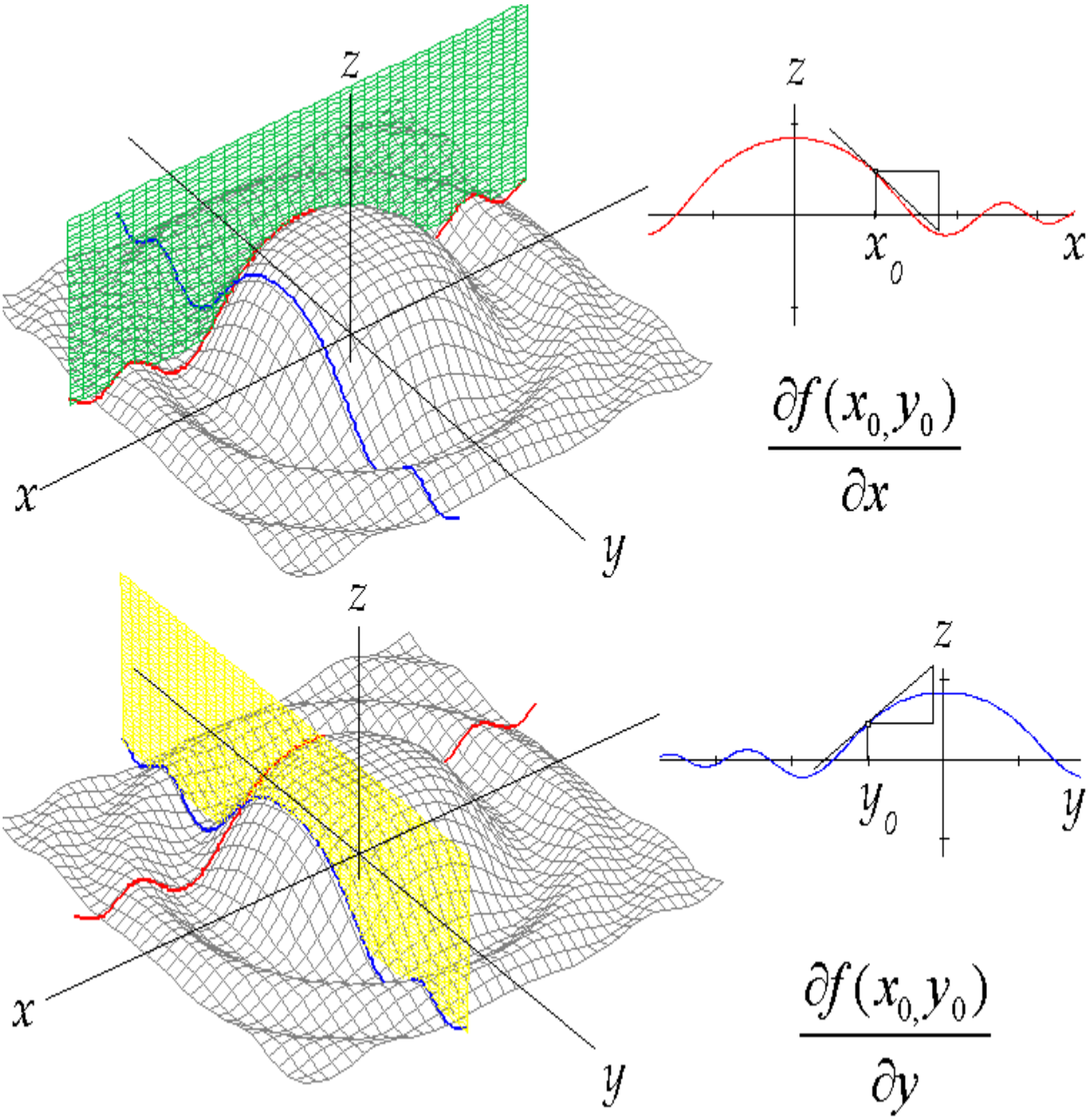
Azt, hogy a neurális hálónk jól vagy rosszul teljesít, végső soron a súlyok határozzák meg.

Azt szeretnék, ha a hibafüggvényünk értéke minimális lenne. Ehhez a súlyok értékét változtatjuk minden egyes tanításkor a következőképpen:

$$W_{pq} := W_{pq} - \eta\frac{\partial J}{\partial W pq}$$

A parciális deriváltat kétféle módon érdemes vizsgálnunk:

* Meredekség (iránymenti derivált)
* arány: Ha picit változtatunk azon, ami szerint deriválunk (itt w), az mekkora változást hoz abban, amit deriválunk (itt J)



Forrás: Wikipédia

$$z_1 = g(W_{10}y_0 + W_{11}y_1 + W_{12}y_2)$$

$$z_1 = g(net_1)$$

A láncszabály segítségével:

$$\frac{\partial J}{\partial W_{1j}} = \frac{\partial J}{\partial z_1} \cdot \frac{\partial z_1}{\partial net_1} \cdot \frac{\partial net_1}{\partial W_{1j}}$$

Általánosan a kimeneti rétegben:

$$\frac{\partial J}{\partial W_{kj}} = \frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial W_{kj}}$$

Egy konkrét súlyra a rejtett rétegben:

$$\frac{\partial J}{\partial w_{10}} = \frac{\partial J}{\partial z_1} \cdot \frac{\partial z_1}{\partial net_1} \cdot \frac{\partial net_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial net_1} \cdot \frac{\partial net_1}{\partial w_{10}}$$

$$\frac{\partial J}{\partial W_{kj}} = \frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial W_{kj}}$$

$$\frac{\partial J}{\partial z_k} = \frac{\partial \frac{1}{2} \sum_{k=1}^{c} (t_k - z_k)^2}{\partial z_k} = -(t_k - z_k)$$

$$\frac{\partial z_k}{\partial net_k} = \frac{\partial g(net_k)}{\partial net_k} = g'(net_k)$$

Az utolsó tényező meghatározásához egy súlyt nézünk.

Lineáris kombináció (skaláris szorzat) parciális deriváltja az a szorzótényező, ami ahhoz a változóhoz tartozik, ami szerint derviálunk.

$$\frac{\partial net_1}{\partial W_{10}} = \frac{\partial (W_{10} y_0 + W_{11} y_1 + W_{12} y_2)}{\partial W_{10}} = y_0 = 1$$

$$\frac{\partial J}{\partial w_{10}} = \frac{\partial J}{\partial z_1} \cdot \frac{\partial z_1}{\partial net_1} \cdot \frac{\partial net_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial net_1} \cdot \frac{\partial net_1}{\partial w_{10}}$$

$$\frac{\partial net_1}{\partial y_1} = \frac{\partial (W_{10} y_0 + W_{11} y_1 + W_{12} y_2)}{\partial y_1} = W_{11}$$

$$\frac{\partial y_1}{\partial net_1} = \frac{\partial g(net_1)}{\partial net_1} = g'(net_1)$$

$$\frac{\partial net_1}{\partial w_{10}} = \frac{\partial (w_{10} x_0 + w_{11} x_1 + w_{12} x_2 + w_{13} x_3)}{\partial w_{10}} = x_0$$

$$\frac{\partial J}{\partial w_{ji}} = \left(- \sum_{k=1}^{c} (t_k - z_k) g'(net_k) W_{kj} \right) \cdot g'(net_j) \cdot x_i$$

Felhasznált anyagok / hivatkozások:

https://www.youtube.com/watch?v=tIeHLnjs5U8

https://www.emathhelp.net/en/calculators
/calculus-3/partial-derivative-calculator/

https://www.youtube.com/watch?v=lWzdntI0WBo