

Deep Learning Szeminárium

2. előadás: neurális hálózatok alapjai

Zombori Zsolt

MTA, Rényi Alfréd Matematikai Kutatóintézet

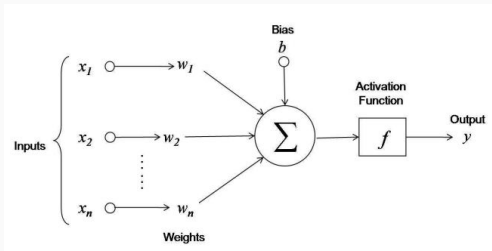
```
https://github.com/elte-ttk-deeplearning/  
elte-ttk-deeplearning.github.io/blob/master/  
homeworks/homework1.txt
```

Amiről ma szó lesz

- Mesterséges neuron, mesterséges neurális háló
- Előreccsatolt/rekurrens hálózat
- Aktivációs függvény
- Veszteségfüggvény (hibafüggvény)
- Neurális hálózatok tanítása
- (Stochastic) Gradient Descent
- Minibatch méret
- Tanulási ráta (learning rate)
- Hibavisszaterjesztés (backpropagation)
- tanító / validációs / teszt adatok
- Kiértékelés: tanulási és általánosítási hiba
- Regularizáció

Mesterséges neuron

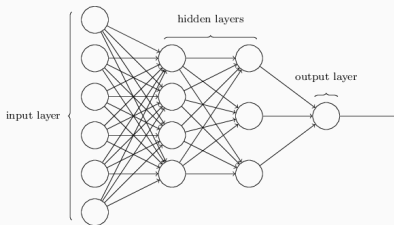
- Az emberi agytól vesz inspirációt, de elrugaszkozik tőle.
- Alap építőeleme a mesterséges neuron
 - Az idegsejt leegyszerűsített modellje
 - Sok bemenet \Rightarrow súlyozott összeg \Rightarrow nemlineáris transzformáció (aktivációs függvény) \Rightarrow kimenet
 - A súlyok (és az eltolás) a neuron paraméterei



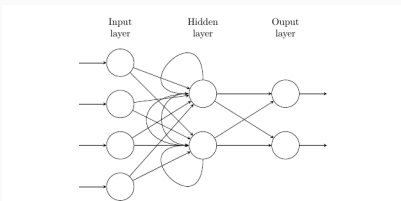
Neurális hálózat

- Sok neuron összekötve
- Rétegekbe rendezzük: bemenet, rejtett rétegek, kimenet
- A rétegek száma a hálózat mélysége.

Előre csatolt hálózat

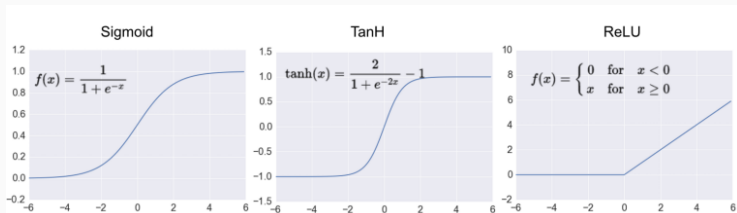


Rekurrens hálózat



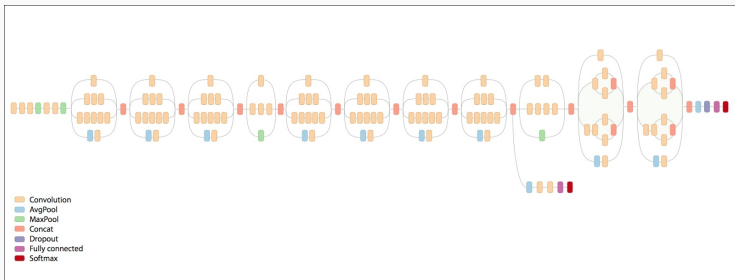
Aktivációs függvény

- Nemlineáris transzformáció
- Enélkül lineáris a leképezés
- Nehezebbé válik az optimalizálás.



Általános modellező eszköz

- A súlyok minden konfigurációja más leképezést eredményez.
- Univerzális függvény approximátor
- Hogyan találjunk meg egy jó beállítást?



Inception v3 (Szegedy et al, 2015), 24 millió paraméter

- Adott egy célfüggvény, ami közvetlenül nem elérhető, de vannak minta adataink (bemenet - kimenet párok).
- Szeretnénk a célfüggvényt közelíteni a súlyok hangolásával.
- Definiálunk egy hiba-függvényt, mely számszerűsíti, hogy mennyire vannak távol az adatok a modelltől.
- A hibafüggvényt minimalizáljuk.

- A hibafüggvény **differenciálható** a súlyok szerint.
- Ki tudjuk számolni a hibafüggvény súlyok szerinti gradiensét.
- A gradiens megmutatja, hogy lokálisan merre kell elmozdulnunk egy picit, hogy a hiba csökkenjen.
- Paraméterek frissítése a negatív gradiens irányában:
$$\underline{\Theta} \leftarrow \underline{\Theta} - \lambda \frac{\partial L}{\partial \underline{\Theta}}$$
- Tanulási ráta (λ): a bátorságunk függvénye, hogy mennyit csavarunk egy lépésben: eleinte nagyot, később kicsit.
- Hosszú, iteratív folyamat.

Stochastic Gradient Descent (SGD)

Egyetlen paraméter frissítés történhet:

- A teljes adathalmaz gradiense alapján
- Egyetlen adatpont gradiense alapján
- Bármilyen a kettő között
 - Sztochasztikus: véletlenszerű csoportok (minibatch-ek)
 - Minibatch méret: elérhető párhuzamosítás felső korlátja
 - Túl kicsi minibatch: lassú, zajos tanulás
 - Túl nagy minibatch: kiátlagolódik, elveszik a tanuló jel

Neurális hálózatok tanítása: mintapélda

$$\underline{z_1} = \mathbf{A}\underline{x} + \underline{b}$$

$$\underline{a_1} = \sigma(\underline{z_1})$$

$$\underline{z_2} = \mathbf{C}\underline{a_1} + \underline{d}$$

$$\underline{a_2} = \sigma(\underline{z_2})$$

$$\hat{y} \equiv \underline{a_2}$$

$$L = \frac{1}{2} \sum (\hat{y} - \underline{y})^2$$

Keressük: $\frac{\partial L}{\partial \mathbf{A}}$, $\frac{\partial L}{\partial \underline{b}}$, $\frac{\partial L}{\partial \mathbf{C}}$, $\frac{\partial L}{\partial \underline{d}}$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

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

$$= \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}}$$

$$= \frac{1}{1 + e^{-x}} \cdot \frac{(1 + e^{-x}) - 1}{1 + e^{-x}}$$

$$= \sigma(x) \cdot (1 - \sigma(x))$$

Neurális hálózatok tanítása: mintapélda

A gradienseket a kimenettől visszafelé számítjuk, a láncszabállyal.

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{C}} &= \frac{\partial L}{\partial \underline{a_2}} \cdot \frac{\partial \underline{a_2}}{\partial \underline{z_2}} \cdot \frac{\partial \underline{z_2}}{\partial \mathbf{C}} \\ &= (\underline{\hat{y}} - \underline{y}) \odot \sigma(\underline{z_2})(\underline{1} - \sigma(\underline{z_2})) \cdot \underline{a_1}^T \\ \frac{\partial L}{\partial d} &= \frac{\partial L}{\partial \underline{a_2}} \cdot \frac{\partial \underline{a_2}}{\partial \underline{z_2}} \cdot \frac{\partial \underline{z_2}}{\partial d} \\ &= (\underline{\hat{y}} - \underline{y}) \odot \sigma(\underline{z_2})(\underline{1} - \sigma(\underline{z_2})) \cdot \underline{1}\end{aligned}$$

Visszaterjesztett hiba:

$$\underline{\delta_2} \equiv \frac{\partial L}{\partial \underline{z_2}} = (\underline{\hat{y}} - \underline{y}) \odot \sigma(\underline{z_2})(\underline{1} - \sigma(\underline{z_2}))$$

Neurális hálózatok tanítása: mintapélda

Kiszámoljuk a rejtett réteghez tartozó gradienseket.

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{A}} &= \frac{\partial L}{\partial \underline{z_2}} \cdot \frac{\partial \underline{z_2}}{\partial \underline{a_1}} \cdot \frac{\partial \underline{a_1}}{\partial \underline{z_1}} \cdot \frac{\partial \underline{z_1}}{\partial \mathbf{A}} \\ &= \mathbf{C}^T \cdot \underline{\delta_2} \odot \sigma(\underline{z_1})(1 - \sigma(\underline{z_1})) \cdot \underline{x}^T \\ \frac{\partial L}{\partial \underline{b}} &= \frac{\partial L}{\partial \underline{z_2}} \cdot \frac{\partial \underline{z_2}}{\partial \underline{a_1}} \cdot \frac{\partial \underline{a_1}}{\partial \underline{z_1}} \cdot \frac{\partial \underline{z_1}}{\partial \underline{b}} \\ &= \mathbf{C}^T \cdot \underline{\delta_2} \odot \sigma(\underline{z_1})(1 - \sigma(\underline{z_1})) \cdot \underline{1}\end{aligned}$$

Visszaterjesztett hiba:

$$\underline{\delta_1} \equiv \frac{\partial L}{\partial \underline{z_1}} = \mathbf{C}^T \cdot \underline{\delta_2} \odot \sigma(\underline{z_1})(1 - \sigma(\underline{z_1}))$$

Hibavisszaterjesztéses algoritmus (Backpropagation)

1. Előreterjesztés
 2. Visszaterjesztés
 - 2.1 A kimenettől visszafelé haladva kiszámoljuk a $\underline{\delta_i}$ visszaterjesztett hibatagokat.
 - 2.2 $\underline{\delta_i}$ alapján kiszámoljuk az i -dik réteg paraméterei szerinti gradienst
- Mindkét fázis lineáris időben megvalósítható.
 - Mára mindezt ajándékba kapjuk a modern tenzor könyvtáráktól.

A tanulás folyamata

Rengeteg optimalizációs módszer a tanulás irányítására

- Tanulási ráta
- Paraméterenként külön-külön tanulási ráta.
- Momentum módszer: megjegyezzük, hogy a múltban merre mozdultak a paraméterek.
- Múltbeli gradiensek második momentumát is figyelembe vehetjük.

Manapság a leggyakrabban használt módszerek:

- Nesterov momentum
- Adaptive Gradient Algorithm (AdaGrad)
- Root Mean Square Propagation (RMSProp)
- Adam

- Tanító adatok: a hálózat paramétereinek beállítása.
- Validációs adatok: a hálózat és a tanulási folyamat hiperparamétereinek beállítása.
- Teszt adatok: végső kiértékelés.

1. Tanulási hiba: mennyire illeszkedik a modell a tanító pontokra?
 - Egy tipikus neurális hálózat erősen túlparametrizált és nagyon sok féle módon tud jól illeszkedni a tanító pontokra.
2. Általánosítási hiba: mennyire jól általánosodik a modell tanítás során nem látott adatokra?
 - Regularizáció

- Aktívan kutatott terület.
 - Minden olyan módszer, mely megakadályozza a túltanulást.
1. Architektúrális változtatások (Dropout ...)
 2. A tanulás folyamatának módosítása (Korai Leállás ...)
 3. Adatok módosítása (Augmentáció ...)
 4. Új tag hozzávétele a hibafüggvényhez (Weight Decay ...)

- Elosztott számítás: neuronok összekötött hálózata
- Függvénykompozíció: Egyszerű $R^n \rightarrow R^m$ függvények kompozíciója
- Differenciálható számítás: a függvényünk majnem mindenhol differenciálható, így optimalizálható