

Mesterséges Intelligencia

Modern mesterséges neurális
hálózatok alapjai

FONTOS

- Az alábbi anyag munkavázlat, hibákat tartalmazhat. Amennyiben hibát találnak, kérem, a portálon keresztül üzenetben jelezzék, hogy melyik heti előadás, vagy jegyzet melyik részében, milyen hibát véltek felfedezni!
- Az anyagok kizárolag a Széchenyi István Egyetem 2021-2022 tavaszi félévében Mesterséges Intelligencia kurzust felvett hallgatói számára készültek, kizárolag az adott félév kurzusaihoz használható fel!
- Az alábbi hivatkozásokon megnyitott minden fájl automatikusan begyűjti a hallgató különböző egyedi azonosítóit, mely alapján beazonosítható lehet. Ennek megfelelően a hivatkozásokat ne osszák meg egymással (különösen a kurzust nem hallgatókkal), mert abból az egyedi azonosítók visszakereshetők és a személyazonosság meghatározható!
- Az alábbi anyagra vonatkozóan minden jog fenntartva!
- Az anyagok bármely részének vagy egészének nyomtatása, másolása, megosztása, sokszorosítása, terjesztése, értékesítése módosítással vagy módosítás nélkül egyaránt szigorúan tilos!

A lecke főbb téma körei

- Modern alkalmazásokban fellelhető neuron modellek főbb típusai
- Modern alkalmazásokban fellelhető topológiák
 - Konvolúciós neurális hálózatok
 - Hopfield hálók
 - Autoenkóderek
 - Generatív versengő hálók
- Neurális hálók tanítása

Modern mesterséges neurális hálózatok alapjai

modern neuron modellek

Modern neuron modellek

- A cím kissé megtévesztő
- napjainkban használt technikák alapjai akár évtizedekkel korábbra vezethető vissza
- 2010-es évek elején megugrott a mesterséges neurális hálózatok bizonyos változatainak és a hozzájuk kapcsolódó módszerek alkalmazása
- Ez részben köszönhető
 - a finomított modelleknek,
 - megfizethető olyan számítási kapacitással rendelkező hardver, amely képes már nagyobb modelleket tanítani
- A mély neurális hálózatok (deep neural network) és azok tanítása a mélytanulási (deep learning) megoldásokkal napjaink legnépszerűbb mesterséges intelligencia eszközének számítanak
 - 2018-tól azonban egy bizonyos fokú visszaesés figyelhető meg a kutatási eredményekben

Modern neuron modellek

- A korábbi leckében már említett mesterséges neuronok változatosak
 - vannak lineáris és nemlineáris egységek
 - A lineáris egységek bonyolultabb rendszerek esetén nem rendelkeznek kellő kifejező erővel
 - A modern megoldások esetén jellemzőbben a nemlineáris aktivációs függvényekkel rendelkező neuronokat (is) használnak

Modern mesterséges neurális hálózatok alapjai

ReLU

ReLU

- Rectified Linear Unit
- leggyakrabban használt
- negatív értékeknél 0 értéket ad vissza, míg pozitív értékre lineáris
- egyszerű számításokat végezni vele
- gyorsabb tanulást tesz lehetővé, mint például a szigmoid aktivációs függvénnyel rendelkező neuronok

ReLU

ReLU vs. Sigmoid aktivációs függvény

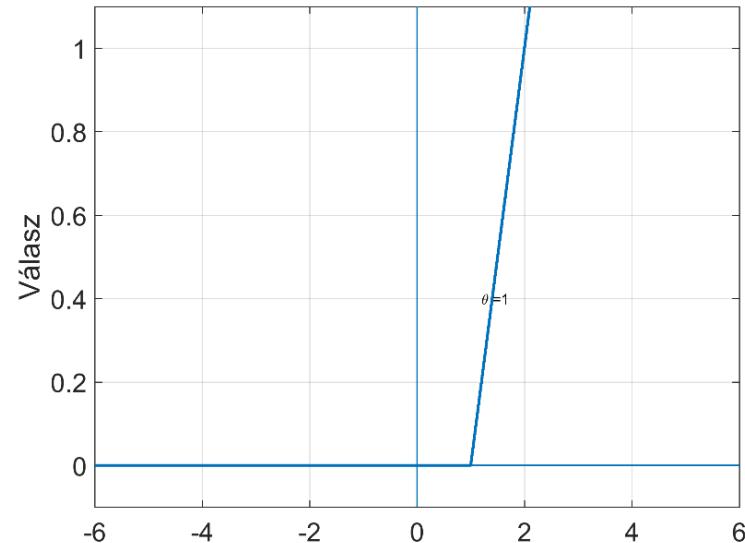
A szigmoid és tangens hiperbolikus aktivációs függvények alkalmazása a kevés réteget tartalmazó neuronhálókra szűkül. Az említett függvények gradiense eltűnik a mély neuronhálók esetében, megakadályozva a neuronháló tanulását. → Azaz hiába adunk újabb rétegeket a hálóhoz, az eredmények nem javulnak (sőt esetleg romlanak is)

A ReLU aktivációs függvény deriváltja mindenhol egyenletes és nincs szaturálva a kimenet. Jó megoldást biztosít a mély neuronhálók esetében.

ReLU: mély neurális hálók aktivációs függvényeként

ReLU

- Komoly hátrányossága, hogy a negatív értékek esetén visszaadott 0 érték csökkenti a rendszer tanulási képességét
- Előfordul, hogy a 0 értékek miatt úgy módosulnak a hálózat súlyai, hogy a neuron többé nem aktiválódik, vagyis meghal (die)
- $f_{ReLU}(x)=\max(0,x)$



Modern mesterséges neurális hálózatok alapjai

Módosított ReLU koncepciók

Módosított ReLU koncepciók

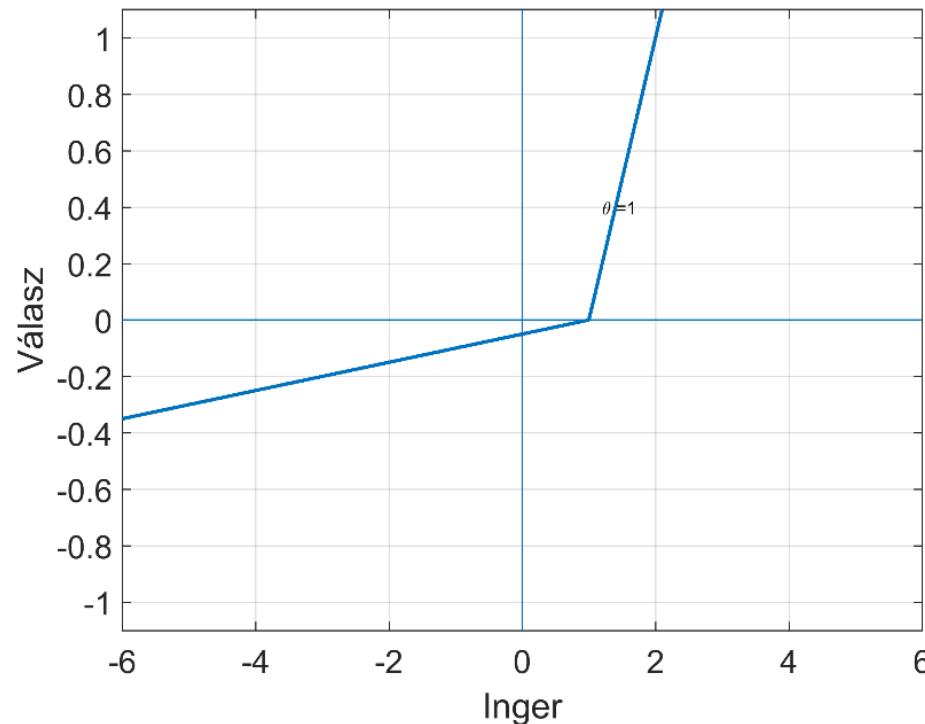
- A ReLU korlátait orvosolandó alkott meg őket
- A nemnegatív értékek esetén 0-tól eltérő válasz adás volt a cél

Leaky ReLU

- Leaky ReLU
 - a negatív értékekre vonatkozó módosítás
 - *a bemeneti érték 0,01 szeresét adja vissza*
- $f_{ReLU}(x) = \max(x; 0,01x)$

Parametrized ReLU koncepciók

- A Leaky ReLU általánosítása
 - A 0,01 szorzó helyett tetszőleges α paraméter érték
- $f_{PReLU}(x) = \max(x, \alpha x)$



Randomized ReLU

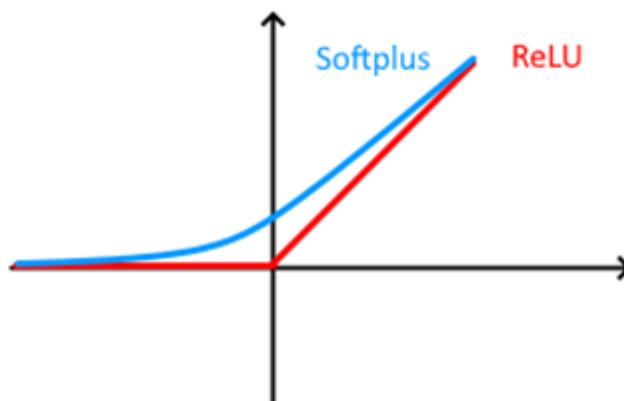
- Amikor a Leaky ReLU által alkalmazott a paraméter véletlenszerűen veszi fel az értékét

Modern mesterséges neurális hálózatok alapjai

Softplus

Softplus

- gyakran alkalmazott ReLU helyettesítő megoldás
- jellegükben hasonló értéket adnak vissza, mint a ReLU-k
- nem szakaszos



Modern mesterséges neurális hálózatok alapjai

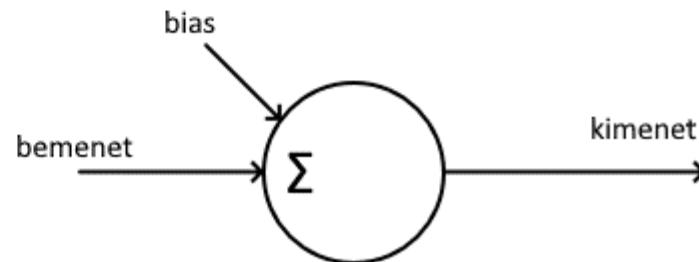
Előrecsatolt neuron struktúrák

Neuron struktúrák

- neuron modelleket nem csak az aktivációs függvény alapján lehet megkülönböztetni,
- a neurális betöltött funkciója, illetve bekötésének módja is fontos

Előrecsatolt neuron

- A legegyszerűbb a sigmoid függvényt alkalmazó általánosított TLU, vagyis egy nem lineáris perceptron
- a neuronhoz kötött súlyozott bemeneteken túl egy bias értéket is szoktak használni
- Egyszerűen előrecsatolt (Feed Forward, FF) neuronnak is hívják



Konvolúciós neuron

- Hasonlítanak az egyszerű előrecsatolt neuronokhoz
- jellemzően egy adott neuroncsoporthoz van csak csatlakoztatva
- kiválóan alkalmas térbeli/helyzeti információk közvetítésére
 - különösen alkalmas a képi és hanginformációk feldolgozására
- Lényegében az adatot blokkokra bontja
 - egy-egy blokkot akár több különálló konvolúciós neuron csoport is felhasználhat, eltérő paraméterezés mellett lényegében eltérő lokális jellemzőket kinyerve.
- A dekonvolúciós neuronok lényegében a konvolúciós neuronok ellentettjei

Konvolúciós neuron

- A konvolúciós neuronokat alkalmazó neurális hálókban gyakran alkalmaznak még további neuronokat
 - A klasszikus értelemben nem neuronok
 - tág értelemben, mint a hálózatba kötött egyszerű feldolgozó egységek már igen
- összevonó (pooling) neuronokat
 - a reprezentáció méretének csökkentése (downsampling)
 - képfeldolgozásnál csak kiválasztott pixelekhez tartozó jelek továbbítását jelenti
- interpolációs (interpolating) neuronokat
 - a pooling neuronok ellentetje
 - kitöltenek a hiányzó, köztes értékeket..

Valószínűségi neuronok

- középérték (mean)
- szórás (deviation)
- gyakran együttesen valószínűségi (probabilistic) neuronként jelennek meg
- Céljuk jellemzően valószínűségi eloszlás információk tárolása a rendszerben
- nem rendelkeznek bias értékkel

Modern mesterséges neurális hálózatok alapjai

Visszacsatolt neuron struktúrák

Visszacsatolt neuron struktúrák

Vannak azonban feladatok, ahol a minták egymásutánisága fontos plusz információt hordoz

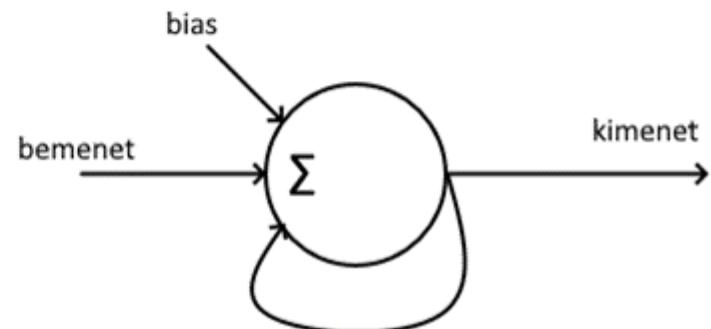
- Tipikusan időbeli sorozatok modellezésénél fordul elő
- Pl.: beszédfelismerés, nyelvi feldolgozás, kézírás-felismerés, videók elemzése, árfolyambecslés

Kérdés, hogy hogyan tudjuk úgy módosítani a hálót, hogy a szomszédokat is figyelembe vegye

- Előrecsatolt háló több szomszédos inputon: Time-delay neural network
- Visszacsatolt hálózat: recurrent neural network
- Visszacsatolt háló hosszú távú memóriával: Long short-term memory network

Rekurrens neuronok

- jellemző a ReLU aktivációs függvény
- a bias érték használata
- a hálózat egymást követő számítási ciklusai során a korábbi eredményt is felhasználja
- lényegében egy egyszerűbb (rövidtávú) memória



Fejlett rekurrens neuronok

- hosszú rövidtávú memória (Long Short Term Memory, LSTM)
- kapuzott visszacsatolt egység (Gated Recurrent Unit, GRU)
- Az egyszerű rekurrens hálók korlátozott memóriaképességeinek leküzdésére hozták létre
- összetett architektúrával rendelkező megoldások
- a tárolt információ fejlettebb kezelésére külön bemeneti csatornákkal rendelkeznek.
- GRU az LSTM egy egyszerűsítésének tekintethető
 - Így gyorsabb

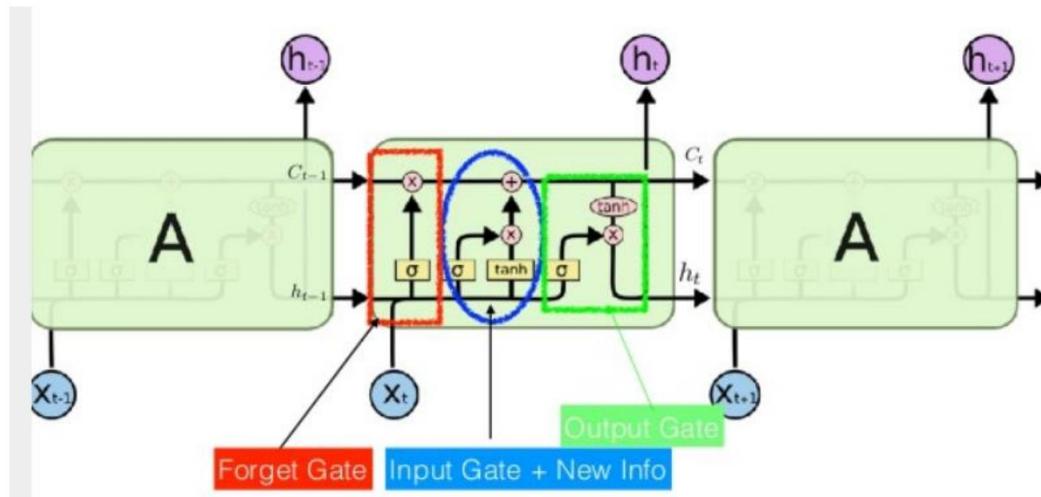
Fejlett rekurrens neuronok

LSTM

Egy külön belső állapotot hozunk létre, amely memóriaként fog működni

Az információ több párhuzamos útvonalon fog áramlani:

- **Forget gate:** mely komponenseket kell elfelejteni a C memóriából
- **Input gate:** mely input komponenseket kell eltárolni C-be
- **Output gate:** hogyan álljon össze a kimenet
- A cell state („memória”) frissítése a forget gate és az input gate segítségével



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \text{ forget gate}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \text{ input gate}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \text{ output gate}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \text{ New cell memory}$$

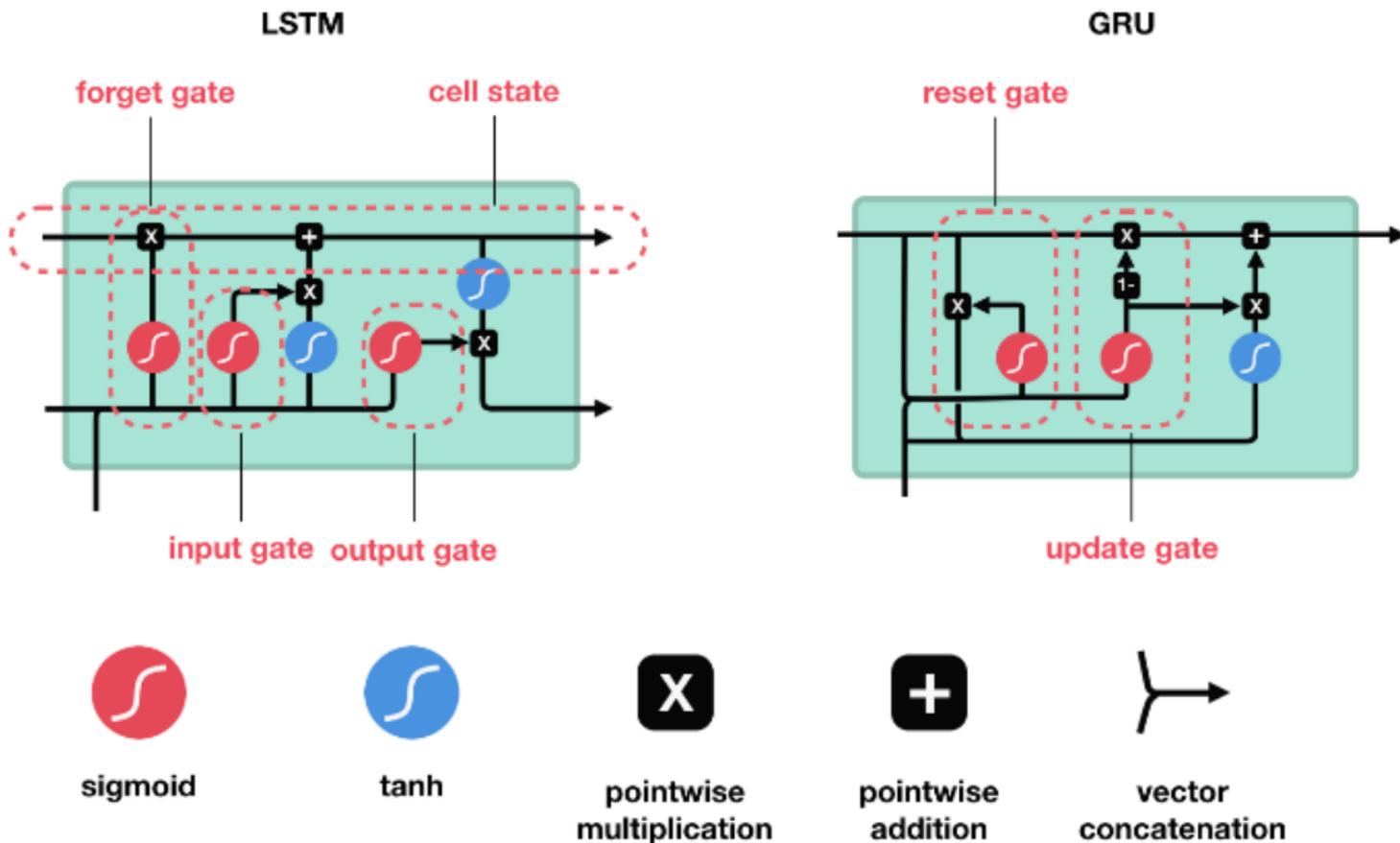
$$h_t = o_t \circ \sigma_h(c_t) \text{ New Hidden}$$

Fejlett rekurrens neuronok

LSTM vs. GRU

A GRU kevesebb kaput használ, de a tapasztalatok szerint könnyebben, gyorsabban tanítható, és hasonló pontosságra képes, mint az LSTM

A kevesebb kapu miatt egyetlen GRU egység kevesebb egyenlettel írható le, mint az LSTM



Modern mesterséges neurális hálózatok alapjai

Architektúrák

mély előrecsatolt

- Deep Feed Forward, DFF architektúra
- lényegében egyszerű, általánosított többrétegű perceptron háló
- jellemzően ReLU, vagy nemlineáris aktivációs függvényeket alkalmaznak
- A „mély” jelzőt annak köszönheti, hogy legalább két rejtett réteg található benne
- Már régebben is számos rendszer rendelkezett ilyen, sőt még bonyolultabb struktúrákkal, de akkoriban nem tüntették ki ezeket külön a mély megnevezéssel
 - A jelentősebb eltérés a jól strukturált felépítés
- jellemző a teljes összecsatolás a rétegek között
- számos változata fellelhető a gyakorlatban

Modern mesterséges neurális hálózatok alapjai

Architektúrák

konvolúciós neurális hálózatok

- Convolutional Neural Networks, ConvNet
- egy mély előrecsatolt hálózat
 - bemeneti rétege és az első rejtett rétege közé konvolúciós és pooling neuronokat tartalmazó rétegeket helyeznek el
- célja a bemeneti adat különböző részeire vonatkozó jellemzők (features) kinyerése és az információ szűrése
- A leggyakrabban a gépi látás területén alkalmazzák
 - de képes akár audió feldolgozásra is
- A konkrét megvalósítások az architektúra részleteiben, számítási igényben és pontosságban eltérnek, azonban az alap koncepció egységes

Konvolúciós neurális hálózatok

Konvolúciós neurális hálózatok első rétege mindig egy **Konvolúciós réteg**

Első Konvolúciós réteg feladata: alacsony szintű tulajdonságok detektálása például élek görbék stb.

További konvolúciós rétegek feladata: magasabb szintű jellemzők detektálása magasabb rétegek egyre komplexebb, egyre absztraktabb fogalmakat tanulnak meg (például szem, orr, arc stb.)

A konvolúciós réteg neuronjai átalakítják a bemenetet annak érdekében, hogy kiemeljék a fontosabb jellemzőket azon. Ezt az átalakítást egy kernel segítségével végzik. A kernel konvolúciós mátrix néven is ismert.

A kernel egy súlyokat tartalmazó, kis méretű matrix. Három dimenzió esetében a magasságát és szélességét tekintve kisebb, mint a bemeneti mátrix, melyen a konvolúciót végezzük, viszont mélységükben megegyeznek.

A konvolúciós művelet a kernelben található súlyértékek felhasználásával alakítja át a bemenetet, a kernelt a bemeneti képen függőleges és vízszintes irányokban végigcsúsztatva. Így kerül kiszámításra a konvolúciós réteg egy neuronja által előállított aktivációs térkép (feature map).

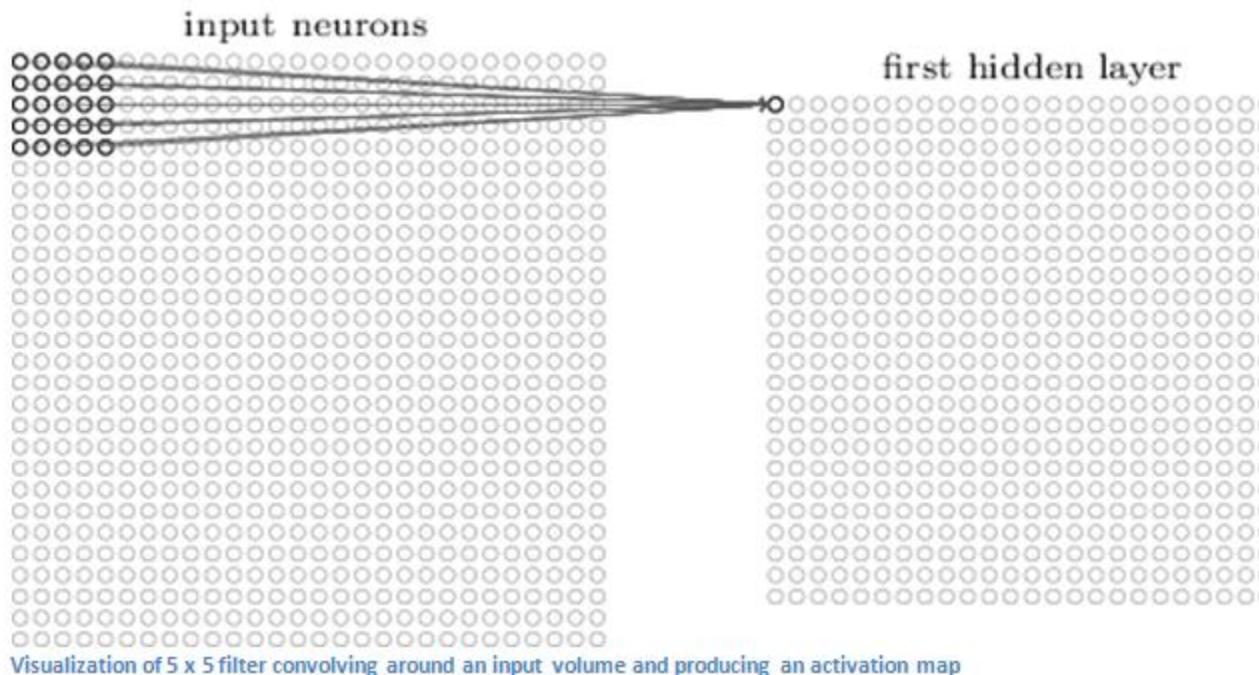
Konvolúciós neurális hálózatok

Konvolúciós réteg

Példa: bemeneti kép 32x32 pixel

Kernel 5x5x3

A kimenet: aktivációs térkép ekkor 28x28-as méretű



Konvolúciós neurális hálózatok

Összevonó (polling) réteg

Jellemzően a Konvolúciós rétegek után alkalmazzuk őket a folyamatos dimenziócsökkentésre, ezáltal a számítások redukálására szolgál

egy **mintavétel** a a Konvolúciós réteg eredményén → a Konvolúciós réteg mögött szoktuk alkalmazni

Vegyél egy xy méretű ablakot, lépkedj végig az aktivációs térképen ugyanúgy, mint a Konvolúciós rétege esetén, és alkalmazd a $p()$ mintavételi függvényt az ablakban lévő adatokon **átfedés nélkül**.

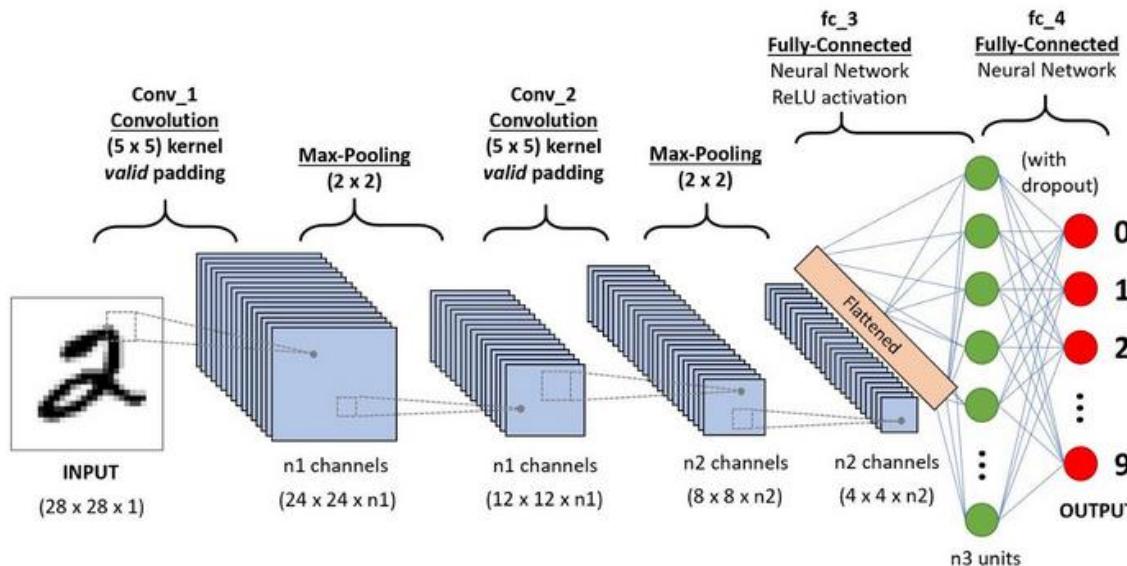
Általában átlagot, maximumot vagy minimumot számolunk

Konvolúciós neurális hálózatok

Teljesen összekapcsolt (Fully connected) réteg

Hálózat végén szerepel

A teljesen összekapcsolt réteg minden neuronja minden előző réteg neuronjával összeköttetésben áll, ezáltal a felsőbb rétegek aktivációs térképeinek eredményét levetítve az ezt követő, klasszifikációs feladatot megoldó rétegekre



Modern mesterséges neurális hálózatok alapjai

Architektúrák

Hopfield hálózatok

- visszacsatolt neurális hálók
- minden egyes neuron kimenete összeköttetésben áll az összes többi neuron bemenetével
 - minden neuron egyben bemeneti és kimeneti is
- Kiválóan alkalmas az önszervező tanulásra
 - a rendszert mintákkal kell ellátni, mely értékeinek megfelelően módosul a hálózat, majd konvergál az ideális értékhez.
- Jellemzően egyetlen rétegű hálózatként, vagy gyűrűként szokták ábrázolni.

Hopfield hálózat

Az alapmodell esetében a Hopfield hálózat neuronai kétértékűek.

A hálózat bemenetének a neuronok kezdeti állapotai tekinthetők, míg a kimenetet ugyanezen neuronok állapotai jelentik a működés során.

A hálózat tanítása a megjegyezendő minták tárolása (a súlyok megfelelő beállításával történik)

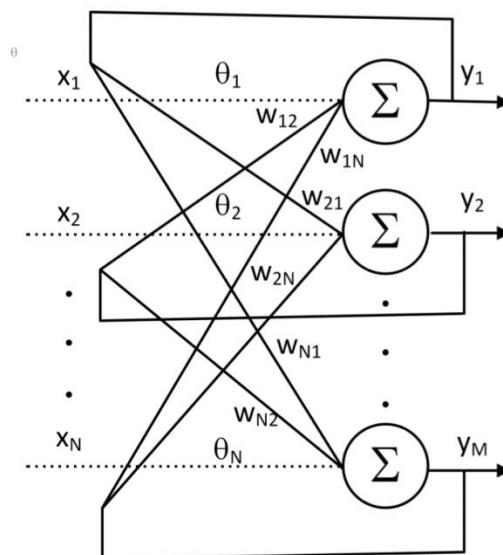
$$W_{ii}=0$$

$$W_{ji}=W_{ji}$$

$$x_i := \operatorname{sgn} \left(\sum_j w_{ij} x_j - \theta_i \right)$$

ahol

$$\operatorname{sgn}(x) = \begin{cases} +1 & \text{ha } x \geq 0 \\ -1 & \text{ha } x < 0 \end{cases}$$



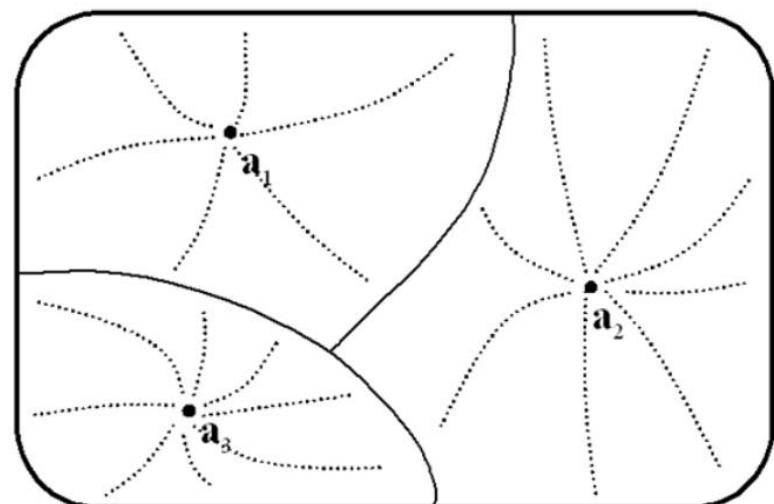
Hopfield hálózatok

A megfelelően beállított súlyokkal rendelkező hálózattól azt várjuk, hogy az adott kezdeti konfigurációból kiindulva az ehhez legközelebbi vonzási pontba stabilizálódik. → energiaminimumok
A hálózat működtetésének alapvetően kétféle módja van:

- **Szinkron:**
az összes neuront minden időlépésnél egyszerre módosítjuk
a rendszer bármely állapotváltozás során a konfigurációs tér bármely pontjába kerülhet
- **Aszinkron:**
egy pillanatban csak egy egység válthat értéket
a hálózat szomszédos konfigurációkon keresztül jut el a vonzási pontba

Alkalmazás:

bináris minták tárolására
hibás, zajos, hiányos minták helyreállítására,
felismerésére



Modern mesterséges neurális hálózatok alapjai

Architektúrák

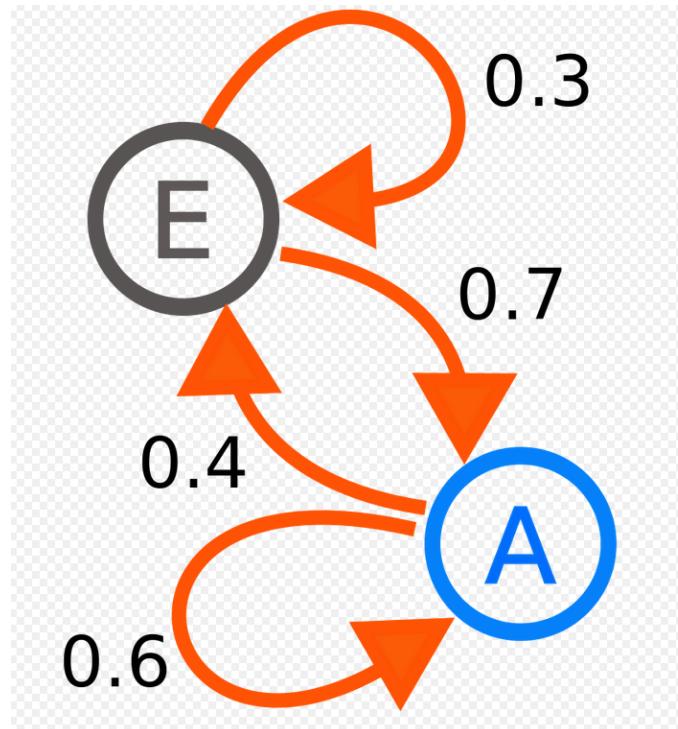
Markov láncok

- A Hopfield hálózatok egy speciális változatának tekinthetőek
- a klasszikus értelemben nem neurális háló
- Markov láncok (Markov Chain), ahol a (jellemzően) teljes összeköttetésben álló neuronok probabilisztikusak
- azt modellezi, hogy egy adott állapotból (neuronból) mekkora a valószínűsége egy másik állapotba való váltás

Markov láncok

Markov-tulajdonság: *adott jelenbeli állapot mellett, a rendszer jövőbeni állapota nem függ a múltbeliekktől*

$$\Pr(X_{n+1} = x | X_n = x_n, \dots, X_1 = x_1) = \Pr(X_{n+1} = x | X_n = x_n)$$



Modern mesterséges neurális hálózatok alapjai

Architektúrák

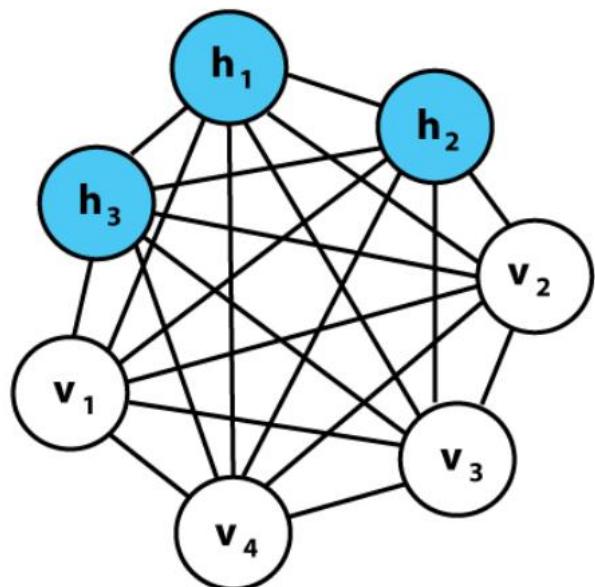
Boltzmann gépek

- Boltzmann Machine, BM
- szintén a Hopfield hálózatok egy speciális változata
- jelentős különbség, hogy egyes neuronok dedikáltan bemeneti neuronok, míg a többi rejtett
- egy teljes hálózati frissítést („lefutást”) követően a bemeneti neuronok kimeneti neuronná válnak
- A neuronok jellemzően bináris jellegűek
- gyakrabban használt változata a korlátos Boltzmann gép
 - annyiban különbözik az eredeti koncepciótól, hogy a (ki/)bemeneti neuronok és a rejtett neuronok elkülönülő csoportot (réteget) alkotnak
 - a rétegeken belül nincs kapcsolat a neuronok között
 - Ennek köszönhetően az előrecsatolt hálózatokhoz hasonló tanulás is megvalósítható az ilyen rendszerek esetében

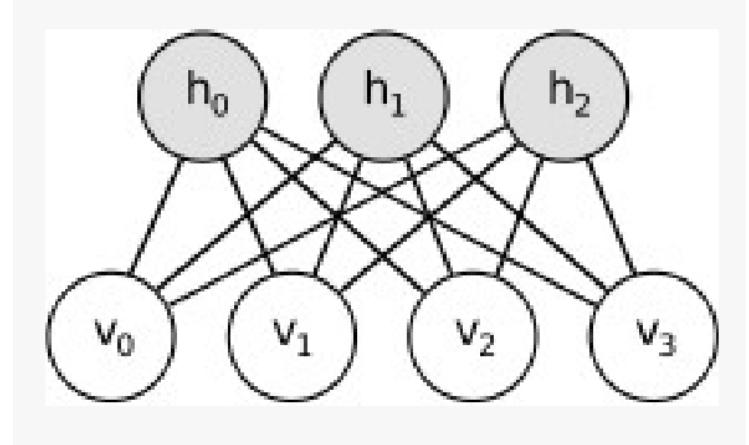
Boltzmann gépek

Két réteg:

- Látható réteg: ez lesz egyszerre az input és az output réteg bináris kimeneti értékekkel ad
- Rejtett réteg



Boltzmann gép



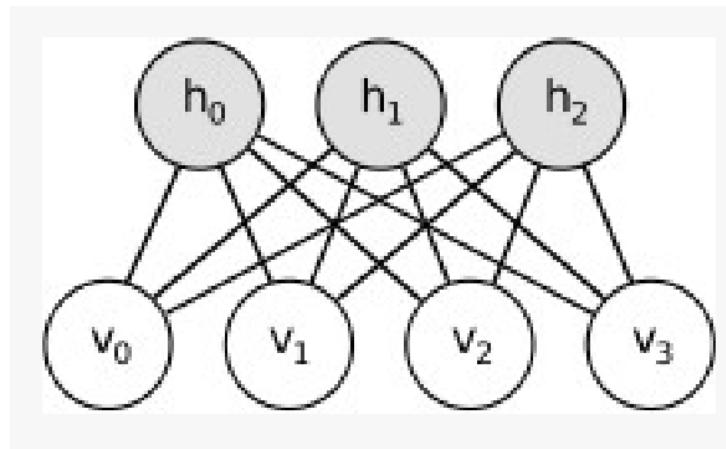
Korlátos Boltzmann gép

Korlátos Boltzmann gépek

Energia függvény:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

Korlátos Boltzmann gép tanítása: paraméterek (w súlyok, b , a eltolás értékek) meghatározása, amelyek minimalizálják az energiavártékot az adott inputra



Korlátos Boltzmann gépek

Alkalmazások:

- Dimenziócsökkentés
- Osztályozás
- Regresszió
- Kollaboratív szűrés

Modern mesterséges neurális hálózatok alapjai

Architektúrák

autóenkóderek

- Autoencoder, AE
- egy speciális előrecsatolt háló
- célja az információ kódolása (tömörítése) asszociáció révén
- variációs autóenkóder (Variational Autoencoder, VAE)
 - az autóenkóderek egy speciális változata
 - más megközelítést alkalmaz
 - a bemenetek közelítő valószínűségi eloszlásain alapul, valószínűségi rejtett neuronokkal
 - elméleti háttere a Bayes-i matematikára és Boltzmann gépekre is visszavezethető

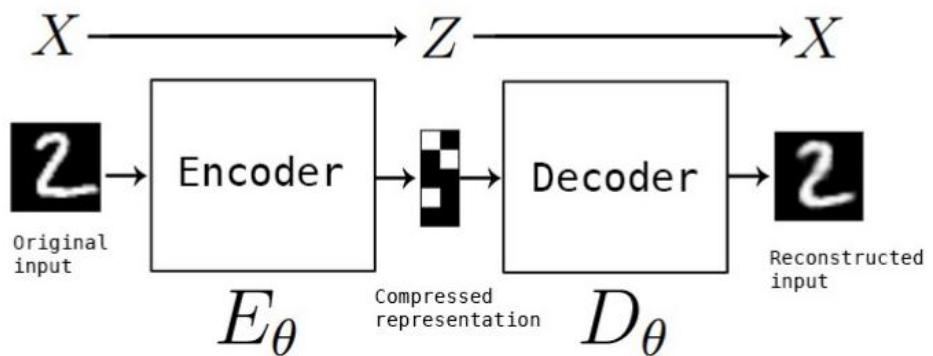
Autóenkóderek

Az autóenkóderek olyan neurális hálók, amelyeknek az első fele egy enkóder részből, a második fele pedig egy dekóder részből áll. A hálózat input tere és output tere azonosak.

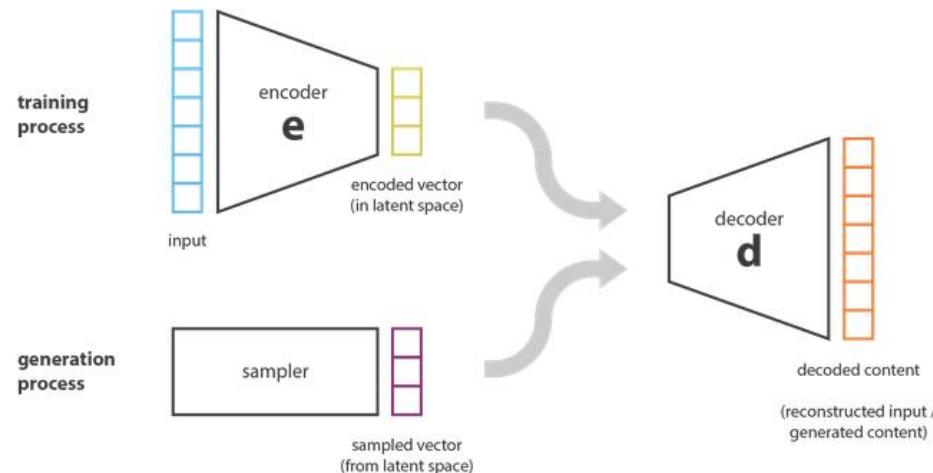
Vagyis egy autoenkóder függvényként úgy írható le, mint egy $f_{\theta} : X \rightarrow X$ függvény, valamint $E_{\theta} : X \rightarrow Z$ az enkóder függvénye, $D_{\theta} : Z \rightarrow X$ a dekóder rész függvénye, és $f_{\theta} = E_{\theta} D_{\theta}$. (θ a kalibrálandó változók, tehát a súlyok és bias-ok)

Általában $X = R^n$ és $Z = R^k$, ahol $k << n$. Továbbá a tanítási veszteségfüggvényünk, melyet a tanítás során minimalizálni kívánunk, hogy az egyes $i \in I$ inputokra „hasonlítsan” a háló $f_{\theta}(i)$ outputja $\rightarrow \sum_{i \in I} |i - f_{\theta}(i)|$ kifejezést kívánjuk minimalizálni

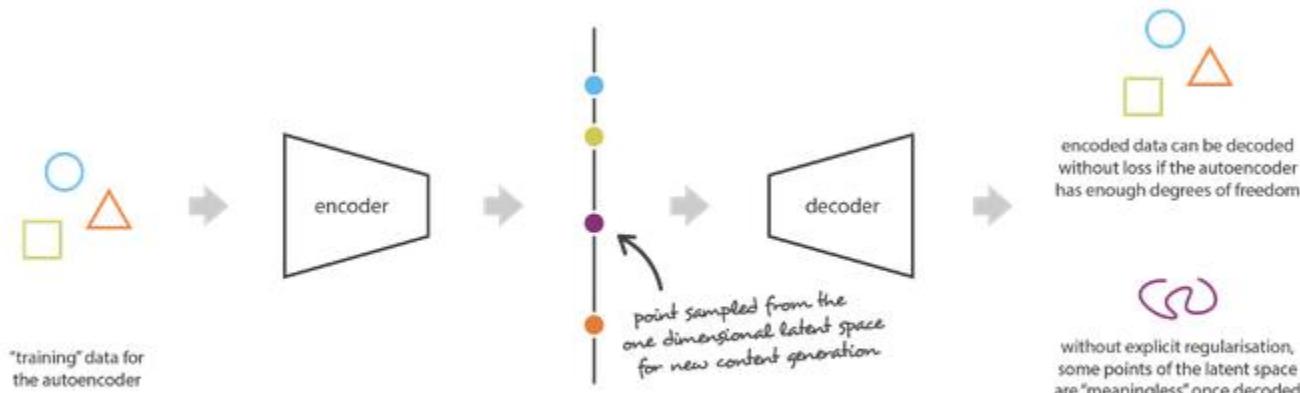
- $I \subset X$ inputok Z kis dimenziós térben való **veszteséges tömörítésére**. A tanítással pedig szeretnénk minél jobb, azaz kisebb átlagos veszteségű, tömörítést előállítani.
- zajcsökkentés
- Adatvizualizálás, dimenziócsökkentés



Adatgenerálás

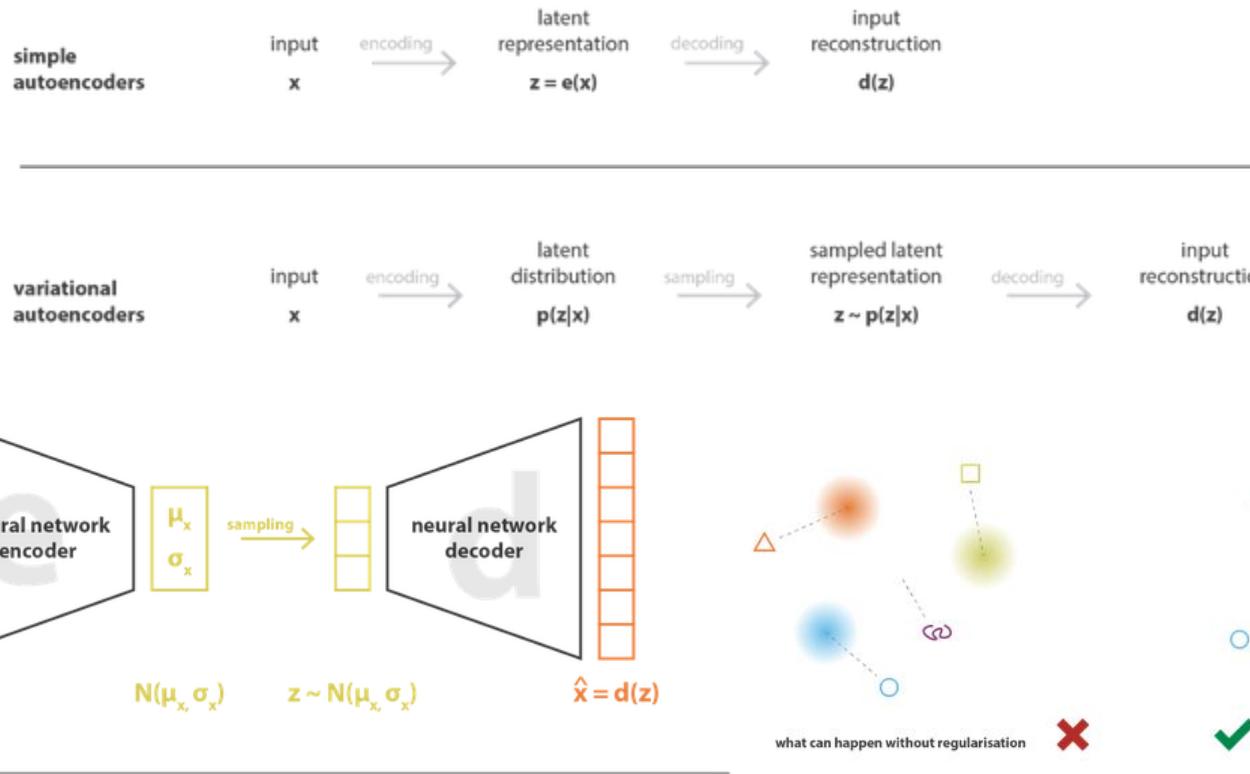


egy adathalmaz segítségével annak elemeihez hasonló objektumok létrehozása
A látens térnek regulárisnak kell lennie



Irregular latent space prevent us from using autoencoder for new content generation.

Variációs autoenkóderek



$$\text{loss} = \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

Költségfüggvény két részből áll:

- Bemenet és kimenet közötti négyzetes hiba
- A látens tér regularitását biztosítja: A kódoló kimenetéből kapott normális eloszlás minél közelebb legyen a standard normális eloszláshoz

Variációs enkóderek

Arcgenerálás



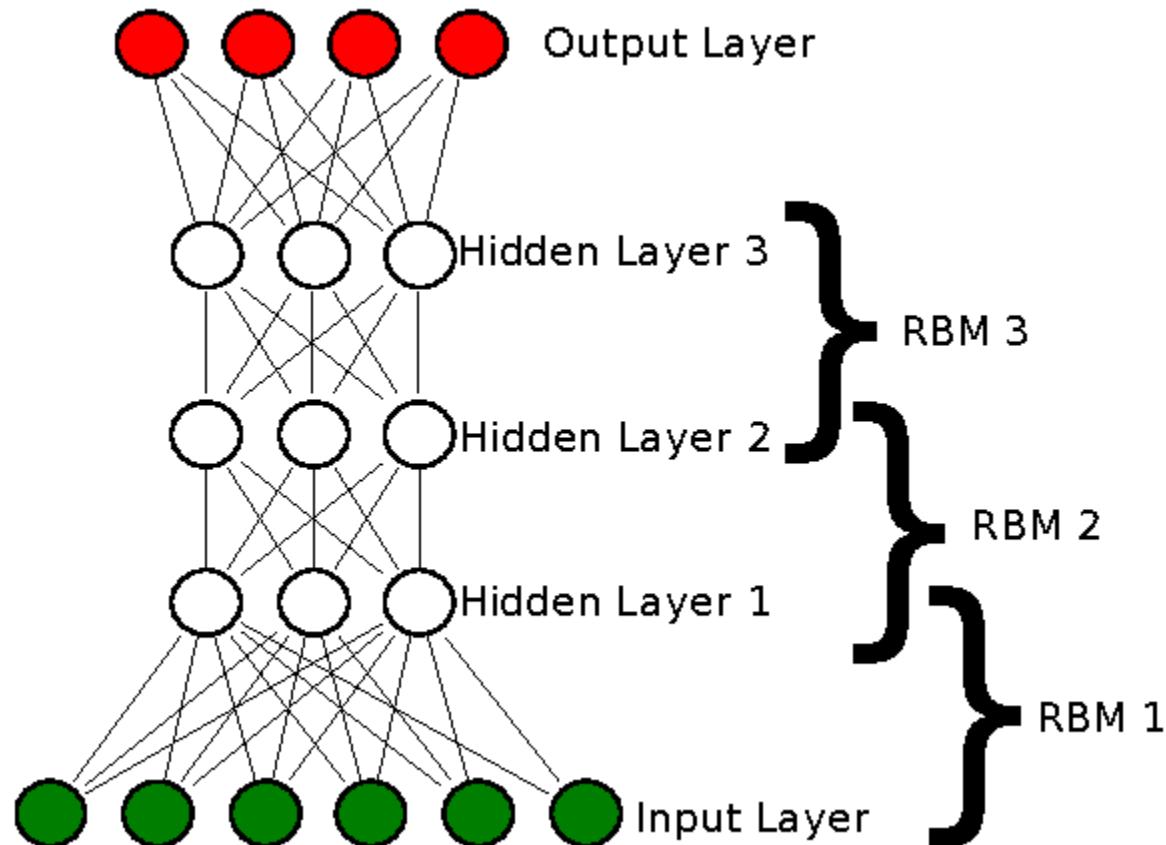
Modern mesterséges neurális hálózatok alapjai

Architektúrák

mély bizonyosságháló

- Deep Belief Network, DBN
- A variációs autoenkóderekhez és a korlátos Boltzmann gépekhez is kapcsolható architektúra
- az egyik legelterjedtebb nem konvolúción alapuló mély architektúra volt
- Generatív modellnek tekinthető
 - egyaránt alkalmas osztályozási feladatokra és nemfelügyelt tanulással új adat generálására is
- Az egyes rétegek hatékonyan taníthatóak külön-külön is
 - mohó (greedy) megoldás segítségével
 - tudatosan rétegenkénti lokális optimumok révén próbál elérni egy kvázioptimális modellt

Mély bizonyosságháló



Modern mesterséges neurális hálózatok alapjai

Architektúrák

generatív versengő hálók

- Generative Adversarial Network, GAN
- hibrid megoldás
- valójában két hálózatból állnak
 - Egy generátor
 - mintákat kell előállítania
 - egy diszkriminátor
 - Generált és valós mintákat kell megkülönböztetnie

Cél: eloszlás megtanulása, és abból példányok generálása

- Generátor és diszkriminátor egy zéró összegű játék két szereplője: hamisító és rendőr
GAN hálózatok összehasonlítása nehéz, mert mindenkor csak a két háló egymáshoz viszonyított teljesítményét látjuk

generatív versengő hálók

- Az ilyen hálózatok kezelése, tanítása komoly kihívást jelenthet
 - a részhálók tanítását és működésük összehangolását egyaránt meg kell valósítani
 - a lokális optimumokban való ragadás nélkül.
- Kiválóan alkalmasak élethű képek generálására
- számos GAN változat elterjedt
- leggyakrabban a különböző képek és videók generálásánál használják

Generatív versengő hálók

GAN által „festett” festmény: 432 ezer dollárért kelt el
15 ezer 14-20. századi képen tanították



Modern mesterséges neurális hálózatok alapjai

Neurális hálók tanítása

Neurális hálók tanítása

- gépi tanulás a mesterséges intelligencia egyik ága
 - nem része a mesterséges neurális hálózatok területének, de elengedhetetlen segítője
- architektúrák ideális paraméter értékeinek felügyelt és nemfelügyelt tanulással való meghatározása a cél
- Mély architektúrák - mély tanulás területe
- számos formában előfordulhat
 - a konkrét megoldások jellemzően alkalmazási területhez köthetők.

Neurális hálók tanítása

- A tanítási módszertől függetlenül különösen fontos a megfelelő mintakészlet kialakítása
- A legtöbb mélytanuló megoldás nagymennyiségű adatot igényel
- Gyakran a neurális hálózatokon alapuló modellek fejlesztési idejének nagyobb részét teszi ki a minták előkészítése, mint a hálózat felépítése
 - Egy egyszerű objektumdetektáló gépi látás megoldás esetén is több ezer példaképre és a kapcsolódó (elvárt kimeneti) adatra szükség van (például a képen berajzolt jelölőnégyzetek és címkék formájában).

Neurális hálók tanítása

- a neuronok bemeneteihez rendelt súlyok ideális értékeinek meghatározása
- a hálózat elvárt és valós kimenetei közötti különbség által számított E hiba, vagy költségfüggvény (error/cost function) segítségével történik
- Cél az olyan súlytényező értékek megtalálása, amikkel a háló a legkisebb hibát éri el

Modern mesterséges neurális hálózatok alapjai

Neurális hálók tanítása

Perceptron Learning Rule

- perceptron tanulási szabály
- egyszerű (egyrétegű, klasszikus) perceptron hálóknál alkalmazzák
- gradiens alapú megoldást használnak
 - minden esetben a hibafüggvény lehető legnagyobb csökkenésének irányába viszi el a paramétert
 - lokális optimumot garantálva
- A módosításához használt érték meghatározására a hibafüggvény deriváltját (gradiensét) használjuk az adott súlytényezők mellett
- csak a klasszikus, küszöbfüggvényt alkalmazó hálózatok esetében használható

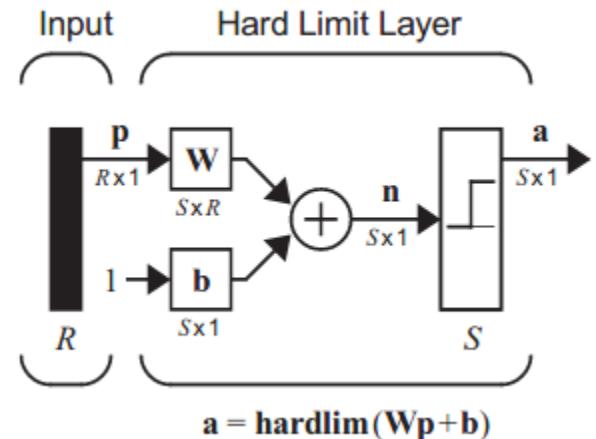
$$E(w) = \frac{1}{2} \sum_p \sum_j (y_d - y)^2$$

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial w} \cdot \frac{\partial y}{\partial w} = -(y_d - y)x = -\delta x$$

Perceptron Learning Rule

Új súly értékek: $\mathbf{W}^{new} = \mathbf{W}^{old} + \epsilon \mathbf{p}^T$

Eltolás új értéke: $\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$



$e = t - a$ az elvárt kimenet és a neurális hálózat kimenetének különbsége

Lineárisan szeparálható feladatokra alkalmazható → ekkor garantáltan véges számú lépés alatt konvergál

Perceptron Learning Rule

Példa:

Egyetlen perceptronból álló hálózat (3 bemenet, 1 kimenet)

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, t_1 = [0] \right\}$$

$$\left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = [1] \right\}$$

1. Inicializálás: kezdeti értékadás $\mathbf{W} = [0.5 \ -1 \ -0.5]$, $b = 0.5$

2. Kimenet és hiba kiszámítása p1-re

$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_1 + b) = \text{hardlim}\left(\begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0.5\right)$$
$$= \text{hardlim}(2.5) = 1$$

$$e = t_1 - a = 0 - 1 = -1$$

Perceptron Learning Rule

Példa:

3. Súlyok és eltolás frissítése

$$\begin{aligned}\mathbf{W}^{new} &= \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} + (-1) \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix}.\end{aligned}$$

$$b^{new} = b^{old} + e = 0.5 + (-1) = -0.5$$

Második iteráció p2-vel:

Kimenet és hiba kiszámítása p1-re

$$\begin{aligned}a &= \text{hardlim}(\mathbf{W}\mathbf{p}_2 + b) = \text{hardlim}(\begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + (-0.5)) \\ &= \text{hardlim}(-0.5) = 0\end{aligned}$$

$$e = t_2 - a = 1 - 0 = 1$$

Súlyok és eltolás frissítése

$$\begin{aligned}\mathbf{W}^{new} &= \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} 0.5 & 1 & -0.5 \end{bmatrix} + (-1) \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} -0.5 & 2 & 0.5 \end{bmatrix}\end{aligned}$$

Addig iterálunk, amíg a hálózat hiba nullára nem csökken (minden tanító pontra a neurális hálózat kimenete megegyezik az elvárt kimenettel)

Delta Learning Rule

- Hasonlít a perceptron tanuláshoz
 - Eltérés, hogy az aktivációs függvény tetszőleges differenciálható függvény lehet
- hívják még Widrow-Hoff tanulási szabálynak is
- A modern megoldások egy speciális esete
 - Backpropagation
- nagy előrelépést jelentett a bonyolultabb aktivációs függvényeket alkalmazó hálózatok területén
- a több réteggel rendelkező topológiák esetén nem volt alkalmazható

Delta Learning Rule

Tanító adat: $\mathbf{x}^n = (X_0^n, \dots, X_N^n)$

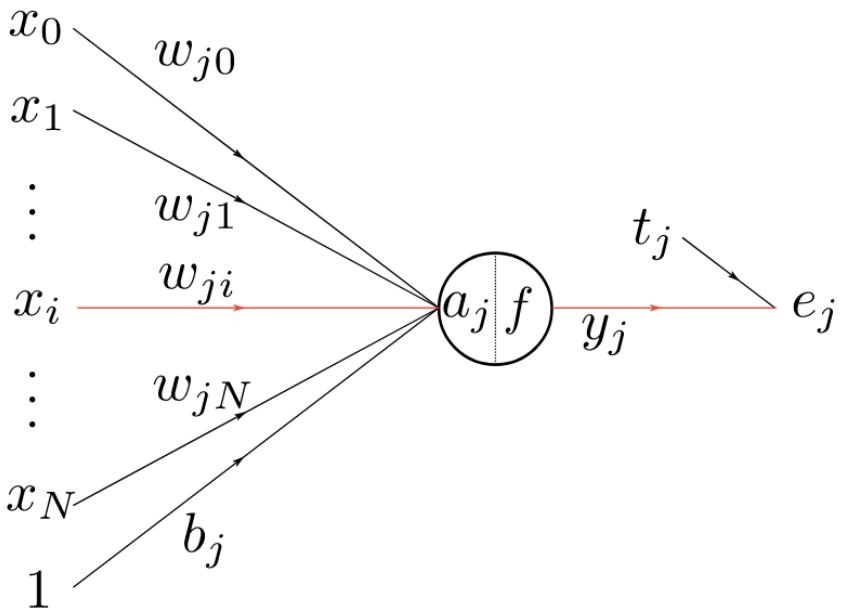
Neuron kimenete: $y_j^n = f(a_j^n)$

Ahol f az aktivációs függvény
(deriválhatónak kell lennie)

$$a_j^n = \sum_i w_{ji} X_i^n$$

Hiba: neuron kimeneti értékének és az elvárt kimenetnek a különbsége: $e_j^n = y_j^n - t_j^n$

Átlagos négyzetes eltérés: $E^n = \frac{1}{2} \sum_j (e_j^n)^2$



Delta Learning Rule

$$\frac{\partial E^n}{\partial w_{ji}} = \frac{\partial E^n}{\partial e_j^n} \frac{\partial e_j^n}{\partial y_j^n} \frac{\partial y_j^n}{\partial a_j^n} \frac{\partial a_j^n}{\partial w_{ji}}$$

$$\frac{\partial E^n}{\partial e_j^n} = e_j^n \quad \frac{\partial e_j^n}{\partial y_j^n} = 1 \quad \frac{\partial y_j^n}{\partial a_j^n} = f' (a_j^n) \quad \frac{\partial a_j^n}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left(\sum_i w_{ji} X_i \right) = X_i,$$

Így:

$$\frac{\partial E^n}{\partial w_{ji}} = e_j^n f' (a_j^n) X_i$$

Lokális gradiens: $\delta_j^n \equiv \frac{\partial E^n}{\partial a_j^n}$

$$\begin{aligned} &= \frac{\partial E^n}{\partial e_j^n} \frac{\partial e_j^n}{\partial y_j^n} \frac{\partial y_j^n}{\partial a_j^n} \\ &= e_j^n f' (a_j^n), \end{aligned}$$
$$\frac{\partial E^n}{\partial w_{ji}} = \delta_j^n X_i$$

→ e_j^n hiba és az aktivációs függvény deriváltjának értékének szorzata

Delta szabály a súlyok módosítására:

$$\boxed{\Delta w_{ji} = -\gamma \delta_j^n x_i}$$

γ tanulási tényező

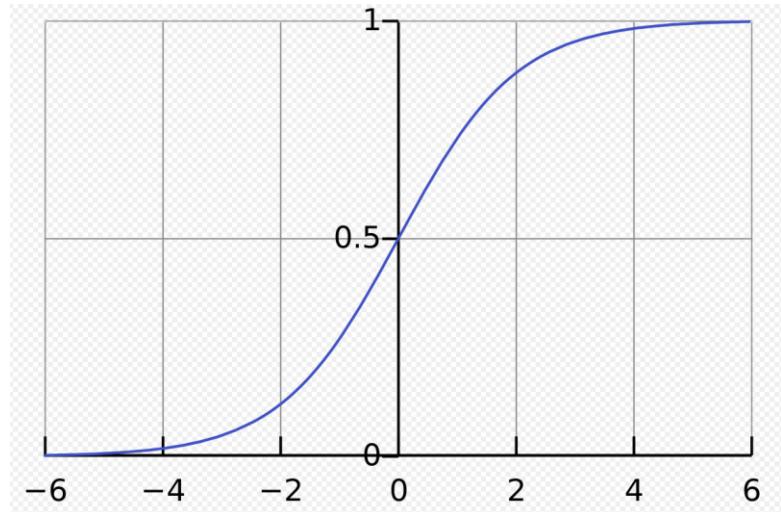
Delta Learning Rule

Szigmoid aktivációs függvény deriváltja:

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

$$\frac{dy}{dx} = -\frac{1}{(1+e^{-x})^2} (-e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{1}{1+e^{-x}} \left(1 - \frac{1}{1+e^{-x}} \right) = y(1-y)$$



Emlékeztető: $f(x) = \frac{g(x)}{h(x)}$ $f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{[h(x)]^2}$

Modern mesterséges neurális hálózatok alapjai

Neurális hálók tanítása:
backpropagation

Backpropagation

- szintén a gradiens módszer segítségével minimalizálja a hibafüggvény értékét
- alkalmas többrétegű neurális hálók kezelésére
- hatékonysága abban rejlik, hogy a hibát a hálózat rétegein külön-külön vezeti vissza
 - ennek megfelelően módosítja a súlytényezőket
 - a láncszabály elvét veszi alapul
- felügyelt tanítási módszer
 - De alkalmazzák nemfelügyelt tanításra is

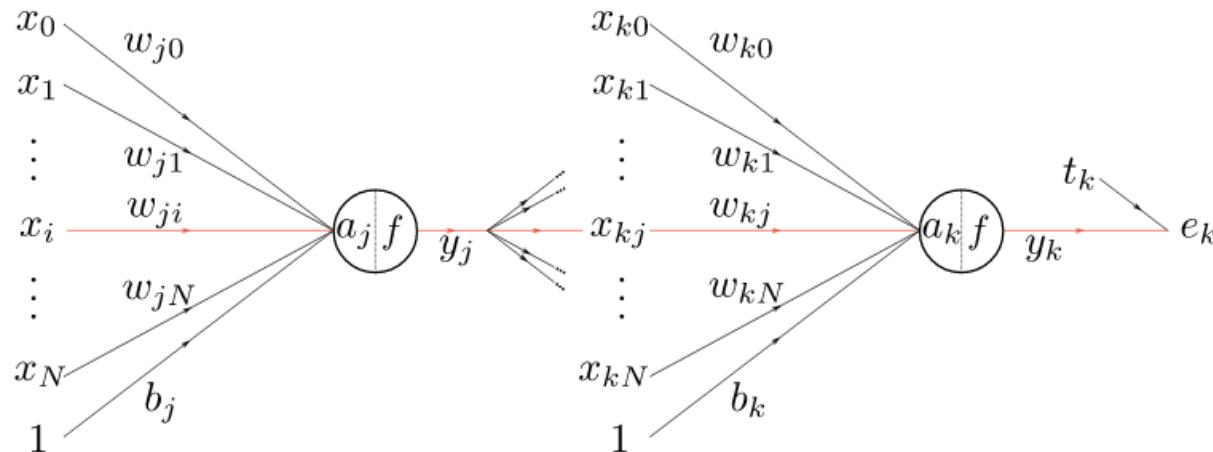
Backpropagation

- A gyakorlatban
 - a hibafüggvény meghatározására összetettebb függvényeket alkalmaznak, mint például a kereszt entrópia (Cross-Entropy), Logit, recall, precision stb
 - sztochasztikus gradiens módszereket szoktak alkalmazni
 - például az Adam keresés.

Backpropagation

- Fő lépései
 - (Tanítási minta kiválasztása)
 - Minta előre terjesztése
 - kiértékelés
 - Hiba kiszámítása
 - elvárt és eredmény kimenetek közti eltérés
 - Hiba visszaterjesztése
 - Módosított súlytényező kiszámítása
 - paraméter gradiensek
 - Súlytényezők módosítása
- A lépéseket addig ismételjük a tanítási mintákra, míg a hiba a kívánt tartományba nem esik

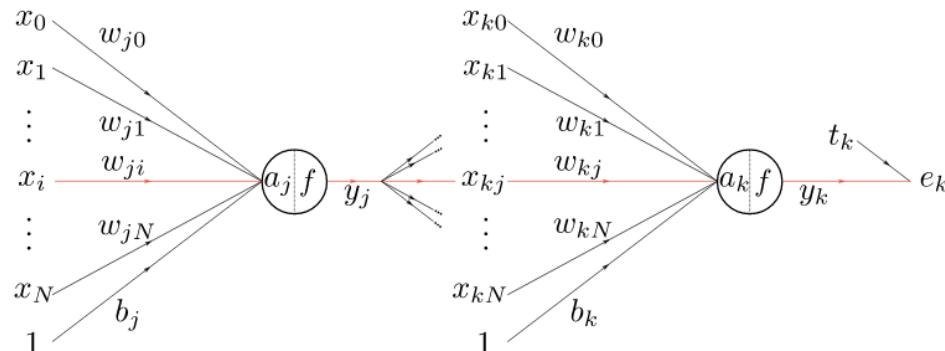
Backpropagation



Kimeneti rétegben:

$$\begin{aligned}
 \frac{\partial E^n}{\partial w_{kj}} &= \frac{\partial E^n}{\partial a_k^n} \frac{\partial a_k^n}{\partial w_{kj}} \\
 \frac{\partial E^n}{\partial w_{kj}} &= \frac{\partial E^n}{\partial e_k^n} \frac{\partial e_k^n}{\partial y_k^n} \frac{\partial y_k^n}{\partial a_k^n} \frac{\partial a_k^n}{\partial w_{kj}} \\
 &= -e_k^n f'(a_k^n) x_{kj}^n \\
 &= \delta_k^n x_{kj}^n \\
 &= \delta_k^n y_j^n.
 \end{aligned}$$

Backpropagation



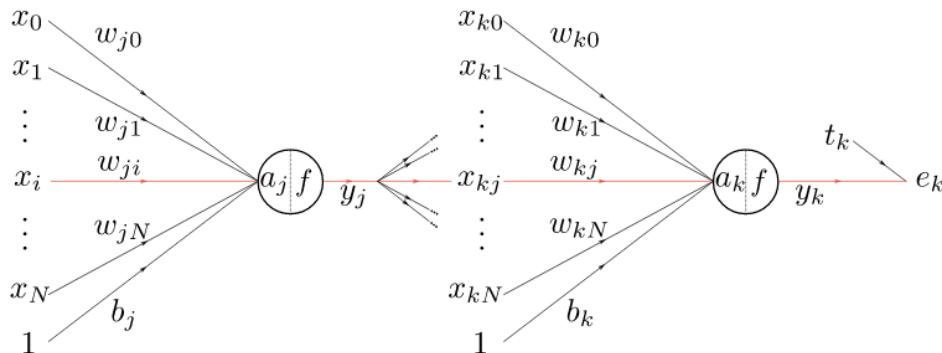
Rejtett rétegen: w_{ji} a következő réteg összes neuronjára hatással van
Láncszabály alkalmazása:

$$\frac{\partial E^n}{\partial w_{ji}} = \underbrace{\left(\sum_k \frac{\partial E^n}{\partial e_k^n} \frac{\partial e_k^n}{\partial y_k^n} \frac{\partial y_k^n}{\partial a_k^n} \frac{\partial a_k^n}{\partial y_j^n} \right)}_{\text{output neurons}} \underbrace{\frac{\partial y_j^n}{\partial a_j^n} \frac{\partial a_j^n}{\partial w_{ji}}}_{\text{hidden neuron}},$$

$$\frac{\partial E^n}{\partial w_{ji}} = \left(\sum_k \delta_k^n y_j^n \frac{\partial a_k^n}{\partial y_j^n} \right) \frac{\partial y_j^n}{\partial a_j^n} \frac{\partial a_j^n}{\partial w_{ji}}$$

$$\frac{\partial E^n}{\partial w_{ji}} = \left(\sum_k \delta_k^n w_{kj} \right) f' \left(a_j^n \right) x_i = \delta_j^n x_i$$

Backpropagation



Összefoglalva: súlyok módosítása az általános delta-szabállyal $\Delta w_{ji} = -\gamma \delta_j^n x_i$

ahol $\delta_i^n = \begin{cases} f' \left(a_j^n \right) e_j^n & \text{ha } j \text{ kimeneti neuron} \\ f' \left(a_j^n \right) \left(\sum_j \delta_j^n w_{ji} \right) & \text{ha } j \text{ rejtett neuron} \end{cases}$

Modern mesterséges neurális hálózatok alapjai

Neurális hálók tanításának
kulcsfogalmai

Tanulás

- modern mély tanulási keretrendszerek
 - például a TensorFlow
 - minimális kódmező
 - egy egyszerű függvényhívás segítségével tanítható
 - egyszerű prototipizálásra alkalmas megoldás
- a mélyben zajló folyamatok megértése elengedhetetlen a komplexebb rendszerek professzionális tervezéséhez és építéséhez
- A területen fellelhető szakirodalom könnyebb megértése érdekében az alábbiakban a neurális hálók tanításánál felmerülő fontosabb kulcsfogalmak.

Neurális hálók tanításának kulcsfogalmai

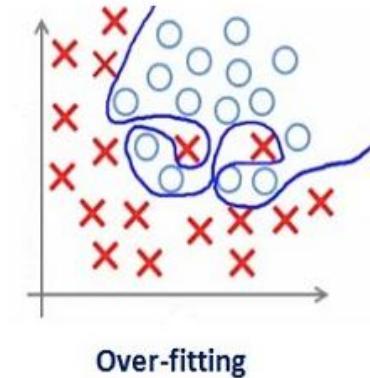
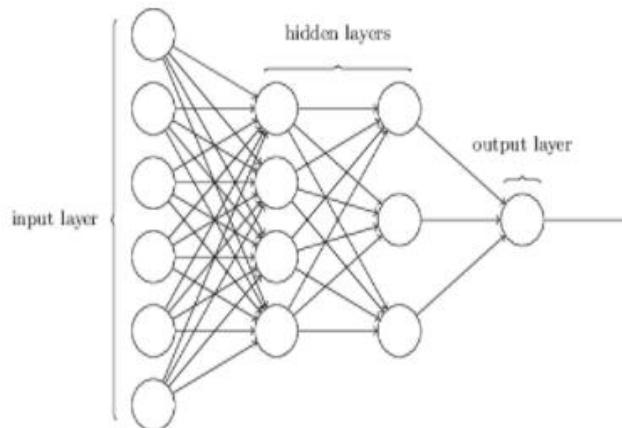
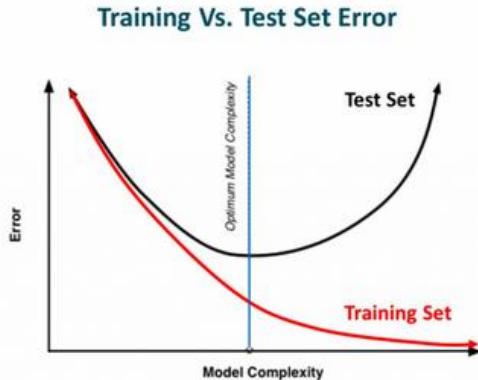
- Minta (sample)
 - egy konkrét bemenet a hálózat számára,
- Tanítási minta (training sample)
 - kimondottan a hálózat tanítására szolgáló minta, jellemzően ez felügyelt tanítás esetén együtt jár az hálózat elvárt kimeneti értékével,
- Validációs minta (validation sample)
 - a neurális hálózat által megtanult modell tesztelésére szolgáló, a tanítási minták között nem szereplő minták,
- Mintakészlet (sample set)
 - minták gyűjteménye,
- Szakasz (epoch)
 - a teljes mintakészlet egyszeri átadása a hálózat számára,
- Soros, vagy szekvenciális (sequential) tanítás
 - online típusú tanítás, amikor minden egyes mintát követően hangolásra kerülnek a hálózatban használt súlyok,

Neurális hálók tanításának kulcsfogalmai

- Köteg (batch)
 - offline típusú tanítás esetén a súlyok a minták egy csoportja alapján kerül módosításra, nem pedig mintánként a soros tanítástól eltérően,
- Kiesés, vagy elhalálozás (dropout)
 - amikor a tanítási folyamat során valamelyen okból kifolyólag neuron(ok) kapcsolódását reprezentáló súlytényezők 0 értéket vesznek fel, vagyis a hálózat nem veszi figyelembe annak az értékét, „kimarad” a számításból
 - Esetenként ez lehet kívánatos (regularizálás miatt), de elkerülendő is, amit az aktivációs függvénnyel, illetve a tanuló algoritmus megfelelő paraméterezésével küszöbölnek ki,
- Túltanítás, vagy túlillesztés (overfitting)
 - amikor a modell nem valós összefüggéseket is megtanul
 - túl jól illeszkedő megoldást talál, ami általános esetekre vonatkozóan azonban már nem teljesít megfelelően
- Alultanulás, vagy alulillesztés (underfitting)
 - akkor jelentkezik, amikor nem megfelelő a hálózat mérete,
- Regularizálás
 - a túltanítás elkerülésére alkalmazott technikák
 - cél a megtanult modell általánosítása

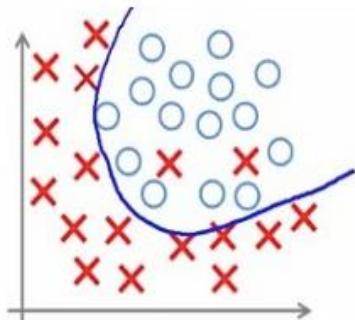
Neurális hálók tanításának kulcsfogalmai

Túltanulás, túlillesztés

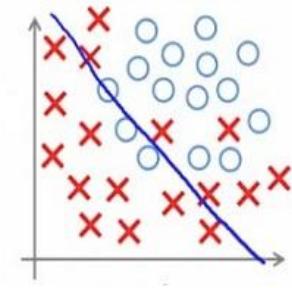
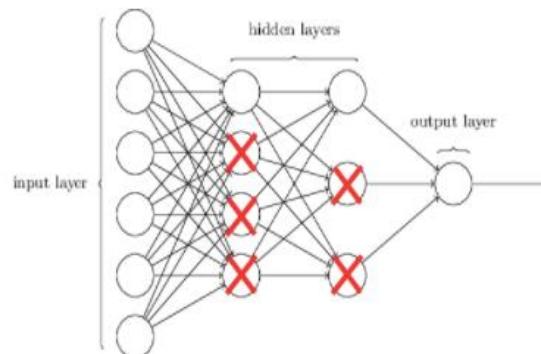


Regularizáció: hálózat súlyainak csökkentése

Túl nagy regularizációs ráta →



Appropriate-fitting



Neurális hálók tanításának kulcsfogalmai

- Generalizálás
 - a neurális hálók alapvető képessége a mintákból való általánosított (generalizált) jellemzők megtanulása,
- Tanulási sebesség (learning rate)
 - a súlytényezők módosításának mértékét befolyásoló paraméter,
 - kicsi érték esetén a tanulás lassú, de stabilan konvergál
 - nagy érték esetén a tanulás gyors, de előfordulhat, hogy nem konvergál
 - általában értékét fokozatosan csökkentik
- Hiper-paraméterek (hyperparameters)
 - a hálózatra jellemző paraméterek
 - a struktúrát írják le
 - például a rétegek és az azokban található neuronok száma, vagy a tanulási sebesség,

Neurális hálók tanításának kulcsfogalmai

- Adat dúsítás (data augmentation)
 - a mély tanulási módszerek hatalmas adatigénye
 - gyakori problémát okoz az olyan területeken, ahol az adatgyűjtés költséges, vagy nehezen megoldható
 - a meglévő mintkából új mintákat állítunk elő és ezeket is felhasználjuk a tanítás során
- Egy lövéses (one-shot) tanulás
 - egyetlen minta alapján képesek a neurális hálózat paramétereit hangolni
 - egy kutatási irány a hiányos adathalmazok orvoslására
- Átadásos tanulás (transfer learning)
 - olyan technikák, amelyek egy már valamilyen mintakészlet alapján betanított hálózatot használunk fel
 - lényegében a forrásként használt modellben található tudást adja át az új hálózatunknak

Felhasznált források

- <https://selfdrivingcars.mit.edu/deeptraffic/>
- <https://www.asimovinstitute.org/neural-network-zoo-prequel-cells-layers/#comments>
- <http://www.asimovinstitute.org/neural-network-zoo/>
- <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>
- <http://www.deeplearningbook.org/>
- <https://deeplearning.mit.edu/>
- <https://arxiv.org/pdf/1602.05179.pdf>
- <https://eng.uber.com/deep-neuroevolution/>
- <https://arxiv.org/pdf/1703.03864.pdf>
- <http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/>
- <https://mml-book.github.io/>
- <http://themlbook.com/wiki/doku.php>
- <http://d2l.ai/>
- <https://christophm.github.io/interpretable-ml-book/>
- https://www.researchgate.net/figure/CRConvNet-the-Character-Recognition-Convolutional-Neural-Network-architecture_fig14_235665033
- <https://arxiv.org/abs/1605.07678>
- <https://arxiv.org/pdf/1412.6980.pdf>
- <https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b>
- <https://openai.com/blog/better-language-models/>
- <https://openai.com/blog/language-unsupervised/>
- http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf
- http://axon.cs.byu.edu/~martinez/classes/678/Papers/Hopfield_Chapter.pdf
- <https://cs.stanford.edu/~quocle/tutorial2.pdf>
- <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>
- <https://www.anotsorandomwalk.com/backpropagation-example-with-numbers-step-by-step/>