

Üzleti Elemzések Módszertana

10. Előadás: Neurális hálózatok

Kuknyó Dániel
Budapesti Gazdasági Egyetem

2023/24
2.félév

1 Neurális hálók felépítése

2 Architektúra

3 Tanítás

4 Konvolúciós hálózatok

1 Neurális hálók felépítése

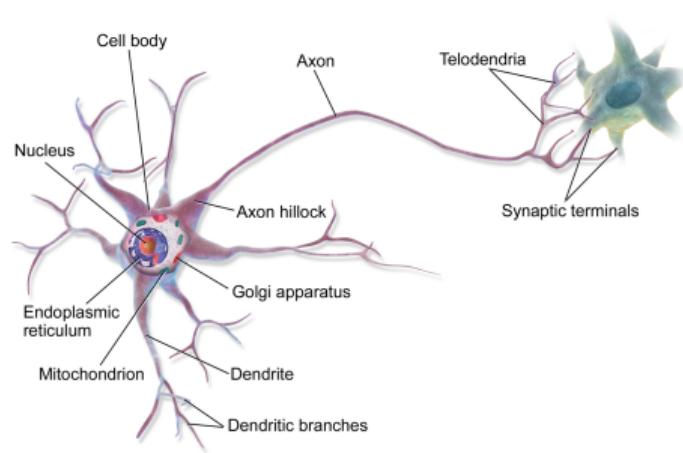
2 Architektúra

3 Tanítás

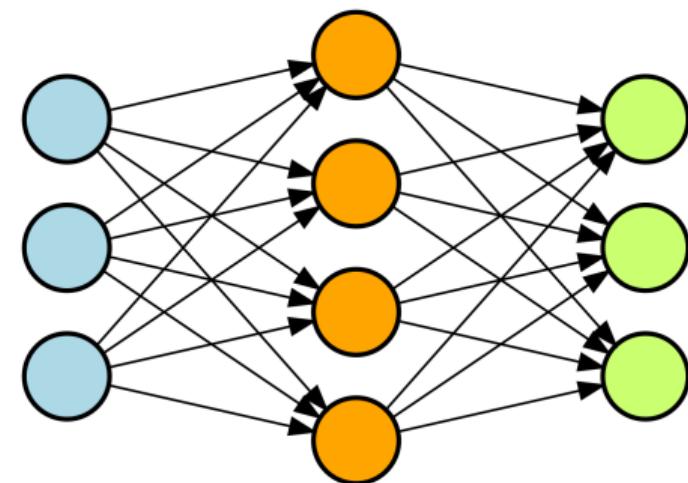
4 Konvolúciós hálózatok

A biológiai neurontól a mesterségesig

A madarak repülésre inspirálták az embert, a bogáncsok a tépőzárat ihlették. A logikus lépés az volt, hogy az agy által inspirálódva megpróbál az ember gépeket létrehozni.



Ahogy repülők sem csapkodnak a szárnyaikkal, a mesterséges neuronok is meglehetősen különböznek a biológiai unokatestvéreiktől.



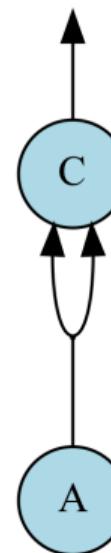
Logikai számítások neuronokkal

Warren McCulloch és Walter Pitts modellezte először a biológiai neuront, ami később a mesterséges neuronként vált ismertté. Egy bináris kimenete, és több bináris bemenete van.

A mesterséges neuron akkor aktiválja az outputját, ha az inputjai meghatározott számban aktiválódnak.

Bármilyen összetett logikai kifejezés kifejezhető ilyen neuronokkal: a példában az identitás, és, vagy, xvagy kifejezések konfigurációi láthatók.

$$B = A$$



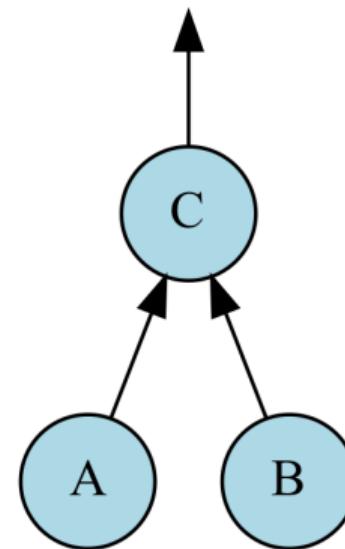
Logikai számítások neuronokkal

Warren McCulloch és Walter Pitts modelleztek először a biológiai neuront, ami később a mesterséges neuronként vált ismertté. Egy bináris kimenete, és több bináris bemenete van.

A mesterséges neuron akkor aktiválja az outputját, ha az inputjai meghatározott számban aktiválódnak.

Bármilyen összetett logikai kifejezés kifejezhető ilyen neuronokkal: a példában az identitás, és, vagy, xvagy kifejezések konfigurációi láthatók.

$$C = A \wedge B$$



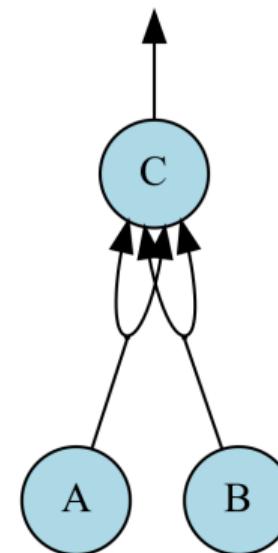
Logikai számítások neuronokkal

Warren McCulloch és Walter Pitts modellezte először a biológiai neuront, ami később a mesterséges neuronként vált ismertté. Egy bináris kimenete, és több bináris bemenete van.

A mesterséges neuron akkor aktiválja az outputját, ha az inputjai meghatározott számban aktiválódnak.

Bármilyen összetett logikai kifejezés kifejezhető ilyen neuronokkal: a példában az identitás, és, vagy, xvagy kifejezések konfigurációi láthatók.

$$C = A \vee B$$



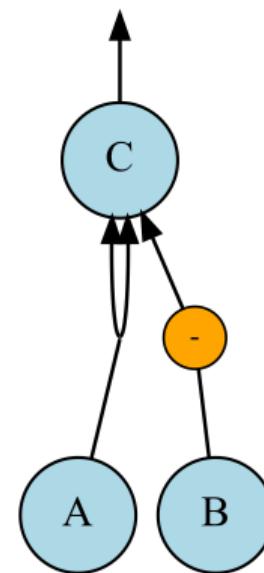
Logikai számítások neuronokkal

Warren McCulloch és Walter Pitts modelleztek először a biológiai neuront, ami később a mesterséges neuronként vált ismertté. Egy bináris kimenete, és több bináris bemenete van.

A mesterséges neuron akkor aktiválja az outputját, ha az inputjai meghatározott számban aktiválódnak.

Bármilyen összetett logikai kifejezés kifejezhető ilyen neuronokkal: a példában az identitás, és, vagy, xvagy kifejezések konfigurációi láthatók.

$$C = A \wedge \neg B$$



1 Neurális hálók felépítése

2 Architektúra

3 Tanítás

4 Konvolúciós hálózatok

A perceptron

Az egyik legegyszerűbb neurális modell a **perceptron**. Alapja a **küszöbönkénti egység**. Inputjai és outputja valós számok.

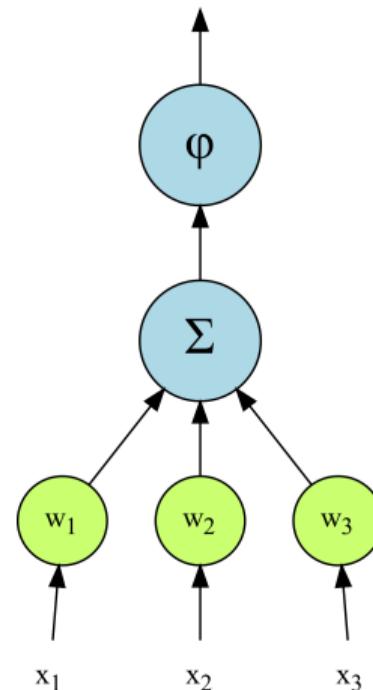
A perceptron minden x_i inputjához egy w_i súly tartozik.

Első lépésben a modell kiszámolja inputjainak súlyozott szorzatösszegét:

$$z = w_1x_1 + w_2x_2 + \dots + w_nx_n = X^T W$$

Majd ezt az eredményt behelyettesíti egy aktivációs függvénybe:

$$h_w(x) = \varphi(z)$$

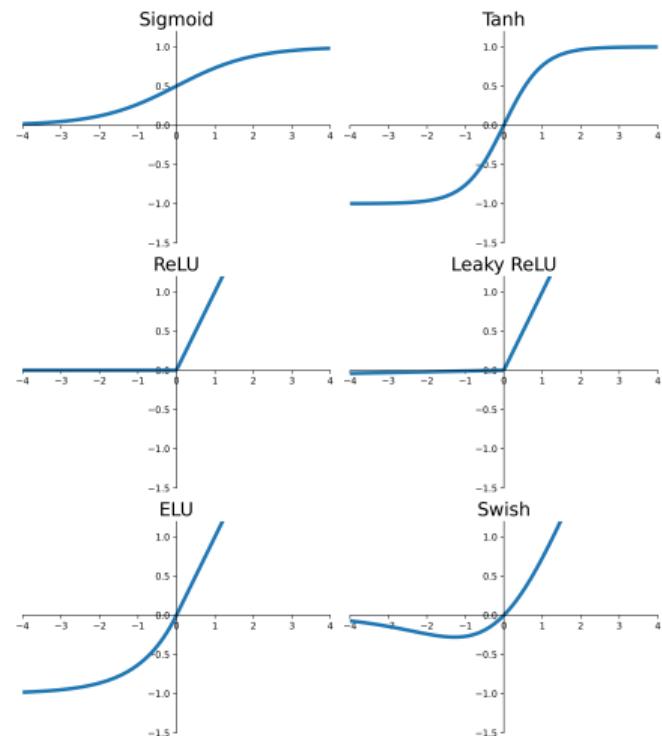


Gyakori aktivációs függvények

A neuron által kiszámolt súlyozott szorzatösszeg egy aktivációs függvénybe kerül behelyettesítésre.

A neurális hálózatok az aktivációs függvények segítségével **sajátítanak el komplex mintázatokat**. Az aktivációs függvény vezeti be a neurális hálózatokba a **nemlineáris transzformációt**, enélkül csak egy lineáris transzformáció lenne.

A különböző alkalmazásokra külön aktivációs függvények használatosak.



Gyakori aktivációs függvények

$$\text{Sigmoid}(x) = \frac{1}{1+e^x}$$

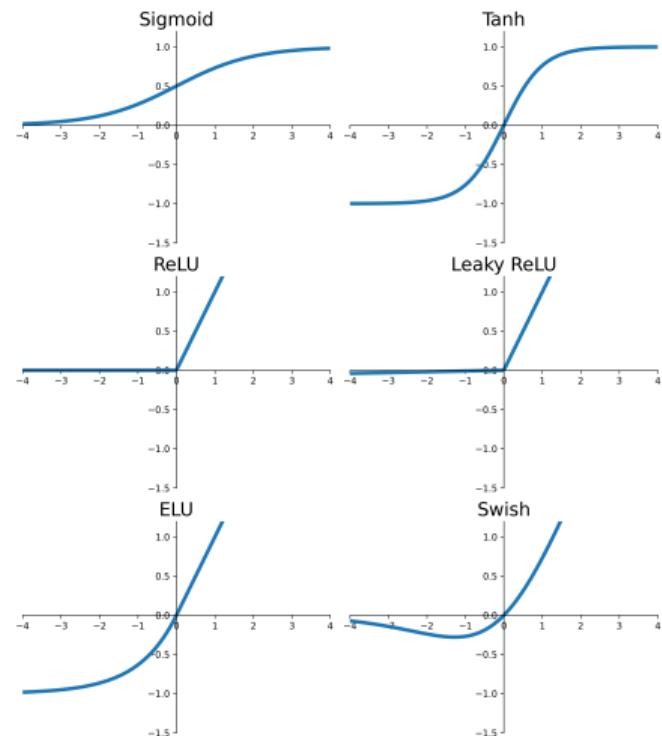
$$\text{ReLU}(x) = \max(0, x)$$

$$\text{ELU}(x) = \begin{cases} x & \text{ha } x \geq 0 \\ \alpha(e^x - 1) & \text{ha } x < 0 \end{cases}$$

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

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{ha } x \geq 0 \\ \alpha \cdot x & \text{ha } x < 0 \end{cases}$$

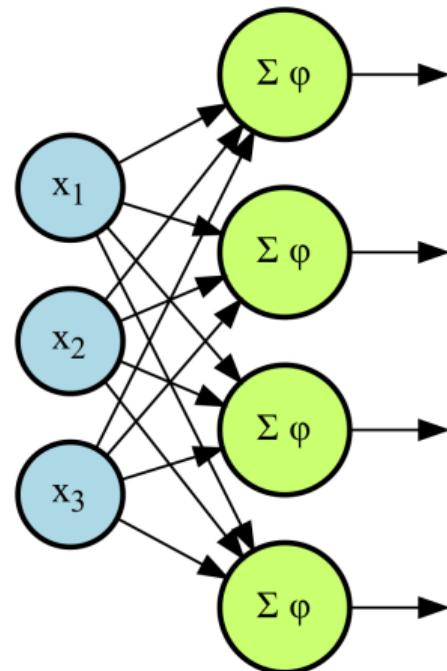
$$\text{Swish}(x, \beta) = \frac{x}{1+e^{-\beta \cdot x}}$$



Többrétegű perceptron

A többrétegű perceptron esetén a neuronok rétegekbe szerveződötten működnek. Az információ a bemeneti réteg felől a kimeneti réteg irányába áramlik.

A legelső réteg neuronjai az adathalmaz jellemzőivel állnak kapcsolatban.
Minden további réteg kapcsolata pedig az előző réteg kapcsolataival.



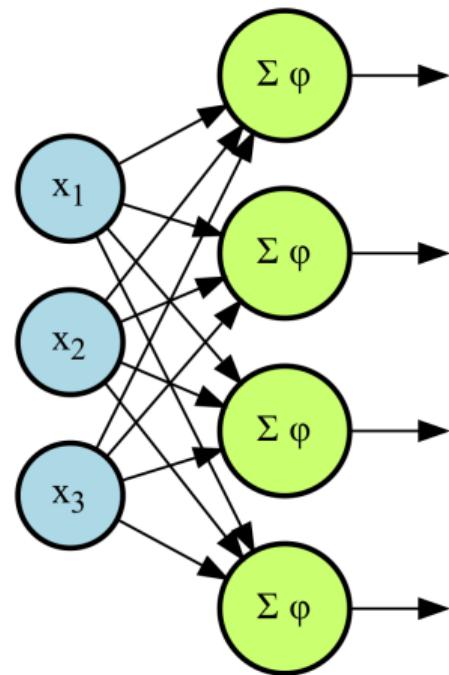
Többrétegű perceptron

Teljesen becsatolt réteg kimenete

$$h_{w,b}(X) = \varphi(XW + b)$$

Ahol:

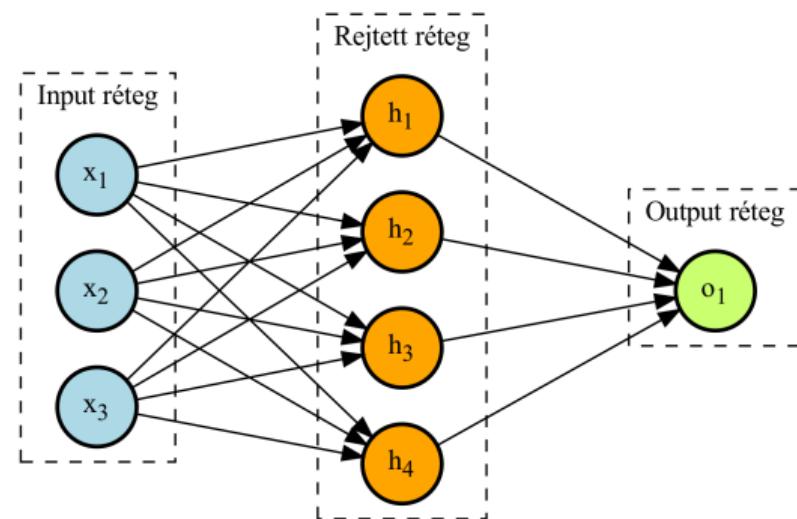
- X : Tanító adathalmaz mátrixa
- W : A súlyok mátrixa
- b : Az eltolásokat tartalmazó vektor
- φ : Az aktivációs függvény



Regressziós architektúra

A többrétegű hálózat egy input réteget, tetszőleges számú rejtett réteget és egy output réteget tartalmaz.

Regressziós problémák esetén az output rétegenből **egy output neuron található**, és a predikció a neuron output értéke.

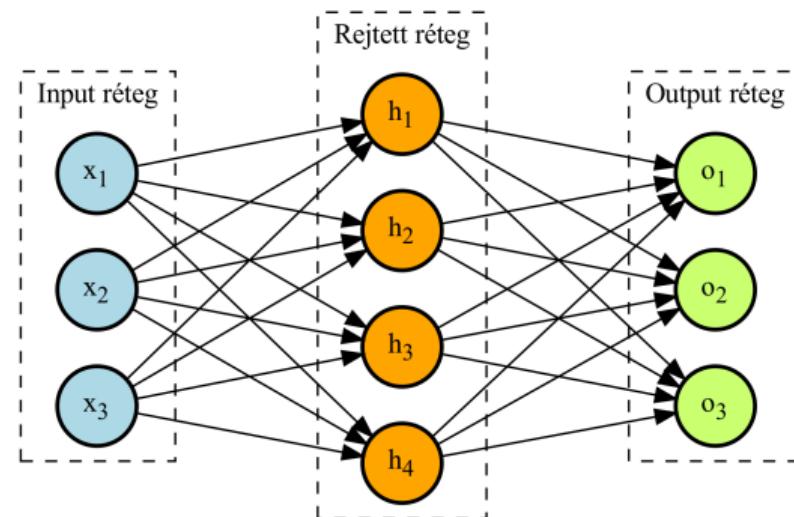


Osztályozó architektúra

Osztályozás esetén a kimeneti rétegbe annyi neuron kerül, ahány lehetséges osztálya van az adatoknak.

Ebben az esetben a neuronok aztbecsülik meg, hogy az adott mintaegyed mekkora valószínűsséggel tartozik az adott neuronhoz tartozó osztályba. Ezáltal egy multinomiális valószínűgeloszlás áll elő a kimeneti rétegen.

Az osztályozó réteg aktivációs függvénye a softmax.



1 Neurális hálók felépítése

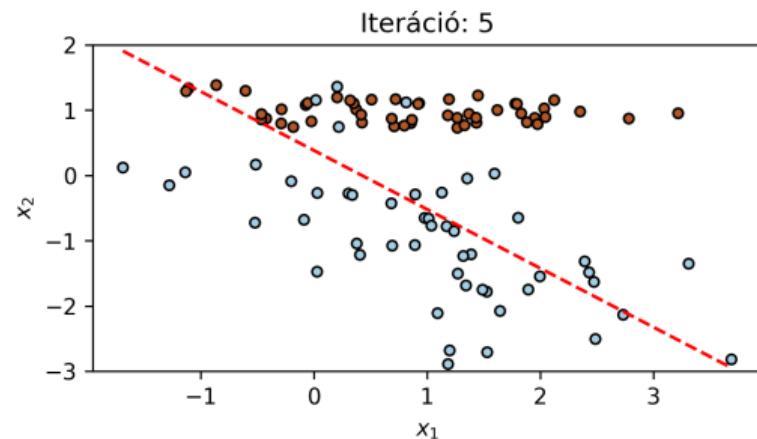
2 Architektúra

3 Tanítás

4 Konvolúciós hálózatok

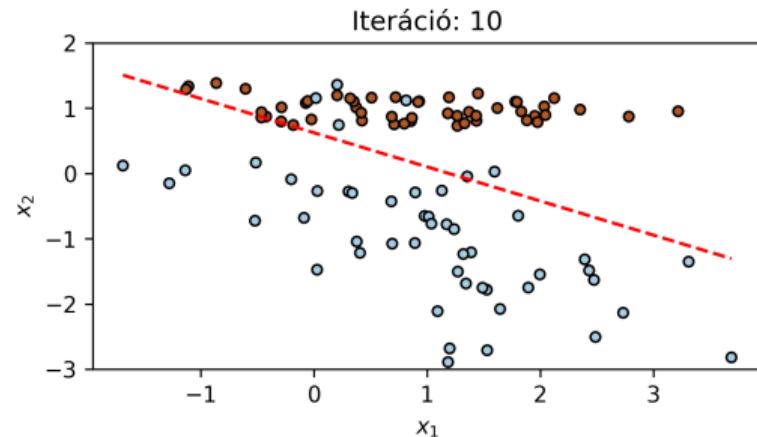
A perceptron tanítása

Tanítás során egyszerre egy mintaegyed áramlik át a hálózaton. **A modell a mintaegyedre predikciót készít, majd összeveti a címkével.** Ennek alapján tudja kiszámítani a költséget és frissíteni a súlyait.



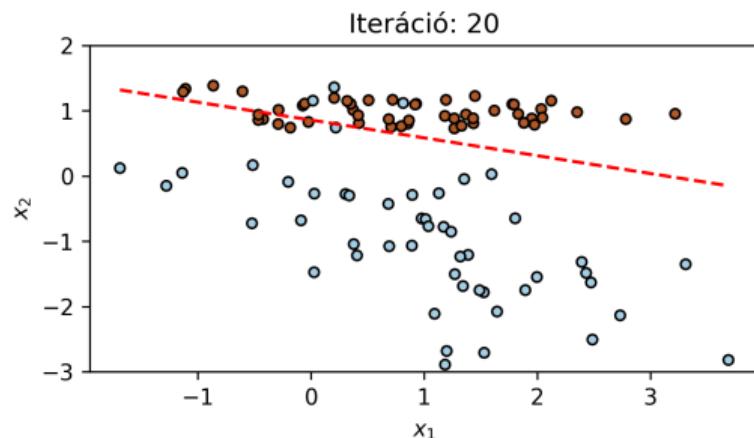
A perceptron tanítása

Tanítás során egyszerre egy mintaegyed áramlik át a hálózaton. **A modell a mintaegyedre predikciót készít, majd összeveti a címkével.** Ennek alapján tudja kiszámítani a költséget és frissíteni a súlyait.



A perceptron tanítása

Tanítás során egyszerre egy mintaegyed áramlik át a hálózaton. A modell a mintaegyedre predikciót készít, majd összeveti a címkével. Ennek alapján tudja kiszámítani a költséget és frissíteni a súlyait.



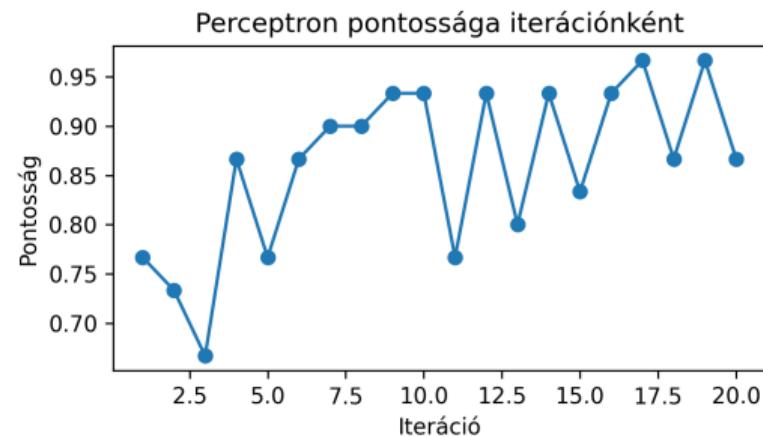
A perceptron tanítása

Perceptron tanítás szabálya

$$w_{i,j} \leftarrow w_{i,j} + \alpha (y_j - \hat{y}_j) x_i$$

Ahol:

- $w_{i,j}$: Kapcsolati súly az i input neuron és j output neuron között
- x_i : Aktuális mintaegyed
- \hat{y}_j : j output neuron predikciója
- y_j : j neuronhoz tartozó címke
- α : Tanulási sebesség

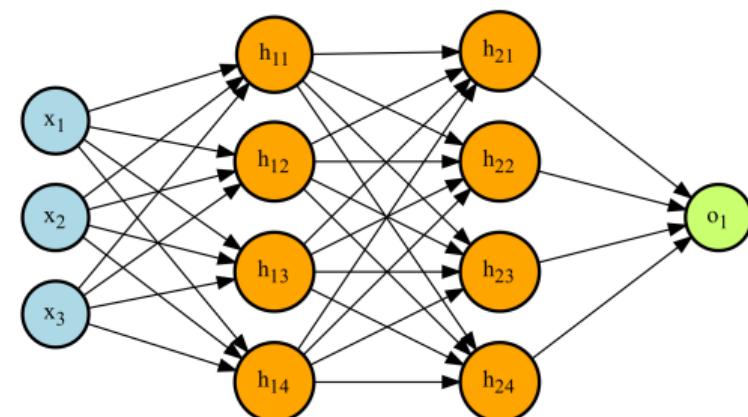


Neurális hálózat tanítása

Egy teljes neurális hálót a perceptronról különbözően kell tanítani, mert több egységből áll. Ennek eljárása a **hiba visszaterjesztő algoritmus**.

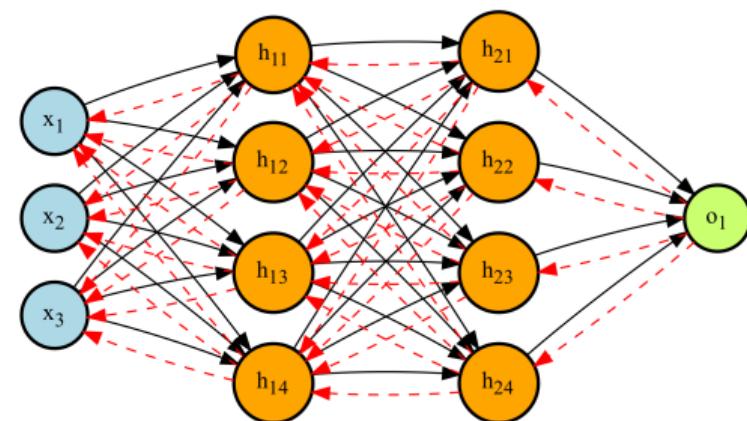
A hiba visszaterjesztő algoritmus összesen két iteráció alatt (egy előre, egy hátra) képes becslést adni a hálózat minden súlyára.

A mintaegyedek kötegekben áramlanak át a hálózaton. A kötegek mérete szabályozható. minden köteg feldolgozása után kerül sor hiba számításra és visszaáramoltatásra.



Neurális hálózat tanítása

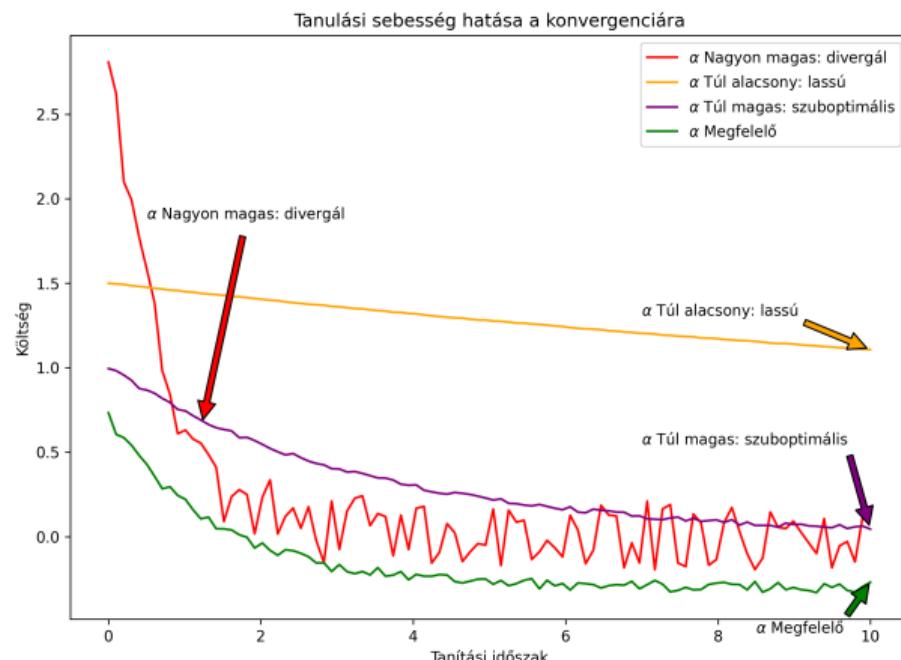
- ➊ Egy köteget átenged a hálózaton, és közben minden réteg predikcióját elmenti.
- ➋ Hálózat output hibájának kiszámítása
- ➌ Kiszámolja, hogy az egyes kimeneti kapcsolatok mennyiben járultak hozzá a teljes hibához.
- ➍ Kiszámolja, hogy a kimeneti kapcsolatok hibái mennyiben származtathatók az előző kapcsolatok hibájából, majd ezt elvégzi minden rétre.
- ➎ Súlyok frissítése a hibák alapján.



Tanulási sebesség

A frissítések mérete a tanulási sebességtől és az optimalizálótól algoritmusról függ. Az optimális tanulási sebességre az iterációnkénti költségből lehet következtetni.

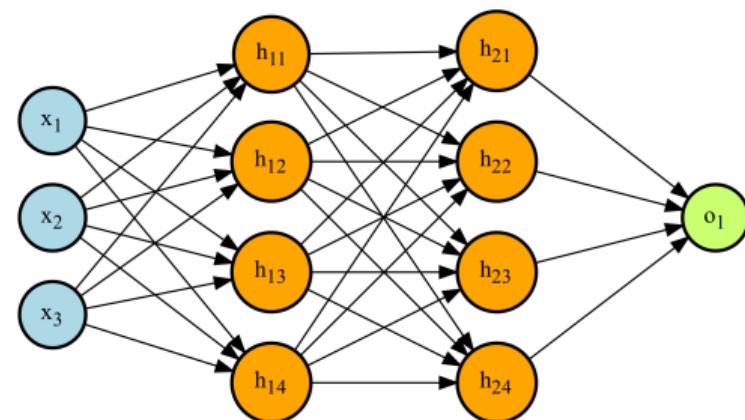
Érdemes először egy magasabb tanulási sebességgel elkezdeni a tanítást, majd folyamatosan csökkenteni.



Regularizáció neurális hálók esetén

A neurális hálózatok fogékonyak a túltanulásra, ha túlságosan magas a paraméterek száma.

Az optimalizálás során lehetséges az ℓ_1, ℓ_2 normák használata, ez nagyon hasonló működést fog eredményezni, mint a LASSO vagy Ridge regressziók esetén.

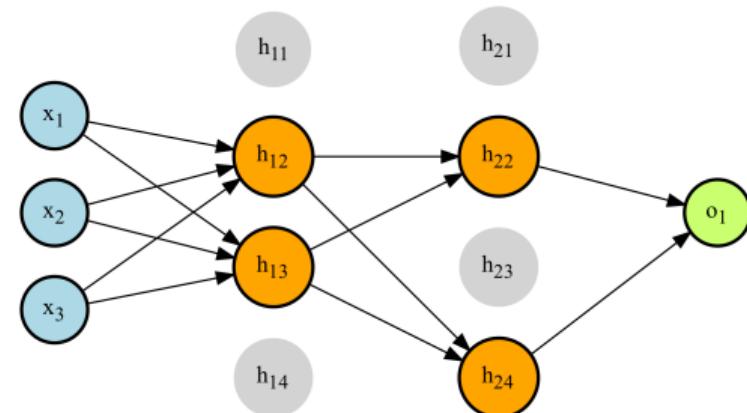


Regularizáció neurális hálók esetén

Egy másik regularizációs eljárás a **kiesés**.

Kiesés során minden neuron kap p valószínűséget arra, hogy ne legyen figyelembe véve a tanítás során. Ez általában 50%.

Kieséssel minden tanítási lépésben egy egyedi architektúra tanul, ez összesen 2^N lehetőség.

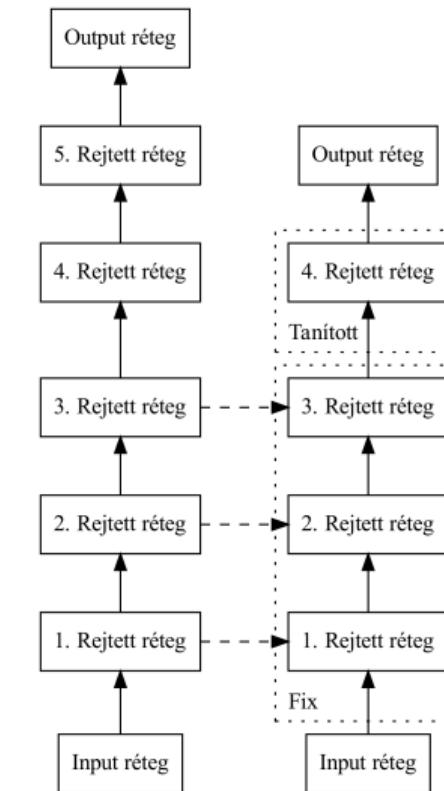


Transzfertanulás

Általában nem célszerű egy komplex mélyhálózatot a semmiből feltanítani.

Egy jobb megoldás egy hasonló feladatot ellátó neurális hálózat súlyainak átemelése az újonnan létrehozott modellbe.

Minél hasonlóbbak a feladatok, annál több réteget lehet felhasználni a már meglévő hálózatból.



1 Neurális hálók felépítése

2 Architektúra

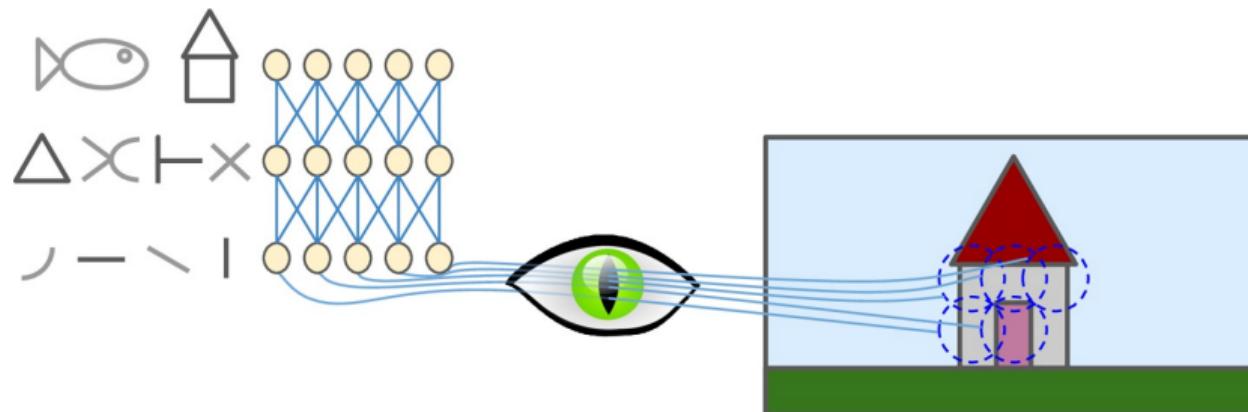
3 Tanítás

4 Konvolúciós hálózatok

A biológiai látás alapjai

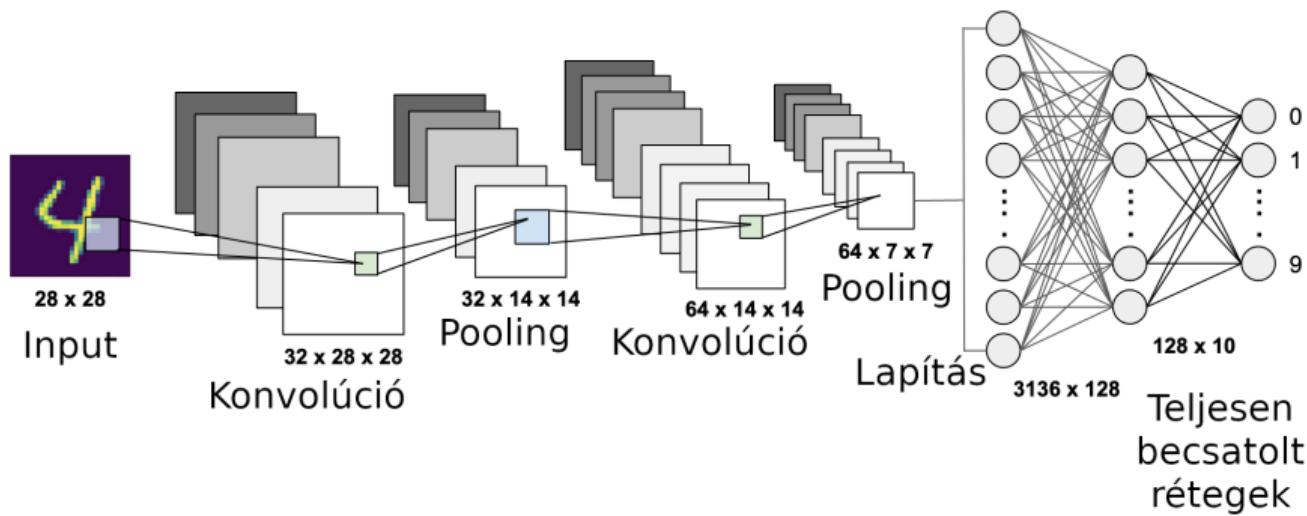
David H. Hubel és Torsten Wiesel macskákon mutatták meg, hogy **a látókéregben a neuronoknak befogadó mezője van**, azaz csak a vizuális térnek egy bizonyos szegmensén elhelyezett ingerületet képesek befogadni.

Vannak olyanok, amelyek csak vertikális, vagy horizontális alakokat érzékelnek. A **neuronok különböző befogadó mezői átfedésben állhatnak egymással**, és együtt alakítják ki a vizuális teret.



Konvolúciós hálózati architektúra

A konvolúciós hálókban a **konvolúciós** és **pooling** rétegek felelnek a képi jellemzők alacsony szintű feldolgozásáért. Ezután teljesen becsatolt rétegek következnek. Az output réteg felépítése megegyezik a korábban definiáltakkal.



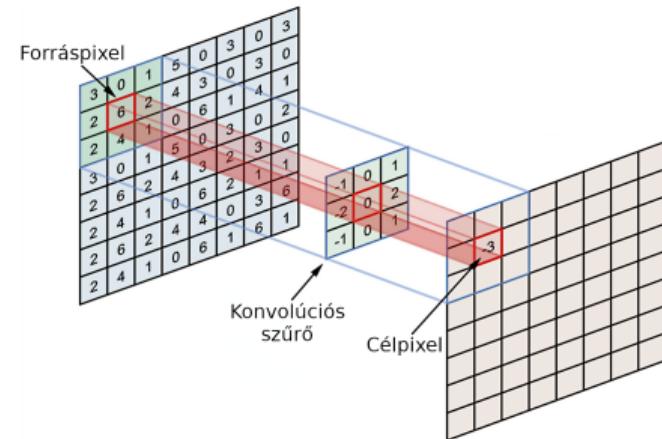
A konvolúció művelete

A konvolúció egy matematikai művelet, ami két függvényt kombinál egy harmadikká. Egy **jelfüggvényt** valamelyen **magfüggvénnyel** vegyít egy outputtá, ami megadja, hogy a **mag mennyiben befolyásolja a jelet**.

Konvolúció (2D, diszkrét)

A diszkrét konvolúció két 2D tömbön értelmezett, és jellemzők kivonatolására, szűrők alkalmazására használatos. Konvolúció f jelen és g magon, i, j képkordinátákra:

$$(f*g)[i, j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} g[i - m, j - n]f[m, n]$$

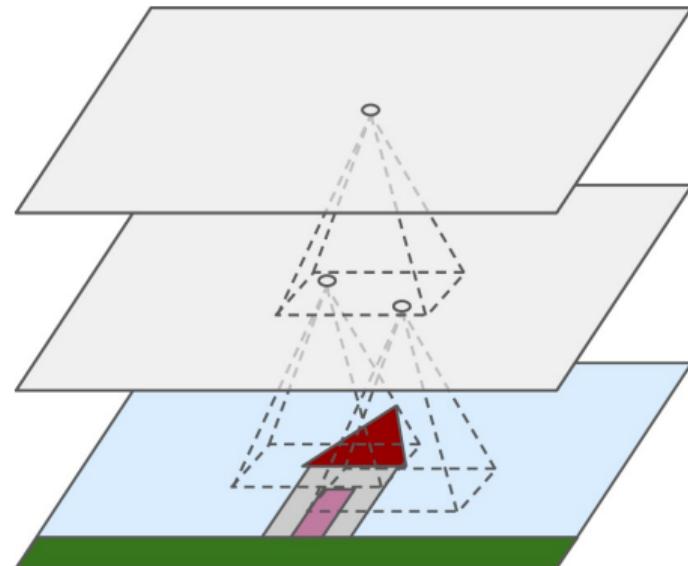


$$\begin{aligned}
 & (-1 \cdot 3) + (0 \cdot 0) + (1 \cdot 1) + \\
 & (-2 \cdot 2) + (0 \cdot 6) + (2 \cdot 2) + \\
 & (-1 \cdot 2) + (0 \cdot 4) + (1 \cdot 1) = -3
 \end{aligned}$$

Konvolúciós rétegek

A neurális hálózatok konvolúciós rétegeiben a tanítható súlyok az egyes forráspixelekhez tartozó konvolúciós szűrőkben elhelyezkedő értékek. Ez minden forráspixelhez egy egyedi szűrőt jelent, de lehetséges tetszőleges számú szűrő is.

Ez az architektúra lehetővé teszi alacsony szintű jellemzők leképezését a mélyrétegekben, és a magas szintűek leképezését a sekély rétegekben.



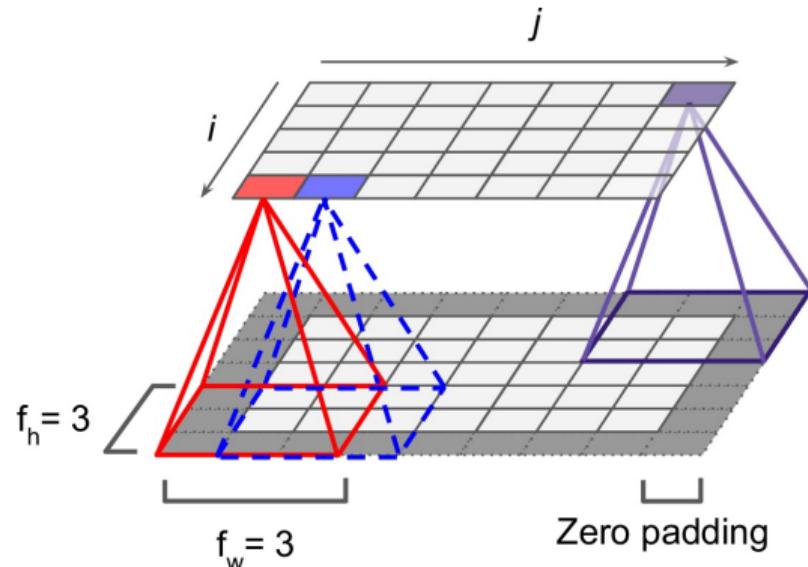
Konvolúciós neuronok kapcsolatai

Ha adott egy $f_h \cdot f_w$ szélességű konvolúciós szűrő az i, j pozíión és s lépésmérettel, ennek a befogadó mezőjének a szélessége:

$$rf_w = f_w + (i - 1) \cdot (s - 1)$$

és magassága:

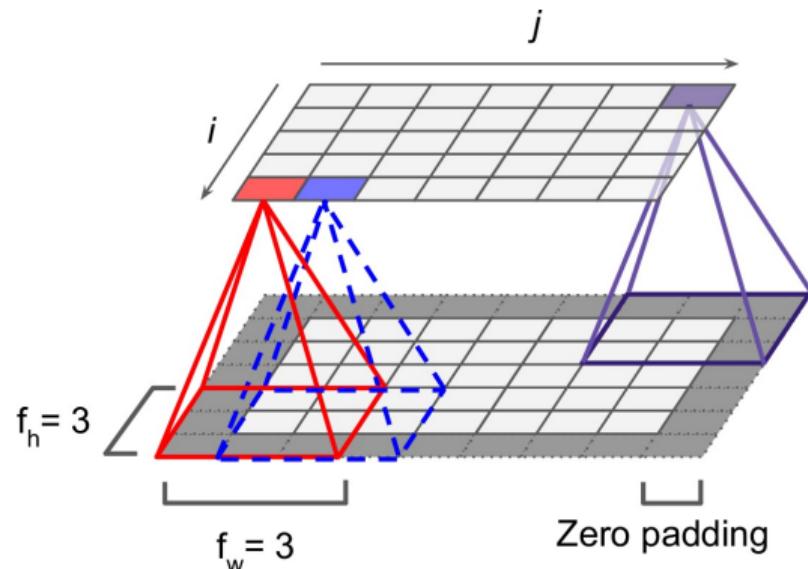
$$rf_h = f_h + (j - 1) \cdot (s - 1)$$



Konvolúciós neuronok kapcsolatai

Zero-padding

Az input kép körbevétele 0 értékekkel, hogy elkerülhető legyen a konvolúciós műveletből adódó méretcsökkenés.

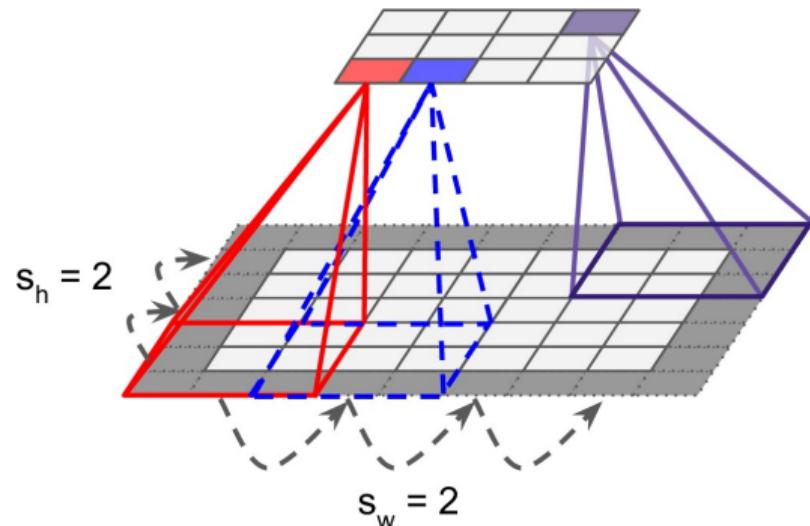


Konvolúciós neuronok kapcsolatai

Lépésméret

A konvolúciós szűrők lépésmérete az x tengely mentén s_w , az y tengely mentén s_h azt szabályozza, hogy a konvolúciós kernel (szűrő) milyen mértékben mozog át az input kép pixelein a konvolúció során.

Nagyobb lépésméret kisebb jellemzőményet eredményez.



Konvolúciós szűrők

A konvolúciós neuron súlyai
reprezentálhatók egy konvolúciós szűrővel.
A neuron ezeket a súlyokat finomhangolja a
céltól függően.

Eredeti kép



Konvolúciós szűrők

Az ábrákon egy vertikális és horizontális szűrővel konvolvált képek láthatók. A szűrők egy tengelyen 1 értéket vesznek fel, mindenhol másol 0 értékeket.

Vertikális szűrő

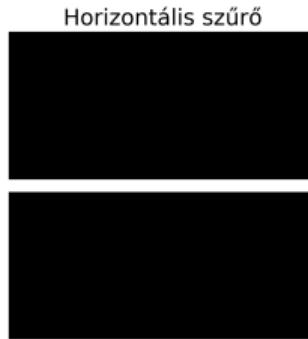


Vertikálisan szűrt kép



Konvolúciós szűrők

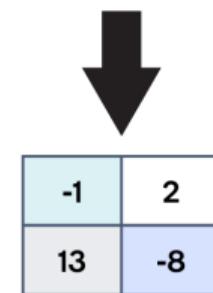
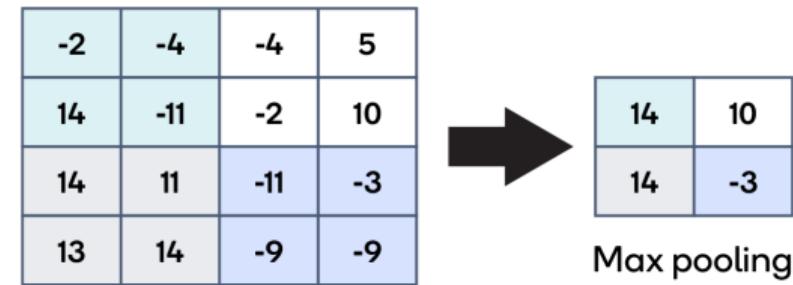
Az ábrákon egy vertikális és horizontális szűrővel konvolvált képek láthatók. A szűrők egy tengelyen 1 értéket vesznek fel, mindenhol másol 0 értékeket.



A lazító (pooling) réteg

A lazító rétegek feladata, hogy almintázzák a bemeneti jellemzőket annak érdekében, hogy kisebb legyen a memóriaigény és a paraméterek száma.

A lazító neuron aggregálja a kapcsolatai által elérhető neuronok értékeit. Az aggregáló művelet lehet például *max*, *avg* stb...



A lazító (pooling) réteg

A lazító rétegek feladata, hogy almintázzák a bemeneti jellemzőket annak érdekében, hogy kisebb legyen a memóriaigény és a paraméterek száma.

A lazító neuron aggregálja a kapcsolatai által elérhető neuronok értékeit. Az aggregáló művelet lehet például *max*, *avg* stb...

Eredeti kép



A lazító (pooling) réteg

A lazító rétegek feladata, hogy almintázzák a bemeneti jellemzőket annak érdekében, hogy kisebb legyen a memóriaigény és a paraméterek száma.

A lazító neuron aggregálja a kapcsolatai által elérhető neuronok értékeit. Az aggregáló művelet lehet például *max*, *avg* stb...

3x3 lazítás



A lazító (pooling) réteg

A lazító rétegek feladata, hogy almintázzák a bemeneti jellemzőket annak érdekében, hogy kisebb legyen a memóriaigény és a paraméterek száma.

A lazító neuron aggregálja a kapcsolatai által elérhető neuronok értékeit. Az aggregáló művelet lehet például *max*, *avg* stb...

4x4 lazítás



A lazító (pooling) réteg

A lazító rétegek feladata, hogy almintázzák a bemeneti jellemzőket annak érdekében, hogy kisebb legyen a memóriaigény és a paraméterek száma.

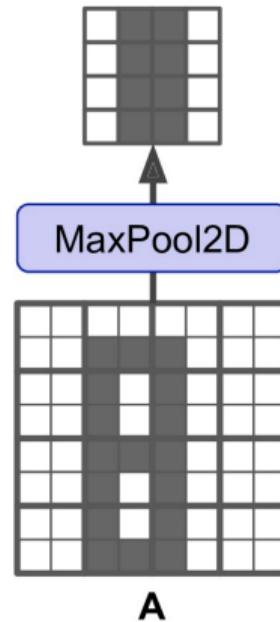
A lazító neuron aggregálja a kapcsolatai által elérhető neuronok értékeit. Az aggregáló művelet lehet például *max*, *avg* stb...



Lazítás a gyakorlatban

A memória- és paraméterszám csökkentése mellett a lazítás művelete bevezet egy alacsony fokú invarianciát kisebb eltolásokkal szemben.

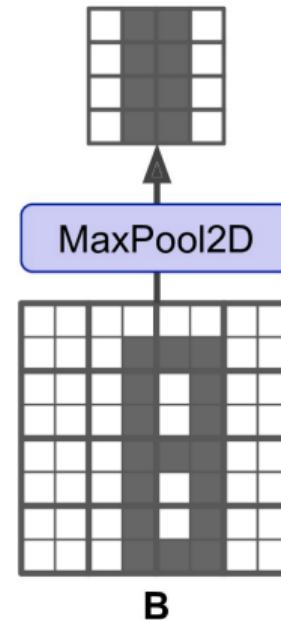
Érdemes megfigyelni, hogy az A és B képek esetén a pooling ugyanazt az outputot eredményezte, viszont ahogy az input kép jobban eltolódik, az output is megváltozik.



Lazítás a gyakorlatban

A memória- és paraméterszám csökkentése mellett a lazítás művelete bevezet egy alacsony fokú invarianciát kisebb eltolásokkal szemben.

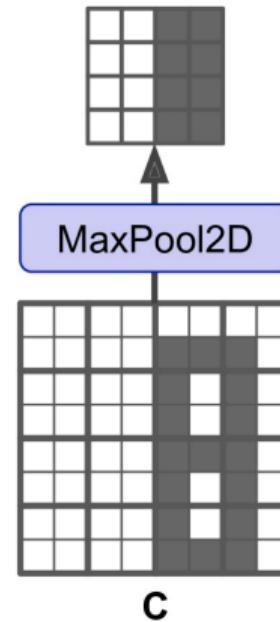
Érdemes megfigyelni, hogy az A és B képek esetén a pooling ugyanazt az outputot eredményezte, viszont ahogy az input kép jobban eltolódik, az output is megváltozik.



Lazítás a gyakorlatban

A memória- és paraméterszám csökkentése mellett a lazítás művelete bevezet egy alacsony fokú invarianciát kisebb eltolásokkal szemben.

Érdemes megfigyelni, hogy az A és B képek esetén a pooling ugyanazt az outputot eredményezte, viszont ahogy az input kép jobban eltolódik, az output is megváltozik.



Adataugmentáció

Új tanító egyedek létrehozása a meglévőkön végzett változtatások segítségével mint:

- Forgatás
- Eltolás
- Sötétítés
- Közelítés
- Távolítás
- Világosítás

