DeepLearning alapjai

Elméleti bevezetés

2019. július 12.

Novák Márton Clementine

Clementine

Tartalomjegyzék

Miről lesz szó?

- DeepLearning, ML alapok bemutatása
- Elmélet + gyakorlati példák
- Google Colab, Keras, Python, GitHub
- Példákon keresztül az alapok elsajátítása

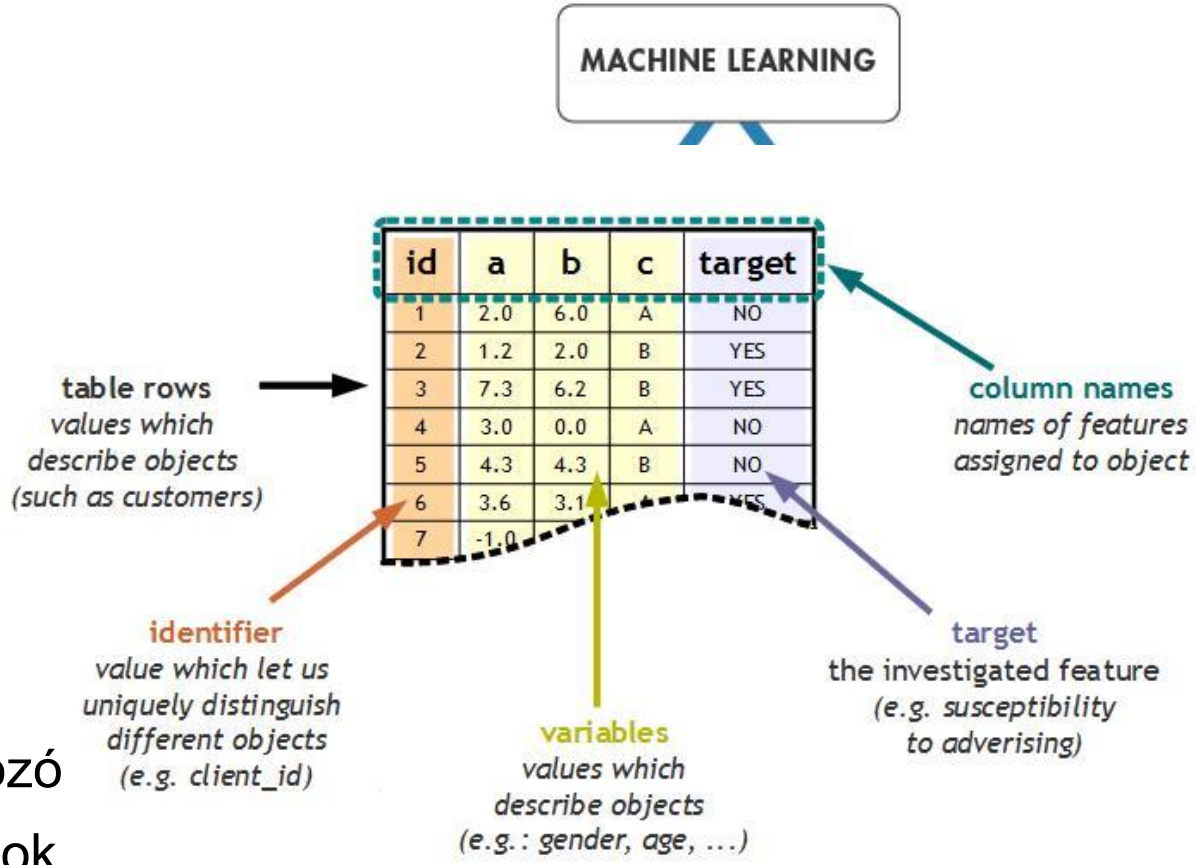
Ezekről pedig nem:

- Mély matematikai háttér – 2x2 óra ehhez kevés, nem ez a cél
- Python, ML, Data Science ismeretek mély elsajátítása

Mi is az a DeepLearning

Alap ML problémák

- Osztályozás
- Regresszió
- Klaszterezés



Felügyelt tanulás: tulajdonságvektor + célváltozó

Felügyelet nélküli tanulás: csak a tulajdonságok, a modell kimenete, a kívánt eredmények nem ismertek

(+ megerősítéses tanulás)

https://www.mathworks.com/help/stats/machine-learning-in-matlab.html
http://algolytics.com/predictive-analytics-for-beginners-part-1/

Mi is az a DeepLearning

DL vs alap ML megoldások

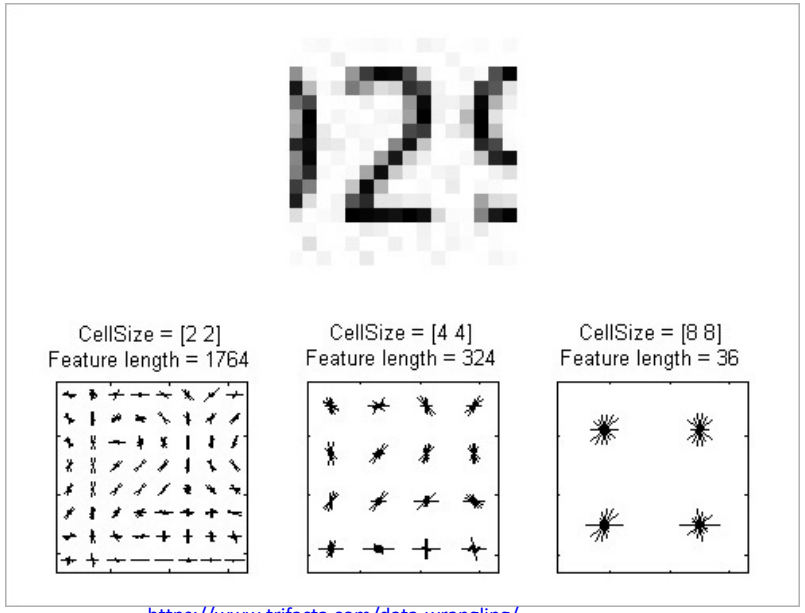
Adathalmaz tisztítása, a megfelelő tulajdonságok megtalálása, kialakítása fontos, időigényes, szakterület specifikus tudást kíván (Data wrangling).

Ezt a folyamatot kéne valahogy lerövidíteni.

Megoldás: DeepLearning!

Architektúrából adódóan megtanulja a hasznos feature-ket.

Cserébe: rengeteg adat + erős hardware szükséges hozzá.



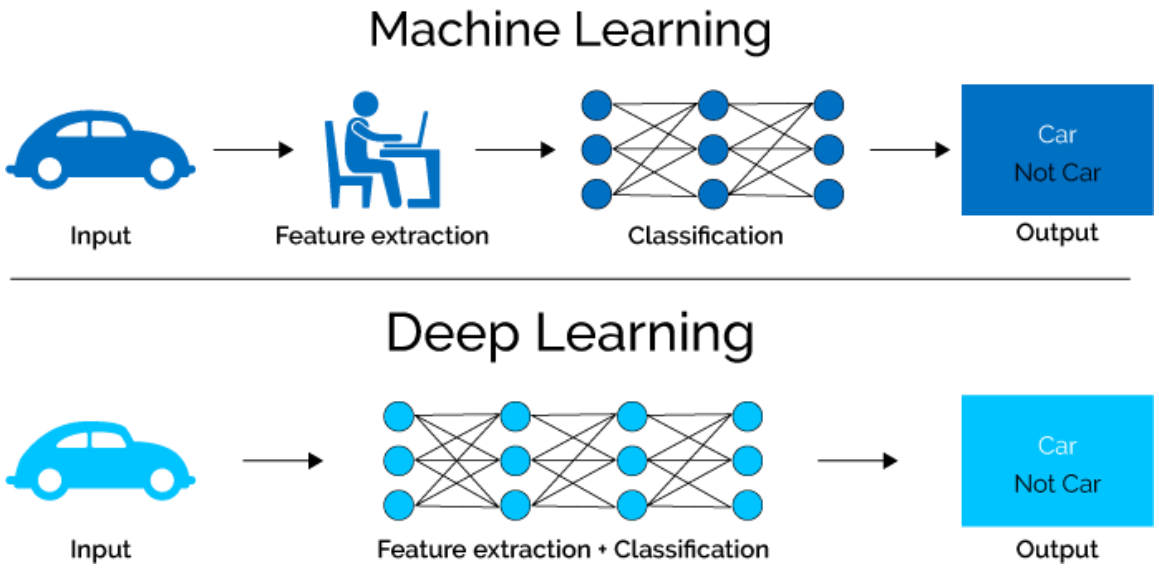
https://www.trifacta.com/data-wrangling/
https://www.mathworks.com/discovery/feature-extraction.html

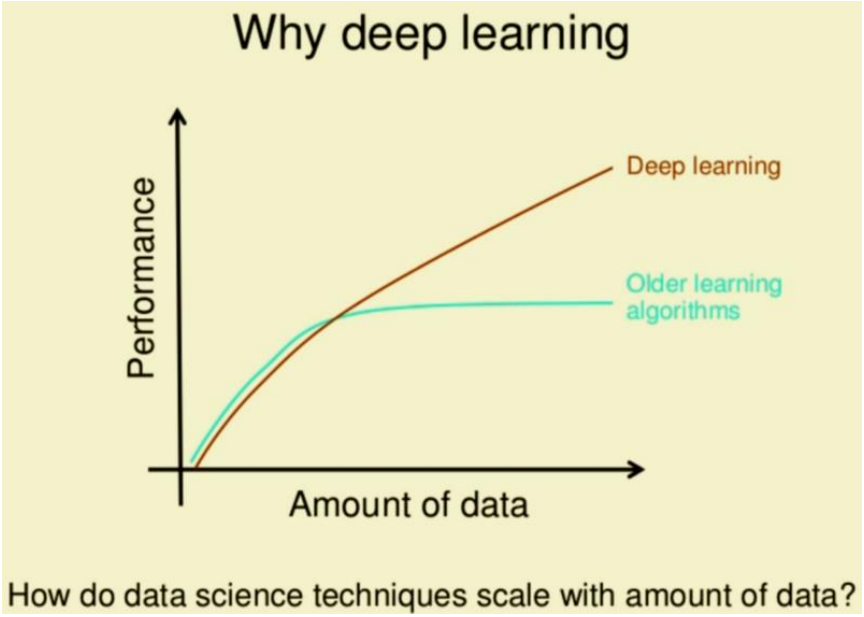
Mi is az a DeepLearning

DL vs alap ML megoldások

Adatmennyiség növelésével a DL megoldások sokkal jobban teljesíthetnek.

A bemenet és kimenet között egy egységes réteget képeznek <-> különböző technikák rétegeinek halmaza.





https://towardsdatascience.com/why-deep-le arning-is-needed-over-traditional-machine-learning-1b6a99177063

Mi is az a DeepLearning

Hol használják?

- Képfeldolgozás
- NLP
- Bioinformatika
- Csalás detektálás
- Mérnöki feladatok
- „Művészet”

https://www.networkworld.com/article/2974718/deep-dream-artificial-intelligence-meets-hallucinations.html

NN VS DNN

Neurális Háló

Alapja: agyi neurális hálók.

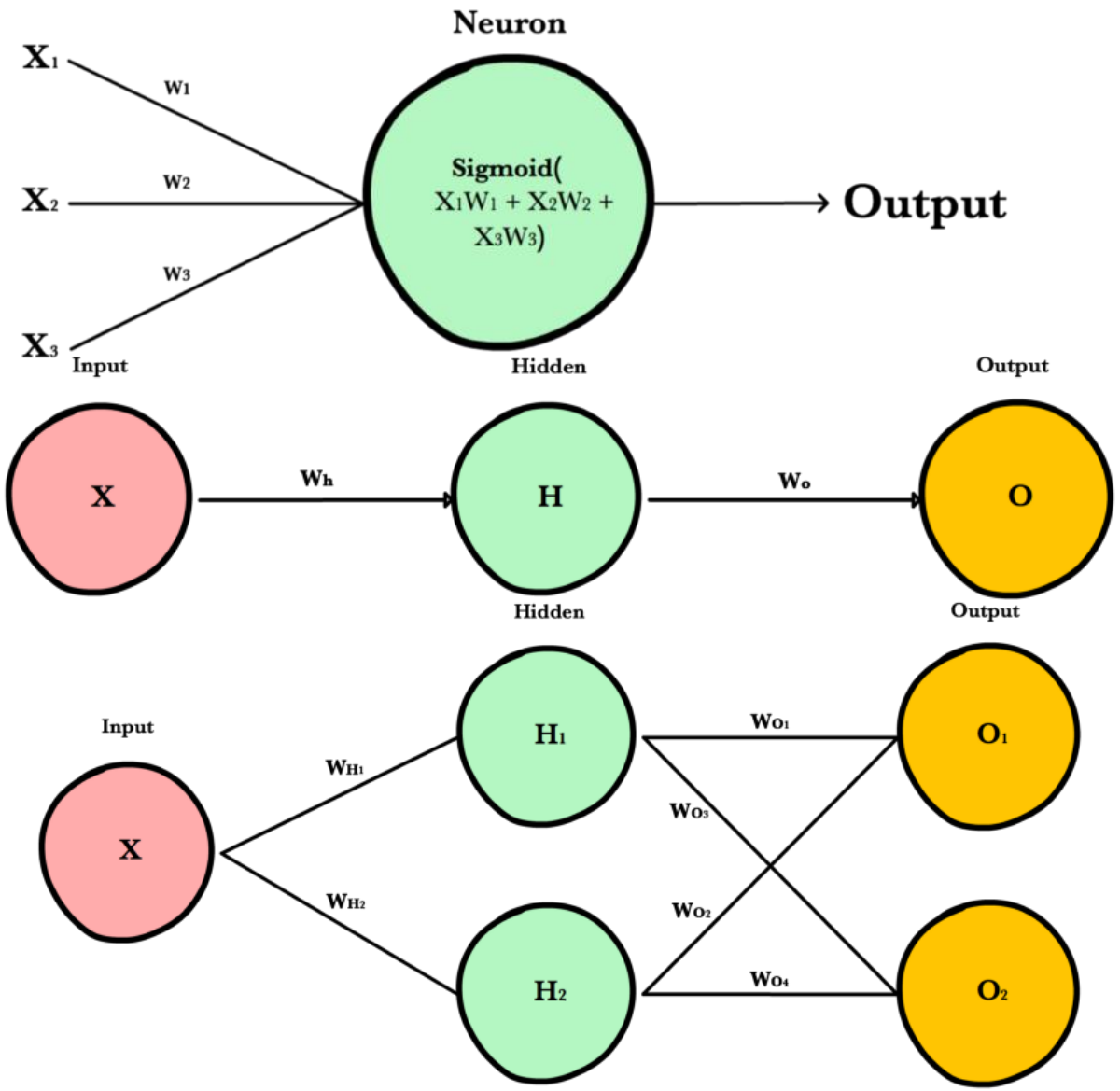
Nemlineáris aktivációs függvénnyel rendelkező entitások struktúrája, mellyel az adatokban rejlő fontos részleteket nyerjük ki.

Gráf struktúra: élek ~ összeköttetések + súlyok

csúcsok ~ neuronok

Felépítés: bemeneti réteg – rejtett réteg(ek) – kimeneti réteg.

Mély neurális háló – DeepLearning architektúra: sok-sok rejtett réteg -> magas absztrakció.



https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html

Strukturális elemek, alap fogalmak

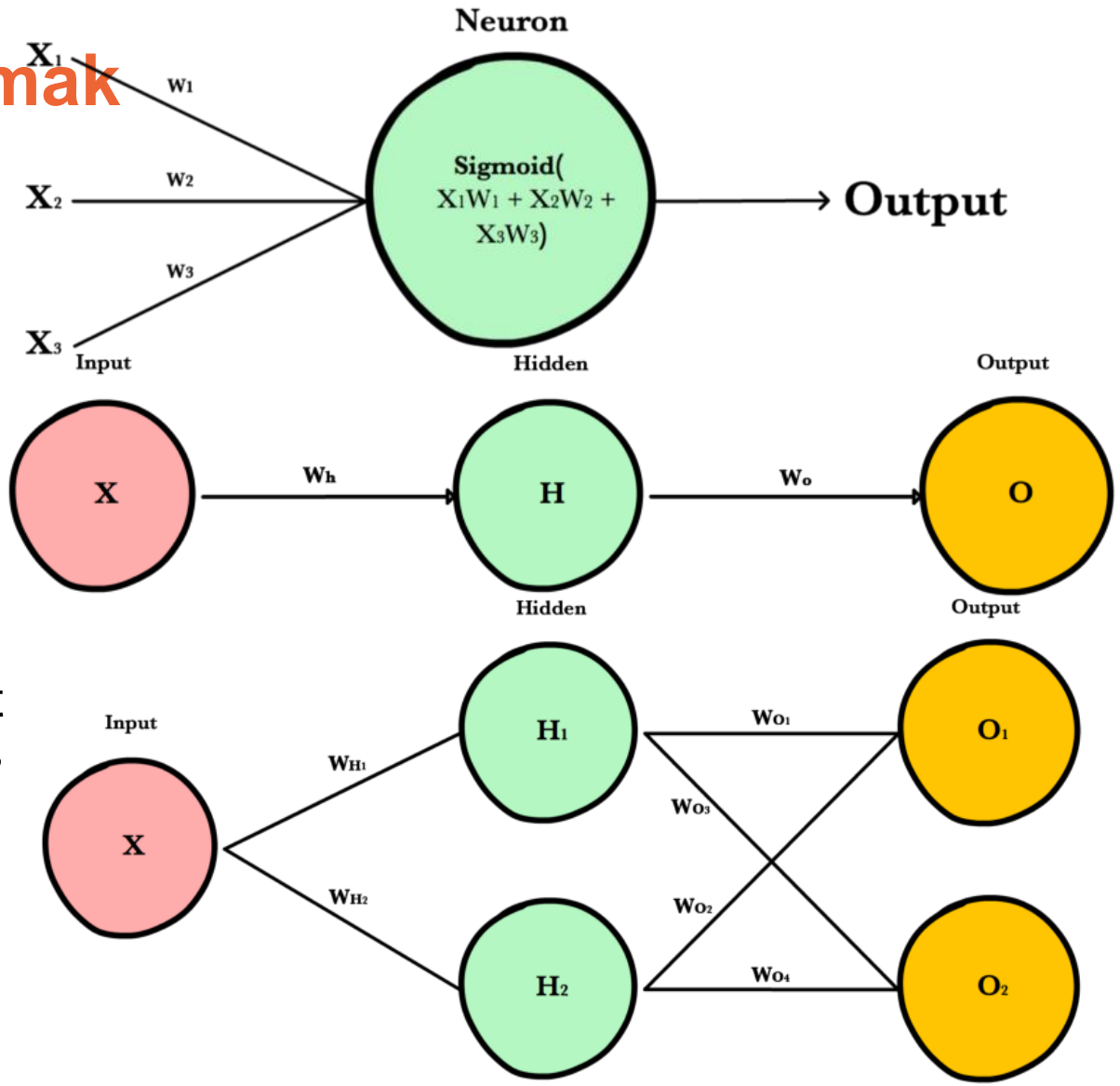
Alap, mély neurális háló architektúrája:

Elemi entitások ~ **neuronok**.

Azonos szinten lévő, együtt tüzelő neuronok ~ **rétegek**.

Egymást követő rétegek neuronjai **súlyozottan** össze vannak kötve.

Egy neuron, a hozzá befutó súlyozott értékeket **összegzi**, majd pedig egy **nemlineáris aktivációs függvényt** alkalmaz.



https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html

Strukturális elemek, alap fogalmak

Alap, mély neurális háló architektúrája:

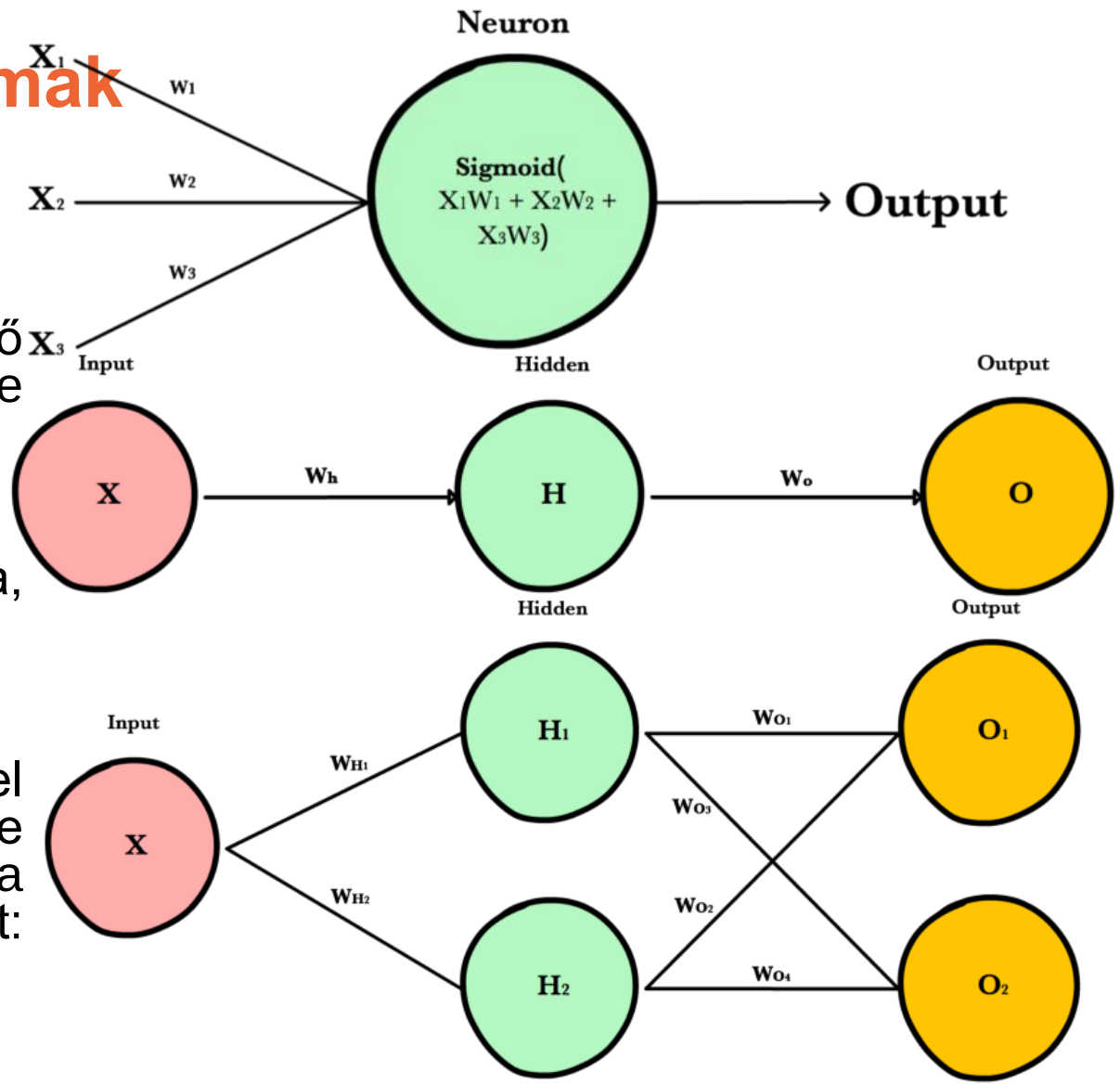
Bemeneti réteg: bejövő adatot a megfelelő struktúrába kell rendezni, hogy a hálózatba be lehessen tölteni.

Rejtett rétegek: egymással összekötött rétegek.

Kimeneti réteg: a feladatnak megfelelő struktúra, valamint aktivációs függvény szükséges.

Tanulás:

Ha a célváltozó ismert: a becsült eredményekkel hibát számolunk. A struktúrában hátulról előre visszaterjesztjük azt, hogy az adott rétegben a neuronok mennyire felelősek az eltérésért: backpropagation.



https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html

Strukturális elemek, alap fogalmak

Vektorizálás, tenzorok, deriválás

Belső réteg egy neuronja: előző rétegekből érkező értékek súlyozott szummája (+ bias), majd erre aktivációs függvény hattatása.
A súlyokat mátrixba rendezve egy adott réteg kimenete vektorműveletekkel is számítható.

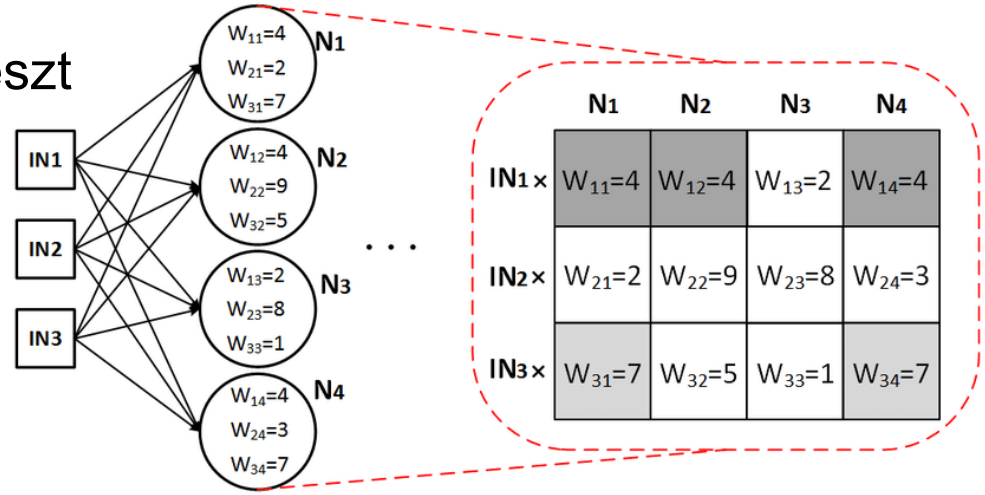
Egy rejtett réteg j. neuronjának kimenete így egyrészt leírható a

$$h_j = f(b_j + \sum_{i=1}^{N} W_{ij} X_i)$$

másrészt pedig a

$$h = f(Wx + b)$$

jelölésekkel is.



https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05

Strukturális elemek, alap fogalmak

Vektorizálás, **tenzorok,** deriválás

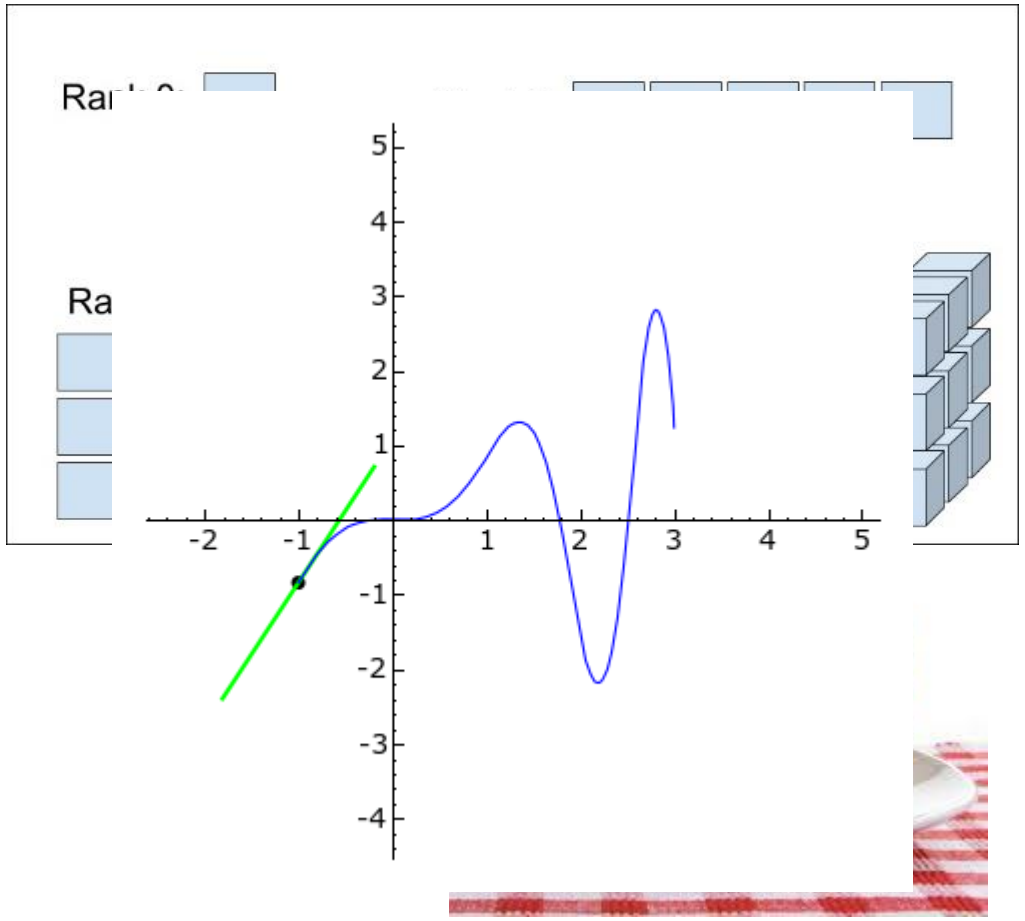
Dimenziók növelése listába fűzéssel:
Skalár -> vektor -> mátrix -> tenzor

Mátrixok listája, olyan, mintha palacsinták rétegei lennének.

Főleg a konvolúciós neurális hálóknál lesz érdekes (később…)

Vektorizálás, tenzorok, **deriválás**

Érintő meredeksége / a függvény változásának mértéke az adott pontban.



https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781787125933/14/ch14lvl1sec85/tensorflow-ranks-and-tensors
https://patriotgetaways.com/blog/gatlinburg-the-pancake-capital-of-the-south/
https://commons.wikimedia.org/wiki/File:Graph_of_sliding_derivative_line.gif

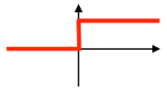
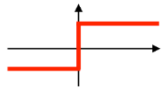
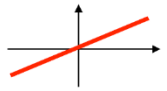
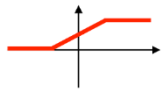
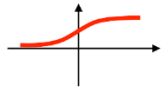
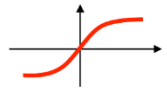
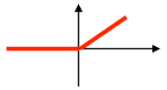
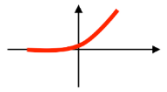
Strukturális elemek, alap fogalmak

Aktivációs függvények

Fontos a nemlinearitás!

A mély neurális hálók (**D**eep **N**eural **N**etworks ~ **DNN**) így tudnak jó eredményeket produkálni. Lineáris aktivációs függvények esetén a rétegeket szummázni lehetne egy lineárisan leképző rétegbe, a kimenet a bemenet egy lineáris kombinációjaként állna elő…

Legtöbbet használt aktivációs függvények: Sigmoid, ReLU (+ ennek különböző változatai), tanh, softplus

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016 (http://sebastianraschka.com)

http://rasbt.github.io/mlxtend/user_guide/general_concepts/activation-functions/

Strukturális elemek, alap fogalmak

Visszamérés

Ha megkaptuk a jósolt eredményeket, azt összevethetjük a tényleges címkékkel.

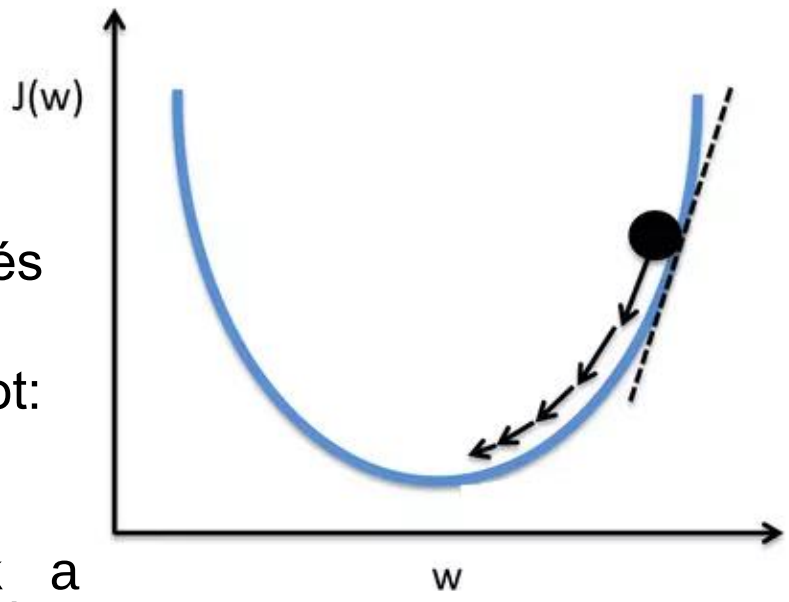
A tanítás során a súlymátrix elemeit állítjuk <- ezek a paraméterek.

Különböző paraméterek -> más és más a valóságtól való eltérés mértéke

A hibafüggvény által kapott eltérések terén keresünk optimumot: hol a legkisebb a hiba?
Deriválással! Fontos, hogy folytonos és deriválható legyen.

Ahogy módosítjuk a súlyokat, egyre közelebb kerülünk a minimumhoz. A súlyok módosításának mértéke az ún. learning rate, azaz a tanulási ráta.

Legnépszerűbb hibafüggvények: Bináris keresztentrópia, többosztályos keresztentrópia, átlagos négyzetes eltérés.



https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c

Strukturális elemek, alap fogalmak

Metrikák

Ezzel értékelhetjük a modellünk teljesítményét. Hasonló a hibafüggvényhez, viszont ezt nem használjuk a tanítás során.

Regresszió esetén: MAE, MSE, RMSE

Osztályozás: tévesztési mátrix (confusion matrix) -> accuracy, precision, recall, f1 score

Klaszterezés: ?

CONFUSUSION MATRIX	ACTUAL	
PREDICTED	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

$$\text{F1-Score} = \frac{2*Precision*Recall}{Precision+Recall}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

https://cubalytictalks.blogspot.com/2018/08/confusion-matrix.html

Néhány szó az adatokról

Mi csak mintavételezünk… Az adathalmaz a valóság egy leképzése. Ha a modell ezen jól fut, az még nem garantálja, hogy valós környezetben mindig jól fog szerepelni. Pl.: periodikus/folytonos jelek mintavételezése.

DNN ~ szürke doboz -> nehéz megtalálni a hiba okát.

Rosszak az adatok? Más struktúrát kéne használni? Hiperparaméterek nem lettek jól optimalizálva?

HPO: adatok feldolgozási része + modell különböző paraméterei. Aktivációs függvény, learning rate, rejtett rétegek száma, neuronok száma, **batch** (egyszerre ennyi mintát adunk be, mielőtt egy backprop-os súlymódosítási lépés megtörténik), **epoch** (ennyiszer töltjük be a teljes adathalmazt)

Data wrangling-nek csak a feature extraction részét spóroljuk meg. Adatokat érdemes vizualizálni, tisztítani.

Train-test-validation: **train** <- tanító adathalmaz, **validation** <- túltanulást ezen vizsgáljuk, **test** <- a modellünk teljesítményét itt mérjük vissza

Konvolúciós neurális hálók

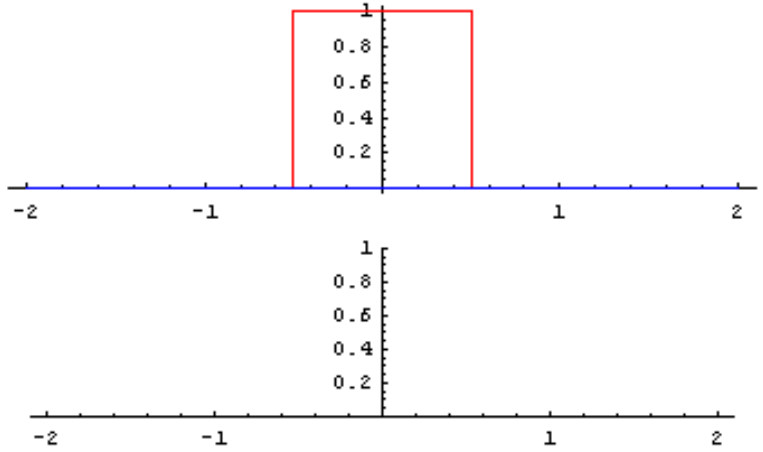
Képi feldolgozáshoz használt DNN architektúra. Bemenete: képek több dimenzióban (rgb channel)

Részei: Konvolúciós + pooling + fully connected réteg

Konvolúció: megnézi, hogy a két bemenő paraméterét folyamatosan elcsúsztatva mi lesz a súlyozott átlag.

Pooling: mintha tömörítenénk, egy mátrix adott részét annak maximumával/átlagával helyettesítjük

Fully connected: az előző rétegek kinyerték a megfelelő feature-ket, így most már ezeket inputként használva végrehajthatunk pl. egy osztályozási feladatot.



https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html

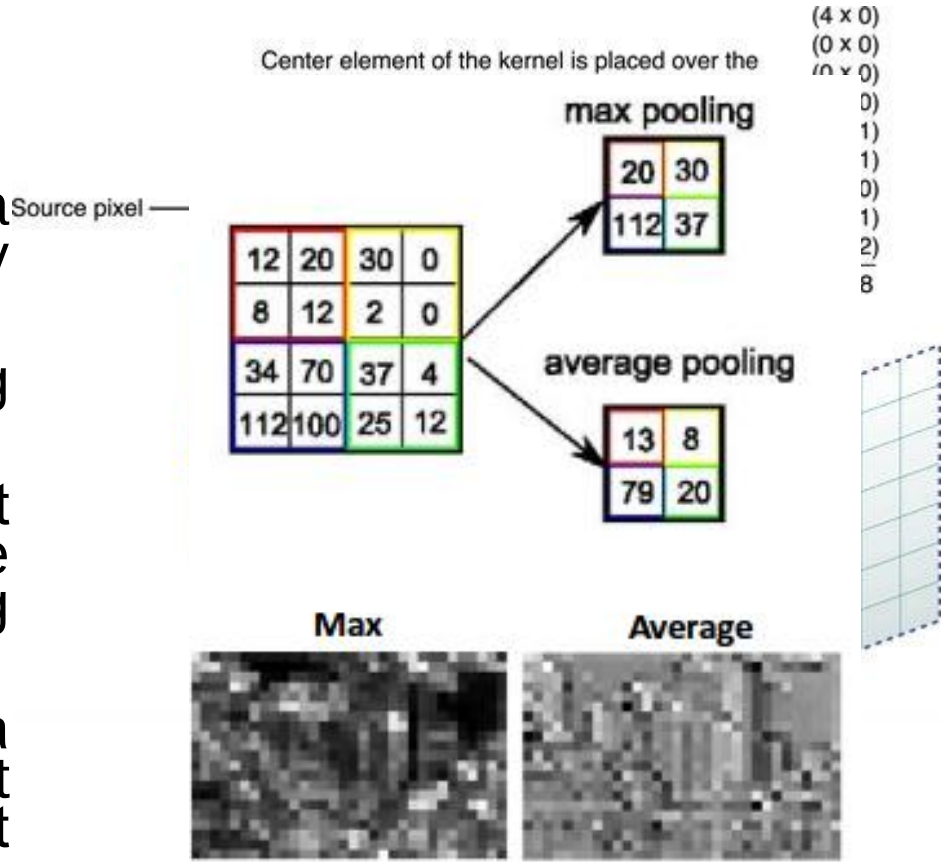
Konvolúciós neurális hálók

A konvolúciós lépés során az adott réteg mátrixán a konvolúciós réteg kerneleit futtatjuk végig. Így kapunk egy aktivációs térképet.

Ezen, hogy a méret csökkenjen, max/avg pooling réteggel tömörítést hajtunk végre.

A pooling réteg után érdemes az aktivációs réteget tenni (a már ismertetett activation function van ide külön szedve), mivel az így kevesebb kimeneten fog dolgozni.

A konvolúciós és pooling rétegeket egymásra stackelve eljutunk egy megfelelően absztrakt struktúráig, ami képes a kulcsfontosságú részleteket kinyerni a képből -> ezt egy fully-connected hálóba kapcsoljuk.

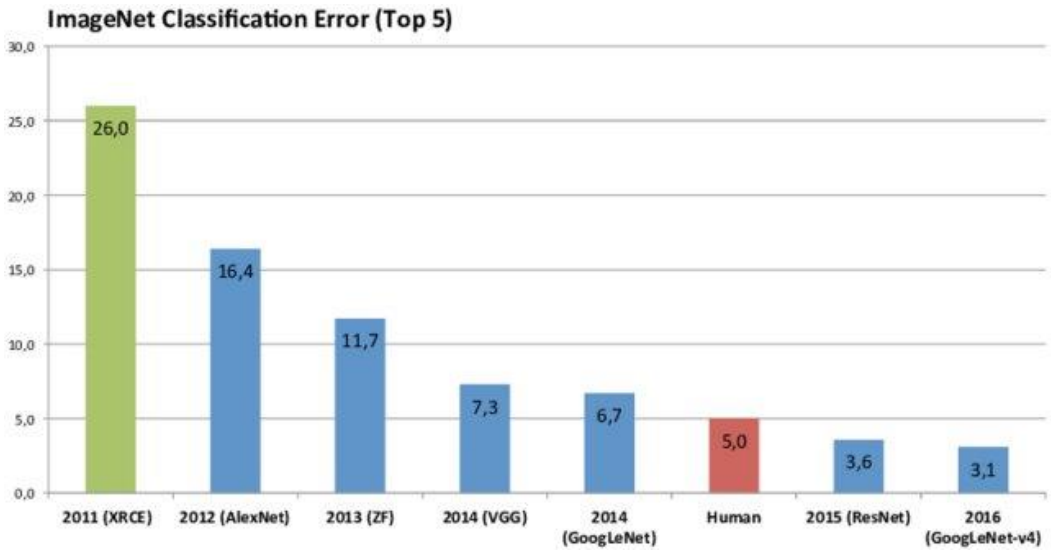


https://stats.stackexchange.com/questions/235032/any-use-of-non-rectangular-shaped-kernels-in-convolutional-neural-networks-espe
https://www.quora.com/What-is-the-benefit-of-using-average-pooling-rather-than-max-pooling

Konvolúciós neurális hálók

Lépések:

- A feldolgozni kívánt képet a megfelelő alakba kell hozni
- Fel kell építeni a megfelelő konvolúciós-pooling réteg struktúrát (+aktivációs rétegek)
- A layer stackre még egy teljesen összekötött hálót teszünk

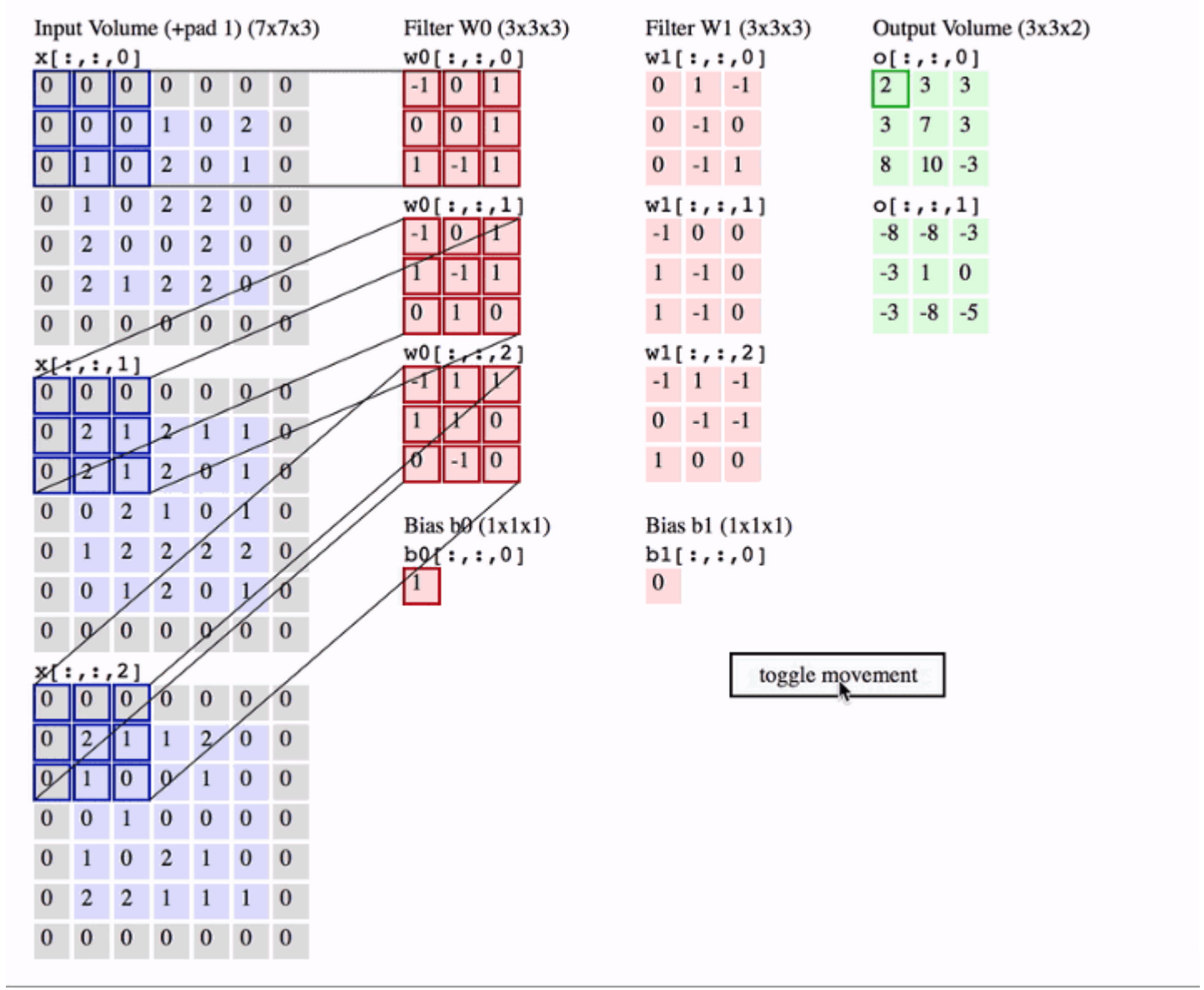


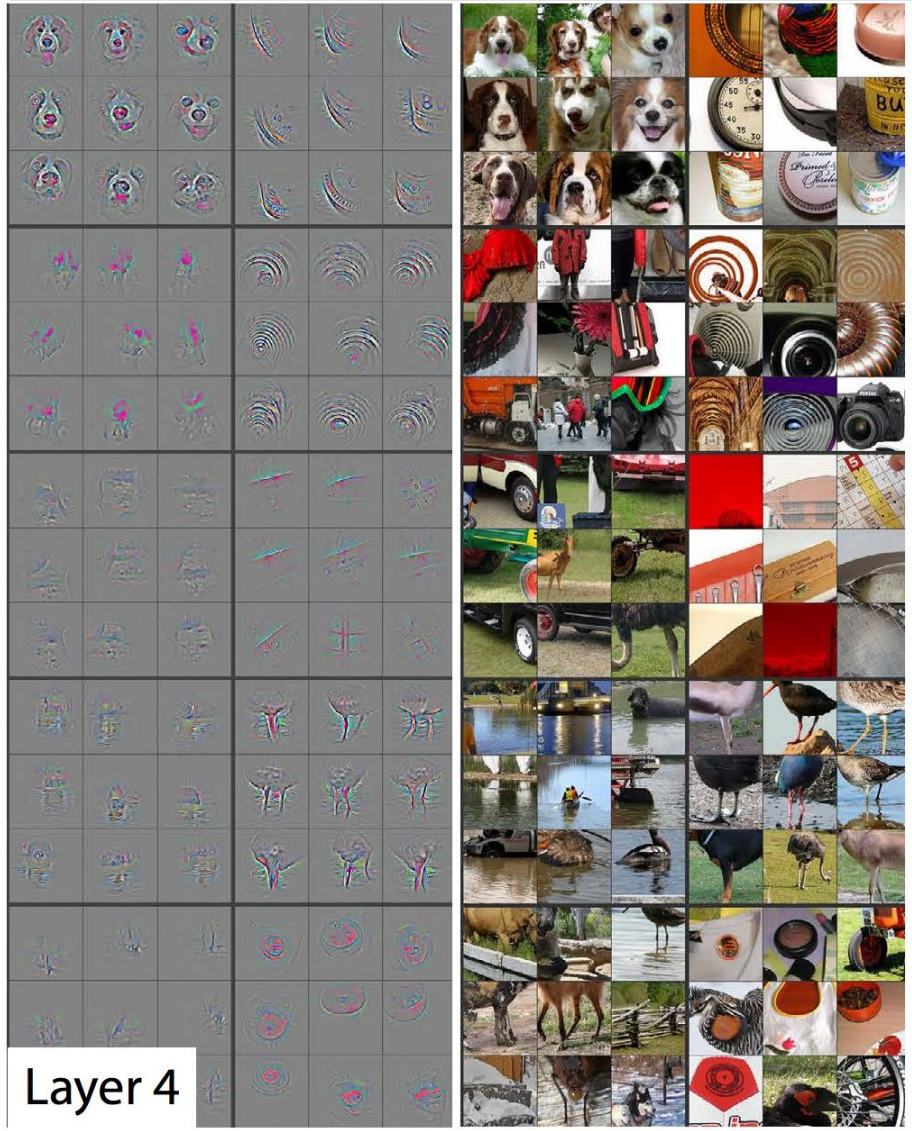
CNN vs FC DNN:

Sokkal kevesebb paramétert kell tanítani, a legtöbb még így is a végső FC részben van. **Viszont:** a konvolúciós műveletek miatt lassabb!

Híres struktúrák: **LeNet, LeNet5, AlexNet, ZFNNet**. Imagenet adathalmazon jobb eredményt érnek el, mint az emberek…

https://www.researchgate.net/publication/324476862_Survey_of_neural_networks_in_autonomous_driving/figures?lo=1

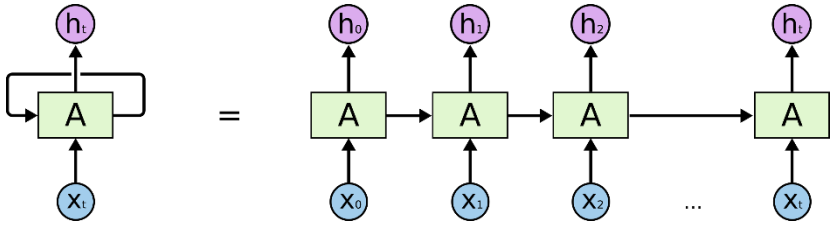


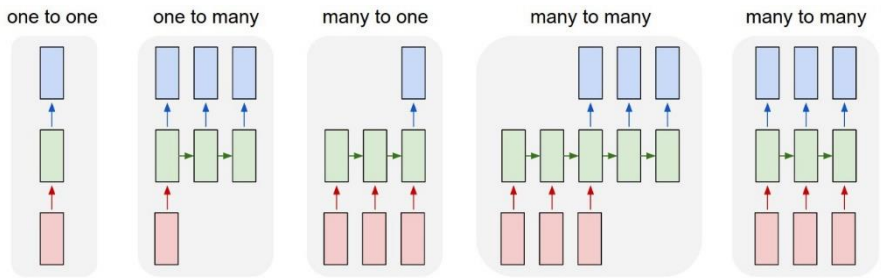


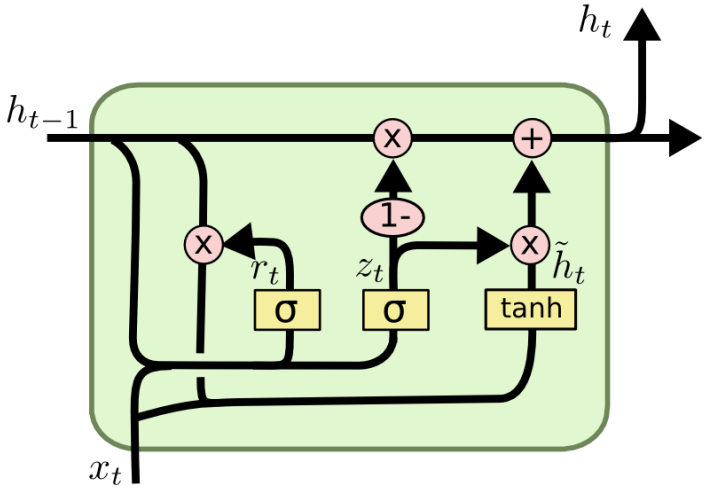
https://medium.com/@phidaouss/convolutional-neural-networks-cnn-or-convnets-d7c688b0a207
https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/

(Mély) Neurális háló struktúrák

Convolutional Neural Network: képfelismerés jellemzőkinyeréssel
Recurrent Neural Network: szekvenciális bemeneten, különböző problémákat old meg (one-many, many-many…), viszont grádiensel kapcsolatos problémái vannak. Rejtett belső állapot átcsatolás + stackelt struktúra
Simple Perceptron: egyszerű struktúra, egy neuron működési modellje
Long-Short Term Memory (LSTM): RNN problémáit oldja meg, hosszú emlékezettel rendelkezik, különböző belső kapukkal működik
Gated Recurrent Units: LSTM módosítva, gyorsabb és könnyebb használni, kevésbé ad kifejező eredményeket







https://colah.github.io/posts/2015-08-Understanding-LSTMs/
https://discuss.pytorch.org/t/example-of-many-to-one-lstm/1728

Egyéb technikák

„HPO: adatok feldolgozási része + modell különböző paraméterei. Aktivációs függvény, learning rate, rejtett rétegek száma, neuronok száma, ***batch*** *(egyszerre ennyi mintát adunk be, mielőtt egy backprop lépés megtörténik),* ***epoch*** *(ennyiszer töltjük be a teljes adathalmazt)”*

+Dropout: tanítás során véletlenszerűen kikapcsolunk néhány neuront -> jobban fog általánosítani.

+Súly inicializálás: valamilyen eloszlásból választunk véletlen értékeket, általában 1 szórással, 0 várható értékkel. He, Glorot, Lecun.

+Optimalizálás: sok optimalizálót használhatunk, amelyek a loss függvényen különböző módon keresnek minimumot. Adagrad, Adam, RMSprop…

Data wrangling-nek csak a feature extraction részét spóroljuk meg. Adatokat érdemes vizualizálni.

Batch normalizálás: adatokat érdemes normalizálni, mielőtt inputra tesszük őket. Akkor a súlyokkal miért nem tesszük ugyan ezt?

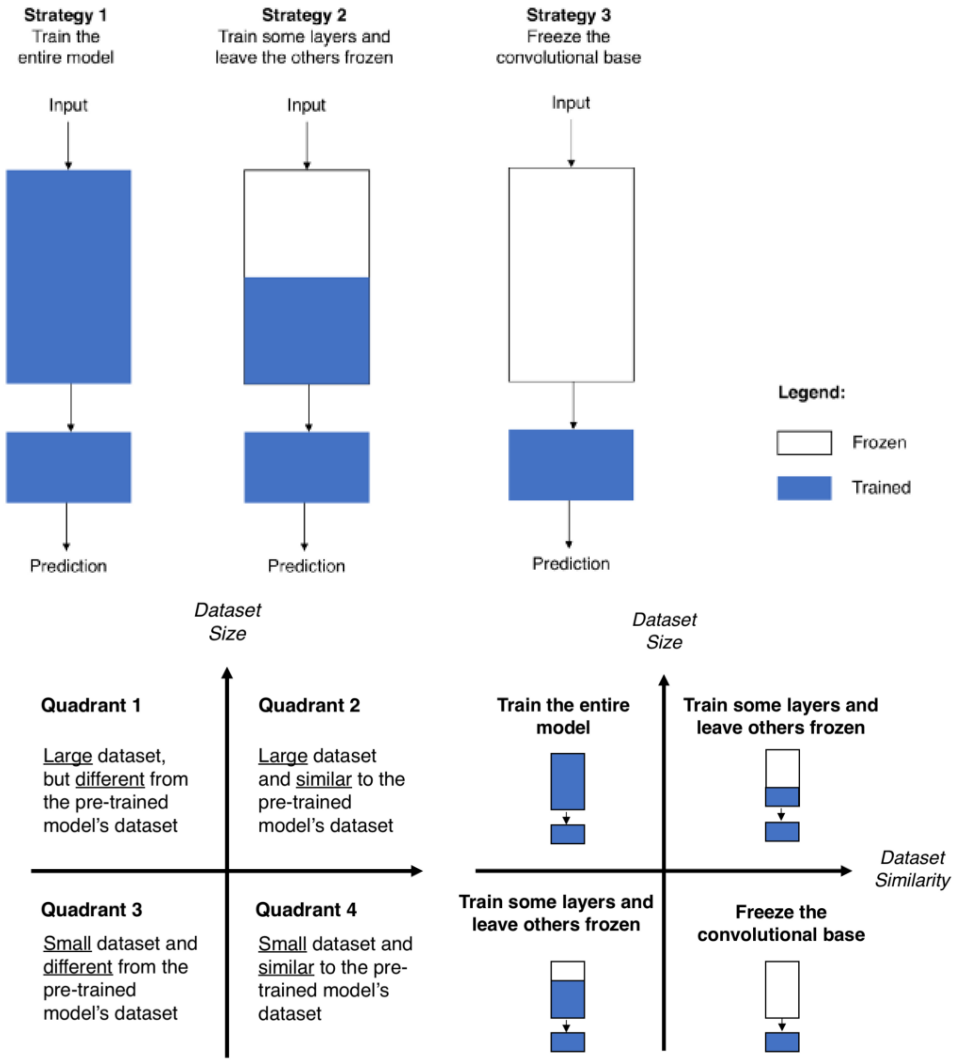
Egyéb technikák

Early stopping: ha túltanítjuk a modellünket, akkor az nem tud jól általánosítani. Ha egy bizonyos érték alatt van a javulás a validációs adathalmazon, akkor álljunk le a tanítással.

A tanítás során az adott epoch végén ki lehet menteni a modellünket. Sőt, különböző adathalmazokra már léteznek előre betanított hálózatok, azok súlymátrixát elérhetjük, így némi módosítással más problémákra is jól lehet ezeket használni.

A különböző rétegek súlyát be tudjuk fagyasztani, így pl. elég csak a legutolsó réteget újratanítani. Nem szükséges a modellünket „from scratch” felépíteni.

Transfer learning



https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

Hasznos források

Sok opensource megoldás: GitHub, arXiv, arXiv Sanity

Könyvek:

François Chollet – *Deep Learning with Python*
Aurélien Géron – *Hands-On Machine Learning with Scikit-Learn and TensorFlow*
Adam Gibson and Josh Patterson - *Deep Learning: A Practitioners Approach*
Sebastian Raschka – *Python Machine Learning*

Online kurzusok:

Coursera – Machine Learning (Andrew Ng): matekosabb, kötelező leckék, ingyenes
Udemy – Zero to deep learning: fizetős (~10 euró, szinte mindig le van értékelve), gyakorlatiasabb, programozási és ML alapismeretek ajánlottak

Keretrendszerek leírásai, dokumentációk!

Fast.ai & medium.

https://www.notion.so/The-Best-Artificial-Intelligence-Machine-Learning-and-Data-Science-Resources-b3b97fa097b747698e87fd3badc657cf

Google ☺

https://cubalytictalks.blogspot.com/2018/08/confusion-matrix.html

Eredmények értékelése

Jól definiált struktúra, de az eredményeket nehéz megmagyarázni.

Döntési fánál jól láthatók a vágások, vagy regressziónál a magyarázó változók koefficiensei.

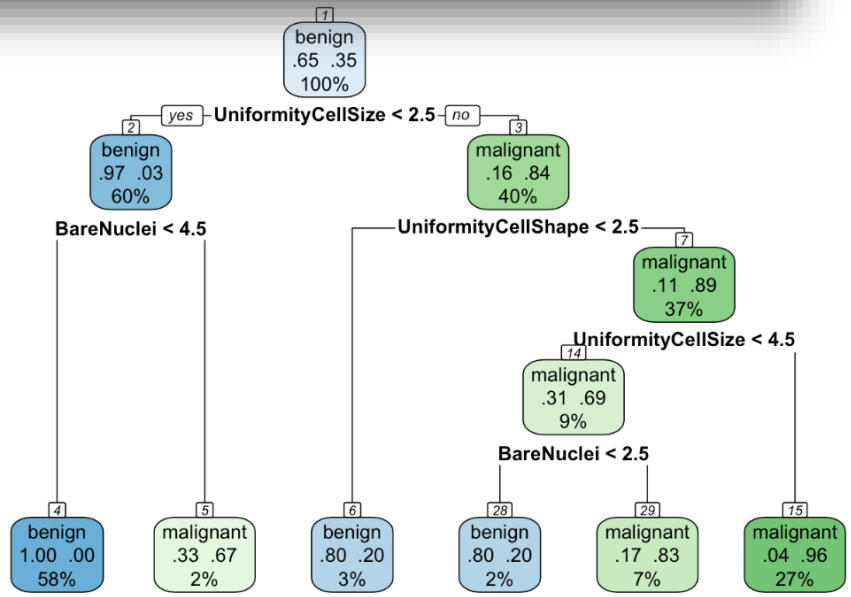
Felelősség kérdése valamint a döntések megmagyarázhatósága fontos:

- Autonóm járművek balesete
- Betegek félrediagnosztizálása
- Hiteligénylés során hátrányos megkülönböztetés

„A problémákat az ember oldja meg. A gép működik.

Felhasználóként nem érdekel hogy működik. Értsen a gép minket, és ne fordítva.

Nincs gépi etika. A felelősség is az emberé.”



https://www.aberdeen.com/techpro-essentials/morality-in-artificial-intelligence-the-drive-for-a-better-future/
http://ml-tutorials.kyrcha.info/dt.html

Áttekintés

- Neurális hálók, biológiai alapmodellük
- DeepLearning, miben különbözik a tradicionális megoldásoktól, Deep Neural Network
- DNN-ek felépítése, alapfogalmak: neuron, réteg, aktivációs függvény, hibafüggvény, nemlinearitás, tanítás, tanulási ráta, backward propagation, batch, epoch
- Visszamérés, adatok gyűjtése és feldolgozása
- Tenzorok, mátrixok, vektorműveletek, derivált, hibafüggvény tere
- CNN, konvolúciós és pool réteg, aktivációs térkép, channel, kernel
- Más DL struktúrák
- Egyéb tippek, technikák
- Felelősség kérdése

Kérdések?

mnovak@clementine.hu

Clementine