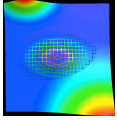


Adatelemzés

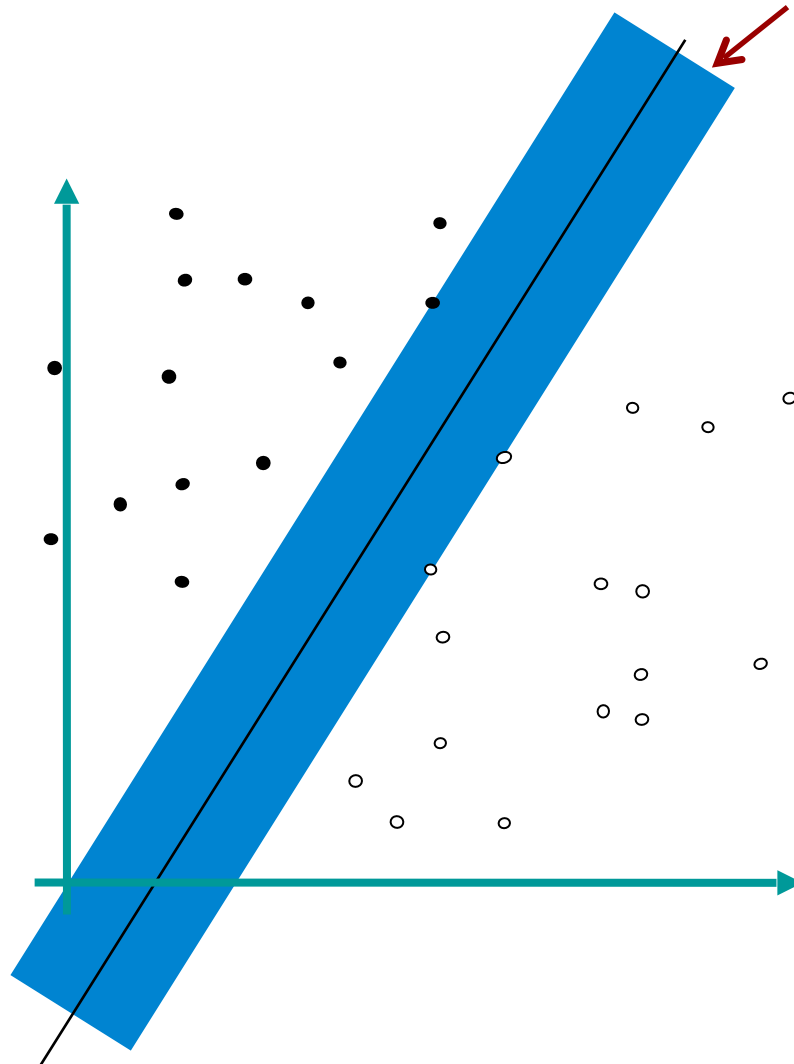
2017-2018 tavaszi félév

2017.04.16-23

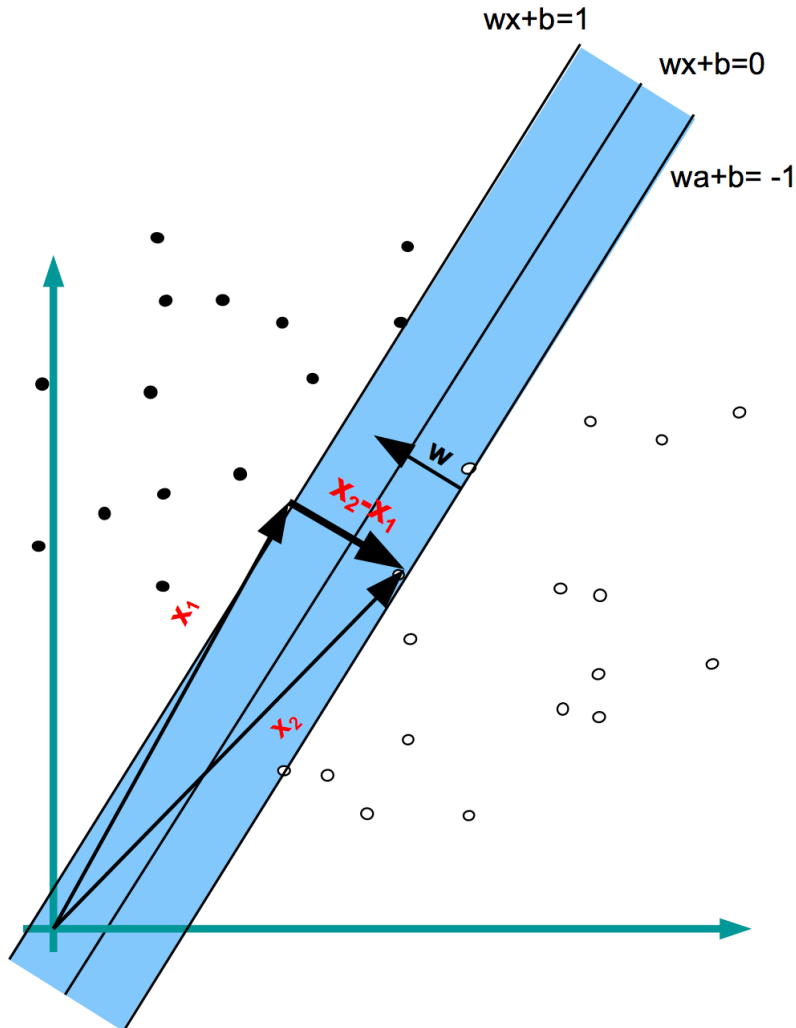
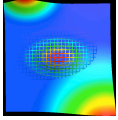
1. Polinomiális szeparálás
2. NN
3. Aporiori



Margó (ism.)



Maximális margó



$$w x_1 + b = 1$$

$$w x_2 + b = -1$$

$$x_2 - x_1 = -n \frac{w}{\|w\|} \longrightarrow x_2 = x_1 - n \frac{w}{\|w\|}$$

$$w x_2 + b = \left(x_1 - n \frac{w}{\|w\|} \right) w + b = -1$$

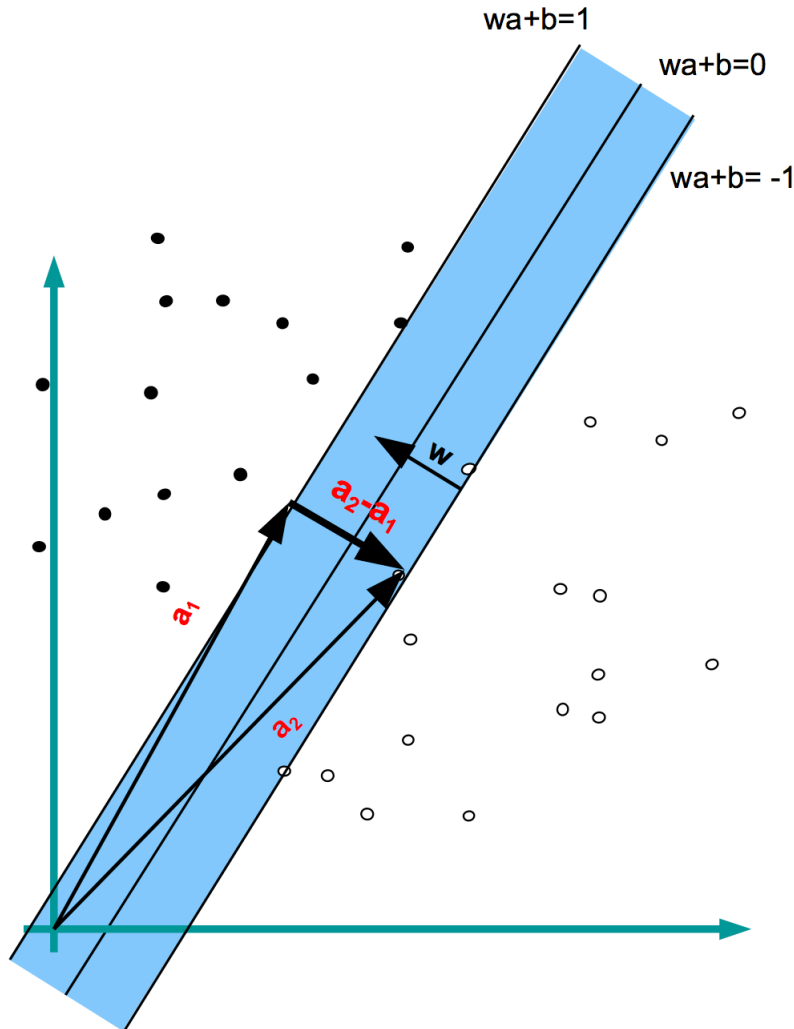
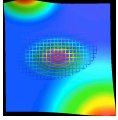
$$w x_1 + b = 1$$

$$w x_1 + b - n \frac{w}{\|w\|} w = -1$$

$$1 - n \frac{w}{\|w\|} w = -1 \longrightarrow$$

$$n = \frac{2}{\|w\|}$$

Maximális margó



$$wa_1 + b = 1$$

$$wa_2 + b = -1$$

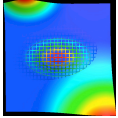
$$a_2 - a_1 = -n \frac{w}{\|w\|} \longrightarrow a_2 = a_1 - n \frac{w}{\|w\|}$$

Maximize $n = \frac{2}{\|w\|}$



Minimize $\|w\|$

$$1 - n \frac{w}{\|w\|} w = -1 \longrightarrow n = \frac{2}{\|w\|}$$



Maximális margó (ism.)

Definíció: egy hipersík margója a hipersík és bármely minta minimális távolsága:

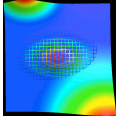
$$\rho_{\omega}(X) = \min_{x \in X} \frac{|x^T \omega|}{\|\omega\|}$$

Ideális megoldás $\leftrightarrow$ maximális margó ÉS jó címkézés:

$$\omega^* = \arg \max_{\omega \in \Omega} \rho_{\omega}(X)$$

$$\text{ahol} \quad y_t x_t^T \frac{\omega}{\|\omega\|} > 0, \forall t$$

Feltéve, hogy az osztálycímkéink: $y_t \in \{-1, +1\}$



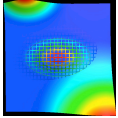
Maximális margó (balról)

A logisztikus regresszióhoz hasonló értelmezés:

$$\omega^* = \arg \max_{\omega \in \Omega} \min_{x \in X} |p(x | \omega) - 0.5|$$

$$\text{ahol} \quad y_t x_t^T \frac{\omega}{\|\omega\|} > 0, \forall t$$

$$\text{ahol} \quad p(x | \omega) = \text{sigm}(x | \omega) = \frac{1}{1 + e^{-(x^T \omega + \omega_0)}}$$



Maximális margó (balról)

Definíció alapján, minden x tanuló mintára igaz, hogy

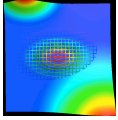
$$y(x^T \frac{\omega}{||\omega||}) \geq \rho_\omega$$

$$\omega' = \frac{\omega}{||\omega||\rho_\omega}$$

Így amennyiben bevezetünk egy új hipersíkot:

$$y(x^T \omega') \geq 1$$

A feltételünk még egyszerűbbé válik:



Maximális margó (balról)

A végső (nem soft) optimalizálásunk átformázható

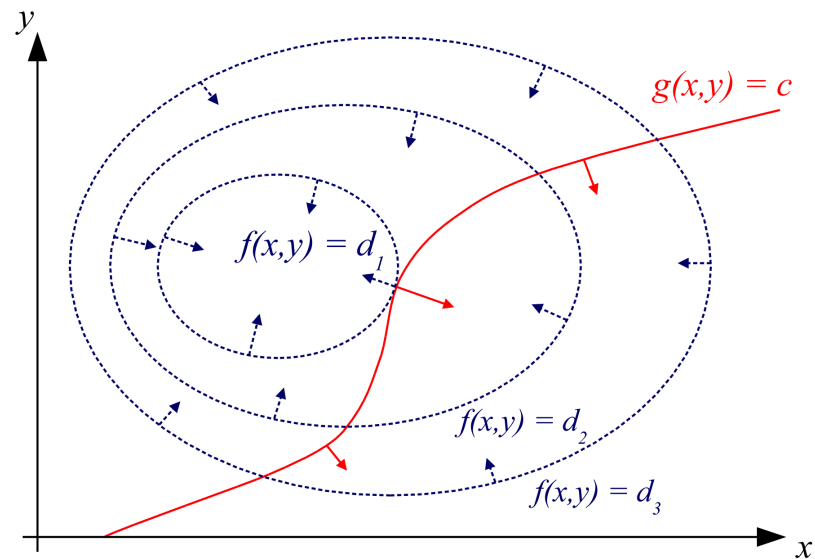
$$\min_{\omega} ||\omega||^2$$

$$\text{ahol } y_t(x_t^T \omega) \geq 1, \forall t$$

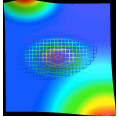
(Lsd. ábra előző óráról)

Miért nem lehet közvetlenül megoldani?

(Megj.: megoldás a Lagrange duális!)



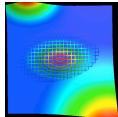
Ábra: wiki



Nem lineárisan szeparálható problémák

Lehetséges-e lineáris szeparálót találni a lenti adathalmazhoz?

-1	+1
+1	-1



Polinomiális szeparátor

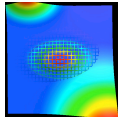
Feltételezésünk szerint létezik egy $p(x)$ D -ed fokú polinom, mely $p(x) > 0$ akkor és csak akkor ha x pozitív címkéjű.

Ha D -ed fokú a polinomunk akkor minden $(i_1, i_2, \dots, i_d)$ -re igaz, hogy

$$i_1 + i_2 + \dots + i_d \leq D$$

Azaz a monomok formája

$$x_1^{i_1} x_2^{i_2} \cdot \dots \cdot x_d^{i_d}$$



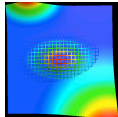
Polinomiális szeparáló

Ha a monomok együtthatói $w_{i_1, i_2, \dots, i_d}$, a polinomunk a következő alakot veszi fel

$$p(x_1, x_2, \dots, x_d) = \sum_{\substack{i_1, i_2, \dots, i_d \\ i_1 + i_2 + \dots + i_d \leq D}} w_{i_1, i_2, \dots, i_d} x_1^{i_1} x_2^{i_2} \cdots x_d^{i_d}$$

Már kis D mellett is lehet nagy az együtthatók száma!

$$\sum_{k=1}^D \binom{d+k-1}{k} + 1$$



Polinomiális szeparáló

Példa: legyen d és D is 2.

Ebben az esetben a monomok száma 6,

i_1, i_2 a lehetséges értékei $\{(0,0), (1,0), (0,1), (2,0), (1,1), (0,2)\}$

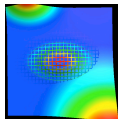
Legyen $(0,0)$ a “bias” (b), a polinomunk végső formája

$$p(x_1, x_2) = b + w_{10}x_1 + w_{01}x_2 + w_{11}x_1x_2 + w_{20}x_1^2 + w_{02}x_2^2$$

Minden $\mathbf{x}_i$ tanulópontra

$$b + w_{10}x_{i1} + w_{01}x_{i2} + w_{11}x_{i1}x_{i2} + w_{20}x_{i1}^2 + w_{02}x_{i2}^2 > 0 \text{ ha a címke } y_i = +1$$

$$b + w_{10}x_{i1} + w_{01}x_{i2} + w_{11}x_{i1}x_{i2} + w_{20}x_{i1}^2 + w_{02}x_{i2}^2 < 0 \text{ ha a címke } y_i = -1$$



Polinomiális szeparáló

A fenti eset egy véges transzformációnak tekinthető.

Legyen minden $i_1, i_2, \dots, i_d$ összege legfeljebb D , akkor a transzformációnk

$$\varphi(\mathbf{x}): \mathbb{R}^d \rightarrow \mathbb{R}^m$$

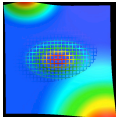
$$\mathbf{a}_i = (x_1, x_2, \dots, x_d) \quad \xrightarrow{\quad} \quad x_1^{i_1} x_2^{i_2} \cdots x_d^{i_d}$$

$$\varphi(\mathbf{x}) = (x_1, x_2, x_1^2, x_1 x_2, x_2^2)$$

ha $d=2$ és $D=2$:

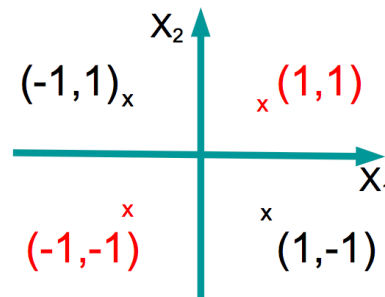
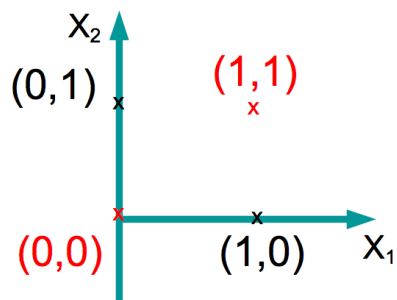
ha $d=3$ és $D=2$:

$$\varphi(\mathbf{x}) = (x_1, x_2, x_3, x_1 x_2, x_1 x_3, x_2 x_3, x_1^2, x_2^2, x_3^2)$$

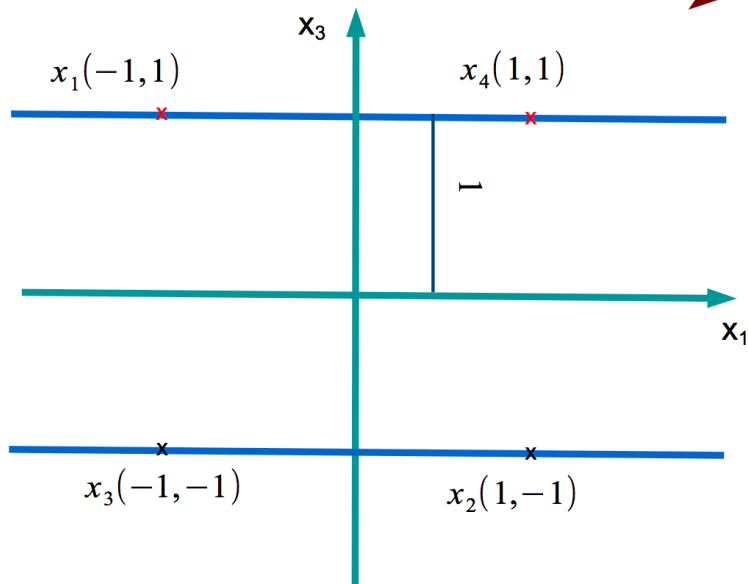


Polinomiális szeparáló

Eredeti tér



Transzformált tér



$\Phi(X)$

$$x_{11} = -1, x_{13} = 1$$

$$y_1 = -1$$

$$x_{21} = 1, x_{23} = -1$$

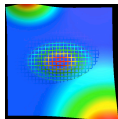
$$y_2 = +1$$

$$x_{31} = -1, x_{33} = -1$$

$$y_3 = +1$$

$$x_{41} = 1, x_{43} = 1$$

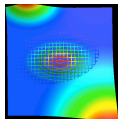
$$y_4 = -1$$



Polinomiális kernel

Illesszük vissza az eredeti problémánkba a polinomiális transzformációnkat.

Álljunk meg egy pillanatra! El tudjuk kerülni a transzformált vektorok kiszámolását?



Support vector machine

Lemma: konvex problémánk megoldása felírható a minták lineáris kombináltjaként

Ebből következik hogy a norma

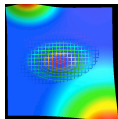
Illetve mi

$$||\omega||^2 = ((\sum \alpha_i x_i)^T (\sum \alpha_i x_i)) = \sum \alpha_i \alpha_j x_i x_j$$

$$y_i(\omega^T x) = y_i \sum \alpha_i x_i x = y_i \sum \alpha_i k(x_i, x)$$

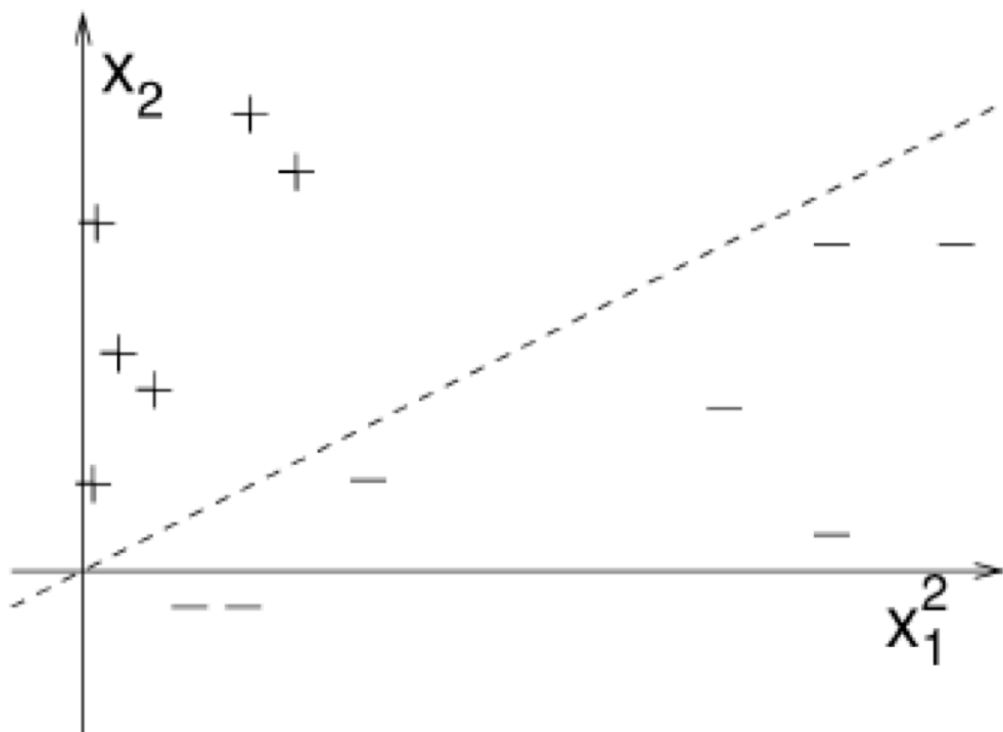
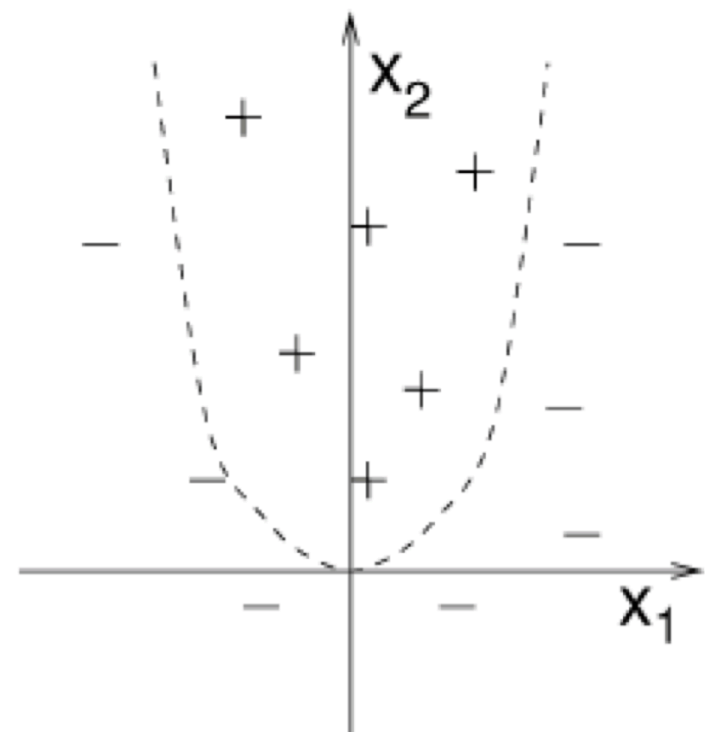
ahol a $k(x, x)$ függvényt kernel (mag) függvénynek nevezzük. A nem 0-a együtthatójú tanulóelemek a support vektorok (innen a név).

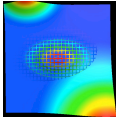
Készítsünk polinomiális szeparálót a belső szorzatok ismeretével!



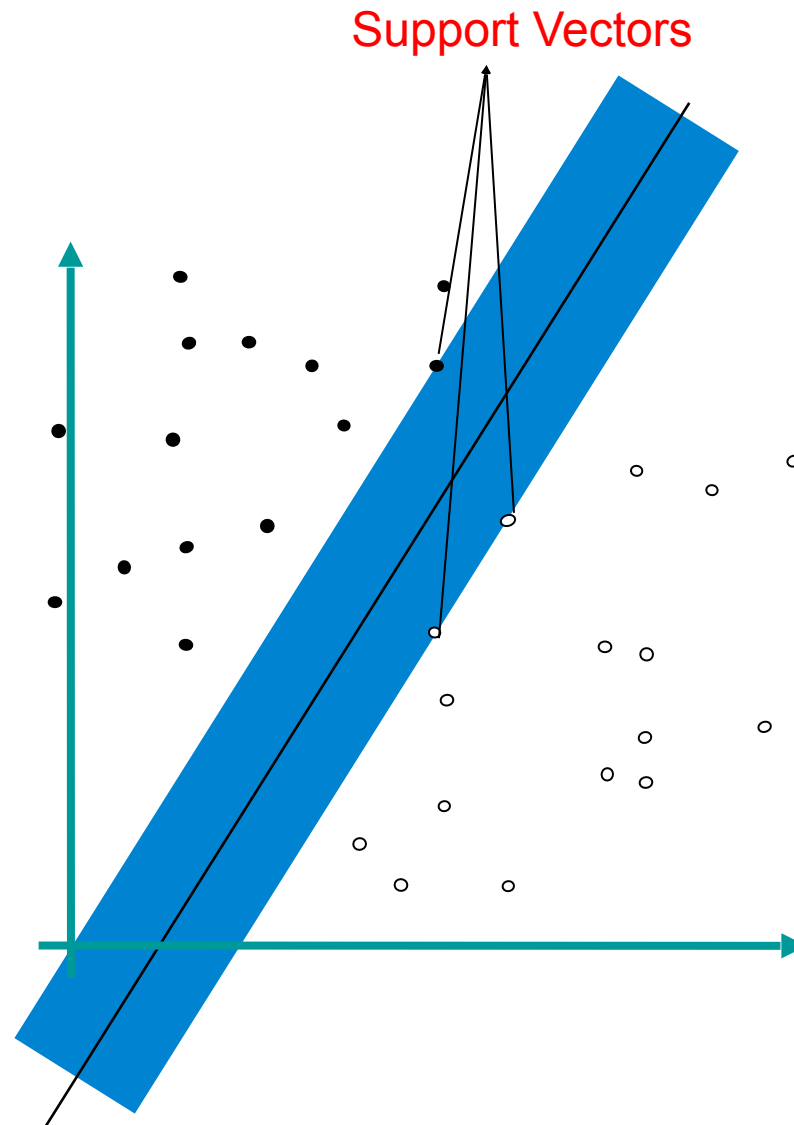
Polinomiális kernel

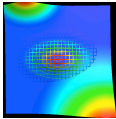
$$\Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$$





Maximális margó





SVM és Python

SVM python: scikit (igazából egy libSVM wrapper)

```
svc = SVC()
```

kernel: rbf, poly, linear, precomputed

```
svc.set_params(kernel='rbf', probability=True, cache_size=30000)
```

```
svc.fit(train[:,1:],train[:,0])
```

$wx = \sum \alpha_i K(x_i, x)$ értékek, folytonos

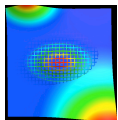
```
pred_svc=svc.decision_function(test[:,1:])
```

threshold és sigmoid (ebben az esetben a probability True), folytonos

```
pred_svc=svc.predict_proba(test[:,1:])
```

sima diszkrét jóslat

```
pred_svc=svc.predict(test[:,1:])
```

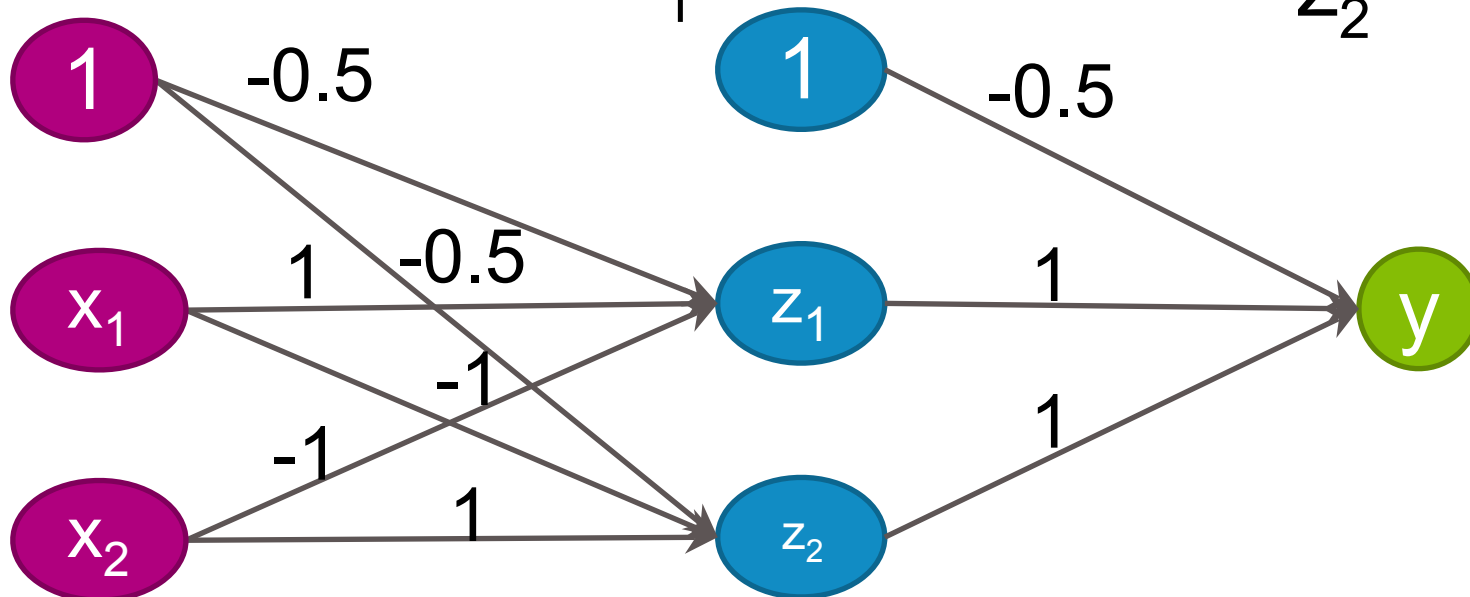


Neurális hálózatok

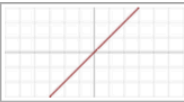
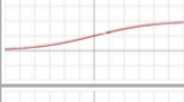
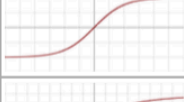
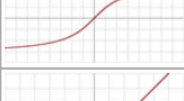
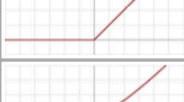
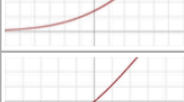
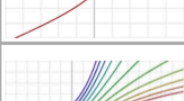
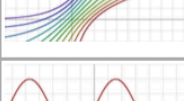
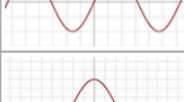
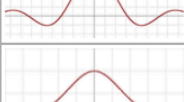
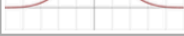
Két fontos jellemző

- Huzalozás: maga a gráf (irányított)

$$\text{XOR} = \underbrace{x_1 \text{ AND NOT } x_2}_{z_1} \text{ OR } \underbrace{\text{NOT } x_1 \text{ AND } x_2}_{z_2}$$



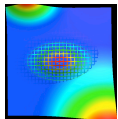
Aktíváló függvények

Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$
Bent identity		$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2 + 1}} + 1$
SoftExponential		$f(\alpha, x) = \begin{cases} -\frac{\log_e(1 - \alpha(x + \alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1 - \alpha(x + \alpha)} & \text{for } \alpha < 0 \\ 1 & \text{for } \alpha = 0 \\ e^{\alpha x} & \text{for } \alpha > 0 \end{cases}$
Sinusoid		$f(x) = \sin(x)$	$f'(x) = \cos(x)$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$

Ez határozza meg a neuron kimenetét
Lehet:

- lineáris/
nem lineáris
- Korlátos/
nem korlátos
értékkészletű
- Általában monoton,
de nem szükségszerű

forrás: wiki



“shallow” vs. “deep”

Hipotézis: deep, hierarchikus modellek exponenciálisan hatékonyabbak mint a shallow modellek (Bengio et al. 2009, Le Roux and Bengio, 2010, Delalleau and Bengio, 2011 etc.)

Delalleau and Bengio, 2011: deep sum-product network exponenciálisan kisebb hálózati elemet igényel

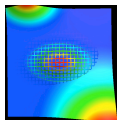
Mit nevezünk egy neurális háló mélységének?

Feed forward hálózatok esetében a bemeneti és a kimeneti réteg közti nem lineáris rétegek száma.

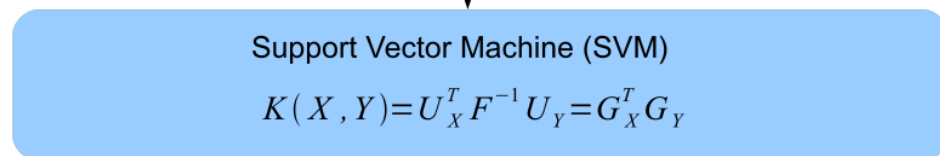
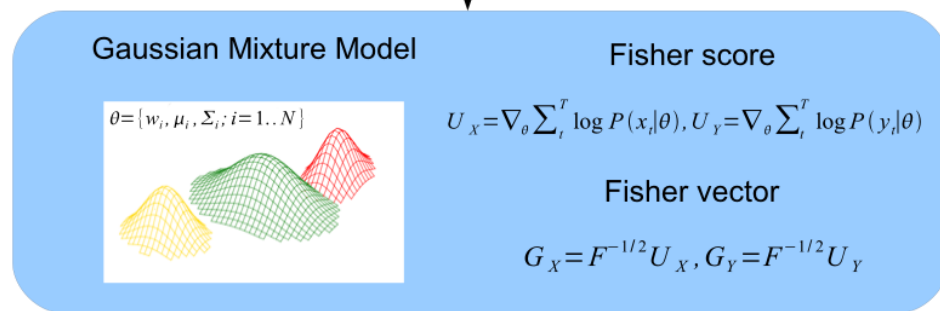
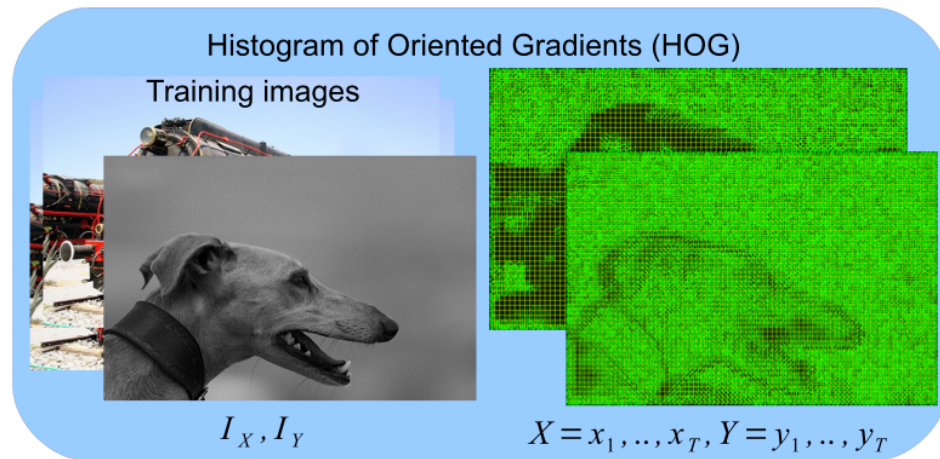
Nem minden típusú hálózatra igaz ez a megállapítás!

Három nagyobb terület:

- generatív (pl. Boltzmann Machine (RBM, DBN), VAE, GAN):
 - Általános modell, adatot modellezi
- Feed-forward NN: pl. Convolutional NN (és iterációi: Inception, ResNet stb.)
 - DAG struktúra, nincs irányított kör
- Recurrent NN (pl. LSTM és GRU):
 - Irányított kör is megengedett (általában az előző állapot)

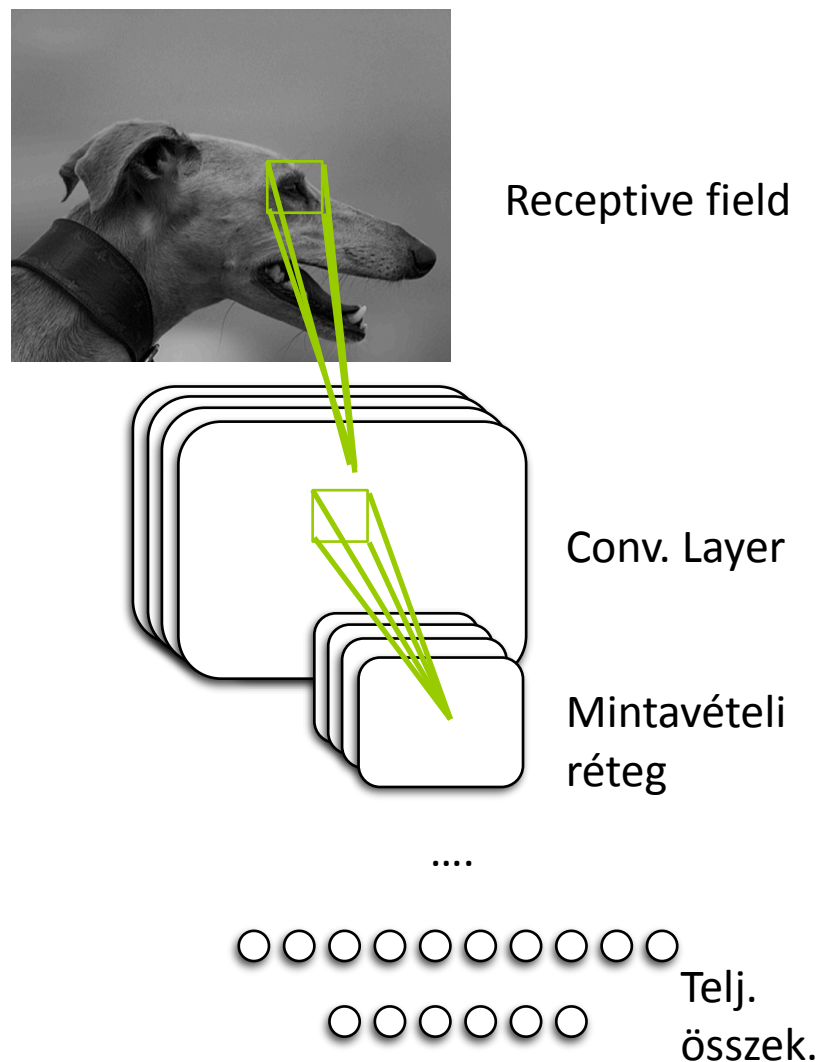


Convolutional Neural Network (CNN)

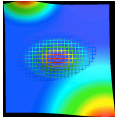


Tradicionális képosztályozás

vs.

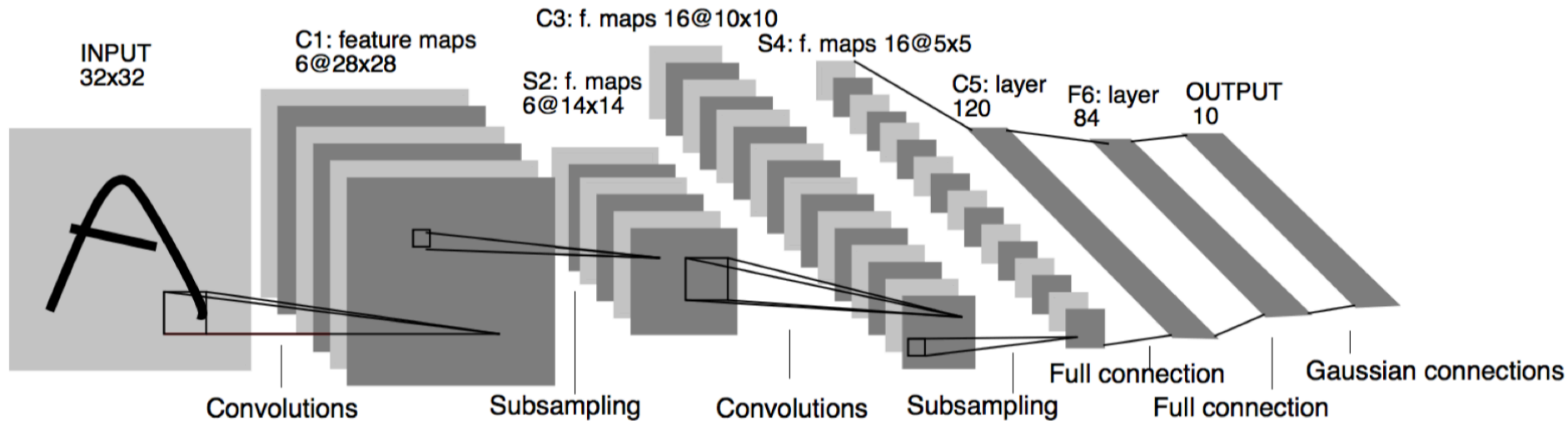


CNN



LeNet-5

Kézírásfelismerés (LeCun et al. 1998)



- Tanult alacsony szintű feature kinyerés
- Nem csak tanult de sokféle is (LeNet 5: 6 +16+120)
- A mintavételi réteg egyfajta térbeli modellt eredményez szemben a BOW-al
- Miután nem teljesen összekapcsolt, kevesebb paraméter

Tanítás: back propagation mini-batch alapon gradiens-el, inicializáció normál eloszlásból

A definíció szerint a CNN mély struktúra

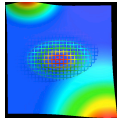
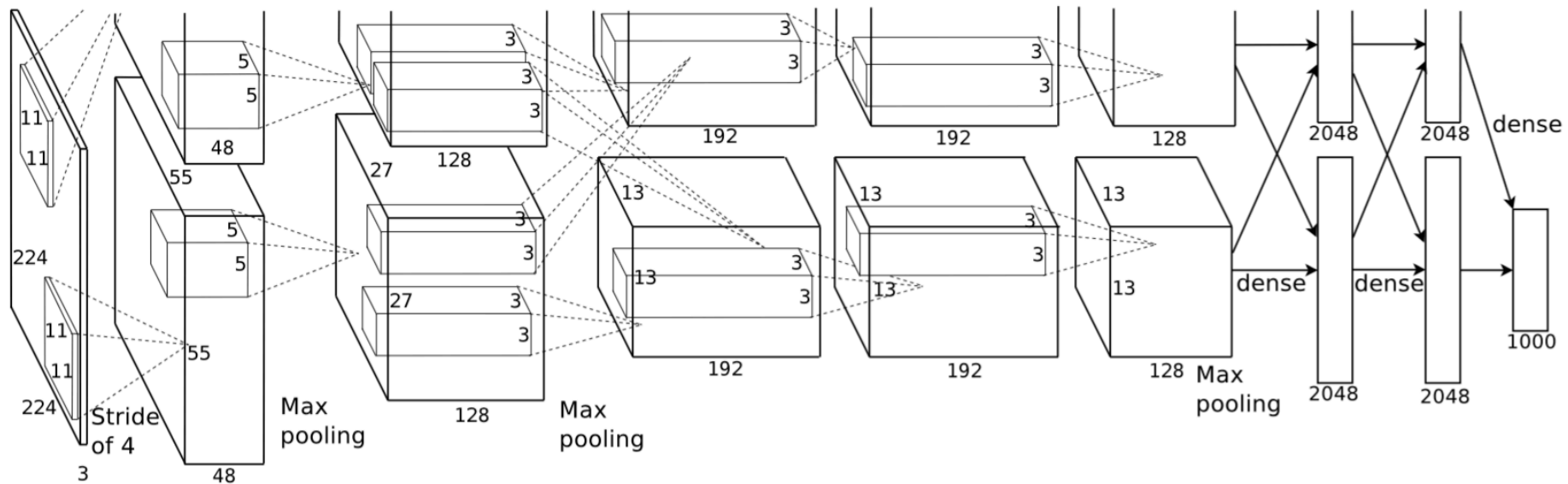


Image classification CNN-el

Krizhevsky et al. 2012 , ők nyerték az ImageNet-et 2012-ben



Fejlődés a LeNet-hez képest:

- ReLU és nem sigmoid és Local response normalization (1.2%..1.4% javulás)
- A mintavételi réteg átfedő (0.3%..0.4% javulás)
- Tradicionális képi trükkök: horizontális tükrözés, véletlen mintavétel, PCA az RGB csatornák felett és zajkhozadáása (1% javulás)
- Dropout

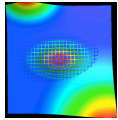


Image classification CNN-el

Krizhevsky et al. 2012

ImageNet: 150k teszt kép és 1.2 millió tanuló kép, 1000 címke.

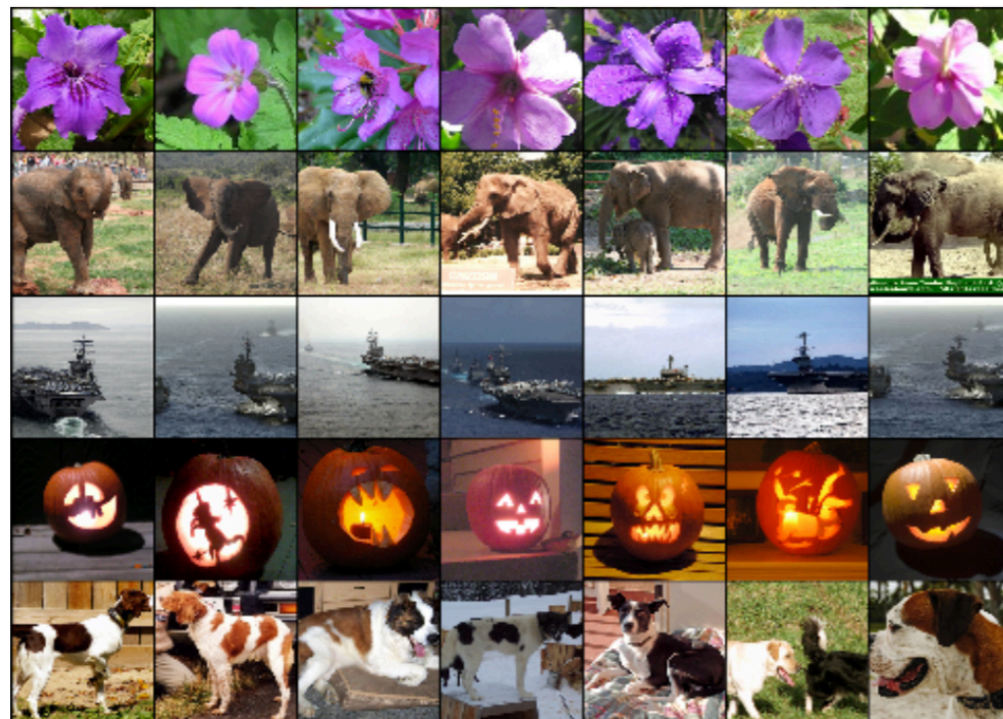
top-1 és top-5 hiba arány

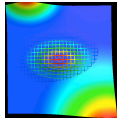
Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
<i>SIFT + FVs [7]</i>	—	—	26.2%
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	16.4%
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	15.3%

* - Külső adat

4096 dimenziós repr.

Tanulás: 5-6 nap két Nvidia GTX 580 3GB

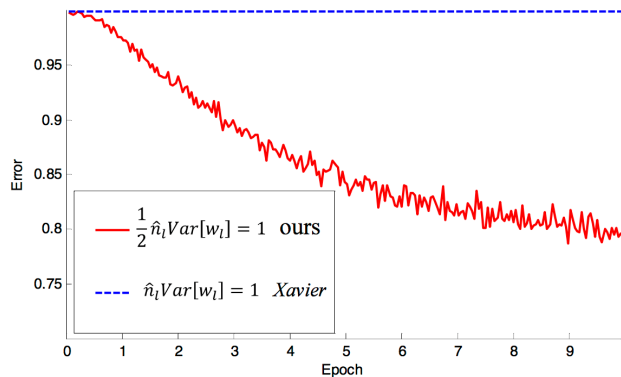




Jelenleg ismert (egyik)legjobb eredmény

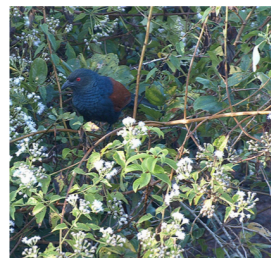
He et al. 2015 (MS research): Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

Parametric ReLU + nem Gaussian inicializálás + extrém méret: A:19 réteg, B/C: 22 réteg,



Tanulás : 8xK40 Nvidia GPU 3..4 hét (!)

model	top-1	top-5
MSRA [11]	29.68	10.95
VGG-16 [25]	28.07 [†]	9.33 [†]
GoogLeNet [29]	-	9.15
A, ReLU	26.48	8.59
A, PReLU	25.59	8.23
B, PReLU	25.53	8.13
C, PReLU	24.27	7.38



GT: coucal
1: coucal
2: indigo bunting
3: lorikeet
4: walking stick
5: custard apple



GT: komondor
1: komondor
2: patio
3: llama
4: mobile home
5: Old English sheepdog



GT: yellow lady's slipper
1: yellow lady's slipper
2: slug
3: hen-of-the-woods
4: stinkhorn
5: coral fungus



GT: torch
1: stage
2: spotlight
3: torch
4: microphone
5: feather boa

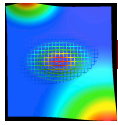


GT: banjo
1: acoustic guitar
2: shoji
3: bow tie
4: cowboy hat
5: banjo

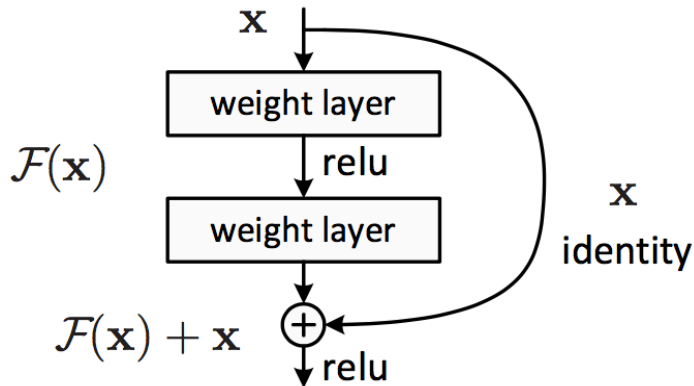


GT: go-kart
1: go-kart
2: crash helmet
3: racer
4: sports car
5: motor scooter

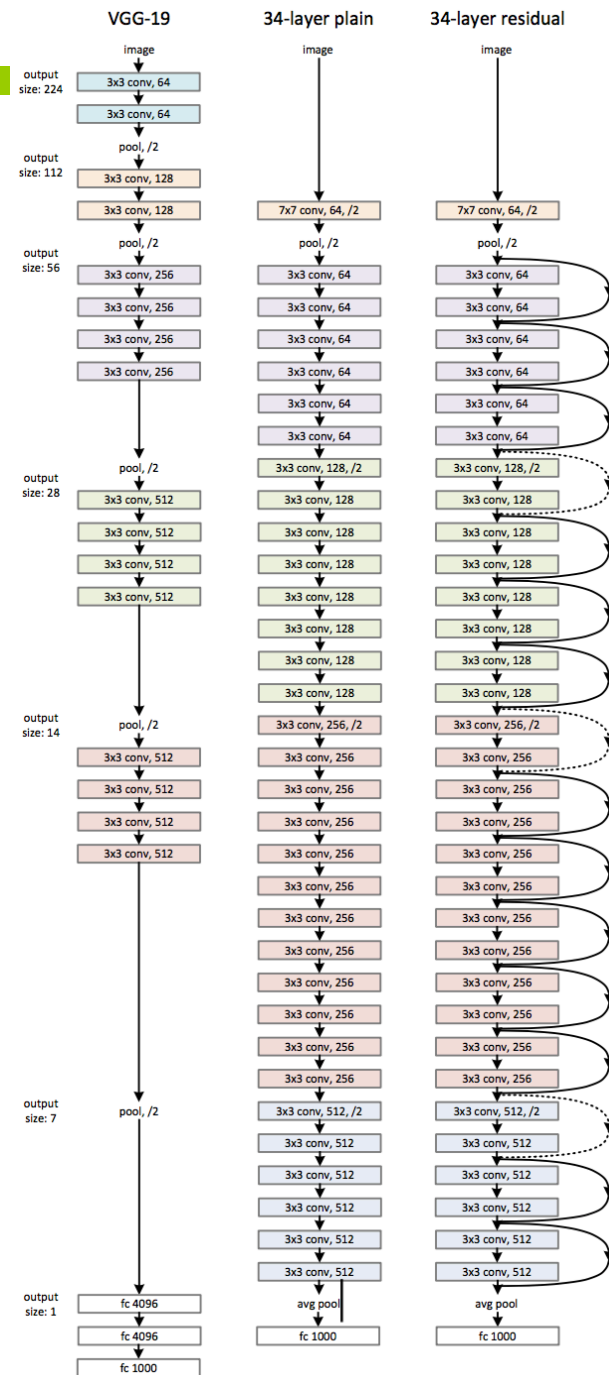
Talán legjobb

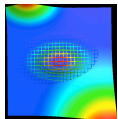


[He et al. 2015]: ResNet: “Is learning better networks as easy as stacking more layers?”



model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PRelu-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	21.43	5.71





Recurrent Neural Networks (RNN)

Dinamikus (diszkrét) rendszerek szimulációja (Rumelhart *et al.* 1986)

Három elem:

x_t : bemeneti réteg a t -dik időpontban

y_t : kimeneti réteg a t -dik időpontban

h_t : a rejtett réteg a t -dik időpontban

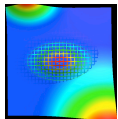
Ahogy sejtjük, elég egyszerű a modell leírása:

$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1})$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t),$$

Memóriaként működik, az előző bemenetek számítanak.

Szavak egy mondatban?



Recurrent Neural Networks (RNN)

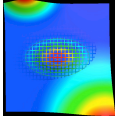
Legyen D egy sorozat:

$$D = \left\{ \left((\mathbf{x}_1^{(n)}, \mathbf{y}_1^{(n)}), \dots, (\mathbf{x}_{T_n}^{(n)}, \mathbf{y}_{T_n}^{(n)}) \right) \right\}_{n=1}^N$$

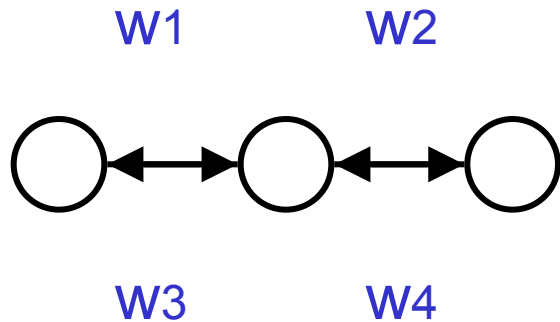
RNN esetében a paraméterek becslése (Θ) a következő költségfüggvény minimalizálási feladata:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N \sum_{t=1}^{T_n} d(\mathbf{y}_t^{(n)}, f_o(\mathbf{h}_t^{(n)}))$$

ahol $\mathbf{h}_t^{(n)} = f_h(\mathbf{x}_t^{(n)}, \mathbf{h}_{t-1}^{(n)})$ és $\mathbf{h}_0^{(n)} = \mathbf{0}$. $d(y, f(h))$ valamilyen divergencia mérték (távolságok, KL, JS stb.).

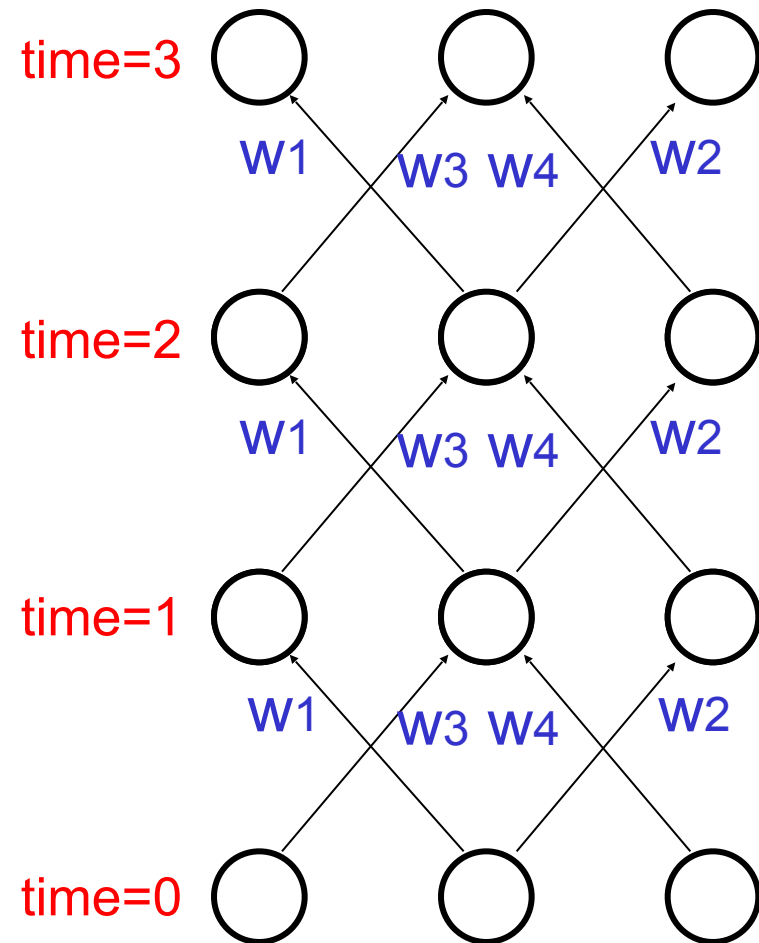


Recurrent Neural Networks (RNN)

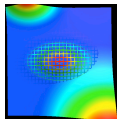


Az RNN kibontása

Ez már önmagában “deep”-szerű 😊
Megkötés, hogy minden szinten a súlyok azonosak (azaz időben nem változnak)



(ábra: Geoffrey Hinton)



Recurrent Neural Networks (RNN)

Egy egyszerűbb példa:

$$\mathbf{h}_t = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}) = \phi_h (\mathbf{W}^\top \mathbf{h}_{t-1} + \mathbf{U}^\top \mathbf{x}_t)$$

$$\mathbf{y}_t = f_o(\mathbf{h}_t, \mathbf{x}_t) = \phi_o (\mathbf{V}^\top \mathbf{h}_t) ,$$

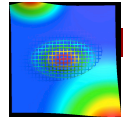
ahol $\mathbf{W}$, $\mathbf{U}$ és $\mathbf{V}$ a paraméter mátrixok és Φ korlátos nem lineáris függvények. pl. szigmoid.

Ebben az esetben a paraméterek becsülhetőek SGD-vel back prop.-al a kibontott hálózaton (Rumelhart *et al.* 1986). A trükk, hogy miután felbontottuk a hálózatot átlagolással azonosan tartjuk a súlyokat.

Jó jó , de hogy lesz ebből “deep” RNN?

Long short term memory (LSTM)

33



[Hochreiter & Schmidhuber 1997]

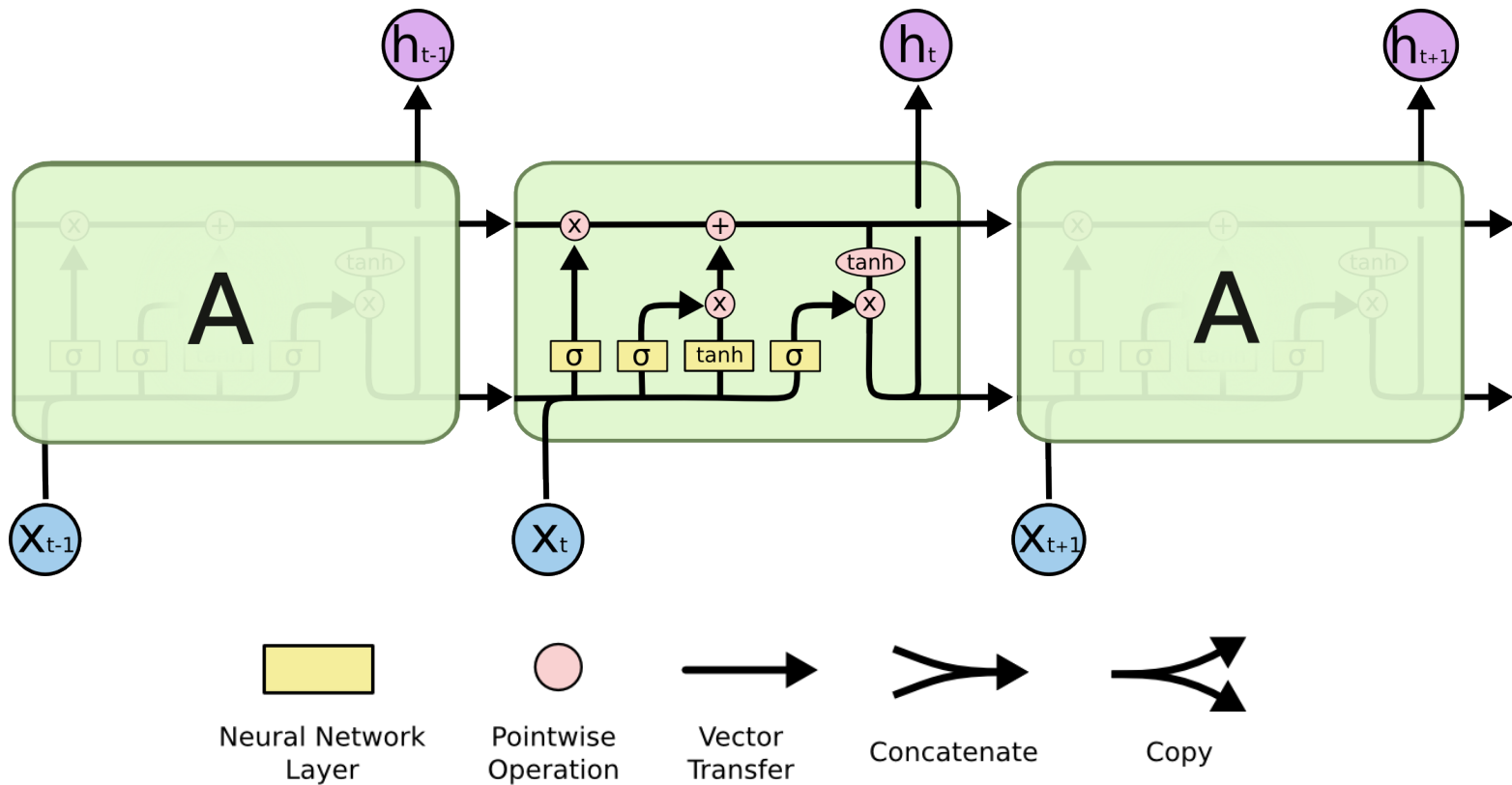
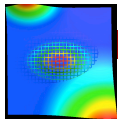
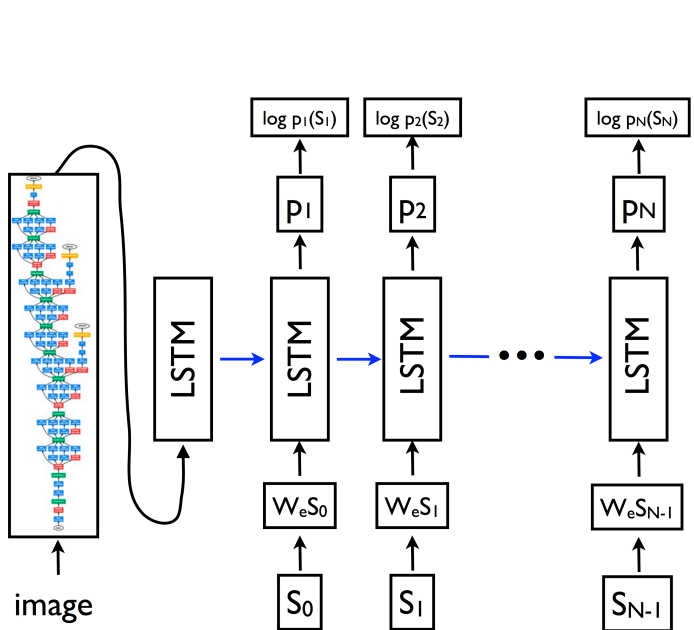


Fig: Christopher Olah



Kép annotáció (Vinyals et al, 2016 szept.)

CNN + WE + LSTM + attention: image caption



Better Image Model

InitialModel: A close up of a plate of food on a table



BestModel: A bunch of bananas and a bottle of wine.



Colors

InitialModel: A train that is sitting on the tracks.



BestModel: A blue and yellow train traveling down train tracks.

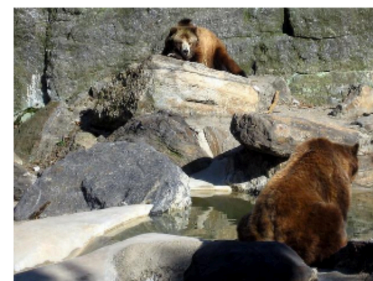


Better Image Model

InitialModel: A close up of a person eating a hot dog.



BestModel: A woman holding a banana up to her face.



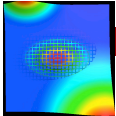
Counting

InitialModel: A brown bear is swimming in the water.



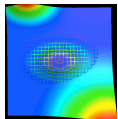
BestModel: Two brown bears sitting on top of rocks.

“Ensemble” és “scheduled sampling”

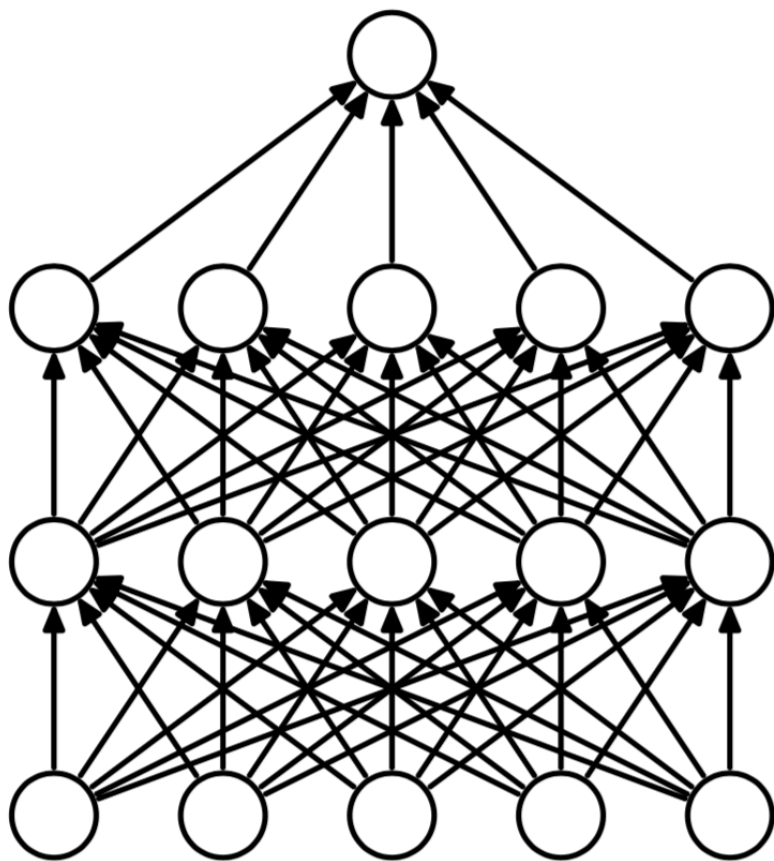


Megoldatlan problémák

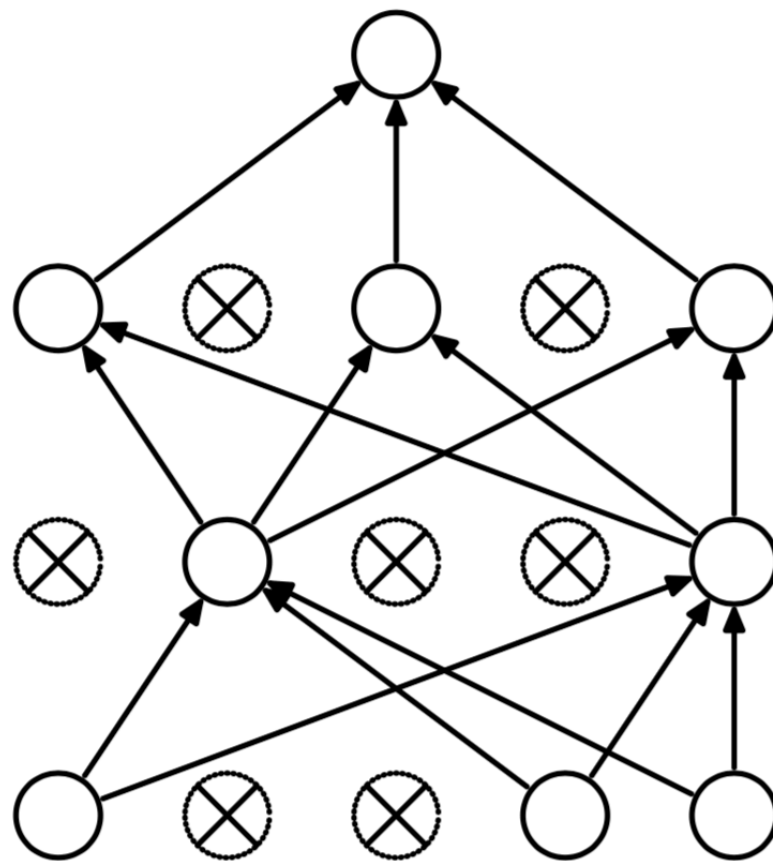
- “Overfitting” (túltanulás):
 - DropOut [Hinton et al., 2012]
 - DropConnect [Wan et al., 2013]
- “Saturation”, “vanishing gradients” és “sparsity”:
 - pReLU [He et al., 2015]
 - Maxout [Goodfellow et al., 2013]
 - Network-in-Network [Lin et al., 2014]
 - local response/batch normalization [Ioffe et al., 2015]
 - BN Maxout NiN [Chang et al., 2015]
- Komplexitás:
 - Convolution
 - FastFood [Yang et al., 2015]
 - Generalizáció [Zhang et al., 2017, Lin & Tegmark, 2017]
 - Lower bounds [Lin & Tegmark, 2016]
- Struktúra:
 - spectral representation (pooling) [Rippel & Snoek, 2015]
 - Identity map és residual block [He et al., 2015]
 - Manifold tangent classifier, high-order contractive auto-encoder [Rifai et al., 2011]



Dropout (Hinton et al. 2012)

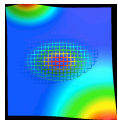


Klasszikus háló



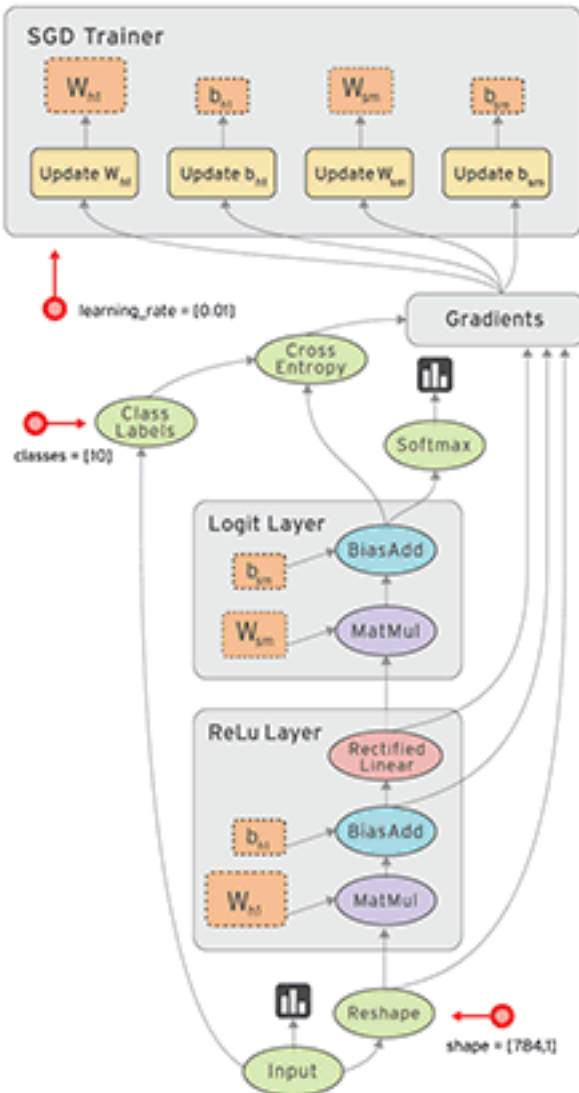
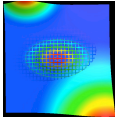
Tanulás során

(ábra: Srivastava et al. 2014)



Pár implementáció

	Restrictions	Wrapper	Architectures	Notes
Theano	feed forward/ recurrent hálók	Python	Multi core/CUDA	
Chainer	feed forward/ recurrent hálók	Python	Multi core(?)/CUDA	
GraphLab	Feed Forward: CNN, DBN	Python/C++	Multi core/CUDA/ distributed	Compact, Hadoop
TensorFlow	feed forward/ recurrent hálók	Python/C++	Multi core/CUDA/ distributed	
Caffe	Feed Forward: CNN, DBN	Python/Matlab	Multi core/CUDA	Blob
Torch	feed forward/ recurrent hálók	Lua	Multi core/CUDA	Eredetileg vizuális adatokra



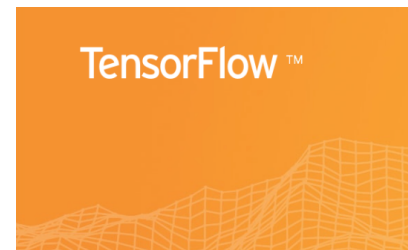
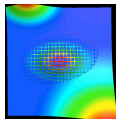
TensorFlow™

Google-nél fejlesztették, alapvetően elosztott, többmagos vagy éppen GPU-s környezetben is használható (utóbbi csak Linux)

Python és C++

Sokféle háló: CNN, DBN, RNN, LSTM

Példa: MNIST



Telepítés:

Lépünk be a saját környezetünkbe.

Mi: Ubuntu/Linux 64-bit, CPU only, Python 3.4

```
export TF_BINARY_URL=https://storage.googleapis.com/tensorflow/linux/cpu/  
tensorflow-0.19.0rc1-cp35-cp35m-linux_x86_64.whl
```

```
pip install --ignore-installed --upgrade pip setuptools
```

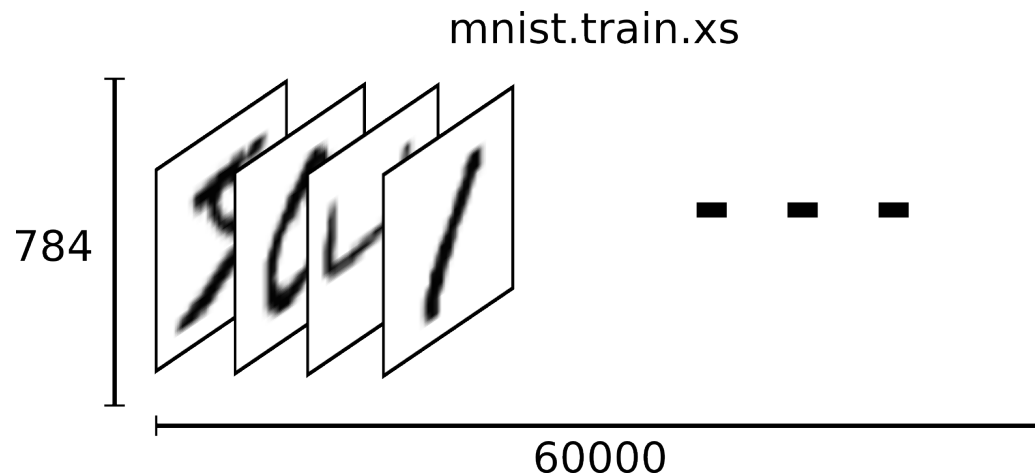
```
pip install --upgrade $TF_BINARY_URL
```

Teszt:

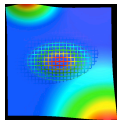
```
Import tensorflow as ts
```



- 60000 kép
- 28x28 felbontás
- egy csatorna
- Tíz címke: 0..9



(a legjobb kezdő tutorial s mivel ha változik bármi a keretben, megváltoztatják :))



MNIST kézírásfelismerés (shallow)

mint előbb a graphlab-nál, itt is a tutorialt követjük

```
import tensorflow as tf
```

```
from tensorflow.examples.tutorials.mnist import input_data
```

```
mnist= input_data.read_data_sets("MNIST_data/", one_hot=True)
```

beállítjuk az alap változókat és az bemenetet

```
x = tf.placeholder("float", [None, 784])
```

```
W = tf.Variable(tf.zeros([784,10]))
```

```
b = tf.Variable(tf.zeros([10]))
```

aktiváló függvényünk többosztályos softmax

```
y = tf.nn.softmax(tf.matmul(x,W) + b)
```

eredeti osztálycímkek

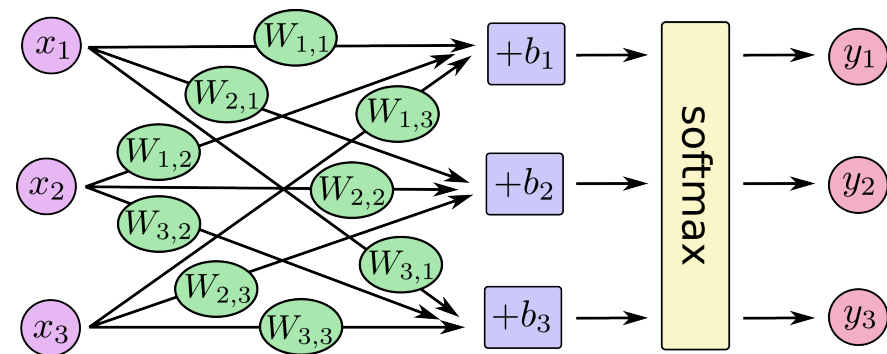
```
y_ = tf.placeholder("float", [None,10])
```

célfüggvény

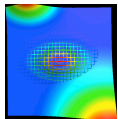
```
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

generálunk egy update lépést

```
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
```



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$



MNIST kézírásfelismerés (shallow)

elindítunk egy session-t, de előbb megadjuk az inicializáció-t

```
init = tf.initialize_all_variables()
```

```
sess = tf.Session()
```

```
sess.run(init)
```

1000 lépésben tanítunk batch alapon

```
for i in range(1000):
```

```
    batch_xs, batch_ys = mnist.train.next_batch(100)
```

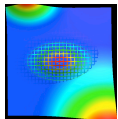
```
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

kiértékelünk

```
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
```

```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
```

```
print sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels})
```



MNIST kézírásfelismerés (CNN)

mint a nevéből is adódik, jobban illeszkedik az interaktív python környezetekbe
sess = tf.InteractiveSession()

definiálunk egyszerű objektumokat az inicializációhoz

```
def weight_variable(shape):  
    initial = tf.truncated_normal(shape, stddev=0.1)  
    return tf.Variable(initial)
```

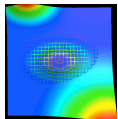
```
def bias_variable(shape):  
    initial = tf.constant(0.1, shape=shape)  
    return tf.Variable(initial)
```

egyszerű 2d conv : lépésköz 1, W a filter formája, x maga a kép amire alkalmazzuk

```
def conv2d(x, W):  
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')
```

alap mintavétel, x-ből 2x2-es csoportonként lesz egy negyed annyi elemből álló kép

```
def max_pool_2x2(x):  
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

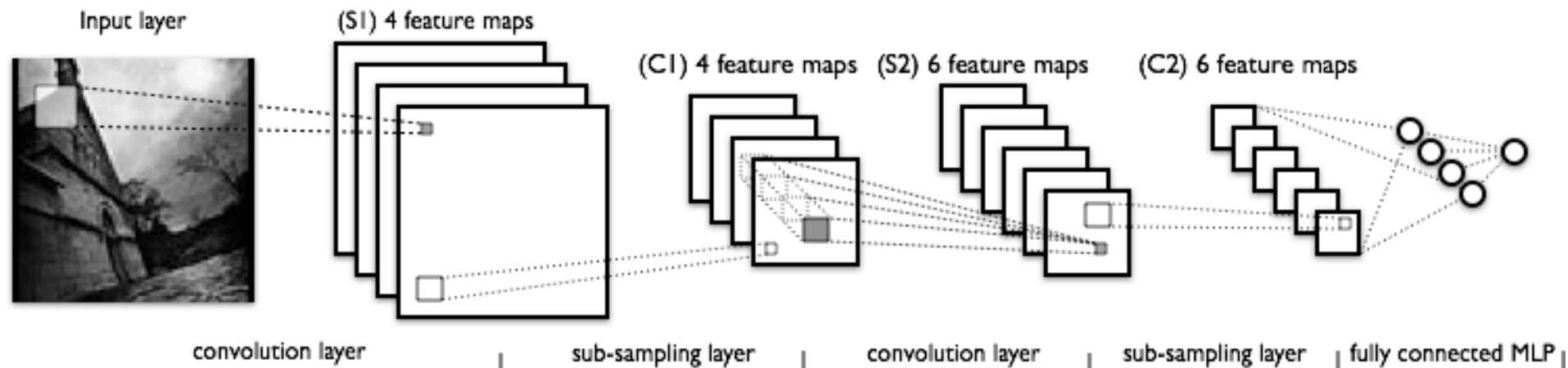


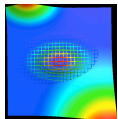
MNIST kézírásfelismerés (CNN)

első és második konvolúciós és mintavételi réteg

```
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])
x_image = tf.reshape(x, [-1,28,28,1])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)
```

```
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```





MNIST kézírásfelismerés (CNN)

teljesen összekapcsolt réteg a 7x7-es képekből (64) egy 1024 dimenziós flat rejtett réteg

```
W_fc1 = weight_variable([7 * 7 * 64, 1024])
```

```
b_fc1 = bias_variable([1024])
```

```
h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
```

ReLU

```
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

dropout során ez a réteg megmarad!

```
keep_prob = tf.placeholder("float")
```

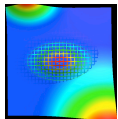
```
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

Kimeneti réteg, megint softmax

```
W_fc2 = weight_variable([1024, 10])
```

```
b_fc2 = bias_variable([10])
```

```
y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```



MNIST kézírásfelismerés (CNN)

ugyanaz a célfüggvény

```
cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
```

ADAM optimizer (Kingma et al. 2015)

```
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
```

```
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
```

```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
```

```
sess.run(tf.initialize_all_variables())
```

```
for i in range(1000):
```

```
    batch = mnist.train.next_batch(50)
```

```
    if i%100 == 0:
```

```
        train_accuracy = accuracy.eval(feed_dict={
```

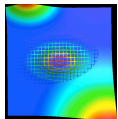
```
            x:batch[0], y_: batch[1], keep_prob: 1.0})
```

```
        print "step %d, training accuracy %g"%(i, train_accuracy)
```

```
        train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
```

```
print "test accuracy %g"%accuracy.eval(feed_dict={
```

```
    x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0})
```



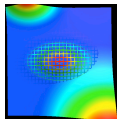
Keras

Install: pip install keras

keret ami Theano és Tensorflow felett (def. TF), GPU def.

```
from keras.datasets import cifar10, mnist
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential, Model
from keras.layers import Dense, Input, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D, normalization
from keras.utils import np_utils
import keras.backend as K
```

```
batch_size = 32
nb_classes = 10
nb_epoch = 10
data_augmentation = True
```



Keras

Bemeneti képfelbontás

```
img_rows, img_cols = 32, 32
```

Cifar10 már színes

```
img_channels = 3
```

Adatok formázása:

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

```
print('X_train shape:', X_train.shape)
```

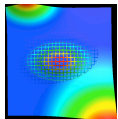
```
print(X_train.shape[0], 'train samples')
```

```
print(X_test.shape[0], 'test samples')
```

Kategórikus címke átalakítás: one-hot kódolás: 3 -> 0,0,0,1,0,0,0,0,0,0

```
Y_train = np_utils.to_categorical(y_train, nb_classes)
```

```
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

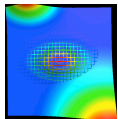


Keras

two Conv + Pooling rétegek

```
model = Sequential()
model.add(Convolution2D(32, 3, 3, border_mode='same', input_shape=X_train.shape[1:]))
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```



Keras

Fully connected réteg + ReLU

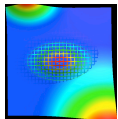
```
model.add(Flatten())  
model.add(Dense(512))  
model.add(Activation('relu'))  
model.add(Dropout(0.5))
```

Kimeneti réteg

```
model.add(Dense(nb_classes))  
model.add(Activation('softmax'))
```

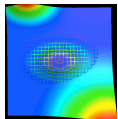
Optimalizáció ADAM (de lehetne stochasztikus gradiens is (SGD))

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])  
model.summary()
```



Keras

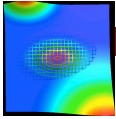
```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
if not data_augmentation:
    print('Not using data augmentation.')
    model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch,
              validation_data=(X_test, Y_test), shuffle=True)
else:
```



Keras

```

else:
    print('Using real-time data augmentation.')
    datagen = ImageDataGenerator(
        featurewise_center=False, # bemenet centrálás, 0 várható érték
        samplewise_center=False, # mintánként
        featurewise_std_normalization=False, # std normalizáció
        samplewise_std_normalization=False, # mintánként
        zca_whitening=False, # ZCA fehérítés
        rotation_range=0, # véletlen rotáció, szög
        width_shift_range=0.1, # véletlen vertikális eltolás
        height_shift_range=0.1, # véletlen horizántális eltolás
        horizontal_flip=True, # tükrözés
        vertical_flip=False # tükrözés
    )
    datagen.fit(X_train) # datagen.flow() generálja az aktuális bemenetet
    model.fit_generator(datagen.flow(X_train, Y_train,
        batch_size=batch_size), samples_per_epoch=X_train.shape[0],
        nb_epoch=nb_epoch, validation_data=(X_test, Y_test))
  
```

Word embedding [Y. Bengio et al., 2006]

Nyelv modell:

Következő szó becslése az előzőekből vagy a kontextusából

Bemenet:

One-hot kódolás, a dimenzió maga a szótárméret

Ez az eredeti, de az újabb modellek már simítanak is

Érdekes hozadék [Mikolov et al. 2013]:

Király + nő -> közel a királynőhöz

-> [Rothe, Ebert & Schütze, 2016]:

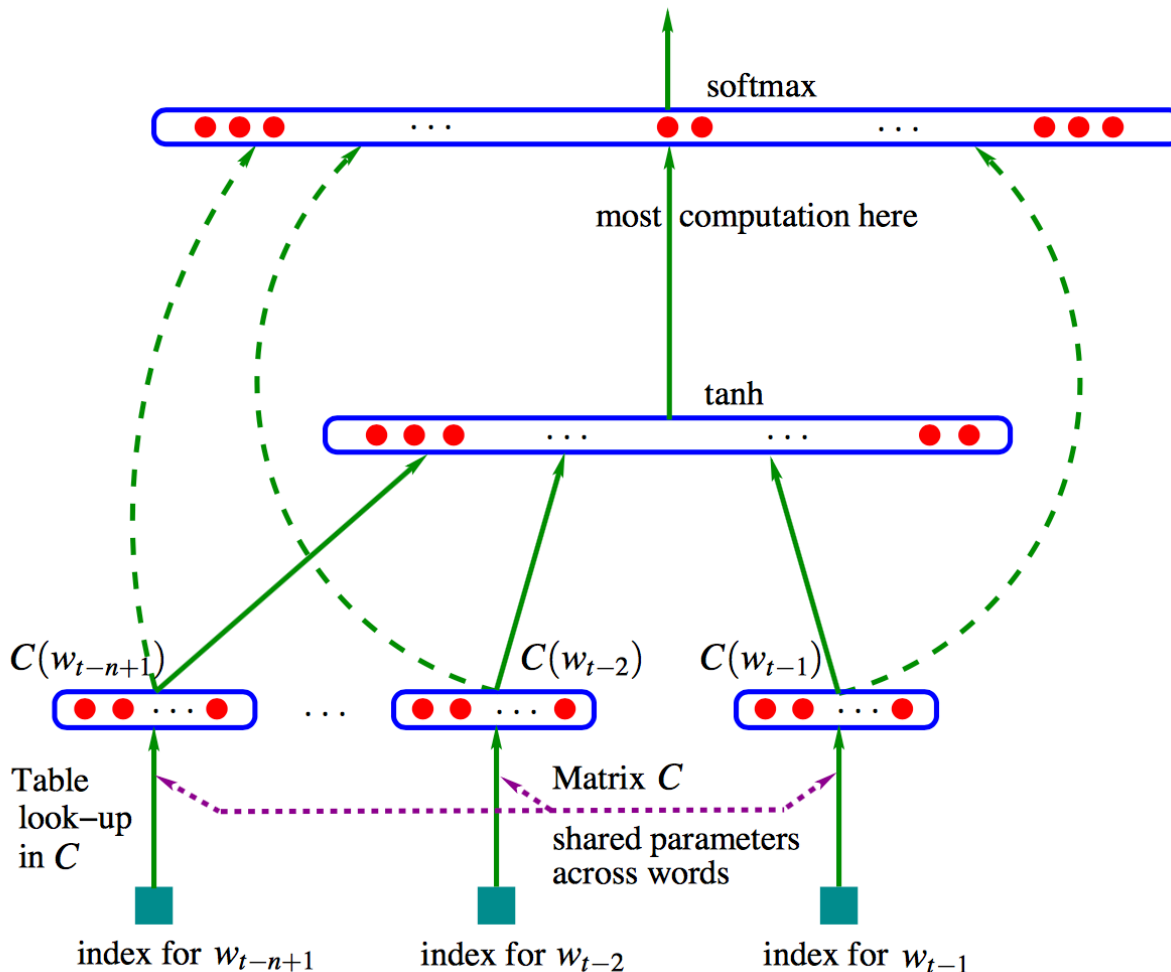
Ortogonális szó beágyazások

Polaritás

Image capt. : 512 dim emb.

Keras: embedding layer

$$i\text{-th output} = P(w_t = i \mid \text{context})$$



Objektum detektálás

Tradicionális modellek:

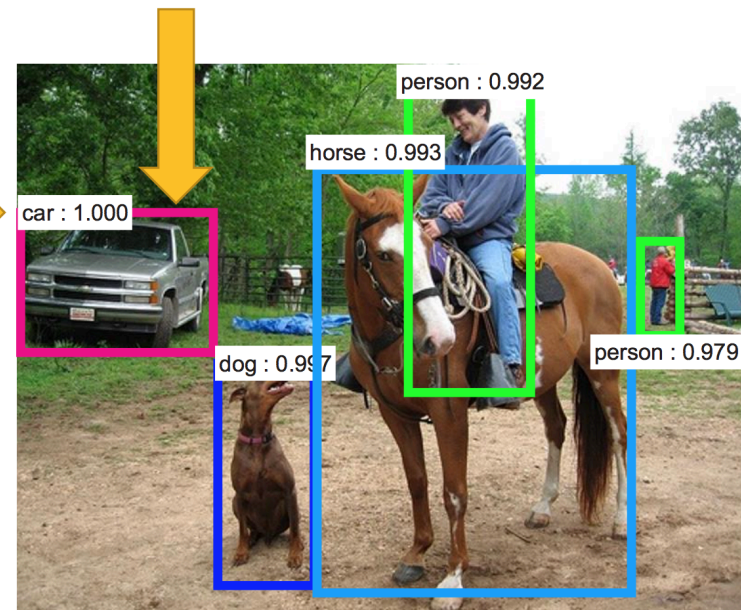
- Haar wavelet [Poggio et al., 1998] és Haar-like features [Viola and Jones, 2001]
Rigid features + SVM vagy AdaBoost
- Deformable parts modell [Felzenswalb et al. 2010]
- 100 Hz: HOG + Boosted Trees [Benenson et al. 2012]

Nem rigid modellek (CNN) :

- R-CNN [Girshick et al., 2014]:
regiók felett CNN
- SPP-net [He et al., 2014]:
Spatial Pyramid Pooling és DNN
- Fast R-CNN [Girschick et al., 2015]:
CNN leképezések
- RegionLet [Wang et al., 2015]:
integrál kép és CNN
- Faster R-CNN [Ren et al., 2015]:
Region Proposal Network

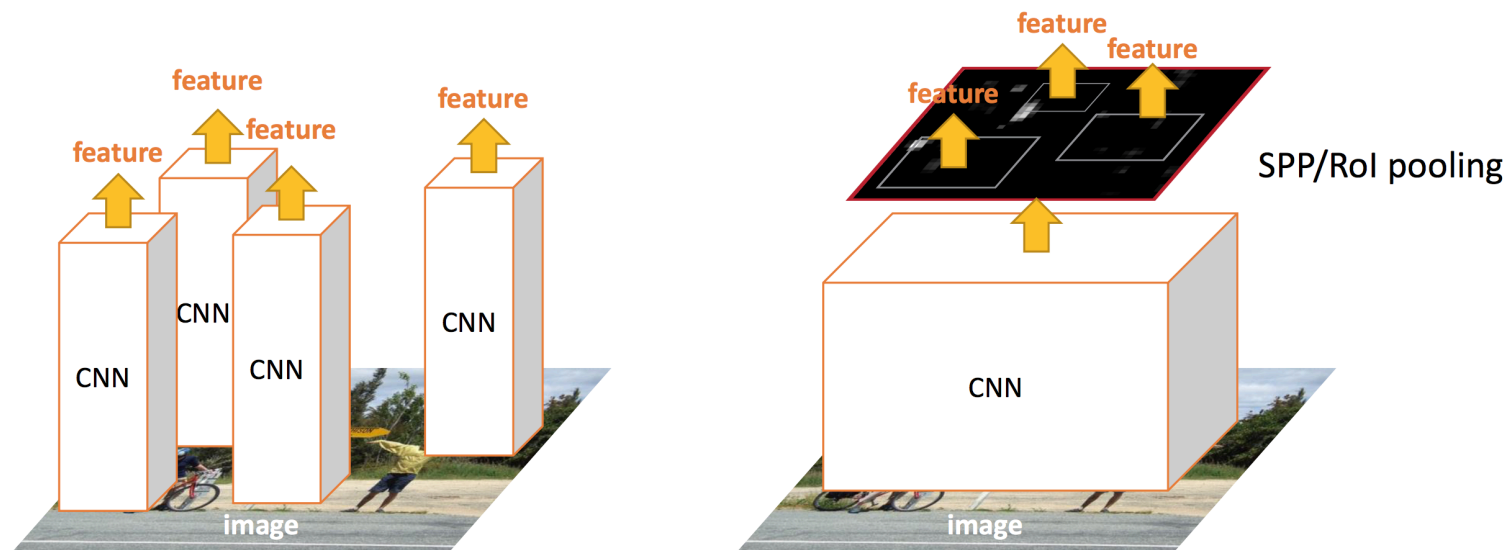
Recognition
What?

Localization
Where?



Kép: Kaiming He

R-CNN és Fast R-CNN

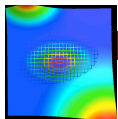


CNN a jelölt területeken
 Rigid méretű területek
 Nagy komplexitás (3-4 sec)
 SVM a tetején
 Sajnos nem CDF 😞

Egyetlen CNN
 Minden pixel egy aktivitás (filters/pixel)
 Bármekkora méretű területeket kezel
 Sokkal gyorsabb (1-1.5 sec)
 De még mindig négyzetes: SVM a jelölt,
 de transzformált területeken
 S nem CDF 😞

Fig. Kaiming He

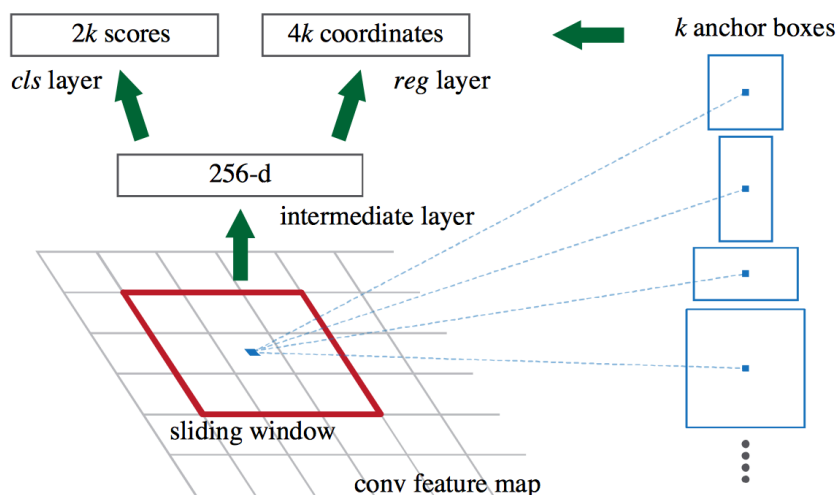
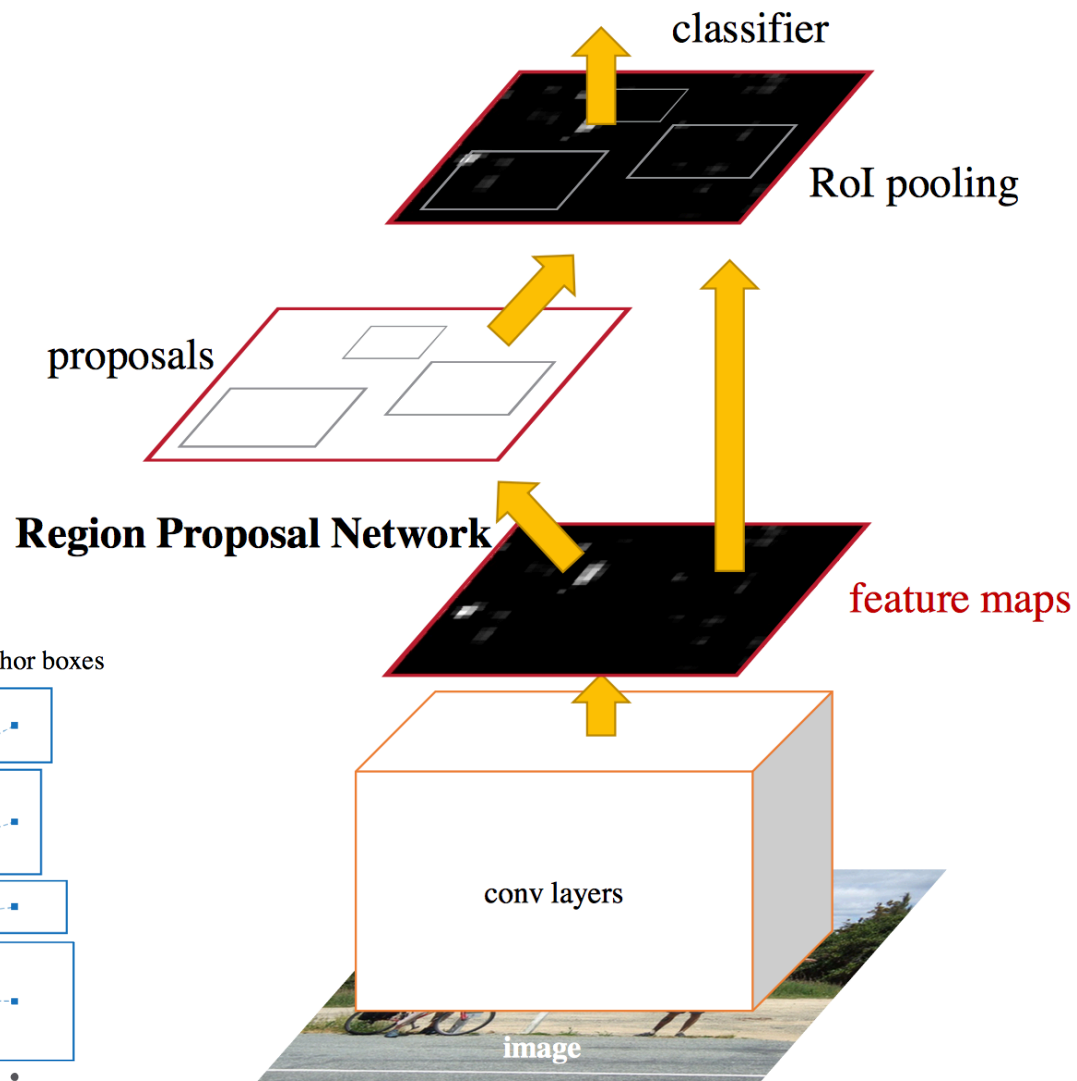
10/04/2017



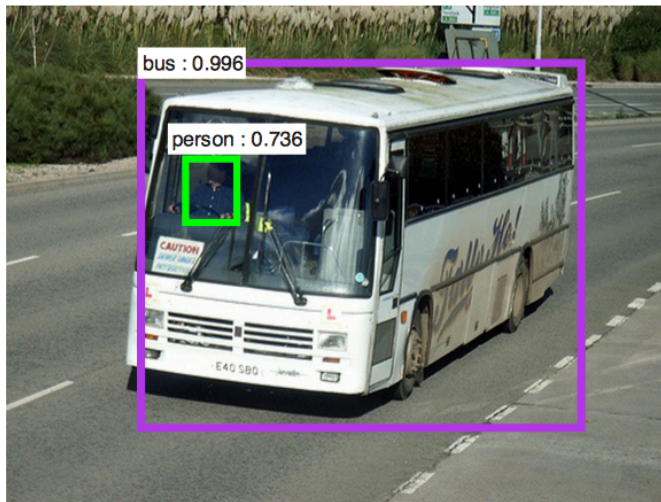
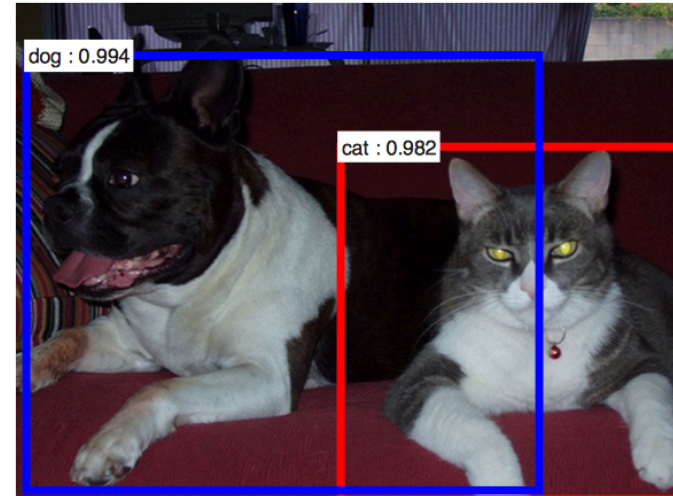
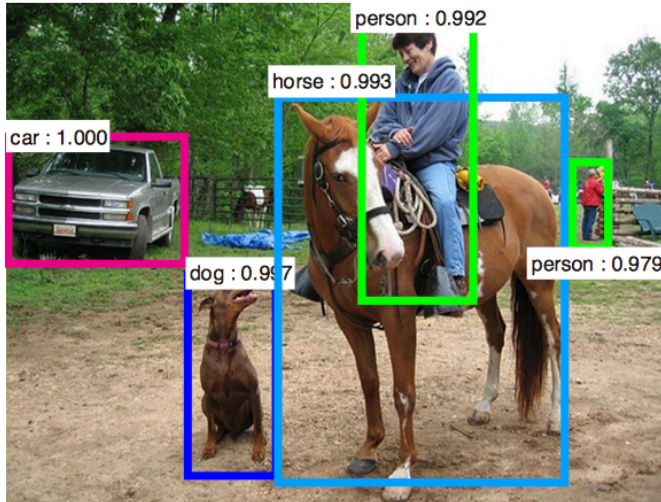
Faster R-CNN [Ren et al., 2015]

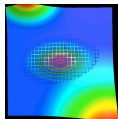
Komplex modellek helyett:

- Region proposal:
különböző méretű de téglalap területek
- Még mindig rigid
- End-to-end optimalizáció: CDF 😊
- Nagyon gyors <200ms



Faster R-CNN [Ren et al., 2015]





Apriori és asszociációs szabályok

Asszociációs szabály

$X \rightarrow Y$, ahol X és Y elemhalmazok

Például:

$\{\text{tej, pelenka}\} \rightarrow \{\text{sör}\}$

Mértékek

Support (s): s előfordulása

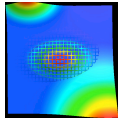
Ha $s = \{X, Y\}$, a közös előfordulás

Confidence (c): egy adott szabály bizonyossága

Pl. $c = X \rightarrow Y$

milyen gyakran kerül Y a kosárba,
ha X bekerült

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke



Apriori

Apriori elv: ha egy elemhalmaz (itemset) gyakori, akkor minden részhalmaza szükségképpen gyakori

Fordítva?

Support: Anti-monoton!

$$\text{supp}(\{\text{kenyér}\})=0.8$$

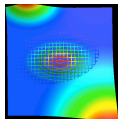
$$\text{supp}(\{\text{tej}\})=0.8$$

$$\text{supp}(\{\text{kenyér}, \text{tej}\})=0.6$$

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke

“Brute force”:

$$\#\text{elemhalmazok} = \sum_{l=1}^K \binom{K}{l}$$



Apriori

Apriori elv: ha egy elemhalmaz (itemset) gyakori, akkor minden részhalmaza szükségképpen gyakori

Fordítva?

Support: Anti-monoton!

$$\text{supp}(\{\text{kenyér}\})=0.8$$

$$\text{supp}(\{\text{tej}\})=0.8$$

$$\text{supp}(\{\text{kenyér}, \text{tej}\})=0.6$$

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke

$$\text{Brute force: } \# \text{elemhalmazok} = \sum_{l=1}^K \binom{K}{l}$$

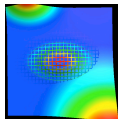
$$\#1\text{-elemű} = \binom{6}{1} = 6$$

$$\#2\text{-elemű} = \binom{6}{2} = 15$$

$$\#3\text{-elemű} = \binom{6}{3} = 20$$

$$\#4\text{-elemű} = \binom{6}{4} = 15$$

$$\#5\text{-elemű} = \binom{6}{5} = 6$$



Apriori

Apriori elv:

csak gyakori elemhalmazok uniója lehet gyakori
(legyen a minimum támogatottság=3)

1-elemű halmazok:

$\text{supp}(\{\text{sör}\})=?$

$\text{supp}(\{\text{kenyér}\})=?$

$\text{supp}(\{\text{coke}\})=?$

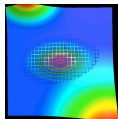
$\text{supp}(\{\text{pelenka}\})=?$

$\text{supp}(\{\text{tojás}\})=?$

$\text{supp}(\{\text{tej}\})=?$

$$\#1\text{-elemű} = \binom{6}{1} = 6$$

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke



Apriori

Apriori elv:

csak gyakori elemhalmazok uniója lehet gyakori
(legyen a minimum támogatottság=3)

1-elemű halmazok:

$\text{supp}(\{\text{sör}\})=3$

$\text{supp}(\{\text{kenyér}\})=4$

$\text{supp}(\{\text{coke}\})=2$

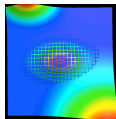
$\text{supp}(\{\text{pelenka}\})=4$

$\text{supp}(\{\text{tojás}\})=1$

$\text{supp}(\{\text{tej}\})=4$

$$\#1\text{-elemű} = \binom{6}{1} = 6$$

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke



Apriori

Apriori elv:

csak gyakori elemhalmazok uniója lehet gyakori
(legyen a minimum támogatottság=3)

1-elemű halmazok:

$\text{supp}(\{\text{sör}\})=3$

$\text{supp}(\{\text{kenyér}\})=4$

$\text{supp}(\{\text{coke}\})=2$

$\text{supp}(\{\text{pelenka}\})=4$

$\text{supp}(\{\text{tojás}\})=1$

$\text{supp}(\{\text{tej}\})=4$

$$\#1\text{-elemű} = \binom{6}{1} = 6$$

Lehetséges 2-elemű halmazok:

$\text{supp}(\{\text{sör}, \text{kenyér}\})=?$

$\text{supp}(\{\text{sör}, \text{pelenka}\})=?$

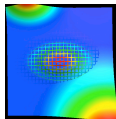
$\text{supp}(\{\text{sör}, \text{tej}\})=?$

$\text{supp}(\{\text{kenyér}, \text{pelenka}\})=?$

$\text{supp}(\{\text{kenyér}, \text{tek}\})=?$

$\text{supp}(\{\text{pelenka}, \text{tej}\})=?$

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke



Apriori

Apriori elv:

csak gyakori elemhalmazok uniója lehet gyakori
(legyen a minimum támogatottság=3)

1-elemű halmazok:

Lehetséges 2-elemű halmazok:

$\text{supp}(\{\text{sör}\})=3$

$\text{supp}(\{\text{kenyér}\})=4$

$\text{supp}(\{\text{coke}\})=2$

$\text{supp}(\{\text{pelenka}\})=4$

$\text{supp}(\{\text{tojás}\})=1$

$\text{supp}(\{\text{tej}\})=4$

$\text{supp}(\{\text{sör}, \text{kenyér}\})=2$

$\text{supp}(\{\text{sör}, \text{pelenka}\})=3$

$\text{supp}(\{\text{sör}, \text{tej}\})=2$

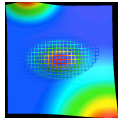
$\text{supp}(\{\text{kenyér}, \text{pelenka}\})=3$

$\text{supp}(\{\text{kenyér}, \text{tek}\})=3$

$\text{supp}(\{\text{pelenka}, \text{tej}\})=3$

$$\#1\text{-elemű} = \binom{6}{1} = 6$$

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke



Apriori

Apriori elv:

csak gyakori elemhalmazok uniója lehet gyakori
(legyen a minimum támogatottság=3)

1-elemű halmazok:

$\text{supp}(\{\text{sör}\})=3$

$\text{supp}(\{\text{kenyér}\})=4$

$\text{supp}(\{\text{coke}\})=2$

$\text{supp}(\{\text{pelenka}\})=4$

$\text{supp}(\{\text{tojás}\})=1$

$\text{supp}(\{\text{tej}\})=4$

$$\#1\text{-elemű} = \binom{6}{1} = 6$$

Lehetséges 2-elemű halmazok:

$\text{supp}(\{\text{sör}, \text{kenyér}\})=2$

$\text{supp}(\{\text{sör}, \text{pelenka}\})=3$

$\text{supp}(\{\text{sör}, \text{tej}\})=2$

$\text{supp}(\{\text{kenyér}, \text{pelenka}\})=3$

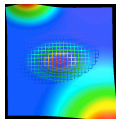
$\text{supp}(\{\text{kenyér}, \text{tek}\})=3$

$\text{supp}(\{\text{pelenka}, \text{tej}\})=3$

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke

Lehetséges 3-elemű halmazok:

$\text{supp}(\{\text{kenyér}, \text{pelenka}, \text{tej}\})=2$



Apriori

Apriori elv:

csak gyakori elemhalmazok uniója lehet gyakori
(legyen a minimum támogatottság=3)

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör

1-elemű halmazok:

`supp({sör})`

`supp({kenyér})`

`supp({coke})`

`supp({pelenka})`

`supp({tojás})`

`supp({tej})=?`

Hogyan generáljunk hatékonyan jelölt halmazokat?

Duplikátumok?

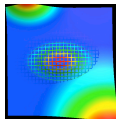
Támogatottság számolása?

3-elemű

`supp({kenyér, pelenka})=?`

`supp({kenyér, tej})=?`

`supp({pelenka, tej})=?`



Apriori

Apriori elv:

csak gyakori elemhalmazok uniója lehet gyakori
(legyen F_k a k -gyakori elemhalmazok halmaza)

Hogyan generáljunk hatékonyan jelölt halmazokat?
Duplikátumok?

1-elemű

→ $F_{k-1} \times F_{k-1}$ módszer:

csak olyan $k-1$ elemű gyakori elemhalmazokat egyesítsünk melyek $k-2$ eleme megegyezik

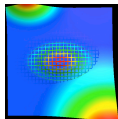
Támogatottság számolása?

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	kenyér, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke

éges 3-elemű halmazok:

$\text{supp}(\{\text{sör, pelenka, tej}\}) = ?$

$\text{supp}(\{\text{pelenka, tej}\}) = ?$



Asszociációs szabályok

Adott: egy gyakori elemhalmaz

Példa:

$I = \{\text{kenyér, pelenka, tej}\}$

Diszjunkt elemhalmaz

$X = \{\text{kenyér, pelenka}\}$

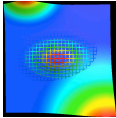
$Y = \{\text{tej}\}$

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke

$$\text{conf}(X \rightarrow Y) = \frac{\sigma(X, Y)}{\sigma(X)} = \frac{\sigma(\{\text{kenyér, pelenka, tej}\})}{\sigma(\{\text{kenyér, pelenka}\})} = \frac{2}{3}$$

$$\text{lift}(Y | X) = \frac{P(X, Y)}{P(Y)P(X)} = \frac{P(Y | X)}{P(Y)} = \frac{0.667}{0.8} = 0.833$$

Negatívan csatolt!



Asszociációs szabályok

Egyéb diszjunkt részhalmazok

{sör, pelenka}

{kenyér, pelenka}

{kenyér, tej}

{pelenka, tej}

{kenyér, pelenka, tej}

Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke

$$\text{conf}(sör \rightarrow pelenka) = \frac{\sigma(sör, pelenka)}{\sigma(sör)} = \frac{3}{3}$$

$$\text{conf}(pelenka \rightarrow sör) = \frac{\sigma(sör, pelenka)}{\sigma(pelenka)} = \frac{3}{4}$$

$$\text{conf}(kenyér \rightarrow pelenka) = \frac{\sigma(kenyér, pelenka)}{\sigma(kenyér)} = \frac{3}{4}$$

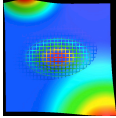
$$\text{conf}(pelenka \rightarrow kenyér) = \frac{\sigma(kenyér, pelenka)}{\sigma(pelenka)} = \frac{3}{4}$$

$$\text{conf}(kenyér \rightarrow tej) = \frac{\sigma(kenyér, tej)}{\sigma(kenyér)} = \frac{3}{4}$$

$$\text{conf}(tej \rightarrow kenyér) = \frac{\sigma(kenyér, tej)}{\sigma(tej)} = \frac{3}{4}$$

$$\text{conf}(pelenka \rightarrow tej) = \frac{\sigma(pelenka, tej)}{\sigma(pelenka)} = \frac{3}{4}$$

$$\text{conf}(tej \rightarrow pelenka) = \frac{\sigma(pelenka, tej)}{\sigma(tej)} = \frac{3}{4}$$



Asszociációs szabályok

Egyéb diszjunkt részhalmazok

{sör, pelenka}

{kenyér, pelenka}

{kenyér, tej}

{pelenka, tej}

{kenyér, pelenka, tej}

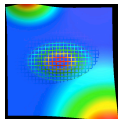
Kosár	Tartalom
1	kenyér, tej
2	kenyér, pelenka, sör, tojás
3	tej, pelenka, sör, coke
4	kenyér, tej, pelenka, sör
5	kenyér, tej, pelenka, coke

$$\text{lift}(sör \rightarrow pelenka) = \text{lift}(pelenka \rightarrow sör) = \frac{5}{4}$$

$$\text{lift}(kenyér \rightarrow tej) = \text{lift}(tej \rightarrow kenyér) = \frac{15}{16}$$

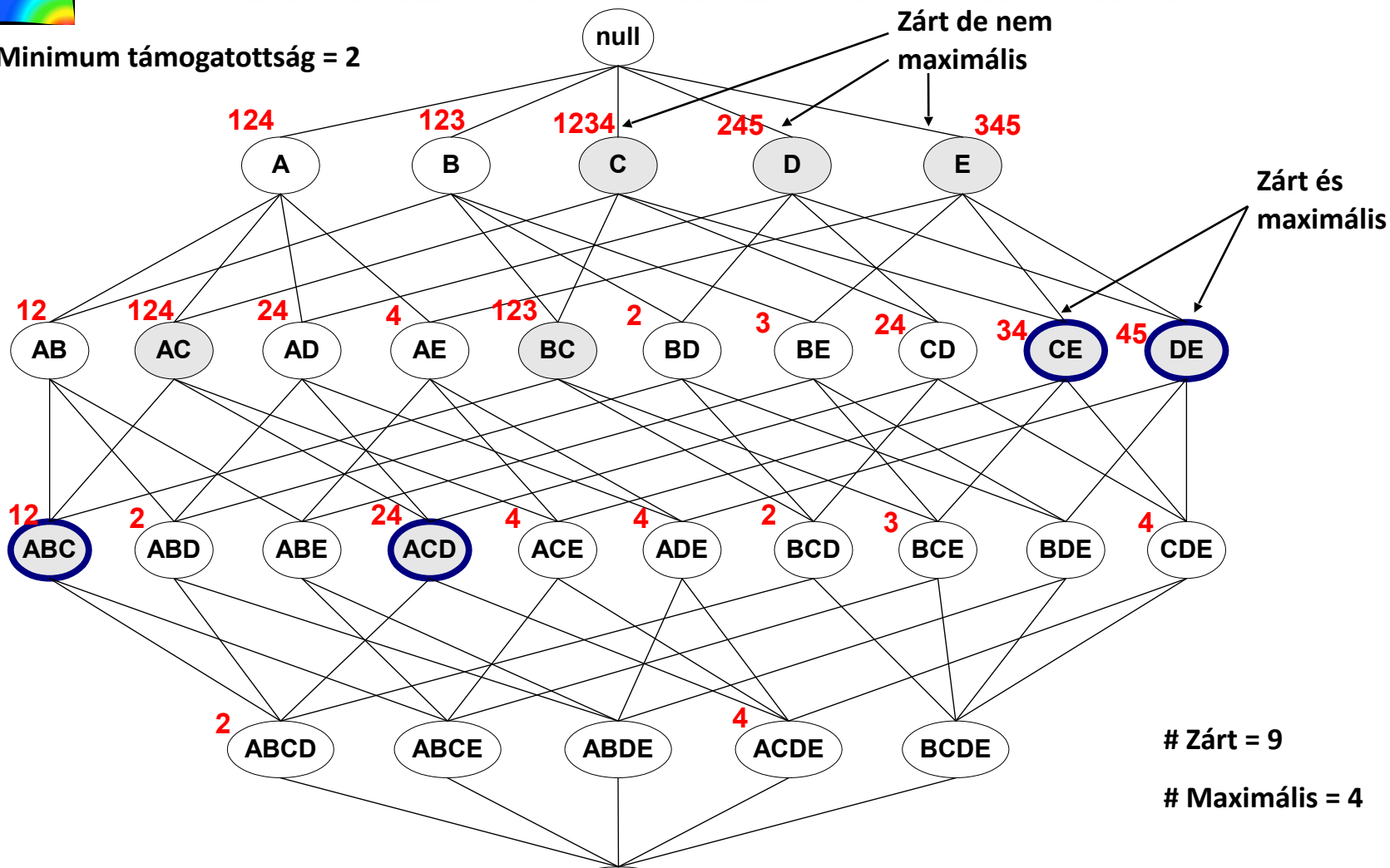
$$\text{lift}(kenyér \rightarrow pelenka) = \text{lift}(pelenka \rightarrow kenyér) = \frac{15}{16}$$

$$\text{lift}(pelenka \rightarrow tej) = \text{lift}(tej \rightarrow pelenka) = \frac{15}{16}$$



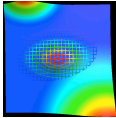
Minimum támogatottság = 2

Zárt és maximális elemhalmazok



Zárt: egy elemhalmaz zárt, ha egyik őt tartalmazó elemhalmaz számossága sem egyezik az elemhalmaz számosságával

Maximális: Egy elemhalmaz maximális, ha egyik őt tartalmazó elemhalmaz sem gyakori.



Zárt és maximális elemhalmazok

Transaction	
ID1	{a,b,d}
ID2	{b,c,d}
ID3	{a,c}
ID4	{a,c,d}

1-elemű halmazok száma: 4

Minimum támogatottság: 2

Legnagyobb elemszámú halmaz?

Mik a gyakori elemhalmazok?

Zárt és maximális elemhalmazok?

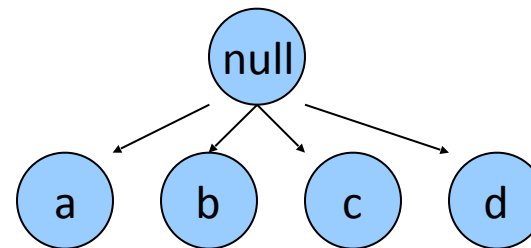
APRIORI:

Órai feladat 1:

Mennyire hatékony ha

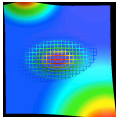
a) gyors a cpu/io-nk, de kevés a memória?

b) lassú a cpu/io-nk, de sok a memória?



Zárt: egy elemhalmaz zárt, ha egyik őt tartalmazó elemhalmaz számossága sem egyezik az elemhalmaz számosságával

Maximális: Egy elemhalmaz maximális, ha egyik őt tartalmazó elemhalmaz sem gyakori.

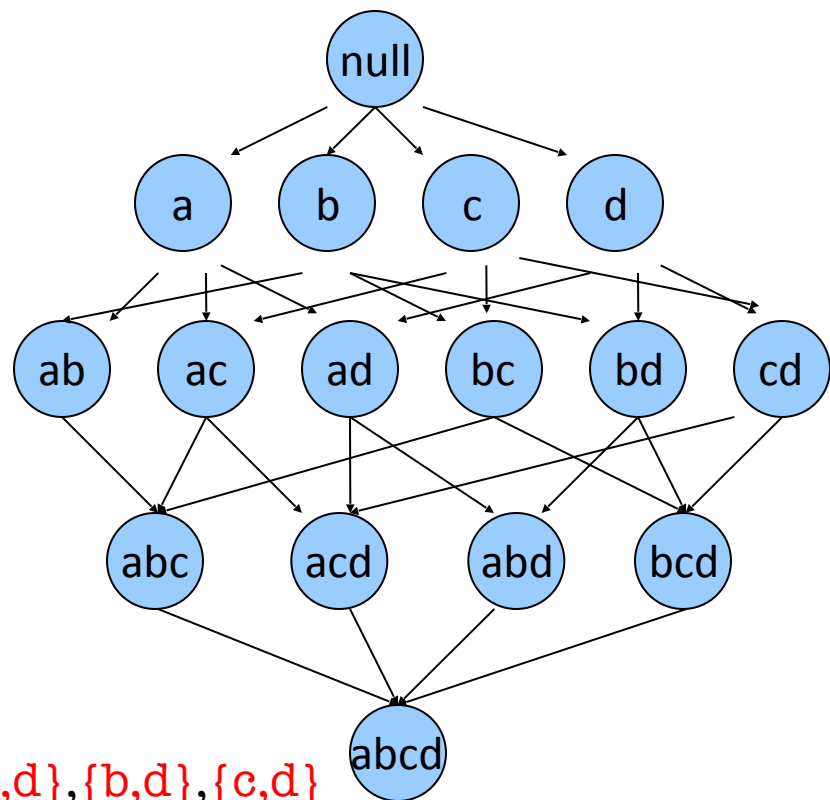


Zárt és maximális elemhalmazok

Transaction	
ID1	{a,b,d}
ID2	{b,c,d}
ID3	{a,c}
ID4	{a,c,d}

1-elemű halmazok száma: 4

Minimum támogatottság: 2

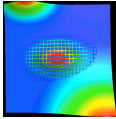


Gyakori elemhalmazok:

{a},{b},{c},{d},{a,c},{a,d},{b,d},{c,d}

Zárt elemhalmazok: {a},{c},{d},{a,c},{a,d},{b,d},{c,d}

Maximális elemhalmazok: {a,c},{a,d},{b,d},{c,d}



Apriori pl. weka-ban

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Associator

Choose Apriori -N 10 -T 1 -C 1.1 -D 0.05 -U 1.0 -M 0.01 -S -1.0 -Z -c -1

Start Stop

Result list (right-click for details)

- 22:42:35 - Apriori
- 22:43:44 - Apriori
- 22:43:56 - Apriori
- 22:46:01 - Apriori
- 22:46:31 - Apriori
- 22:46:40 - Apriori

Associator output

```

children
car
save_act
current_act
mortgage
pep
=== Associator model (full training set) ===

Apriori
=====
Minimum support: 0.25 (150 instances)
Minimum metric <lift>: 1.1
Number of cycles performed: 15

Generated sets of large itemsets:

Size of set of large itemsets L(1): 10
Size of set of large itemsets L(2): 17
Size of set of large itemsets L(3): 6

Best rules found:
1. married=YES save_act=YES 277 ==> pep=NO 175   conf:(0.63) < lift:(1.16)> lev:(0.04) [24] conv:(1.23)
2. pep=NO 326 ==> married=YES save_act=YES 175   conf:(0.54) < lift:(1.16)> lev:(0.04) [24] conv:(1.15)
3. age='(50.666667-inf)' 191 ==> save_act=YES 151   conf:(0.79) < lift:(1.15)> lev:(0.03) [19] conv:(1.44)
4. save_act=YES 414 ==> age='(50.666667-inf)' 151   conf:(0.36) < lift:(1.15)> lev:(0.03) [19] conv:(1.07)
5. married=YES 396 ==> save_act=YES pep=NO 175   conf:(0.44) < lift:(1.13)> lev:(0.03) [19] conv:(1.09)
6. save_act=YES pep=NO 235 ==> married=YES 175   conf:(0.74) < lift:(1.13)> lev:(0.03) [19] conv:(1.31)
7. married=YES 396 ==> pep=NO 242   conf:(0.61) < lift:(1.12)> lev:(0.04) [26] conv:(1.17)
8. pep=NO 326 ==> married=YES 242   conf:(0.74) < lift:(1.12)> lev:(0.04) [26] conv:(1.3)
9. married=YES current_act=YES 293 ==> pep=NO 177   conf:(0.6) < lift:(1.11)> lev:(0.03) [17] conv:(1.14)
10. pep=NO 326 ==> married=YES current_act=YES 177   conf:(0.54) < lift:(1.11)> lev:(0.03) [17] conv:(1.11)
  
```

Status OK

Log x 0

$$\text{conf}(Y|X) = \frac{P(X, Y)}{P(X)}$$

$$\text{lift}(Y|X) = \frac{P(X, Y)}{P(Y)P(X)} = \frac{P(Y|X)}{P(Y)}$$

$$\text{lev}(Y|X) = P(X, Y) - P(Y)P(X)$$

$$\text{conv}(Y|X) = \frac{1 - \text{supp}(Y)}{1 - \text{conf}(X \rightarrow Y)}$$