Osztályozási feladatok képdiagnosztikában

Orvosi képdiagnosztikai 2018 ősz

Osztályozás

- Szeparáló felületet keresünk
- Leképezéseket tanulunk meg azok mintáiból
  - A tanuláshoz használt minták a tanító minták / pontok
  - A leképezés folytonos / diszkrét bementből képez diszkrét és véges számú halmazba

    $$f\left(\mathbf{x_i}\right) = d_i \ , \ d_i \in \left\{1, 2, \ldots, k\right\} \ : \ \forall i \in \left\{1, 2, ..., P\right\}$$
  - Feltételezhetünk még bemeneti / kimeneti zajt:

    $$f\left(\mathbf{x_i} + \varepsilon_{b,i}\right) + \varepsilon_{k,i} = d_i$$
  - $f\left(\cdot\right)$ szerint lehet lineáris / paramétereiben lineáris / nemlineáris

Terminológia, jelölések

- Osztályozó:
  - $f : \mathrm{R}^N \times \Theta \to \{1, 2, \ldots, d\}$
- Veszteségfüggvény (loss function, hibafüggvény)
  - $L(d, y) \propto -\log\left(p(y \mid d)\right)$
  - általában közvetlenül definiáljuk *p* kihagyásával
- Tanítás: $\min. \quad R(\mathbf{w}) = \underset{(\mathbf{x},d)\in D}{\mathrm{E}} \left\{ L\left(d, f(\mathbf{x}, \mathbf{w})\right) \right\}$

  $s.t. \qquad \mathbf{w} \in \Theta$
- Tapasztalati kockázat:
  - $R_{emp}(\mathbf{w}) = \sum_i L\left(d_i, f(\mathbf{x_i}, \mathbf{w})\right)$

Orvosi Döntéstámogató Rendszerek

- Computer Aided Detection
  - Cél a leletezések specificitásának, és érzékenységének a növelése
  - Számos megbetegedés esetén már jóval annak diagnosztizálása előtt láthatóak bizonyos elváltozások
  - Megvalósításuk tanuló rendszerrel történik
    - Klasszikus megközelítés: probléma specifikus jel / képfeldolgozás állítja elő az osztályozás bemeneteit
    - Újabb megközelítés: kimarad a szakértői jelfeldolgozás, az MI-re bízzuk a teljes folyamatot

Empirikus kockázatminimalizálás

- Elv: a tapasztalati kockázat (tanítóminták osztályozási hibájának) minimalizálásával minimalizálható a kockázat (valódi minták osztályozási hibája), ha közben rögzítjük az osztályozó VC-dimenzióját:
- Osztályozó hibája $1-\eta$ valószínűséggel

$$R(\mathbf{w}) \le R_{emp}(\mathbf{w}) + \frac{\varepsilon(h)}{2}\sqrt{1+\frac{4R_{emp}(\mathbf{w})}{\varepsilon(h)}}$$

$$\varepsilon(h) = 4\frac{h(\ln(2l/h)+1) - \ln(\eta/4)}{l}$$

l: tanítópontok száma
h: osztályozó VC dimenziója (osztályozó komplexitása)
$\hat{f}(\cdot)$: megtanult leképezés

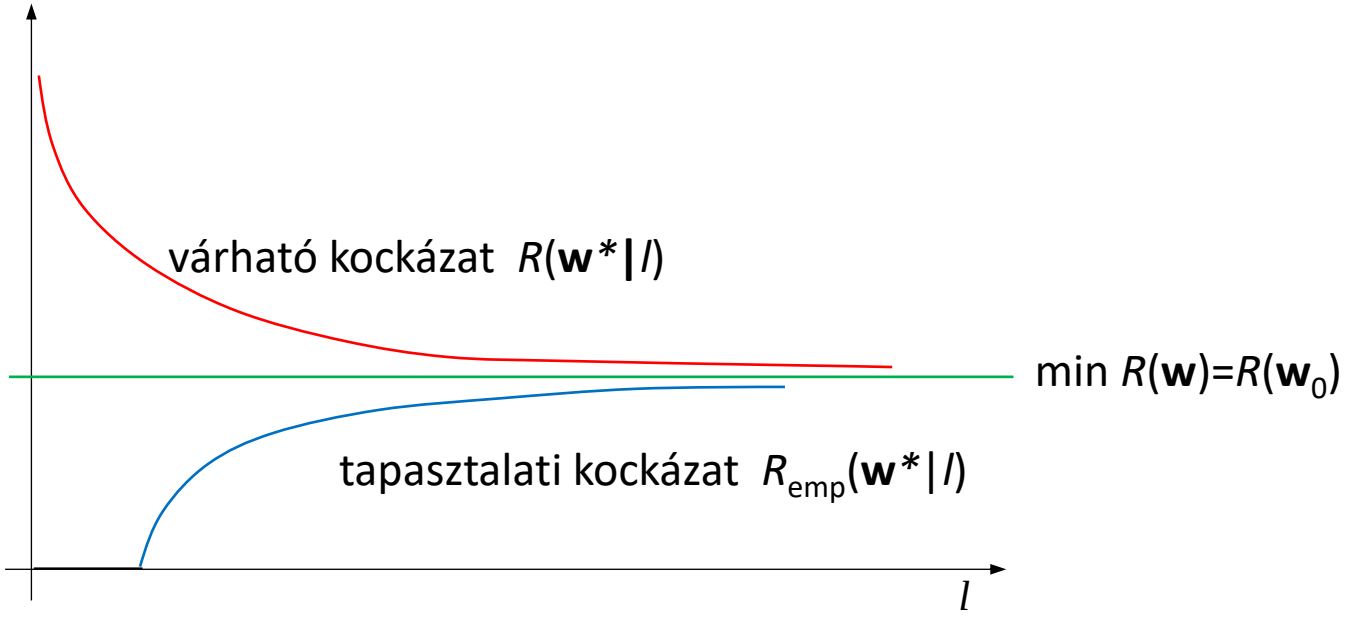
D: mintákat generáló eloszlás

Empirikus kockázatminimalizálás

- Fontos, hogy ugyanazon háttéreloszlás mintái adják az éles (valódi) és a tanítómintákat
- VC dimenzió:
  - Maximum hány olyan tetszőleges elrendezésű és osztályozású mintapont létezik, melyet az osztályozó képes hibátlanul (adott osztályozás szerint) szeparálni.
- Cél a kockázat ($R(\mathbf{w})$) minimalizálása – inverz probléma
  - Azon belül is leginkább MAP becslés
  - Likelihood tag: tanítóminták osztályozási hibáját bünteti
  - Regularizációs tag: osztályozó VC dimenzióját bünteti

Osztályozók általánosítási hibája

- Lényegében $R(\mathbf{w}) - R_{emp}(\mathbf{w})$:
  - Tanítóminták számának növelésével csökken
  - VC dimenzió növekedésével nő



Példák osztályozókra

- Lineáris / Kernel diszkrimináns analízis
- Rosenbladt perceptron
- Adaline
- Neurális háló (MLP – multi layer perceptron)
- Bázisfüggvényes lineáris hálók (pl. RBF, CMAC)
- Mély neurális hálók
  - Konvolúciós neurális hálók

Lineáris (Fisher) diszkrimináns analízis

- Cél olyan vetítőirány keresése, mely mentén maximális

$$C(\mathbf{w}) = \frac{\left(\mathbf{w}^T\left(\mathbf{m_2} - \mathbf{m}_1\right)\right)^2}{\sum_{k\in\{1,2\}}\sum_{i\in P_k}\left(\mathbf{w}^T\left(\mathbf{x_i} - \mathbf{m_k}\right)\right)^2}$$

$$\mathbf{m_k} = \mathrm{E}_{i\in P_k}\left\{\mathbf{x_i}\right\}$$

  - Vezessük be az alábbi jelöléseket:

$$\mathbf{S_B} = \left(\mathbf{m_2} - \mathbf{m}_1\right)\cdot\left(\mathbf{m_2} - \mathbf{m}_1\right)^T$$

$$\mathbf{S_{W}}k = \sum_{i\in P_k}\left(\mathbf{x_i} - \mathbf{m_k}\right)\left(\mathbf{x_i} - \mathbf{m_k}\right)^T$$

$$\mathbf{S_W} = \sum_k \mathbf{S_{W}}k$$

Lineáris (Fisher) diszkrimináns analízis

- Egy Rayleigh hányadost minimalizálnunk

$$\arg\max\left\{\frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}}\right\}=\arg\max\left\{\frac{\mathbf{w}^T\mathbf{S}_W{}^{-1}\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{w}}\right\}$$

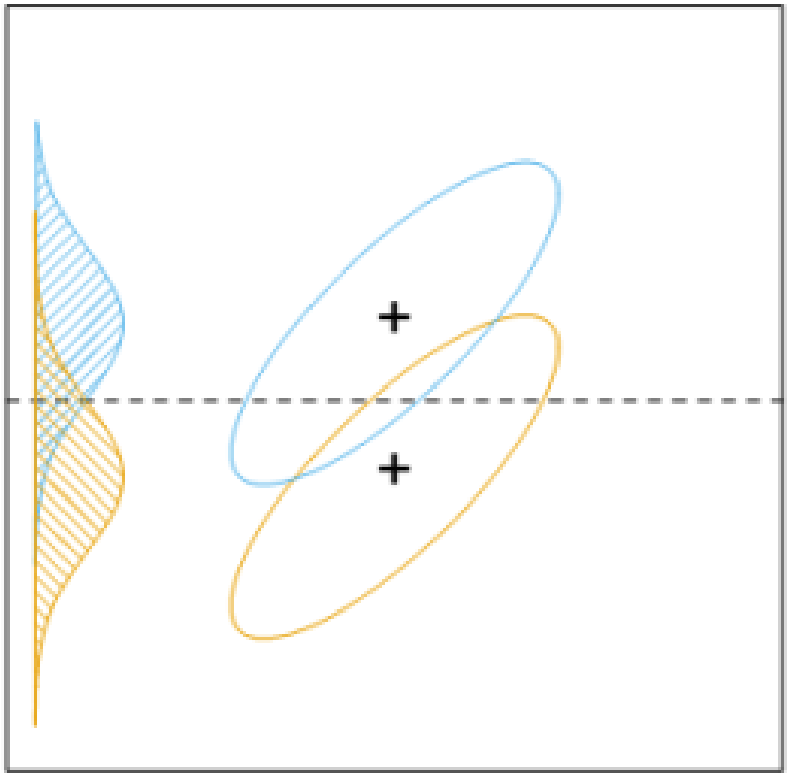
  - Ekvivalens optimalizálási probléma $\max. \quad \mathbf{w}^T\mathbf{S}_W{}^{-1}\mathbf{S}_B\mathbf{w}$

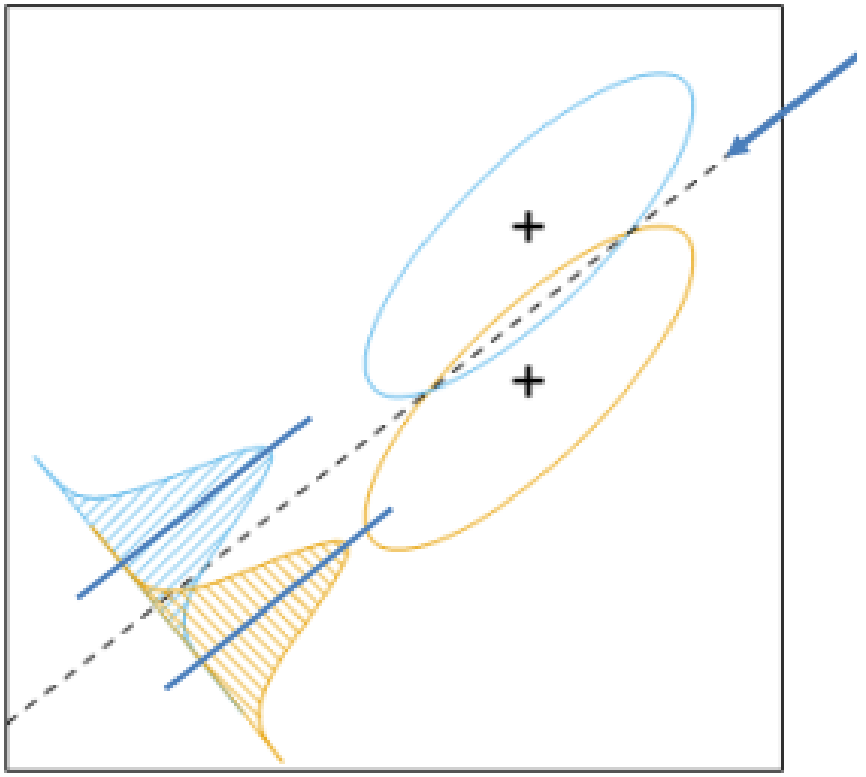
$$s.t. \qquad \mathbf{w}^T\mathbf{w}\le 1$$

  - Lagrange duális nyeregpontjában: $\mathbf{S}_W{}^{-1}\mathbf{S}_B\mathbf{w}=\lambda\mathbf{w}$
  - Tehát a legkisebb sajátérték, meg hozzátartozó sajátvektor a megoldás (gyakorlatban azonban nem szükséges a sajátérték egyenlet megoldása)

Lineáris (Fisher) diszkrimináns analízis

- Tekinthető a PCA osztályozási feladatokra adekvát változatának is – ez is dimenziót redukál





Adaptív lineáris neuron

- Négyzetes veszteségfüggvény alkalmazásával

$$\mathbf{y} = \mathbf{X}\mathbf{w}\ ; \quad \mathbf{X} = \begin{bmatrix} \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_n} \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T ; \quad R_{emp}\left(\mathbf{w}\right) = \left\|\mathbf{d} - \mathbf{X}\mathbf{w}\right\|_2^2$$

- Analitikus megoldás: $\mathbf{w}^* = \mathbf{X}^\dagger \mathbf{d} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{d}$
- Ha jobb általánosító képességet akarunk:
  - Csonkoljuk $\mathbf{X}^\dagger = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{U}^*$ esetén a szinguláris értékeket
  - Folytonos átmenete – Thikhonov regularizáció: $\mathbf{X}^\dagger = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T$

Adaptív lineáris neuron

- Iteratív optimalizáció – Least Mean Squares
  - Pillanatnyi négyzetes hiba:

$$C\left(\mathbf{w}\left(k\right)\right)=\varepsilon^{2}\left(k\right)=\left(d_{i(k)}-\mathbf{x_{i(k)}}^{T}\mathbf{w}\left(k\right)\right)^{2}$$

  - Sztochasztikus Gradiens optimalizáció – μ-LMS
    - Négyzetes hiba gradiensét a pillanatnyi négyzetes hiba gradiensével becsüljük:

$$\mathbf{w}\left(k+1\right)=\mathbf{w}\left(k\right)-\mu\nabla C\left(k\right)=\mathbf{w}\left(k\right)-2\mu\varepsilon\left(k\right)\mathbf{w}\left(\mathbf{k}\right)$$

  - SGD kondicionálással – α-LMS

$$\mathbf{w}\left(k+1\right)=\mathbf{w}\left(k\right)+\alpha\frac{\varepsilon\left(k\right)}{\mathbf{x}_{\mathbf{i}(\mathbf{k})}^{T}\mathbf{x}_{\mathbf{i}(\mathbf{k})}}\mathbf{x}_{\mathbf{i}(\mathbf{k})}\qquad 0<\alpha<2$$

Bázisfüggvényes osztályozók

- Problémák nagy része nem lineárisan szeparábilis.
- Paramétereiben lineáris, nemlineáris osztályozók:

$$f\left(\mathbf{x},\mathbf{w}\right)=\varphi\left(\mathbf{x}\right)^{T}\cdot\mathbf{w}=y$$

  - Lényegében $\varphi\left(\cdot\right)$(ú.n. jellemzőtérbe transzformáló leképezés) határozza meg, hogy mire képes az osztályozó
  - $\varphi\left(\cdot\right)$is tanulható (pl. OLS eljárás RBF hálók esetén)
  - Ha nem inkonzisztensek a tanítóminták, akkor mindig létezik olyan $\varphi\left(\cdot\right)$, mellyel lineárisan szeparábilis a probléma
  - Cserébe baj lehet a VC dimenzióval / háló általánosító képességével

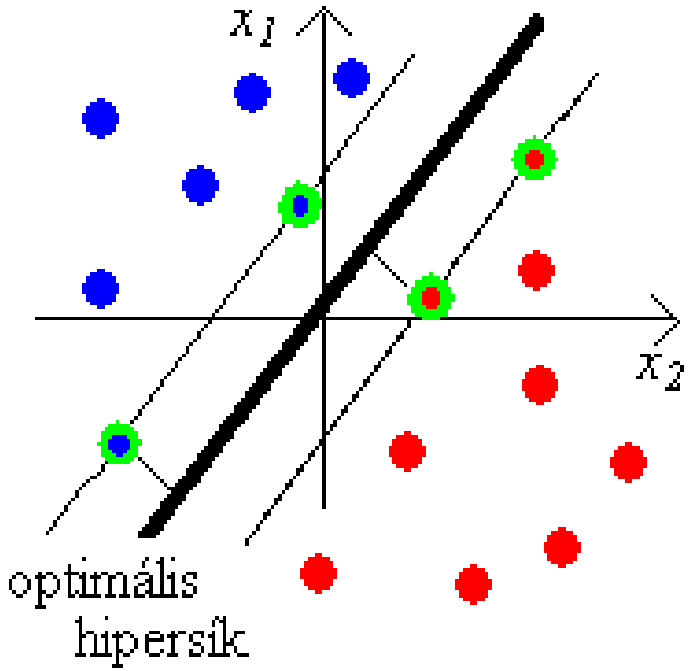
Szupport vektor gépek

- Keressük az alábbi szeparáló síkot:

$$\min. \quad 1/2\,\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

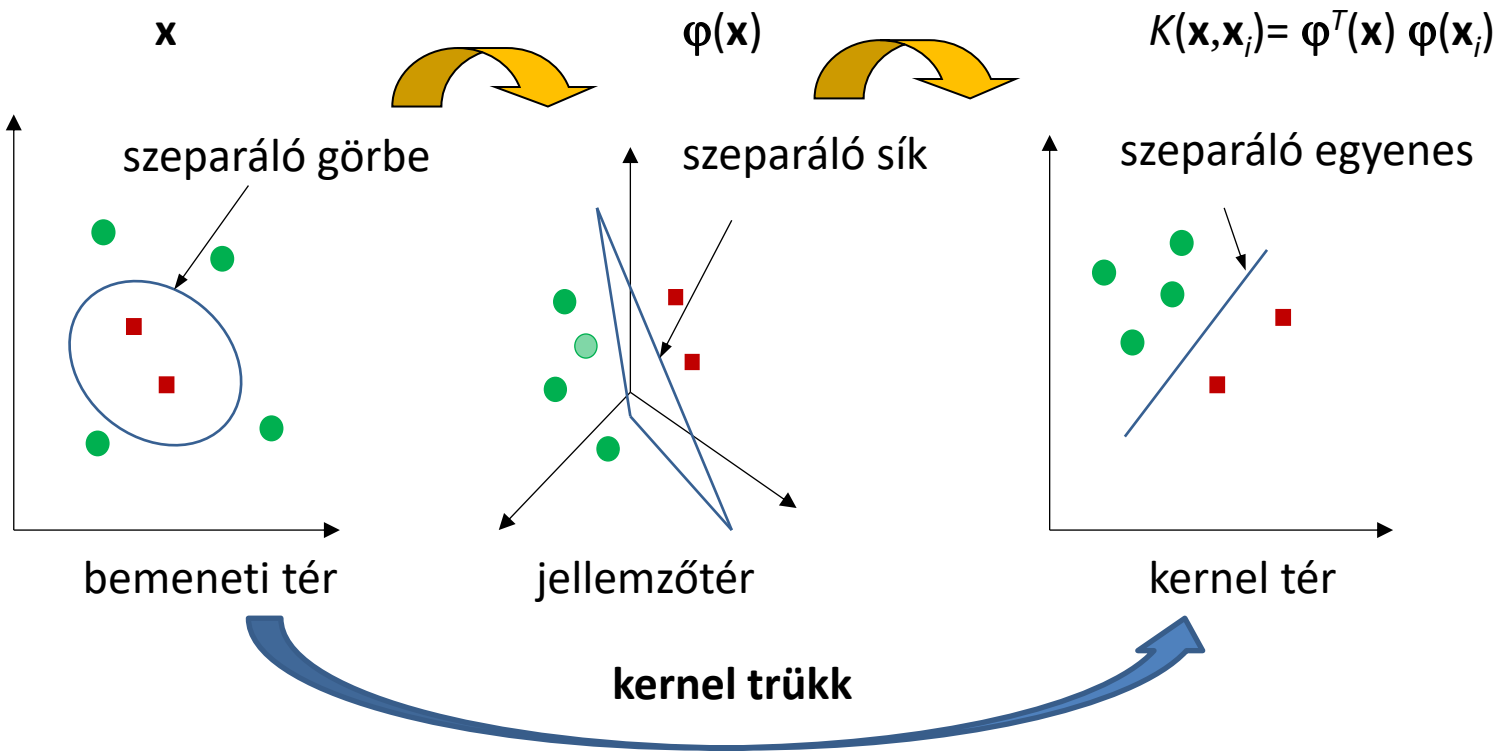
$$s.t. \quad d_i\left(\mathbf{w}^{\mathrm{T}}\varphi\left(\mathbf{x}_{\mathbf{i}}\right)+b\right)\geq 1$$

  - Lényegében Thikhonov regularizálunk (hasonlóan, ahogy azt az SART / SIRT-nél tettük)



- Lagrange duális alapján:
  - Megússzuk $\varphi(\cdot)$ definiálását
  - Helyette $K\left(\mathbf{x},\mathbf{x}_{\mathbf{i}}\right)=\varphi\left(\mathbf{x}\right)^{T}\varphi\left(\mathbf{x}_{\mathbf{i}}\right)$ úgynevezett kernelt kell definiálnunk (Kernel trükk)
  - Végtelen dimenziós φ is lehet véges VC dimenzió mellett

Szupport vektor gép kernel leképezése



$$y = \sum_{j=0}^{M} w_j \varphi_j(\mathbf{x}) = \mathbf{w}^{\mathbf{T}} \boldsymbol{\varphi}(\mathbf{x})$$

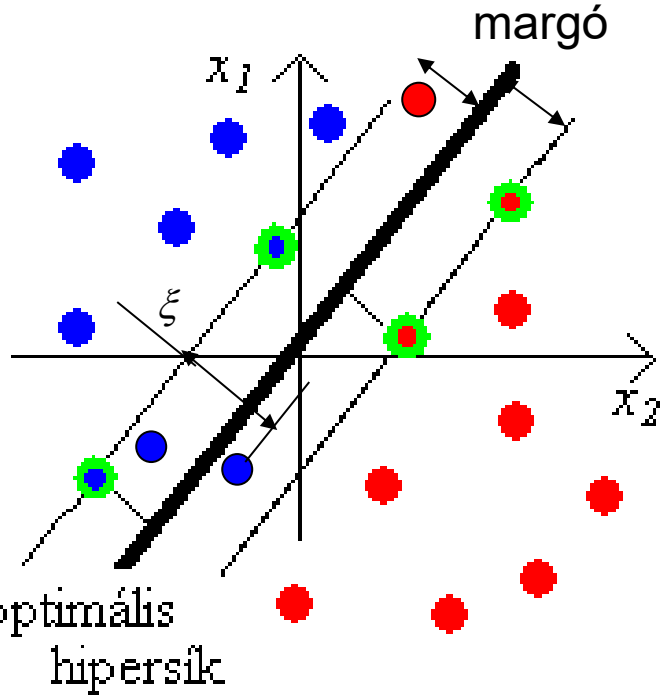
$$\mathbf{w} = [w_0, w_1, \ldots, w_M]^T \qquad \boldsymbol{\varphi} = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_M(\mathbf{x})]^T$$

Szupport vektor gépek - gyengítés

- Keressük az alábbi szeparáló síkot:

$$\min. \quad 1/2\,\mathbf{w}^{\mathbf{T}}\mathbf{w} + c\cdot\mathbf{1}^{T}\cdot\boldsymbol{\xi}$$

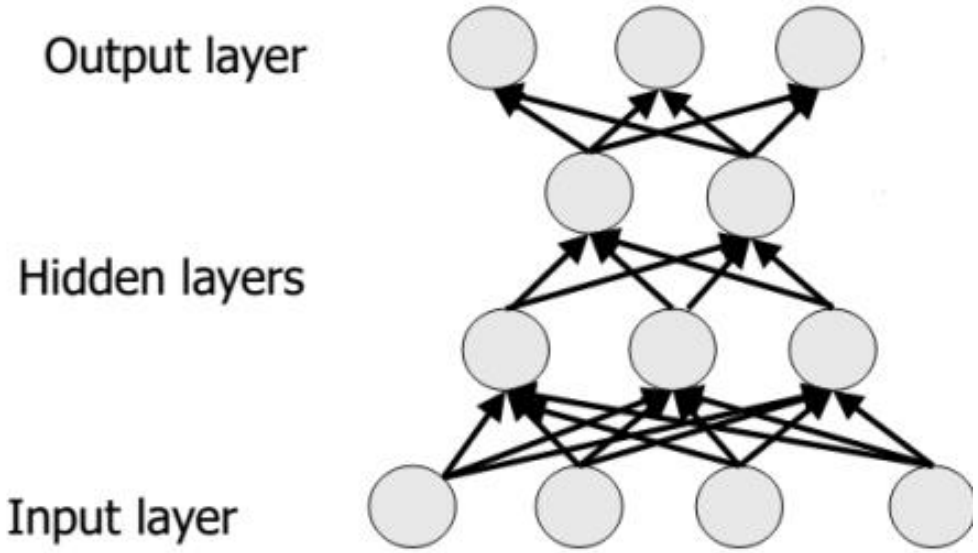
$$s.t. \quad d_i\left(\mathbf{w}^{\mathbf{T}}\varphi\left(\mathbf{x_i}\right)+b\right)\geq 1-\boldsymbol{\xi}_{(i)} \qquad \boldsymbol{\xi}\in\mathbb{R}_{+}^{|\mathrm{P}|}$$



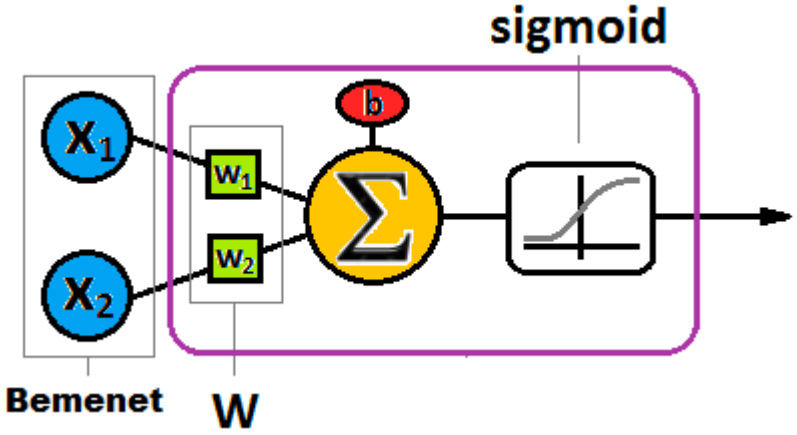
  - Megengedjük, hogy bizonyos minták belógjanak a margóba, esetleg rosszul osztályozzuk őket
  - *c* lényegében egy trade – off szabályozó az empirikus kockázat (likelihood tag, gyengítő változó), és a regularizációs ($\|\mathbf{w}\|_2^2$) tag súlya között.
  - Lényegében $R(\mathbf{w})$ tagjainak súlyát szabályozzuk

Többrétegű Perceptron (MLP)

- Klasszikus neurális hálónak is hívják



Egy neuron felépítése:



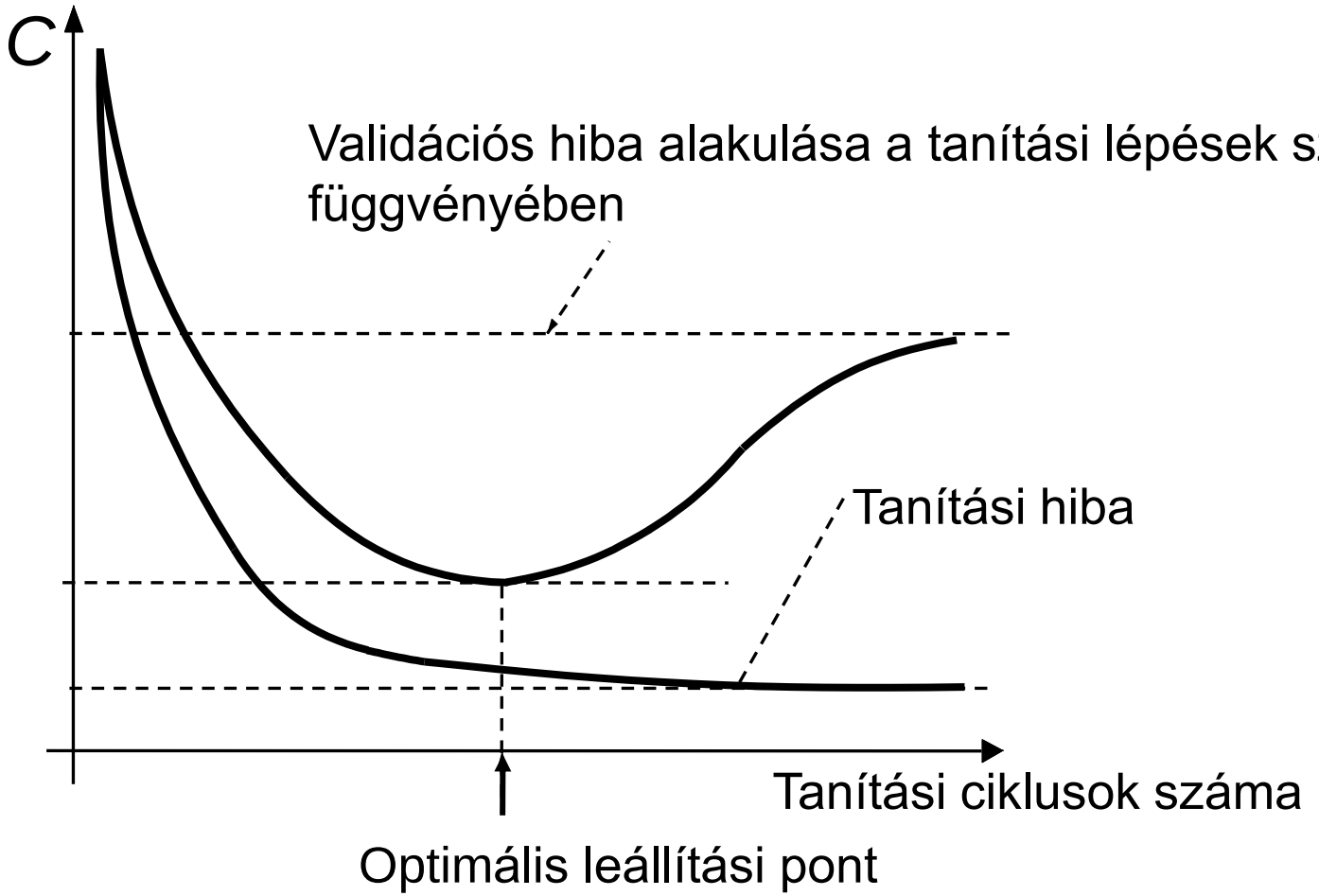
  - Nem linearitásokra példák:
    - Pl. tanh , Logisztikus szigmoid ($1/\left(1+\exp\left(-s\right)\right)$)
    - Kritérium, hogy gradiens alapú optimalizációnál (pl. hiba visszaterjesztés) ne „sikkassza el” a hibát

Többrétegű Perceptron (MLP)

- Hány rejtett réteg legyen
  - Egy darab elegendően sok (de véges számú) neuronnal elég – univerzális apprixomátor
  - Mostanában mégis az 1-nél jóval több a népszerű
    - Összetettebb leképezésekhez így kevesebb neuron elég
    - Eltérő típusú rejtett rétegek
- Egy rétegen belül hány neuron legyen
  - Elegendően sok a problémához
  - De ne legyen túl sok
    - Túl sok szabad paraméter, nagyobb VC dimmenzió

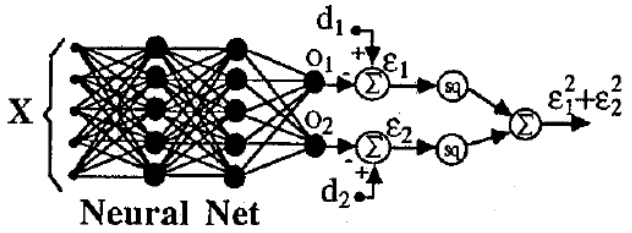
MLP tanítása

- Korai leállás (a túltanulás elkerüléséért):

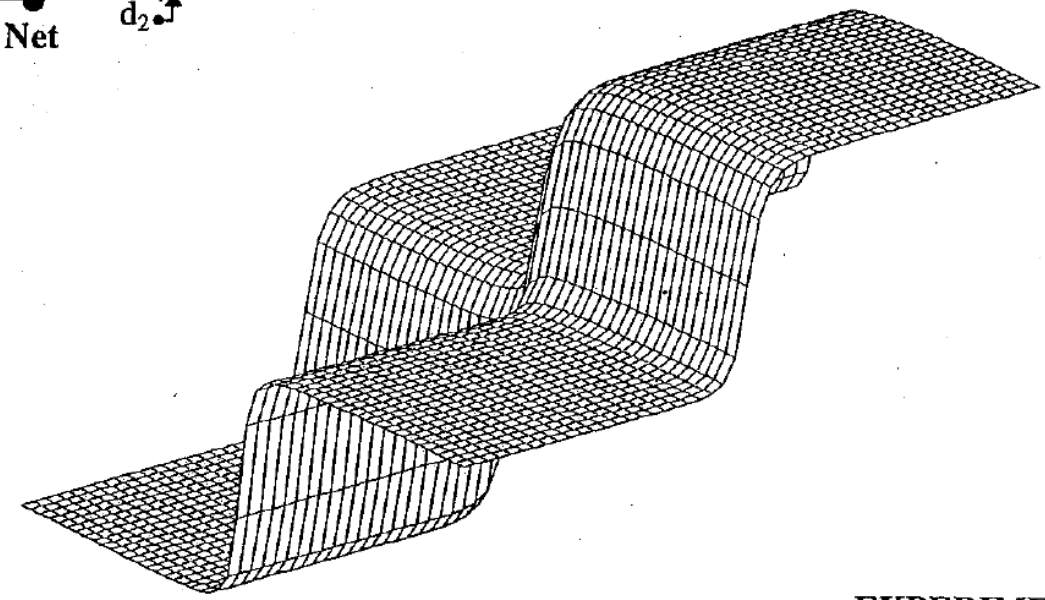


MLP hibafelület példa (1)

Ellaposodó / felrobbanó hibafelület



- 4 inputs
- 3-layer network: 5 feed 8 feed 2
- sigmoids

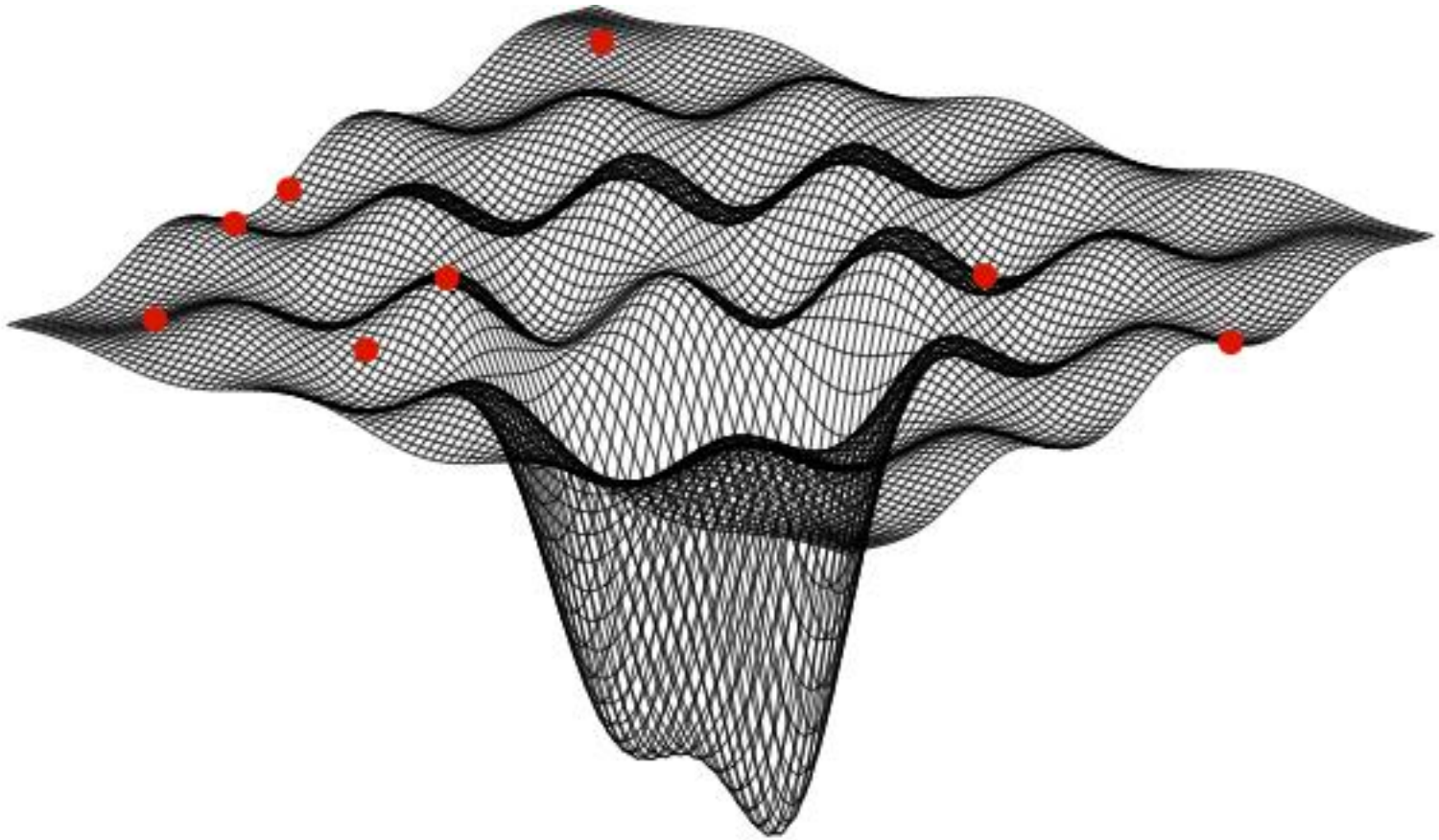


EXPERIMENT 3a.
Randomize weights, then vary two first-layer weights.

Mean Square Error Surface

MLP hibafelület példa (2)

- Sok a lokális szélsőérték



Objektum / mintafelismerés megközelítése

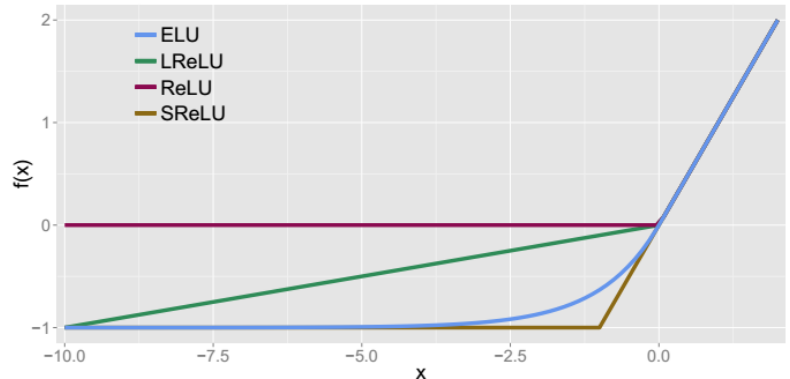
- **Traditional Pattern Recognition:** Fixed/Handcrafted Feature Extractor

- **Mainstream Pattern Recognition**

- **Deep Learning:** Multiple stages/layers trained end to end

Konvolúciós neurális hálózat (CNN)

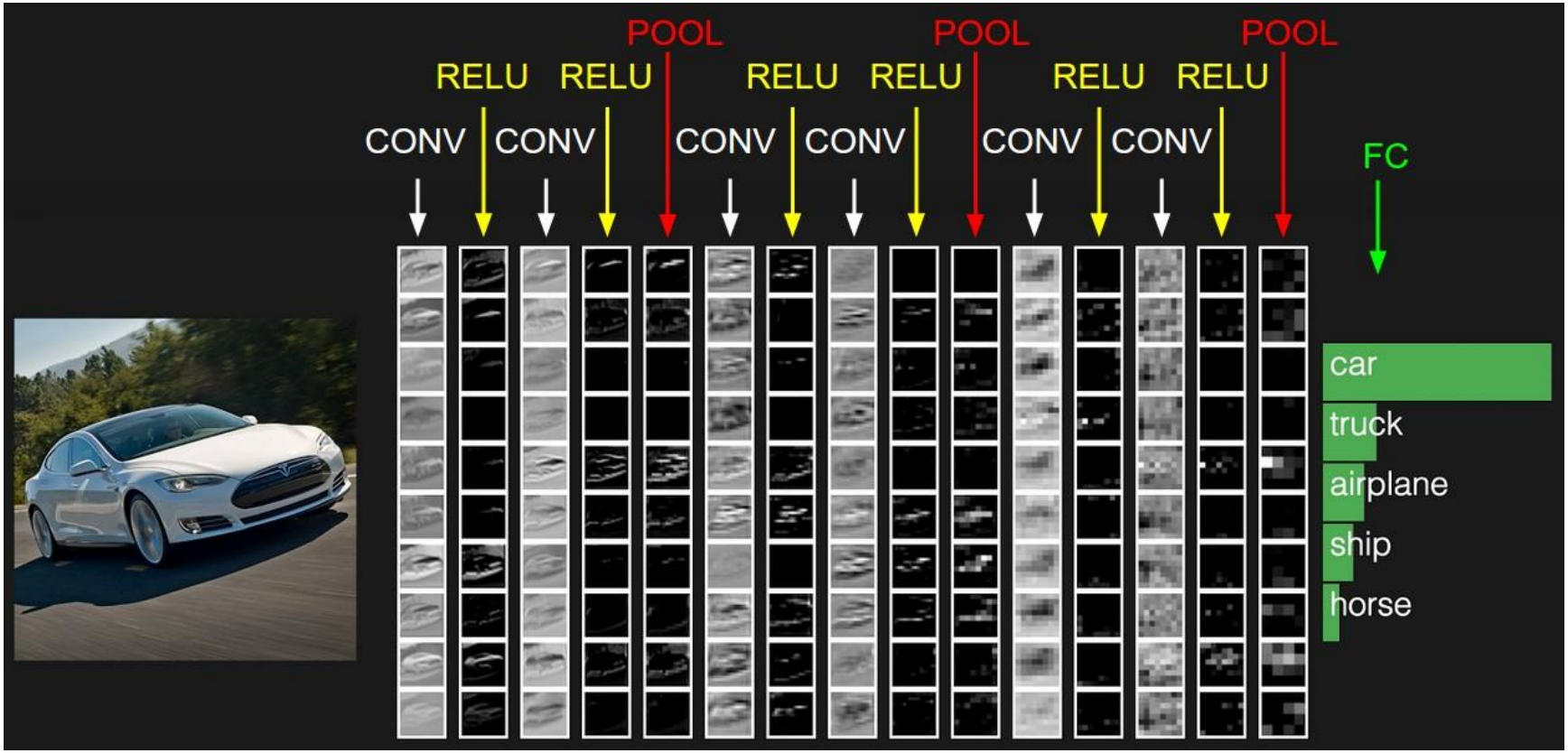
- Alapötlet – a szűrőkernelt is tanulja meg a háló:
  - Így nem kell megtippelni, hogy mi alapján ismerünk fel egy adott objektumot
  - A neurális háló bizonyos rétegei LSI lineáris rendszerek kernelét tanulják
- Cserébe nagy idő és memória igény:
  - Mélyebb hálók – kisebb kernel is elég
  - Viszont a klasszikus MLP-nél használt nem-linearitások „elnyelik” a hibát
  - Újítás nem-linearitások:



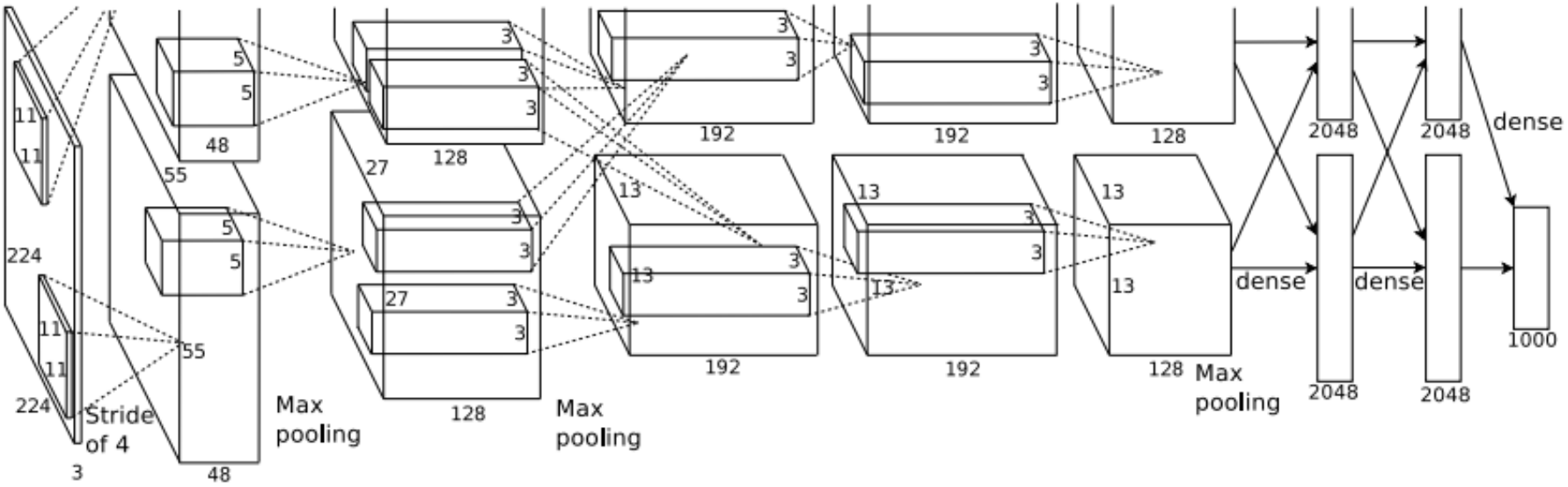
CNN

- 4 különböző típusú réteg:
  - Konvolúciós: kernelt tanul (tipikusan 3×3, 5×5)
    - Paraméterek: kernel méret, stride, kép szélének kezelése
  - Nem-linearitás:
    - ReLU, szivárgó ReLU, zajos ReLU, MaxOut, …
  - Pooling: szűrt képek alul-mintavételezése
    - Általában nemlineáris (maximum pooling)
    - Paraméterek számát korlátozza, de a bijektív jelleg biztosan elveszik – jelentősen nehezíti az interpretációt
  - Fully Connected Layer: klasszikus MLP
    - Osztálycímkéket ettől várjuk

CNN példák


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

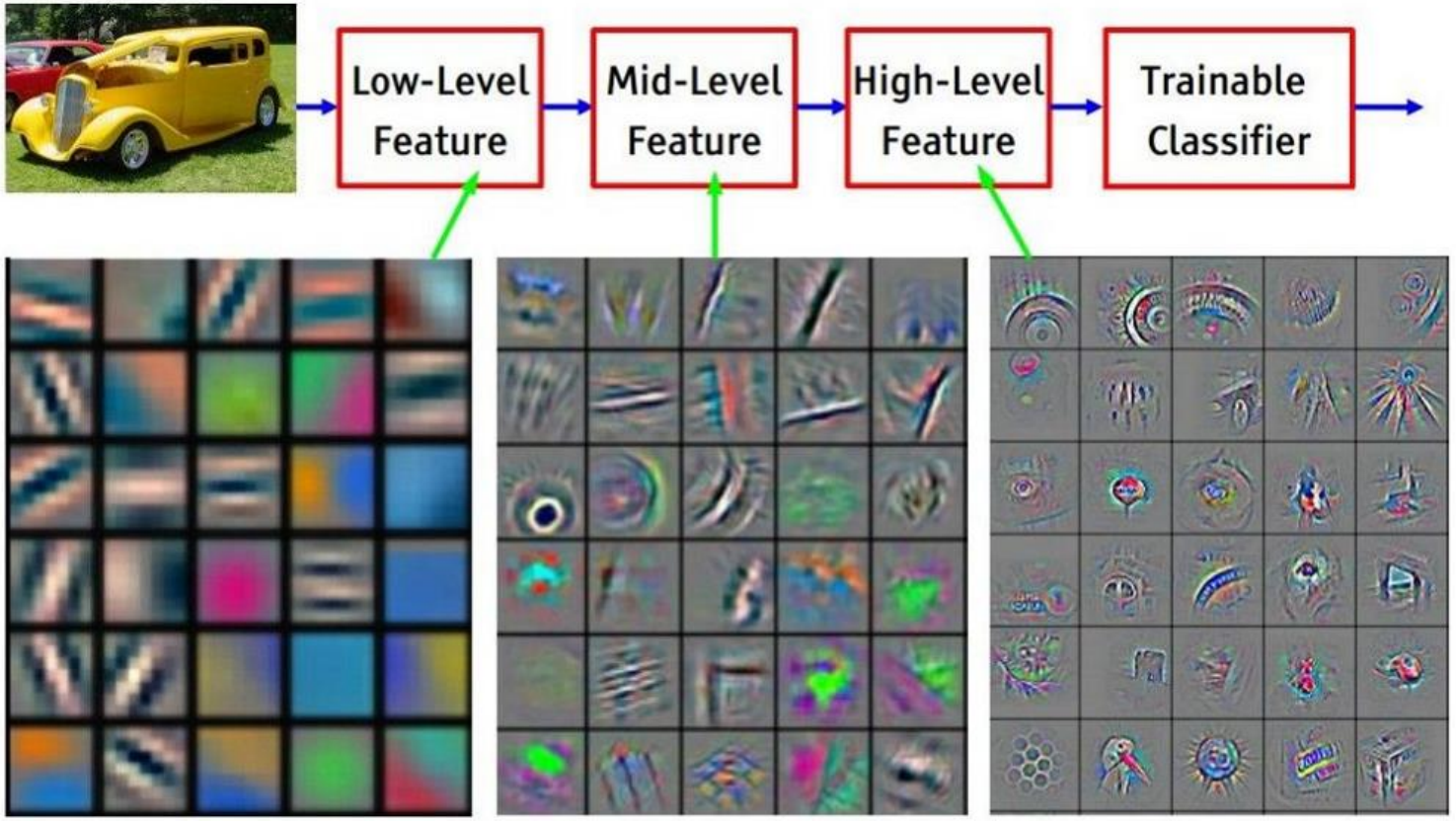
CNN példák



Bemenet: 150.528-dimenziós
a neuronok száma a háló további rétegeiben: 253.440; 186.624; 64.896; 64.896; 43.264; 4096;4096; 1000.

CNN példák

A szűrők, mint tulajdonság-érzékelők



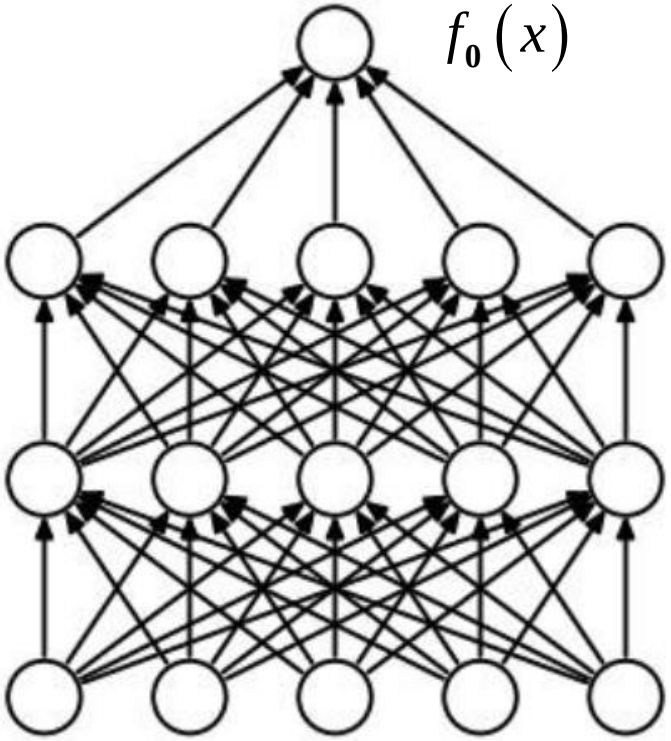
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

CNN regularizáció - dropout

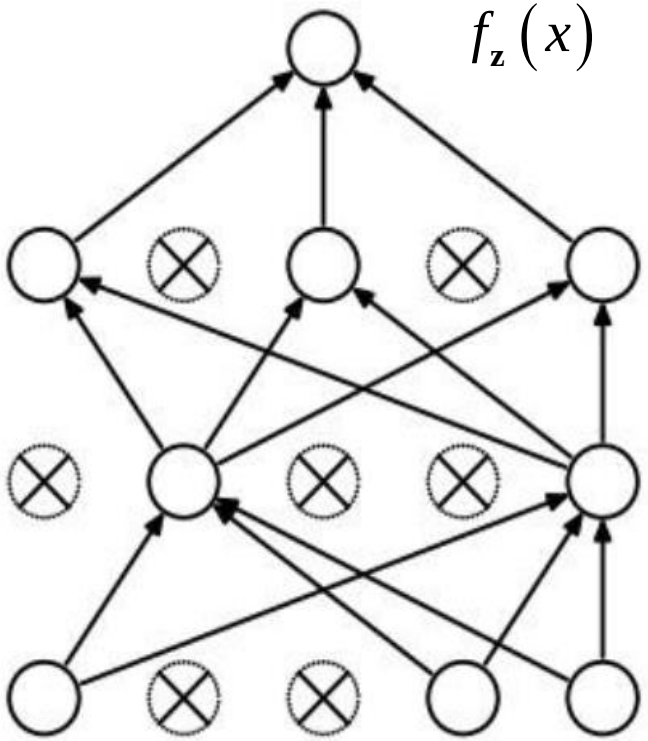
- Implicit regularizáció – Dropout:
  - Neuronok kimenetét multiplikatív impulzus zajjal (i.i.d. Bernoulli eloszlás mintavételezése terheli: *1-p várható érték*)
  - Hatása visszavezethető a szakértő együttesek elméletére
    - *Bayes-i modellátlagolás* (teszt időben közelítjük)
  - Könnyen számolható
  - Redundáns jellemzők tanulására készteti a hálót => ha valamelyik neuron kimenete hibás, az nem rántja magával az egészet.
  - Persze növeli a szükséges epochok számát

CNN regularizáció - dropout

- Tanítás során:



Teljes háló

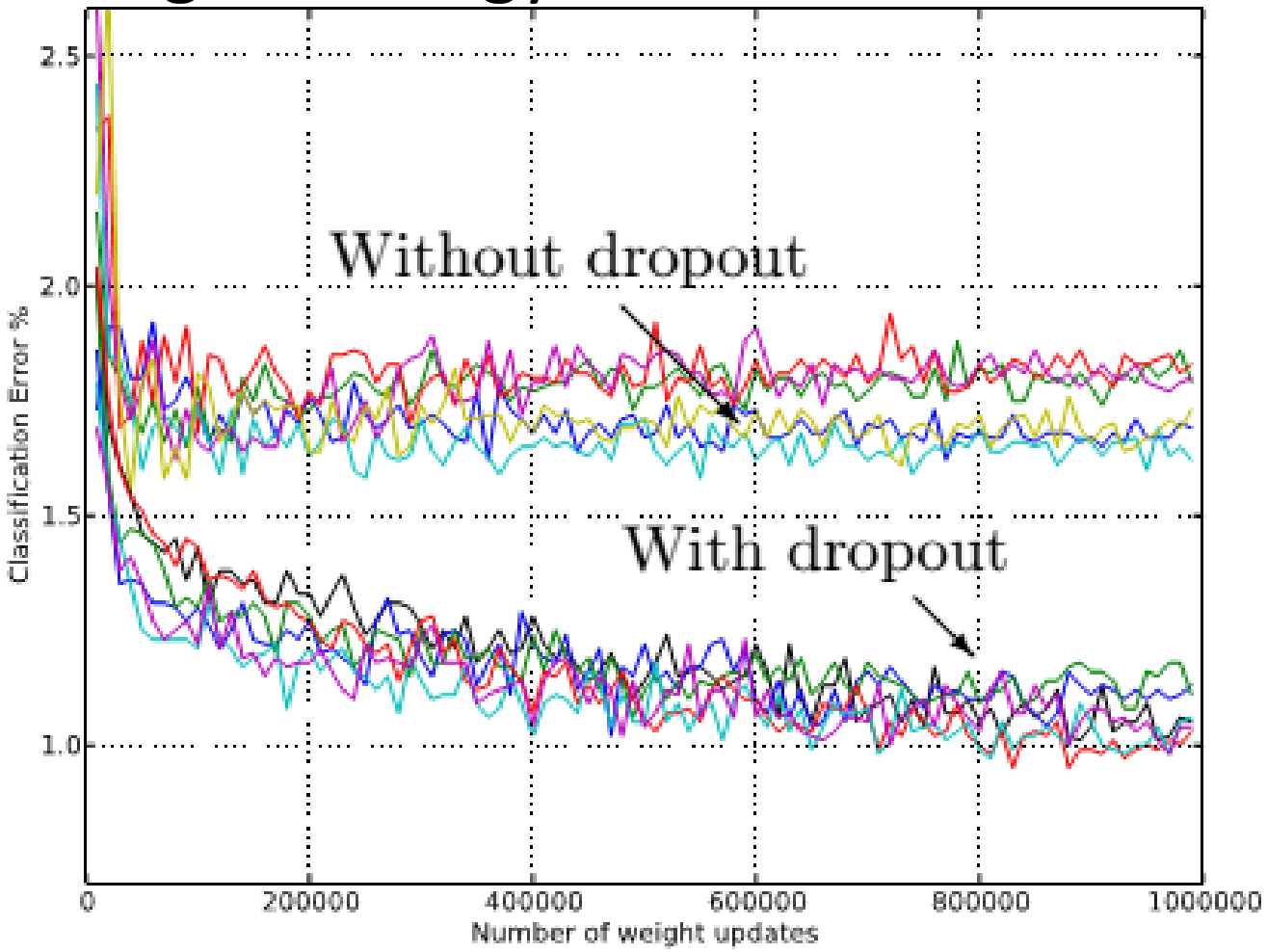


Dropout „sorsolás” utáni háló

$$\mathbf{z}_{(i)} = \begin{cases} 1 & \left| s \sim U(0,1) < p \right. \\ 0 & \left| s \sim U(0,1) \geq p \right. \end{cases}$$

CNN regularizáció - dropout

- Tanulási görbékre gyakorolt hatás:



CNN tanítása

- SGD, illetve annak különböző variánsai
  - Sztochasztikus jelleg a mini-batch-ből következik: tanító minták egy részhalmaza alapján becsli adott epochban a háló hibáját
  - Nagy rétegszám – jelentősen le tud lassulni a tanulás kedvezőtlen inicializáció esetén
  - Batch normalizáció:
    - Minden réteg bemenetét 0 várható értékűvé , 1 szórásúvá transzformálja, majd ez alapján módosítja a hálót – redukálja a paraméterek egyidejű módosításának kedvezőtlen hatásait
    - Jelentősen gyorsítja a konvergenciát

CNN tanítása – transfer learning

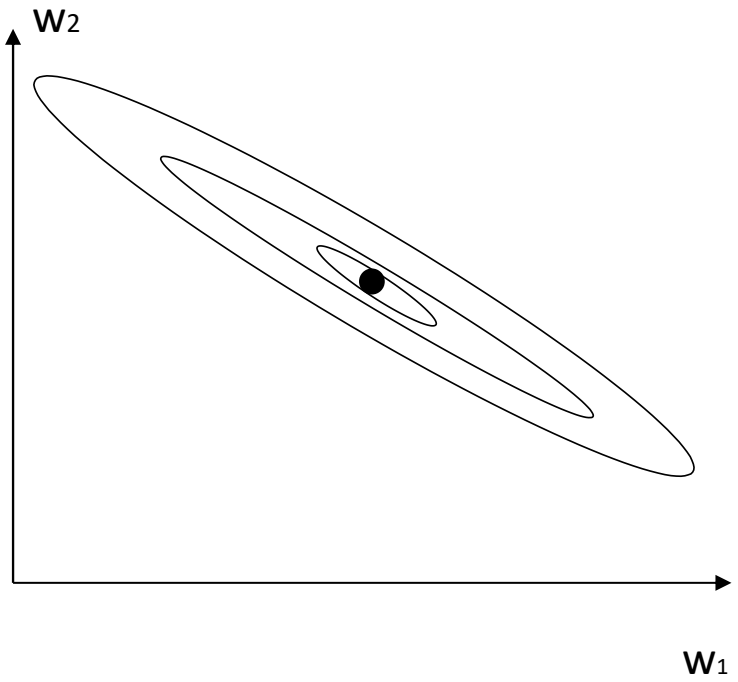
- Sok paraméterhez sok tanítóminta szükséges
  - lsd. $R(\mathbf{w})$ majorálása
  - Alacsonyabban lévő rétegek kis absztrakciójú jellemzőket detektálnak (pl. él, sarokpont. stb.)
  - A probléma specifikusabb jellemzők az FC réteg bemenetének közelében állnak elő
  - Transfer learning: Csak az FC-t tanítjuk, jellemző kiemelő rétegek paramétereit más (több mintás) problémára tanítjuk
  - Fine tuning: Csak a kimenethez közeli jellemző kiemelő rétegeket tanítjuk
- Minta augmentáció:
  - Olyan torzításokkal generálunk még mintákat, melyekre invariáns osztályozást várunk el (pl. elforgatás, eltolás, stb.).

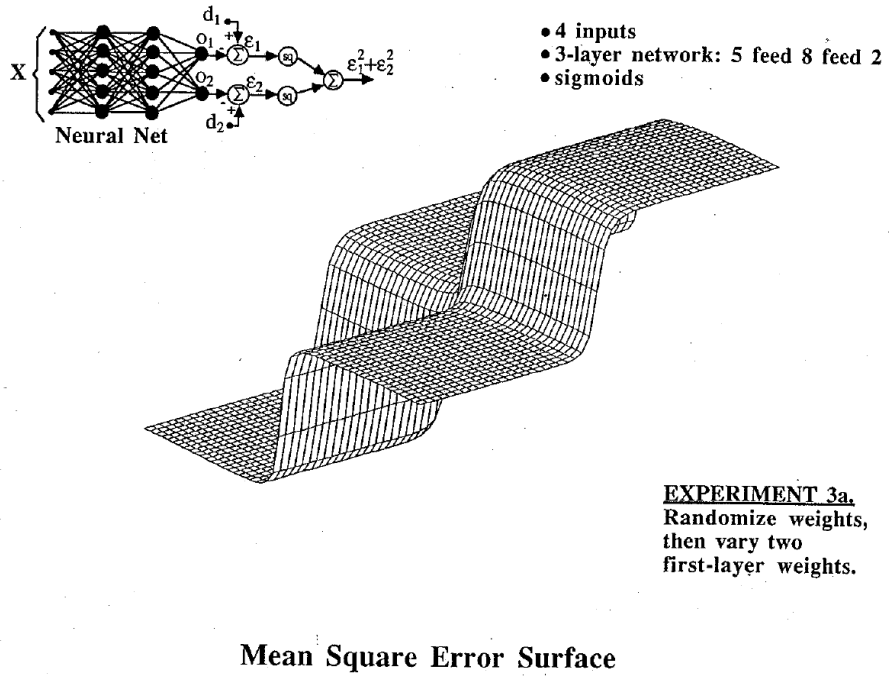
Optimalizáló eljárások

- Másodrendű módszerek:
  - Newton módszer: hibafüggvény lokális másodfokú közelítését minimalizálja iterációnként
  - Kvázi Newton módszer: Hesse mátrixot / annak invertálását csak közelíti (pl. BFGS, L-BFGS)
  - Levenberg Marquardt: Tihonov regularizált Newton (Trust Region Newton módszer)
- Elsőrendű módszerek:
  - Konjugált gradiens: konjugált irányú hibajavítás, (kvadratikus függvény esetén másodrendű módszer)
  - Nesterov Momentumos / adaptív gradient descent eljárások
  - Sztochasztikus gradiensek esetén hatékonyabbak

Elsőrendű optimalizáló eljárások

- Rosszul kondícionált hibafelületek – bajban a GD:
  - Ellaposodó platók / hirtelen letörések
  - Erősen nem izotropikus felületek





Elsőrendű optimalizáló eljárások

- Ötlet: szűrjük a súlymódosító vektorokat
  - Polyak átlagolás (momentum) alapú módszerek – simítja a lépés irányát az iterációk felett (alul- átereszti)
    - Nesterov momentum aszimptotikusan optimális elsőrendű eljárás, ha gradienst és nem sztochasztikus gradienst használunk (tehát itt általában nem az)
  - Adagrad / RMSProp: átlagolja az elmúlt iterációkon belüli paraméter módosításokat
  - Adaptive Momentum (Adam): kombinálja az előző két módszert

Elsőrendű optimalizáló eljárások