Neuronhálózatok a részecskefizikában

Bagoly Attila
ELTE TTK Fizikus MSc, 2. évfolyam

Integrating Machine Learning in Jupyter Notebooks

Google Summer of Code Project

2016.10.10

Tartalom

1. Bevezető
2. Lineáris regresszió
3. Logisztikus regresszió
4. Illesztés jellemzése
5. Problémák
6. Neuronhálózatok
7. Mély neuronhálózatok
8. Bevezető: ROOT TMVA
9. Higgs adatszett
10. Analízis folyamata

Bevezető

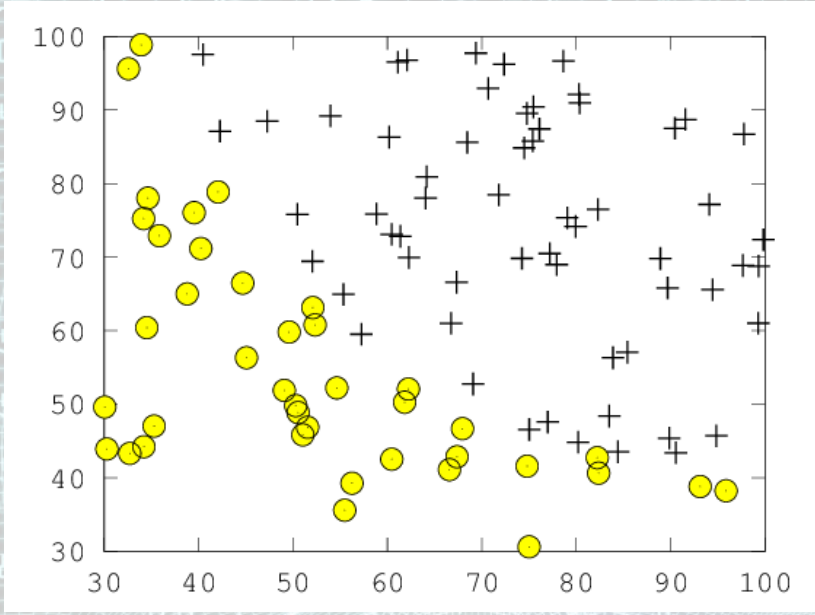
- Nagyon magas dimenziós problémákra, komplex modellt szeretnénk
- "Klasszikus illesztés": nem működik
- Probléma "megkerülése": neuronhálózatok
- Klasszifikáció: modell kimenete diszkrét (valahány osztályba sorolás)
- DNN manapság népszerű: erős számítógép (GPU), sok adat
- Machine learning: hihetetlenül gyorsan fejlődő terület
- DNN: része mindennapjainknak
- HEP-ben is egyre népszerűbb: CERN IML(`http://iml.web.cern.ch/`), tagok száma 100-as nagyságrend
- CMS L1 trigger: boosted decision tree hardver

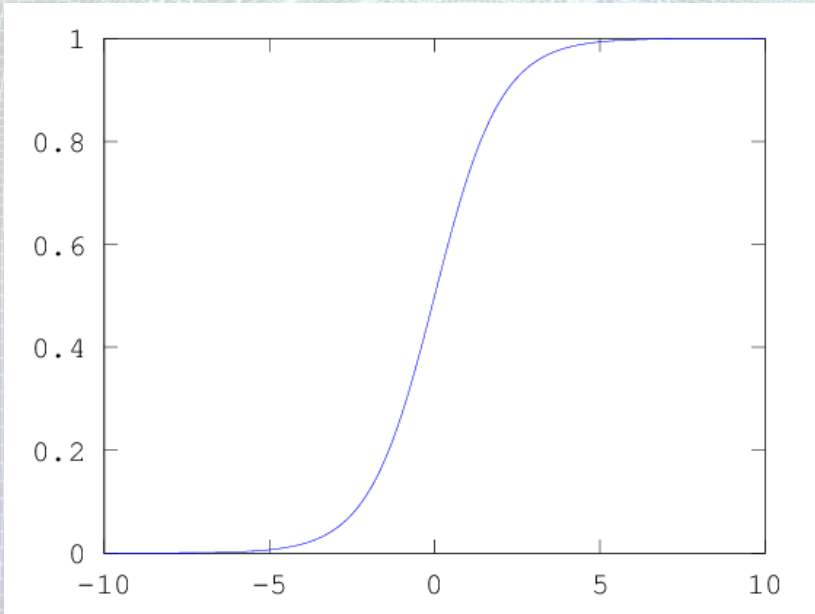
Lineáris regresszió

- Mindenki által ismert "egyszerű" függvényillesztés
- $x_j^{(i)}$ a j-edik változó az i-edik adatsorban
- Modell: $h_\Theta(x) = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + ...$
- Nem feltétlenül lineáris: x_j kiegészíthető, ha $j = 1...N$ akkor $x_{N+k} = x_k^2$, stb.
- nem linearitás $\Rightarrow$ magasabb dimenziós lineáris illesztés
- Költségfüggvény: $J(\Theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2$
- Minimalizálás: pl. gradiens módszer $\Theta \leftarrow \Theta - \alpha \Delta J(\Theta)$

Logisztikus regresszió

- Klasszifikáció?
- Diszkrét kimenet
- Két kategória: 0 vagy 1
- $h_\Theta(x)$ tetszőleges kimenetet vehet fel $\Rightarrow$ beszorítjuk 0 és 1 közé
- Sigmoid aktivációs függvény: $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$





Logisztikus regresszió

- Mi a modell költségfüggvénye?
- Tudnia kell:
  - Ha $h_\Theta(x) = y$ akkor 0
  - Ha $h_\Theta(x) = 1$ és $y = 0$ akkor nagy
  - Ha $h_\Theta(x) = 0$ és $y = 1$ akkor nagy
- Egyszerű választás:

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

- N kategória esetén: $y \in \{0, 1...N-1\}$

$h_\theta^{(0)}(x) = P(y = 0|x;\theta)$

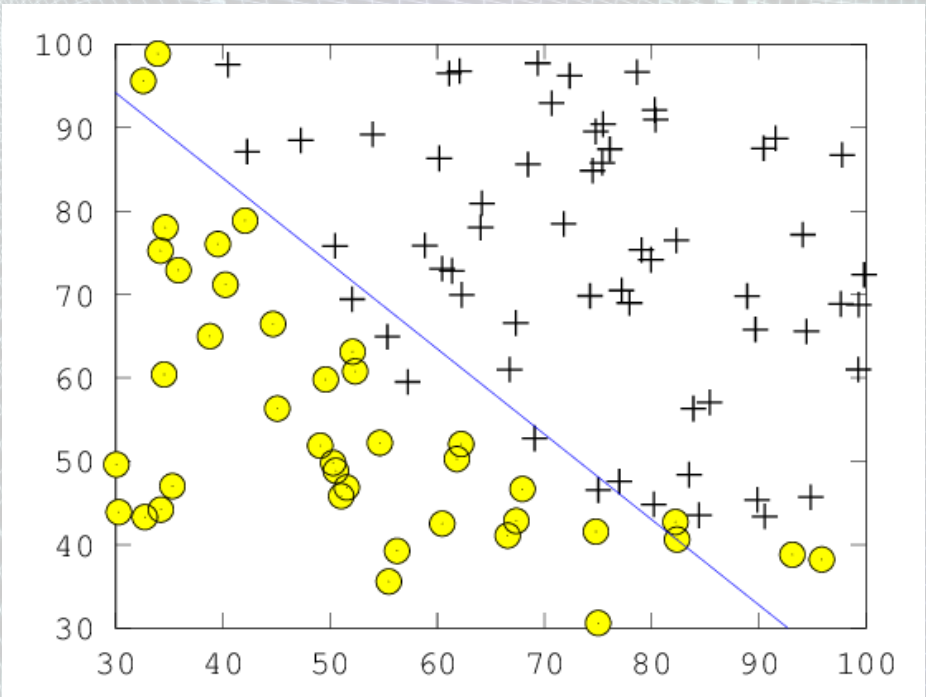
$h_\theta^{(1)}(x) = P(y = 1|x;\theta)$

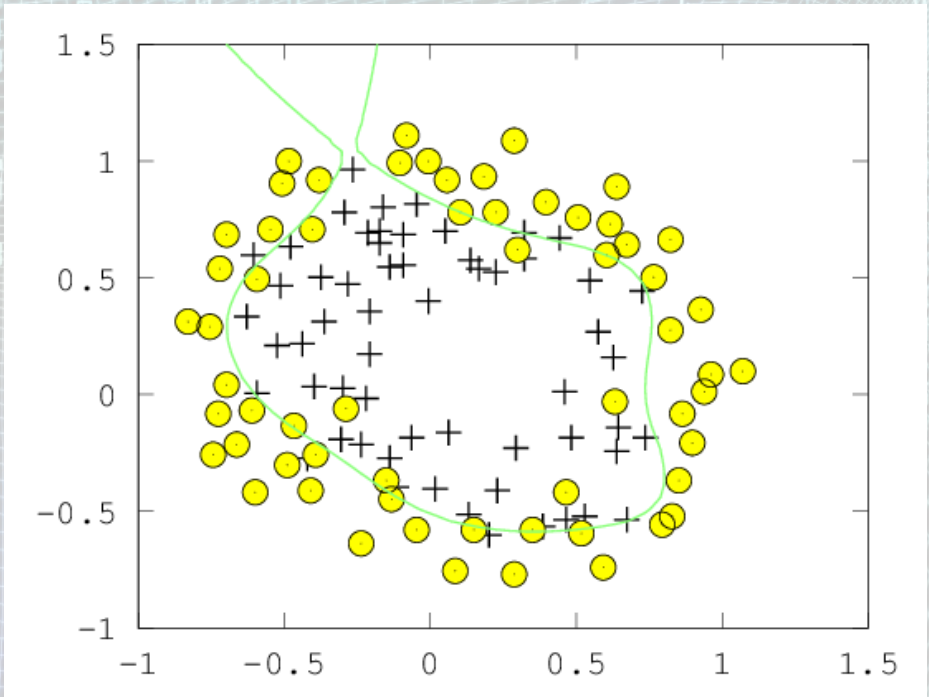
...

$\text{pred} = \max_i(h_\theta^{(i)}(x))$

Logisztikus regresszió

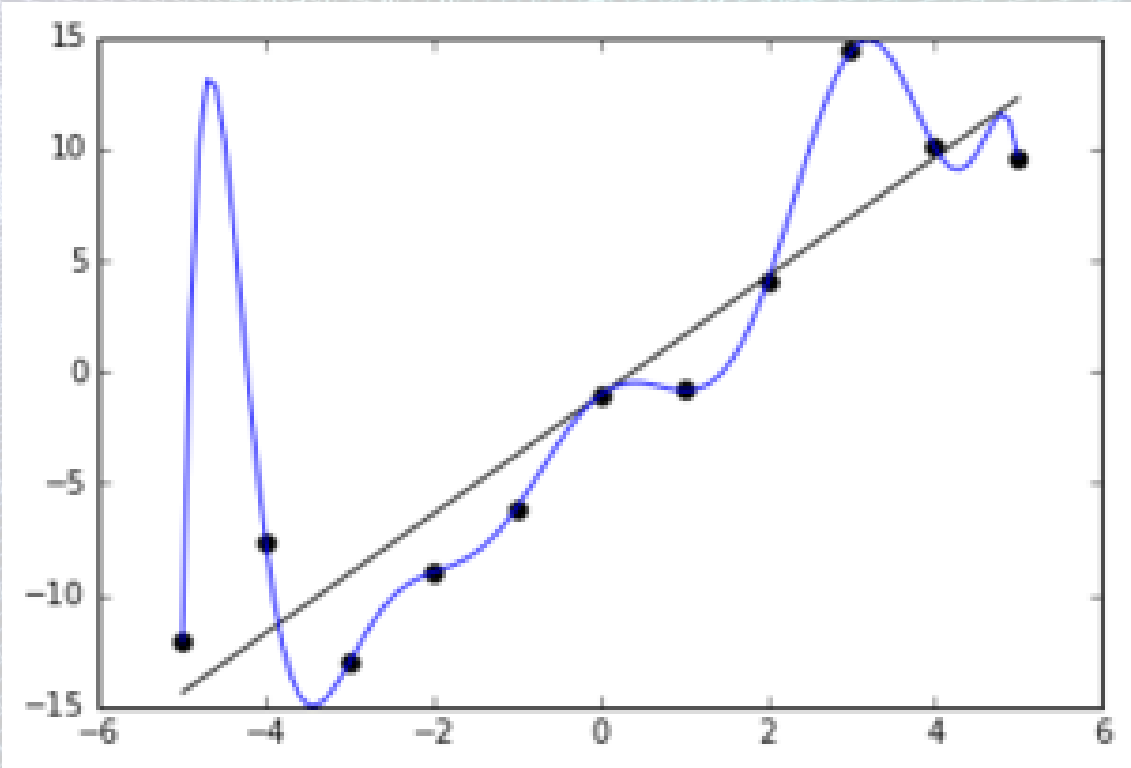
- Lineáris modell: x_1, x_2, $h_\Theta(x) = \text{sigmoid}(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2)$
- Nem lineáris modell: x_1, x_2 featureket kiegészítjük
  $x_3 = x_1^2, x_4 = x_2^2, x_5 = x_1 x_2$, és $x_0 = 1$, ekkor
  $h_\Theta(x) = \text{sigmoid}(\Theta^T x)$





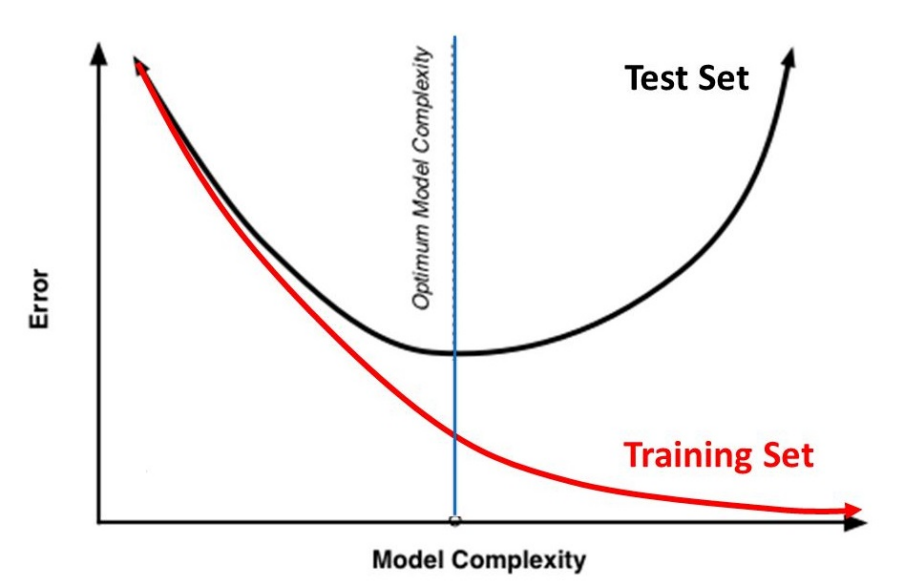
Hogy általánosít a modellünk?

- Hogy teljesít a modellünk? Nem lineáris problémánál 2D⇒5D
- $J(\Theta)$ hiba kicsi, jó az illesztés?
- $J(\Theta)$ kicsi mégis új adatra teljesen rossz eredmény $\Rightarrow$ nem modelleztünk hanem adatokat kódoltunk (overfit, high variance)
- Kevés feature: underfit, high bias



Megoldás

- Felosztjuk az adatokat: training set, test set
- Fontos: adatok random keverve legyenek
- Csak az egyiken tanul a modellünk
- Probléma: paraméterek csavargatásával a modell a szemünkön keresztül tanulja meg a test set-et
- Megoldás: k-fold Cross-Validation



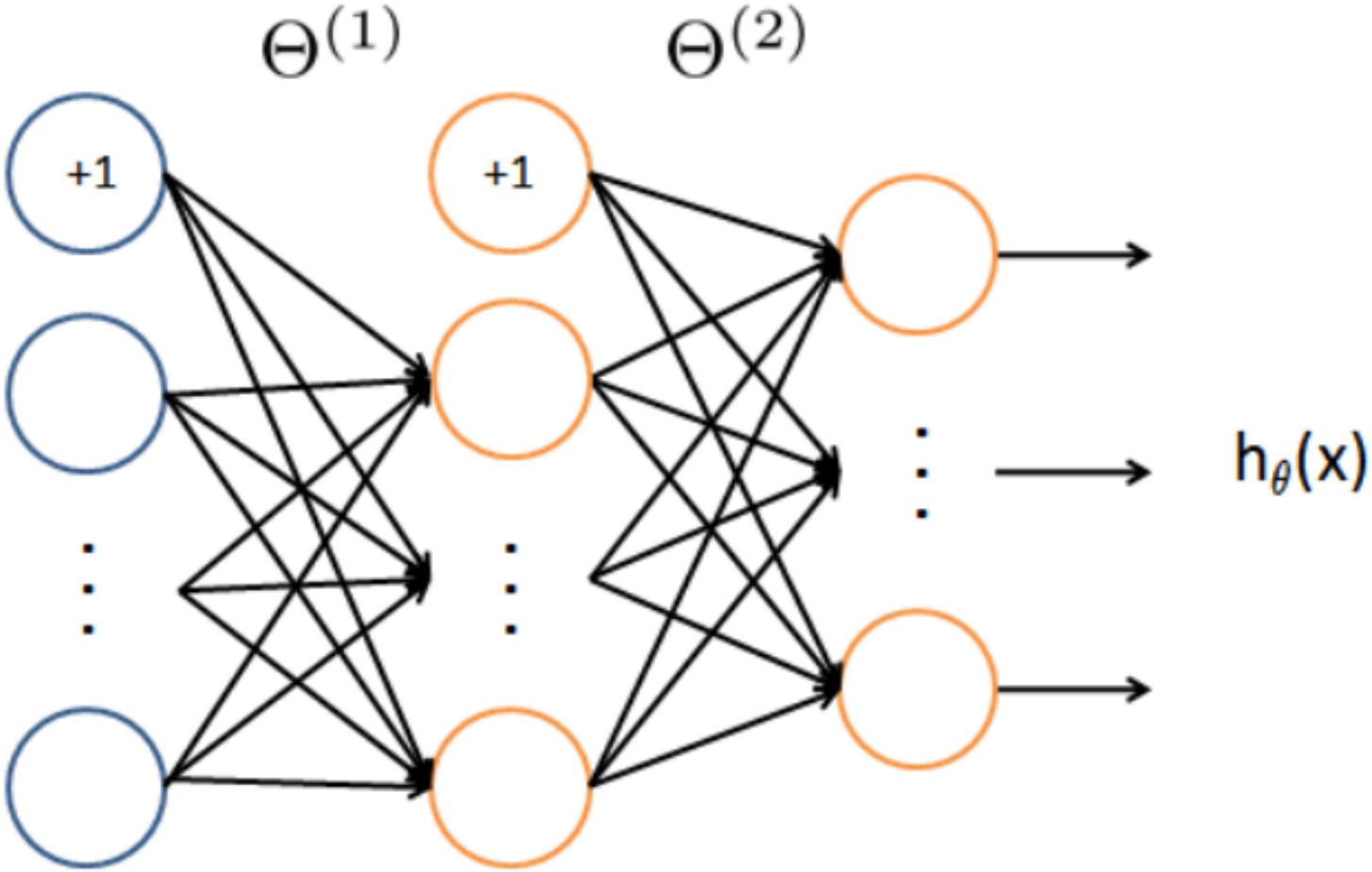
Problémák

- Valós életben általában sokkal több feature van mint kettő
- n változó, kvadratikus modell paramétereinek száma $\approx \mathcal{O}(n^2/2)$
- Pl. 50x50 pixeles szürkeárnyalatos képeket osztályozunk $\Rightarrow$ 2500 változó $\Rightarrow$ 3125000 paraméter
- Ráadásul kvadratikus modell valószínűleg high bias-t eredményez

Neuronhálózatok bevezetés

- Előbb vázolt probléma megoldására született
- Lehetővé teszi nagyon komplex modell illesztését anélkül, hogy a paraméterek száma divergálna
- Használjuk a nagyon egyszerű logisztikus modellt, de csak lineáris featurekkel $\Rightarrow$ Neuron
- Sigmoid aktivációs függvény: neuron aktiválási valószínűség (agyban is hasonló a neuronok karakterisztikája)
- Komplex modell: neuronokat összekötjük

Neuronhálózatok



$$a^{(1)} = x \qquad z^{(2)} = \Theta^{(1)} a^{(1)} \qquad z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$(\text{add } a_0^{(1)}) \qquad a^{(2)} = g(z^{(2)}) \qquad a^{(3)} = g(z^{(3)}) = h_\theta(x)$$

$$(\text{add } a_0^{(2)})$$

Neuronhálózatok: bementet, kimenet

- Amennyi bemenet annyi neuron az első rétegben: minden bemenet minden neuronra kapcsolva
- Annyi kimeneti neuron az utolsó rétegben amennyi osztályunk van
- Minden kimeneti neuronok 0 1 közti értéket vesznek fel: a legnagyobb az adott adat osztálya

Neuronhálózatok tanítása

- Költségfüggvény: csak össze kell rakni a logisztikus modellből
- Probléma: könnyű overfittelni neuronhálókkal
- Megoldás: regularizáció bevezetése: négyzetesen elnyomjuk a súlyokat valamilyen paraméterrel: $\lambda \sum_{i=1(i\neq 0)} (\Theta_i)^2$, $\lambda \to 0$ overfitting, $\lambda \to \infty$ underfitting
- Tehát ezt kell minimalizálni:

$$J(\Theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} [y_k^{(i)} \log((h_\Theta(x^{(i)}))_k) + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{j,i}^{(l)})^2$$

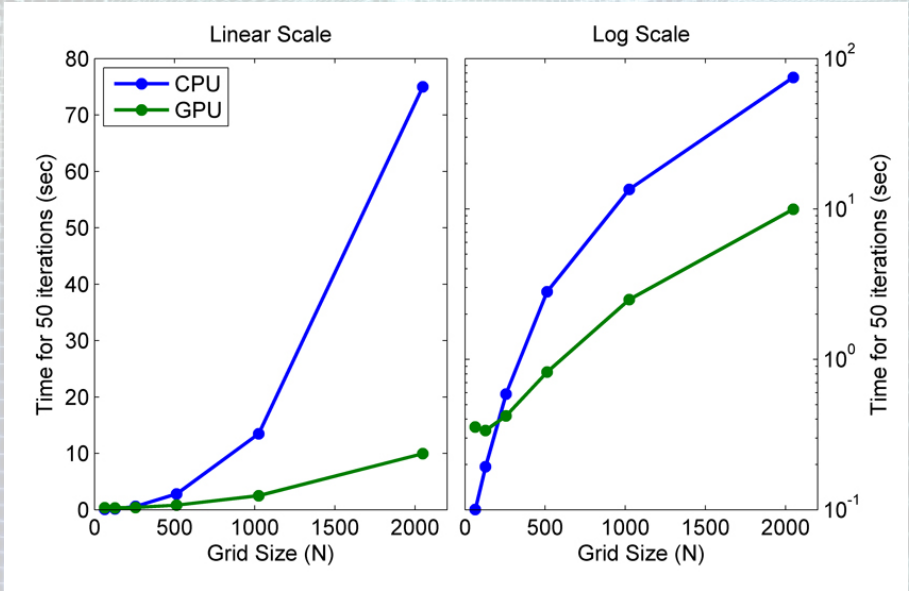
- m adatpont, K kategória, s_l neuron az l-edik rétegben

Neuronhálózatok tanítása

- Tehát neuronháló tanítása a fenti költségfüggvény minimalizálása
- Nem könnyű feladat, de elvégezhető
- Backpropagation algorithm:
  - Előre propagálás: az adatokat végigvisszük a hálón, hogy megkapjuk a neuronok aktivációit
  - Hátra propagálás: aktivációkat és cél tanuló mintát visszafele propagáltatjuk a hálózaton, így kapunk egy delta mátrixot
  - Delta mátrix segítségével felírhatjuk a költségfüggvény deriváltját
  - Derivált alapján súlyokat frissítünk
- Regularizációs paraméter optimumát is meg kell keresi (hyper parameter optimalization): k-fold coross-validation pár csoport hibái alapján

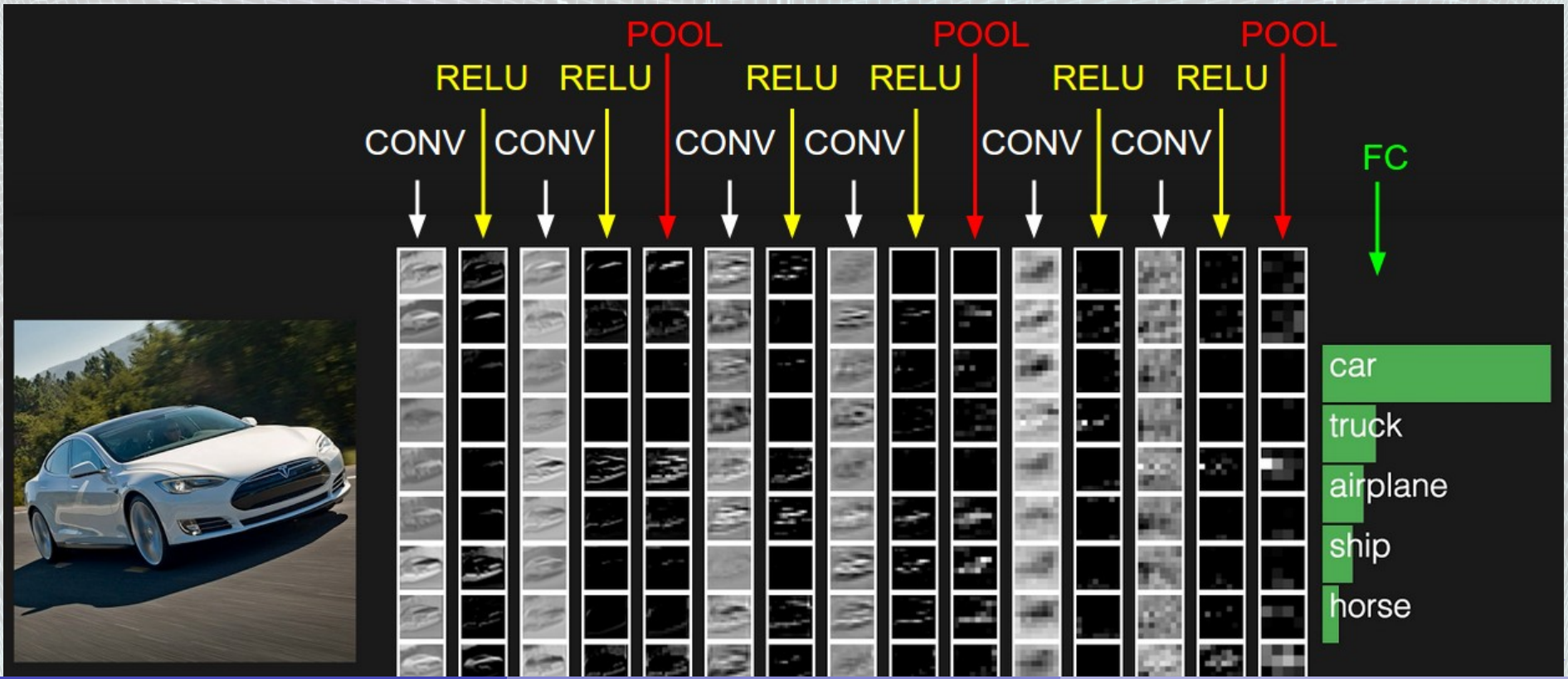
Mély neuronhálózatok

- Sok neuron a rétegekben
- Sok réteg
- Nagy hálózatok esetén a tanulás során sok nagy mátrixot kell szorozgatni $\Rightarrow$ nagy műveletigény
- Régi technológia de nem használták, mert extrém nehéz tanítani
- Manapság nagyon népszerű: hála a gémereknek
- GPU: gyors mátrixszorzásnak hála lehet deep learning

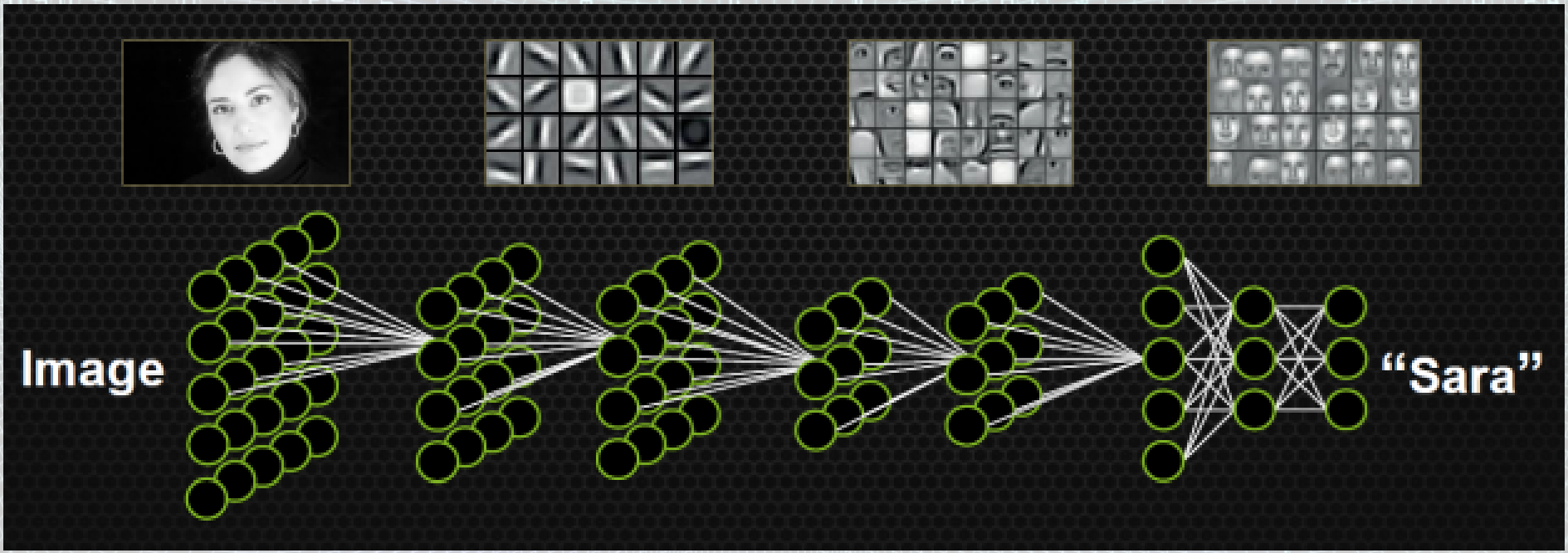


Kitekintő: konvolúciós neuronhálók

- Kép: DNN probléma $\Rightarrow$ vizuális kortex
- CNN: speciális rétegek $\Rightarrow$ új featurek DNN-hez
- 3D rétegek: pixelek fölött neuronok, lokálisan kapcsolva
- Különböző alakokat emelünk ki a képekről



Kitekintő: konvolúciós neuronhálók



Térjünk rá a fizikára

ROOT TMVA

- Toolkit for Multivariate Data Analysis (TMVA) a ROOT programcsomag része
- Célja: ML könyvtár biztosítása a fizikusok számára, megszokott környezetben
- Nem csak neuron hálókat biztosít (pl. boosted decision tree)
- Klasszifikáció mellett regressziót is támogat, de erről nem beszélek
- Alapvetően két osztály amit a TMVA névtérből kívülről használunk:
  - TMVA::Factory: ML módszerek elérése
  - TMVA::DataLoader: adatok elérése

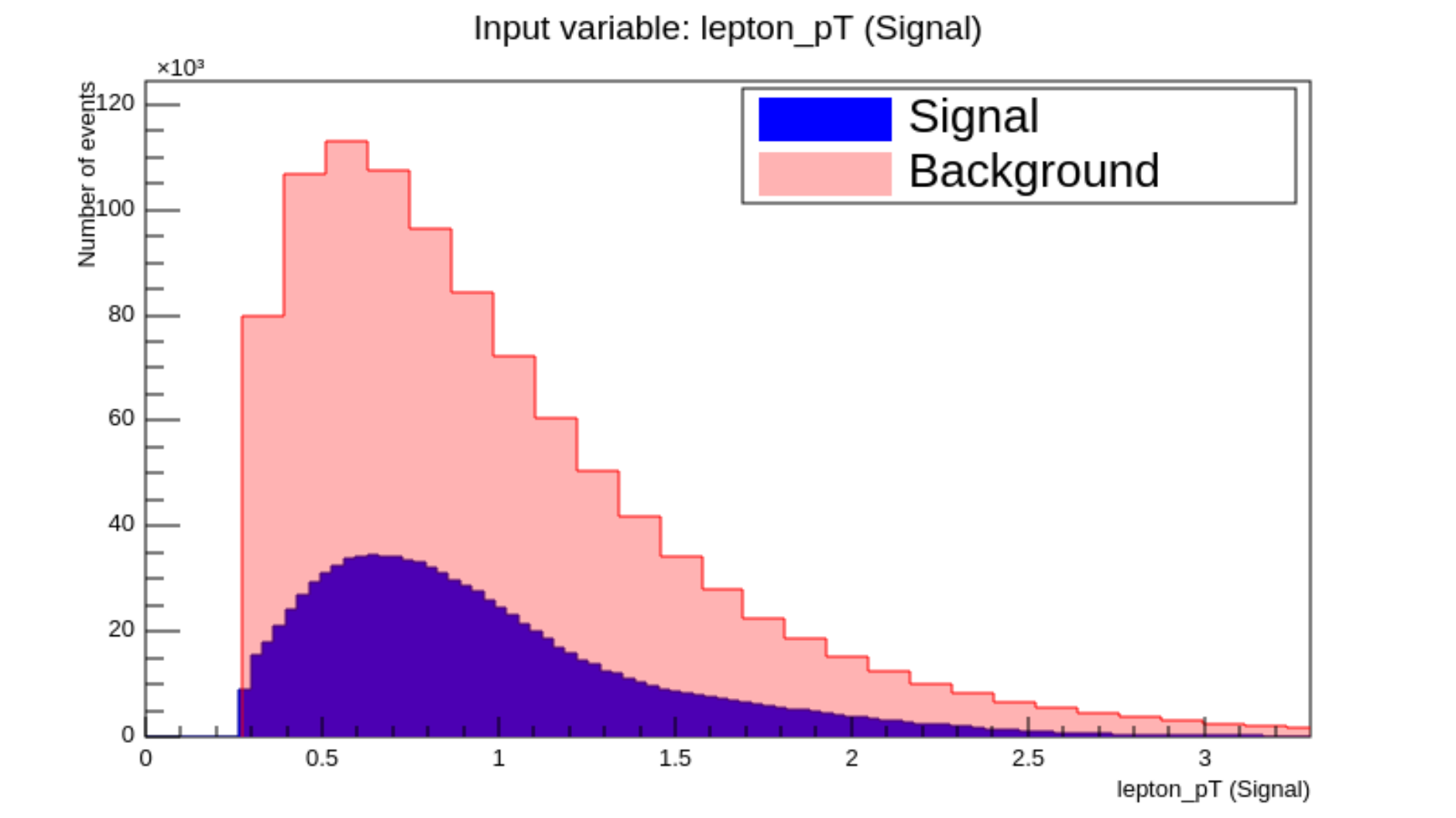
Higgs bozon keresés

- Feladat: Higgs bozon keresése eseményekben
- Nem látjuk csak a bomlás termékeket
- Lehetséges bomlás: $H^0 \to \gamma\gamma / Z^0 Z^0 \to e^+e^- + e^+e^-$ (elektron párok cserélhetők müonra), ...
- Csak a végkimenetelt látjuk, más folyamat is produkálhat hasonló kimenetet
- Meghatározzunk sok fizikai paramétert (energiát, szögeket, pszeudorapiditást): ezek alapján mondjuk meg, hogy Higgs bomlás történt-e
- DNN-t taníthatunk be a felismerése

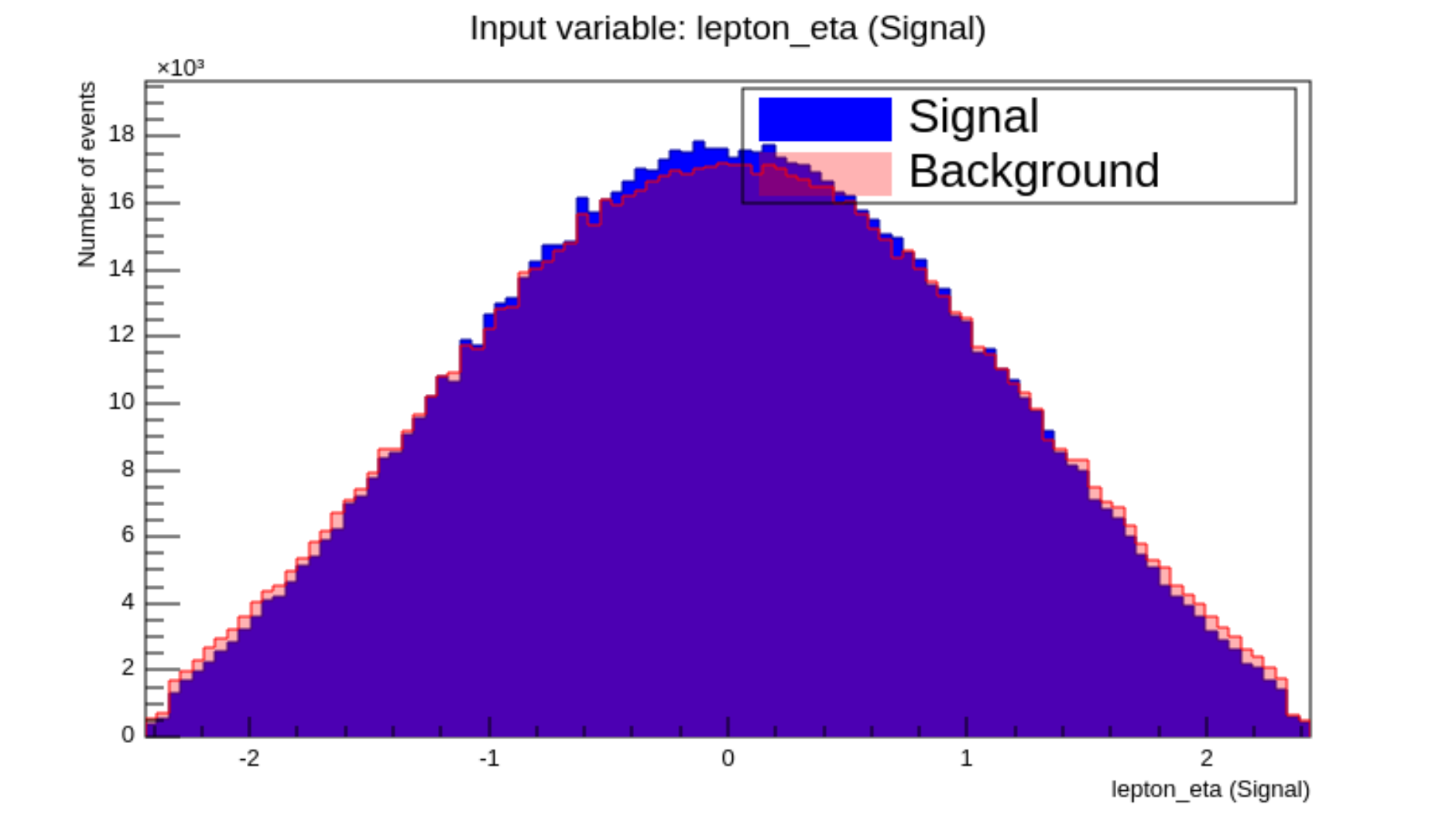
Higgs adatszett

- CERN Opendata program keretében tanuló adatszet
- 5829123 esemény, 800MB
- 21 feature
- Signal: Higgs bomlás
- Background: nem Higgs bomlás

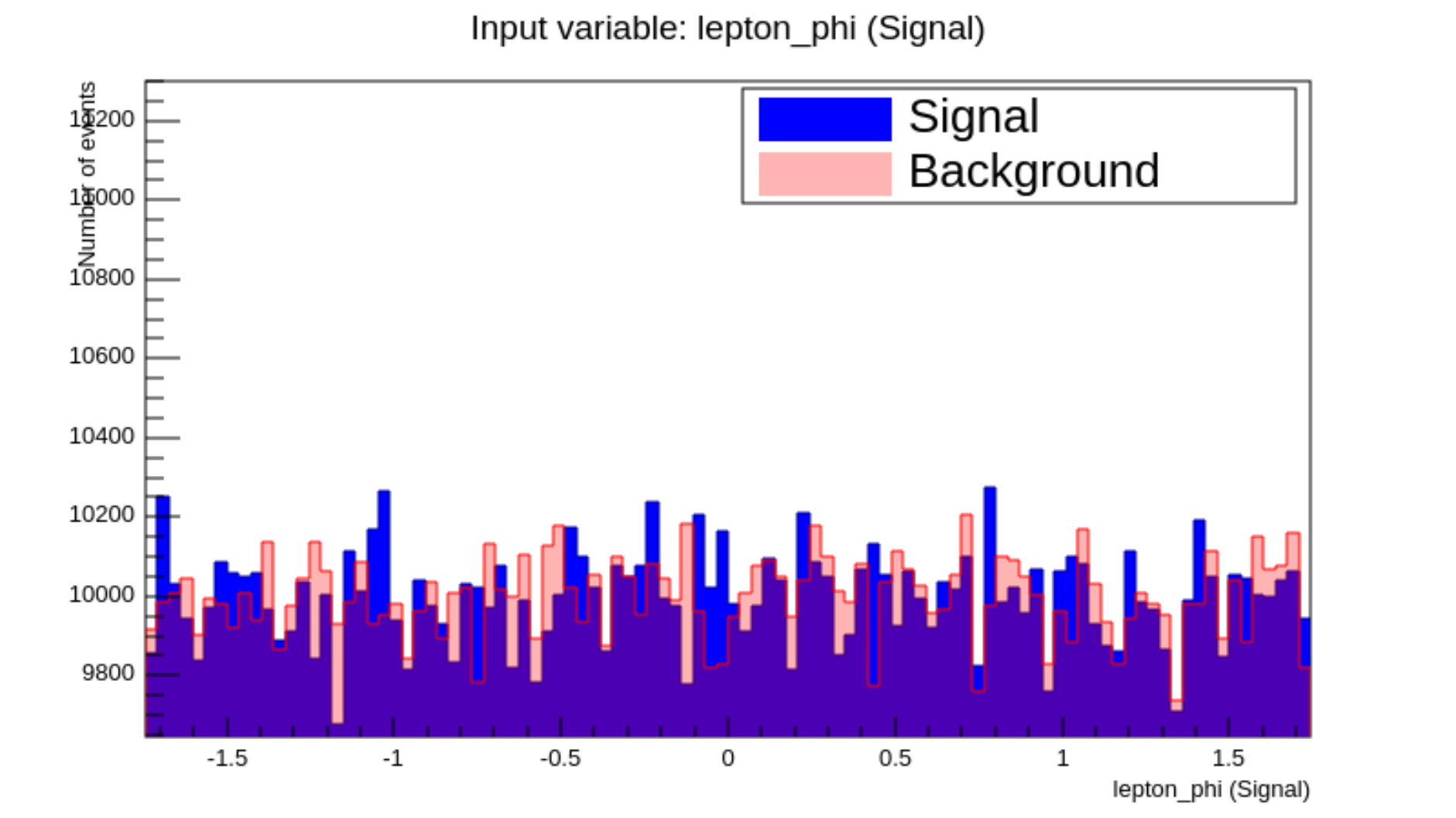
Higgs adatszett



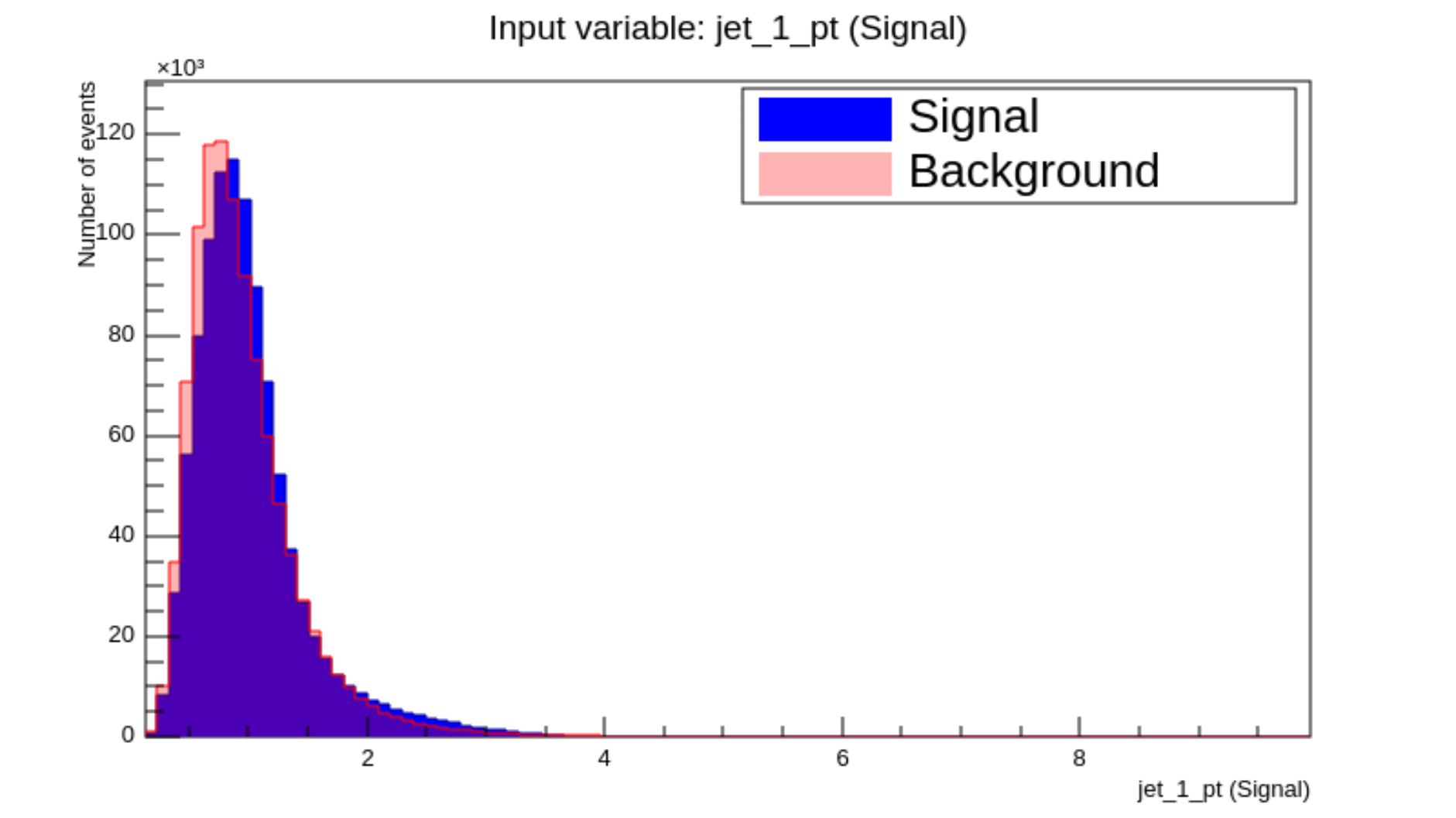
Higgs adatszett



Higgs adatszett



Higgs adatszett



Higgs challenge: https://www.kaggle.com/c/higgs-boson

- Ki tanítja be a legjobb modellt?
- $\tau\tau$ bomlások megtalálása
- Modellt beküldve: új eseményeken teszt $\Rightarrow$ jól általánosít-e a modell?
- Nyertes:
  - 70 db. 3 rejtett rétegű, rétegenként 600 neuronos hálózat
  - 2-fold cross-validation
  - 35 random keverés az adatokon
  - GTX Titan GPU (mérések CPU párhuzamosan 10x lassabb)
  - Tanulás: 1 nap (1 háló, szimpla pontossággal csak 15 perc)

Analízis folyamata

- https://github.com/qati/Presentations/blob/master/seminar.ipynb
- http://nbviewer.jupyter.org/github/qati/Presentations/blob/master/seminar.ipynb