Gyires-Tóth Bálint

Deep Learning a gyakorlatban Python és LUA alapon Tanítás: alap tippek és trükkök

SmartLab
Intelligent Interactions

http://smartlab.tmit.bme.hu

NVIDIA
GPU
EDUCATION
CENTER

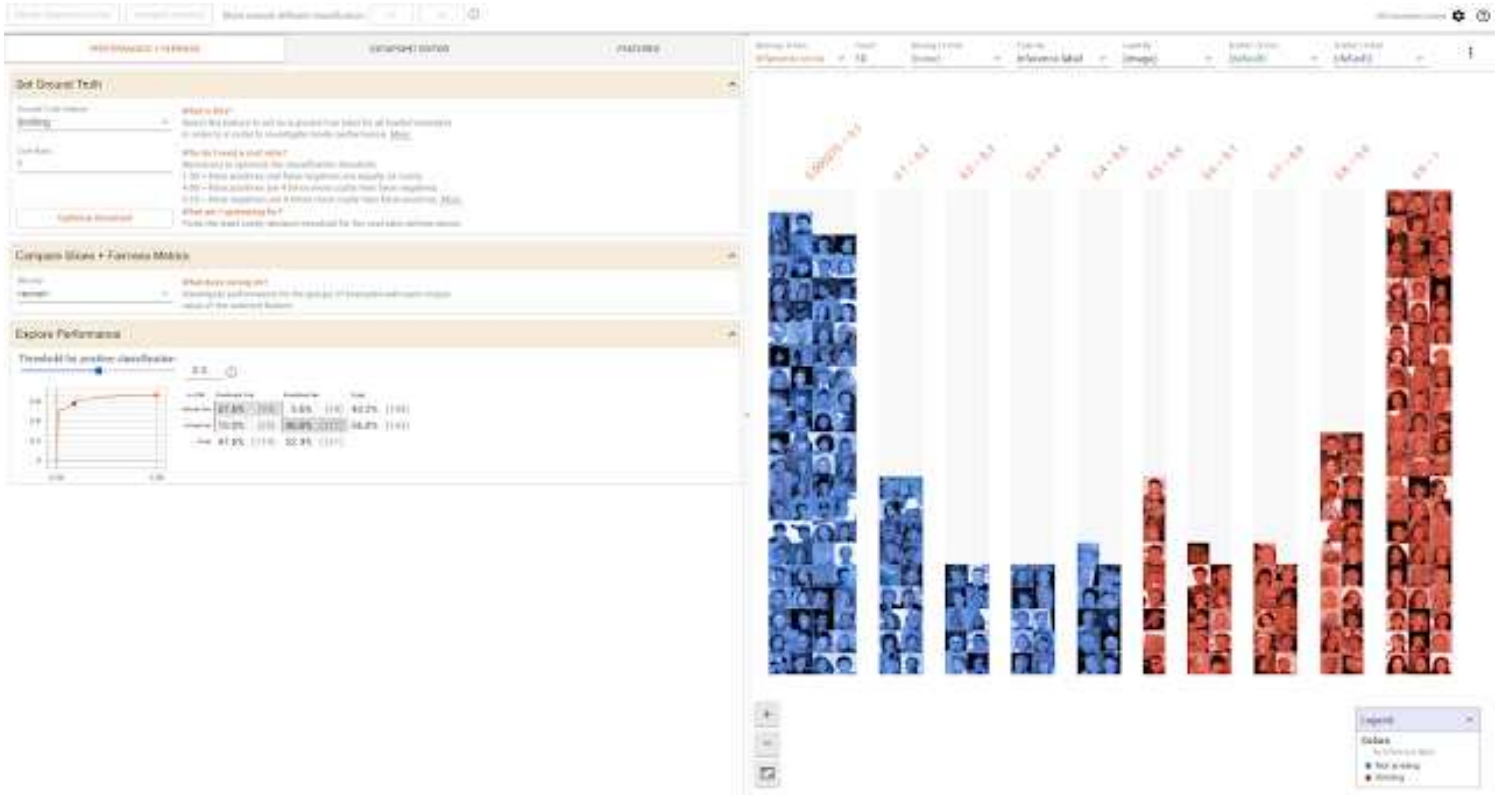
Deep Learning Híradó

Hírek az elmúlt 168 órából

Deep Learning Híradó

- Google What-If

https://pair-code.github.io/what-if-tool/



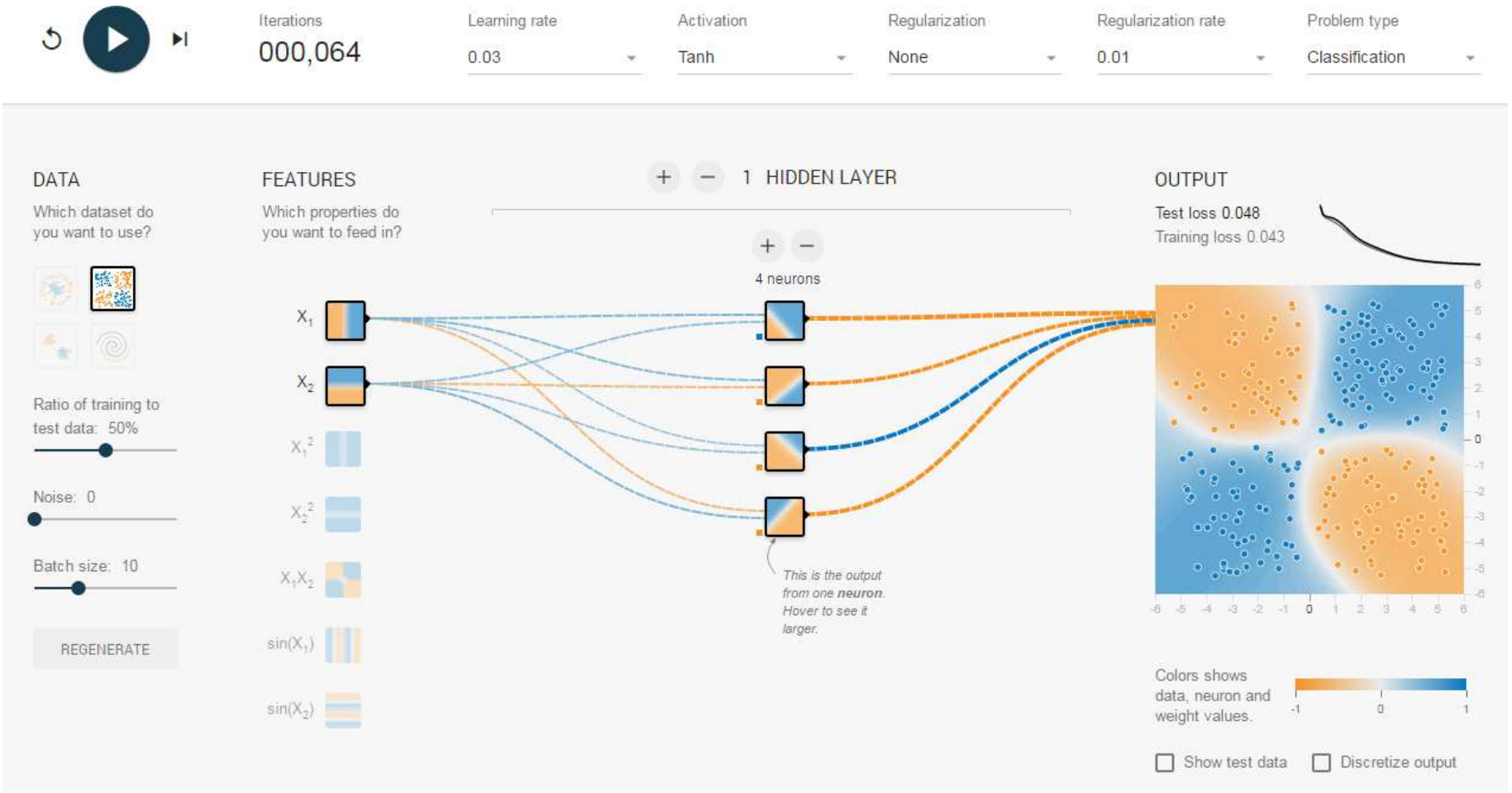
Jogi nyilatkozat

Jelen előadás diái a „*Deep Learning a gyakorlatban Python és LUA alapon*” című tantárgyhoz készültek és letölthetők a http://smartlab.tmit.bme.hu honlapról.

A diák nem helyettesítik az előadáson való részvételt, csupán emlékeztetőül szolgálnak.

Vizualizáció

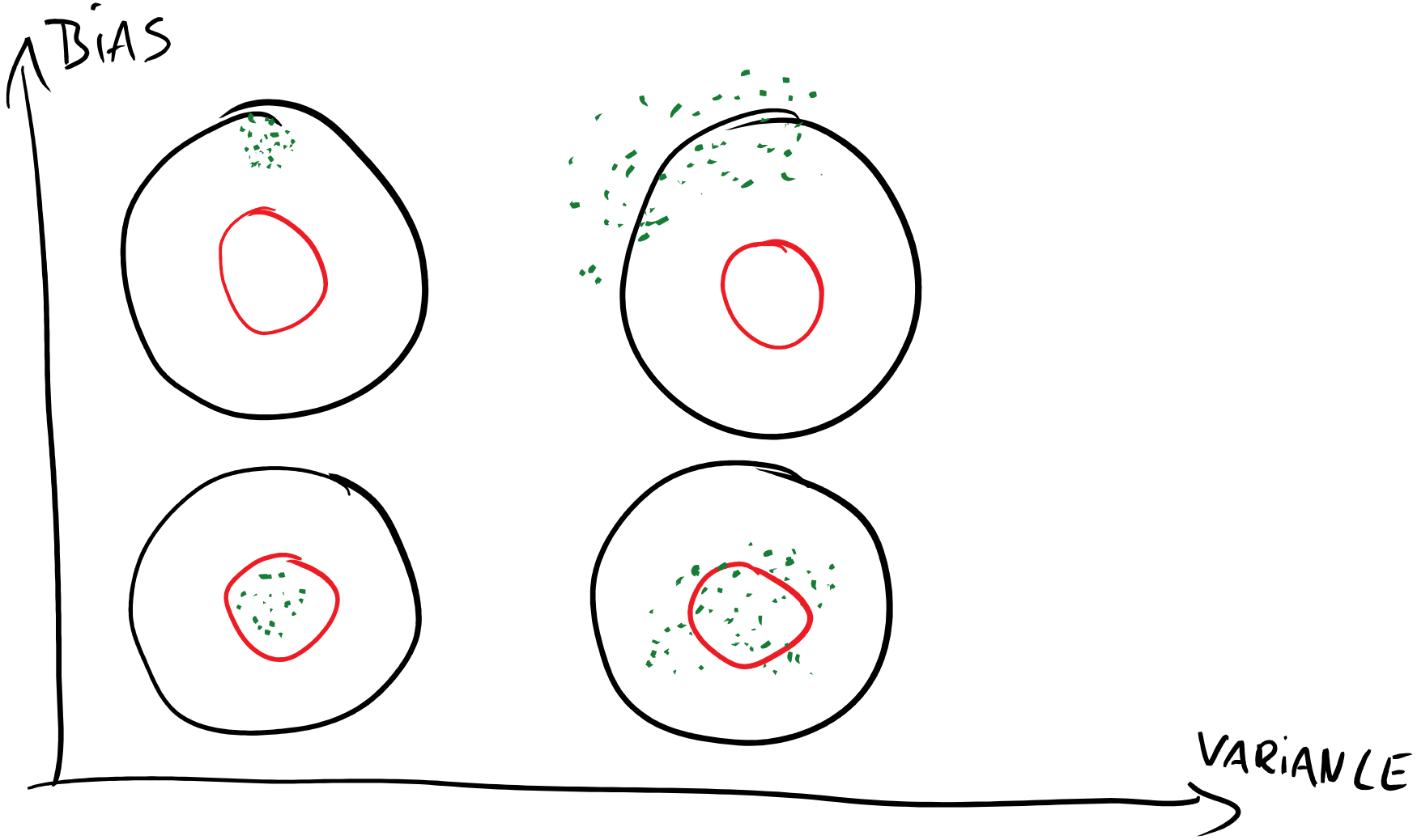
http://playground.tensorflow.org/



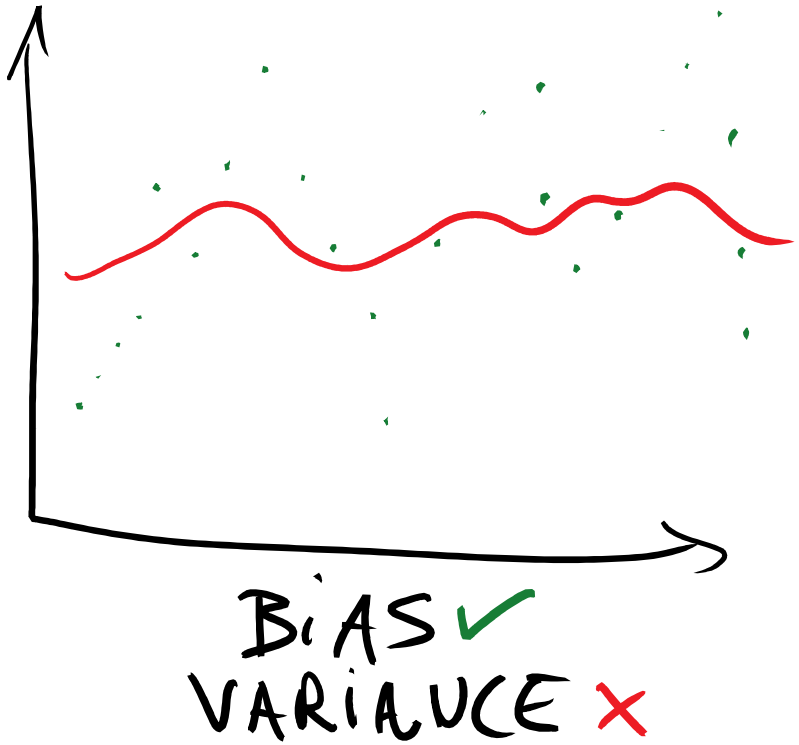
Alapvető problémák

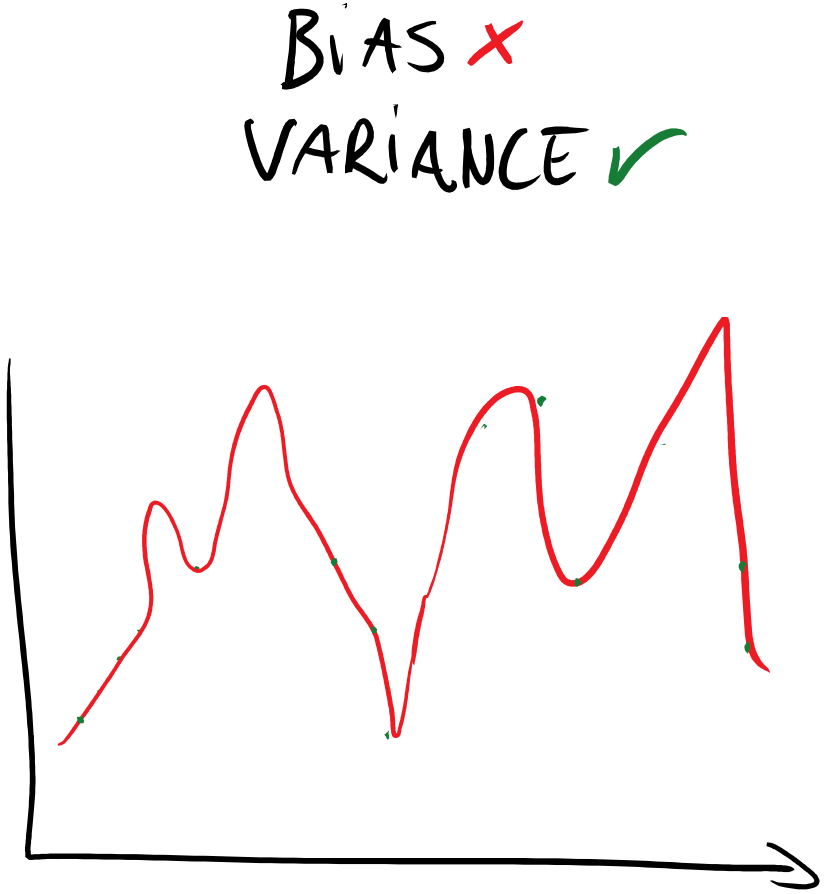
- Hatalmas hiperparaméter tér
- Bias-variance dilemma
- Nagyon lassan vagy nem konvergál
  - Adatok standardizálása és min-max skálázás
  - Mini-batch learning
  - Más költségfüggvény és optimizációs algoritmus
  - Tanítóadatok összekeverése
- Túltanulás – train-validation-test
  - Több adat
  - Early stoping
  - Regularizáció:
    - Momentum
    - L1 és L2 (Weight decay)
    - Dropout
    - Stb.

Bias-variance dilemma



Bias-variance dilemma

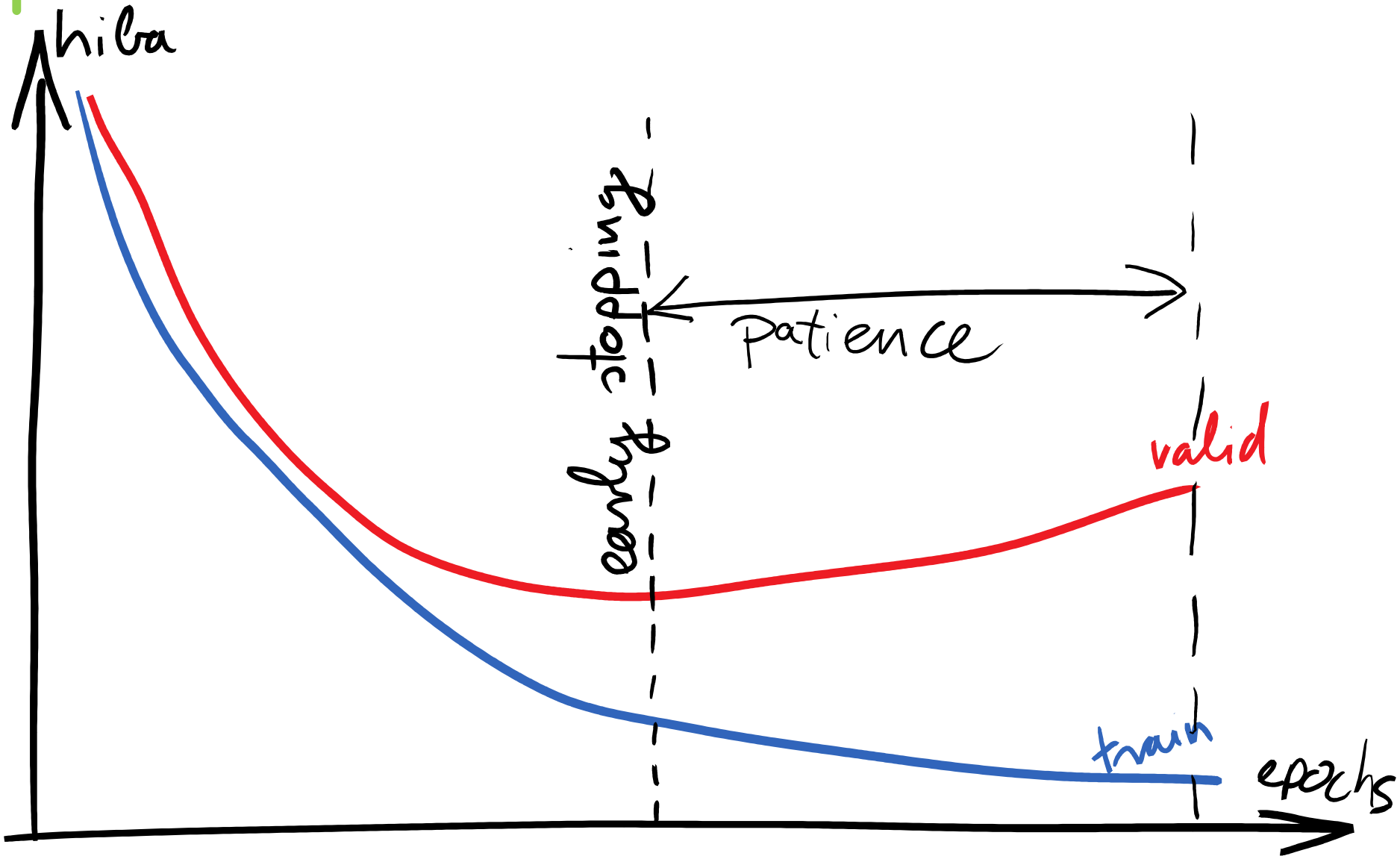




Tanító-, validációs- és tesztadatok

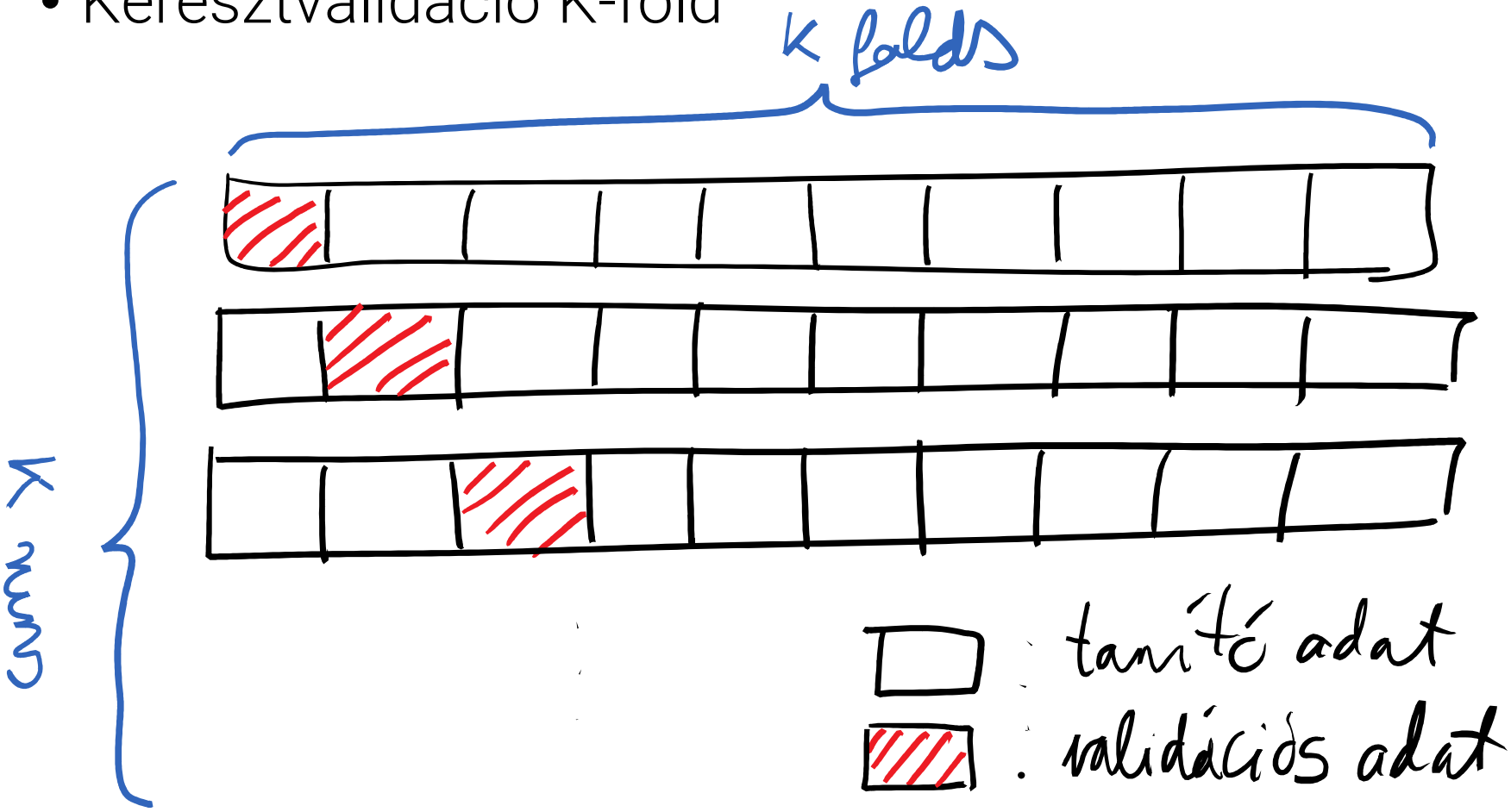
- Adatok szétszedés 3 részre:
  - Tanító, validációs és tesztadatok.
  - Saját függvénnyel vagy a `sklearn.cross_validation.train_test_split` függvényt 2x meghívni.
- Adatok összekeverése segíti a tanulást:
  - Két egymást követő minta minél távolabb legyen egymástól (pl. külön osztályból).
  - Legyen nagy a hiba a többi tanítómintához képest.
  - Véletlenszerű.
- *Early stoping after „patience" epochs*
  - Ha a validációs hiba adott lépésig nem csökken, akkor leállítjuk a tanítást és visszatöltjük a legjobb modellt.

Tanító-validációs-teszt adatok I.



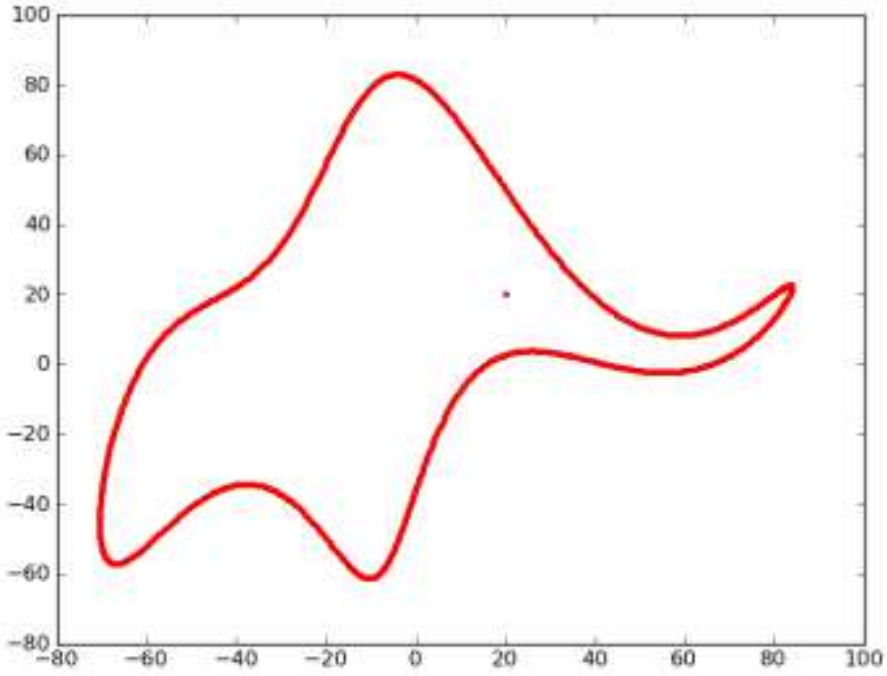
Tanító-validációs-teszt adatok III.

- 50-30-20, 70-20-10, 80-15-5
- Keresztvalidáció K-fold



Tanító-validációs-teszt adatok II.

- Túltanítás
- Neumann János: *Négy paraméterrel még egy elefántot is le lehet írni, egy ötödikkel pedig el lehet érni, hogy az ormányát is csóválja.*



Forráskód: http://www.johndcook.com/blog/2011/06/21/how-to-fit-an-elephant/

Standardizálás I.

Eltolás kivonása és szórással való osztás.

Gyakorlatban: várható érték kivonása és szórással való osztás.

Csak a tanító adatokra!!!

Max>1!!!

Feature szerint

`sklearn.preprocessing.StandardScaler` vagy:

$$X = (x - \text{mean}(X)) \,/\, \text{std}(X)$$

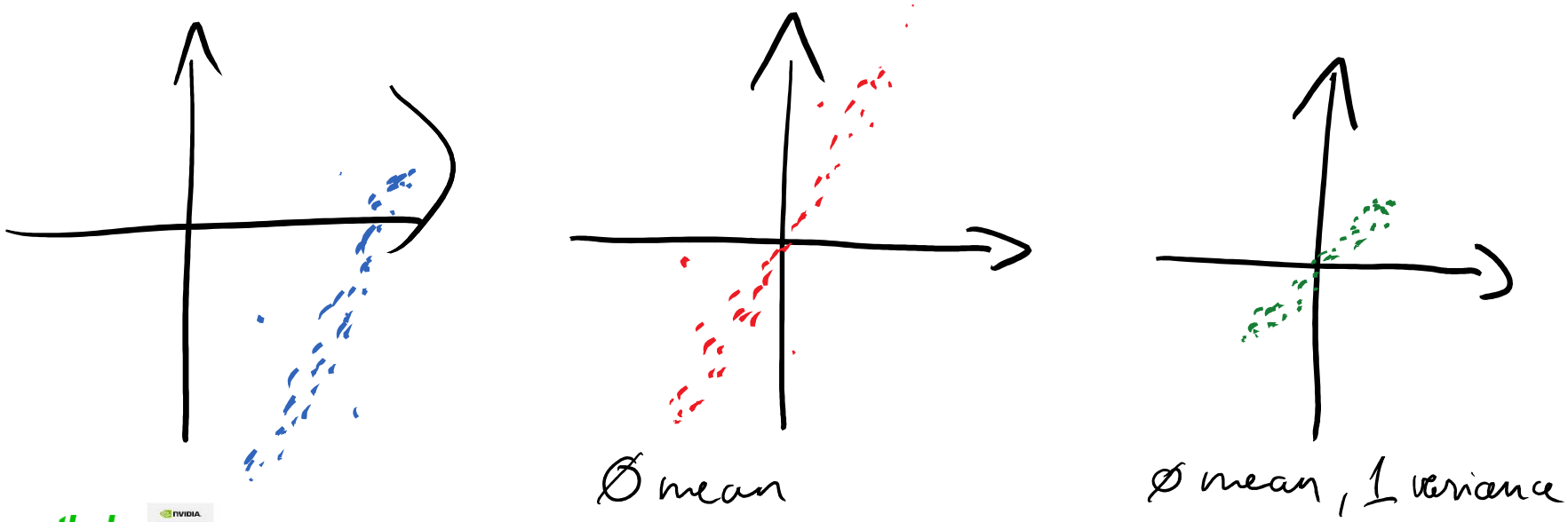
Standardizálás II.

Miért van rá szükség

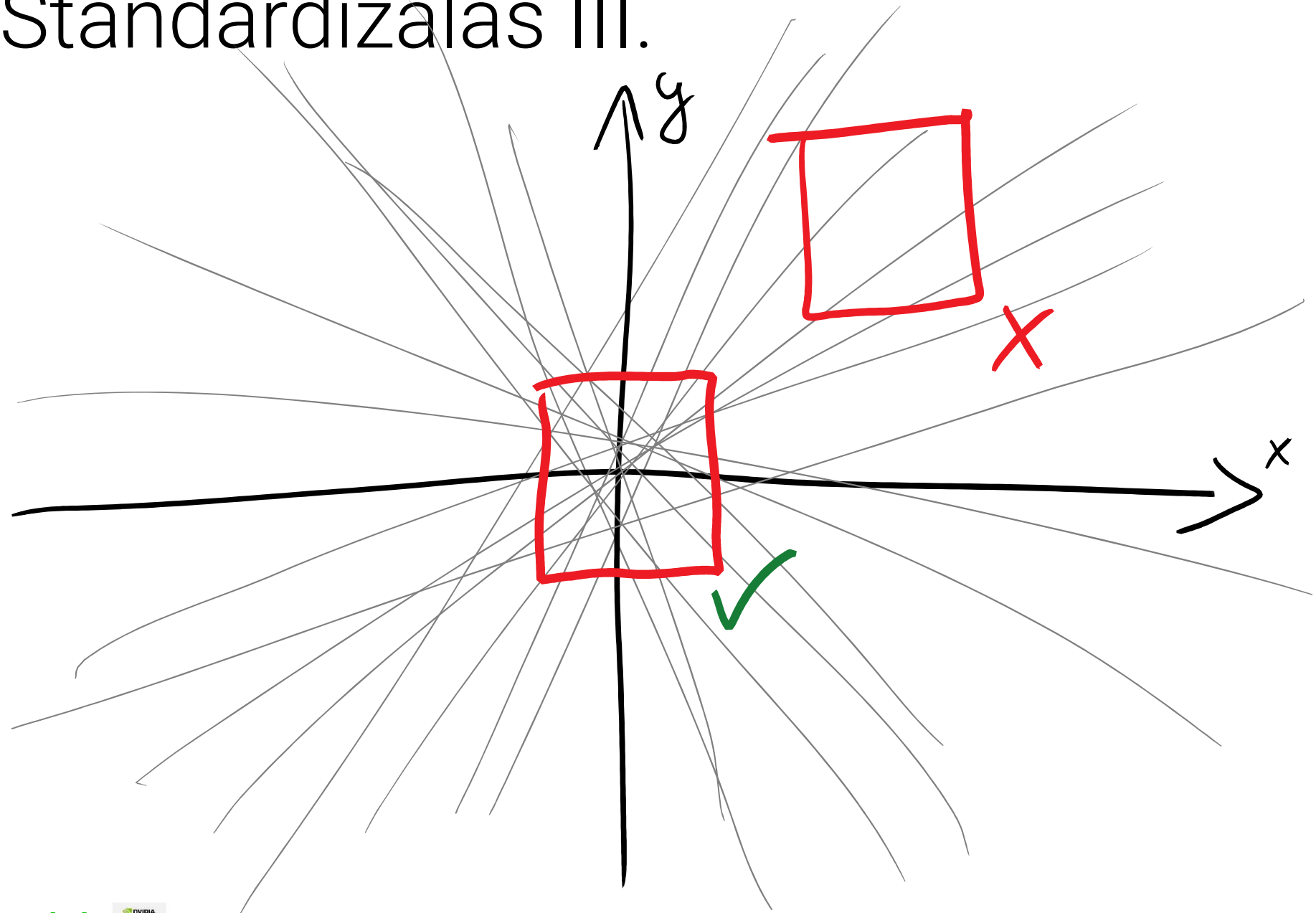
- Súlyok inicializálása: azonos nagyságrendben legyen, egyébként szaturáció lehetséges.
- A neuronokhoz tartozó hipersíkok:
  - Irányát a súlyok határozzák meg
  - Eltolást a bias határozza meg

  Kis bias inicializáláskor → jó, ha az adat is közel van az origóhoz.

  Lehetne a súlyokat és a bias-t is a bemenethez illeszteni, de sokkal bonyolultabb.



Standardizálás III.



Min-max skálázás

Kimenet átskálázása

Kimeneti aktivációs függvény sigmoid vagy tanh

„Harmóniában” a súlyokkal és a bemenettel

Probléma: *outlier*-ek.

Csak a tanító adatokra!!!

`sklearn.preprocessing.StandardScaler` vagy:

$$y\_std = (y - \min y) / (\max y - \min y)$$

$$y\_scaled = y\_std \cdot (max - min) + min$$

$max = 1$

$min = 0$

Költségfüggvények és optimizáció

- Leggyakrabban használt költségfüggvények
  - Regresszió:
    - MSE (Mean Squared Error)
  - Osztályozás:
    - Cross-entropy
  - Több költségfüggvény kombinálása a kimeneten.

- Leggyakrabban használt optimizációs algoritmusok
  - SGD (Stochastic Gradient Descent)
  - L-BFGS
  - AdaDelta
  - AdaGrad
  - ADAM
  - RMSProp

Momentum módszer

Segít elkerülni a súly „fluktuációt” és kijutni a lokális minimumokból.

0.5-0.99 közötti érték.

“Sebessége” lesz a súlyfrissítésnek, amerre a gradiens halad.

$$\Delta w_{(i)}(t) = -\mu \frac{\partial C}{\partial w_{(i)}(t)} + \alpha \cdot \Delta w_{(i)}(t-1)$$

$$w_{(i)}(t+1) = w_{(i)}(t) + \Delta w_{(i)}(t)$$

Újabb változata: Nesterov momentum.

Névnek a NEPTUN kódodat add meg!

https://kahoot.it/

L1 és L2 regularizáció

L1 regularizáció

$$C = C_0 + \lambda_1 \cdot \sum_w |w|$$

$$w^{(i)}(t+1) = w^{(i)}(t) - \mu \frac{\partial C}{\partial w^{(i)}(t)} - \mu \lambda_1 \cdot \operatorname{sgn}\left(w^{(i)}(t)\right)$$

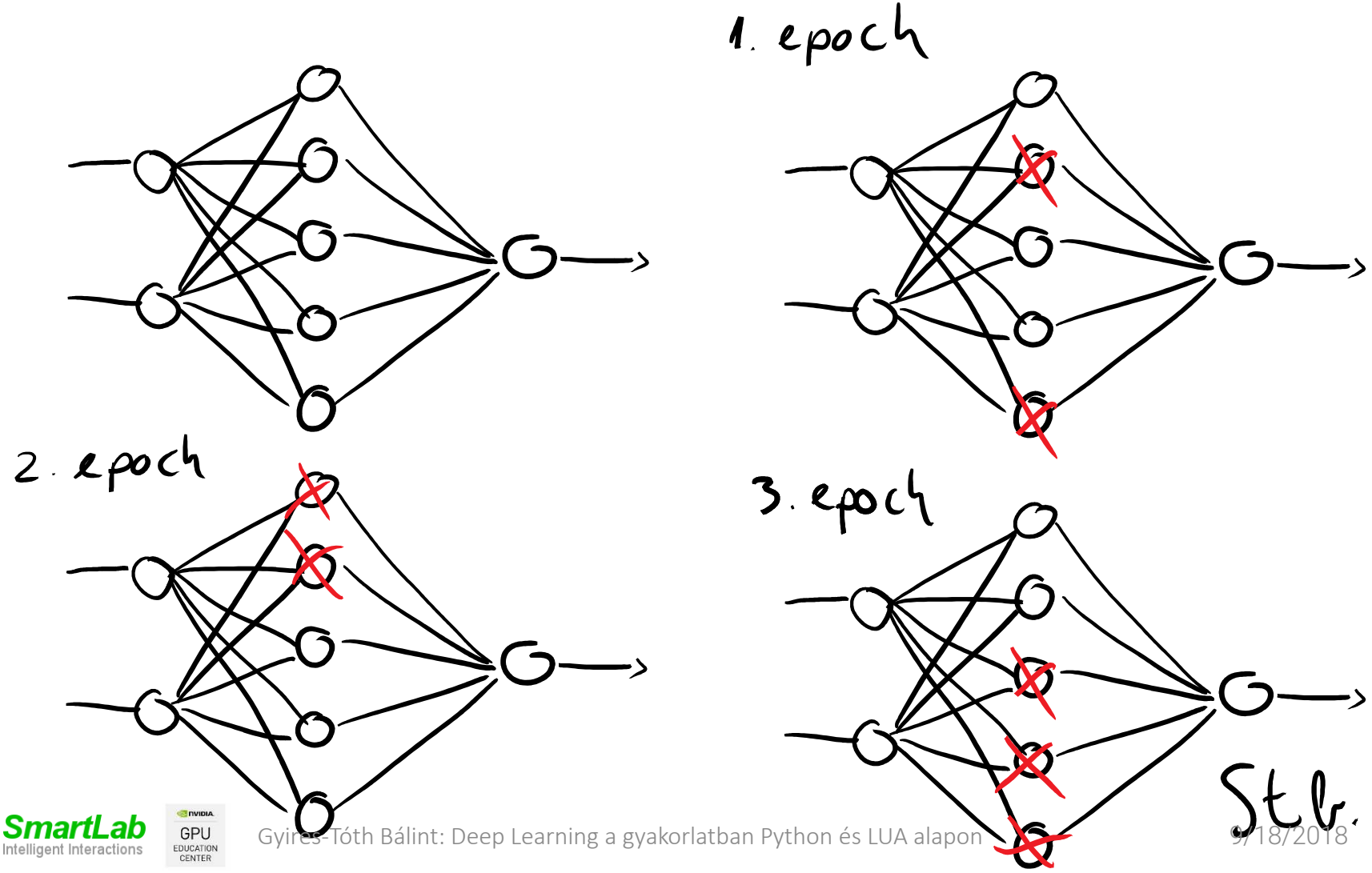
L2 regularizáció (weight decay)

$$C = C_0 + \frac{1}{2} \lambda_2 \sum_w w^2$$

$$w^{(i)}(t+1) = w^{(i)}(t) - \mu \frac{\partial C}{\partial w^{(i)}(t)} - \mu \lambda_2 \cdot w^{(i)}(t)$$

Dropout I.

Hinton prof. egyik „AHA” pillanata: banki ügyintézők.



Dropout II.

Az egyes neuronok eldobásának valószínűsége általában 30-50%.

Miért működhet?

> Olyan, mintha több hálót tanítanánk, és azok aggregált becslését átlagolnánk.
>
> Így kerüli el a túltanítást.

https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

Software I

Software II

TensorFlow (Google)

- „Theano 2.0” azonos tulajdonságokkal (negatívumokkal),
- Elosztott tanításokhoz,
- Fejlett vizualizáció,
- Mobil implementáció,
- Erős marketing és Google támogatás.

Keras (Fchollet @Google)

- Python,
- Theano és TensorFlow backend,
- TensorFlow része,
- Nagyon népszerű,
- Könnyen olvasható kód, gyors fejlesztés,
- Kaggle.com-on legtöbbet használt.

Software III

Torch7, PyTorch (Facebook, Twitter, NVidia, IBM, IDIAP, stb.)

- LUA, binárisra fordítható, leggyorsabb,
- Több szintű megvalósítás (C, LUA alacsony és magas szinten, Python),
- Jól olvasható kód,
- Mobil implementáció,
- Tudományos életben standard.

Theano (University of Montreal)

- Python
- Yosuah Bengio, folyamatos fejlesztés,
- Szimbolikus nyelv, fordítást igényel,
- Debuggolni nehézkes,
- Nehezebben olvasható kód,
- Tudományos életben standard.

theano

Software IV

Caffe (University of California, Berkeley)

- C++, képfelismerés

Kaldi

- C++, beszédfelismerés
- 2017 augusztustól TensorFlow integráció

Nervana (Intel), Lasagne, Chainer, Microsoft CNTK, Matlab Deep Learning Toolbox, stb.

Szoftverkörnyezet telepítés

GPU:

NVidia videókártya driver

CUDA, cuDNN

```
pip install tensorflow-gpu
pip install keras
```

CPU:

```
pip install tensorflow
pip install keras
```

Kis házi feladat 2.

A Numpy alapú neurális hálózat kódjába valósítsd meg a következő funkciókat:

1. Momentum
2. L1 regularizáció
3. L2 regularizáció
4. Mini-batch 16-os batch mérettel

A forráskód új részeit “# HF2 start *eljárás*” és “# HF2 end *eljárás*” (*eljárás*=momentum, l1reg, l2reg, mini-batch) kommentek közé tedd és kommentbe írd bele, hogy pontosan mit-miért csináltál.

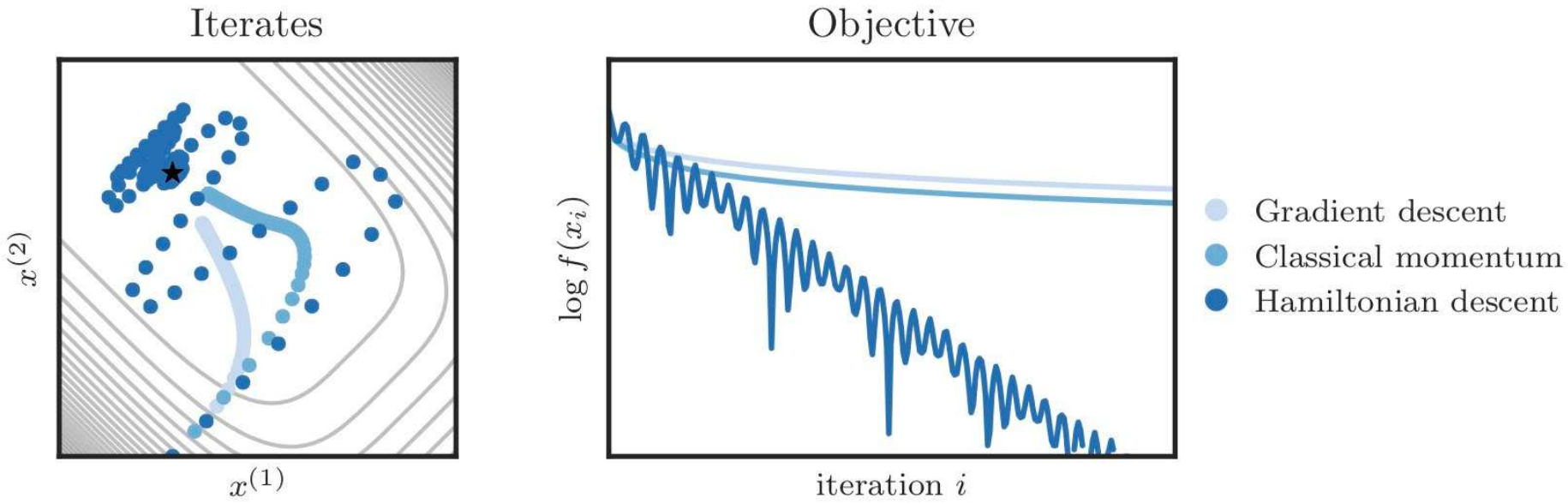
Határidő: 2018. október 2.

Deep Learning Híradó

Hírek az elmúlt 168 órából

Hamiltonian descent

https://arxiv.org/abs/1809.05042



BME TMIT

Köszönöm a figyelmet!

toth.b@tmit.bme.hu

SmartLab
Intelligent Interactions

http://smartlab.tmit.bme.hu

NVIDIA
GPU
EDUCATION
CENTER