# Perceptron és MLP

Aradi Bernadett

2017/18 ősz

# Feladatok

1. Olvassuk be egy tömbbe a
   `https://arato.inf.unideb.hu/baran.agnes/NH2017/elemek.txt`
   címen található adatállományt!
   Erre egy lehetőség:
   ```
   import urllib.request
   url='...'      # ide kell a fenti link
   points = np.loadtxt(urllib.request.urlopen(url))
   ```
2. Vizsgáljuk meg az adathalmazunk felépítését!
   1-esekkel és -1-esekkel felcímkézett síkbeli pontok szerepelnek benne.
   Hány 1-es és hány -1-es jelzésű pontunk van?
3. Ábrázoljuk egy közös koordináta-rendszerben különböző színnel az 1-es, valamint a -1-es címkével rendelkező pontokat.

Perceptronnal kettéválasztjuk a 2 ponthalmazt.

# A perceptron algoritmus – adatelőkészítés

```
import numpy as np
import urllib.request
import matplotlib.pyplot as plt

url='https://arato.inf.unideb.hu/baran.agnes/NH2017/elemek.txt'
points = np.loadtxt(urllib.request.urlopen(url))
    # vagy points=np.loadtxt('elemek.txt')

A=points[points[:,2]==1,:2]      # 1-esekkel címkézett pontok
B=points[points[:,2]==-1,:2]      # -1-esekkel címkézett pontok
print(A.shape,B.shape)      # Hány pontunk van?
```

Ábrázolás:

```
plt.plot(A[:,0],A[:,1],'bo')
plt.plot(B[:,0],B[:,1],'r*')
t=np.arange(-7.5,12.5,0.1)      # egyenes ábrázoláshoz
plt.plot(t,-2*t+10,'g')      # egy minta egyenes behúzása
plt.show()
```

# A perceptron algoritmus

```
eta = 1      # learning rate, tanulási paraméter, korrekciós tényező
R2=np.max(points[:,0]**2+points[:,1]**2)     # max norma^2
w = np.zeros((2,1))      # 0 kezdősúlyok
b=0                      # 0 kezdőtorzítás
k=0                      # számláló a korrekciós lépésekhez

x=points[:,:2]           # input tömb
y=points[:,2]            # ismert outputok

while True:
    corr=0
    for i in range(400):      # összes tanítópont
        if y[i]*(np.dot(x[i,].reshape(1,2),w)+b)<=0:
            w=(w.T+eta*y[i]*x[i,]).T
            b=b+eta*y[i]*(R2)
            k+=1
            corr=1
    if corr==0:
        break
```

# A perceptron algoritmus – az eredmény ábrázolása

```
print(k)      # hányszor történt korrekció?
print(w)      # kapott súlyvektor
print(b)      # kapott torzítás
```

Mi a kapott elválasztó egyenes egyenlete?

$$w_1 x_1 + w_2 x_2 + b = 0$$

```
t = np.arange(-7.5,16.5, 0.2)     # szeparáló egyenes ábrázoláshoz
plt.plot(A[:,0],A[:,1],'bo')
plt.plot(B[:,0],B[:,1],'r*')
plt.plot(t,-w[0,]/w[1,]*t-b/w[1,],'g')
plt.show()
```

# A perceptron algoritmus vizsgálata

1. Vizsgáljuk meg, hogyan függ a szükséges korrekciós lépések száma a tanulási paramétertől!
2. Mi történik, ha közelebb hozzuk egymáshoz a két halmazt?
   Pl. legyen $B = B + 1$, $B = B + 2$, stb.
3. Módosítsuk úgy az algoritmust, hogy az iterációk száma egy maximális érték legyen!
4. Ábrázoljuk grafikonon, hogy hogyan alakul a hiba az epochok függvényében!
   Ehhez készítsünk egy üres NumPy vektort:
   `errors =np.empty(0)`
   majd minden epoch után fűzzük hozzá a vektorhoz (például az `np.append` paranccsal) az aktuális négyzetes hibát, ami

$$\sum_{i=1}^{400} (\text{valódi output} - \text{hálózat általi output})^2.$$

Ábrázoljuk az eljárás végén kapott `errors` vektort!

# A XOR probléma

Próbáljuk meg perceptronnal megoldani a XOR problémát, azaz

- tekintsük a $(0,0)$, $(0,1)$, $(1,0)$ és $(1,1)$ pontokat,
- a pontok címkéje legyen a XOR logikai függvény által hozzájuk rendelt érték (0 vagy 1) átkonvertálva -1-re illetve 1-re,
- alkalmazzuk ezekre a pontokra a múlt órán használt perceptron algoritmust.

`http://playground.tensorflow.org`

# MLP – Többrétegű perceptron

A SciKit Learn csomag beépített MLPClassifier parancsát fogjuk használni.

```
from sklearn.neural_network import MLPClassifier
```

A függvény dokumentációja:
`http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier`
Ehhez tekintsük a Holdak.ipynb fájlban szereplő adathalmazt!
Feladat:
Írjunk egy moons nevű függvényt, amely adott $n, r, w, t$ esetén visszaad egy tömböt, amely tartalmazza a 2 hold koordinátáit és 1-es illetve 0-s címkéket!
(Ehhez jól jöhetnek az np.vstack, np.hstack, np.concatenate parancsok!)

# MLP – Regresszió

Közelítsük az $f(x) = x^3 - x$ függvényt a $[-2, 3]$ intervallumon!
A SciKit Learn csomag beépített MLPRegressor parancsát fogjuk használni.

```
from sklearn.neural_network import MLPRegressor
```

A függvény dokumentációja:
`http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html`

Próbáljuk ki az ún. grid search módszert is!
A grid search alkalmas arra, hogy több paraméterkombinációval kipróbáljuk a hálózatunkat, és a legalkalmasabb paramétereket válasszuk. Ehhez a SciKit Learn csomag beépített GridSearchCV parancsát fogjuk használni.

```
from sklearn.grid_search import GridSearchCV
```

A függvény dokumentációja:
`http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

# Regresszió – Boston dataset

- Boston starter kód
- Használjuk az MLPRegressor parancsot!
- Tanító- és teszthalmazra bontás:
  ```
  from sklearn.model_selection import train_test_split
  X_train, X_test, y_train, y_test = train_test_split(X, y,
  test_size=0.1)
  ```

Ez a dataset az UCI oldalán is megtalálható:
`https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data`
CSV beolvasása:

```
import pandas as pd
data = pd.read_csv('wine.txt',delimiter=',')
```

A wine.txt fájlt lementhetjük az alábbi linkről:
`http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data`