Laborator 3

Saptamana 11-15 martie 2019

Exercitii simple folosind API de NumPy

Nota: in rezolvarea exercitiilor se vor folosi cat mai mult functii NumPy vectorizate. Functiile scrise de voi vor fi documentate cu docstrings (tutoriale: Docstrings in Python, urmati NumPy Style Python Docstrings). Folositi type annotations. Utilizati doar pachetele NumPy si SciPy.

1. (0.5 puncte) Folosind functia `np.where`, determinati care sunt liniile si coloanele care contin un element dat intr-o matrice.
2. (0.5 puncte) Pentru o matrice de numere, sa se determine produsul elementelor pe linii.
3. (0.5 puncte) Determinati daca toate elementele unei matrice `a` sunt mai mari sau egale decat toate de pe pozitiile corespondente din `b`.
4. (0.5 puncte) Pentru un vector de numere, care sunt *toate pozitiile* pe care apare valoarea maxima? Folositi functie NumPy.
5. (0.5 puncte) Se considera vectorul: `a = np.array([230, 10, 284, 39, 76])`. Folosind indexarea logica, sa se inmulteasca elementele din vector care sunt mai mici ca 100 cu 2 (restul raman neschimbate). Folosind ciclare, sa se aplice aceeasi operatie pana cand toate elementele devin mai mari ca 100. Sa se afiseze din vectorul final elementele care sunt mai mari de 150 si mai mici de 200.
6. (0.5 puncte) Sa se scrie o functie care preia doi vectori de aceeasi lungime si returneaza maximele pe pozitiile corespunzatoare:

```
a = np.array([3, 7, 9, 13, -10, 200, 3])
b = np.array([4, 5, -9, 100, 300, 230, 1])
pair_max(a, b)
#iesire dorita:
array([  4,   7,   9, 100, 300, 230,   3])
```

7. (1 punct) Sa se calculeze suma elementelor non-NaN intr-un 2d numpy array, in 3 feluri. Folosind comanda magica `%%timeit` Built-in magic commands (http://ipython.readthedocs.io/en/stable/interactive/magics.html#cell-magics), IPython Magic Commands (https://jakevdp.github.io/PythonDataScienceHandbook/01.03-magic-commands.html), sa se determine care e mai rapida.
   A. Peste o copie a tabelului initial (functia membru `copy()` a unui tablou) se suprascriu elementele nan cu 0, apoi se face suma;
   B. Folosind functia `np.nansum`;
   C. Folosind indiciere booleana si functia `np.isnan`, se determina colectia elementelor non-nan si se calculeaza suma lor in mod obisnuit.
8. (1 punct) Incarcati setul de date Iris, de la adresa 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data (https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data)', folosind `np.genfromtxt`. Alegeti aleator 10 pozitii in matricea de 150 linii si 4 coloane (omiteti ultima coloana, de tip text), setati aceste valori pe NaN. Construiti o functie care, primid la intrare o matrice, returneaza un tuplu cu indicii de linie respectiv de coloana in care se gasesc valori NaN.
9. (1 punct) Construiti o functie care, primind o matrice, determina pe ce pozitii se afla valorile in afara unui interval [min, max] dat prin parametri.
10. (1 punct) Sa se construiasca o functie care returneaza cele mai mari `k` valori dintr-un vector de cel putin `k` elemente, impreuna cu pozitiile lor.

Precizari:

1. Se acorda 3 puncte din oficiu.
2. Tema se va prezenta in sapatamana 18-22 martie.
3. Modalitatea de predare a temei:
   A. se vor forma si se va lucra in echipe de cate doua persoane (in cazul in care nu se poate, se va lucra individual)
   B. exercitiile rezolvate se vor comite pe un repository **privat** de Github (https://github.com/), la care vor avea doar acces membrii echipei si profesorii, Lucian Mircea Sasu (cont github: lmsasu) respectiv Arpad Kerestely (cont github: akerestely)
   C. repository-ul va contine un fisier "readme.md" (in radacina repository-ului) in care vor fi mentionati autorii temelor, cu nume complet si grupa din care fac parte.