Numpy

Cours M2 CCN

Modélisation des problèmes scientifiques

2019/2020

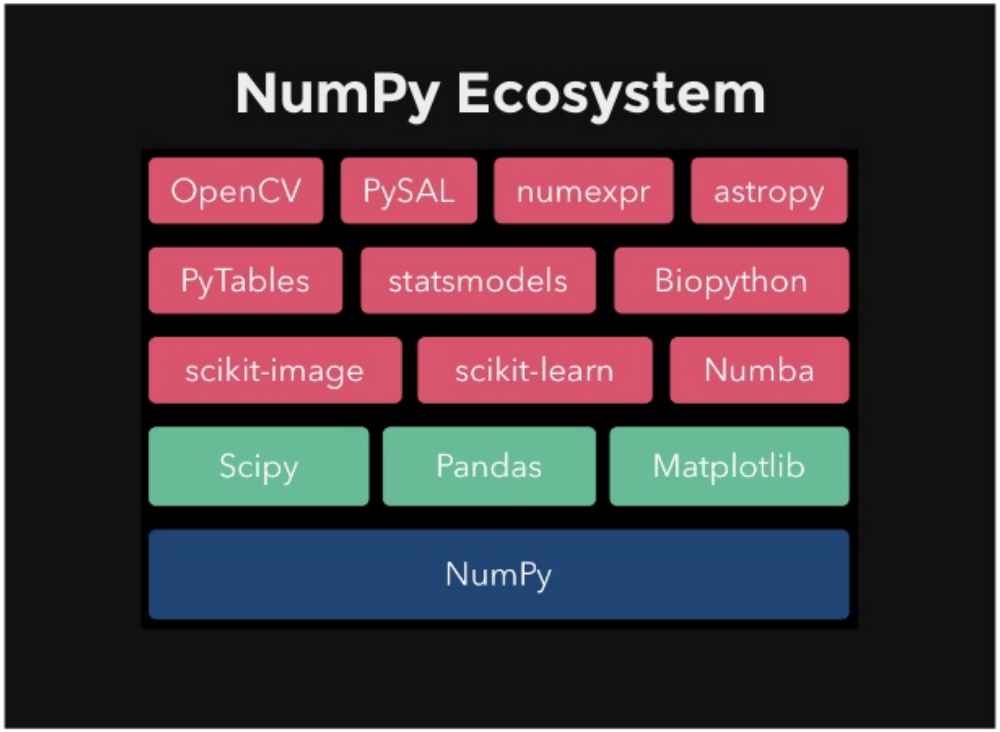
Qu’est-ce que c’est?

- Librairie python dédiée au traitement des tableaux et des matrices

Pourquoi utiliser les tableaux de numpy plutôt que les objets natifs de python, comme les listes ou les tuples ?

→ accès et traitements optimisés

→ des fonctions déjà codées



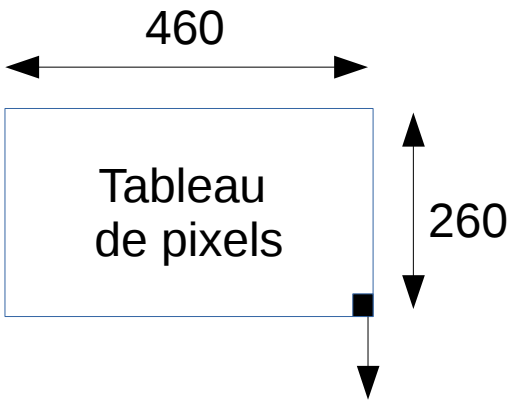
D’autres librairies basées sur numpy

Concrètement, pourquoi travailler sur des tableaux ?

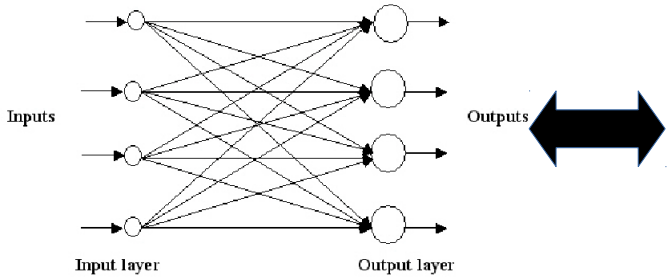
Image (r,g,b)

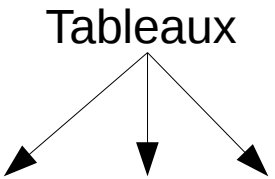
de résolution

460*260 pixels



Traitement d'images





Ouputs = Inputs*weights

Optimiser le calcul matriciel permet de converger plus rapidement vers la solution

Machine learning

Prénom	Age	Taille
Adele	45	168
Alex	30	182
Alice	28	153
Bob	52	161
...	...	...

Tableau

Analyse de données

On démarre avec numpy !

Import de la librairie

```
Entrée [1]: import numpy as np
```

Définition de tableau

```
Entrée [2]: np.array([1,2,3])
    Out[2]: array([1, 2, 3])
```

Mais on peut aussi spécifier un type dans la définition de tableaux

```
Entrée [3]: np.array([1,2,3], dtype= str)
            #Par exemple, ce tableau est un tableau de chaînes de caractères
    Out[3]: array(['1', '2', '3'], dtype='<U1')

Entrée [4]: np.array([1,2,3], dtype= float)
            # Alors que celui-ci est un tableau de flottants
    Out[4]: array([1., 2., 3.])
```

```
Entrée [5]: np.array([1,2,3], dtype= np.int8)
    Out[5]: array([1, 2, 3], dtype=int8)

Entrée [6]: np.array([1,2,3], dtype= np.float32)
    Out[6]: array([1., 2., 3.], dtype=float32)
```

⟺

```
Entrée [1]: import numpy

Entrée [2]: numpy.array([1,2,3])
    Out[2]: array([1, 2, 3])
```

Ici, [1,2,3] est une liste python, mais la conversion marche aussi avec le tuple (1,2,3).

Un tableau est homogène en type, tous ses éléments doivent avoir le même type.

Numpy a défini des types spécifiques, qui permettent d'adapter l'espace mémoire alloué au tableau.

Par exemple, tout élément int8 sera codé sur un octet seulement, mais ne doit pas dépasser 255 =2^8-1.

Attributs des tableaux

On crée un tableau monTableau

Entrée [7]:
```
monTableau = np.array([32,45,68,54])
monTableau
```

Out[7]:
```
array([32, 45, 68, 54])
```

Ce tableau a différents attributs

Entrée [8]:
```
# Longueur du tableau (pour une matrice, le nombre de lignes)
len(monTableau)
```

Out[8]:
```
4
```

Entrée [9]:
```
# la dimension du tableau (nbLignes, nbColonnes)
monTableau.shape
```

Out[9]:
```
(4,)
```

Entrée [10]:
```
#le type d'éléments qu'il contient
monTableau.dtype
```

Out[10]:
```
dtype('int32')
```

Entrée [11]:
```
# accéder au i-ème élément d'un tableau : monTableau[i]
monTableau[0]
```

Out[11]:
```
32
```

Entrée [12]:
```
# accéder aux i premiers éléments : monTableau[0:i] ou monTableau[:i]
monTableau[:2]
```

Out[12]:
```
array([32, 45])
```

Entrée [13]:
```
# accéder aux éléments entre les indices i et j (i inclus, j exclu): monTableau[i:j]
monTableau[1:3]
```

Out[13]:
```
array([45, 68])
```

Le premier élément
d'un tableau est indexé à 0

Pour remplacer la première valeur du tableau (par exemple par 2) :

monTableau[0] = 2

Pour supprimer la première valeur :

delete monTableau[0]

Code : Fizzbuzz

- Créez un tableau numpy en python qui contiendra les nombres de 1 à 50. Mais remplacez les valeurs par « fizz » pour tous les indices multiples de 3, par « buzz » pour tous les multiples de 5, et par « fizzbuzz » pour les multiples de 15. Affichez le tableau final !

- Parfois utilisé comme test technique en entretien d'embauche

Fonctions des tableaux

J'aimerai générer un tableau de nombres aléatoires

```
Entrée [14]: # Génère un tableau de 5 entiers aléatoires entre 0 et 10
             monTableau = np.random.randint(0,10,5)
             monTableau
Out[14]: array([4, 0, 6, 8, 5])
```

J'aimerai que monTableau soit trié...

```
Entrée [15]: np.sort(monTableau)
Out[15]: array([0, 4, 5, 6, 8])
```

Quelle est sa valeur minimale?

```
Entrée [16]: np.min(monTableau)
Out[16]: 0
```

Et on la trouve à quel indice?

```
Entrée [17]: np.argmin(monTableau)
Out[17]: 1
```

Marche de manière similaire avec max et argmax

Quelle est sa moyenne?

```
Entrée [18]: np.mean(monTableau)
Out[18]: 4.6
```

np.average donne aussi la moyenne (oui, c'est bizarre!)

Quelle est sa variance? Son écart-type?

```
Entrée [19]: np.var(monTableau)
             np.std(monTableau)
Out[19]: 2.65329983228432
```

Percentile

```
Entrée [20]: np.percentile(monTableau, 50)
Out[20]: 5.0
```

Filtrer certaines valeurs du tableau

```
Entrée [21]: monTableau[monTableau<5]
Out[21]: array([4, 0])
```

Soit $x = (x_1, \dots, x_n)$ un tableau de n valeurs

La moyenne $\bar{x} = \frac{1}{n}\sum_{i=0}^{n} x_i$

La variance $Var(x) = \frac{1}{n}\sum_{i=0}^{n} (x_i - \bar{x})^2$

L'écart-type $\sigma(x) = \sqrt{(Var(x))}$

Le percentile d'ordre alpha : s tel que $P(x < s) = \alpha$

Le percentile d'ordre alpha d'une variable correspond à la valeur minimale pour laquelle au moins alpha des données sont supérieures ou égales à cette valeur. Par exemple, la médiane Q2 correspond au percentile d'ordre 50%.

Ne vous endormez pas !

Que rendent ces fonctions ?

```
def enigma():
    tab = np.array([41,16,10,8,1])
    position = np.argmin(tab)
    tab = np.sort(tab)
    return np.min(tab)+tab[position]
```

```
def secret():
    tab = np.array([k*5 for k in range(10)])
    tab = tab[tab<25]
    return np.mean(tab)+len(tab)
```

Pour finir sur les tableaux

Définition simplifiée de tableaux

Entrée [26]:
```
print(np.zeros(10)) # crée un tableau de 10 zéros
print(np.ones(8)) # crée un tableau de 8 fois 1
print(8*np.ones(9)) # crée un tableau de 9 fois 8
print(np.arange(1, 35, 2)) # tableau des impairs jusqu'à 33
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[1. 1. 1. 1. 1. 1. 1. 1.]
[8. 8. 8. 8. 8. 8. 8. 8. 8.]
[ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33]
```

np.arange(min, max, step)

Concaténation de deux tableaux

Entrée [27]:
```
num = np.array(['n','u','m'])
py = np.array(['p','y'])

conc = np.concatenate([num,py]) # tableau composé des lettres de num et des lettres de py

conc
```

Out[27]:
```
array(['n', 'u', 'm', 'p', 'y'], dtype='<U1')
```

Opération sur des tableaux

Entrée [28]:
```
a = np.array([1]*5+[2]*10, dtype=np.int32)

print("a = ", a)
print("a + a = ",np.add(a,a))
print("4*a - a = ",np.subtract(4*a,a))
print("a / 4 = ",np.divide(a,4*np.ones(len(a))))
```

```
a =  [1 1 1 1 1 2 2 2 2 2 2 2 2 2 2]
a + a =  [2 2 2 2 2 4 4 4 4 4 4 4 4 4 4]
4*a - a =  [3 3 3 3 3 6 6 6 6 6 6 6 6 6 6]
a / 4 =  [0.25 0.25 0.25 0.25 0.25 0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
 0.5 ]
```

a + a donne le même résultat

Plongée dans le code source

Les fonctions appelées dans le code python correspondent à du vrai code, présent sur votre machine !

Le plus souvent, avant de recoder une fonction, il vaut mieux vérifier qu'une librairie ne propose pas déjà une implémentation.

Dans le cas d'une installation python simple, les différents modules sont disponibles :

- Pour Windows : C:\Python3x\Lib\site-packages\

- Pour Linux : /usr/local/lib/python3.x/dist-packages/

(Remplacez x par votre version, 3 par 2 si vous utilisez python2)

Cas particulier pour Anaconda ;

Pour Windows : C:\Users\NAME\Documents\Anaconda3\Lib\site-packages\

Pour Linux : /home/NAME/anaconda3/lib/python3.x/site-packages/

NAME à remplacer par votre nom d'utilisateur

Les notions à approfondir sur numpy

- Voir les fonctions de math (np.log, np.sin)
- Voir les matrices
- S’informer sur les types, sur le format npy
- S’intéresser aux fonctions de chargement de données (np.loadtxt)

Pour approfondir sur numpy

- Le manuel officiel https://docs.scipy.org/doc/numpy/reference/
- Une présentation générale https://www.slideshare.net/PyData/introduction-to-numpy
- https://eric.univ-lyon2.fr/~ricco/cours/slides/PG%20-%20en%20-%20numpy%20vectors.pdf
- Un script sur le github qui explique quelques fonctions de numpy
- Des exos! → https://github.com/rougier/numpy-100
- Le mieux, c'est encore de tester toutes les fonctions par vous-même ! Si vous avez besoin d'aide sur une fonction, vous pouvez taper help(le_nom_de_ma_fonction) sur python.