numpy

Introduction à l'apprentissage automatique
Master Sciences Cognitives
Aix Marseille Université

Alexis Nasr

Tableaux

- Les objets principaux manipulés par `numpy` sont les tableaux multidimensionnels.
- Ces derniers sont des tableaux de nombres d'un même type, indexés par un tuple d'entiers positifs.
- En numpy les dimensions sont appelés axes.
- Un vecteur, par exemple, possède un axe, tandis qu'une matrice possède deux axes.
- Les tableaux sont des instances d'une classe appelée `array` (ou encore `ndarray`).

La classe `array`

Les principales variables d'instances de `array` sont :

- `ndim` : nombre d'axes (de dimensions) du tableau.
- `shape` : dimensions du tableau. Il s'agit d'un tuple d'entiers qui indiquen la taille du tableau pour chacune de ses dimensions. Pour une matrice de de l lignes et c colonnes, `shape` aura pour valeur `(l,c)`. Le nombre d'éléments du tuple est donc égal au nombre d'axes (`ndim`).
- `size` : nombre total d'éléments d'un tableau. Ce nombre est égal au produit des éléments de `shape`
- `dtype` : objet décrivant le type des éléments du tableau. On peut utiliser les types standard de Python, ou des types spécifiques à `numpy`, tel que `numpy.int32`, `numpy.int16`, ou `numpy.float64`.
- `itemsize` : taille en octets de chaque élément d'un tableau. Un tableau dont les éléments sont de de type `float64` aura 8 pour valeur de `itemsize`.

Exemples

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int64'
>>> a.itemsize
8
>>> a.size
15
>>> type(a)
<type 'numpy.ndarray'>
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<type 'numpy.ndarray'>
```

Création de tableaux

Il y a plusieurs façons de créer des tableaux.

- A partir d'un liste ou d'un tuple Python à l'aide de la fonction `array`.
- A partir d'une valeur donnée, en précisant les dimensions du tableau
- A partir de la fonction `arange` et de la méthode `reshape`.

Création d'un tableau à partir d'une liste python

- Le type du tableau résultant est déduit du type des éléments de la séquence.

```
>>> import numpy as np
>>> a = np.array([2,3,4])
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int64')
>>> b = np.array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
```

- la fonction `array` transforme des listes de listes en tableaux bi-dimensionnels, les listes de listes de listes en tableaux tri-dimensionnels et ainsi de suite.

```
>>> b = np.array([(1.5,2,3), (4,5,6)])
>>> b
array([[ 1.5,  2. ,  3. ],
       [ 4. ,  5. ,  6. ]])
```

- Le type d'un tableau peut être spécifié de manière explicite au moment de la création :

```
>>> c = np.array( [ [1,2], [3,4] ], dtype=complex )
>>> c
array([[ 1.+0.j,  2.+0.j],
       [ 3.+0.j,  4.+0.j]])
```

Création de tableaux garnis d'une valeur donnée

- Dans de nombreux cas, la taille d'un tableau est connue sans que le soient la valeur de ses éléments.
- Il est possible dans ce cas de créer des tableaux contenant des éléments dont la valeur est donnée.
- La fonction `zeros`, crée un tableau dont tous les éléments valent zéro
- La fonction `ones`, crée un tableau dont tous les éléments valent un
- la fonction `empty` crée un tableau dont la valeur des éléments est aléatoire (dépend du contenu de la mémoire au moment de la création du tableau).
- Le type par défaut des tableaux ainsi créés est `float64`.

Création de tableaux garnis d'une valeur donnée

```
>>> np.zeros( (3,4) )
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
>>> np.ones( (2,3,4), dtype=np.int16 )
array([[[ 1, 1, 1, 1],
        [ 1, 1, 1, 1],
        [ 1, 1, 1, 1]],
       [[ 1, 1, 1, 1],
        [ 1, 1, 1, 1],
        [ 1, 1, 1, 1]]], dtype=int16)
>>> np.empty( (2,3) )
array([[  3.73603959e-262,   6.02658058e-154,   6.55490914e-260],
       [  5.30498948e-313,   3.14673309e-307,   1.00000000e+000]])
```

La fonction arange

Il est possible de créer des séquences de nombres, à la manière de `range` de Python, qui renvoie des tableaux plutôt que des listes.

```
>>>
>>> np.arange( 10, 30, 5 )
array([10, 15, 20, 25])
>>> np.arange( 0, 2, 0.3 )
array([ 0. ,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8])
```