ALGORITMA DAN PEMROGRAMAN

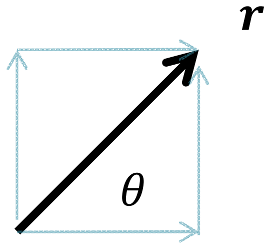
2:42:35 PM

KULIAH 9 :Vektor dan Numpy

Dosen Pengampu:
Hasanuddin, S.Si., M.Si., Ph.D.

Vektor

Dalam fisika, vektor adalah suatu besaran yang memiliki besar dan arah.



Representasi vektor:

Polar:

$$\boldsymbol{r} = (r, \theta)$$

Kartesian:

$$\boldsymbol{r} = (r\cos\theta\,, r\sin\theta) = (x, y)$$

Vektor Abstrak

Vektor 3 D

$$r = (x, y, z)$$

Vektor Abstrak

$$r = (x_1, x_2, x_3, x_4, \dots, x_n)$$

Operasi terhadap vektor

(a) Penjumlahan

(b) Perkalian dengan skalar

(c) Perkalian titik

(d) Perkalian silang (secara fisis, untuk vektor 3D)

Operasi pada Vektor

Penjumlahan

$$\boldsymbol{a} = (a_1, a_2, a_3)$$

$$\boldsymbol{b} = (b_1, b_2, b_3)$$

$$\boldsymbol{a} + \boldsymbol{b} = (a_1 + b_1\ , a_2 + b_2, a_3 + b_3)$$

Pengurangan

$$\boldsymbol{a} - \boldsymbol{b} = (a_1 - b_1, a_2 - b_2, a_3 - b_3)$$

Perkalian dengan skalar

$$k\boldsymbol{a} = (ka_1, ka_2, k_{a3})$$

Operasi pada Vektor

Perkalian titik

$$\boldsymbol{a} \cdot \boldsymbol{b} = (a_1 b_1 + a_2 b_2 + a_3 b_3)$$

Perkalian silang

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

$$= (a_2 b_3 - a_3 b_2, a_3 b_1 - a_1 b_3, a_1 b_2 - a_2 b_1)$$

Representasi vektor dalam Program

Sejauh ini, kita ketahui list:

$$\boldsymbol{a} = (a_1, a_2, a_3)$$

$$\boldsymbol{a} = (1,2,3)$$

Dalam list

```
a = [1,2,3]
b = [2,0,0]
c = a + b
Menghasilkan c=[1,2,3,2,0,0]
```

Adding/Subtracting two vectors

```
def add(a,b):
    c = []
    for i in range(len(a)):
        c.append(a[i]+b[i])
    return c
def sub(a,b):
    c = []
    for i in range(len(a)):
        c.append(a[i]-b[i])
    return c
x = add(a,b);
```

Guarding same length

```
def add(a,b):
     n = len(a)
     assert (n==len(b)),”
different size“
     c = []
     for i in range(n):
         c.append(a[i]+b[i])
     return c
```

scalar

```
def scalar_mult(k,a):
    c = []
    for i in range(len(a)):
        c.append(k*a[i])
    return c
```

```
>>> x = [1,2,3]; k = -0.5;
>>> v = scalar_mult(k,x)
```

Perkalian titik

```
def dot(a,b):
    n = len(a)
    assert (n==len(b))
    hasil_dot = 0
    for i in range(n):
        hasil_dot += (a[i]*b[i])
    return hasil_dot
>>> a = [1,2,3]; b = [2,1,2];
>>> print(dot(a,b))  # 2+2+6 = 10
```

Perkalian Silang

```
def cross(a,b):
    n = len(a)
    assert (n==len(b)and n==3 ),”pesan“
    … # latihan
```

Magnitudo

Besar vektor: $a = \sqrt{\boldsymbol{a}.\boldsymbol{a}}$

```
from math import sqrt

def mag(a):
    asquare = dot(a,a)
    return sqrt(asquare)
```

Storing a vector in list

Vektor dapat disimpan atau dinyatakan sebagai list tetapi list memiliki beberapa kekurangan.

(1) tidak memiliki operasi aritmatika (*, +,^, …)

(2) tidak effisien untuk menyimpan data yang banyak dan multi-dimensi

numpy

- Numpy adalah module dalam python.

```
import numpy
import numpy as np
```

- Numpy hampir sama dengan list. Tetapi panjang dan tipenya tetap.
- Object yang dibangun oleh numpy dinamakan array.

Membangun Array dari List

```
import numpy as np
#from numpy import *
x = [1,2,3]
x = np.array(x)
# atau secara langsung
y = np.array([1.2, 3.33, -1.0])
z = np.array([4,5],float)
a = np.array([1,2],int)
```

Panjang array

Array ~ list

```
>>> x = np.array([1,2,3])
>>> len(x)
3
>>> type(x)
<class ‘numpy.ndarray’>
>>> type([1,2,3])
<class ‘list’>
```

Membuat Array dari List Comprehension

```
>>> x = [k for k in range(1,101)]
>>> # x= [1,2,3,4,…100]
>>> y = [k for k in range(1,101,2)]
>>> # y = [1,3,5,…99]
>>> X = np.array(x)
>>> Y = np.array(y)
>>> a = np.array([1,2,3])
>>> b = np.array([2,1,0])
>>> c = a + b  # c=array([3,3,3])
```

Membuat array dengan metode dalam Numpy

```
>>> x = np.linspace(0,10,101)
# x = array([0., 0.1, 0.2, … 10.])

>>> x = np.arange(0,10.1,0.1)
# x = array([0.,0.1,0.2,…, 10.])

>>> z = np.zeros(5,dtype=float)
# z = array([0.,0.,0.,0.,0.])
>>> o = np.ones(5, dtype=int)
# o = array([1, 1, 1, 1, 1])
>>> y = np.zeros_like(x)
# y = array([0.,0.,0.,…,0.])
>>> u = np.ones_like(x)
# u = array([1.,1.,1.,…,1.])
```

Aritmatika Array

```
>>> a = np.array([1,2,3])
>>> b = np.array([5,2,6])
>>> a + b
 array([6,4,9])
>>> a – b
 array([-4, 0 , -3])
>>> a * b
 array([5, 4, 18])
>>> b / a
 array([5, 1, 2])
>>> a % b
 array([1, 0, 3])
>>> b**a
 array([5, 4, 216])
```

Aritmatika Array dengan Skalar

```
>>> a = np.array([1,2,3])
>>> 2*a
 array([2, 4, 6])
>>> a/2
 array([0.5, 1, 1.5])
>>> a**2
 array([1, 4, 9])
>>> a+2
 array([3, 4, 5])
>>> b = np.array([3,4])
>>> a + b
 ValueError : shape mismatch.
```

Others Array Methods

```
>>> a = np.array([2,4,3])
>>> a.sum()
 9
>>> a.prod()
 24
>>> a.mean()
 3
>>> a.min()
 2
>>> a.max()
 4
>>> a.std()
 0.816…
>>> a.size
 3
```

Fungsi dalam Numpy

```
>>> a = np.array([2,4,3])
>>> np.sum(a)
 9
>>> np.prod(a)
 24
>>> np.mean(a)
 3
>>> np.min(a)
 2
>>> np.max(a)
 4
>>> np.std(a)
 0.816…
>>> np.size(a)
 3
```

Fungsi len, konstanta pi dan e

```
>>> a = np.array([2,4,3])
>>> len(a)
 3
>>> np.pi
 3.141592653589793
>>> np.e
 2.718281828459045
```

Array Iteration

```
>>> a = np.array([1,2,3])
>>> for e in a:
        print(e, end=‘ ‘)
  1 2 3
>>> for i in range(len(a)):
        print(a[i], end=‘’)
  123
>>> a[1] = 5
>>> a
  array([1,5,3])
```

Slicing Array = Slice List

```
>>> a = np.array([1,2,3,4])
>>> a[:]
 array([1,2,3,4])
>>> a[:2]
 array([1,2])
>>> a[2:]
 array([3,4])
>>> a[1:3]
 array([2,3])
```

Array Functions

```
>>> x = np.pi* np.array([1,2,3,4])/4
 array([ 0.78539816,  1.57079633, 2.35619449,
3.14159265])
>>> y = np.sin(2*x)
 array([  1.00000000e+00,   1.22464680e-16,
-1.00000000e+00,-2.44929360e-16])
>>> # cos, tan, sinh, cosh, tanh, exp, log,
log10, sqrt, sign, arcsin, arccos, arctan,
arcsinh, arccosh, arctanh
```

Bilangan Acak dalam Numpy

```
>>> np.random.seed(123)
>>> np.random.rand(3) #[0,1)
array([ 0.69646919,  0.28613933,
0.22685145])
>>> np.random.rand(3)
array([ 0.55131477,  0.71946897,
0.42310646])
>>> np.random.rand(3)
array([ 0.9807642 ,  0.68482974,
0.4809319 ])
>>> np.random.random() #[0,1)
 0.3921175181941505
```

Kuliah 11: Array berdimensi N

Hasanuddin

09-11-2021

ND –Array (Creation)

```
>>> import numpy as np
>>> L = [[1,2,3],[4,5,6],[7,8,9]]
>>> a  = np.array(L)
>>> a.shape
(3,3)
>>> np.shape(a)  # (3,3)
>>> a.dtype()    # dtype(‘int32’)
>>> a[0]   # array([1,2,3])
>>> a[1,2] # 6
>>> a[1,1:3] # array([5,6])
>>> a[:,1]   # array([2,5,8])
```

ND –Array (Slice)

```
>>> a[1,2] = 7
>>> print(a)
[[1 2 3]
 [4 5 7]
 [7 8 9]]
>>> a[:,0] = [0,9,3]
>>> print(a)
[[0 2 3]
 [9 5 7]
 [3 8 9]]
>>> b = np.zeros((3,3))
>>> b
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
>>> b=b.reshape(9)
array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
```

3D –Array (Boolean Index)

```
>>> c = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> c
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
>>> c_b = c%2 ==0
>>> c_b
array([[False,  True, False],
       [ True, False,  True],
       [False,  True, False]], dtype=bool)
>>> c[c_b]
array([2, 4, 6, 8])
>>> c.T
array([[1, 4, 7],
       [2, 5, 8],
       [3, 6, 9]])
```

3D –Array (Arithmetic Operation)

```
>>> c = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> c
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
>>> d = np.ones((3,3),dtype=‘int32’)
>>> e = c+d
array([[ 2,  3,  4],
       [ 5,  6,  7],
       [ 8,  9, 10]])
>>> c*e
array([[ 2,  6, 12],
       [20, 30, 42],
       [56, 72, 90]])
```

3D –Array (Matrix Multiplication)

```
>>> d =
np.array([[1,0,2],[1,3,6],[4,5,2]])

>>> c
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
>>> c@d
array([[15, 21, 20],
       [33, 45, 50],
       [51, 69, 80]])
```

3D – Array (Function)

```
>>> c
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
>>> c*np.pi/3
array([[ 1.04719755,  2.0943951 ,  3.14159265],
       [ 4.1887902 ,  5.23598776,  6.28318531],
       [ 7.33038286,  8.37758041,  9.42477796]])
>>> c*np.pi/3*180/np.pi
array([[  60.,  120.,  180.],
       [ 240.,  300.,  360.],
       [ 420.,  480.,  540.]])
>>> np.sin(c*np.pi/3)
array([[  8.66025404e-01,   8.66025404e-01,   1.22464680e-16],
       [ -8.66025404e-01,  -8.66025404e-01,  -2.44929360e-16],
       [  8.66025404e-01,   8.66025404e-01,   3.67394040e-16]])
```

Recarray

Digunakan untuk menggabungkan beberapa tipe data

```
>>> x = np.arange(100)
# array([0, 1, 2, … , 99])
>>> y = np.sqrt(x)
# array([0. ,1. ,1.41421356, … , 9.94987437]
>>> z = y.astype(np.int)
#array([0, 1, 1, … , 9])
>>> r =
np.rec.array((x,y,z),names=(‘x’,’y’,’z’))
>>> r.x
>>> r.y
>>> r.z
```