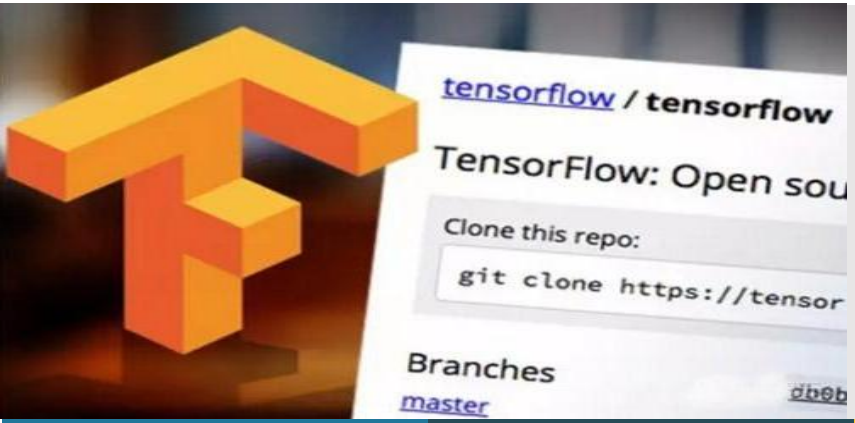
神经网络&深度学习
TENSORFLOW
Google



5.4 矩阵和随机数

中国大学MOOC

矩阵——numpy.matrix

matrix (字符串/列表/元组/数组)

mat (字符串/列表/元组/数组)

参数为字符串

```
>>>a = np.mat(' 1  2  3 ;  4  5  6 ')
>>>a
matrix([[1, 2, 3],
        [4, 5, 6]])
```

参数为Python列表

```
>>>b = np.mat([[1, 2, 3],[4, 5, 6]])
>>>b
matrix([[1, 2, 3],
        [4, 5, 6]])
```

使用NumPy二维数组创建矩阵

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

>>>m = np.mat(a)
>>>m
matrix([[1, 2, 3],
        [4, 5, 6]])

>>>type(a)
<class 'numpy.ndarray'>
>>>type(m)
<class 'numpy.matrix'>
```

参数为NumPy数组

```
>>>m.ndim
2

>>>m.shape
(2, 3)

>>>m.size
6

>>>m.dtype
dtype( 'int32' )
```

矩阵对象的属性

属性	说 明
.ndim	矩阵的维数
.shape	矩阵的形状
.size	矩阵的元素个数
.dtype	元素的数据类型

矩阵运算——矩阵相乘

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 8 & 2 \end{bmatrix}$$

```
>>>a1=np.mat([[0,1],[2,3]])
>>>a2=np.mat([[1,1],[2,0]])
>>>a3=a1*a2
>>>a3
matrix([[ 2,  0],
        [ 8,  2]])
```

矩阵运算——转置、求逆

- 矩阵转置： .T
- 矩阵求逆： .I

```
>>>n=np.mat([[1,2],[-1,-3]])
>>>n
matrix([[ 1,  2],
        [-1, -3]])

>>>n.T
matrix([[ 1,  -1],
        [ 2,  -3]])

>>>n.I
matrix([[ 3,  2],
        [-1, -1]])

>>>n*n.I
matrix([[ 1,  0],
        [ 0,  1]])
```

一个矩阵**乘以它的逆**等于**单位阵**

非方阵的转置、求逆运算

```
>>>a = np.array([[1,2,3],[4,5,6]])
>>>m = np.mat(a)
>>>m
matrix([[1, 2, 3],
        [4, 5, 6]])

>>> m.T
matrix([[ 1,  4],
        [ 2,  5],
        [ 3,  6]])

>>> m.I
matrix([[-0.94444444,  0.44444444],
        [-0.11111111,  0.11111111],
        [ 0.72222222, -0.22222222]])
```

矩阵 VS **二维数组**

- 运算符号简单

  A*B

- 能够表示**高维数组**
- 数组更加灵活，速度更快

随机数模块——numpy.random

函　数	功能描述	返回值
np.random.rand(d0,d1,…,dn)	元素在[0,1)区间均匀分布的数组	浮点数
np.random.uniform(low,hige, size)	元素在[low,hige)区间均匀分布的数组	浮点数
numpy.random.randint(low,hige, size)	元素在[low,hige)区间均匀分布的数组	整　数
np.random.randn(d0,d1,…,dn)	产生标准正态分布的数组	浮点数
np.random.normal(loc, scale, size)	产生正态分布的数组	浮点数

产生随机数

```
#创建2*3的随机数组，取值是在[0,1)之间均匀分布的浮点数
>>> np.random.rand(2,3)
array([[ 0.84543047,  0.76450077,  0.87173984],
       [ 0.01204357,  0.1531674 ,  0.67527698]])

#参数为空，返回一个数字
>>> np.random.rand()
0.289648133266711
```

```
#创建3*2的随机数组，取值是在1至5之间均匀分布的浮点数
>>> np.random.uniform(1,5,(3,2))
array([[ 2.6034975 ,  1.12862773],
       [ 3.24128894,  4.97185042],
       [ 2.77097208,  1.48494326]])
```

```
#创建3*2的随机数组，取值是在1至5之间均匀分布的整数
>>> np.random.randint(1,5,(3,2))
array([[2, 2],
       [4, 1],
       [4, 2]])
```

```
#创建2*3的随机数组，符合标准正态分布
>>> np.random.randn(2,3)
array([[-0.94594743, -1.10163142, -0.40212785],
       [-1.04332498, -1.35958875,  1.5320874 ]])
```

```
#创建3*2的随机数组，符合正态分布，均值为0，方差为1
>>> np.random.normal(0,1,(3,2))
array([[ 1.48764022, -1.52437091],
       [ 0.73473077,  1.51170983],
       [-0.99776822, -0.89273968]])
```

```
#创建2*3的随机数组，符合标准正态分布
>>> np.random.randn(2,3)
array([[-0.94594743, -1.10163142, -0.40212785],
       [-1.04332498, -1.35958875,  1.5320874 ]])
```

```
#创建3*2的随机数组，符合正态分布，均值为0，方差为1
>>> np.random.normal(0,1,(3,2))
array([[ 1.48764022, -1.52437091],
       [ 0.73473077,  1.51170983],
       [-0.99776822, -0.89273968]])
```

因为是随机数，使用同样的语句，所得到的结果也是不同的。

```
>>>np.random.rand(2,3)
array([[0.19151945, 0.62210877, 0.43772774],
       [0.78535858, 0.77997581, 0.27259261]])

>>>np.random.rand(2,3)
array([[0.68346294, 0.71270203, 0.37025075],
       [0.56119619, 0.50308317, 0.01376845]])
```

- **伪随机数：** 由**随机种子**，根据一定的算法生成的。
- **随机种子：** 指定随机数生成时所用算法**开始**的整数值。
  - 如果使用相同的seed( )值，则每次生成的随即数都相同。
  - 如果不设置这个值，则系统根据时间来自己选择这个值，此时每次生成的随机数因时间差异而不同。
  - 设置的seed()值仅一次有效。
  - 随机数产生的算法，和系统有关。

seed()函数——设置随机种子

```
>>>np. random.seed(612)
>>>np.random.rand(2,3)
array([[0.14347163, 0.49589878, 0.95454587],
       [0.13751674, 0.85456667, 0.42853136]])

>>>np. random.seed(612)
>>>np.random.rand(2,3)
array([[0.14347163, 0.49589878, 0.95454587],
       [0.13751674, 0.85456667, 0.42853136]])

>>>np.random.rand(2,3)
array([[0.27646426, 0.80187218, 0.95813935],
       [0.87593263, 0.35781727, 0.50099513]])
```

采用seed()函数设置随机种子，产生了相同的随机数

采用seed()函数设置随机种子，产生了相同的随机数

没有设置随机种子，产生了不一样的结果。

shuffle()——打乱顺序函数

np.random.shuffle(序列)

Python列表、umPy数组等

```
>>> arr = np.arange(10)
>>>print(arr)
[0 1 2 3 4 5 6 7 8 9]

>>> np.random.shuffle(arr)
>>>print( arr)
[1 7 5 2 9 4 3 6 0 8]
```

对于多维数组，使用shuffle()函数只打乱第一维元素

```
>>>b=np.arange(12).reshape(4,3)
>>>b
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

>>>np.random.shuffle(b)
>>>b
array([[ 3,  4,  5],
       [ 9, 10, 11],
       [ 6,  7,  8],
       [ 0,  1,  2]])
```