AI基础：Numpy简易入门

原创：机器学习初学者 机器学习初学者 11月22日

> 本文提供最简易的 Numpy 的入门教程，适合初学者。（黄海广）

1.Numpy 简易入门

NumPy（Numeric Python）提供了许多高级的数值编程工具，如：矩阵数据类型、矢量处理，以及精密的运算库。专为进行严格的数字处理而产生。多为很多大型金融公司使用，以及核心的科学计算组织如：Lawrence Livermore，NASA 用其处理一些本来使用 C++，Fortran 或 Matlab 等所做的任务。

我曾经整理过两篇关于Numpy的文章，好评如潮：

- **Numpy练习题100题-提高你的数据分析技能**

本文总结了Numpy的常用操作，并做成练习题，练习题附答案建议读者把练习题完成。作者认为，做完练习题，Numpy的基本操作没有问题了，以后碰到问题也可以查这些习题。

- **惊为天人，NumPy手写全部主流机器学习模型，代码超3万行**

用 NumPy 手写所有主流 ML 模型，普林斯顿博士后 David Bourgin 最近开源了一个非常剽悍的项目。超过 3 万行代码、30 多个模型。

怎么学：

先完整运行本文的代码，预计用一天时间就够了，再尝试完成Numpy练习题100题，巩固下，接着呢？可以看看上面那篇文章的大神手写的主流机器学习模型代码，看懂就行。

备注：本文代码可以在github下载

https://github.com/fengdu78/Data-Science-Notes/tree/master/2.numpy

1.1 认识 NumPy 数组对象

Numpy 是一个用 python 实现的科学计算的扩展程序库，包括：

- 1、一个强大的 N 维数组对象 Array；
- 2、比较成熟的（广播）函数库；
- 3、用于整合 C/C++和 Fortran 代码的工具包；
- 4、实用的线性代数、傅里叶变换和随机数生成函数。numpy 和稀疏矩阵运算包 scipy 配合使用更加方便。

```
import numpy as np                    # 导入NumPy工具包
```

```
data = np.arange(12).reshape(3, 4)  # 创建一个3行4列的数组
```

```
data
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
type(data)
```

```
numpy.ndarray
```

```
data.ndim          # 数组维度的个数，输出结果2，表示二维数组
```

```
2
```

```
data.shape         # 数组的维度，输出结果（3，4），表示3行4列
```

```
(3, 4)
```

```
data.size          # 数组元素的个数，输出结果12，表示总共有12个元素
```

```
12
```

```
data.dtype # 数组元素的类型，输出结果dtype('int64'),表示元素类型都是int64
```

```
dtype('int32')
```

1.2 创建 NumPy 数组

```

```

```
import numpy as np
```

```
data1 = np.array([1, 2, 3])                    # 创建一个一维数组
```

```
data1
```

```
array([1, 2, 3])
```

```
data2 = np.array([[1, 2, 3], [4, 5, 6]])   # 创建一个二维数组
```

```
data2
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
np.zeros((3, 4))#创建一个全0数组
```

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

```
np.ones((3, 4))#创建全一数组
```

```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

```
np.empty((5, 2))# 创建全空数组，其实每个值都是接近于零的数
```

```
array([[ 6.95312756e-310,  2.12199579e-314],
       [ 2.12199579e-314,  4.94065646e-324],
       [ 0.00000000e+000, -7.06252554e-311],
       [ 0.00000000e+000, -8.12021073e-313],
       [ 1.29923372e-311,  2.07507571e-322]])
```

```
np.arange(1, 20, 5)
```

```
array([ 1,  6, 11, 16])
```

```
np.array([1, 2, 3, 4], float)
```

```
array([1., 2., 3., 4.])
```

```
np.ones((2, 3), dtype='float64')
```

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

1.3 ndarry 对象的数据类型

1.3.1 查看数据类型

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

```
data_one.dtype.name
```

```
'int32'
```

1.3.2 转换数据类型

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

```
data.dtype
```

```
dtype('int32')
```

```
float_data = data.astype(np.float64) # 数据类型转换为float64
```

```
float_data.dtype
```

```
dtype('float64')
```

```
float_data = np.array([1.2, 2.3, 3.5])
```

```
float_data
```

```
array([1.2, 2.3, 3.5])
```

```
int_data = float_data.astype(np.int64) # 数据类型转换为int64
```

```
int_data
```

```
array([1, 2, 3], dtype=int64)
```

```
str_data = np.array(['1', '2', '3'])
```

```
int_data = str_data.astype(np.int64)
```

```
int_data
```

```
array([1, 2, 3], dtype=int64)
```

1.4 数组运算

1.4.1 向量化运算

```
import numpy as np
```

```
data1 = np.array([[1, 2, 3], [4, 5, 6]])
```

```
data2 = np.array([[1, 2, 3], [4, 5, 6]])
```

```
data1 + data2        # 数组相加
```

```
array([[ 2,  4,  6],
       [ 8, 10, 12]])
```

```
data1 * data2        # 数组相乘
```

```
array([[ 1,  4,  9],
       [16, 25, 36]])
```

```
data1 - data2        # 数组相减
```

```
array([[0, 0, 0],
       [0, 0, 0]])
```

```
data1 / data2       # 数组相除
```

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

1.4.2 数组广播

numpy 数组间的基础运算是一对一，也就是`a.shape==b.shape`，但是当两者不一样的时候，就会自动触发广播机制，如下例子：

```
import numpy as np
```

```
arr1 = np.array([[0], [1], [2], [3]])
```

```
arr1.shape
```

```
(4, 1)
```

```
arr2 = np.array([1, 2, 3])
```

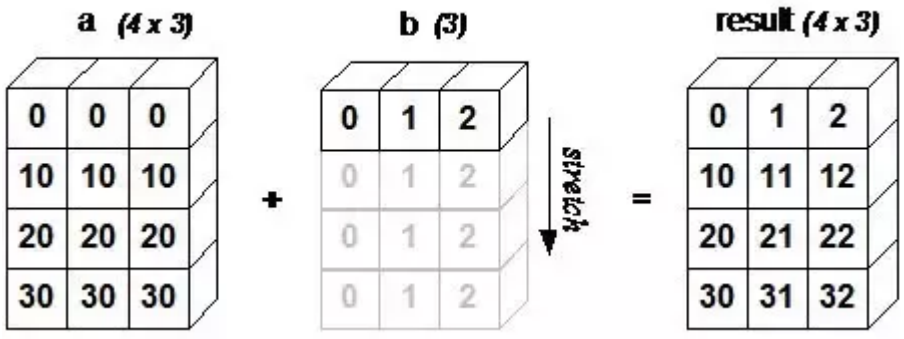
```
arr2.shape
```

```
(3,)
```

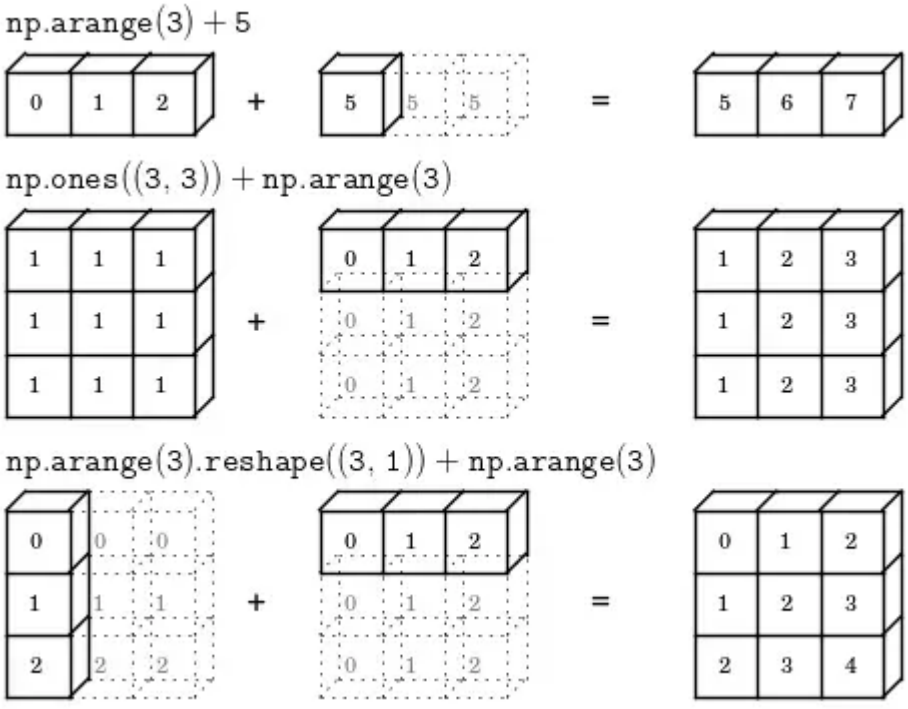
```
arr1 + arr2
```

```
array([[1, 2, 3],
       [2, 3, 4],
       [3, 4, 5],
       [4, 5, 6]])
```

到这里，我们来给出一张图：



也可以看这张图：



1.4.3 数组与标量间的运算

```
import numpy as np
```

```
data1 = np.array([[1, 2, 3], [4, 5, 6]])
```

```
data2 = 10
```

```
data1 + data2      # 数组相加
```

```
array([[11, 12, 13],
       [14, 15, 16]])
```

```
data1 * data2       # 数组相乘
```

```
array([[10, 20, 30],
       [40, 50, 60]])
```

```
data1 - data2        # 数组相减
```

```
array([[-9, -8, -7],
```

```
       [-6, -5, -4]])
```

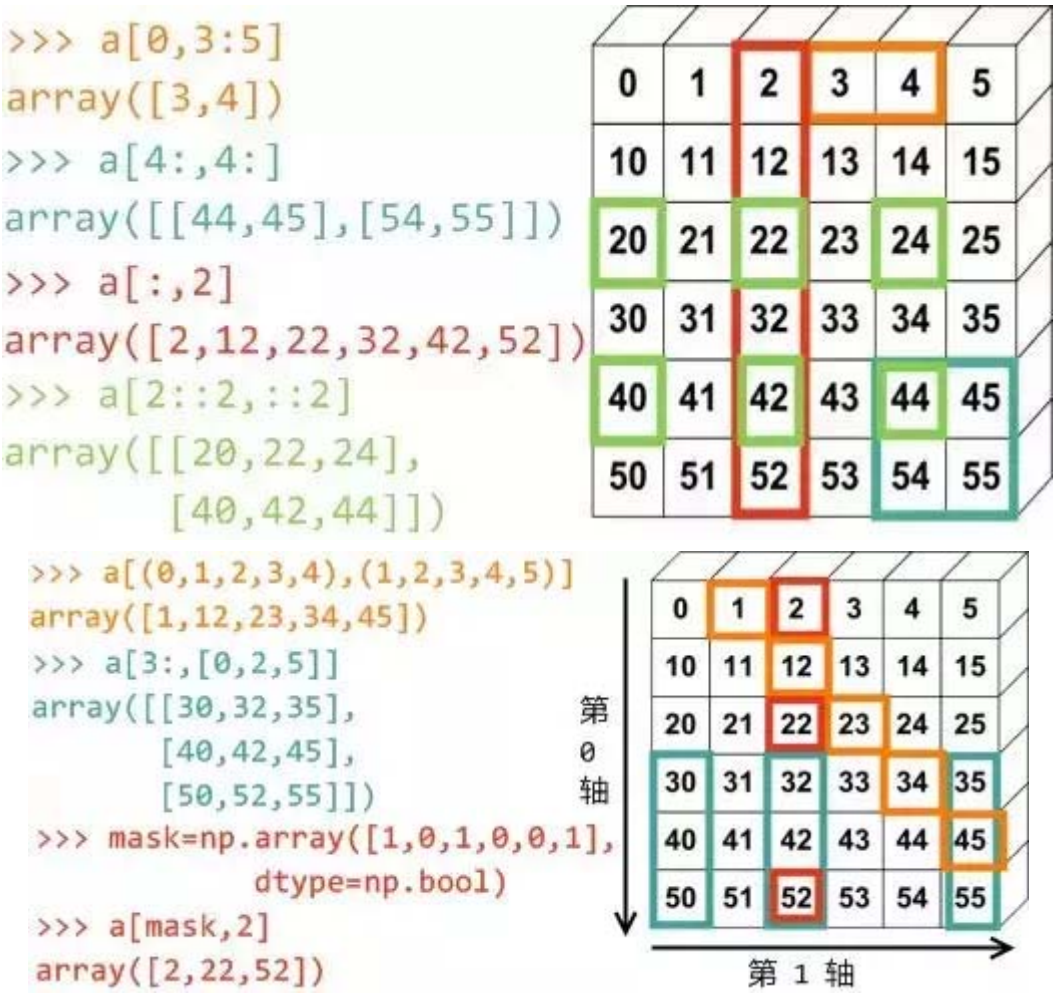
```
data1 / data2        # 数组相除
```

```
array([[0.1, 0.2, 0.3],
       [0.4, 0.5, 0.6]])
```

1.5 ndarray 的索引和切片

1.5.1 整数索引和切片的基本使用

我们一起来来总结一下，看下面切片取值方式（对应颜色是取出来的结果）：




```
import numpy as np
```

```
arr = np.arange(8)    # 创建一个一维数组
```

```
arr
```

```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

```
arr[5]                  # 获取索引为5的元素
```

```
5
```

```
arr[3:5]                # 获取索引为3~5的元素，但不包括5
```

```
array([3, 4])
```

```
arr[1:6:2]              # 获取索引为1~6的元素，步长为2
```

```
array([1, 3, 5])
```

```
import numpy as np
```

```
arr2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]]) # 创建二维数组
```

```
arr2d
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
arr2d[1]            # 获取索引为1的元素
```

```
array([4, 5, 6])
```

```
arr2d[0, 1]         # 获取位于第0行第1列的元素
```

```
2
```

```
arr2d[:2]
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
arr2d[0:2, 0:2]
```

```
array([[1, 2],
       [4, 5]])
```

```
```

```
arr2d[1, :2]
```

```
array([4, 5])
```

1.5.2 花式（数组）索引的基本使用

```
import numpy as np
```

```
demo_arr = np.empty((4, 4))                # 创建一个空数组
for i in range(4):
    demo_arr[i] = np.arange(i, i + 4)    # 动态地为数组添加元素
```

```
demo_arr
```

```
array([[0., 1., 2., 3.],
       [1., 2., 3., 4.],
       [2., 3., 4., 5.],
       [3., 4., 5., 6.]])
```

```
demo_arr[[0, 2]]         # 获取索引为[0,2]的元素
```

```
array([[0., 1., 2., 3.],
       [2., 3., 4., 5.]])
```

```
demo_arr[[1, 3], [1, 2]]     # 获取索引为(1,1)和(3,2)的元素
```

```
array([2., 5.])
```

1.5.3 布尔型

```
# 存储学生姓名的数组
student_name = np.array(['Tom', 'Lily', 'Jack', 'Rose'])
```

```
student_name
```

```
array(['Tom', 'Lily', 'Jack', 'Rose'], dtype='<U4')
```

```
# 存储学生成绩的数组
student_score = np.array([[79, 88, 80], [89, 90, 92], [83, 78, 85], [78, 76, 80]])
```

```
student_score
```

```
array([[79, 88, 80],
       [89, 90, 92],
       [83, 78, 85],
       [78, 76, 80]])
```

```
# 对student_name和字符串“Jack”通过运算符产生一个布尔型数组
student_name == 'Jack'
```

```
array([False, False,  True, False])
```

```
# 将布尔数组作为索引应用于存储成绩的数组student_score，
# 返回的数据是True值对应的行
student_score[student_name=='Jack']
```

```
array([[83, 78, 85]])
```

```
student_score[student_name=='Jack', :1]
```

```
array([[83]])
```

1.6 数组的转置和轴对称

```
arr = np.arange(12).reshape(3, 4)
```

```
arr
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
arr.T      # 使用T属性对数组进行转置
```

```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

```
arr = np.arange(16).reshape((2, 2, 4))
```

```
arr
```

```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7]],

       [[ 8,  9, 10, 11],
        [12, 13, 14, 15]]])
```

```
arr.transpose(1, 2, 0)   # 使用transpose()方法对数组进行转置
```

```
array([[[ 0,  8],
        [ 1,  9],
        [ 2, 10],
        [ 3, 11]],

       [[ 4, 12],
        [ 5, 13],
        [ 6, 14],
        [ 7, 15]]])
```

```
arr
```

```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7]],

       [[ 8,  9, 10, 11],
        [12, 13, 14, 15]]])
```

```
arr.swapaxes(1, 0)    # 使用swapaxes方法对数组进行转置
```

```
array([[[ 0,  1,  2,  3],
        [ 8,  9, 10, 11]],

       [[ 4,  5,  6,  7],
        [12, 13, 14, 15]]])
```

1.7 NumPy 通用函数

```
arr = np.array([4, 9, 16])
```

```
np.sqrt(arr)#开方
```

```
array([2., 3., 4.])
```

```
np.abs(arr)#求绝对值
```

```
array([ 4,  9, 16])
```

```
np.square(arr)#求平方
```

```
array([ 16,  81, 256], dtype=int32)
```

```
x = np.array([12, 9, 13, 15])
```

```
y = np.array([11, 10, 4, 8])
```

```
np.add(x, y)      # 计算两个数组的和
```

```
array([23, 19, 17, 23])
```

```
np.multiply(x, y) # 计算两个数组的乘积
```

```
array([132,  90,  52, 120])
```

```
np.maximum(x, y)  # 两个数组元素级最大值的比较
```

```
array([12, 10, 13, 15])
```

```
np.greater(x, y)  # 执行元素级的比较操作
```

```
array([ True, False,  True,  True])
```

1.8 利用 NumPy 数组进行数据处理

1.8.1 将条件逻辑转为数组运算

```
arr_x = np.array([1, 5, 7])
```

```
arr_y = np.array([2, 6, 8])
```

```
arr_con = np.array([True, False, True])
```

```
result = np.where(arr_con, arr_x, arr_y)
```

```

```

```
result
```

```
array([1, 6, 7])
```

1.8.2 数组统计运算

```
arr = np.arange(10)
```

```
arr.sum()      # 求和
```

```
45
```

```
arr.mean()     # 求平均值
```

```
4.5
```

```
arr.min()      # 求最小值
```

```
0
```

```
arr.max()       # 求最大值
```

```
9
```

```
arr.argmin()   # 求最小值的索引
```

```
0
```

```
arr.argmax()   # 求最大值的索引
```

```
9
```

```
arr.cumsum()   # 计算元素的累计和
```

```
array([ 0,  1,  3,  6, 10, 15, 21, 28, 36, 45], dtype=int32)
```

```
arr.cumprod()  # 计算元素的累计积
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

```
x = np.arange(1 , 16).reshape((3 , 5))
```

```
print(x)
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]]
```

```
np.diff(x,axis=1) #默认axis=1
```

```
array([[1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1]])
```

```
np.diff(x,axis=0)
```

```
array([[5, 5, 5, 5, 5],
       [5, 5, 5, 5, 5]])
```

```
np.floor([-0.6,-1.4,-0.1,-1.8,0,1.4,1.7])
```

```
array([-1., -2., -1., -2.,  0.,  1.,  1.])
```

看到没，负数取整，跟上述的 around 一样，是向左！

```
np.ceil([1.2,1.5,1.8,2.1,2.0,-0.5,-0.6,-0.3])
```

```
array([ 2.,  2.,  2.,  3.,  2., -0., -0., -0.])
```

取上限！找这个小数的最大整数即可！

查找，利用 np.where 实现小于 0 的值用 0 填充吗，大于 0 的数不变！

```
x = np.array([[1, 0],
       [2, -2],
      [-2, 1]])
print(x)
```

```
[[ 1  0]
 [ 2 -2]
 [-2  1]]
```

```
np.where(x>0,x,0)
```

```
array([[1, 0],
       [2, 0],
       [0, 1]])
```

1.8.3 数组排序

```
arr = np.array([[6, 2, 7], [3, 6, 2], [4, 3, 2]])
```

```
arr
```

```
array([[6, 2, 7],
       [3, 6, 2],
       [4, 3, 2]])
```

```
arr.sort()
```

```
arr
```

```
array([[2, 6, 7],
       [2, 3, 6],
       [2, 3, 4]])
```

```
arr = np.array([[6, 2, 7], [3, 6, 2], [4, 3, 2]])
```

```
arr
```

```
array([[6, 2, 7],
       [3, 6, 2],
       [4, 3, 2]])
```

```
arr.sort(0)       # 沿着编号为0的轴对元素排序
```

```
arr
```

```
array([[3, 2, 2],
       [4, 3, 2],
       [6, 6, 7]])
```

1.8.4 检索数组元素

```
arr = np.array([[1, -2, -7], [-3, 6, 2], [-4, 3, 2]])
```

```
arr
```

```
array([[ 1, -2, -7],
       [-3,  6,  2],
       [-4,  3,  2]])
```

```
np.any(arr > 0)      # arr的所有元素是否有一个大于0
```

```
True
```

```
np.all(arr > 0)      # arr的所有元素是否都大于0
```

```
False
```

1.8.5 唯一化及其他集合逻辑

```
arr = np.array([12, 11, 34, 23, 12, 8, 11])
```

```
np.unique(arr)
```

```
array([ 8, 11, 12, 23, 34])
```

```
np.in1d(arr, [11, 12])
```

```
array([ True,  True, False, False,  True, False,  True])
```

1.9 线性代数模块

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

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

```
arr_x.dot(arr_y)   # 等价于np.dot(arr_x, arr_y)
```

```
array([[22, 28],
       [49, 64]])
```

1.10 随机数模块

```
import numpy as np
```

```
np.random.rand(3, 3)     # 随机生成一个二维数组
```

```
array([[0.90422833, 0.57874299, 0.36084718],
       [0.46674697, 0.59189161, 0.88876503],
       [0.51836003, 0.30765097, 0.79668824]])
```

```
np.random.rand(2, 3, 3) # 随机生成一个三维数组
```

```
array([[[0.21438832, 0.58877977, 0.86120009],
        [0.15222229, 0.53060997, 0.0562486 ],
        [0.88035435, 0.32505223, 0.9045713 ]],

       [[0.32907094, 0.88987195, 0.34523123],
        [0.90645746, 0.61257549, 0.83944649],
        [0.2015535 , 0.84522463, 0.87759584]]])
```

```
import numpy as np
```

```
np.random.seed(0)   # 生成随机数的种子
```

```
np.random.rand(5)   # 随机生成包含5个元素的浮点数组
```

```
array([0.5488135 , 0.71518937, 0.60276338, 0.54488318, 0.4236548 ])
```

```
np.random.seed(0)
```

```
np.random.rand(5)
```

```
array([0.5488135 , 0.71518937, 0.60276338, 0.54488318, 0.4236548 ])
```

```
np.random.seed()
```

```
np.random.rand(5)
```

```
array([0.19299506, 0.41434116, 0.90011257, 0.37469705, 0.69775797])
```

备注：本文代码可以在github下载

https://github.com/fengdu78/Data-Science-Notes/tree/master/2.numpy

往期精彩回顾

- 那些年做的学术公益-你不是一个人在战斗
- 适合初学者入门人工智能的路线及资料下载
- 机器学习在线手册

备注：加入本站微信群或者qq群，请回复“加群”

加入知识星球（4500+用户，ID：92416895），请回复“知识星球”