《人工智能与Python程序设计》—— Python数值计算

人工智能与Python程序设计 教研组

Python AI
Numpy与科学计算

提纲

- numpy库简介
- numpy数组的创建
- numpy数组的索引和切片
- numpy数组的运算
- nunpy科学计算实践

科学计算：矩阵

- 科学计算是解决科学和工程中的数学问题利用计算机进行的数值计算
- 组织数据是运算的基础，也是将客观世界数字化的必要手段
- 科学计算以矩阵而不是单一数值为基础，增加了计算密度，能够表达更为复杂的数据运算逻辑

- 数学的矩阵（Matrix）是一个按照长方阵列排列的复数或实数集合
- 矩阵有维度概念，一维矩阵是线性的，类似于列表，二维矩阵是表格状的二维数组

numpy库概述

- Python 标准库中提供了一个array 类型，用于保存数组类型数据，但这个类型不支持多维数据，处理函数也不够丰富，不适合用于做数值运算。
- numpy 是用于处理含有同种元素的多维数组运算的第三方库。
- numpy 已经成为了Python科学计算事实上的标准库。

numpy库概述

- numpy相比于list的优势：
  - 内存效率高
  - 使用更方便
  - 内置了很多强大的功能：FFT、卷积、快速搜索、基本统计、线性代数、直方图等
  - 速度更快

```
%timeit sum(range(10000000))
```

```
342 ms ± 14.9 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
from numpy import arange
%timeit arange(10000000).sum()
```

```
16.2 ms ± 149 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

numpy库概述

- numpy 库处理的最基础数据类型是由同种元素构成的多维数组（ndarray），简称“数组”。
- 数组中所有元素的类型必须相同。
- 数组中元素可以用整数索引，序号从0开始。ndarray 类型的维度(dimensions)叫做轴(axes)。
- 由于numpy库中函数较多且命名容易与常用命名混淆，建议采用如下方式引用numpy 库：

import numpy as np

  - 其中，as 保留字与import 一起使用能够改变后续代码中库的命名空间，有助于提高代码可读性。
  - 在程序的后续部分中，np 代替numpy。

numpy库数组创建

函数	描述
np.array([x,y,z], dtype=int)	从 Python 列表和元组创造数组
np.arange(x,y,i)	创建一个由 x 到 y，以 i 为步长的数组
np.linspace(x,y,n)	创建一个由 x 到 y，等分成 n 个元素的数组
np.indices((m,n))	创建一个 m 行 n 列的矩阵
np.random.rand(m,n)	创建一个 m 行 n 列的随机数组
np.ones((m,n),dtype)	创建一个 m 行 n 列全 1 的数组，dtype 是数据类型
np.empty((m,n),dtype)	创建一个 m 行 n 列全 0 的数组，dtype 是数据类型

numpy库数组创建

- 数组创建

In [1]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\subing\anaconda3\lib\site-packages (1.18.5)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import numpy as np
x1 = np.zeros((2,3))
print(x1)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

numpy库数组属性

属性	描述
ndarray.ndim	数组轴的个数，也被称作秩
ndarray.shape	数组在每个维度上大小的整数元组
ndarray.size	数组元素的总个数
ndarray.dtype	数组元素的数据类型，dtype 类型可以用于创建数组中
ndarray.itemsize	数组中每个元素的字节大小
ndarray.data	包含实际数组元素的缓冲区地址
ndarray.flat	数组元素的迭代器

numpy库数组属性

- 数组属性

```
In [3]: x2 = np.ones((5,6))
        print(x2)
```

```
[[1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]]
```

```
In [4]: x2.ndim
```

```
Out[4]: 2
```

```
In [5]: x2.shape
```

```
Out[5]: (5, 6)
```

```
In [6]: x2.dtype
```

```
Out[6]: dtype('float64')
```

numpy库数组形态操作方法

- 数组是ndarray类型对象，可以采用<a>.<b>()方式调用一些方法。
- 改变数组基础形态的操作方法：例如改变和调换数组维度等。
- np.flatten()函数用于数组降维，相当于平铺数组中数据，该功能在矩阵运算及图像处理中用处很大。

方法	描述
ndarray.reshape(n,m)	不改变数组 ndarray，返回一个维度为(n,m)的数组
ndarray.resize(new_shape)	与 reshape()作用相同，直接修改数组 ndarray
ndarray.swapaxes(ax1, ax2)	将数组 n 个维度中任意两个维度进行调换
ndarray. flatten()	对数组进行降维，返回一个折叠后的一维数组
ndarray.ravel()	作用同 np.flatten()，但是返回数组的一个视图

numpy库数组形态操作方法

```
In [7]: x2.flatten()

Out[7]: 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., 1., 1., 1., 1.])
```

```
In [11]: x3 = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], dtype=int)
         x4 = x3.reshape((3,4))
         print(x4)

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

```
In [12]: x3.resize((2,6))
         print(x3)

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

ndarray 类的索引和切片方法

- 数组切片得到的是原始数组的视图，所有修改都会直接反映到源数组。如果需要得到的ndarray 切片的一 份 副 本 ， 需 要 进 行 复 制 操 作 ， 比 如arange[5:8].copy()

方法	描述
x[i]	索引第 i 个元素
x[-i]	从后向前索引第 i 个元素
x[n:m]	默认步长为 1，从前往后索引，不包含 m
x[-m:-n]	默认步长为 1，从后往前索引，结束位置为 n
x[n,m,i]	指定 i 步长的由 n 到 m 的索引

ndarray 类的索引和切片方法

```
In  [13]: x = np.random.rand(6,3)
          x[2]

Out[13]: array([0.71340748, 0.37877357, 0.02473368])

In  [14]: x[2:4]

Out[14]: array([[0.71340748, 0.37877357, 0.02473368],
                [0.84463945, 0.50083296, 0.76809101]])

In  [15]: x[-5:-2:2]

Out[15]: array([[0.14797886, 0.71228022, 0.0721306 ],
                [0.84463945, 0.50083296, 0.76809101]])
```

ndarray 类的算术运算函数

- 加减乘除等算术运算函数

函数	描述
np.add(x1, x2 [, y])	y = x1 + x2
np.subtract(x1, x2 [, y])	y = x1 - x2
np.multiply(x1, x2 [, y])	y = x1 * x2
np.divide(x1, x2 [, y])	y = x1 / x2
np floor_divide(x1, x2 [, y])	y = x1 // x2，返回值取整
np.negative(x [,y])	y = -x
np.power(x1, x2 [, y])	y = x1**x2
np.remainder(x1, x2 [, y])	y = x1 % x2

ndarray 类的算术运算函数

- 这些函数中，输出参数y 可选，如果没有指定，将创建并返回一个新的数组保存计算结果；如果指定参数，则将结果保存到参数中。例如，两个数组相加可以简单地写为a+b，而np.add(a,b,a)则表示a+=b。

In [4]:
```
import numpy as np
x = np.random.rand(3,2)
y = np.ones((3,2))
np.add(x,y,x)
print(x)
```

```
[[1.68169165 1.54105006]
 [1.66019779 1.07333519]
 [1.2191459  1.74021582]]
```

In [5]:
```
z = x + y
print(z)
```

```
[[2.68169165 2.54105006]
 [2.66019779 2.07333519]
 [2.2191459  2.74021582]]
```

ndarray 类的比较运算函数

- 比较运算函数：返回一个布尔数组，包含两个数组中对应元素值的比较结果

函数	符号描述
np. equal(x1, x2 [, y])	y = x1 == x2
np. not_equal(x1, x2 [, y])	y = x1 != x2
np. less(x1, x2, [, y])	y = x1 < x2
np. less_equal(x1, x2, [, y])	y = x1 <= x2
np. greater(x1, x2, [, y])	y = x1 > x2
np. greater_equal(x1, x2, [, y])	y = x1 >= x2
np.where(condition[x,y])	根据给出的条件判断输出 x 还是 y

ndarray 类的比较运算函数

- where()函数是三元表达式x if condition else y 的矢量版本

```
In  [6]: np.less(z,2.5)

Out[6]: array([[False, False],
               [False,  True],
               [ True, False]])

In  [7]: np.less(z,[[2.5,2.5],[2.5,2.5],[2.5,2.5]])

Out[7]: array([[False, False],
               [False,  True],
               [ True, False]])

In  [9]: np.where(z>2.5,0,z)

Out[9]: array([[0.        , 0.        ],
               [0.        , 2.07333519],
               [2.2191459 , 0.        ]])
```

numpy库矩阵运算

- 矩阵乘法、转置、逆

```
import numpy as np
a1 = np.array([[1,2,3],[4,5,6]]) # a1为2*3矩阵
a2 = np.array([[1,2],[3,4],[5,6]]) # a2为3*2矩阵

print(a1.dot(a2))
b = np.dot(a1,a2)
print(b)

c = np.matmul(a1,a2)
print(c)
print(c.transpose())
```

```
[[22 28]
 [49 64]]
[[22 28]
 [49 64]]
[[22 28]
 [49 64]]
[[22 49]
 [28 64]]
```

```
import numpy.linalg as lg
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(lg.inv(a))

a = np.eye(3) # 3阶单位矩阵
print(lg.inv(a)) # 单位矩阵的逆为他本身

c = a.tolist()
print(c)
print(type(c))
```

```
[[ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]
 [-6.30503948e+15  1.26100790e+16 -6.30503948e+15]
 [ 3.15251974e+15 -6.30503948e+15  3.15251974e+15]]
[[1.  0.  0.]
 [0.  1.  0.]
 [0.  0.  1.]]
[[1.0, 0.0, 0.0], [0.0, 1.0, 0.0], [0.0, 0.0, 1.0]]
<class 'list'>
```

numpy库矩阵运算

- 乘法*

```
a1 = np.array([[1,2,3],[4,5,6]])
a2 = np.array([[3,2,1],[-1,-2,1]])
a3 = 3
print(a1*a2)
print(a3*a1)
```

```
[[  3   4   3]
 [ -4 -10   6]]
[[ 3  6  9]
 [12 15 18]]
```

numpy库其它运算函数

- numpy 库还包括三角运算函数、傅里叶变换、随机和概率分布、基本数值统计、位运算等非常丰富的功能

函数	描述
np.abs(x)	计算基于元素的整形，浮点或复数的绝对值。
np.sqrt(x)	计算每个元素的平方根
np.squre(x)	计算每个元素的平方
np.sign(x)	计算每个元素的符号：1(+)，0， -1(-)
np.ceil(x)	计算大于或等于每个元素的最小值
np.floor(x)	计算小于或等于每个元素的最大值
np.rint (x[, out])	圆整,取每个元素为最近的整数,保留数据类型
np.exp(x[, out])	计算每个元素指数值
np.log(x), np.log10(x), np.log2(x)	计算自然对数(e),基于 10,2 的对数,log(1 + x)

numpy库数据存储

- numpy能够读写磁盘上的文本数据或二进制数据
- 以二进制格式将数组保存到磁盘：np.save，默认情况下，数组是以未压缩的原始二进制格式保存在扩展名为.npy的文件中
- np.load从磁盘读取.npy文件

```
In [13]: np.save('C:/RUC/课程/PythonAI/课程课件/testz.npy',z)
         y = np.load('C:/RUC/课程/PythonAI/课程课件/testz.npy')
         print(y)
```

```
[[2.68169165 2.54105006]
 [2.66019779 2.07333519]
 [2.2191459  2.74021582]]
```

numpy库数据存储

- numpy.savez函数可以将多个数组保存到同一个文件中：第一个参数是文件名，其后的参数都是需要保存的数组
- 可以使用关键字参数为数组起一个名字，非关键字参数传递的数组会自动起名为arr_0, arr_1, ...
- 输出的是一个压缩文件(扩展名为npz)，其中每个文件都是一个save函数保存的npy文件，文件名对应于数组名
- load函数自动识别npz文件，并且返回一个类似于字典的对象，可以通过数组名作为关键字获取数组的内容