Numpy几个判别函数

总结numpy中判别函数：nonzero()、logical\_and、logical\_or()和logical\_not()的用法

**np.nonzero()**

    no.nonzero()函数是numpy中用于得到数组array中非零元素的位置（数组索引）的函数，很适合用来对数据下标的提取。nonzero函数中不仅可以放数值矩阵/行列，同样可以放布尔型（True、False）矩阵/行列，由于这个特性使其适用范围更加的广泛。

注：

* nonzero()只会返回数组中非零元素的索引值，那些零值元素的索引值不会返回；
* 返回的索引数值是一个二维的tuple数组，tuple中包含一维的array数组。每个array数组都是对一维数据判断的描述；
* 索引值数组的每一个array均是从一个维度上来描述其索引值。比如，如果原始数组是一个二维数组，则索引值数组有两个array，第一个array从行维度来描述索引值；第二个array从列维度来描述索引值；
* 函数np.transpose(np.nonzero(arr)) 能够描述出每一个非零元素在不同维度的索引值；
* 可以通过arr[nonzero(arr)]得到所有arr中的非零值；
* nonzero函数中可以适用布尔型矩阵，等同于对0、1矩阵的判断（这个布尔值可以是我们对矩阵的判断条件）。

案例：

|  |
| --- |
| import numpy as np  arr = np.array([[1, 0, 1], [2, 2, 0, ], [0, 3, 2]])  print(arr)  # 原始矩阵  # [[1 0 1]  # [2 2 0]  # [0 3 2]]  # 返回每个非0元素坐标的tuple数组（包含行、列描述信息）  result = np.nonzero(arr)  print(result)  # (array([0, 0, 1, 1, 2, 2], dtype=int64), array([0, 2, 0, 1, 1, 2], dtype=int64))  print(result[0], result[1]) # 行描述、列描述  # [0 0 1 1 2 2] [0 2 0 1 1 2]  # 返回每个非0元素坐标  result = np.transpose(np.nonzero(arr))  print(result)  # [[0 0]  # [0 2]  # [1 0]  # [1 1]  # [2 1]  # [2 2]]  # 返回所有非0值  result = arr[np.nonzero(arr)]  print(result)  # [1 1 2 2 3 2]  # 对矩阵第二列判断  result = np.nonzero(arr[:, 1])  print(result)  # (array([1, 2], dtype=int64),) # 由于数据为1列，所以tuple返回一个array数组  # 布尔值判断  # 判断矩阵第三列中大于1值值的行号  result = np.nonzero(arr[:, 2] > 1)  print(arr[:, 2] > 1)  # [False False True]  print(result)  # (array([2], dtype=int64),) |

**logical\_and、logical\_or和logical\_not**

logical\_and、logical\_or和logical\_not是逻辑与、或、非运算。

案例：

|  |
| --- |
| import numpy as np  arr = np.array([[1, 0, 1], [2, 2, 0], [0, 3, 2]])  # 原始矩阵  print(arr)  # 获得第一列和第二列的布尔值，并将两者做与操作  result = np.logical\_and(arr[:, 0] > 0, arr[:, 1] > 0)  print(arr[:, 0] > 0, arr[:, 1] > 0)  # [ True True False] [False True True]  print(result)  # [False True False]  # 或操作一个为真全为真  result = np.logical\_or(arr[:, 0] > 0, arr[:, 1] > 0)  print(arr[:, 0] > 0, arr[:, 1] > 0)  # [ True True False] [False True True]  print(result)  # [ True True True]  result = np.logical\_not(arr[:, 0] > 0)  result2 = np.logical\_not(arr[:, 1] > 0)  print(arr[:, 0] > 0, arr[:, 1] > 0)  # [ True True False] [False True True]  print(result, result2)  # [False False True] [ True False False] |

最后附上nonzero函数与logical\_and函数结合的一个例子：

|  |
| --- |
| import numpy as np  arr = np.array([[1, 0, 1], [2, 2, 0, ], [0, 3, 2]])  # 原始矩阵  print(arr)  # [[1 0 1]  # [2 2 0]  # [0 3 2]]  # 获取矩阵第一列与第二列同时不为0的行号  result = np.nonzero(np.logical\_and(arr[:, 0] > 0, arr[:, 1] > 0))  print(arr[:, 0] > 0, arr[:, 1] > 0)  # [ True True False] [False True True]  print(result[0])  # [1] |

参考：https://blog.csdn.net/qq\_36523839/java/article/details/82318061