NumPy 百题大冲关

NumPy 是 Python 语言的一个扩充程序库。支持大量高维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。机器学习涉及到大量对数组的变换和运算，NumPy 就成了必不可少的工具之一。

NumPy 百题大冲关分为基础篇和进阶篇，每部分各有 50 道练习题。基础部分的练习题在于熟悉 NumPy 常用方法的使用，而进阶部分则侧重于 NumPy 方法的组合应用。

如果你在学习课程之前已经有 NumPy 使用基础，那么可以对照着单元格复习一遍。如果你对 NumPy 并不熟练，就一定要亲自动手在每个示例单元格下方的空白单元格中练习。

基础部分

1. 导入 NumPy

练习 NumPy 之前，首先需要导入 NumPy 模块，并约定简称为 `np`。

教学代码：

```
import numpy as np
```

动手练习：

```
# 在空白单元格中重复输入上面的教学代码练习，亲自动手，不要复制粘贴
```

2. 查看 NumPy 版本信息

```
print(np.__version__)
```

创建数组

NumPy 的主要对象是多维数组 Ndarray。在 NumPy 中维度（dimensions）叫做轴（axes），轴的个数叫做秩（rank）。

例如，下方数组是一个秩为 1 的数组，因为它只有一个轴，而轴的长度为 3。

```
[1, 2, 3]
```

又例如，下方数组的秩为 2。第一个维度长度为 2,第二个维度长度为 3。

```
[[1., 2., 3.],
 [4., 5., 6.]]
```

3. 通过列表创建一维数组

注意：`numpy.array` 和 Python 标准库 `array.array` 并不相同，前者更为强大，这也就是我们学习 NumPy 的重要原因之一。

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

4. 通过列表创建二维数组

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

5. 创建全为 0 的二维数组

```
np.zeros((3, 3))
```

6. 创建全为 1 的三维数组

```
np.ones((2, 3, 4))
```

注意：务必想清楚上面 4 个数组的维度关系

7. 创建一维等差数组

```
np.arange(5)
```

8. 创建二维等差数组

```
np.arange(6).reshape(2, 3)
```

9. 创建单位矩阵（二维数组）

```
np.eye(3)
```

10. 创建等间隔一维数组

```
np.linspace(1, 10, num=6)
```

11. 创建二维随机数组

```
np.random.rand(2, 3)
```

12. 创建二维随机整数数组（数值小于 5）

```
np.random.randint(5, size=(2, 3))
```

13. 依据自定义函数创建数组

```
np.fromfunction(lambda i, j: i + j, (3, 3))
```

数组运算

生成一维示例数组

```
a = np.array([10, 20, 30, 40, 50])
b = np.arange(1, 6)
a, b
```

14. 一维数组加法运算

```
a + b
```

15. 一维数组减法运算

```
a - b
```

16. 一维数组乘法运算

```
a * b
```

17. 一维数组除法运算

```
a / b
```

生成二维示例数组（可以看作矩阵）

```
A = np.array([[1, 2],
              [3, 4]])
B = np.array([[5, 6],
              [7, 8]])
A, B
```

18. 矩阵加法运算

```
A + B
```

19. 矩阵减法运算

```
A - B
```

20. 矩阵元素间乘法运算

```
A * B
```

21. 矩阵乘法运算（注意与上题的区别）

```
np.dot(A, B)
```

** 如果不了解矩阵乘法运算，点击此链接 学习。*

```
# 如果使用 np.mat 将二维数组准确定义为矩阵，就可以直接使用 * 完成矩阵乘法计算
np.mat(A) * np.mat(B)
```

22. 数乘矩阵

```
2 * A
```

23. 矩阵的转置

```
A.T
```

24. 矩阵求逆

```
np.linalg.inv(A)
```

数学函数

25. 三角函数

```
print(a)

np.sin(a)
```

26. 以自然对数函数为底数的指数函数

```
np.exp(a)
```

27. 数组的方根的运算（开平方）

```
np.sqrt(a)
```

28. 数组的方根的运算（立方）

```
np.power(a, 3)
```

数组切片和索引

29. 一维数组索引

```
a = np.array([1, 2, 3, 4, 5])
a[0], a[-1]
```

30. 一维数组切片

```
a[0:2], a[:-1]
```

31. 二维数组索引

```
a = np.array([(1, 2, 3), (4, 5, 6), (7, 8, 9)])
a[0], a[-1]
```

32. 二维数组切片（取第 2 列）

```
print(a)

a[:, 1]
```

33. 二维数组切片（取第 2，3 行）

```
a[1:3, :]
```

数组形状操作

生成二维示例数组

```
a = np.random.random((3, 2))
a
```

34. 查看数组形状

```
a.shape
```

35. 更改数组形状（不改变原始数组）

```
# reshape 并不改变原始数组
a.reshape(2, 3)
```

```
a
```

36. 更改数组形状（改变原始数组）

```
# resize 会改变原始数组
a.resize(2, 3)
```

```
a
```

37. 展平数组

```
a.ravel()
```

38. 垂直拼合数组

```
# 生成示例数组
a = np.random.randint(10, size=(3, 3))
b = np.random.randint(10, size=(3, 3))

a, b
```

```
np.vstack((a, b))
```

39. 水平拼合数组

```
np.hstack((a, b))
```

40. 沿横轴分割数组

```
np.hsplit(a, 3)
```

41. 沿纵轴分割数组

```
np.vsplit(a, 3)
```

数组排序

```
# 生成示例数组
a = np.array(([1, 4, 3], [6, 2, 9], [4, 7, 2]))
a
```

42. 返回每列最大值

```
np.max(a, axis=0)
```

43. 返回每行最小值

```
np.min(a, axis=1)
```

44. 返回每列最大值索引

```
np.argmax(a, axis=0)
```

45. 返回每行最小值索引

```
np.argmin(a, axis=1)
```

数组统计

46. 统计数组各列的中位数

```
# 继续使用上面的 a 数组
np.median(a, axis=0)
```

47. 统计数组各行的算术平均值

```
np.mean(a, axis=1)
```

48. 统计数组各列的加权平均值

```
np.average(a, axis=0)
```

49. 统计数组各行的方差

```
np.var(a, axis=1)
```

50. 统计数组各列的标准偏差

```
np.std(a, axis=0)
```

进阶部分

51. 创建一个 5x5 的二维数组，其中边界值为1，其余值为0

```
Z = np.ones((5,5))
Z[1:-1,1:-1] = 0
Z
```

52. 使用数字 0 将一个全为 1 的 5x5 二维数组包围

```
Z = np.ones((5,5))
Z = np.pad(Z, pad_width=1, mode='constant', constant_values=0)
Z
```

53. 创建一个 5x5 的二维数组，并设置值 1, 2, 3, 4 落在其对角线下方

```
Z = np.diag(1+np.arange(4),k=-1)
Z
```

54. 创建一个 10x10 的二维数组，并使得 1 和 0 沿对角线间隔放置

```
Z = np.zeros((10,10),dtype=int)
Z[1::2,::2] = 1
Z[::2,1::2] = 1
Z
```

55. 创建一个 0-10 的一维数组，并将 (1, 9] 之间的数全部反转成负数

```
Z = np.arange(11)
Z[(1 < Z) & (Z <= 9)] *= -1
Z
```

56. 找出两个一维数组中相同的元素

```
Z1 = np.random.randint(0,10,10)
Z2 = np.random.randint(0,10,10)
print("Z1:", Z1)
print("Z2:", Z2)
np.intersect1d(Z1,Z2)
```

57. 使用 NumPy 打印昨天、今天、明天的日期

```
yesterday = np.datetime64('today', 'D') - np.timedelta64(1, 'D')
today     = np.datetime64('today', 'D')
tomorrow  = np.datetime64('today', 'D') + np.timedelta64(1, 'D')
print("yesterday: ", yesterday)
print("today: ", today)
print("tomorrow: ", tomorrow)
```

58. 使用五种不同的方法去提取一个随机数组的整数部分

```
Z = np.random.uniform(0,10,10)
print("原始值: ", Z)

print ("方法 1: ", Z - Z%1)
print ("方法 2: ", np.floor(Z))
print ("方法 3: ", np.ceil(Z)-1)
print ("方法 4: ", Z.astype(int))
print ("方法 5: ", np.trunc(Z))
```

59. 创建一个 5x5 的矩阵，其中每行的数值范围从 1 到 5

```
Z = np.zeros((5,5))
Z += np.arange(1,6)

Z
```

60. 创建一个长度为 5 的等间隔一维数组，其值域范围从 0 到 1，但是不包括 0 和 1

```
Z = np.linspace(0,1,6,endpoint=False)[1:]

Z
```

61. 创建一个长度为10的随机一维数组，并将其按升序排序

```
Z = np.random.random(10)
Z.sort()
Z
```

62. 创建一个 3x3 的二维数组，并将列按升序排序

```
Z = np.array([[7,4,3],[3,1,2],[4,2,6]])
print("原始数组: \n", Z)

Z.sort(axis=0)
Z
```

63. 创建一个长度为 5 的一维数组，并将其中最大值替换成 0

```
Z = np.random.random(5)
print("原数组: ",Z)
Z[Z.argmax()] = 0
Z
```

64. 打印每个 NumPy 标量类型的最小值和最大值

```
for dtype in [np.int8, np.int32, np.int64]:
   print("The minimum value of {}: ".format(dtype), np.iinfo(dtype).min)
   print("The maximum value of {}: ".format(dtype),np.iinfo(dtype).max)
for dtype in [np.float32, np.float64]:
   print("The minimum value of {}: ".format(dtype),np.finfo(dtype).min)
   print("The maximum value of {}: ".format(dtype),np.finfo(dtype).max)
```

65. 将 `float32` 转换为整型

```
Z = np.arange(10, dtype=np.float32)
print(Z)

Z = Z.astype(np.int32, copy=False)
Z
```

66. 将随机二维数组按照第 3 列从上到下进行升序排列

```
Z = np.random.randint(0,10,(5,5))
print("排序前: \n",Z)

Z[Z[:,2].argsort()]
```

67. 从随机一维数组中找出距离给定数值（0.5）最近的数

```
Z = np.random.uniform(0,1,20)
print("随机数组: \n", Z)
z = 0.5
m = Z.flat[np.abs(Z - z).argmin()]

m
```

68. 将二维数组的前两行进行顺序交换

```
A = np.arange(25).reshape(5,5)
print(A)
A[[0,1]] = A[[1,0]]
print(A)
```

69. 找出随机一维数组中出现频率最高的值

```
Z = np.random.randint(0,10,50)
print("随机一维数组:", Z)
np.bincount(Z).argmax()
```

70. 找出给定一维数组中非 0 元素的位置索引

```
Z = np.nonzero([1,0,2,0,1,0,4,0])
Z
```

71. 对于给定的 5x5 二维数组，在其内部随机放置 p 个值为 1 的数

```
p = 3

Z = np.zeros((5,5))
np.put(Z, np.random.choice(range(5*5), p, replace=False),1)

Z
```

72. 对于随机的 3x3 二维数组，减去数组每一行的平均值

```
X = np.random.rand(3, 3)
print(X)

Y = X - X.mean(axis=1, keepdims=True)
Y
```

73. 获得二维数组点积结果的对角线数组

```
A = np.random.uniform(0,1,(3,3))
B = np.random.uniform(0,1,(3,3))

print(np.dot(A, B))

# 较慢的方法
np.diag(np.dot(A, B))
```

```
# 较快的方法
np.sum(A * B.T, axis=1)
```

```
# 更快的方法
np.einsum("ij, ji->i", A, B)
```

74. 找到随机一维数组中前 p 个最大值

```
Z = np.random.randint(1,100,100)
print(Z)

p = 5

Z[np.argsort(Z)[-p:]]
```

75. 计算随机一维数组中每个元素的 4 次方数值

```
x = np.random.randint(2,5,5)
print(x)

np.power(x,4)
```

76. 对于二维随机数组中各元素，保留其 2 位小数

```
Z = np.random.random((5,5))
print(Z)

np.set_printoptions(precision=2)
Z
```

77. 使用科学记数法输出 NumPy 数组

```
Z = np.random.random([5,5])
print(Z)

Z/1e3
```

78. 使用 NumPy 找出百分位数（25%，50%，75%）

```
a = np.arange(15)
print(a)

np.percentile(a, q=[25, 50, 75])
```

79. 找出数组中缺失值的总数及所在位置

```
# 生成含缺失值的 2 维数组
Z = np.random.rand(10,10)
Z[np.random.randint(10, size=5), np.random.randint(10, size=5)] = np.nan
Z
```

```
print("缺失值总数: \n", np.isnan(Z).sum())
print("缺失值索引: \n", np.where(np.isnan(Z)))
```

80. 从随机数组中删除包含缺失值的行

```
# 沿用 79 题中的含缺失值的 2 维数组

Z[np.sum(np.isnan(Z), axis=1) == 0]
```

81. 统计随机数组中的各元素的数量

```
Z = np.random.randint(0,100,25).reshape(5,5)
print(Z)

np.unique(Z, return_counts=True) # 返回值中，第 2 个数组对应第 1 个数组元素的数量
```

82. 将数组中各元素按指定分类转换为文本值

```
# 指定类别如下
# 1 → 汽车
# 2 → 公交车
# 3 → 火车


Z = np.random.randint(1,4,10)
print(Z)

label_map = {1: "汽车", 2: "公交车", 3: "火车"}

[label_map[x] for x in Z]
```

83. 将多个 1 维数组拼合为单个 Ndarray

```
Z1 = np.arange(3)
Z2 = np.arange(3,7)
Z3 = np.arange(7,10)

Z = np.array([Z1, Z2, Z3])
print(Z)

np.concatenate(Z)
```

84. 打印各元素在数组中升序排列的索引

```
a = np.random.randint(100, size=10)
print('Array: ', a)

a.argsort()
```

85. 得到二维随机数组各行的最大值

```
Z = np.random.randint(1,100, [5,5])
print(Z)

np.amax(Z, axis=1)
```

86. 得到二维随机数组各行的最小值（区别上面的方法）

```
Z = np.random.randint(1,100, [5,5])
print(Z)

np.apply_along_axis(np.min, arr=Z, axis=1)
```

87. 计算两个数组之间的欧氏距离

```
a = np.array([1, 2])
b = np.array([7, 8])

# 数学计算方法
print(np.sqrt(np.power((8-2), 2) + np.power((7-1), 2)))

# NumPy 计算
np.linalg.norm(b-a)
```

88. 打印复数的实部和虚部

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

print("实部: ", a.real)
print("虚部: ", a.imag)
```

89. 求解给出矩阵的逆矩阵并验证

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

inverse_matrix = np.linalg.inv(matrix)

# 验证原矩阵和逆矩阵的点积是否为单位矩阵
assert np.allclose(np.dot(matrix, inverse_matrix), np.eye(2))

inverse_matrix
```

90. 使用 Z-Score 标准化算法对数据进行标准化处理

Z-Score 标准化公式：

$$Z = \frac{X - \text{mean}(X)}{\text{sd}(X)}$$

```
# 根据公式定义函数
def zscore(x, axis = None):
    xmean = x.mean(axis=axis, keepdims=True)
    xstd  = np.std(x, axis=axis, keepdims=True)
    zscore = (x-xmean)/xstd
    return zscore

# 生成随机数据
Z = np.random.randint(10, size=(5,5))
print(Z)

zscore(Z)
```

91. 使用 Min-Max 标准化算法对数据进行标准化处理

Min-Max 标准化公式：

$$Y = \frac{Z - \min(Z)}{\max(Z) - \min(Z)}$$

```
# 根据公式定义函数
def min_max(x, axis=None):
    min = x.min(axis=axis, keepdims=True)
    max = x.max(axis=axis, keepdims=True)
    result = (x-min)/(max-min)
    return result

# 生成随机数据
Z = np.random.randint(10, size=(5,5))
print(Z)

min_max(Z)
```

92. 使用 L2 范数对数据进行标准化处理

L2 范数计算公式：

$$L_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_i^2}$$

```
# 根据公式定义函数
def l2_normalize(v, axis=-1, order=2):
    l2 = np.linalg.norm(v, ord = order, axis=axis, keepdims=True)
    l2[l2==0] = 1
    return v/l2

# 生成随机数据
Z = np.random.randint(10, size=(5,5))
print(Z)

l2_normalize(Z)
```

93. 使用 NumPy 计算变量直接的相关性系数

```
Z = np.array([
    [1, 2, 1, 9, 10, 3, 2, 6, 7], # 特征 A
    [2, 1, 8, 3, 7, 5, 10, 7, 2], # 特征 B
    [2, 1, 1, 8, 9, 4, 3, 5, 7]]) # 特征 C

np.corrcoef(Z)
```

相关性系数取值从 `[-1, 1]` 变换，靠近 1 则代表正相关性较强，-1 则代表负相关性较强。结果如下所示，变量 A 与变量 A 直接的相关性系数为 `1`，因为是同一个变量。变量 A 与变量 C 之间的相关性系数为 `0.97`，说明相关性较强。

```
        [A]     [B]     [C]
array([[ 1.  , -0.06,  0.97]   [A]
       [-0.06,  1.  , -0.01],  [B]
       [ 0.97, -0.01,  1.  ]]) [C]
```

94. 使用 NumPy 计算矩阵的特征值和特征向量

```
M = np.matrix([[1,2,3], [4,5,6], [7,8,9]])

w, v = np.linalg.eig(M)

# w 对应特征值，v 对应特征向量
w, v
```

我们可以通过 `P'AP=M` 公式反算，验证是否能得到原矩阵。

```
v * np.diag(w) * np.linalg.inv(v)
```

95. 使用 NumPy 计算 Ndarray 两相邻元素差值

```
Z = np.random.randint(1,10,10)
print(Z)

# 计算 Z 两相邻元素差值
print(np.diff(Z, n=1))

# 重复计算 2 次
print(np.diff(Z, n=2))

# 重复计算 3 次
print(np.diff(Z, n=3))
```

96. 使用 NumPy 将 Ndarray 相邻元素依次累加

```
Z = np.random.randint(1,10,10)
print(Z)

"""
[第一个元素, 第一个元素 + 第二个元素, 第一个元素 + 第二个元素 + 第三个元素, ...]
"""
np.cumsum(Z)
```

97. 使用 NumPy 按列连接两个数组

```
M1 = np.array([1, 2, 3])
M2 = np.array([4, 5, 6])

np.c_[M1, M2]
```

98. 使用 NumPy 按行连接两个数组

```
M1 = np.array([1, 2, 3])
M2 = np.array([4, 5, 6])

np.r_[M1, M2]
```

99. 使用 NumPy 打印九九乘法表

```
np.fromfunction(lambda i, j: (i + 1) * (j + 1), (9, 9))
```

100. 使用 NumPy 将实验楼 LOGO 转换为 Ndarray 数组

```
from io import BytesIO
from PIL import Image
import PIL, requests

# 通过链接下载图像
URL = 'https://static.shiyanlou.com/img/logo-black.png'
response = requests.get(URL)

# 将内容读取为图像
I = Image.open(BytesIO(response.content))

# 将图像转换为 Ndarray
shiyanlou = np.asarray(I)
shiyanlou
```

```
# 将转换后的 Ndarray 重新绘制成图像
from matplotlib import pyplot as plt
%matplotlib inline

plt.imshow(shiyanlou)
plt.show()
```

实验总结

如果你亲自动手做完了上面的 100 道练习题，相信你已经对 NumPy 模块的熟练程度又提升了不少。我们推荐你定期回顾这些题目，相信你一定会熟能生巧。本次实验涉及的知识点主要有：

- 创建数组
- 数组运算
- 数学函数
- 数组切片和索引
- 数组形状操作
- 数组排序
- 数组统计

实验中少量题目编译自：[100 numpy exercises](https://github.com/rougier/numpy-100) 。