

【北京大学】class2-神经网络优化-TF2.1学习笔记

[numpy.mgrid 探究](#)

[手写理解](#)

[参考：](#)

[tf.math.log](#)

[numpy.reshape](#)

[举例子](#)

[plt 绘图](#)

[绘制单点](#)

[绘制一组点](#)

[机器学习常用数据集](#)

[鸢尾花数据集-iris](#)

[加载数据集](#)

[数据内部存储](#)

[手写数字数据集-mnist](#)

[加载数据集](#)

[数据内部存储](#)

[fashion-mnist](#)

[加载数据集](#)

[数据内部存储](#)

numpy.mgrid 探究

[手写理解](#)

np.mgrid 用法. (生成2D网格点)

实数: 2 就表示间隔为2 [起, 终)
复数: 2j 就表示产生2个点 [起, 终] 求中间值

$\text{var1}, \text{var2} \dots = \text{np.mgrid}[\text{起:终:间隔}, \text{起:终:间隔}, \dots]$
第1维 dim 2

$0:5:3j$
[0, 5] 区间均匀
产生3个点, 即
0, 2.5, 5

举例: $x, y = \text{np.mgrid}[0:5:1, 0:2:3j]$

区间间隔为1
左闭右开

区间产生3个点
左闭右开

$x: [0, 0, 0],$
 $[1, 1, 1],$
 $[2, 2, 2],$
 $[3, 3, 3],$
 $[4, 4, 4]$

向右指
(按几列看x的值)

$y: [0, 1, 2],$
 $[0, 1, 2],$
 $[0, 1, 2],$
 $[0, 1, 2],$
 $[0, 1, 2]$

向下指
(按几行看y的值)

总结: 直观理解. $\text{np.mgrid}[0:5:1, 0:2:3j]$

dim1

dim2

X轴

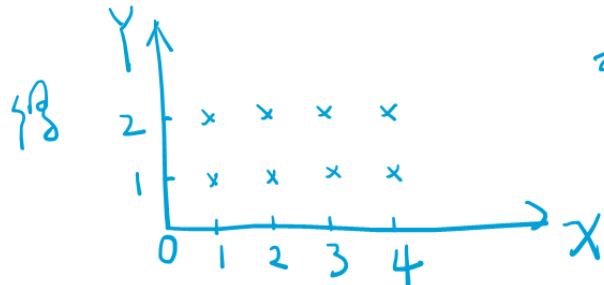
Y轴

坐标点: 0, 1, 2, 3, 4

坐标点: 0, 1, 2

打印出来的Tensor
让人看得摸不着
头绪, 但这样
几何理解还是
很直观的.

2022.7.10. 23点



于是我们得到2D中
的一系列散点

```
dot.csv
Folder where.py
p3_RandomState.py
p6_vstack.py
p7_mgrid.py
p10_backpropagation_decaylr.py
p19_mse.py
p20_custom.py
p22_ce.py
p23_softmaxce.py
p29_regularizationcontain.py

14 a, b = np.mgrid[0:5:1, 0:2:3j]
15 print("a:\n", a)
16 print("b:\n", b)
17
18
19
x 生成一列
0
1
2
3
4 该列向右拷贝 y(3)列，如下a所示

y 生成一行
0 1 2 该行向下拷贝 x(5)行，如左b所示

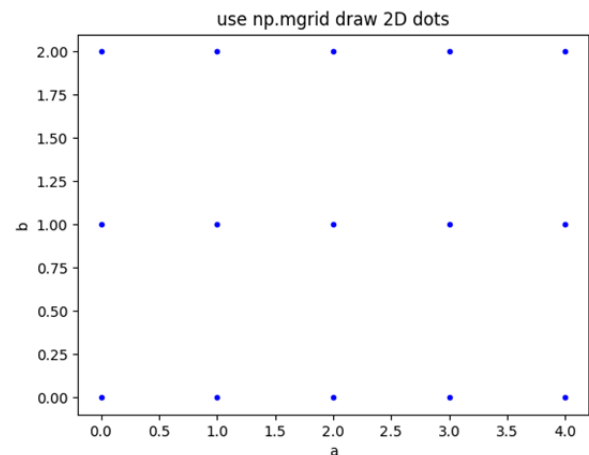
a:
[[0. 0. 0.]
 [1. 1. 1.]
 [2. 2. 2.]
 [3. 3. 3.]
 [4. 4. 4.]]
b:
[[0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]]
```

```
chapter2 E:\NIOU\【北京大学】TF2.14
class2
dot.csv
p4_where.py
p5_RandomState.py
p6_vstack.py
p7_mgrid.py
p10_backpropagation_decaylr.py
p19_mse.py
p20_custom.py
p22_ce.py
p23_softmaxce.py
p29_regularizationcontain.py
p29_regularizationfree.py
p32_sgd.py
p34_sgdm.py
v36_adasrad iv

15 a, b = np.mgrid[0:5:1, 0:2:3j]
16 print("a:\n", a)
17 print("b:\n", b)
18
19
20
21 plt.title("use np.mgrid draw 2D dots") # 图像标题
22 plt.xlabel("a") # x轴坐标
23 plt.ylabel("b") # y轴坐标
24 plt.plot(a, b, marker='.', color='b', linestyle='none') # 绘制点
25 plt.show() # 显示（勿忘）

Run: p7_mgrid x
a:
[[0. 0. 0.]
 [1. 1. 1.]
 [2. 2. 2.]
 [3. 3. 3.]
 [4. 4. 4.]]
b:
[[0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]]

Process finished with exit code 0
```



参考：

- [1] <https://xiaoiedu.blog.csdn.net/article/details/109526042>
- [2] <https://blog.csdn.net/naruhina/article/details/106824118>

参考：

- [1] <https://xiaoiedu.blog.csdn.net/article/details/109526042>
- [2] <https://blog.csdn.net/naruhina/article/details/106824118>

2022/7/10 23:30

tf.math.log

使用 tensorflow 求对数。

numpy.reshape

将 array 指定为几行几列。

reshape (行, 列)

行 = -1, 表示行不改变

列 = -1, 表示列不改变

举例子

Python | [复制代码](#)

```
1  import numpy as np
2
3  x = np.array([[2, 0], [1, 1], [2, 3]])
4  print("x:", x)
5
6  y1 = x.reshape(-1, 1) # 行不动, array改为一列
7  print("改为一列 y1:", y1)
8
9  y2 = x.reshape(1, -1) # 列不动, array 改为一行
10 print("改为一行 y2:", y2)
11
12 y3 = x.reshape(2, 3) # 行和列都改变, array 改为两行, 三列
13 print("改为两行三列 y3:", y3)
```

main.py

```
1 import numpy as np
2
3 x = np.array([[2, 0], [1, 1], [2, 3]])
4 print("x:", x)
5
6 y1 = x.reshape(-1, 1) # 行不动, array改为一列
7 print("改为一列 y1:", y1)
8
9 y2 = x.reshape(1, -1) # 列不动, array 改为一行
10 print("改为一行 y2:", y2)
11
12 y3 = x.reshape(2, 3) # 行和列都改变, array 改为两行, 三列
13 print("改为两行三列 y3:", y3)
```

input

```
x: [[2 0]
     [1 1]
     [2 3]]
改为一列 y1: [[2]
              [0]
              [1]
              [1]
              [2]
              [3]]
改为一行 y2: [[2 0 1 1 2 3]]
改为两行三列 y3: [[2 0 1]
                  [1 2 3]]
```

plt 绘图

from matplotlib import pyplot as plt

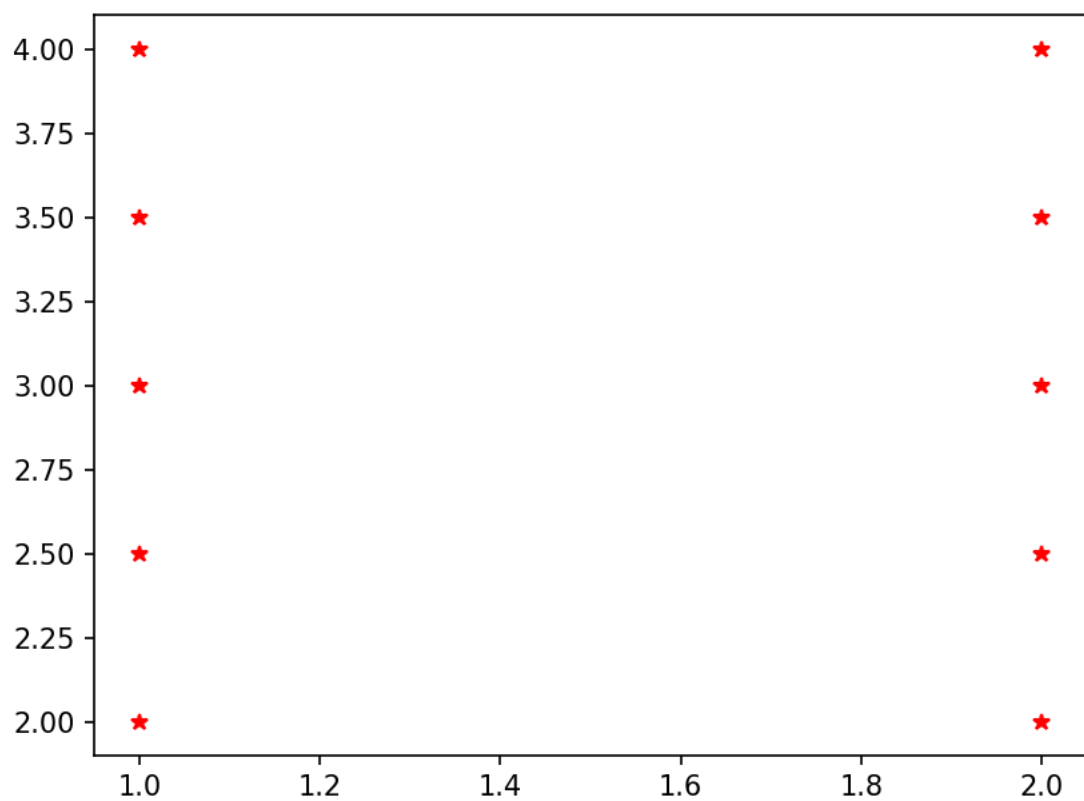
单凭经验+实验验证, 没查资料哦

```
1  import tensorflow as tf
2  import numpy as np
3  from matplotlib import pyplot as plt
4
5  # x: 1,2   y:2,2.5,3,3.5,4
6  x, y = np.mgrid[1:3:1, 2:4:5j]
7  x_flatton = x.ravel()
8  y_flatton = y.ravel()
9
10 grid = np.c_[x_flatton, y_flatton]
11
12 plt.title("np.mgrid generate 2D dots")
13 plt.xlabel("x: 1, 2 two ")
14 plt.ylabel("y: 2, 2.5, 3, 3.5, 4 five")
15
16 # 一组点绘制
17 x_dots = grid[:, 0]   # 效果等于 x_flatton
18 y_dots = grid[:, 1]   # 效果等于 y_flatton
19 plt.scatter(x_dots, y_dots, marker='*', color='b')
20 plt.show()
21
22 # 单点绘制
23 for index, dot in enumerate(grid):
24     print("%i: dot[%f][%f]" % (index, dot[0], dot[1]))
25     plt.plot(dot[0], dot[1], marker='*', color='r', linestyle='none')
26 plt.show()
27
```

```
1  点:
2  grid: [[1.  2. ]
3         [1.  2.5]
4         [1.  3. ]
5         [1.  3.5]
6         [1.  4. ]
7         [2.  2. ]
8         [2.  2.5]
9         [2.  3. ]
10        [2.  3.5]
11        [2.  4. ]]
12
13  单点:
14  0: dot[1.000000][2.000000]
15  1: dot[1.000000][2.500000]
16  2: dot[1.000000][3.000000]
17  3: dot[1.000000][3.500000]
18  4: dot[1.000000][4.000000]
19  5: dot[2.000000][2.000000]
20  6: dot[2.000000][2.500000]
21  7: dot[2.000000][3.000000]
22  8: dot[2.000000][3.500000]
23  9: dot[2.000000][4.000000]
24
25
26  一组点:
27  x_dots: [1.  1.  1.  1.  1.  2.  2.  2.  2.  2.]
28  y_dots: [2.  2.5  3.  3.5  4.  2.  2.5  3.  3.5  4. ]
29
30
```

绘制单点

```
1  for index, dot in enumerate(grid):
2      #         x坐标    y坐标
3      plt.plot(dot[0], dot[1], marker='*', color='r', linestyle='none')
```



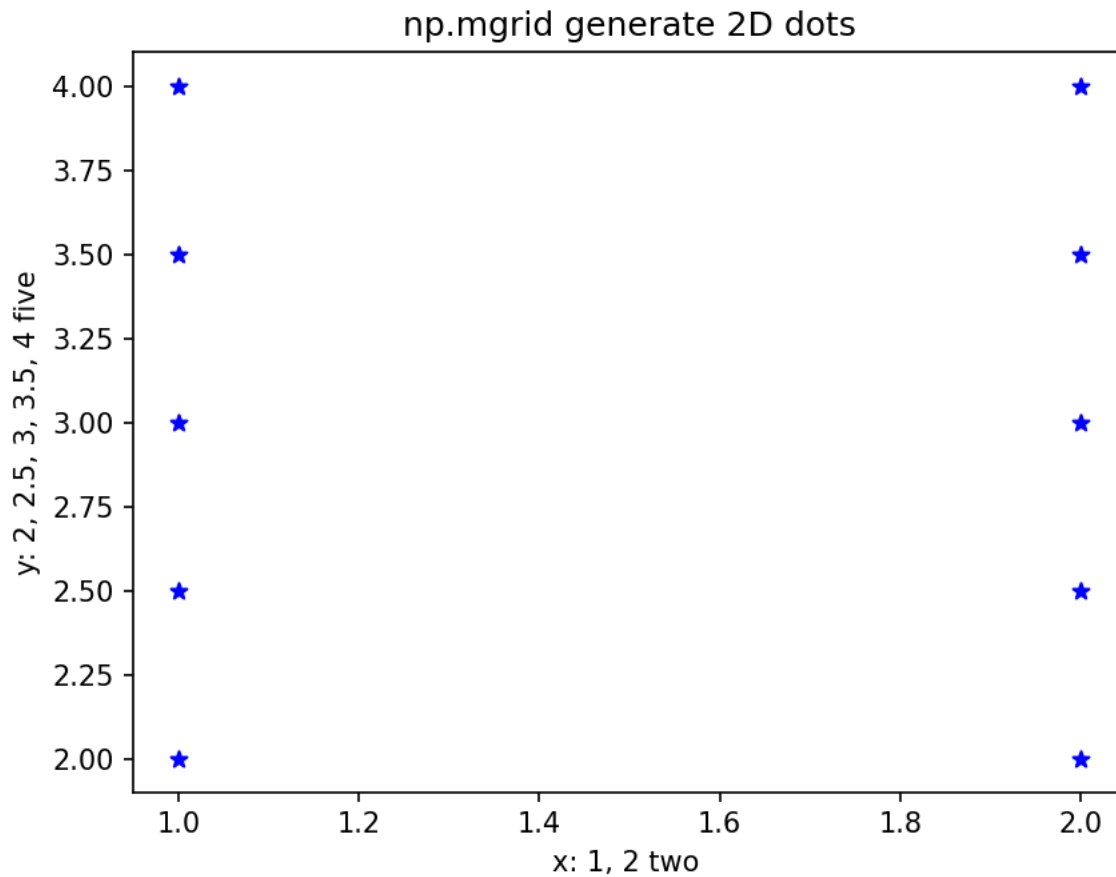
绘制一组点



Python

复制代码

```
1         x坐标向量 y坐标向量
2 plt.scatter(x_dots, y_dots, marker='*', color='b')
```

机器学习常用数据集

鸢尾花数据集-iris

[p45_iris_main.py](#)

加载数据集

```
import tensorflow as tf
from sklearn import datasets
from matplotlib import pyplot as plt
import numpy as np

# 导入数据，分别为特征和标签
x_features = datasets.load_iris().data
y_labels = datasets.load_iris().target
y_classes = datasets.load_iris().target_names
# print(datasets.load_iris())
# print(y_classes)
```

数据内部存储

[illegible]

PC x_features

	0	1	2	3
131	7.90000	3.80000	6.40000	2.00000
132	6.40000	2.80000	5.60000	2.20000
133	6.30000	2.80000	5.10000	1.50000
134	6.10000	2.60000	5.60000	1.40000
135	7.70000	3.00000	6.10000	2.30000
136	6.30000	3.40000	5.60000	2.40000
137	6.40000	3.10000	5.50000	1.80000
138	6.00000	3.00000	4.80000	1.80000
139	6.90000	3.10000	5.40000	2.10000
140	6.70000	3.10000	5.60000	2.40000
141	6.90000	3.10000	5.10000	2.30000
142	5.80000	2.70000	5.10000	1.90000
143	6.80000	3.20000	5.90000	2.30000
144	6.70000	3.30000	5.70000	2.50000
145	6.70000	3.00000	5.20000	2.30000
146	6.30000	2.50000	5.00000	1.90000
147	6.50000	3.00000	5.20000	2.00000
148	6.20000	3.40000	5.40000	2.30000
149	5.90000	3.00000	5.10000	1.80000

x_features Format: %.5f

☒ Colored cells
☒ Resize automatically

Close

手写数字数据集-mnist

[p13_mnist_datasets.py](#)

加载数据集

```

1 import tensorflow as tf
2 from matplotlib import pyplot as plt
3
4 mnist = tf.keras.datasets.mnist
5 # print("mnist:\n", mnist) # 给出的是路径。并不是实际的数据
6 (x_train, y_train), (x_test, y_test) = mnist.load_data()
7

```

```

import tensorflow as tf x_test.min: <built-in method min of numpy.ndarray object at 0x000001697FE0D710>
from matplotlib import pyplot as plt

mnist = tf.keras.datasets.mnist
# print("mnist:\n", mnist) # 给出的是路径。并不是实际的数据
(x_train, y_train), (x_test, y_test) = mnist.load_data() x_test: [[[0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], ..., [0 0 0 ...

# 可视化训练集输入特征的第一个元素
plt.imshow(x_train[0], cmap='gray') # 绘制灰度图
plt.show()

```

数据内部存储

```

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
> x_test.min = (builtin_function_or_method) <built-in method min of numpy.ndarray object at 0x000001697FE0D710>
> x_test = (ndarray: (10000, 28, 28)) [[[0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], ..., [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0]...View as Array
> x_train = (ndarray: (60000, 28, 28)) [[[0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], ..., [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0]...View as Array
> y_test = (ndarray: (10000,)) [7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4 0 7 4 0 1 3 1 3 4 7 2 7, 1 2 1 1 7 4 2 3 5 1 2 4 4 6 3 5 5 6 0 4 1 9 5 7 8 9 3 7 4 6 4 3 0 7 0 2 9, 1 7 3 2 9 7 7 ...View as Array
> y_train = (ndarray: (60000,)) [5 0 4 1 9 2 1 3 1 4 3 5 3 6 1 7 2 8 6 9 4 0 9 1 1 2 4 3 2 7 3 8 6 9 0 5 6, 0 7 6 1 8 7 9 3 9 8 5 9 3 3 0 7 4 9 8 0 9 4 1 4 4 6 0 4 5 6 1 0 0 1 7 1 6, 3 0 2 1 1 7 9 ...View as Array
> Special Variables

```

训练样本: 60000张, 28x28 pixel的灰度图像
 训练标签: 60000张, 0~9 之间的数字 (ground truth)

测试样本: 10000张, 28x28 pixel的灰度图像
 测试标签: 10000张, 0~9 之间的数字 (ground truth)

```

> x_test.min = (builtin_function_or_method) <built-in method min of numpy.ndarray object at 0x000001697FE0D710>
> x_test = (ndarray: (10000, 28, 28)) [[[0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], ..., [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0]...View as Array
> x_train = (ndarray: (60000, 28, 28)) [[[0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], ..., [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0], [0 0 0 ... 0 0 0]...View as Array
  min = (str) 'ndarray too big, calculating min would slow down debugging'
  max = (str) 'ndarray too big, calculating max would slow down debugging'
> shape = (tuple: 3) (60000, 28, 28) uint8 格式存储像素灰度值
> dtype = (dtype[uint8]: 0) uint8
  size = (int) 47040000
> array = (NdArrayItemsContainer) <pydevd_plugins.extensions.types.pydevd_plugin_numpy_types.NdArrayItemsContainer object at 0x000001697FE246D8>
> y_test = (ndarray: (10000,)) [7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4 0 7 4 0 1 3 1 3 4 7 2 7, 1 2 1 1 7 4 2 3 5 1 2 4 4 6 3 5 5 6 0 4 1 9 5 7 8 9 3 7 4 6 4 3 0 7 0 2 9, 1 7 3 2 9 7 7 ...View as Array
> y_train = (ndarray: (60000,)) [5 0 4 1 9 2 1 3 1 4 3 5 3 6 1 7 2 8 6 9 4 0 9 1 1 2 4 3 2 7 3 8 6 9 0 5 6, 0 7 6 1 8 7 9 3 9 8 5 9 3 3 0 7 4 9 8 0 9 4 1 4 4 6 0 4 5 6 1 0 0 1 7 1 6, 3 0 2 1 1 7 9 ...View as Array
  min = (uint8: 0) 0
  max = (uint8: 0) 9
  shape = (tuple: 1) 60000 unit8 格式存储手写数字标签值
  dtype = (dtype[uint8]: 0) uint8
  size = (int) 60000
  array = (NdArrayItemsContainer) <pydevd_plugins.extensions.types.pydevd_plugin_numpy_types.NdArrayItemsContainer object at 0x000001697FE249B0>
> Special Variables

```

第一张测试图片: 可以看出, 是手写数字5

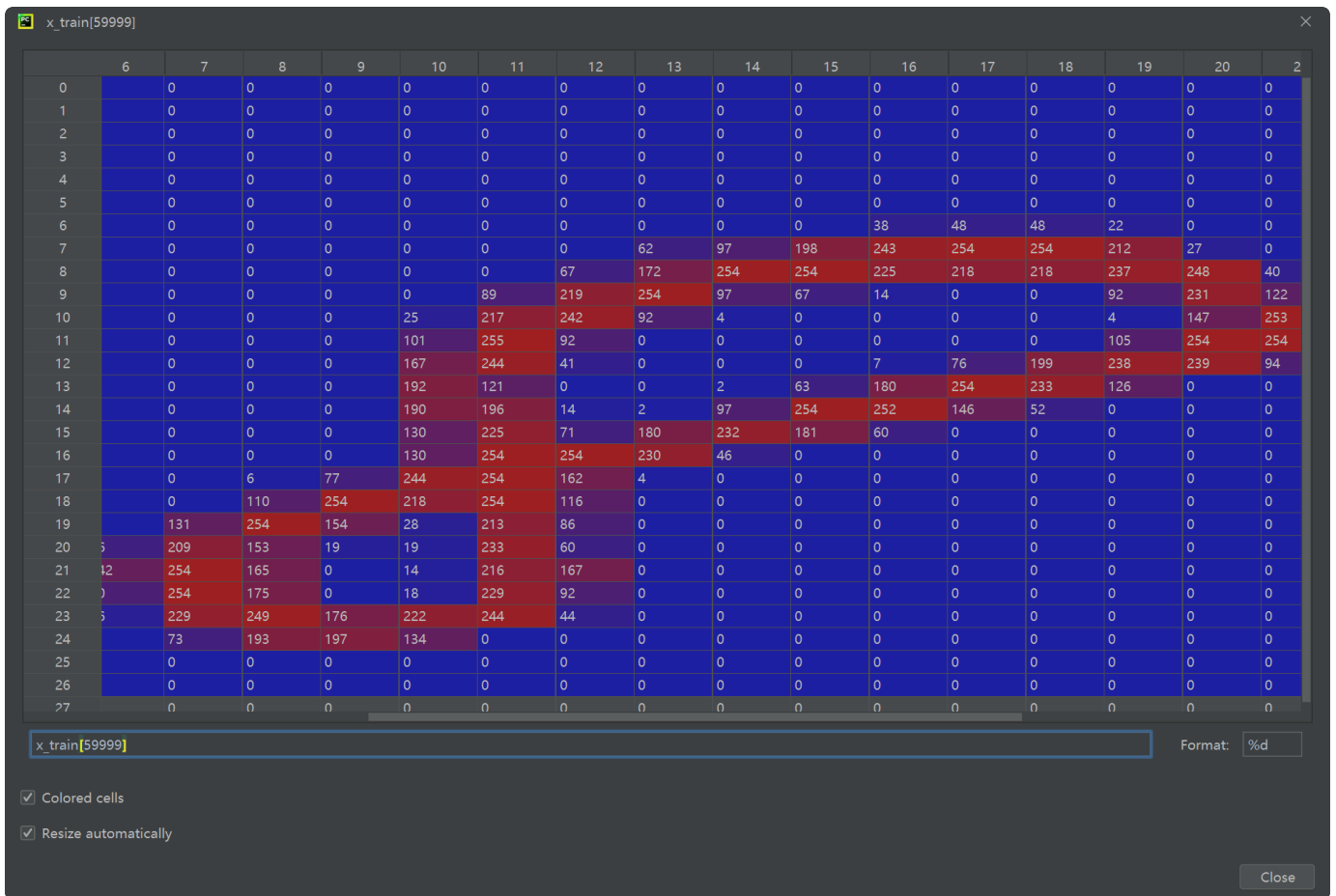
	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127	0
6	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0
7	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0
8	219	253	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0
9	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0
10	0	14	1	154	253	90	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	11	190	253	70	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	35	241	225	160	108	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0
15	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0
16	0	0	0	0	0	0	0	16	93	252	253	187	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	249	253	249	64	0	0	0	0	0
18	0	0	0	0	0	0	46	130	183	253	253	207	2	0	0	0	0	0
19	0	0	0	0	39	148	229	253	253	253	250	182	0	0	0	0	0	0
20	0	0	24	114	221	253	253	253	253	201	78	0	0	0	0	0	0	0
21	23	66	213	253	253	253	253	198	81	2	0	0	0	0	0	0	0	0
22	219	253	253	253	253	195	80	9	0	0	0	0	0	0	0	0	0	0
23	253	253	253	244	133	11	0	0	0	0	0	0	0	0	0	0	0	0
24	212	135	132	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

x_train[0]
Format: %d

☒ Colored cells
☒ Resize automatically

Close

第60000张测试图片：可以看出，是手写数字8



画面中每一格的数字就是该像素点的灰度值（0-255）

训练标签：一共60000个手写数字类别，一张图片对应一个数字标签

PC

y_train

×

	59992	59993	59994	59995	59996	59997	59998	59999
0	9	5	1	8	3	5	6	8

y_train

Format: %d

☒ Colored cells

☒ Resize automatically

Close

测试集特征, x_test

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

测试集第一个特征

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

测试集第二个特征

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

测试集第三个特征

...| 一共10000个特征

mnist训练/测试集数据
存储方式不是张量tensor,
直接是一个numpy三维数组, 里面
存放这10000个二维手写数字灰度值矩阵

...

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

测试数据的特征

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

index=9998 的特征图像

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

index=9999 的特征图像

测试集特征对应标签, y_test

```
[7 2 1 ... 4 5 6]
```

测试集的标签: 10000个0-9的一维numpy数组

fashion-mnist

<https://github.com/zalandoresearch/fashion-mnist>

Fashion-MNIST is a dataset comprising of **28×28** grayscale images of **70,000** fashion products from **10** categories, with **7,000** images per category. The training set has **60,000** images and the test set has **10,000** images. Fashion-MNIST shares the same image size, data format and the structure of training and testing splits with the original MNIST.

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

加载数据集

```

1  # 导入相应模块
2  import tensorflow as tf
3  from matplotlib import pyplot as plt
4
5  # 数据读取
6  fashion = tf.keras.datasets.fashion_mnist
7  (x_train, y_train), (x_test, y_test) = fashion.load_data()

```

数据内部存储

```

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
> x_test = (ndarray: (10000, 28, 28)) [[[ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0], ..., [ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0]]...View as Array
> x_train = (ndarray: (60000, 28, 28)) [[[0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0], ..., [ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0]],, [[0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0], [ 0 0 0 ... 0 0 0]]...View as Array
  min = (str) 'ndarray too big, calculating min would slow down debugging'
  max = (str) 'ndarray too big, calculating max would slow down debugging'
> shape = (tuple: 3) (60000, 28, 28)
> dtype = (dtype[uint8]: ()) uint8
  size = (int) 47040000
> array = (NdArrayItemsContainer) <pydevd_plugins.extensions.types.pydevd_plugin_numpy_types.NdArrayItemsContainer object at 0x000002B3F80F9550>
> y_test = (ndarray: (10000,)) [9 2 1 1 6 1 4 6 5 7 4 5 7 3 4 1 2 4 8 0 2 5 7 9 1 4 6 0 9 3 8 8 3 3 8 0 7, 5 7 9 6 1 3 7 6 7 2 1 2 2 4 4 5 8 2 2 8 4 8 0 7 7 8 5 1 1 2 3 9 8 7 0 2 6, 2 3 1 2 8 4 1 ...View as Array
> y_train = (ndarray: (60000,)) [9 0 0 3 0 2 7 2 5 5 0 9 5 5 7 9 1 0 6 4 3 1 4 8 4 3 0 2 4 4 5 3 6 6 0 8 5, 2 1 6 6 7 9 5 9 2 7 3 0 3 3 3 7 2 2 6 6 8 3 3 5 0 5 5 0 2 0 0 4 1 3 1 6 3, 1 4 4 6 1 9 1...View as Array
  min = (uint8: ()) 0
  max = (uint8: ()) 9
  shape = (tuple: 1) 60000
  dtype = (dtype[uint8]: ()) uint8
  size = (int) 60000
> array = (NdArrayItemsContainer) <pydevd_plugins.extensions.types.pydevd_plugin_numpy_types.NdArrayItemsContainer object at 0x000002B3D336F9E8>
> Special Variables

```

2022-7-15