【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

np.mgrid 用法.（生成2D网格点）

实数：2就表示间隔为2 [起,终)

复数：3j就表示产生3个点 [起,终] 求闭区间值

0:5:3j

[0,5]之间均匀产生3个点，即 0, 2.5, 5

var1, var2 ... = np.mgrid[起:终:间隔, 起:终:间隔, ...]

var1 操作第1维 ... 第1维 dim2 ...

举例：x, y = 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，左闭右闭

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

竖着看 5

向右拷贝（拷几列看y的值）

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

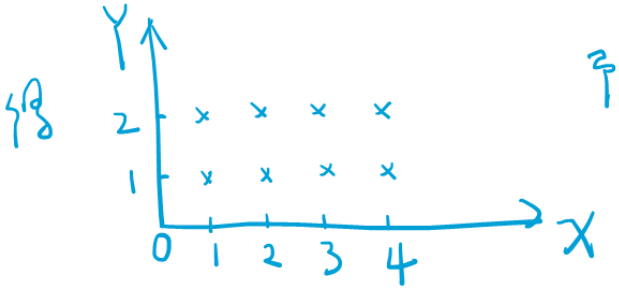
横着看 5

向下拷贝（拷几行看x的值）

总结：直观理解 np.mgrid[0:5:1, 0:2:3j]

dim1 → X轴 → 数据点：0, 1, 2, 3, 4

dim2 → Y轴 → 数据点：0, 1, 2

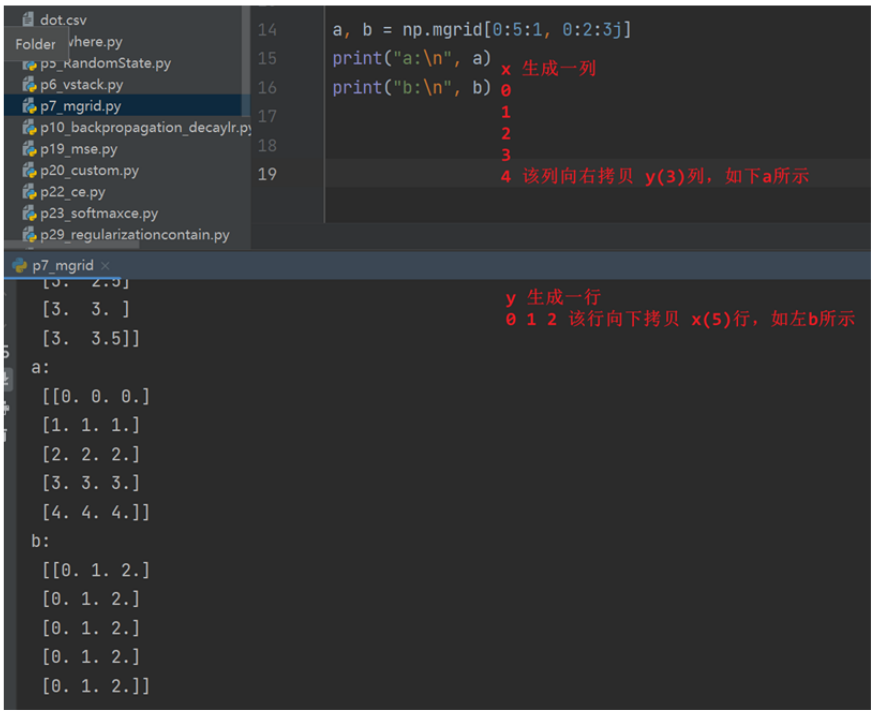


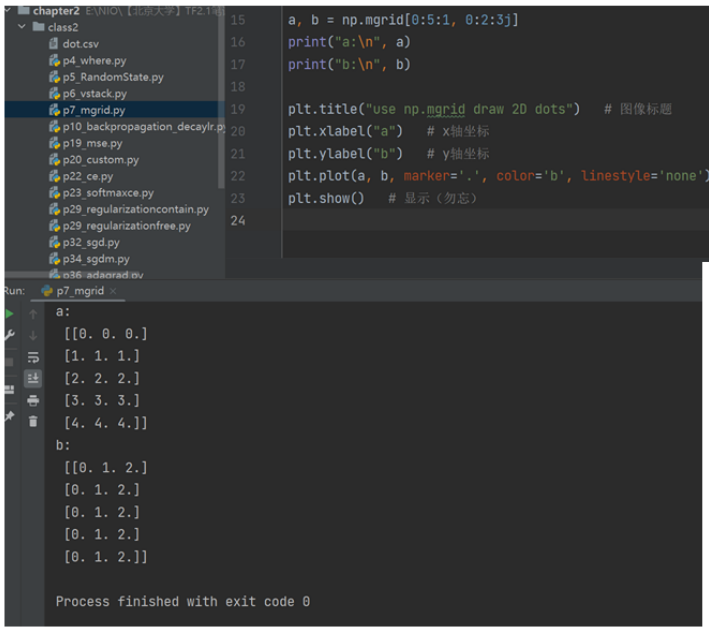
得

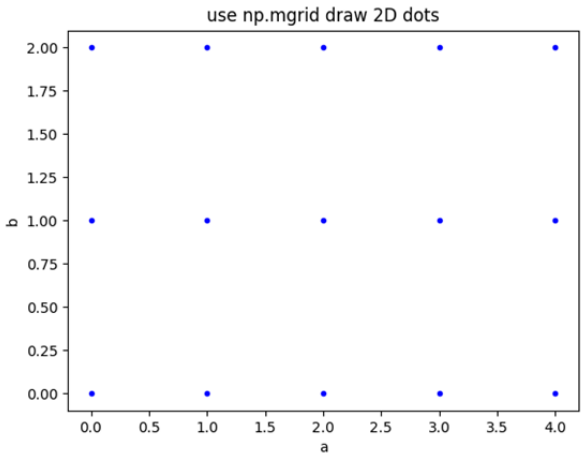
于我们得到2D中的一系列数点

打印出来的Tensor让人看得摸不着头脑，但这样几何理解还是很直观的.

2022.7.10. 23点







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

```python
import numpy as np

x = np.array([[2, 0], [1, 1], [2, 3]])
print("x:", x)

y1 = x.reshape(-1, 1)  # 行不动，array改为一列
print("改为一列 y1:", y1)

y2 = x.reshape(1, -1)  # 列不动，array 改为一行
print("改为一行 y2:", y2)

y3 = x.reshape(2, 3)  # 行和列都改变，array 改为两行，三列
print("改为两行三列 y3:", y3)
```

main.py

```python
import numpy as np

x = np.array([[2, 0], [1, 1], [2, 3]])
print("x:", x)

y1 = x.reshape(-1, 1)  # 行不动，array改为一列
print("改为一列 y1:", y1)

y2 = x.reshape(1, -1)  # 列不动，array 改为一行
print("改为一行 y2:", y2)

y3 = x.reshape(2, 3)  # 行和列都改变，array 改为两行，三列
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

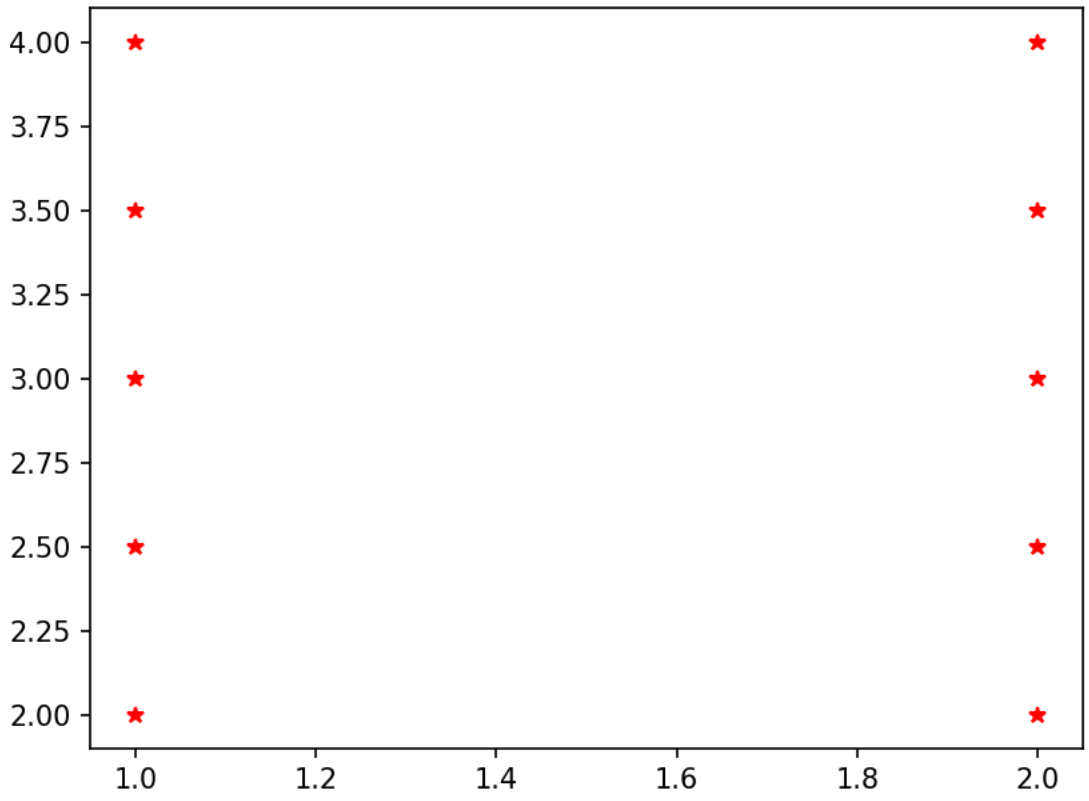
Python | 复制代码

```python
import tensorflow as tf
import numpy as np
from matplotlib import pyplot as plt

# x: 1,2  y:2,2.5,3,3.5,4
x, y = np.mgrid[1:3:1, 2:4:5j]
x_flatton = x.ravel()
y_flatton = y.ravel()

grid = np.c_[x_flatton, y_flatton]

plt.title("np.mgrid generate 2D dots")
plt.xlabel("x: 1, 2 two ")
plt.ylabel("y: 2, 2.5, 3, 3.5, 4 five")

# 一组点绘制
x_dots = grid[:, 0]   # 效果等于 x_flatton
y_dots = grid[:, 1]   # 效果等于 y_flatton
plt.scatter(x_dots, y_dots, marker='*', color='b')
plt.show()

# 单点绘制
for index, dot in enumerate(grid):
    print("%i: dot[%f][%f]" % (index, dot[0], dot[1]))
    plt.plot(dot[0], dot[1], marker='*', color='r', linestyle='none')
plt.show()

```

```
点:
grid: [[1.  2. ]
 [1.  2.5]
 [1.  3. ]
 [1.  3.5]
 [1.  4. ]
 [2.  2. ]
 [2.  2.5]
 [2.  3. ]
 [2.  3.5]
 [2.  4. ]]

单点:
0: dot[1.000000][2.000000]
1: dot[1.000000][2.500000]
2: dot[1.000000][3.000000]
3: dot[1.000000][3.500000]
4: dot[1.000000][4.000000]
5: dot[2.000000][2.000000]
6: dot[2.000000][2.500000]
7: dot[2.000000][3.000000]
8: dot[2.000000][3.500000]
9: dot[2.000000][4.000000]


一组点:
x_dots: [1. 1. 1. 1. 1. 2. 2. 2. 2. 2.]
y_dots: [2.  2.5 3.  3.5 4.  2.  2.5 3.  3.5 4. ]
```

绘制单点

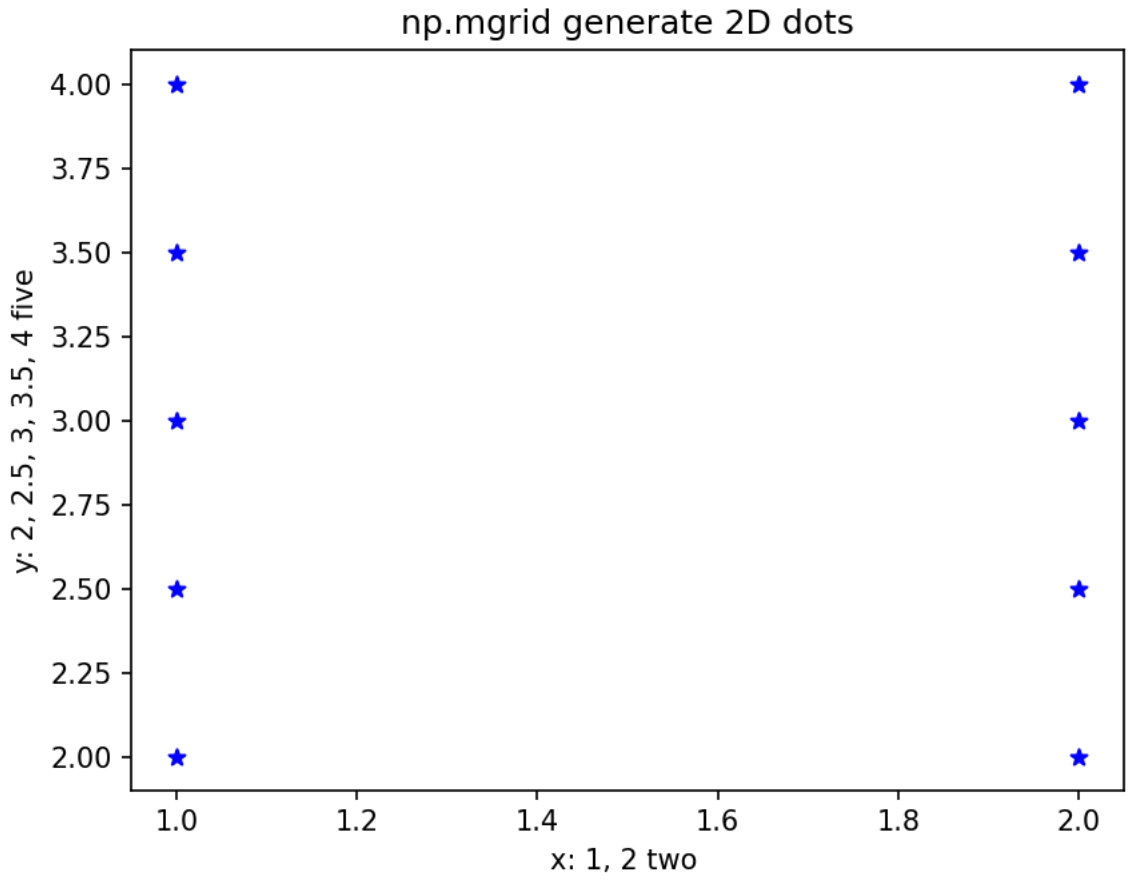
```python
for index, dot in enumerate(grid):
    #          x坐标    y坐标
    plt.plot(dot[0], dot[1], marker='*', color='r', linestyle='none')
```



绘制一组点

Python | 复制代码

```
            x坐标向量   y坐标向量
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

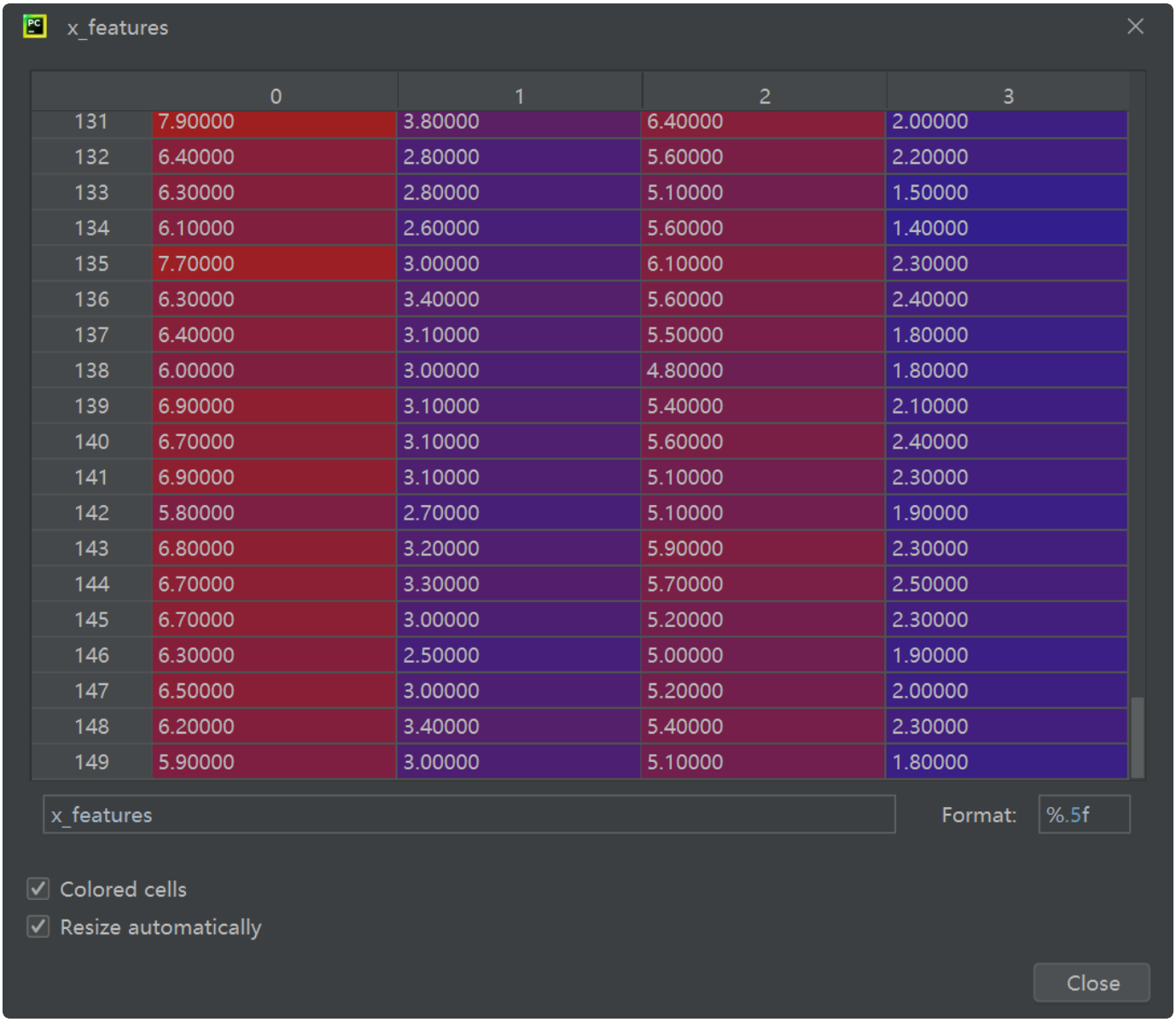
加载数据集

```python
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)
```

```python
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   x_features: [[5.1 3.5 1.4 0.2], [4.9 3.  1.4 0.2], [4.7 3.2 1.3 0.2], [4.6 3.1 1.5 0.2], [5.  3.6 1.4 0.
y_labels = datasets.load_iris().target   y_labels: [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
y_classes = datasets.load_iris().target_names   y_classes: ['setosa' 'versicolor' 'virginica']
# print(datasets.load_iris())
# print(y_classes)
```

数据内部存储

```
x_features = {ndarray: (150, 4)} [[5.1 3.5 1.4 0.2], [4.9 3.  1.4 0.2], [4.7 3.2 1.3 0.2], [4.6 3.1 1.5 0.2], [5.  3.6 1.4 0.2], [5.4 3.9 1.7 0.4], [4.6 3.4 1.4 0.3], [5.  3.4 1.5 0.2], [4.4 2.9 1.4 0.2], [4.!...View as Array
    min = {float64: ()} 0.1
    max = {float64: ()} 7.9                    每个样本是4维特征，共150个样本。
    shape = {tuple: 2} (150, 4)
    dtype = {dtype[float64]: ()} float64
    size = {int} 600
    array = {NdArrayItemsContainer} <pydevd_plugins.extensions.types.pydevd_plugin_numpy_types.NdArrayItemsContainer object at 0x00000264ADB199B0>
y_classes = {ndarray: (3,)} ['setosa' 'versicolor' 'virginica'] ...View as Array
    min = {str} 'not a numeric object'
    max = {str} 'not a numeric object'         三个类别：狗尾鸢尾，杂色鸢尾，弗吉尼亚鸢尾
    shape = {tuple: 1} 3                                 0          1            2          引伸：model.compile(..., metrics=["accuracy"])
    dtype = {dtype[str_]: ()} <U10                                                          (假设网络输出为值，这里标签也是值，用“accuracy”)
    size = {int} 3
    array = {NdArrayItemsContainer} <pydevd_plugins.extensions.types.pydevd_plugin_numpy_types.NdArrayItemsContainer object at 0x00000264ADB19DA0>
y_labels = {ndarray: (150,)} [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 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 1 ...View as Array
    min = {int32: ()} 0                        标签就是：对应150个0，1，2类别
    max = {int32: ()} 2
    shape = {tuple: 1} 150
    dtype = {dtype[int32]: ()} int32
    size = {int} 150
    array = {NdArrayItemsContainer} <pydevd_plugins.extensions.types.pydevd_plugin_numpy_types.NdArrayItemsContainer object at 0x00000264ADB27048>
```



手写数字数据集–mnist

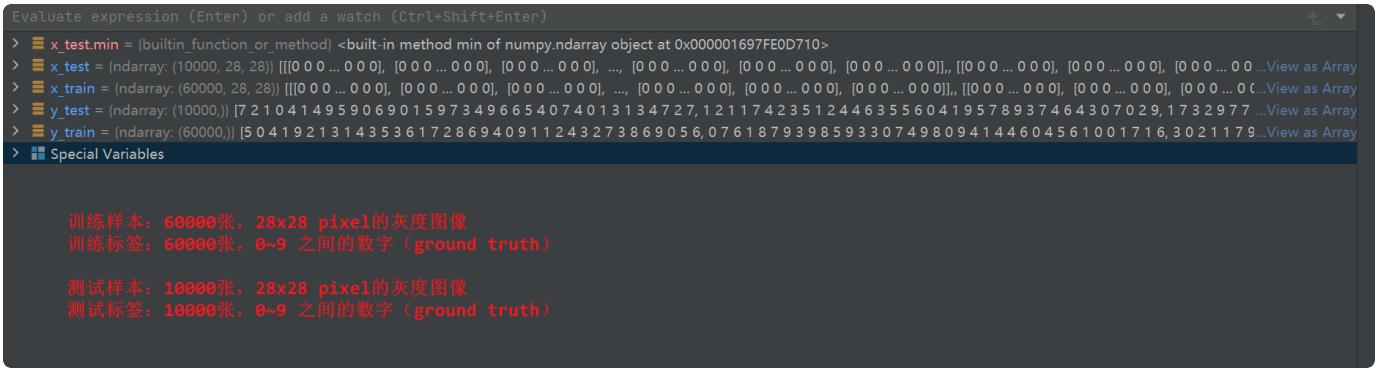
📎p13_mnist_datasets.py

加载数据集

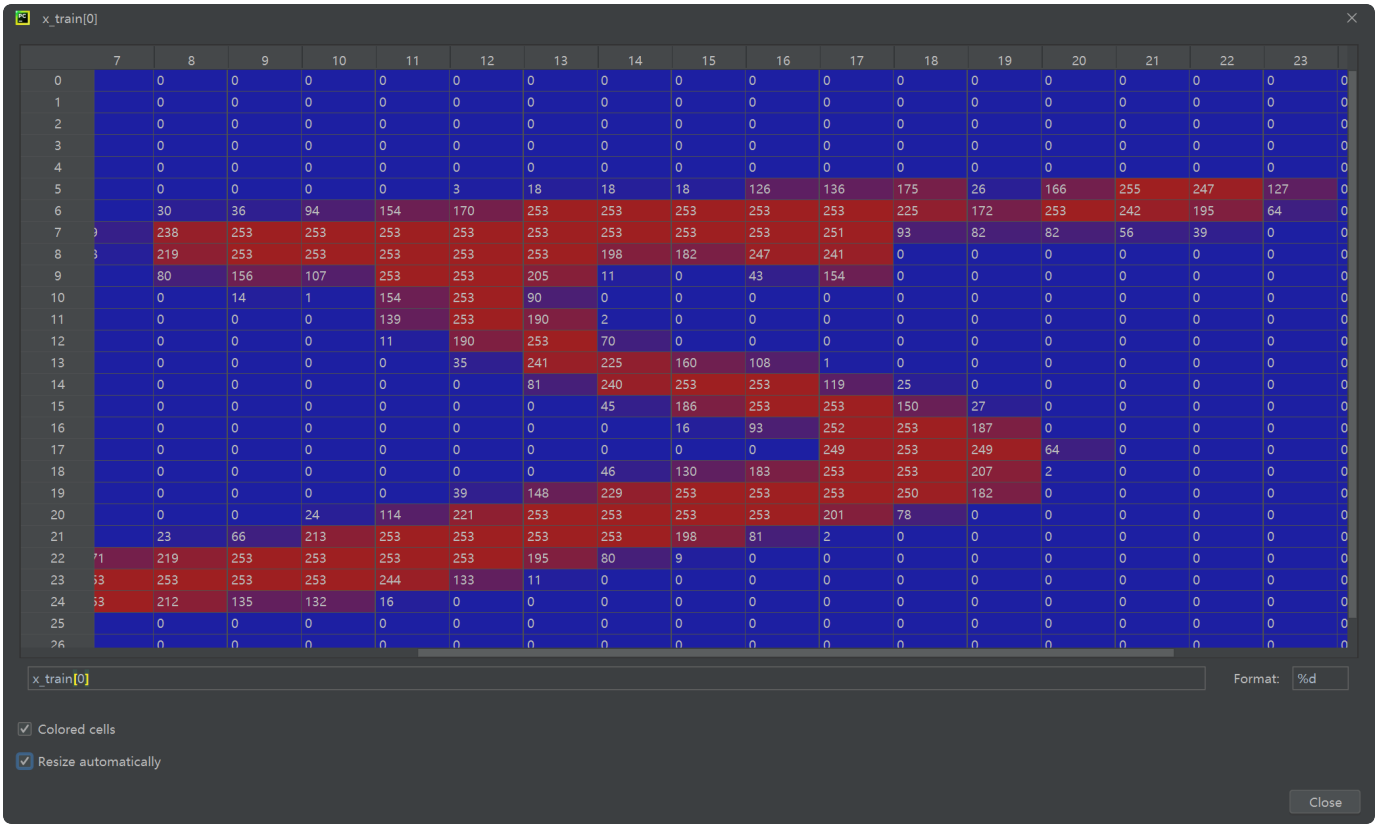
```python
import tensorflow as tf
from matplotlib import pyplot as plt

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()

```

```python
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()
```

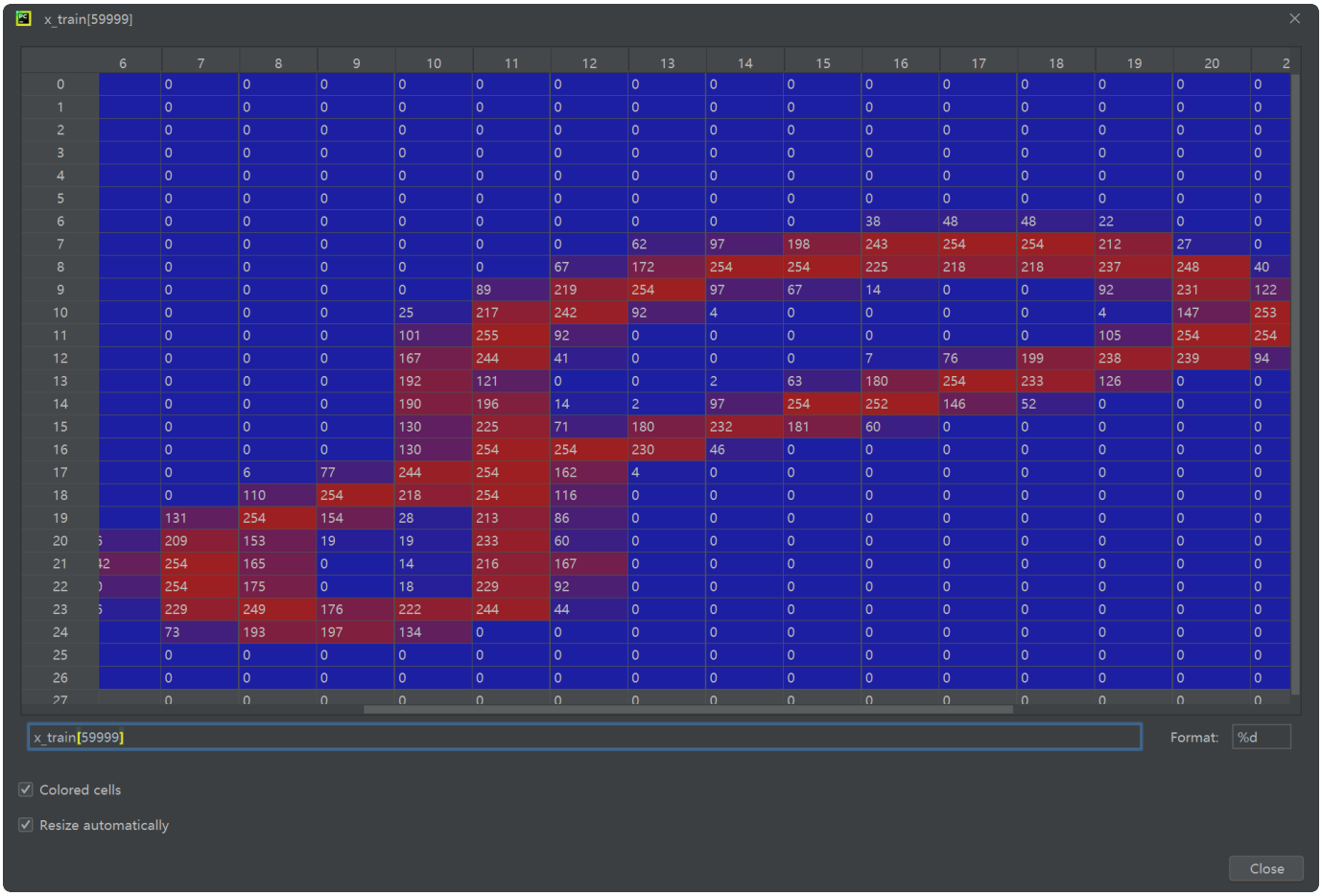
数据内部存储



第一张测试图片：可以看出，是手写数字5

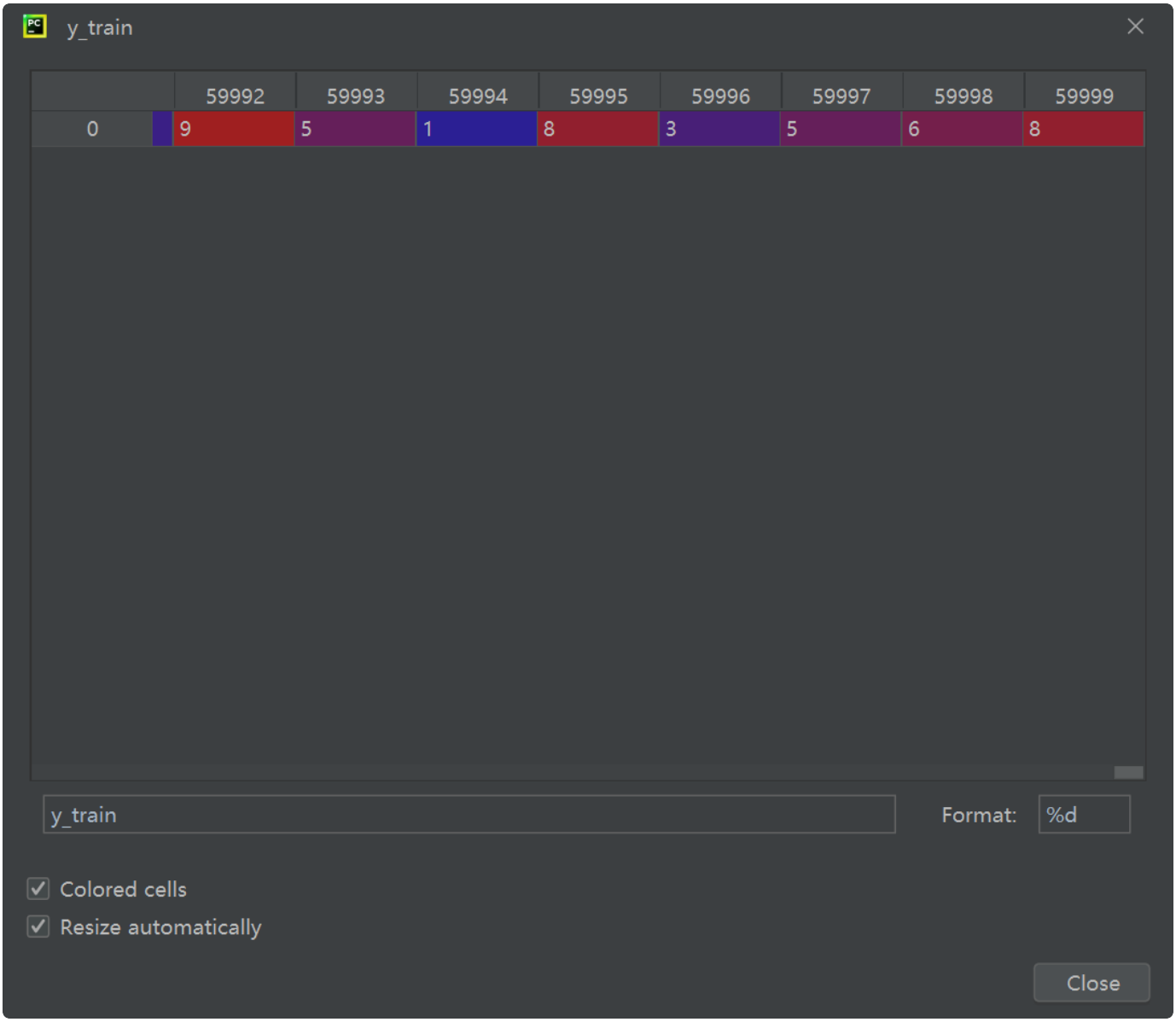


第60000张测试图片：可以看出，是手写数字8



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


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

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]
 ...
 [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

```
  ...
  [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

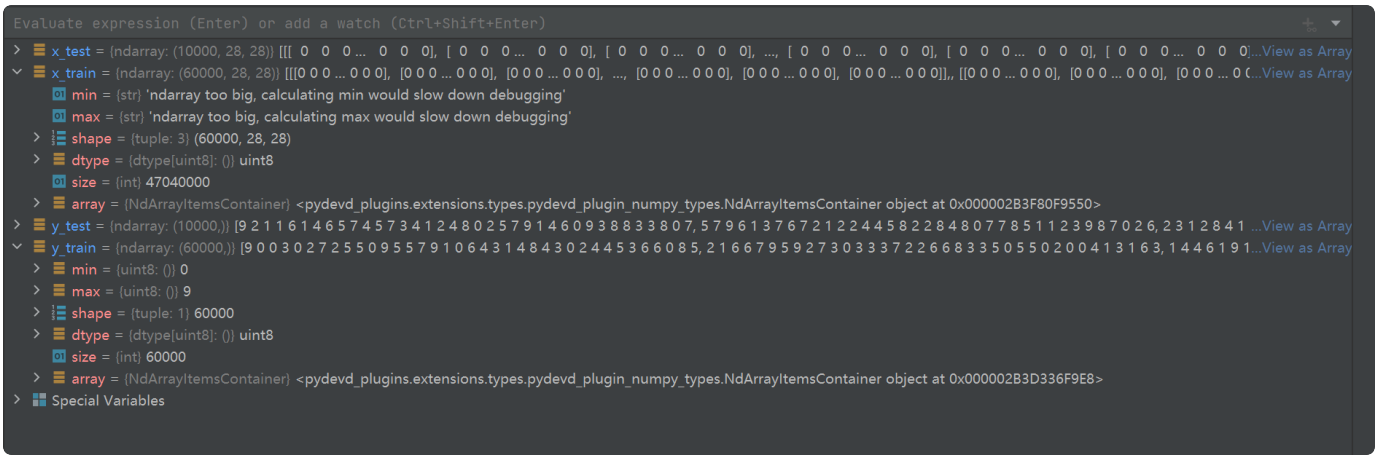
Fashion-MNIST is a dataset comprising of **28×28** grayscale images of **70,000** fashion products from **10** categories, with **7,000** images per category. The training set has **60,000** images and the test set has **10,000** images. Fashion-MNIST shares the same image size, data format and the structure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

```python
# 导入相应模块
import tensorflow as tf
from matplotlib import pyplot as plt

# 数据读取
fashion = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion.load_data()
```

数据内部存储



2022–7–15