機器學習入門

既然咱們學 Python，也肯定要講講機器學習。因為它的很多庫都是用 Python 寫的。開始先裝上它們玩一下。

Tensorflow

安裝一下。

```
$ pip install tensorflow
ERROR: Could not find a version that satisfies the requirement tensorflow
ERROR: No matching distribution found for tensorflow

$ type python
python is aliased to `/usr/local/Cellar/python@3.9/3.9.1_6/bin/python3'
```

然而 Tensorflow 2 只支持 Python 3.5–3.8。我們用的是 3.9 。

```
 % type python3
python3 is /usr/bin/python3
% python3 -V
Python 3.8.2
```

注意到我系統裡的 python3 是 3.8.2 版本。這個 Python 版本對應的 pip 安裝到哪兒呢。

```
% python3 -m pip -V
pip 21.0.1 from /Users/lzw/Library/Python/3.8/lib/python/site-packages/pip (python 3.8)
```

對應的 pip 在這裡。那我更改一下.zprofile 文件裡。最近我更改了我的 shell。.zprofile 就相當於之前的.bash_profile。加入一行。

```
alias pip3=/Users/lzw/Library/Python/3.8/bin/pip3
```

這樣，我們用 python3 和 pip3 來玩 Tensorflow。

```
% pip3 install tensorflow
...
Successfully installed absl-py-0.12.0 astunparse-1.6.3 cachetools-4.2.1 certifi-2020.12.5 chardet-4.0.0
```

安裝了很多庫。用上官網的一個例子。

```
import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])

predictions = model(x_train[:1]).numpy()
print(predictions)
```

運行一下。

```
$ /usr/bin/python3 tf.py
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 10s 1us/step
[[ 0.15477428 -0.3877643   0.0994779   0.07474922 -0.26219758 -0.03550266
   0.32226565 -0.37141111  0.10925996 -0.0115255 ]]
```

可見下載了數據集，接著輸出了結果。

接下來，看看圖片分類的例子。

```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

報錯。

```
ModuleNotFoundError: No module named 'matplotlib'
```

安裝一下。

```
% pip3 install matplotlib
```

正確了。

```
$ /usr/bin/python3 image.py
2.4.1
```

進行複製粘貼例子代碼。

```
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
print(train_images.shape)
print(len(train_labels))
```

輸出了結果。注意到這裡有 `train_images`、`train_labels`、`test_images`、`test_labels`。就是分為訓練數據集和測試數據集。

```
(60000, 28, 28)
60000
```

接著試試打印出圖片來。

```
print(train_images[0])
```

看下結果。

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   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   0   0  13  73   0
    0   1   4   0   0   0   0   1   1   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3   0  36 136 127  62
   54   0   0   0   1   3   4   0   0   3]
 [  0   0   0   0   0   0   0   0   0   0   0   0   6   0 102 204 176 134
  144 123  23   0   0   0   0  12  10   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 155 236 207 178
  107 156 161 109  64  23  77 130  72  15]
 [  0   0   0   0   0   0   0   0   0   0   0   1   0  69 207 223 218 216
  216 163 127 121 122 146 141  88 172  66]]
  ....
```

這裡節選了部分結果。

```
print(len(train_images[0][0]))
```

輸出 28。所以很清楚，這是一個橫寬為 28 的矩陣。繼續打印。

```
    print(len(train_images[0][0][0])
TypeError: object of type 'numpy.uint8' has no len()
```

所以很明白。每張圖片都是 28*28*3 的數組。最後一維數組保存的是 rgb 值。然而發現我們的想法可能是錯的。

```
print(train_images[0][1][20])
```

```
0
```

```
print(train_images[0][1])
```

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

說明每張圖片是 28*28 的數組。搗鼓了一陣。我們終於知道了秘密。

先來看看輸出的圖。

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```

看到右邊的顏色條嗎。0 到 250。原來這是在兩種顏色裡的漸變。可是它怎麼知道是哪兩種顏色。我們哪裡告訴它了。

接著我們把第二張圖也打印出來。

```
plt.imshow(train_images[1])
```

很有意思。這難道是 pyplot 依賴庫默認的嗎。繼續運行官網給的代碼。

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

注意到這裡顯示了圖片以及它們的分類。終於我們知道了 cmp 參數。如果 cmp 什麼都不寫，一定會是剛剛我們那種色彩的。果然。

```
    plt.imshow(train_images[i])
```

這會我們搜索 pyplot cmap。找到一些資料。

```
    plt.imshow(train_images[i], cmap=plt.cm.PiYG)
```

改一下代碼。

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(2,5,i+1)   ## 改這行
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.Blues)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

然而報錯了。

```
ValueError: num must be 1 <= num <= 10, not 11
```

這意味著什麼。之前的 `5,5,i+1` 到底什麼意思。為什麼改成 `2` 就不行了。儘管我們直觀地知道大概是 5 行 5 列的意思。但為什麼會報這個錯誤。`11` 是怎麼計算出來的。`num` 又是什麼意思。`10` 是什麼意思。注意到 `2*5=10`。所以也許當 `i=11` 的時候出錯了。當改成 `for i in range(10):` 時，得到了以下結果。

這會稍微看一下文檔，得知 `subplot(nrows, ncols, index, **kwargs)`。嗯，到此我們很明白了。

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    # plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.Blues)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

注意到 `0 25` 這種就叫 `xticks`。當我們放大縮小這個框的時候，會有不同的展示。

plot_scale

注意到放大縮小框，`xticks` 和 `xlabels` 會有不同的顯示。

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=10)

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

注意到了這裡定義 model 的方式，用到了類 `Sequential`。注意這些參數，28,28、128、relu、10。注意到需要 `compile` 和 `fit`。`fit` 是擬合的意思。注意到 28,28 就是圖形大小。

```
Epoch 1/10
1875/1875 [==============================] - 2s 928us/step - loss: 0.6331 - accuracy: 0.7769
Epoch 2/10
1875/1875 [==============================] - 2s 961us/step - loss: 0.3860 - accuracy: 0.8615
Epoch 3/10
1875/1875 [==============================] - 2s 930us/step - loss: 0.3395 - accuracy: 0.8755
Epoch 4/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3071 - accuracy: 0.8890
Epoch 5/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2964 - accuracy: 0.8927
Epoch 6/10
1875/1875 [==============================] - 2s 985us/step - loss: 0.2764 - accuracy: 0.8955
Epoch 7/10
1875/1875 [==============================] - 2s 961us/step - loss: 0.2653 - accuracy: 0.8996
Epoch 8/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2549 - accuracy: 0.9052
Epoch 9/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2416 - accuracy: 0.9090
```

```
Epoch 10/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2372 - accuracy: 0.9086
313/313 - 0s - loss: 0.3422 - accuracy: 0.8798

Test accuracy: 0.879800021648407
```

模型已經訓練出來了。來改下參數。

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(28, activation='relu'),    # 128 -> 28
    tf.keras.layers.Dense(10)
])
```

修改一下 `Dense` 的第一個參數。

```
Epoch 1/10
1875/1875 [==============================] - 2s 714us/step - loss: 6.9774 - accuracy: 0.3294
Epoch 2/10
1875/1875 [==============================] - 1s 715us/step - loss: 1.3038 - accuracy: 0.4831
Epoch 3/10
1875/1875 [==============================] - 1s 747us/step - loss: 1.0160 - accuracy: 0.6197
Epoch 4/10
1875/1875 [==============================] - 1s 800us/step - loss: 0.7963 - accuracy: 0.6939
Epoch 5/10
1875/1875 [==============================] - 2s 893us/step - loss: 0.7006 - accuracy: 0.7183
Epoch 6/10
1875/1875 [==============================] - 1s 747us/step - loss: 0.6675 - accuracy: 0.7299
Epoch 7/10
1875/1875 [==============================] - 1s 694us/step - loss: 0.6681 - accuracy: 0.7330
Epoch 8/10
1875/1875 [==============================] - 1s 702us/step - loss: 0.6675 - accuracy: 0.7356
Epoch 9/10
1875/1875 [==============================] - 1s 778us/step - loss: 0.6508 - accuracy: 0.7363
Epoch 10/10
1875/1875 [==============================] - 1s 732us/step - loss: 0.6532 - accuracy: 0.7350
313/313 - 0s - loss: 0.6816 - accuracy: 0.7230
```

```
Test accuracy: 0.7229999899864197
```

注意到 Test accuracy 前後發生了變化。Epoch 這樣的是 fit 函數輸出的日誌。注意到當是 128 時，accuracy 從 0.7769 變到 0.9086。而當是 28