

# 基于 RNN 模型的天气预测

## 1. 实验目的

学习 RNN 经典模型 GRU，实现对天气的预测

## 2. 实验环境

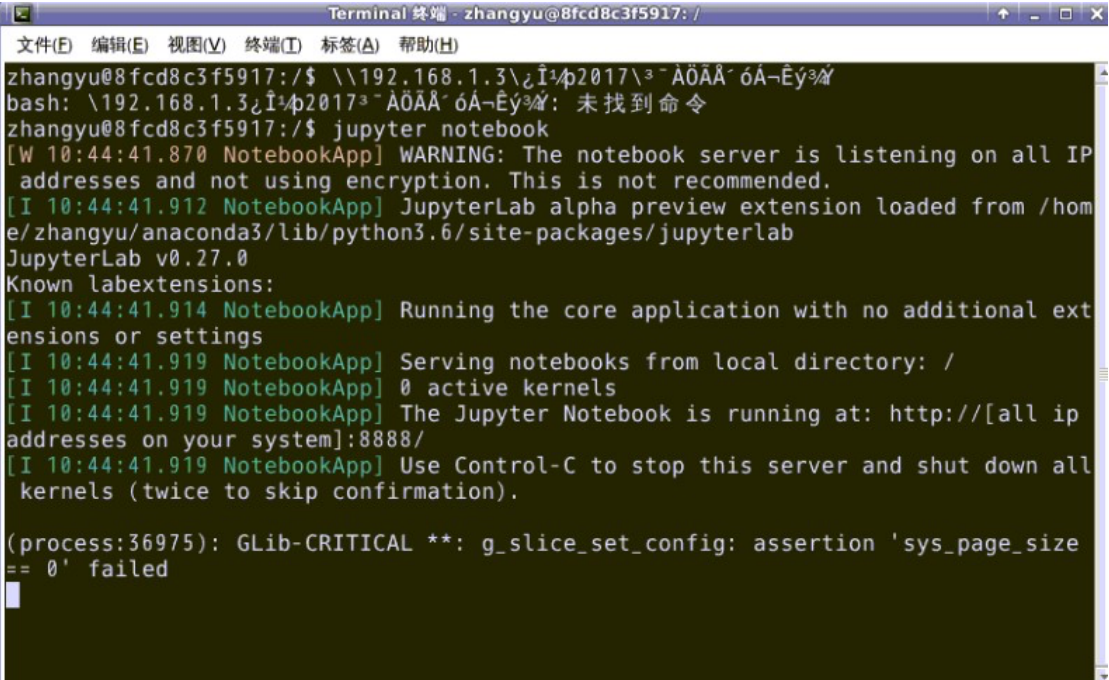
Linux Ubuntu 16.04

Python 3.6.1

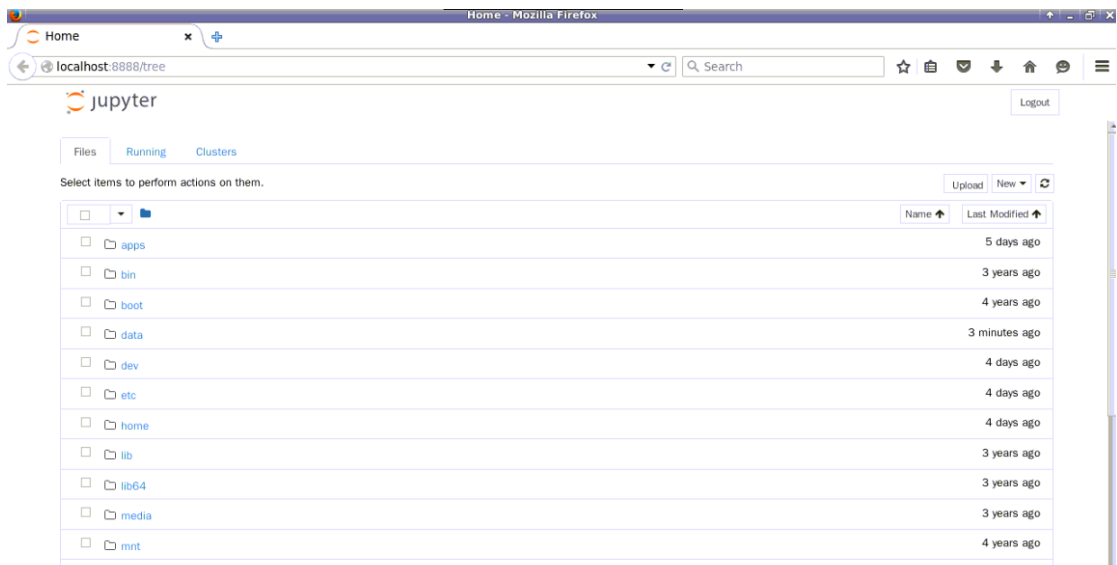
Jupyter

## 3. 实验步骤

1. 首先打开终端模拟器，输入下面命令：jupyter notebook -ip= '127.0.0.1'

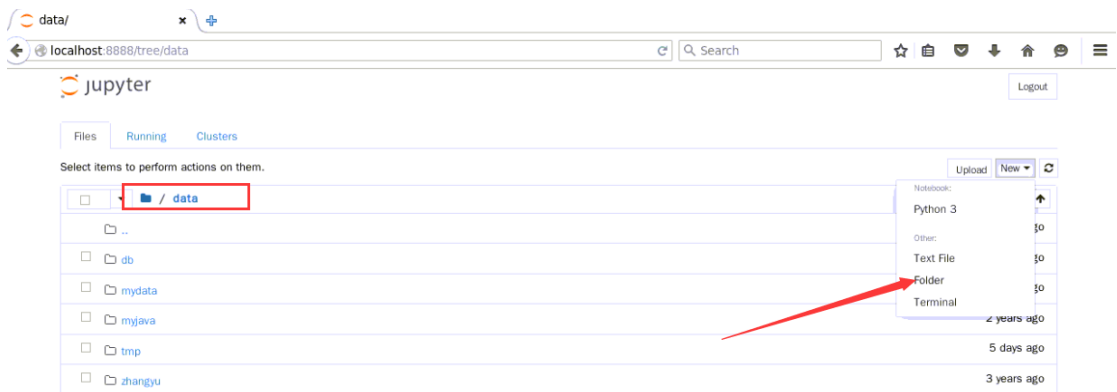
A terminal window titled 'Terminal 终端 - zhangyu@8fcd8c3f5917: /'. The prompt is 'zhangyu@8fcd8c3f5917:/\$'. The user enters '\192.168.1.3\p2017\3~ÄÖÅÄ~óÁ-Ëý¾'. The prompt changes to 'bash: \192.168.1.3\p2017\3~ÄÖÅÄ~óÁ-Ëý¾: 未找到命令'. The user then enters 'jupyter notebook'. The terminal shows several log messages from JupyterLab: '[W 10:44:41.870 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using encryption. This is not recommended.', '[I 10:44:41.912 NotebookApp] JupyterLab alpha preview extension loaded from /home/zhangyu/anaconda3/lib/python3.6/site-packages/jupyterlab', 'JupyterLab v0.27.0', 'Known labextensions:', '[I 10:44:41.914 NotebookApp] Running the core application with no additional extensions or settings', '[I 10:44:41.919 NotebookApp] Serving notebooks from local directory: /', '[I 10:44:41.919 NotebookApp] 0 active kernels', '[I 10:44:41.919 NotebookApp] The Jupyter Notebook is running at: http://[all ip addresses on your system]:8888/', and '[I 10:44:41.919 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).'. At the bottom, there is a message: '(process:36975): GLib-CRITICAL \*\*: g\_slice\_set\_config: assertion 'sys\_page\_size == 0' failed'.

如上图所示，该终端不要关闭，在浏览器中会打开下面界面，

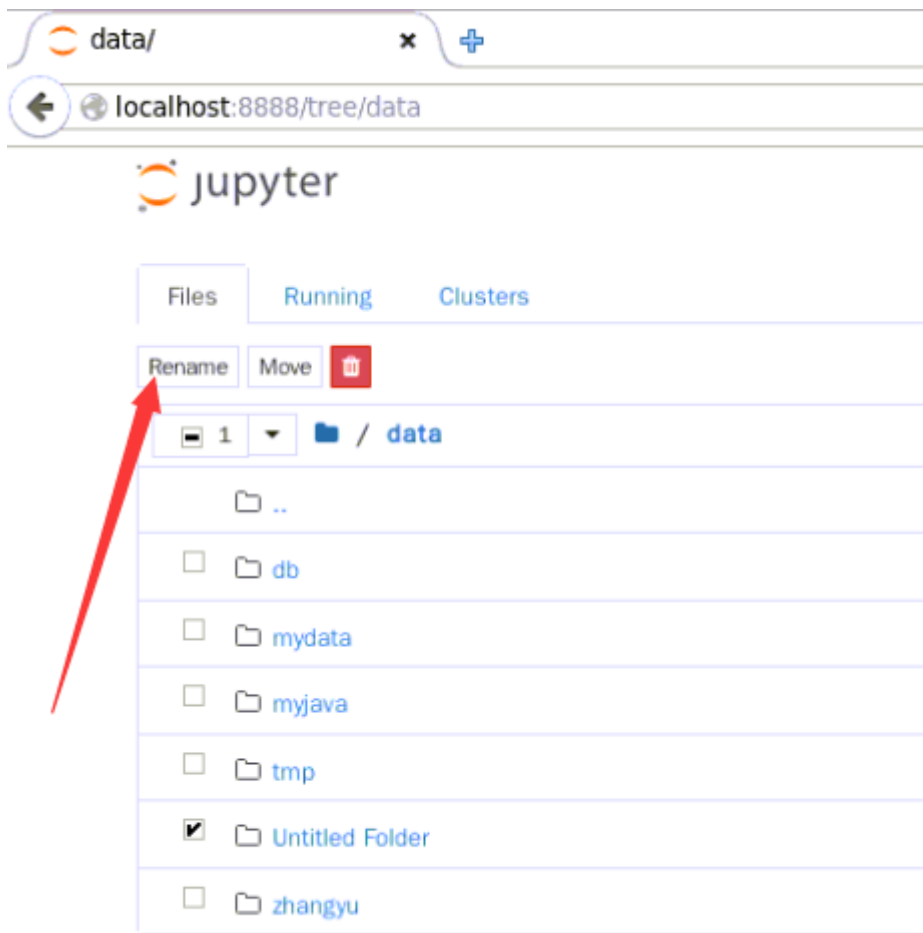


如果是第一次打开，浏览器界面会要求输入密码，密码为 zhangyu

2. 切换到/data 目录下，点击 New，在其下拉框中选择 folder

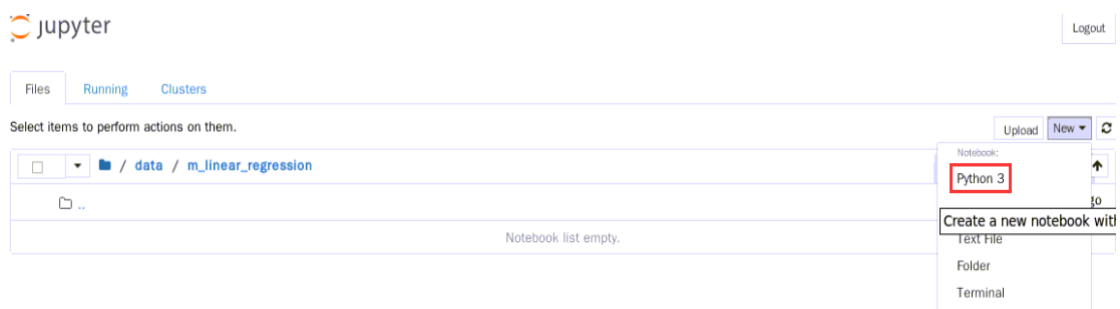


选中刚才创建的文件夹，点击页面左上角的【Rename】

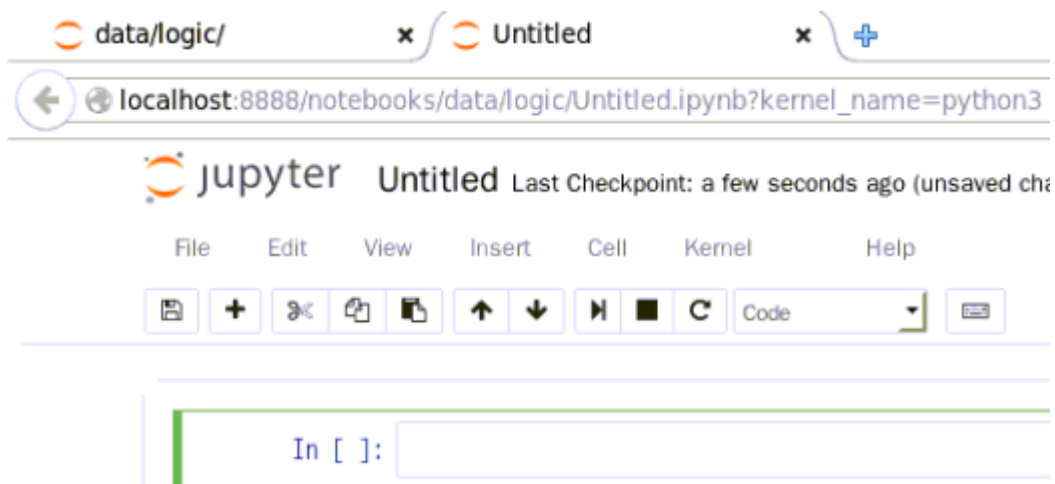


重命名为 logic（命名无要求）

3. 切换到 myapp 目录下，新建一个 logic 文件，用于编写并执行代码。点击页面右上角的 New，选中【Python3】



新建 ipynb 文件如下所示，在此可以编写代码了



## 4. python 包导入

载入各类程序需要的库和包（pycharm 和 jupyter 对一些包的版本要求可能不同）

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers, callbacks
from tensorflow.keras.optimizers import RMSprop
```

没有的包可以使用 `pip install` 命令安装。

## 5. 数据集处理

本程序使用的数据集为 jena\_climate\_2009\_2016 耶拿天气数据集，用于训练循环神经网络。在数据集中，十分钟一个记录，144 记录/天。

通过下载链接将数据下载到本地后，解压压缩包，将得到的 csv 放入指定位置（可自定义修改）。读取 csv，将原格式进行分割转换保存。

```
# 读取数据
# 数据下载链接 :https://s3.amazonaws.com/keras-datasets/jena_climate_2009_2016.csv.zip
f = open('./datasets/jena_climate_2009_2016.csv')
data = f.read()
f.close()

lines = data.split('\n')
header = lines[0].split(',')
lines = lines[1:]

print(header)
print(len(lines))
```

```
float_data = np.zeros((len(lines), len(header) - 1))
for i, line in enumerate(lines):
    float_data[i, :] = [float(x) for x in line.split(',')[1:]]
```

---

## 6. 检查数据

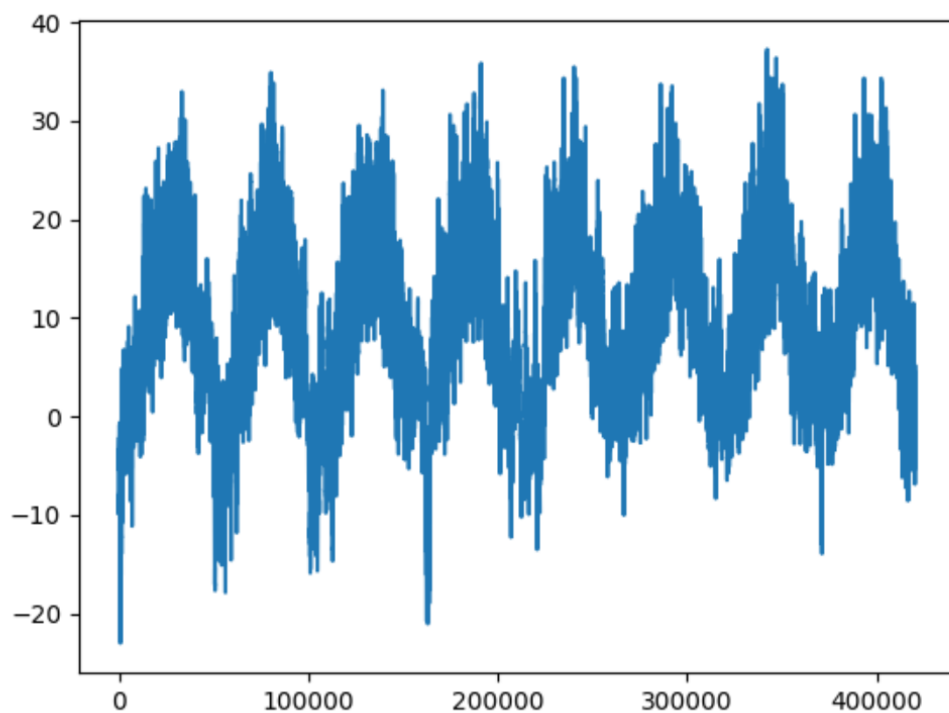
使用 plt 查看 2009-2016 年温度变化图与前十天的温度变化图。

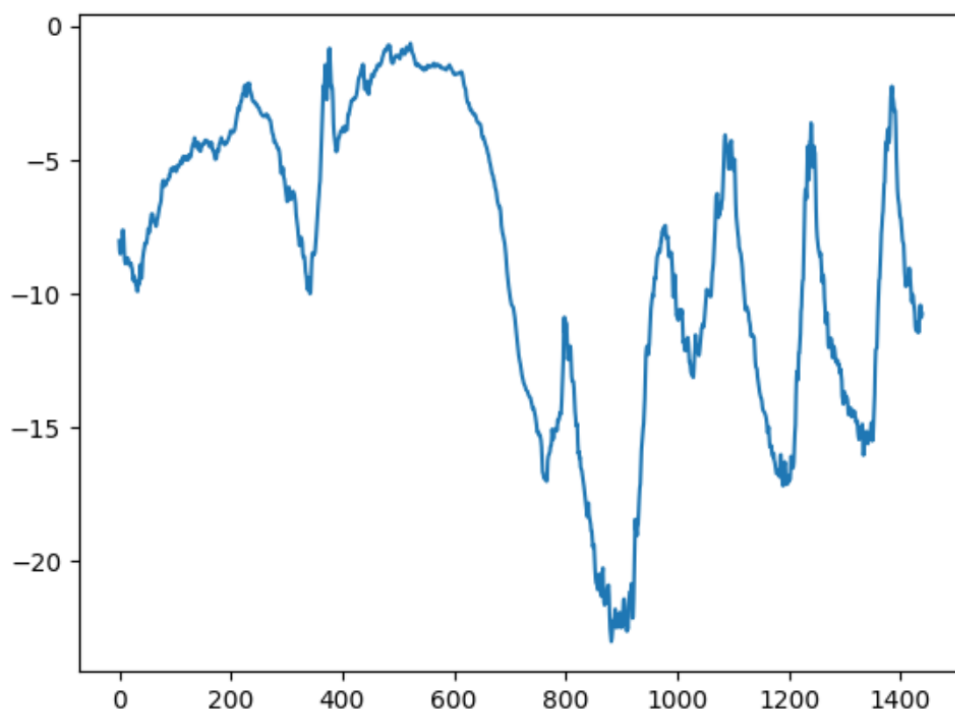
---

```
# 显示 2009-2016 温度变化图
temp = float_data[:, 1]
plt.plot(range(len(temp)), temp)
plt.show()

# 前 10 天温度变化图(十分钟一个记录,144 记录/天)
plt.plot(range(1440), temp[:1440])
plt.show()
```

---





## 7. 设置数据生成器

data:浮点数据的原始数组, 我们刚刚在上面的代码片段中将其规范化。

lookback:我们的输入数据应该返回多少个时间步骤。

delay:未来我们的目标应该是多少个时间步骤。

min\_index 和 max\_index:指数 data 数组, 该数组分隔要从哪个时间步骤绘制的时间步骤。

这对于保存一段数据以进行验证和另一段数据用于测试非常有用。

shuffle:是洗牌我们的样品, 还是按时间顺序提取样品。

batch\_size:每批样品的数量。

step:按时间步骤对数据进行抽样的时间。我们将设置为 6, 以便每小时绘制一个数据点。

假设现在是 1 点, 我们要预测 2 点时的气温, 由于当前数据记录的是每隔 10 分钟时的气象数据, 1 点到 2 点间隔 1 小时, 对应 6 个 10 分钟, 这个 6 对应的就是 delay。

要训练网络预测温度, 就需要将气象数据与温度建立起对应关系, 我们可以从 1 点开始倒推 10 天, 从过去 10 天的气象数据中做抽样后, 形成训练数据。由于气象数据是每 10 分钟记录一次, 因此倒推 10 天就是从当前时刻开始回溯 1440 条数据, 这个 1440 对应的就是 lookback。

我们无需把全部 1440 条数据作为训练数据, 而是从这些数据中抽样, 每隔 6 条取一条, 因此有  $1440/6=240$  条数据会作为训练数据, 这就是代码中的 `lookback//step`。于是就把 1 点前 10 天内的抽样数据作为训练数据, 2 点的是的气温作为数据对应的正确答案, 由此可以对网络进行训练。

---

```
def generator(data, lookback, delay, min_index, max_index,
              shuffle=False, batch_size=128, step=6):
    if max_index is None:
```

```

        max_index = len(data) - delay - 1
    i = min_index + lookback
    while 1:
        if shuffle:
            rows = np.random.randint(
                min_index + lookback, max_index, size=batch_size)
        else:
            if i + batch_size >= max_index:
                i = min_index + lookback
            rows = np.arange(i, min(i + batch_size, max_index))
            i += len(rows)
        samples = np.zeros((len(rows),
                            lookback // step,
                            data.shape[-1]))
        targets = np.zeros((len(rows),))
        for j, row in enumerate(rows):
            indices = range(rows[j] - lookback, rows[j], step)
            samples[j] = data[indices]
            targets[j] = data[rows[j] + delay][1]
        yield samples, targets

```

---

## 8. 进行数据生成与划分

调用数据生成器函数，进行训练集、验证集和测试集的数据生成和划分。

---

```

# ***** 调用数据生成器
# 选用前 200,000 个时间戳作为训练数据
# 减去均值除以标准差
mean = float_data[:200000].mean(axis=0)
float_data -= mean
std = float_data[:200000].std(axis=0)
float_data /= std

lookback = 1440 # 观察将追溯到 5 天前。
step = 6 # 观测将在每小时一个数据点取样。
delay = 144 # 目标是未来 24 小时。
batch_size = 128

train_gen = generator(float_data, lookback=lookback, delay=delay,
                      min_index=0, max_index=200000, shuffle=True,
                      step=step, batch_size=batch_size)
val_gen = generator(float_data, lookback=lookback, delay=delay,
                    min_index=200001, max_index=300000,
                    step=step, batch_size=batch_size)

```

```
test_gen = generator(float_data, lookback=lookback, delay=delay,
                    min_index=300001, max_index=None,
                    step=step, batch_size=batch_size)
```

```
# 要遍历整个验证集需要的 gen 调用次数
val_steps = (300000 - 200001 - lookback) // batch_size
test_steps = (len(float_data) - 300001 - lookback) // batch_size
```

---

## 9. 常识基线评估

总是预测明天某时的温度等于今天某时的温度的损失。

---

```
def evaluate_naive_method():
    batch_maes = []
    for step in range(val_steps):
        samples, targets = next(val_gen)
        preds = samples[:, -1, 1]
        mae = np.mean(np.abs(preds - targets))
        batch_maes.append(mae)
    print(np.mean(batch_maes))
```

```
evaluate_naive_method()
```

---

## 10. 建立模型

设计网络模型，包含两层 GRU 和最后的一层全连接层。优化器选用 RMSprop，损失函数使用 mae 损失函数。

---

```
model = Sequential([
    layers.GRU(32,
               dropout=0.1,
               recurrent_dropout=0.5,
               return_sequences=True,
               input_shape=(None, float_data.shape[-1])),
    layers.GRU(64, activation='relu',
               dropout=0.1,
               recurrent_dropout=0.5),
    layers.Dense(1)
])
```

```
model.compile(optimizer=RMSprop(), loss='mae')
```

---

## 11. 训练模型



设置回调函数以在训练中保存验证集损失最低的模型参数，并用 fit\_generator 方法开始训练。

---

```
callback = callbacks.ModelCheckpoint(filepath='./saved_models/weather_pred.h5',
                                     monitor='val_loss',
                                     save_weights_only=True,
                                     save_best_only=True,
                                     verbose=1,
                                     period=1)

history = model.fit_generator(train_gen,
                             steps_per_epoch=500,
                             epochs=20,
                             validation_data=val_gen,
                             validation_steps=val_steps,
                             callbacks=callback)
```

---

## 12. 显示训练曲线

使用绘图包绘制训练集损失和验证集损失训练曲线，。

---

```
# ***** 显示训练曲线
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(loss))

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

---

