【TechGym】ゼロからはじめる機械学習入門講座「AI x IoT(センサーデータ分析と画像分類・物体検出)」(テックジムオープン講座)
サンプルソースの公開場所 https://github.com/techgymjp/techgym_ai
実行環境がない場合は anaconda を install してください。(抜粋版なので問題番号は連番ではないです)

■13-1：センサーデータ分析：eP4q.py
【問題】
センサーデータを分析することで IoT からくるデータ分析の方法を実践してみよう
気象庁から温度湿度データをオープンデータとして得ることが出来る
そこで、気象庁のセンサーデータで以下のことをやってみよう

□takamatsu.csv を github からダウンロードしてデータフレームで表示する
　このとき以下の条件でファイルを読み込む
　・”日時”を”data_hour”として index に指定する
　・data_hour を datetime 型として読み込む
　・”x”のデータは NaN として読み込む
　・「時」の列は使わないので削除する
□列の名前を英語に変更する
　"降水量(mm)": "rain","気温(℃)": "temperature","湿度(%)": "humid"
□欠損値を 0 にする
□'2012/07/01 00:00:00'を起点としての経過時間を計算して、time という列を追加する
　まず、起点の時刻との差分として delta という変数つくる
　このとき time の列は 次の値に設定する: delta.days + delta.seconds / 3600.0 / 24.0
□グラフを表示する
　"降水量(mm)": "rain","気温(℃)": "temperature","湿度(%)": "humid"
■解答は sZ8c.py

■13-2：センサーデータ分析：q8Wh.py
【問題】
温湿度センサーデータの分析をする準備で、
電力消費量のオープンデータについて以下を実施してみよう
□shikoku_electricity_2012.csv を github からダウンロードしてデータフレームで表示する
　このとき以下の条件でファイルを読み込む
　・先頭 3 行の読み込みをスキップする
　・header として names=['DATE', 'TIME', 'consumption']を指定する
　・['DATE', 'TIME']を data_hour として datetime 型で読み込む
　・”data_hour”を index に指定する
□'2012/07/01 00:00:00'を起点としての経過時間を計算して、time という列を追加する
　まず、起点の時刻との差分として delta という変数つくる
　このとき time の列は 次の値に設定する: delta.days + delta.seconds / 3600.0 / 24.0
□電力消費量の時間変化の散布図を表示する(consumption と time)
□電力消費量のヒストグラムを表示する(分割数を 100 とする)
■解答は Re5V.py

■13-3：センサーデータ分析：Rf6G.py
【問題】
温湿度センサーデータから電力消費量を推定してみよう
□気象データと電力消費量データをいったん統合して時間軸を合わせたうえで、再度分割する
　(ヒント：elec_data.join(tmp[["temperature","sunhour","month","hour"]]).dropna().values)
□SVM を用いて気温, 日照時間, 月, 時間から電力消費量を推定する
□訓練スコアと検証スコアを計算する
□推定した電力消費量から統計値を計算して、電力消費量についての施策を自由に考えてみよう

■13-17：CNN(モデル構造)：e5Us.py
【問題】
Keras を使用して簡単にモデルを作成することができるが、
各層の重みや層の構造を調べることも出来る
手書き文字データのモデルで層の重みを調べてみよう

□モデル構造を表示しよう(summary()を使用する)
□モデルの重みを表示しよう(weights 変数を使用する) → これにより各層の行列の大きさがわかる
□各レイヤーの weight と bias を取得する(get_weights()を使用する)
□取得した weight と bias の行列の大きさを表示して、もう一度、各層の行列の大きさを確認する
□具体的な重みとバイアスを表示してみる
■解答は E4rW.py

■13-20：画像分類：J6gF.py
【問題】
画像分類の問題に取り組んでみよう
□以下の URL に分類分けされた 101 種類の写真があるのでファイルを Download する
http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_ObjectCategories.tar.gz
画像データを読み込んで処理しやすいように変形して一度、保存をする
(このとき、学習データと訓練データに分けておく)
■解答は h8H9.py

■13-21：画像分類：y7FU.py
【問題】
□前問で保存したデータをロードする
□学習用のデータを正規化する（float 型へ変換して 256 で除算する）
□以下の学習モデルを作成する

```
model = Sequential()
model.add(Convolution2D(32, 3, 3,
 border_mode='same',
 input_shape=X_train.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
```

□モデルを訓練する(batch_size=32, nb_epoch=50)、その後、モデルを評価する
■解答は V9j7.py

■13-23：移転学習(画像分類)：Ln7g.py
【問題】
大量のデータで画像分類をするときは学習のコストが大きくなる。そこで、すでに学習済みの重みを使って画像分類をすると効率的に処理が出来る。さらに、keras には学習済みのモデルがすでに用意されている。
また、用途に応じて学習済みの重みの一部を変更することも出来る。

□カテゴリーが記述された JSON ファイルを Download する
□分類対象となる画像を Download する
□配列の形状を(1, Height, Width, Channels)という形に変形する
□画像を予測して、予測した名前を表示する
どの程度一致していたかを推論して TOP5 に一致する名前を出力する
(分類名は英語になっているので、わかりやすいように日本語化している)

□自分の好きな画像を WEB から持ってきて何に分類されるかを試してみよう
　今回は MobileNetV2 というモデルを使用した
■解答は Ht2r.py

■13-24：移転学習(画像分類)：V5sd.py
VGG16 を用いて同じ画像を分類してみよう
　※VGG16 は 2014 年の ILSVRC（ImageNet Large Scale Visual Recognition Challenge）で提案された畳み込み 13 層とフル結合 3 層の計 16 層から成る畳み込みニューラルネットワーク
■解答は A4f8.py

■13-25：動画物体認識：B8df.py

　以降の問題で使用する仮想環境を構築する
□python3.8 系と必要なライブラリをインストールする

```
$ conda create -n yolov5 python=3.8
$ conda activate yolov5
$ git clone https://github.com/ultralytics/yolov5.git
$ cd yolov5
$ pip install -U -r requirements.txt
$ pip install pyqt5
$ pip install opencv-python-headless
```

【問題】
動画物体認識をするための準備をする
(Github にある動画を再生する)
再生する動画ファイル名は” Dogs10820.mp4”
□動画が再生できることを確認する
■解答は kJ9R.py

■13-26：画像物体認識：Yr6M.py
【問題】
YOLO(You Only Look Once)で物体認識をやってみよう。YOLO は画像認識アルゴリズムであり、リアルタイムかつ高精度に処理が可能なものである。
これまでのアルゴリズムでは境界設定と物体検出という二段階で行われていた処理を一度に行うことによって高速に処理出来るようにしたものである。

□以下を実行してインストールが完了しているかを確認する

```
import torch
from IPython.display import Image, clear_output  # to display images

clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__,
 torch.cuda.get_device_properties(0) if torch.cuda.is_available() else 'CPU'))
```

□写真に映し出されている物体が何であるかの認識を実行してみる
(実行結果は runs/detect/以下に格納されている)

```
$ cd yolov5
$ python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images/
```

■解答は mA37.py

■13-27：動画物体認識：Q4rA.py
【問題】
動画に映し出されている物体が何であるかの認識を実行してみる
動画ファイルは github にある以下の 3 つのファイルとする
(Dogs10820.mp4 , Horse_44738.mp4 , Summer_24541.mp4)

```
$python detect.py --source  ./Dogs10820.mp4
$python detect.py --source  ./Horse_44738.mp4
$python detect.py --source  ./Summer_24541.mp4
```

□認識した物体の名前と枠が上書きされた動画が生成されるので動画プレイヤーで再生してみる
(実行結果は runs/detect/以下に格納されている)

■解答は j6Gu.py (動画：Dogs10820_A.mp4 , Horse_44738_A.mp4 , Summer_24541_A.mp4)