人工智慧之應用

用人工智慧處理工具機之顫振效應

程式介紹:

import numpy as np

import matplotlib.pyplot as plt

from os import walk

from sklearn.preprocessing import MinMaxScaler

from sklearn.neural\_network import MLPClassifier

from sklearn.model\_selection import LeaveOneOut

from sklearn.model\_selection import cross\_val\_predict

from sklearn.metrics import confusion\_matrix

from numpy import genfromtxt

train\_input=[]

train\_input\_std=[]

train\_output=[]

folder\_name=['stable','unstable']

i = 0;

for folder in folder\_name:

path = 'Data/'+str(folder)+'/'

for root,dirs,files in walk(path):

for f in files:

filename = path + f

print(filename)

acc = genfromtxt(filename, delimiter=',')

acc = acc[:,1].tolist()

train\_input.append(acc[60000:80000])

train\_input\_std.append(np.std(acc))

if folder == 'unstable':

train\_output.append(1)

title = 'Original Signal With Chatter #'

saved\_file\_name = 'Fig/Original/unstable\_'

if folder == 'stable':

train\_output.append(0)

title = 'Original Signal Without Chatter #'

saved\_file\_name = 'Fig/Original/stable\_'

plt.figure(figsize=(7,4))

plt.plot(acc,'b-',lw=1)

plt.title(title + str(i+1))

plt.xlabel('Samples')

plt.ylabel('Acceleration')

plt.savefig(saved\_file\_name + str(i+1) + '.png')

plt.show()

i = i + 1

train\_input = np.array(train\_input\_std)

train\_output = np.array(train\_output)

scaler = MinMaxScaler(feature\_range=(0,1))

train\_input=scaler.fit\_transform(train\_input.reshape(-1,1))

loo = LeaveOneOut()

model = MLPClassifier(max\_iter=500, batch\_size=1, solver='adam')

y\_pred = cross\_val\_predict(model, train\_input, train\_output, cv=loo)

y\_true = train\_output

print('Prediction: \t', y\_pred)

print('Ground Truth: \t',y\_true)

cf\_m = confusion\_matrix(y\_true, y\_pred)

print('Confusion Matrix: \n', cf\_m)

tn, fp, fn, tp = cf\_m.ravel()

accuracy = (tn+tp)/(tn+fp+fn+tp)

print('Accuracy: ', accuracy)

實驗過程:

利用郭秉寰教授的影片中的程式，但因助教給的數據為csv檔，和影片中的mat檔不同，所以做了變化，用了 genfromtxt來拿出數據，其餘和郭教授的程式一樣，且助教的數據為x,y,z方向的，也有各用過，而x,y,z方向切換用 acc = acc[:,1].tolist()這一行的數字切換就行，x:0，y:1，z:2。

訓練模型:

train\_input = np.array(train\_input\_std)

train\_output = np.array(train\_output)

scaler = MinMaxScaler(feature\_range=(0,1))

train\_input=scaler.fit\_transform(train\_input.reshape(-1,1))

loo = LeaveOneOut()

model = MLPClassifier(max\_iter=500, batch\_size=1, solver='adam')

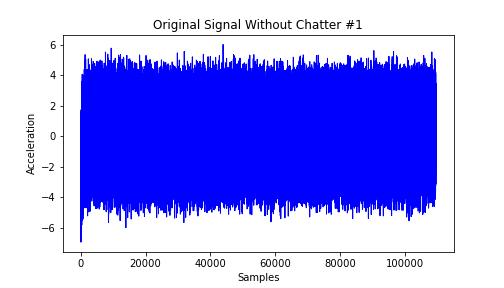
y\_pred = cross\_val\_predict(model, train\_input, train\_output, cv=loo)

y\_true = train\_output

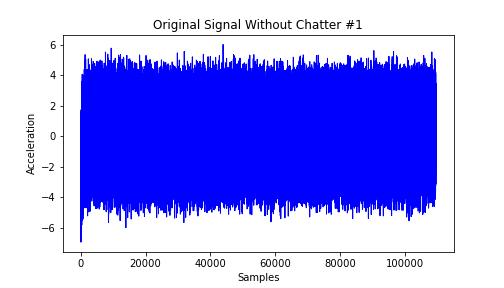
成果:

X方向:

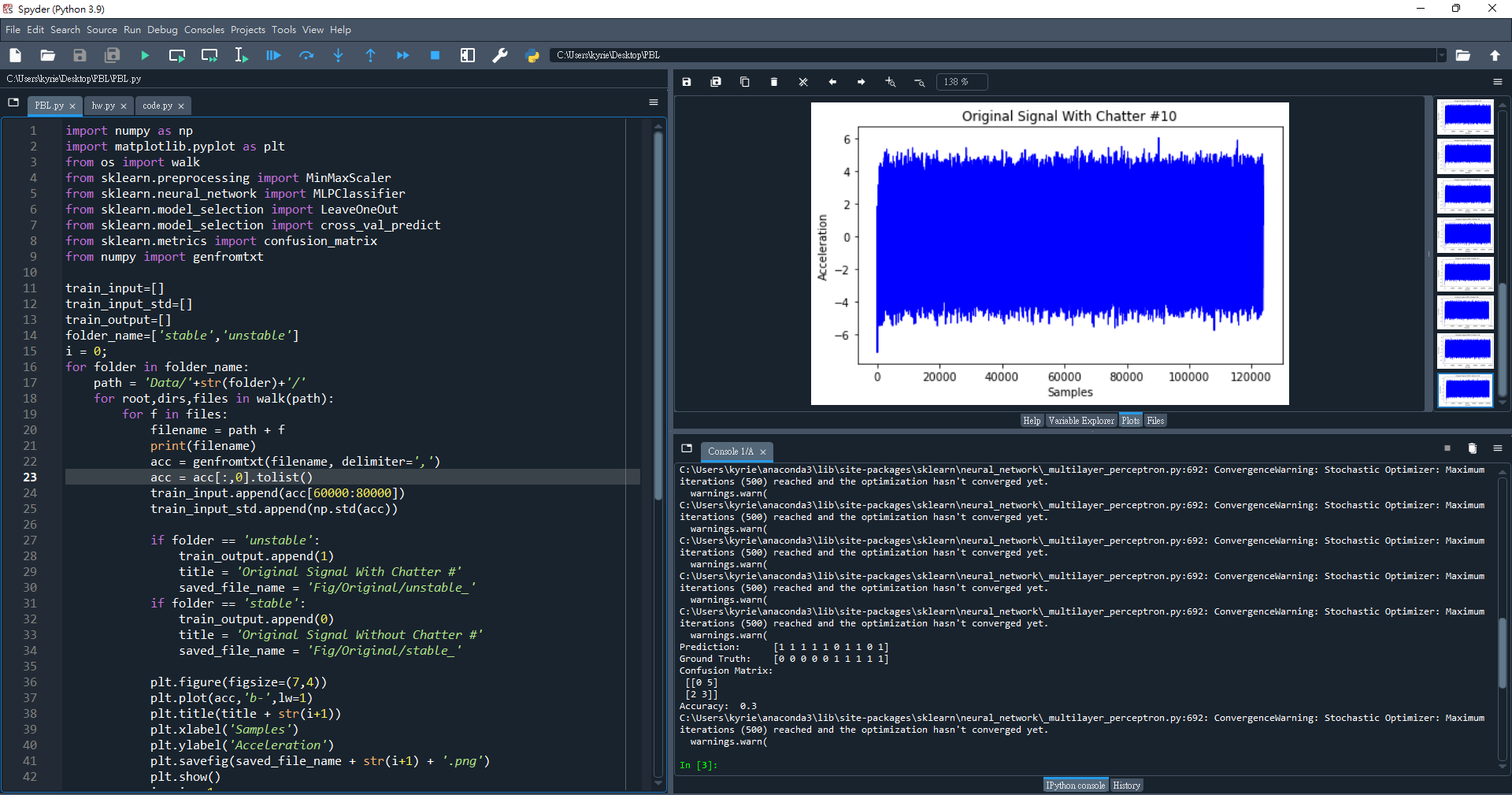
unstable的其中一張圖



stable的其中一張圖

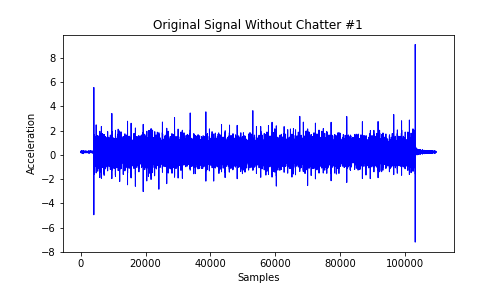


前面看得出兩張圖幾乎沒差，所以測試出的成果準確率也很低，以下是60000~80000的成果，這算準了，有些區間準確度甚至是0

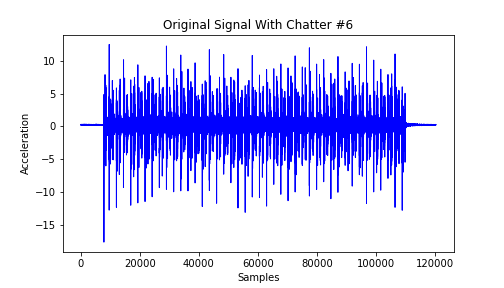


Y方向:

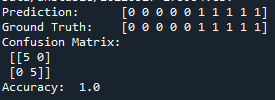
stable的其中一張圖



unstable的其中一張圖



前面看得出兩張圖的差距，而測出的數據也都準確度為1，相當成功



Z方向因和Y一樣準，所以就沒放上來

由實驗成果得知，X方向訊號做為有無顫振的判斷相當失敗，而YZ方向則準確度皆為1，用來判斷顫振依據是相當成功的，可以信任。