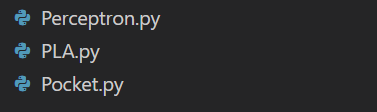
**HW2**

612415013 蕭宥羽

1. How to execute codes.



有Preceptron.py PLA.py Pocket.py三段程式碼

python PLA.py –-path filename -–save\_img True 實現PLA分類

python PLA.py –-path filename -–save\_img True 實現Pocket Algorithm 分類

其中 –-path filename 是我們所要使用的數據，-–save\_img True 決定是否將圖片存下

* PLA 執行過程

首先根據權重計算所有樣本的預測值，接著計算錯誤的點數，如果所有資料都正確就結束迴圈，如果有錯誤就更新權重

* Pocket Algorithm 執行過程

前面都跟PLA一樣，但會記錄下最少錯誤的權重值，如果下一輪的錯誤更少，則在更新最佳權重

1. Experimental results

* PLA iterations次數:

data\_30\_1.txt : 3

data\_30\_2.txt : 8

data\_30\_3.txt : 6

平均 : 5.66

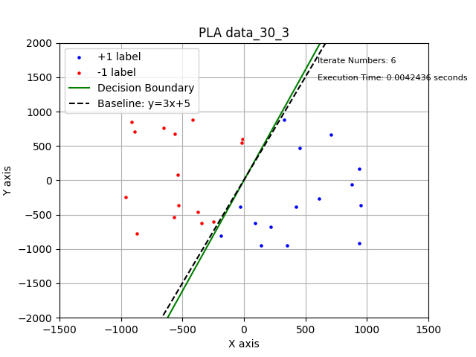
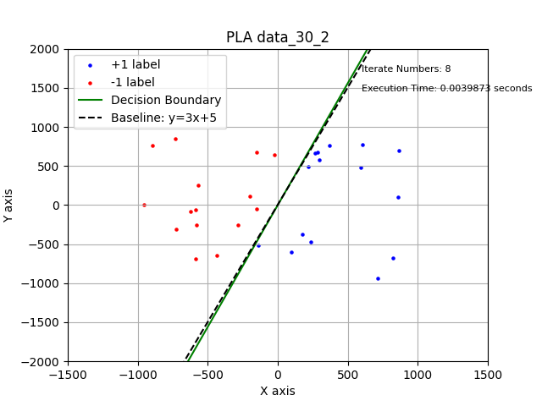
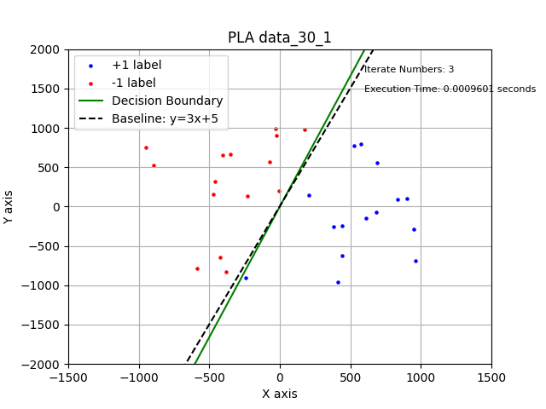


圖１

* data\_2000.txt

PLA : 0.1327s

Pocket Algorithm : 0.3054s

相差 : 0.2727s

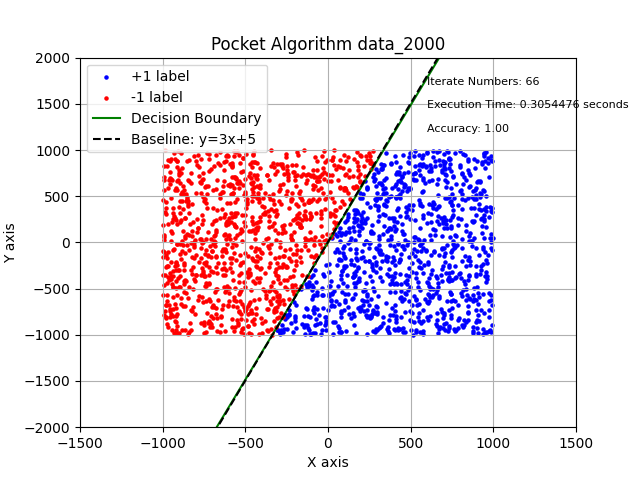
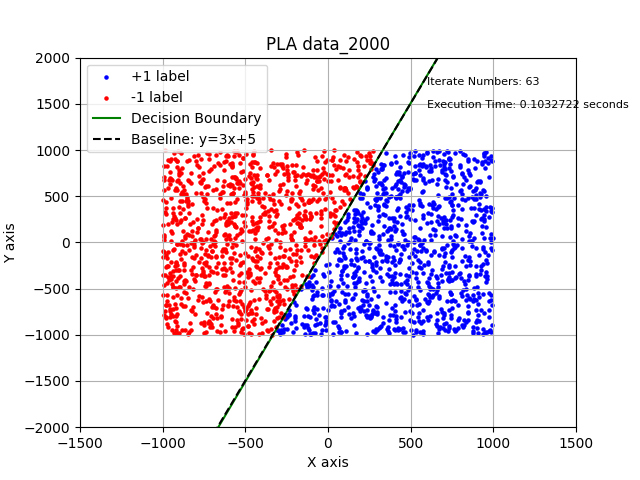


圖2

* + 正確率:0.95

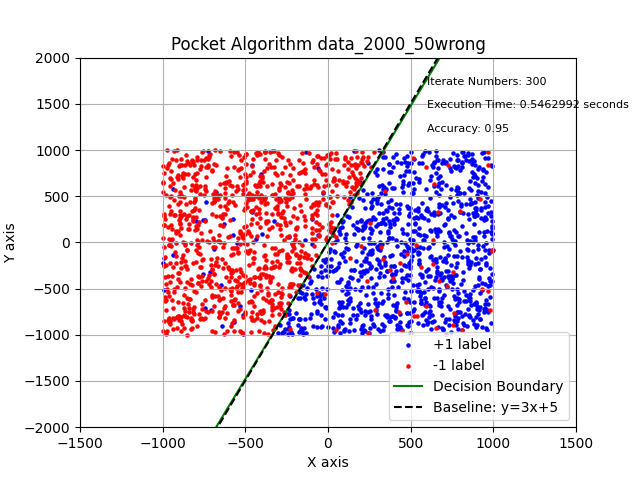


圖3

1. Conclusion
2. 根據圖一，我們用PLA對30筆正負樣本的資料做分類，因為所使用的資料是線性可分割的，所以使用PLA可以成功地將其分為兩類，平均迭代5.66次就可以找出正確的分割線。

根據觀察算出的分割線不一定會是 y=3x+5 ，因為當PLA計算到錯誤點的數量為0時就會結束權重的更新

1. 根據圖二，我們分別使用PLA跟Pocket Algorithm對2000筆正負樣本的資料做分類，因為這2000筆資料也是線性可分割的，所以用這兩種方法也可以將其分為兩類

根據觀察我們可以看出在差不多的迭代次數中PLA的執行時間是小於Pocket Algorithm的，因為Pocket Algorithm需要在每次迭代中檢查wt+1是否比w更好，所以Pocket Algorithm會比PLA還要慢

1. 圖三中總共有2000筆資料但是各有50筆資料是正負樣本不正確的(錯邊)，這是一種線性不可分割的情況，所以我們會使用Pocket Algorithm來實現分類，如果使用PLA算法則會發散

根據觀察Pocket Algorithm找到最佳的分類方式，分類的正確率為95%，跟實際的情況是一樣的

1. Discussion

在剛開始做這個作業的時候，對PLA算法並不是非常了解，所以花了一些時間去了解其中的算法，一開始我是直接用for迴圈去遍歷整個資料算出錯誤的點並記錄下來，但我後來發現我可以直接使用numpy的方式去做計算，這樣會比for迴圈的方式有效率許多。

雖然說在線性可分割的例子中PLA是會比Pocket Algorithm還要快，但是由於我們是隨機抓取錯誤點，所以有時候Pocket Algorithm所迭代的次數會比PLA 還要少，所以反而Pocket Algorithm的執行時間還比較少。