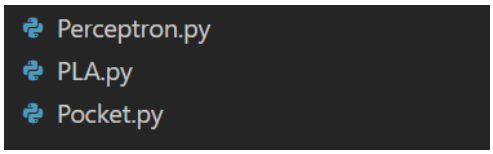
HW2

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

2. 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

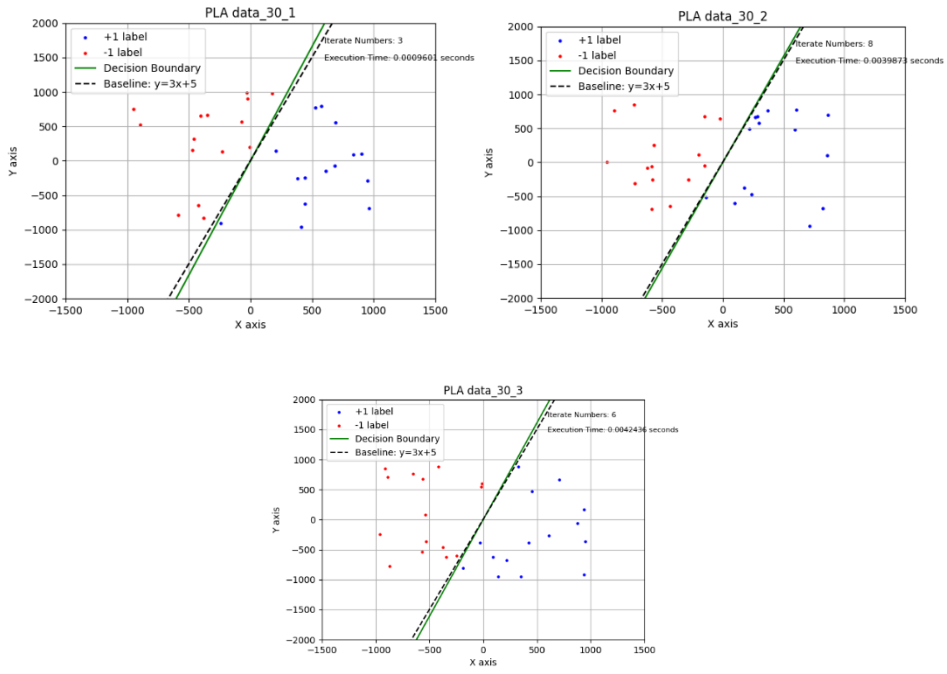


圖 1

- data_2000.txt

  PLA : 0.1327s

  Pocket Algorithm : 0.3054s

  相差 : 0.2727s

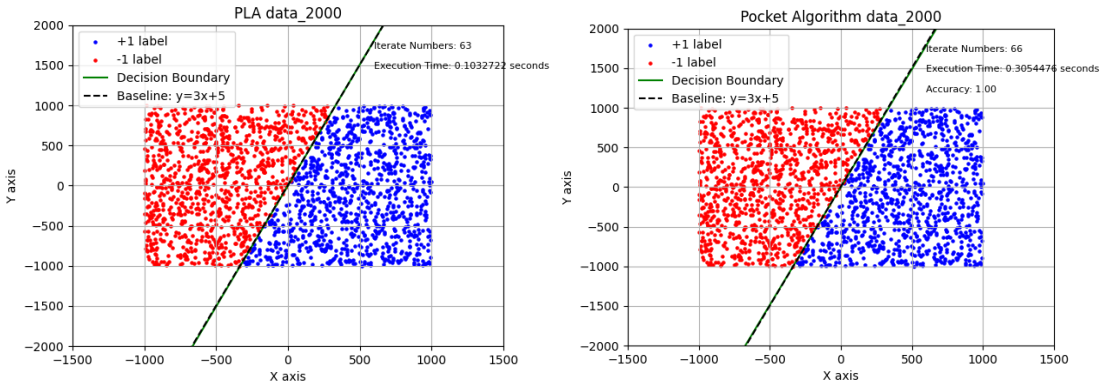


圖 2

- 正確率:0.95

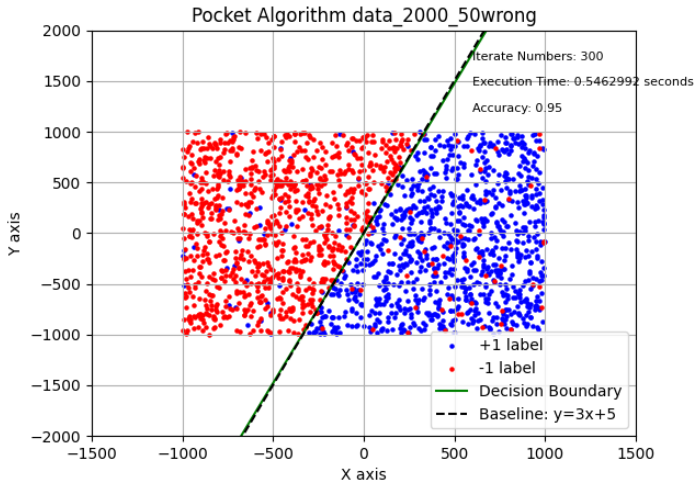


圖 3

3. Conclusion

I. 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

II. 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

III. 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

4. 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。