**機器學習< Assignment #1 – Perceptron>**

**姓名: 翁佳煌　　　學號: 409430030**

# 要求1:

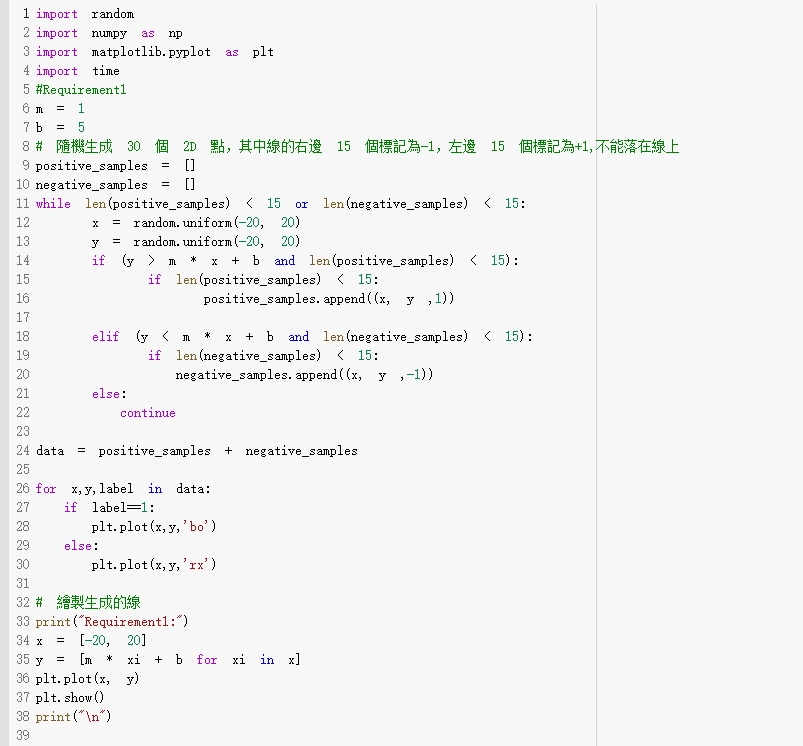
下圖1.1為要求1的code。

1.設定一條線的斜率m和截距b，用來區分正負標籤。

2.隨機生成30個二維的數據點，每個數據點的x和y坐標都是在-20到20之間的隨機值。

3.判斷每個數據點是否在線的上方或下方，如果在線的上方且正標籤的數量還不足15個，則將其標籤設為1，並添加到正標籤的列表中；如果在線的下方且負標籤的數量還不足15個，則將其標籤設為-1，並添加到負標籤的列表中。

4.最後將正負標籤的數據合併成一個數據集，並繪製出來。其中，正標籤的數據用藍色圓圈表示，負標籤的數據用紅色叉號表示，並且製出了原來的線圖。

****

▲圖1.1

## 下圖1.2為要求1的執行結果:

▲圖1.2

## 要求1問題討論:

在要求1上並無遇到太大題目上的問題，但由於我之前對Python不太熟悉，所以困難的點在於需要另外花時間搞懂其random和matplotlib的用法。

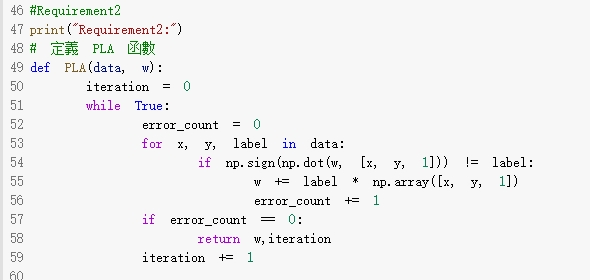
# 要求2:

下圖2.1是實現PLA(Perceptron Learning Algorithm)的訓練過程。

參數data是包含訓練資料的清單，其中每個元素都是一個三元組 (x, y, label)，代表一個二維點 (x, y) 和它的類別標籤 label。

w 是一個包含三個權重值的 NumPy 陣列，初始化為全零當作初始點。

1. 逐個檢查訓練資料中的每個點，計算其預測類別，若預測結果錯誤，則更新權重向量 w，並將錯誤計數器 error\_count 加一。
2. 如果所有訓練點都被正確分類，那麼 error\_count 保持為零，表示現有的權重向量 w 已經可以完美地分類所有資料，此時函式返回 w 和迭代次數 iteration。
3. 否則，繼續執行 PLA，直到所有資料都被正確分類為止。在這個過程中，迭代次數 iteration 會隨著每一次權重向量的更新而增加。



▲圖2.1

再來，下圖2.2是執行三個實驗。每個實驗產生一組隨機的資料，其中都各有15個正樣本和15個負樣本。接著使用 PLA 演算法訓練一個分類器，直到演算法收斂為止，並紀錄所需的迭代次數。最後，將產生的三個模型顯示在三個子圖中，並顯示每次實驗收斂時所需的迭代次數。最後，顯示三次實驗的平均迭代次數。

1. total\_iteration 設為0，以便計算三個實驗的平均迭代次數。

2. 在 for 迴圈中，進行三個實驗。在每次實驗中，開始產生隨機資料。

3. 初始化 positive\_samples 和 negative\_samples 空列表，並使用 while 迴圈生成正樣本和負樣本。並使用要求1的方法，如果正樣本數少於15個或負樣本數少於15個，則產生一個新的隨機資料點 (x, y)，並將其添加到正樣本或負樣本中，具體取決於它是否在隨機產生的直線上方或下方。

4. 然後，將正樣本和負樣本合併為一個列表，以供 PLA 演算法使用。初始化權重向量 w0 為 0.0、0.0、0.0。

5. 執行 PLA 演算法，直到演算法收斂為止。對於每個資料點 (x, y, label)，如果 PLA 的預測值不等於 label，則更新權重向量 w。

6. 如果錯誤次數等於 0，則表示 PLA 收斂，並將所需的迭代次數返回，以及更新的權重向量 w。如果錯誤次數不等於 0，則繼續迭代。

7. 計算每個實驗所需的迭代次數，並將其加到總迭代次數中，以便計算平均迭代次數。

8. 繪製每個實驗的結果，其中包括點和所學習的直線。

9. 顯示三次實驗的平均迭代次數。

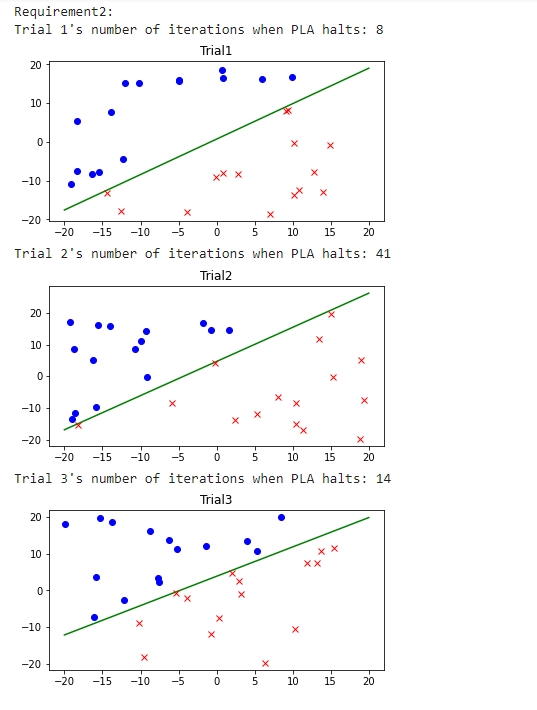


▲圖2.2

## 下圖2.3為要求2的結果:

三次試驗下都有收斂，在試驗1迭代了8次，試驗2為41次，試驗3則是14次。

總平均為21次。





▲圖2.3

## 要求2問題討論:

在一開始困難的點為實現PLA演算法的時候還不太清楚運作方法，自己查了許多文章才了解其設計方式。此外，我發現每次收斂的迭代次數都不同，我認為收斂的迭代次數會因為每次隨機生成的樣本不同而有差異。每次生成的樣本是隨機取樣，因此每次取樣得到的樣本可能會有所不同，導致PLA算法的收斂速度也有所不同。此外，初始權重向量也可能對收斂次數產生影響。在這個例子中，初始權重向量都是[0, 0, 0]，因此可能會出現一些情況下PLA算法的收斂速度較慢。

# 要求3:

下圖3.1為利用**要求1**的方法是隨機生成1000個正樣本和1000個負樣本，並將這些樣本畫在座標平面上，正樣本用藍色圓點表示，負樣本用紅色叉號表示。程式碼首先建立了空的正樣本列表和負樣本列表，然後進行while循環，直到正樣本和負樣本列表都至少包含1000個樣本。在每次迭代中，程式碼使用random.uniform()函數生成一對隨機坐標x和y，然後判斷它們是否位於一條直線上，如果是，就將它們添加到正樣本列表或負樣本列表中。如果正樣本和負樣本列表都不足1000個樣本，則繼續迭代。最後，程式碼將所有樣本畫在座標平面上，並使用藍色圓點和紅色叉號表示正樣本和負樣本。



▲圖3.1

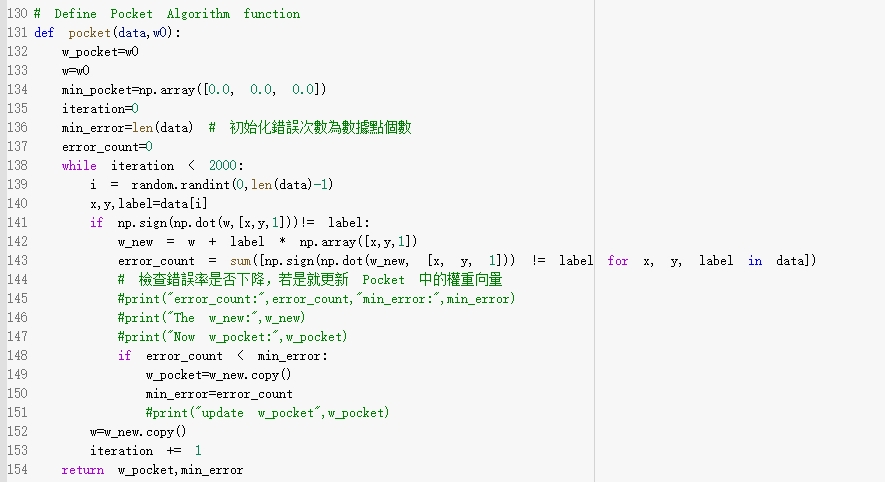
下圖3.2為實作Pocket Algorithm ，其輸入參數為一組數據集和一個權重向量 w0，輸出結果為在訓練過程中錯誤率最小的權重向量和對應的錯誤次數。

1. 首先初始化 Pocket Algorithm 中的權重向量 w\_pocket 和一個用於計算最小錯誤率的變量 min\_error，以及錯誤次數 error\_count，迭代次數 iteration。

2. 在每一次迭代中，從數據集 data 中隨機選取一個樣本 (x, y, label)，計算其在當前權重向量 w 下的預測結果，如果與 label 不一致，則更新權重向量 w\_new = w + label \* [x, y, 1]，並計算更新後的錯誤次數 error\_count。

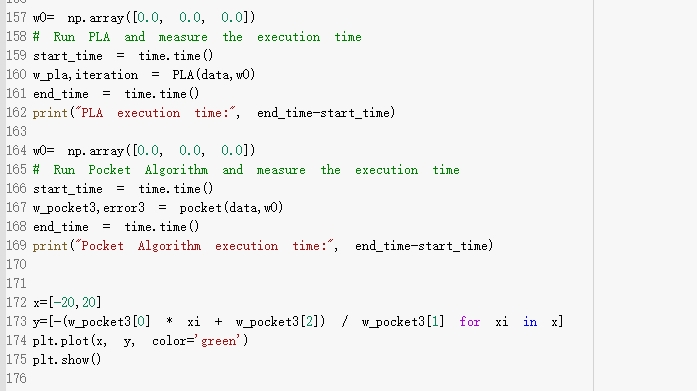
3. 然後檢查錯誤次數 error\_count 是否小於最小錯誤次數 min\_error，如果是，就更新 Pocket 中的權重向量 w\_pocket，以及最小錯誤次數 min\_error。

4. 最後更新迭代次數 iteration，如果迭代次數超過了我自己設定的最大值 2000，則停止算法並返回最小錯誤率對應的權重向量 w\_pocket。



▲圖3.2

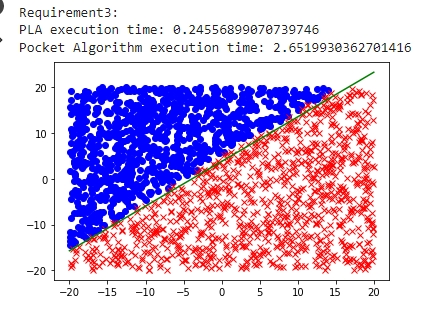
下圖3.3，定義了一個起始權重向量 w0，並使用 PLA 和 Pocket Algorithm 兩個演算法來分別訓練線性分類器，使用同一組數據集 data。接著，按照**要求3**計算並印出 PLA 和 Pocket Algorithm 的執行時間，並把pocket的結果畫出。



▲圖3.3

## 下圖3.4.為要求3的結果:

按照**要求3**，從下圖3.4可發現，Pocket演算法的執行時間比PLA長，這是因為Pocket演算法在每次更新權重時都要先計算錯誤率，而且要與目前Pocket中最小錯誤率進行比較，這樣才能確定是否要更新Pocket中的權重向量。這些額外的計算使得Pocket比PLA更加複雜，因此執行時間更長。



▲圖3.4

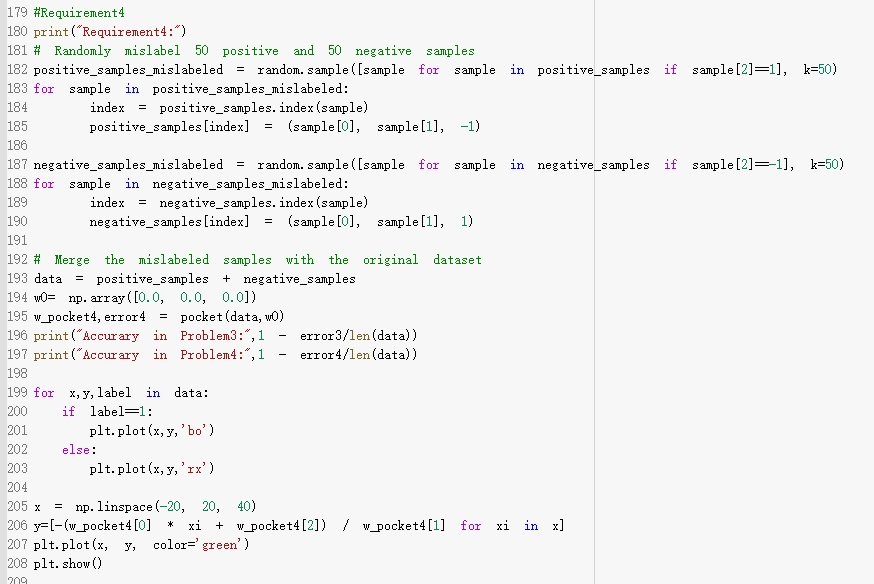
## 要求3問題討論:

在這遇到最大的問題就是實作Pocket演算法，雖然與PLA非常相似，但還是需要額外花時間搞懂，其關鍵核心為PLA目的在找出一個絕對好的分類器，但Pocket則是找出相對好的即可。另外，在實作時，我一直在猶豫最大迭代次數應該要設多少才合適，這問題解法就是需要根據數據的大小等因素來慢慢調整和實驗。

# 要求4:

下圖4.1的程式碼為對數據集進行了一些修改以創建一些錯誤的標籤。我們隨機選擇了50個原來是正樣本的樣本，將它們標記為負樣本，同時隨機選擇了50個原來是負樣本的樣本，將它們標記為正樣本，這樣我們就創建了一些具有錯誤標籤的樣本。

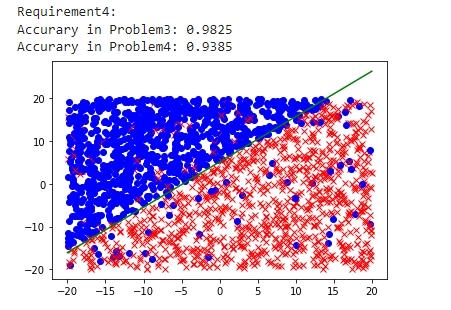
然後，我們將這些樣本與原始數據集合併，並使用**要求3**中定義的 POCKET 算法訓練模型。最後，我們印出在需求3和需求4中的準確率，並繪製出分類平面，藍色圓圈代表正樣本，紅色叉代表負樣本，綠色線條代表 POCKET 算法找到的分類平面。



▲圖4.1

## 下圖4.2為要求4的結果:

從下圖4.2的結果圖可知，在Problem3中的Pocket Algorithm 的情況下，最終權重向量的準確度為 0.9825，而在Problem4由於把50 個正樣本和 50 個負樣本標記錯誤，因此可知準確度比Problem3來的小。



▲圖4.2

## 要求4問題討論:

實作問題4沒什麼大問題，只要各挑50個點標記錯誤就好，隨機選擇50個正樣本和50個負樣本標記錯誤可能導致一些偏差。在實際應用中，標記錯誤的樣本可能分佈在不同的區域，並且可能與原始數據分佈的特徵不同。另外我查到許多不同對於錯誤樣本處理的機器學習演算法，例如Boosting和Random Forest等，該選擇使用哪種演算法，需要考慮多個因素，包括資料的特性、資料的大小、計算資源的限制、以及對準確度和解釋性的需求等等。