ML Assignment1

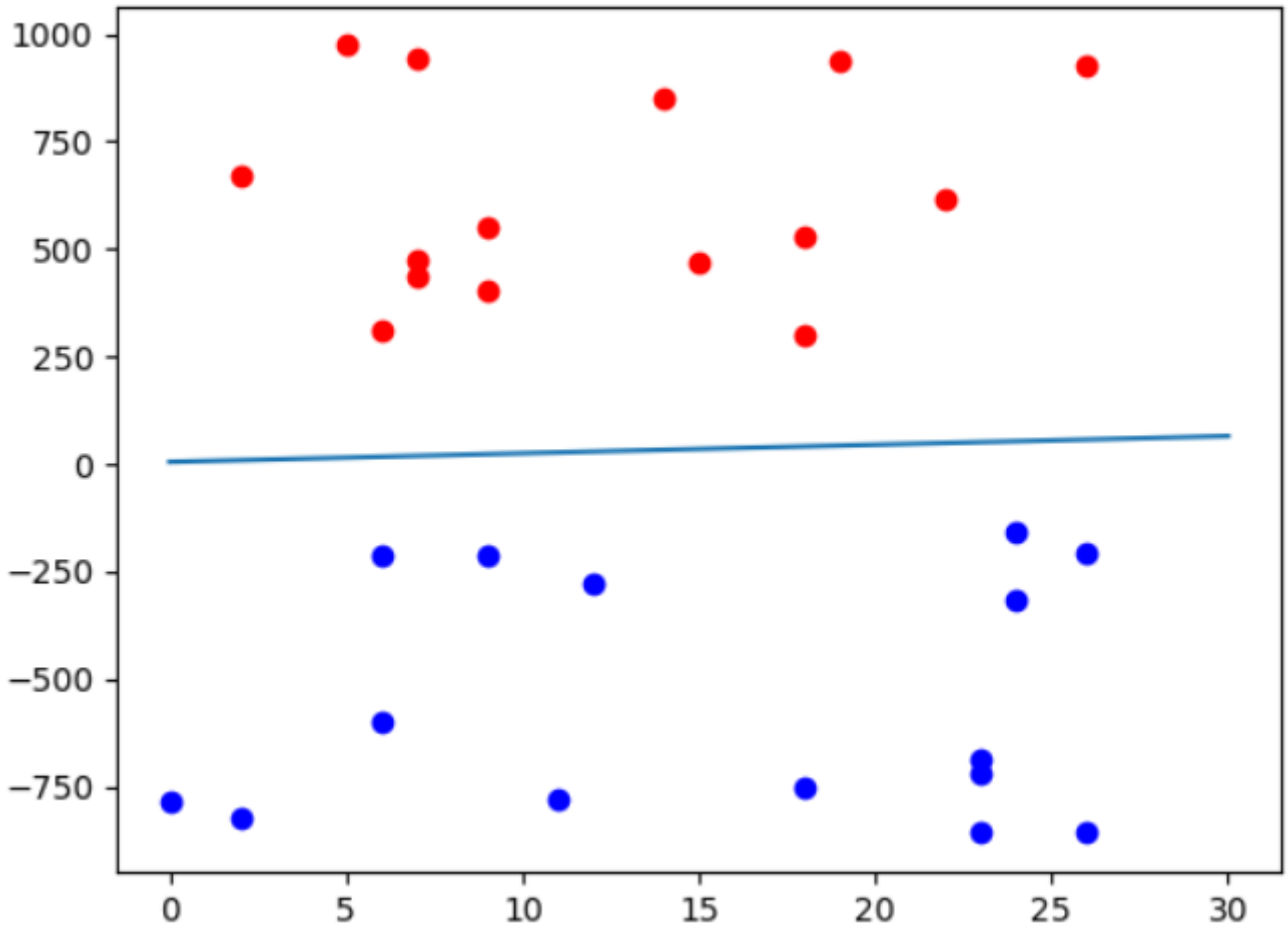
4084100094 資工三 葉丞勛

1.實作方法

(1)第一小題: 執行problem1.py後，即可產生第一小題的要求，並將結果存在problem1.png。首先先決定直線y=mx+b的m,b係數，以及欲產生的點個數，接下來將以上的變數丟進rand_samples()這個函數中，此函數會分別回傳位於直線左邊的點座標以及右邊的點座標，然後將回傳的座標點透過plot畫出來，最後再將畫出來的圖存在problem1.png。在rand_samples()中，會用兩個變數紀錄目前建立的點在線的左邊或右邊，然後用for去產生點座標，直到兩邊的點個數相等以及兩個變數相等於欲產生的點個數，最後回傳直線左邊的點座標以及右邊的點座標。 (2)第二小題: 執行problem2.py後，即可產生第二小題的要求，並將結果存在problem2.png。首先先決定直線y=mx+b的m,b係數，以及欲產生的點個數，接下來將以上的變數丟進rand_samples()這個函數中，此函數會回傳一個dataset，型態為[((1, x01, x02),y0),((1, x11, x12),y1),((1, x21, x22),y2),((1, x31, x32),y3)...]，然後將此dataset丟入pla()，對此dataset做PLA。在pla()中，我先設定w=[0,0,0]，然後把w跟x做內積來決定是否正確分類此點，若是遇到分類錯誤的的情況就將w更新成w+yx，更新完後再用新的w從頭開始對每個點進行分類，直到將每個點都分類正確為止，最後即可求得目標w。接著再將預測的直線及每個點畫出來，將結果存在problem2.png。由於我是從(1)的程式產生dataset，因此其一定是可以線性分割的資料，PLA最後一定會停下來。 (3)第三小題: 執行problem3.py後，即可產生第三小題的要求，並將結果存在problem3.png。此題跟第二小題不同的點在於需要實作一個pocket演算法，其他步驟跟他(2)類似。首先一樣決定y=mx+b的係數，並將產生的點設為2000，再透過rand_samples()產生dataset，並將此dataset分別丟進pla()以及pocket()，然後透過time.time()來計時。在pocket()中，首先將w設為[0,0,0]，然後設定iteration的最大次數為1000，接下來隨機從dataset選取一個點，並將w_t設定成w+yx，並分別計算出w及w_t發生錯誤的次數。若是w_t犯的錯誤比w還要少的話，就將w更新成w_t，若是發現w_t發生錯誤的個數為0的話就代表提前找到目標w，可以直接回傳更新完的w，然後不斷重複上述步驟直到w沒有分類錯誤或是迭代達到1000次，最後的結果就是所求的目標w。後來我還有將pocket()得出的結果畫成圖表並存在problem3.png。 (4)第四小題: 執行problem4.py後，即可產生第四小題的要求，並將結果存在problem4.png。跟其他題的差異主要是我在rand_samples()中產生dataset的過程中，故意標錯50個左邊的點以及50個右邊的點，其餘的做法都跟第三小題類似。最後我將把得出的結果存成一張圖表，存在problem4.png中。

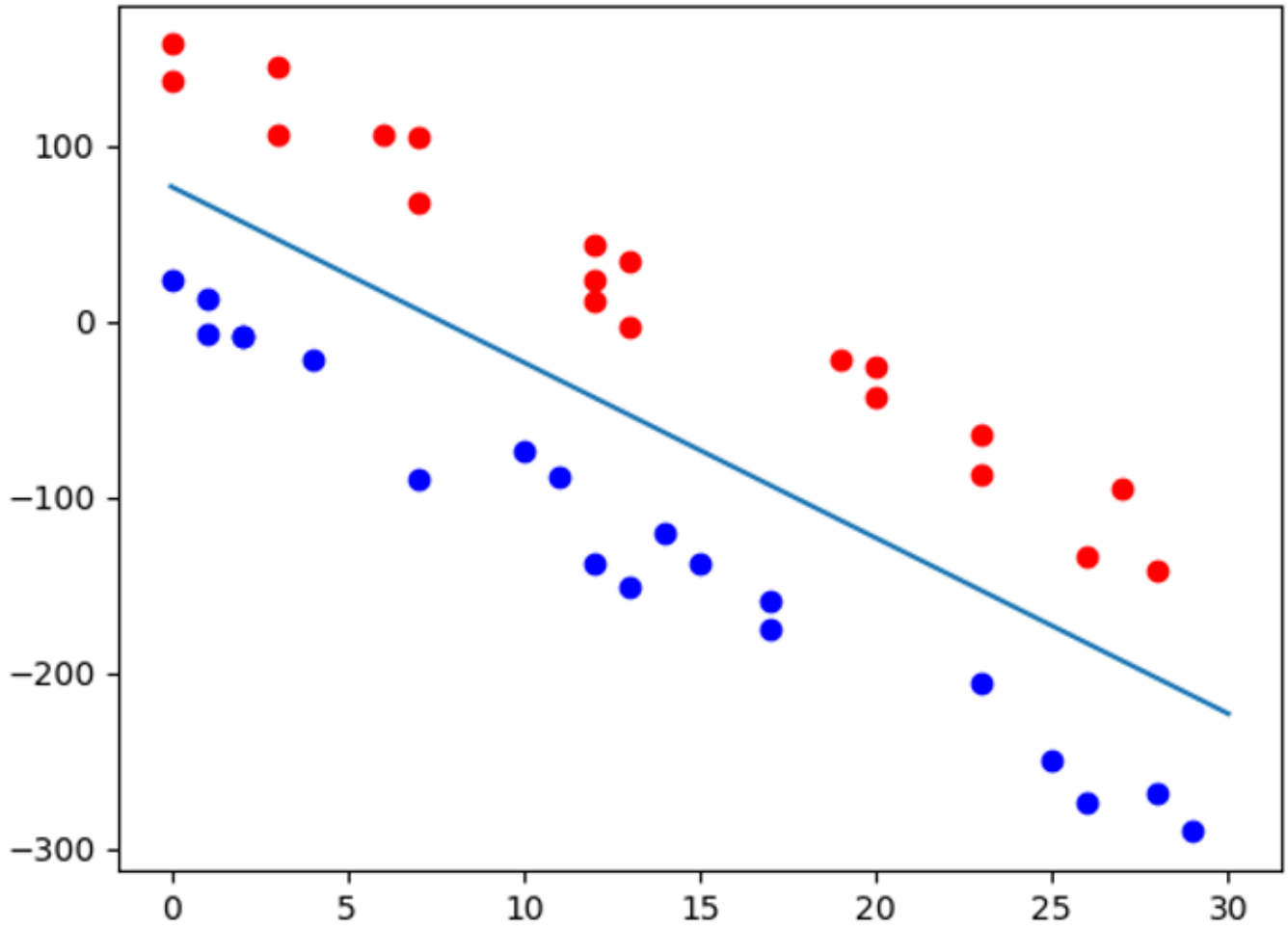
2.實驗結果

(1)



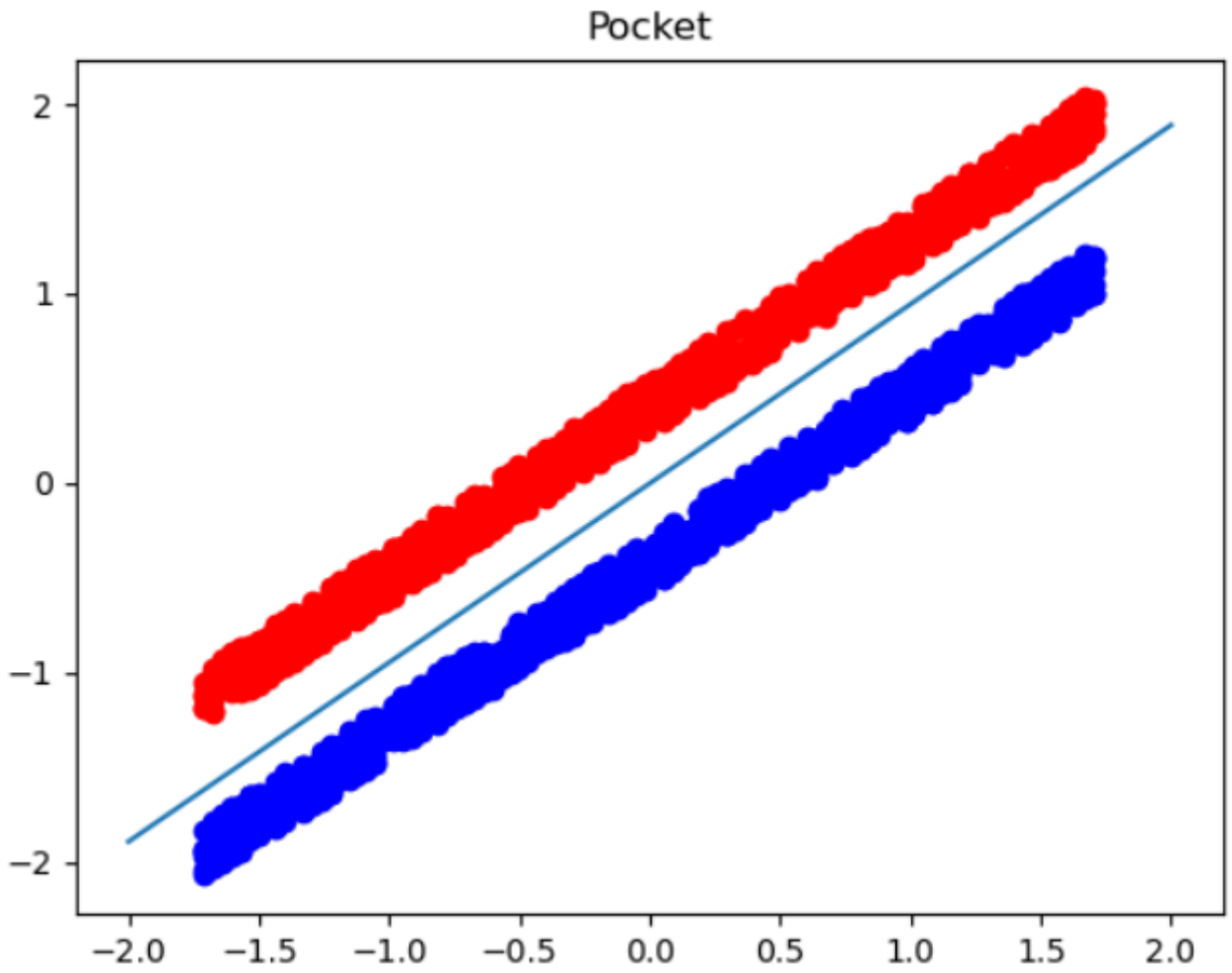
---

(2)



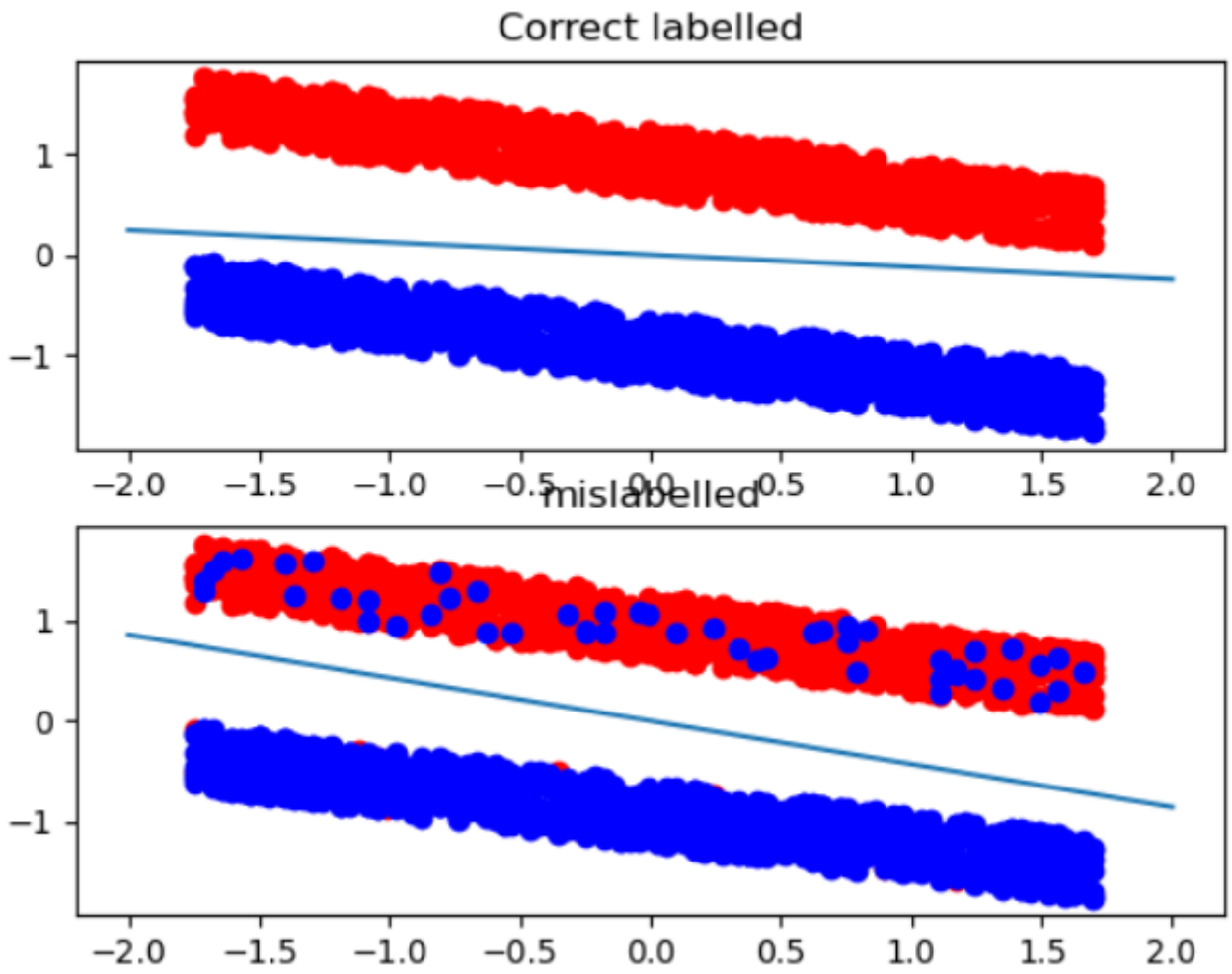
我總共做了五組的數據，並將結果整理在以下表格。

組數	1	2	3	平均
1	7781	2	29	2604
2	2	50	5	19
3	11584	2	123	3903
4	2	7300	42190	16497
5	4665	24	188	1626



(3) 我同樣做了五組數據，並將結果整理在以下表格。差距為透過PLA及pocket求出目標直線的時間差距。

組數	PLA(s)	pocket(s)	pocket的準確率(%)	差距(s)
1	0.01	0.17	100	0.16
2	0.01	0.1	100	0.09
3	0.02	0.11	100	0.09
4	0.03	7.50	90.5	7.47
5	0.02	0.43	100	0.41



(4) 我同樣做了五組數據，並將結果整理在以下表格。

組數	正確dataset得出的直線對正確dataset的準確率(%)	錯誤標記dataset得出的直線對正確dataset準確率(%)	錯誤標記dataset得出的直線對錯誤標記dataset準確率(%)
1	100	100	95
2	92.25	88.60	84
3	100	100	95
4	100	95	91.3
5	64.75	100	95

3.結論

(2)從實驗結果中可以發現PLA所需的迭代次數跟dataset有很大的關聯，若是此dataset很容易就能找到一個w來完美分類兩群點的話，PLA一下就結束了，但若是此dataset很難找到一個目標w分割的話，就需要更多的迭代才能成功找到目標w，差異甚大。 (3)從實驗結果中可以發現PLA找到目標直線的速度比pocket()還要快的多，而且在pocket演算法中會隨機挑選一點來更新當前w，若是一直都沒挑到好的點的話，此演算法就會一直迭代下去直到最大迭代次數為止，所需的時間也會因而增加許多。而PLA比pocket還快找到目標w的前提是dataset為線性可分的情況下，若是線性不可分的話PLA就會一直跑下去，不會收斂。 (4)從實驗結果中可以發現就算有標記錯誤的資料存在於dataset中，透過pocket()得出的直線對於正確dataset的準確率並不會太差，有時候可以到100%或是比由正確dataset產生的直線的準確率還要好。而對於錯誤標記dataset的準確率最高只有95%的原因

在於一開始就先故意標錯100個點了，因此準確率最高為1900/2000=95%。透過此實驗可以發現pocket演算法就算遇到非線性可分割的dataset照樣能找到一個直線盡可能的分類兩群點。而這同時也是pocket演算法優於PLA的地方，若是本來可線性分割的資料由於標記錯誤變成線性不可分的話，PLA就永遠不會停下來，無法收斂。

4.遇到的困難

在產生dataset的時候，因為剛開始接觸python的緣故，還不曉得該如何將每個點的x,y整合在一起，後來上網查後才得知可以先將list透過tuple()轉成tuple的型態後再透過zip的方式將不同的tuple合併成一個tuple，最後再將此tuple轉成list後再在最前面加一項常數1的項數，就可以產生一個型態為list且包含每個點的x, y的dataset，便於之後操作時取得每個點的x,y。