ML HW 6

- **Adaboost algorithm in the textbook:**

首先先利用csvfile讀取training data並將training data的attribute及label分開，label的部分前45筆的label設為0，後45筆資料的label設為1，用相同的方法讀取testing data，前5筆的label設為0，後5筆資料的label設為1。

接著先定義kNN需要使用的function分別為dis、takeSecond、kNN:

dis(ex1, ex2):

這是用來算距離的function，將兩個data set傳入，實作距離公式後return 距離。

takeSecond (elem):

這在後面用到sort()時會使用到，傳入一個list，回傳list第二個值。

kNN (ex, adaSet):

這是主要實作K-NN的function，傳入data及利用random出的adaSet，用一個for-loop跑10次，算出data與每個adaSet裡index對應的10個training data的距離，並append [training data的label, 距離]到ansset，由於我們是要找其中3個距離最近的點，於是用ansset.sort(key=takeSecond)讓距離由小排到大，接著用一個for-loop跑3次，計算前3小的距離對應的label個數(ans[label]=label個數)，最後回傳label個數較多的label。

再定義normalize function，將傳入的probability做normalize並回傳做完normalize的probability。

接著到主要執行程式的地方，利用雙層for-loop來做，最外面的for-loop跑9次，表示重複選取9個subset，每次選10個，初始的probability均為1/90(利用numpy套件來實現有probability的random)，裡面包了兩個for-loop，第一個跑10次，做10個testing data的kNN，並將結果放入testAttr(表示為testing data的Attribute，再後面用來測試做完的PLA)，第二個跑90次，用一樣的方法將kNN將果放入exAttr(表示為training data的Attribute)，接著執行AdaBoost的algorithm，用一個e陣列，如果kNN做出來的結果與自己原先的label不符則為1，一樣則為0，接著算90個probability與e的積的總和，並讓b=總和/(1-總和)，最後更新probability，只要更

新e為0的data的probability，更新方法為probability = probability*b，將全部更新完後做normalize，重複跑9次後我們即可得到所有testing data及training data的Attributes。

接著是perceptron的部分，define了兩個function，分別為 CH、Weight。

CH(pos, data):

這個function是在做公式裡c(x)-h(x)的部分，傳入pos(第幾個example)及data(example 的資料)，利用for-loop確認h(x)的值(做四捨五入確保正確性)後return h-C[pos]的值。

Weight(ch, x):

這個function在做weight更改的動作，依據講義的公式為原本的weight加上learning rate*(c(x)-h(x))*example的attribute值，(w0不需乘上example的attribute值，因為均為1)，並四捨五入到小數點下第一位。

接著是實作perceptron的部分，learning rate=0.2，將done、count初始化為0，將rund初始化為10000(避免無法收斂，如果做超過10000次則停止)，done用來判斷是否做完，count用來算已經連續對了幾個(c(x)=h(x))，對90個表示完成。用while-loop done!=1來做，在這個loop裡面跑一個for-loop(做90個examples)，call CH function 回傳值為0則count+1，回傳值不為0則call Weight function來更改weight的值並將count歸零，最後判斷當count=90或rund=0時表示完成，將done設為1。下圖為將testing data帶入做出來的正確性：我跑了兩次，一次為70%，一次為90%，可看出這個演算法的正確性高於單純只用PLA，但卻比original Adaboost差。

Accuracy of textbook adaboost: 70.0 %

Accuracy of textbook adaboost: 90.0 %

- **The original Adaboost algorithm by its inventors:**

讀取資料與textbook Adaboost相同，唯一不同的是將前45筆Label設為1，後45筆Label設為-1，中間KNN演算法及實作Adaboost的部分大致也與textbook Adaboost相同，不同的地方除了實作Adaboost時要將“b”值設為：總和/(1-總和)再開根號，還有需另外利用公式存alpha陣列(0~8)之外，剩下做法均與上面Adaboost algorithm in textbook相同，最後testing data的部分則是使用alpha陣列乘上對應testing data的attributes，最後算出accuracy。下圖為將testing data帶入做出來的正確性：我跑了兩次正確性均為100，可見Original Adaboost Algorithm的準確率是最高的。

Accuracy of original adaboost: 100.0 %

- **perceptron learning**

這次作業時做perceptron learning的方法與第三次作業的方法大致一樣，唯一不同的地方在於終止點，由於這次的training data無法收斂，我在終止點多加入一個條件為跑超過10000個epoch即停止。圖為將testing data帶入做出來的正確性：只有50%，為此次作業3個演算法中最差的。

Accuracy of PLA: 50.0 %