# MACHINE LEARNING assignment 1

Linear Classifer

苗鈺
*Institute of Data Science*
*National Cheng Kung University*
Tainan, Taiwan
a94035@gmail.com

## I. INTRODUCTION

本次作業內容包含手刻 update rule linear classifier、least-squared linear classifier，以及使用 margina 及 slack variablel 改進 update rule linear classifier 等三個部分

## II. DATA

本次作業包含兩個 dataset，分別叫做 crx 及 data:

### *A. crx*

crx 中包含 690 筆資料、16 個變項，其中"label"這個變項將作為預測目標，它包含"+"及"-"兩種類別。

在這個 data 中，某些觀察值是"?"，由於不清楚其意義，因此將其當作缺失值；又因為後續配適模型不能有缺失值因此會把任何包含缺失值的資料刪除，經過整理後的資料剩 653 筆。

針對缺失值作完處理後，下一步需要把類別型變數轉為 dummy variable，類別型變數包含"att4"、"att5"、"att6"、"att7"、"att9"、"att10"、"att12"、"att13"等 8 個變數，經過轉換後資料維度變為 653 筆資料，37 個變項。

轉換完成之後，把所有變數各自做標準化，要這麼做的原因是避免觀察值太大的幾個變數因算法導致重要性不一致。

最後為了分類器的需要，因此把 label 為"+"的觀察值替換成 1、label 為"-"的觀察值替換成-1。

### *B. data*

data 中包含 569 筆資料、30 個變項，沒有任何缺失值或類別型變數，而預測目標則是 Diagnosis 這個變數，它包含 M 及 B 兩種類別。

一樣把所有觀察值各自進行標準化，且把為了分類器的需要，把 Diagnosis 的 M 改為 1，B 改為-1。

## III. METHOD

假設 X 是一個 N 筆資料，K 個觀察值的資料，而本次作業的目標是希望能找到一組參數估計$\hat{B}$，使得回歸線$Y = \hat{B}X + \hat{C}$能夠盡可能的把資料預測到正確的那一側，其中$\hat{B}$是長度為 k 的一個向量、$\hat{C}$代表的是回歸線的常數項估計值，具體算法如下:

假設已知一條回歸線$Y = \hat{B}X + \hat{C}$，帶入第 i 筆觀察值以得到此觀察值在該分類線下的分數

$$score_i = \hat{B}X_i + \hat{C} \qquad \text{公式一}$$

最後帶入公式一預測這筆資料的 label:

$$\hat{Y_i} = \begin{cases} 1 & ,if\ \text{score}_i > 0 \\ -1 & ,if\ \text{score}_i \le 0 \end{cases} \qquad \text{公式二}$$

為了找出這個參數估計$\hat{B}$，本次作業使用四種方式來尋找這個$\hat{B}$，分別是 update rule、least-squared 、update rule with the minimum marginal distance、update rule with the minimum marginal distance and add the slack variable.

### *A. Least-squared*

直接使用 Least-squared 公式來估計參數估計值 B，公式 2 如下:

$$\hat{B} = (X^T X)^{-1} X^T Y \qquad \text{公式三}$$

### *B. Update rule*

Update rule 是一筆一筆看過所有資料，並慢慢進行參數調整的一個方式，具體步驟可以分為三個部分:

- 步驟一:先把所有資料隨機排列，其原因是這個算法很容易受到次序的影響。
- 步驟二設置一組初始權重(本次作業是直接預設$\hat{B}$初始值為零向量，常數$\hat{C}$也是 0)
- 步驟三:將第 1 筆資料的觀察值 $X_i$ 帶入公式一與公式二，算出目前的預測值$\hat{Y_i}$，如果$\hat{Y_i} = Y_i$則不改變$\hat{B}$，如果$\hat{Y_i} \neq Y_i$則進入步驟四。
- 步驟四:對 B 進行調整，調整方式如下:

$$new\ \widehat{B_k} = \begin{cases} \widehat{B_k} + label_i\ ,if\ label_i \times x_k > 0 \\ \widehat{B_k} - label_i\ ,if\ label_i \times x_k \le 0 \end{cases}$$

$$new\hat{C} = \begin{cases} \hat{C} + step\ ,if\ \ label_i > 0 \\ \hat{C} - step\ if\ \ label_i \le 0 \end{cases} \qquad \text{公式四}$$

其中 step 是需要自己指定的一個參數，意義是每次調整時常數 C 要增加或減少。

- 步驟五:把:$\widehat{B_k}$ *更新成*$new\ \widehat{B_k}$ ，$\hat{C}$ *更新成*$new\hat{C}$

  其中 i 代表第 i 個觀察值，k 代表第 k 個變數，其意義就是如果第 i 筆資料的第 k 個觀察值和第 i 筆資料的 label 同號，那就直接把第 k 個參數估計值$\widehat{B_k}$加上第 i 筆資料的 label(如果都是正的，$\widehat{B_k}$*會加一，如果都是負的*$\widehat{B_k}$*會減一*)，如果異號則是減去第 i 筆資料的 label；最後在重新算一次在新的參數估計$\hat{B}$下預測結果如何，如果正確那就換看下一筆資料，如果錯誤，那要一直疊代到正確為止，或者也可以設置一個疊代上限，避免過度學習某個樣本的特徵。

*C. Update rule with voted perception and average perception*

*在 Update rule 底下，套用 voted perception 及 average perception 的算法，這兩種算法都是配適好幾個模型(令為 model_n)，之後把模型結果進行合併，差別在於 voted perception 是所有模型各自預測結果，之後看是預測為哪一個類別的比較多，就以該類別為最終預測結果，average perception 則是把好幾個模型訓練完成後，將他們所有的參數估計$\hat{B}$平均作為最終的參數估計，並使用這組參數估計去進行預測。*

*D. Update rule with the minimum marginal distance*

在 update rule 的算法基礎之下，額外新增一個變數 W，W 算法如下:

$$W = \frac{|(\hat{B}X+\hat{C})|}{\sqrt{\hat{B}\times\hat{B}^T}} \qquad \text{公式五}$$

這個是在算點到線歐式距離，最後算出||W||^2，在幾何意義上就是說，所有的點到線的距離之和，而我們希望能夠最小化這個||W||^2，因此原始 Update rule 的步驟之下，步驟五需要改成這樣:

- 步驟五:套用公式五，分別算出$\widehat{B_k}$及$new\ \widehat{B_k}$的||W||^2，(以$\|W_{\widehat{B_k}}\|$^2 及$\|W_{new\ \widehat{B_k}}\|$^2)，如果$\|W_{\widehat{B_k}}\|$^2 比較小，那就維持原參數估計$\widehat{B_k}$，如果$\|W_{new\ \widehat{B_k}}\|$^2 比較小，那就使用更新後的參數估計$new\ \widehat{B_k}$。

另外步驟二初始參數的設置也不能都設置為 0，否則在公式五的分母會等於零，造成無法計算 W，因此此部分的初始值改為$\hat{B}$ = (6,6,6,…,6)，長度為變項的數量 k，常數項$\hat{C}$則為-k。

*E. Update rule + marginal distance + slack variable*

*在 Update rule with the minimum marginal distance* 的算法之上再額外多一個 slack variable S:

$$SW_i = \begin{cases} SW_i, SW_i \geq S \\ 0 \quad , SW_i < S \end{cases} \qquad \text{公式六}$$

做完此更新後，再計算||SW||^2，然後一樣套用到步驟五之中。

而這個 slack variable S 在幾何的意義上就是代表容許這條線有一定的誤差，在誤差範圍 S 之內的 W 視為不確定的因素，不去計算它點到線的距離，然而這樣一來會有一個缺點，只有 S 設為無限大，那||W||^2 永遠都是 0，這樣會沒有意義，因此這個參數 S 不可以太大。

*F. SVC*

套用 sklearn 已經寫好的函式做線性判別模型，並以此作為對照組，看看自己寫的分類器的表現。

## IV. EXPERIMENT

綜合比較整體的結果，其中 A 方法代表: Least-squared，B 方法代表基礎的 Update rule linear classfier，C1 代表使用 voted perception 的 Update rule，C2 代表使用 average perception 的 Update rule，D 方法代表加入了 marginal distance 的 Update rule linear classfier，E 方法代表加入了 marginal distance 及 slack variable 的 Update rule linear classfier，F 方法代表 sklearn 的 SVC。

| | Accuracy | |
|---|---|---|
| Model | crx | data |
| A | 0.6845 | 0.9649 |
| B(step=30) | 0.7718 | 0.9174 |
| C1(step=30,model_n=25) | 0.8760 | 0.9772 |
| C2(step=30,model_n=25) | 0.8821 | 0.9824 |
| D(step=2) | 0.7029 | 0.7680 |
| E(step=2,S=0.5) | 0.6968 | -- |
| E(step=2,S=3) | -- | 0.8981 |
| F | 0.8668 | 0.9877 |

*註:由於 Update rule 本身包含隨機排序的部分，種子碼統一設為 6111016。

結論而言，可以發現自己做的 model 裡，都是 C2(average perception)表現是最佳的，但實際上影響結果的原因很多，根據我的嘗試，可以列出以下幾點影響的原因:

- 種子碼:種子碼可以讓結果差異非常大，在其他參數不變的前提下，正確率可以相差 0.2 左右
- 起始位置:起始的位置如何調整其實也會大幅影響結果，以模型 D、E 來說，起始位置我是設$\hat{B}$ = (6,6,6,…,6)，會是這組數字也只是因為他的表現是最好的。
- 常數項每次調整大小(step):step 的大小也會對結果產生很大的影響，如果 step 設太大，很容易讓結果變成全部都預測為 1 或-1。

另外，由於我並沒有將資料分割為 train data 及 test data，因此就算正確率比較高，也不一定代表該模型是好的，因為有可能有過度擬合的問題存在。

最後探討為何 minimum marginal distance 的方法正確率反而變低，我認為是因為起始位置設置不好，以及 B 的疊代都只有+1 或-1 的關係，導致式子無法收斂到全域最佳解，只收斂在局部最佳解，導致正確率無法提升。

程式碼: github link