

# SP\_HW4

# Programming Assignment

---

TA 李承軒



# Problem Description

- ✿ 根據 *training\_data* 做出一個 *random forest model*  
( *Training step* ! )
- ✿ 接著讀入 *testing\_data* 並做出解答 ( *Testing step* ! )



Simple right ?



# Data Description

- ✿ *Training\_data* :

第一欄為*ID*，第二到三十四欄為*feature*，最後一欄為*label*

- ✿ *Testing\_data* :

第一欄為*ID*，第二到三十四欄為*feature*，你的工作就是要做出最後一欄



HOW?



# Random Forest

## ✿ *Training step* :

(a). 從*training\_data*讀出資料以下記為*training\_dataset*，**切記不要把ID丟進去**

(b). 從*training\_dataset*裡取出跟*training\_dataset*(等量)的資料，**取後放回，所以可能會取到重覆的資料**

(c). 拿第二步取出的*data*，當做*input*拿去做一棵*decision tree*

(d). 重複*step(b)* & *step(c)*數次，把做出來的一堆*decision tree*集合起來就是*random forest*了



# Random Forest

- ✿ *Testing step :*

- (a). 自 *testing\_data* 取出資料

- (b). 把每一筆資料丟進去你剛剛做出來的每一棵 *decision tree*，每一棵 *decision tree* 會告訴你它是好人或是壞人

- (c). 用投票的方式決定最後的答案!



# Decision Tree

## ✿ *Training step :*

(a). 把剛剛的*input data*丟進*root node*中

(b). 從各個維度中尋找最佳切點

第*X*維尋找最佳切點的方式

(1). 先對第*X*維的資料*sort*過

(2). 從最小的數值開始切，計算出*gini impurity*

(3). 最小的那個*gini impurity*的那個切點即該維的最佳切點

(c). 比較哪個維度切點的*gini impurity*最小，即為最佳切點，記錄維度及  
*threshold*

(d). 把比最佳切點*threshold*小的*data*丟給左邊*node*，其他丟給右邊

(e). 重複步驟(b) (c) (d)直至該*node*中的*data label*皆為0 或 1，即*gini impurity*為0



# Gini Impurity

- ✦ 某個 $node$ 的 $gini\ impurity$  :

其中 $J$  指的是有幾種 $label$  ,  $f_i$  指的是該 $label$ 在全部 $data$ 的比率

$$I_G(f) = \sum_{i=1}^J f_i (1 - f_i)$$



# Example

ID	Weight	Horse Speed	Label
1	1000	300	1
2	500	500	1
3	300	150	0
4	200	100	0



# Example

- ✦ 第一維自 1000 跟 500 中間往下切

*1000 vs 500 300 200 :*

*Gini impurity of 1000 :*

$$0*(1-0) + 1*(1-1) = 0$$

*Gini impurity of 500 300 200 :*

$$0.66*(1-0.66) + 0.33*(1-0.33) = 4/9$$

$$\text{Total gini impurity} = 0 + 4/9 = 4/9$$



# Example

- ✦ 第一維自500跟300中間往下切

*1000 500 vs 300 200 :*

*Gini impurity of 1000 500 :*

$$0^*(1-0) + 1^*(1-1) = 0$$

*Gini impurity of 300 200 :*

$$1^*(1-1) + 0^*(1-0) = 0$$

$$\text{Total gini impurity} = 0$$



# Example

- ✦ 依此類推，做完第一維跟第二維  
最後可以得到從第一維的500跟300中間切會是最佳



# Decision Tree

- ✿ *Testing step* :

- (a). 將*Random Forest*丟進來的*test data*，依照每個*node*所記錄的維度及*threshold*看是走左邊還是右邊
- (b). 一直走到底，看最後的*node*的*label*為何，即為所求



演算法就到這!  
開始計結!



# What is instructions ?

- ✿ *gcc hw4 -o a.out*
- ✿ *What's a.out?*



# What is instructions ?

===== BEGINNING OF PROCEDURE =====

```

;
; Section __text
;
; Range 0x100000aa0 - 0x100000ea9 (1033 bytes)
; File offset 2720 (1033 bytes)
; Flags : 0x80000400
;
_main:
00000000100000aa0      push      rbp                      ; XREF=0x1000000d0
00000000100000aa1      mov       rbp, rsp
00000000100000aa4      sub       rsp, 0xa0
00000000100000aab      lea       rdi, qword [ds:0x100000f28] ; "Please input y:\\n", a
00000000100000ab2      xorps     xmm0, xmm0
00000000100000ab5      mov       dword [ss:rbp+var_4], 0x0
00000000100000abc      mov       dword [ss:rbp+var_1C], 0x0
00000000100000ac3      mov       dword [ss:rbp+var_20], 0x64
00000000100000aca      mov       byte [ss:rbp+var_29], 0x30
00000000100000ace      movsd     qword [ss:rbp+var_38], xmm0
00000000100000ad3      mov       al, 0x0
00000000100000ad5      call      imp__stubs__printf
00000000100000ada      lea       rdi, qword [ds:0x100000f39] ; "%lf", argument "format
00000000100000ae1      lea       rsi, qword [ss:rbp+var_10]
00000000100000ae5      mov       dword [ss:rbp+var_54], eax
00000000100000ae8      mov       al, 0x0
00000000100000aea      call      imp__stubs__scanf
00000000100000aef      lea       rdi, qword [ds:0x100000f3d] ; "Please input cashflow:
00000000100000af6      mov       dword [ss:rbp+var_58], eax
00000000100000af9      mov       al, 0x0
00000000100000afb      call      imp__stubs__printf
```



# What is instructions ?

- ✿ 簡單來說，*instructions* 數量跟工作量成正比



# Perf

- ✿ *Perf* 是一個 *Linux* 系統效能評估的工具，請使用以下指令來得到你從讀檔開始到預測出答案所使用的 *instruction* 數量，此工具在工作站上就有了不必額外下載

```
perf stat -e instructions:u -v ./hw4
```

- ✿ *perf list* 可以看更多！！！！



# You will get

- ✿ 1. *training\_data*
- ✿ 2. *testing\_data*
- ✿ 3. *sample\_submission.csv* : 你*output*的格式
- ✿ 4. *ans.csv* : *testing\_data*的解答(給你衡量自己做出來的正確性，因為不是*Machine Learning*課，所以給此解答，但你的程式執行時不得使用任何此解答的資訊，也不得嵌入程式內)



# Submission

- ✦ 命名：*hw4\_你的學號.zip* (Ex: *hw4\_b01902020.zip*)

- ✦ 其中需包含

  - (a). *hw4.c*

  - (b). *Makefile*

執行*make*可正確*compile*你的*code*，不得使用*-O1 -O2 -O3 -Os*

執行*make run*可以*run* 你的*code*，*data\_dir*預設為“*../data*”，*output*預設為“*./submission.csv*”，*tree\_number*及*thread\_number*為你自行設定最好的參數，其中*thread\_number*需大於等於2，時限為3分鐘

  - (c). *report.pdf*



# Score

- ✿ (a). [Code]: 你的code能被compile，可以執行以下指令，在時限(3分鐘)內跑出結果，結果正確率需大於80% (1%)

*./hw4 -data data\_dir -output submission.csv -tree tree\_number -thread thread\_number*

其中：

*-data* : *data\_dir*代表*training\_data testing\_data*所在的資料夾

*-output* : *submission.csv*代表結果的輸出檔案

*-tree* : *tree\_number*代表種幾顆樹

*-thread* : *thread\_number*代表開的*thread*數量(因為你有可能開*thread*開在很多地方，所以此數字是同一時間所有*thread*的數量，此數量需大於等於2)

切記要是此項沒拿到分，後面的項次皆不予計分



# Score

- ✿ (b). [Report]: 試說明你將`thread`開在哪裡，是分工在哪裡？(1%)
- ✿ (c). [Report]: 試畫出或以表格做出`thread`數量與時間的比較，以紅色標出時間最快的位置，並說明此圖表(2%)
- ✿ (d). [Report]: 試畫出或以表格做出`thread`數量與`instructions`數量的比較，並說明此圖表(1%)
- ✿ (e). [Report]: 試畫出或以表格做出樹的數量與`intructions`數量的比較，並說明此圖表(1%)
- ✿ (f). [Report]: 說說你的其他發現! (可以是與正確率的比較啦，或是哪個`function`會造成大量`cache miss`啦 都可以 都來都來!) (1%)



# Rules

✿ *Other rules :*

(a). 不得遲交

(b). 不得抄襲

(c). 只許使用 *C* , *C++*不行!