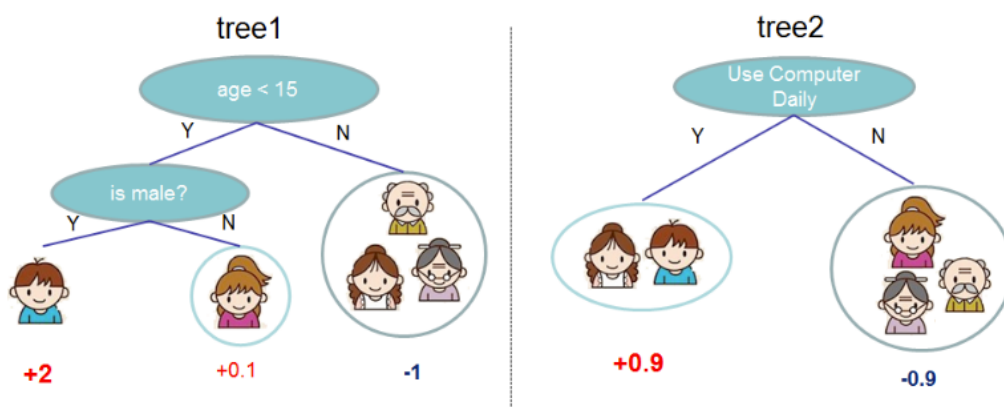


XGboost (eXtreme Gradient Boosting) 極限梯度提升

XGBoost 即是數顆「分類與回歸樹(Classification and Regression Tree；CART)」組合成一個高準確率的模型。每加入一次新的函數至原模型中，修正上一顆樹的錯誤，不斷的迭代以提升整體的模型。如圖一所示，Tree1 根據年紀與性別分類，藍衣小男孩所得的葉子分數為+2；Tree2 為每日使用電腦習慣，小男孩所得的葉子分數為+0.9，因此最後的模型結果，小男孩總共獲得 2.9 分，反之，阿公的分數為-1.9，而這個累加結果則會作為預測結果。



圖一、數集成模型（Chen and Guestrin, 2016）。

每棵樹的模型可被定義為：

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^K f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F}, \quad (1)$$

其中， $f(x) = w_{q(x)}$ ， w 為權重、 q 為這棵樹中該葉子的 index（這棵樹共有 T 片葉子； $q: \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T$ ）； K 為樹的總數量。換句話說，對於第 i 個樣本而言，在第 k 顆樹的中，所對應的葉子權重為 $f_k(x) = w_{q(x)}$ ；在第 $k+1$ 顆樹的中，所對應的葉子權重為 $f_{k+1}(x) = w_{q(x)}$ ，而這個樣本所得的分數即為 $f_k(x) + f_{k+1}(x) + \dots$ 。

為了修正上顆樹的錯誤，則需要定義目標函數 $Obj(\theta)$ 來優化，目標函數函數可表示成：

$$\sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \quad (2)$$

前項為損失函數（loss function），用來量化預測 \hat{y}_i 與真實 y_i 的差異；後項為懲罰項，其描述樹的複雜度，也用來避免迭代有 overfitting 的狀況出現。而 XGB 是基於「修正上一顆樹的錯誤」來求得最佳模型，因此，在第 t 次迭代中，都在修正第 $t-1$ 次的迭代情形，故式 2 可寫成：

$$\sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t) \quad (3)$$

每一次的迭代（ t ）即希望 $f_k(x_i)$ 擁有最小值。接續使用泰勒展開式（Taylor expansion）：

$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x - a)^n$ ，來尋找目標函數的近似值，式 3 的 $\hat{y}_i^{(t-1)}$ 可對應泰勒展開式的 x 、 $f_t(x_i)$ 則對應 a ，進而將式 3 寫成：

$$\sum_{i=1}^n [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) + constant \quad (4)$$

其中， $g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ 、 $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ ，也就是對應泰勒展開式的一

（二）階導數，尤其，二階導數使梯度收斂的更快更準確。因為對於第 t 次迭代而言， $l(y_i, \hat{y}_i^{(t-1)})$ 為常數，故後續推倒可省略。此外，懲罰項 $\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$ 、每棵樹可得的分數 $f(x) = w_{q(x)}$ ，可將式 4 改寫（詳細推導見 <https://www.hrwhisper.me/machine-learning-xgboost/>）：

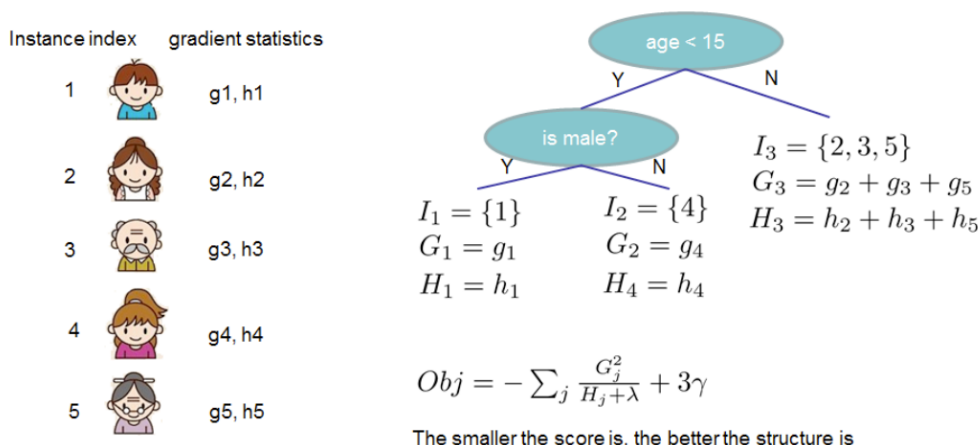
$$\sum_{j=1}^T [G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2] + \gamma T \quad (5)$$

其中， $G_i = \sum_{i=1}^n g_i$ 、 $H_i = \sum_{i=1}^n h_i$ 。式 5 也就寫成了 $ax^2 + bx + c$ 的形式，若欲求此式

的最小值，則 $w_j^* = \frac{-G_j}{H_j + \lambda}$ ，將 w_j^* 代回式 5，Loss function 可改寫為：

$$-\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T \quad (6)$$

該式可用於判斷該模型的好壞（圖二）。

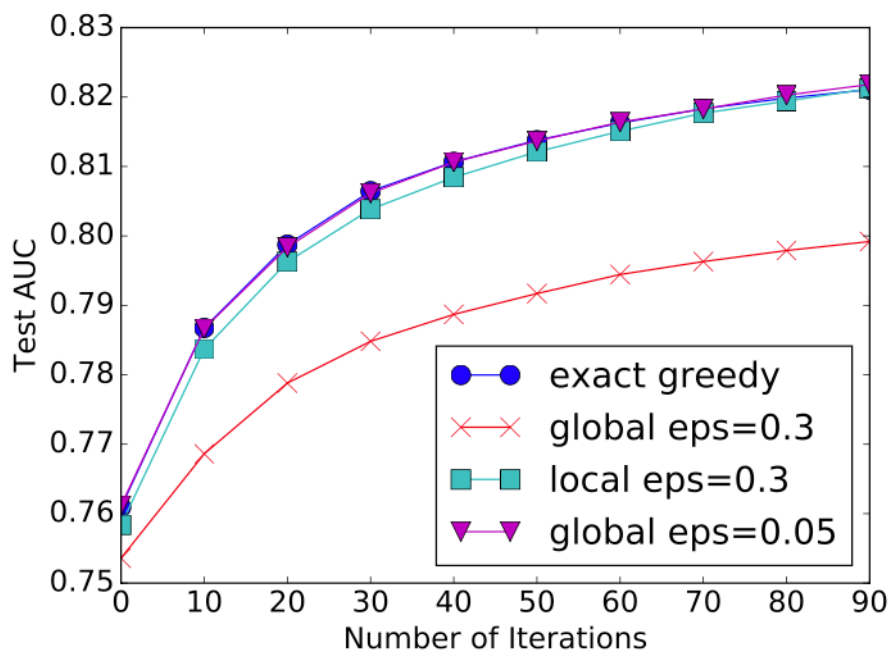


圖二、樹結構計算範例（Chen and Guestrin, 2016）。

一棵樹主要使用兩種方法進行劃分：Exact Greedy Algorithm 與 Approximate Algorithm，前者較適用於數據量小的狀況，該算法會將「所有的特徵列出且劃分」，計算「劃分後的左葉與右葉分數和」及「劃分前的分數」差（式七），即為 Gain，當 Gain 越大時代表 Loss function 下降的越多，進而找出最好的劃分點；後者則適用於數據量大的狀況，先根據k個特徵選擇分位點（Quartile），將這些分位點視為要觀察的劃分點，只要考慮這些劃分點的結果即可衡量該模型的好壞。

$$\frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \quad (7)$$

Approximate Algorithm 的分點選取時間點策略可分為 Global 與 Local，前者在「學習每棵樹前」，提出候選的分位點；後者在「每次分裂前」，重新提出候選的分位點，直覺上來說，Local 需要較多計算步驟、Global 需要提出較多的候選分位點。從圖三來看，global 的分位點夠多（ $1/0.05=200$ ）時，與 Exact Greedy Algorithm 的效能一致，同時，local 的分位點少量（ $1/0.3=33.3$ ）即可達到一樣的效果。



圖三、以 Approximate Algorithm 建立樹模型的分位點方式 (Chen and Guestrin, 2016)。
eps 的倒數為分位點數量。

對於缺失值的處理，在 GBDT (Gradient Boosting Decision Tree；梯度提升決策樹) 中，會先手動對缺失值填充 (沒有任何依據的狀況)，當作有值的特徵處理；相反地，XGB 僅先利用有值的數據尋找特徵建立模型，再嘗試將缺失值劃入對應分支，選擇損失最優的值作為分裂點，並且其複雜度降低，計算速度提升。

對於抽樣，XGB 借用了隨機森林 (Random forest) 的欄抽樣 (每一次分裂只使用被抽取的特徵)，防止 overfitting 也加速訓練過程。

XGB 參數設定 (詳見....ipynb)

1. `n_estimators`: 樹的數量 [default: 10]
2. `booster`: 模型種類 [default: gbtrees]
 - `gbtree`: tree-based models
 - `gblinear`: linear models
3. `max_depth`: 每棵樹的最大深度，越大越能學到更局部的樣本，但也有可能 over fitting [default: 6]
4. `eta/learning_rate`: 學習速率，完成一次迭代後會把葉子權重乘上此係數，削弱前棵樹的影響，讓後面的樹有更大的空間可學習/修正 [default: 0.3]
5. `tree_method`: 樹的劃分方法 [default: "auto"]
 - `auto`: 對於小數據集--> "exact"; 對於大數據集--> "approx"
 - `exact`: Exact Greedy Algorithm
 - `approx`: Approximate Algorithm
 - `hist`: 更快的直方圖優化法 (based on Exact Greedy Algorithm)
 - `gpu_hist`: 加入 GPU 的 hist 計算:
6. `min_child_weight`: 葉子的最小樣本數，與式 5 的 H_j (二階導數項的和) 有關。假設 h 在 0.01 附近，`min_child_weight` 為 1，代表葉子最少需要 100 個樣本。此值愈小愈容易 overfitting [default: 1]
7. `gamma`: 即「 γ 」，越大越保守 [default: 0]
8. `lambda`: L2 正歸化參數，用於控制模型複雜度，參數越大越不容易 overfitting [default: 1]
9. `alpha`: L1 正歸化參數，用於控制模型複雜度，參數越大越不容易 overfitting [default: 0]
10. `max_leaves`: 不與 `tree_method="exact"` 同用 [default: 0]
11. `colsample_bytree`: 生成樹時所進行的 column 採樣，通常設置為 0.5-1 [default: 1]

- 12. `seed`: 隨機種子 [default: 1000]
- 13. `max_delta_step`: 取正值使迭代步驟更加保守，若該值為 0 則沒有約束
[default: 0]