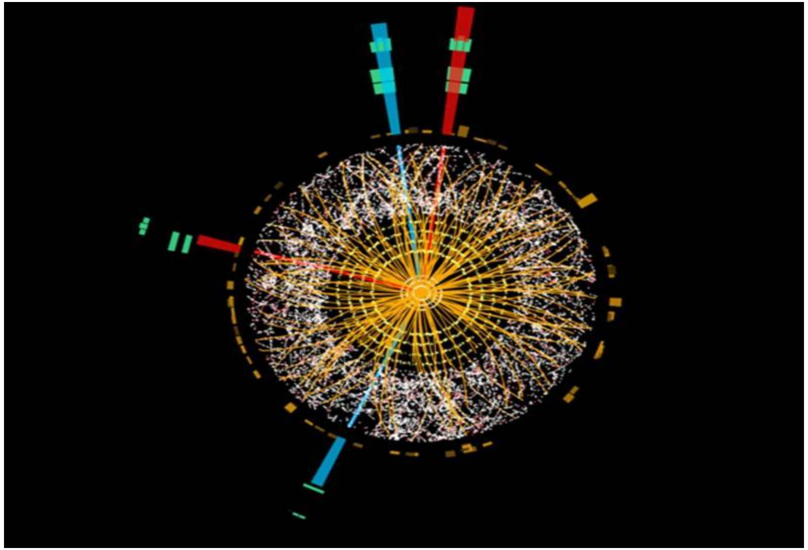
XGBOOST



EXtreme Gradient Boosting

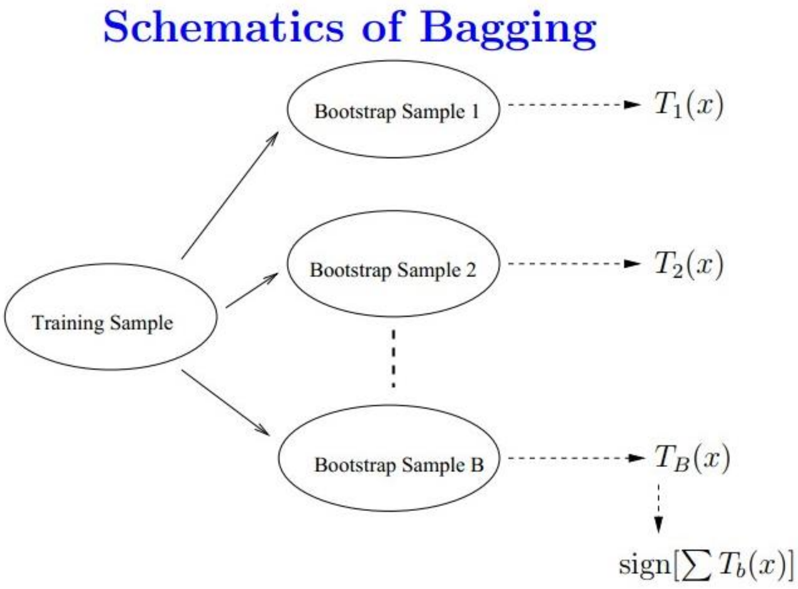
- Large Scale Tree-Based model
- Computation in C++
- Linear model + Tree

Difference

Bagging

- 訓練集的選擇是隨機的且相互獨立
- 取後放回

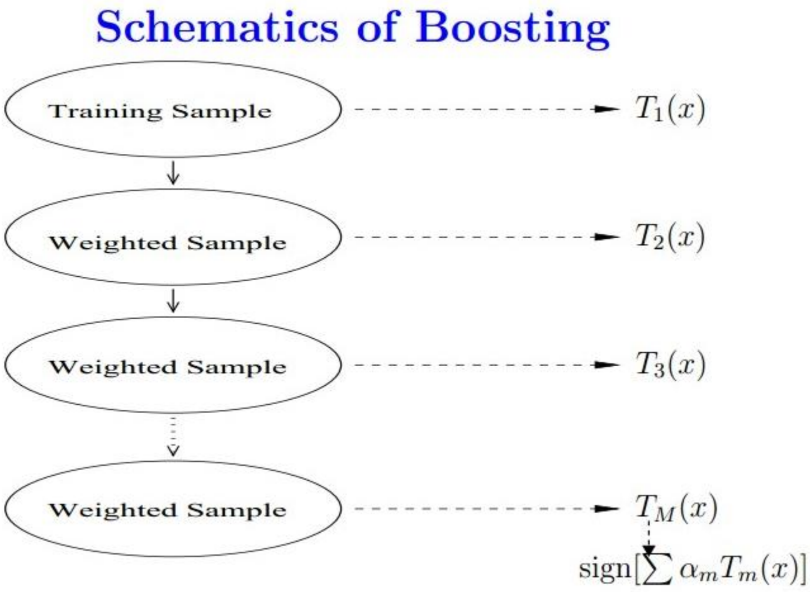
Bagging


Schematics of Bagging
Bootstrap Sample 1
$T_1(x)$
Bootstrap Sample 2
$T_2(x)$
Training Sample
Bootstrap Sample B
$T_B(x)$
$\text{sign}[\sum T_b(x)]$

Boosting

- 根據錯誤率給樣本一個weight值，各輪訓練集的選擇與前面各輪的學習結果有關（i.e.分錯類的樣本weight大）
- 準確性較高，但會造成overfitting

Boosting


Schematics of Boosting
Training Sample
Weighted Sample
Weighted Sample
Weighted Sample
$T_1(x)$
$T_2(x)$
$T_3(x)$
$T_M(x)$
$\text{sign}[\sum \alpha_m T_m(x)]$

Bagging vs Boosting

Bias-variance tradeoff

- Bagging → 將訓練出來的模型取平均，降低variance
- Boosting → 優化loss function，降低bias

一句話形容Boosting

知錯能改

AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in \{-1, +1\}$

Initialization: $D_1(i) = \frac{1}{m}, i = 1, \ldots, m$

For $t = 1, \ldots, T$:

- Find classifier $h_t : X \to \{-1, +1\}$ which minimizes error wrt D_t ,i.e.,

$$h_t = \arg\min_{h_j} \varepsilon_j \quad \text{where } \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$

- Weight classifier: $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

- Update distribution: $D_{t+1}(i) = \frac{D_t(i) \exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$, Z_t is for normalization

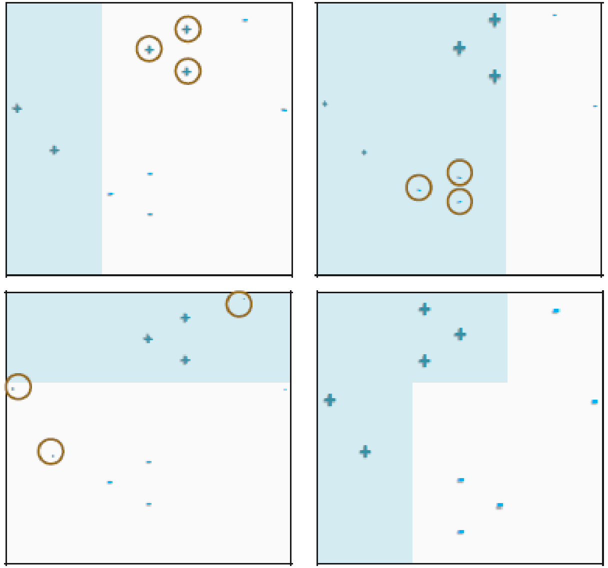
Output final classifier: $sign\left(H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \right)$

AdaBoost

AdaBoost – adaptive boosting

h_1
$\varepsilon_1 = 0{,}30$
$\alpha_1 = 0{,}42$

h_2
$\varepsilon_2 = 0{,}21$
$\alpha_2 = 0{,}65$



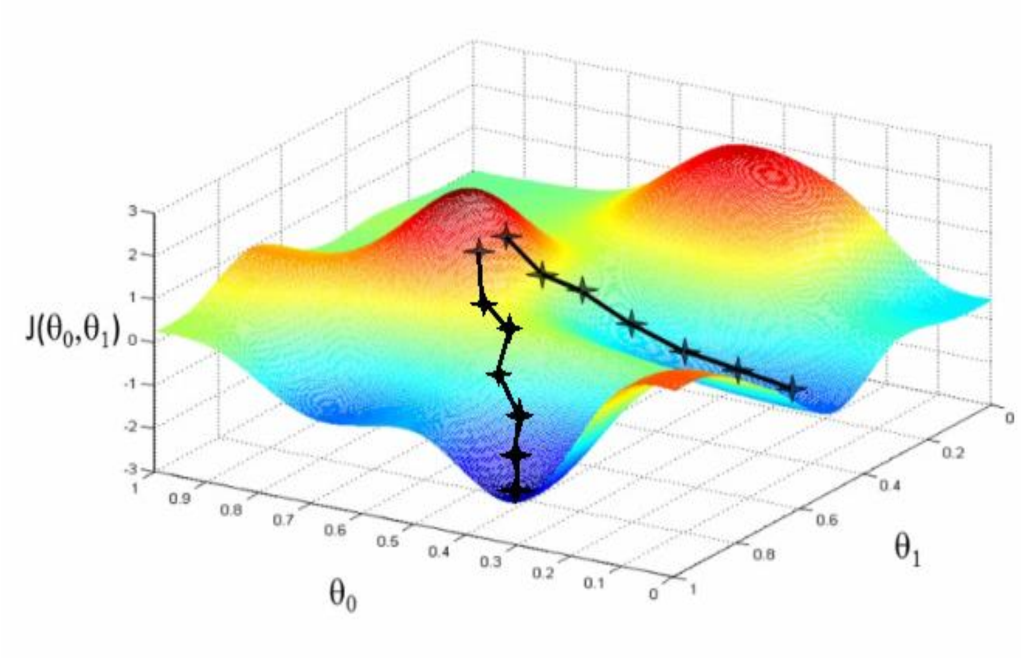
h_3
$\varepsilon_3 = 0{,}14$
$\alpha_3 = 0{,}92$

$$H = 0{,}42h_1 + 0{,}65h_2 + 0{,}92h_3$$

AdaBoost

Gradient Boosting

- 模型是建立在之前模型損失函數的梯度下降方向。



Gradient Boosting

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

    $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

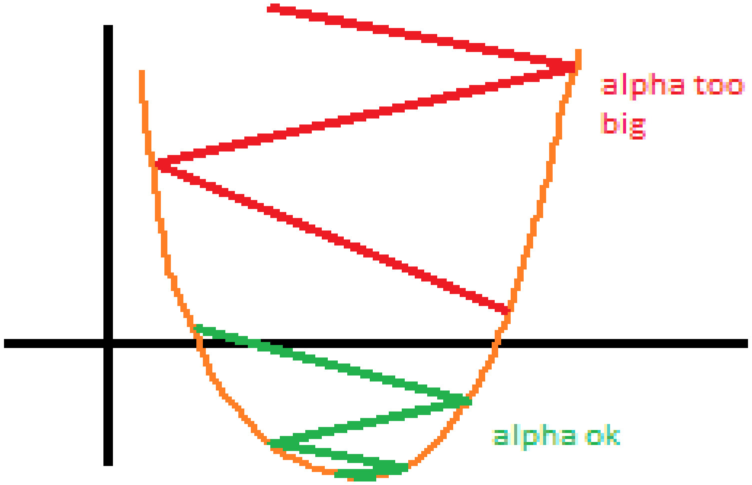
    $$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

    $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Gradient Boosting




「然後他就死掉了。」
— 羅瑩雪

Extreme Gradient Boosting

- 目標函數中顯示的加上了正則化項，正則化項與樹的葉子節點的數量T和葉子節點的值有關。

- Loss Function使用到了一階、二階導數。

Extreme Gradient Boosting

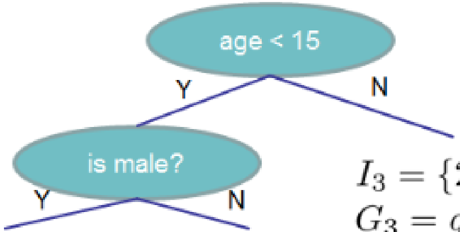
$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^{n} \left[g_i f_t(x_i) + \tfrac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \\ &= \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \tfrac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \tfrac{1}{2} \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \tfrac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

- where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

$$G_j = \sum_{i \in I_j} g_i \quad H_j = \sum_{i \in I_j} h_i$$

Extreme Gradient Boosting

样本号	梯度数据
1	g1, h1
2	g2, h2
3	g3, h3
4	g4, h4
5	g5, h5



$$I_1 = \{1\} \quad G_1 = g_1 \quad H_1 = h_1$$

$$I_2 = \{4\} \quad G_2 = g_4 \quad H_4 = h_4$$

$$I_3 = \{2, 3, 5\} \quad G_3 = g_2 + g_3 + g_5 \quad H_3 = h_2 + h_3 + h_5$$

$$Obj = -\frac{1}{2}\sum_j \frac{G_j^2}{H_j+\lambda} + 3\gamma$$

这个分数越小，代表这个树的结构越好


太可怕了

Extreme Gradient Boosting

$$Gain = \frac{1}{2}\left[\underbrace{\frac{G_L^2}{H_L+\lambda}}_{\text{左子树分数}} + \underbrace{\frac{G_R^2}{H_R+\lambda}}_{\text{右子树分数}} - \underbrace{\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}}_{\text{不分割我们可以拿到的分数}}\right] - \underbrace{\gamma}_{\text{加入新叶子节点引入的复杂度代价}}$$

Method

- 僅適用於數值型向量。
- One Hot Encoding
- Sparse Matrix

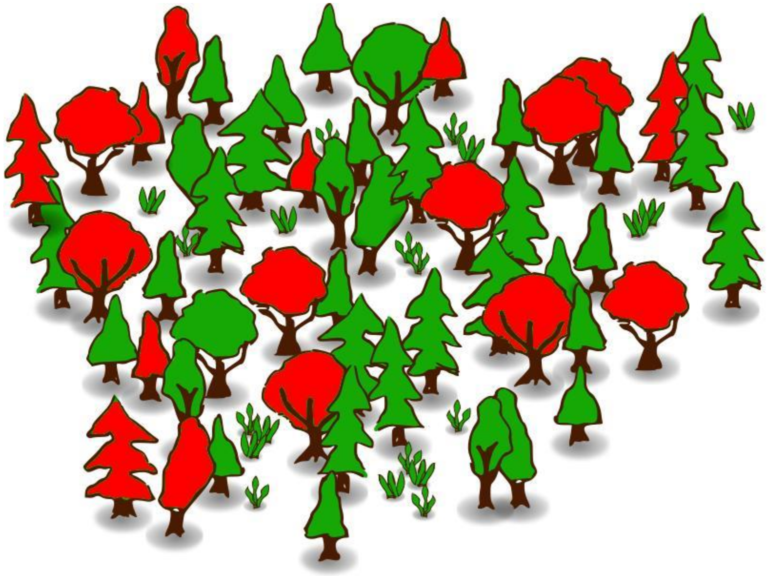
Random Forest

- 其實就是Bagging → 樹與樹之間無關連
- 每棵樹很弱 →

一棵不夠

那很多棵呢

Random Forest



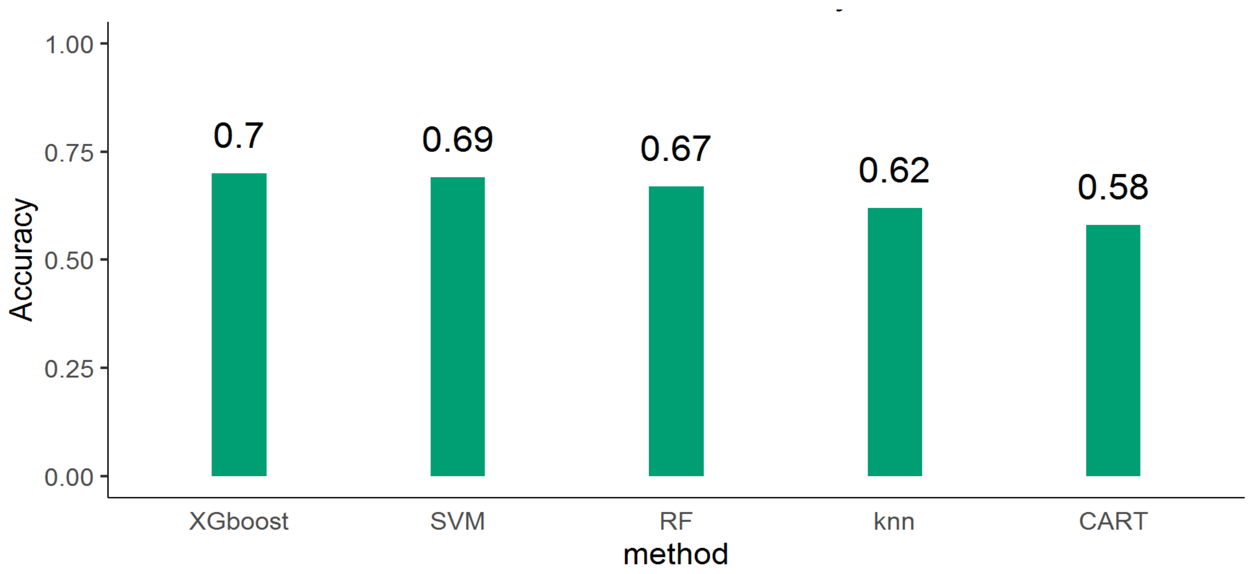
$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x')$$

一句話形容Random Forest

三個臭皮匠

勝過諸葛亮

ML method Accuracy



Let's Party!