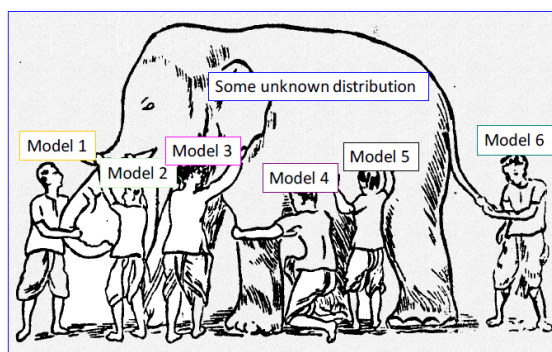


# Ensemble - Bagging, Boosting & Stacking

Ensemble 方法其實就是“團隊合作”好幾個模型一起上的方法。假如我們手上有一堆 Classifier，這些 Classifier 每個都有不同的屬性。隨著 ensemble 規模的增加，分類準確率不斷上升。

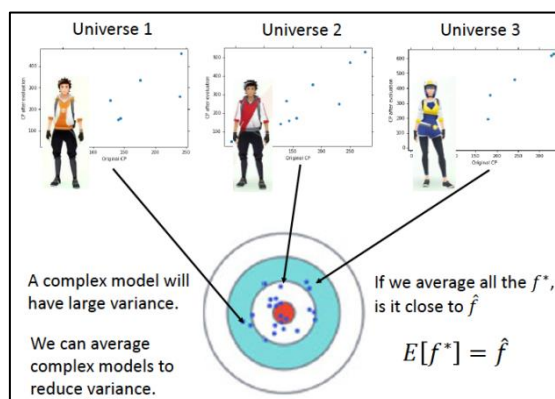


Ensemble gives the global picture!

## A. Bagging

Bias 跟 Variance 是有 trade-off，比較簡單的 model 會有 Bias 比較大，Variance 就比較小，比較複雜的 model 會有 Bias 小 Variance 就比較大。兩者的組合下，error rate 隨著 model 複雜度增加逐漸下降，然後再逐漸上升 (因為 overfitting)。如果不同世界裡面都用很複雜的模型，variance 很大，bias 小，把不同模型的輸出，通通集合起來，做一個平均，得到一個新的模型  $\hat{f}$  可能就會跟正確的模型很接近，Bagging 就是要體現這件事情。

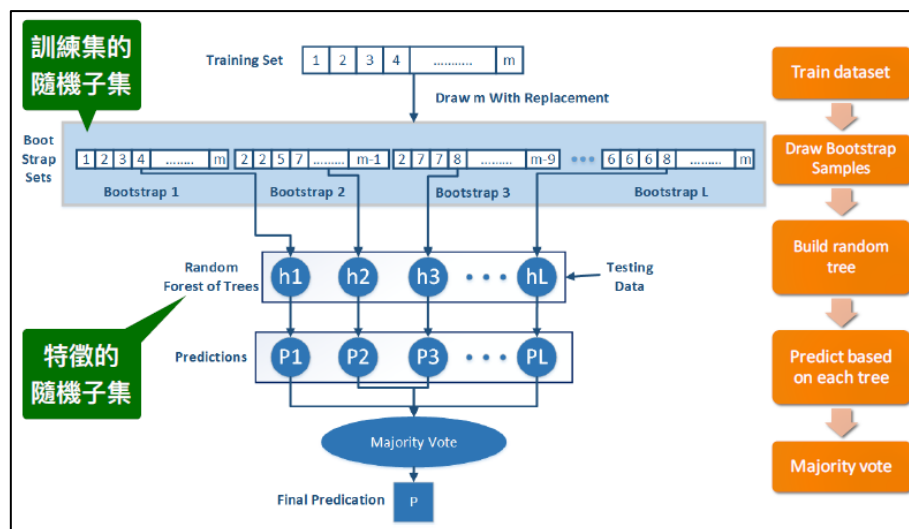
- High variance/Low bias: 高抽樣誤差，但抽樣分配的平均接近母體平均。
- Low variance/High bias: 低抽樣誤差，但抽樣分配的平均離母體平均很遠。



結合具有不同優缺點的預測模型，那些準確預測的模型往往會互相加強，同時抵銷錯誤的預測。



#### 4. 以多數決 (majority voting)的方式匯總所有決策樹的預測



一般在做 train 時會把手上的 label data 切成 training set 跟 validation set 但使用 bagging 的時候，可以不用把 data 切成 training set 跟 validation set 同樣可以擁有 validation 的效果 → 這樣做 Out-of-bag validation：

做 bagging 的時候，訓練出來的 function 都是用部分的 data train 出來的：

f1 只用 x1,x2 train

f2 只用 x3,x4 train

f3 只用 x1,x3 train

f4 只用 x2,x4 train

所以

可以用 f2 跟 f4 bagging 的結果在 x1 資料集上去測試

可以用 f2 跟 f3 bagging 的結果在 x2 資料集上去測試

可以用 f1 跟 f4 bagging 的結果在 x3 資料集上去測試

可以用 f1 跟 f3 bagging 的結果在 x4 資料集上去測試

接下來把 x1 ~ x4 的所有測試結果把它平均計算 error rate，得到 out-of-bag (OOB) error，這個結果可以反應測試的正確率。

### C. Boosting

bagging 是用在很強的 model，boosting 則是用在弱的 model。

Boosting 保證假設你有一個 Model 錯誤率高過 50% 只要能夠做到這件事情，

boosting 可以把這些錯誤率值略高於 50% 的 Model 降到 0%。

核心想法是找多個 classifier f1,f2,f3, ... 如果找到 f2 跟 f1 是互補的，f2 要去做 f1 無法做到的事情，然後再找到 f3 也要跟 f2 互補，這樣一直找一大把後集合起來，就會很強。使用 Boosting 在找 f1, f2, f3 的過程中，是要有順序的，

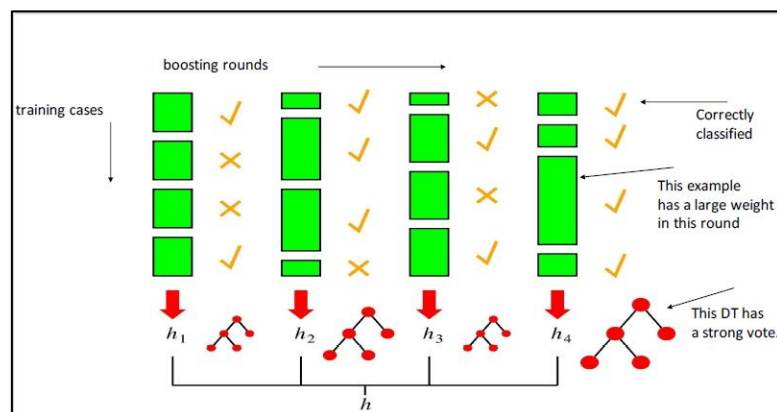
因為每次找的人都要能夠做到上一個人沒辦法辦到的事情。

- 透過 re-weighting 得到不同的 classifier:

每個 classifier 的結構都一樣，差別在於所使用的資料權重不同，導致訓練出多個不同的 classifier。re-weighting 的概念是在每個 classifier 所使用的資料集中對每一筆資料乘上不同的權重，每個資料集中所乘上的權重組合都不同，因而導致有不同的 dataset 供 classifier 訓練，而有不同的 classifier。

## D. AdaBoost

首先先給定  $f_1$  的資料集一組初始的權重，接著要找一組新的 training data 使得讓  $f_1$  在新的 data 上面結果會爛掉，正確率會接近 50%，然後再用這組新的 training data 上面來訓練  $f_2$ 。假設原來的 training data weight 是  $u_1$  新的 training data weight 是  $u_2$ ，我們目標是把  $u_1$  換成  $u_2$ ，讓  $f_1$  的 error rate  $\epsilon_{f_1} = 0.5$ ，使得  $f_1$  的表現看起來就像是隨機的一樣。得到  $u_2$  後，拿  $u_2$  作為 weight data 然後拿這樣的 dataset 去訓練  $f_2$ ，這樣訓練出來的  $f_2$  就會補足  $f_1$  的弱點。



變化權重的方法是，如果  $x_n$  被  $f_1$  分類錯誤，那我們就把  $x_n$  乘上一個權重  $d_1$  其中  $d_1 > 1$ ；如果  $x_n$  被  $f_1$  分類正確，那我們就把  $x_n$  除上一個權重  $d_1$ ， $d_1 > 1$ 。而  $d_1 = \sqrt{(1-\epsilon_{f_1})/\epsilon_{f_1}}$ ， $\epsilon_{f_1}$  為  $f_1$  在 training data 上的 error rate。

**Re-weighting Training Data**

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \quad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5 \quad \begin{array}{l} f_1(x^n) \neq \hat{y}^n \quad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \\ f_1(x^n) = \hat{y}^n \quad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \end{array}$$

$$= \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 = \sum_{f_1(x^n) \neq \hat{y}^n} u_2^n + \sum_{f_1(x^n) = \hat{y}^n} u_2^n$$

$$= \sum_n u_2^n = \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1$$

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 2$$

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \quad Z_1 = \sum_n u_1^n$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5 \quad \begin{array}{l} f_1(x^n) \neq \hat{y}^n \quad u_2^n \leftarrow u_1^n \text{ multiplying } d_1 \\ f_1(x^n) = \hat{y}^n \quad u_2^n \leftarrow u_1^n \text{ divided by } d_1 \end{array}$$

$$\frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 2 \quad \frac{\sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1} = 1$$

$$\sum_{f_1(x^n) = \hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n d_1 \quad \frac{1}{d_1} \sum_{f_1(x^n) = \hat{y}^n} u_1^n = d_1 \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n$$

$$\varepsilon_1 = \frac{\sum_{f_1(x^n) \neq \hat{y}^n} u_1^n}{Z_1} \quad \sum_{f_1(x^n) \neq \hat{y}^n} u_1^n = Z_1 \varepsilon_1 \quad \begin{array}{l} Z_1(1 - \varepsilon_1) / d_1 = Z_1 \varepsilon_1 d_1 \\ d_1 = \sqrt{(1 - \varepsilon_1) / \varepsilon_1} > 1 \end{array}$$

training data 一開始初始的 weight 都是 1，接著跑 T 次 iteration

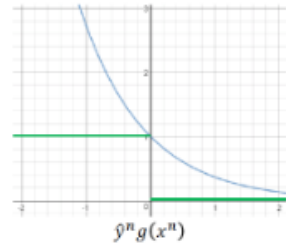
每一筆 data 都有一個 weight 值 u；每一次的 iteration 會計算出一個 epsilon\_t 然後根據 epsilon\_t 的值來更新 weight，因為  $dt = \sqrt{(1 - \text{epsilon}_t) / \text{epsilon}_t}$ 。另一種表示法，定義  $\alpha_t = \ln(dt) = \ln(\sqrt{(1 - \text{epsilon}_t) / \text{epsilon}_t})$ ，有了  $\alpha_t$  就可以把式子變得更簡便一點：

$$u_{t+1}^n \leftarrow u_t^n \times \exp(-\hat{y}^n f_t(x^n) \alpha_t)$$

但是 T 個 Classifiers 當中的表現有好有壞，所以在每一個 classifier 的結果前面

再乘以一個權重  $\alpha_t$  這樣下去整合結果會更好， $\alpha_t$  為該 classifier 的 epsilon 所計算出來。其中如果某個 classifier 錯誤率  $\text{epsilon}_t = 0.1$  可以計算出  $\alpha_t = 1.10$  如果另一個 classifier 錯誤率  $\text{epsilon}_t = 0.4$  計算出來的  $\alpha_t = 0.20$ ，所以 classifier 的錯誤率越小，分辨的比較準的時候  $\alpha_t$  就會比較大，因而達到 voting 的效果。

AdaBoost 跑的 iteration 越多，performance 越好，原因在於其 loss function 的上界會隨著 iteration 的增加而下降，因此推論隨著 iteration 的增加 Adaboost 的 loss 會下降。

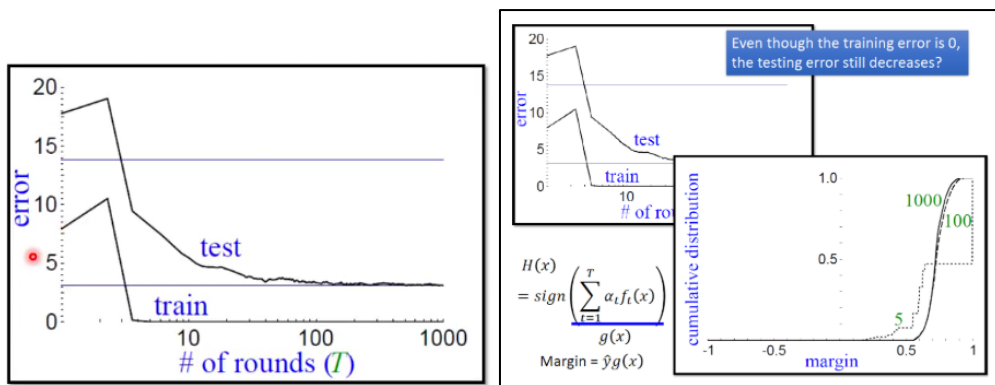
<p style="color: red; text-align: center;">Warning of Math</p> $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t f_t(x)\right) \quad \alpha_t = \ln\sqrt{(1-\epsilon_t)/\epsilon_t}$ <p style="color: blue;">As we have more and more <math>f_t</math> (T increases), <math>H(x)</math> achieves smaller and smaller error rate on training data.</p>	<p style="text-align: center;">Error Rate of Final Classifier</p> <ul style="list-style-type: none"> <li>Final classifier: <math>H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t f_t(x)\right) g(x)</math></li> <li><math>\alpha_t = \ln\sqrt{(1-\epsilon_t)/\epsilon_t}</math></li> </ul> <p>Training Data Error Rate</p> $= \frac{1}{N} \sum_n \delta(H(x^n) \neq \hat{y}^n)$ $= \frac{1}{N} \sum_n \delta(\hat{y}^n g(x^n) < 0)$ $\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n))$  <p>假設我們有 N 筆 data</p>
<p>Training Data Error Rate</p> $\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n)) = \frac{1}{N} Z_{T+1}$ <p><math>Z_t</math>: the summation of the weights of training data for training <math>f_t</math></p> <p>What is <math>Z_{T+1}</math>? <math>Z_{T+1} = \sum_n u_{T+1}^n</math></p> $u_1^n = 1$ $u_{t+1}^n = u_t^n \times \exp(-\hat{y}^n f_t(x^n) \alpha_t) \quad \left. \vphantom{u_{t+1}^n} \right\} u_{T+1}^n = \prod_{t=1}^T \exp(-\hat{y}^n f_t(x^n) \alpha_t)$ $Z_{T+1} = \sum_n \prod_{t=1}^T \exp(-\hat{y}^n f_t(x^n) \alpha_t)$ $= \sum_n \exp\left(-\hat{y}^n \sum_{t=1}^T f_t(x^n) \alpha_t\right) = \sum_n \exp\left(-\hat{y}^n g(x^n)\right)$	<p>Training Data Error Rate</p> $\leq \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x^n)) = \frac{1}{N} Z_{T+1}$ <p><math>g(x) = \sum_{t=1}^T \alpha_t f_t(x)</math></p> <p><math>\alpha_t = \ln\sqrt{(1-\epsilon_t)/\epsilon_t}</math></p> <p><math>Z_1 = N</math> (equal weights)</p> $Z_t = Z_{t-1} \epsilon_t \exp(\alpha_t) + Z_{t-1} (1-\epsilon_t) \exp(-\alpha_t)$ <p>Misclassified portion in <math>Z_{t-1}</math>      Correctly classified portion in <math>Z_{t-1}</math></p> $= Z_{t-1} \epsilon_t \sqrt{(1-\epsilon_t)/\epsilon_t} + Z_{t-1} (1-\epsilon_t) \sqrt{\epsilon_t/(1-\epsilon_t)}$ $= Z_{t-1} \times 2\sqrt{\epsilon_t(1-\epsilon_t)}$ $Z_{T+1} = N \prod_{t=1}^T 2\sqrt{\epsilon_t(1-\epsilon_t)}$ <p>Training Data Error Rate <math>\leq \prod_{t=1}^T \frac{2\sqrt{\epsilon_t(1-\epsilon_t)}}{1} &lt; 1</math> <span style="background-color: orange; padding: 2px;">Smaller and smaller</span></p>

我們發現 training data 的 error rate 在五個 weak classifier 合力下就會變 0 了

加了更多的 weak classifier 後，training data error 已經不會再下降，但是 testing error 居然還能持續下降。原因在於 Adaboost 的 margin 隨著 iteration 增加而持續變大，導致 error 還能一直下降。(又或是 error 的 upper bound 隨 iteration 增加會持續下降，導致 error 也能持續下降)

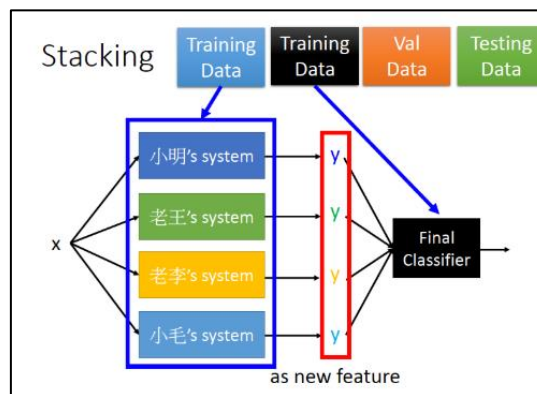
$g(x)$ : 代表 weak classifiers 整合後的 output 我們希望  $g(x)$  是個非常大的值  
定義:  $g(x) * \hat{y}$  為 Margin 也就是使用整合後的分類器，分類正確的情形。  
Adaboost 的 (upper bound) Margin 變化圖，可以發現，只有 5 個 weak classifier 時，margin 是虛線，但是增加 weak classifier 的數量時，增加到 1000 個 classifiers 時我們發現 Margin 會增加，如圖中的黑色實線，雖然在 training set 的 error rate 已經是 0 不會再下降，也就是  $\hat{y}$  已經跟所有的  $g(x)$  同號了，但是增加 classifier 數量時，仍然可以讓 margin 增加，使得在 testing 的 error rate 下降。





## E. Stacking

類似於 voting 的方法，差別在於用一個 classifier 決定最後權重結果。其中會把 training data 切成兩部分一部分拿來 train 前面的 classifier，另一部分拿來 train 後面的 Final Classifier。



## 參考資料

1. ML Lecture 22: Ensemble, Hung-yi Lee.  
<https://www.youtube.com/watch?v=tH9FH1DH5n0>
2. [ML 筆記] Ensemble - Bagging, Boosting & Stacking  
<http://violin-tao.blogspot.com/2018/01/ml-ensemble.html>
3. 機器學習: Ensemble learning 之 Bagging、Boosting 和 AdaBoost  
<https://medium.com/@chih.sheng.huang821/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-ensemble-learning%E4%B9%8Bbagging-boosting%E5%92%8Cadaboost-af031229ebc3>
4. Class Handout, Lee, Chia Jung professor, MDM64001, School of Big Data Management, Soochow University