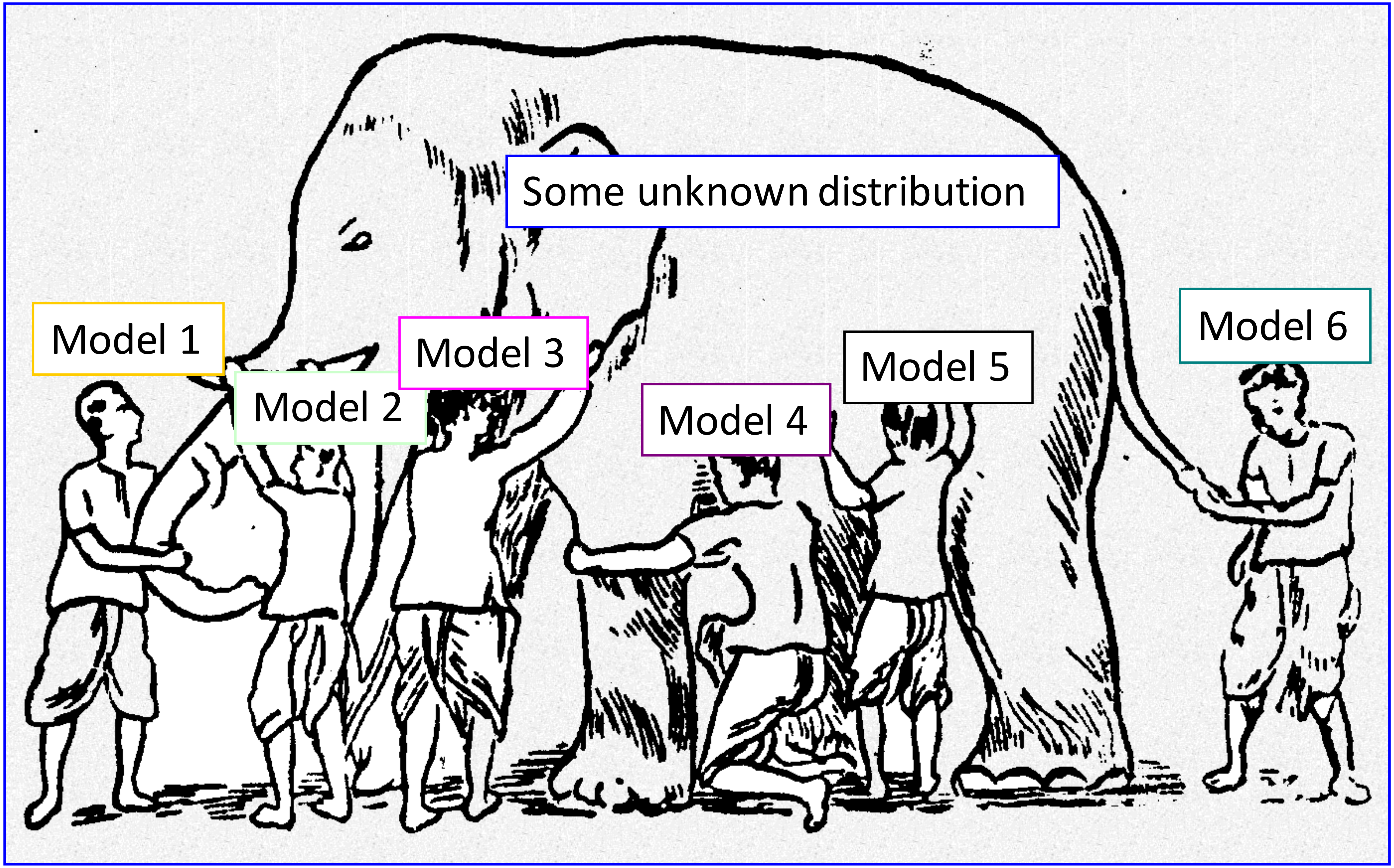
機器學習

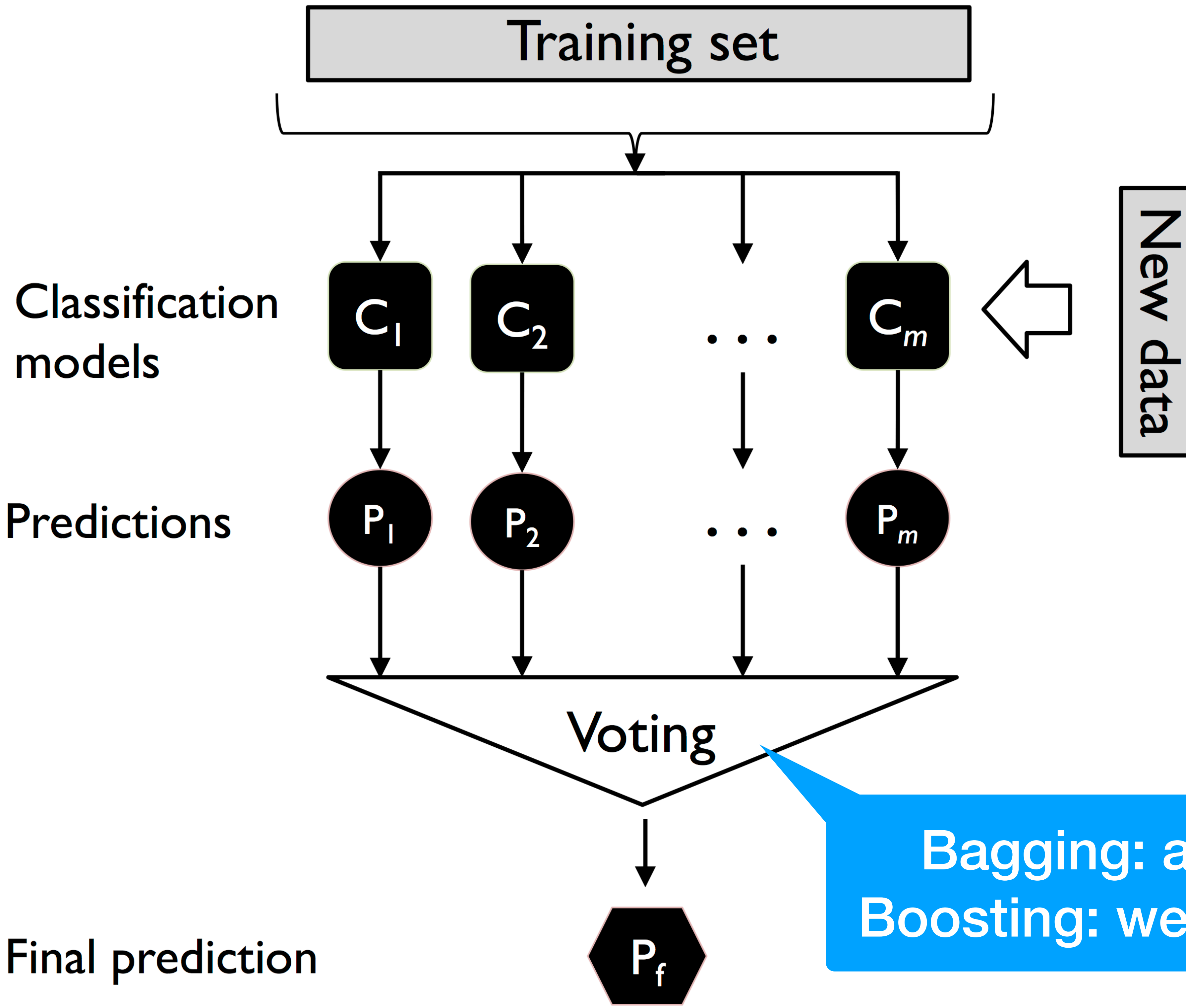
Lecture 7　集成方法

Why Ensemble Works?



Ensemble gives the global picture!

Ensemble Methods (集成)

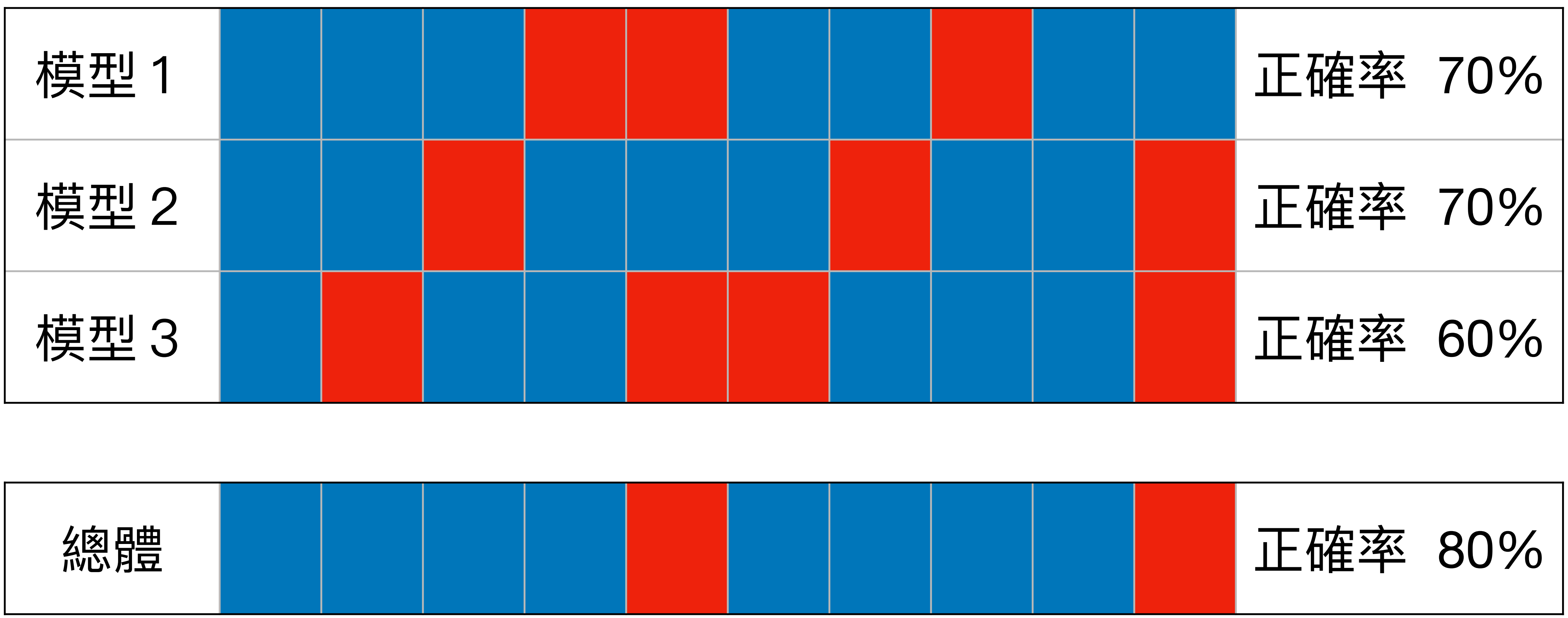


Bagging: averaging
Boosting: weighted vote

Sebastian Raschka, Vahid Mirjalili, Python Machine Learning, 2/e

Why Ensemble Works?

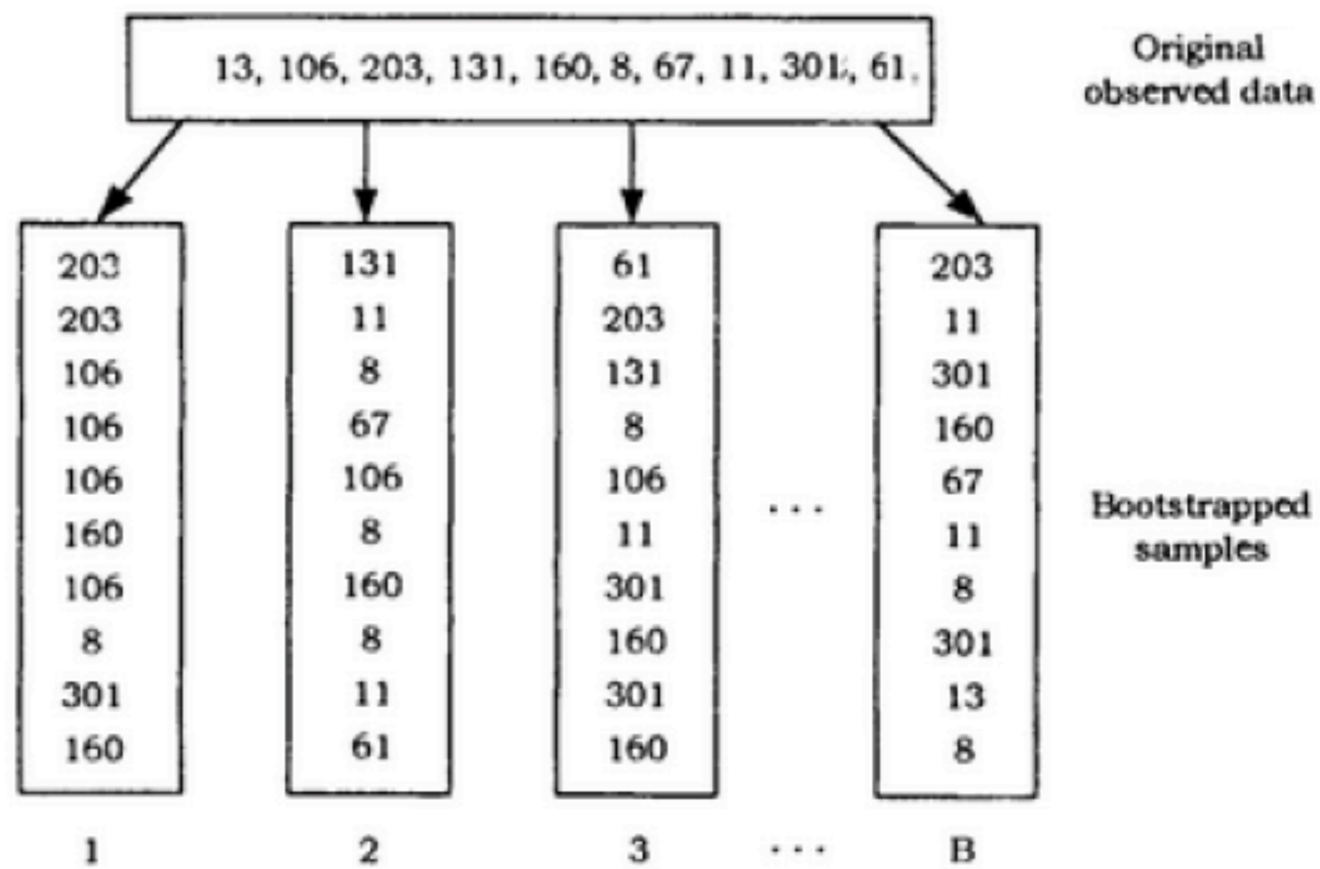
結合具有不同優缺點的預測模型，那些準確預測的模型往往會互相加強，同時抵銷錯誤的預測



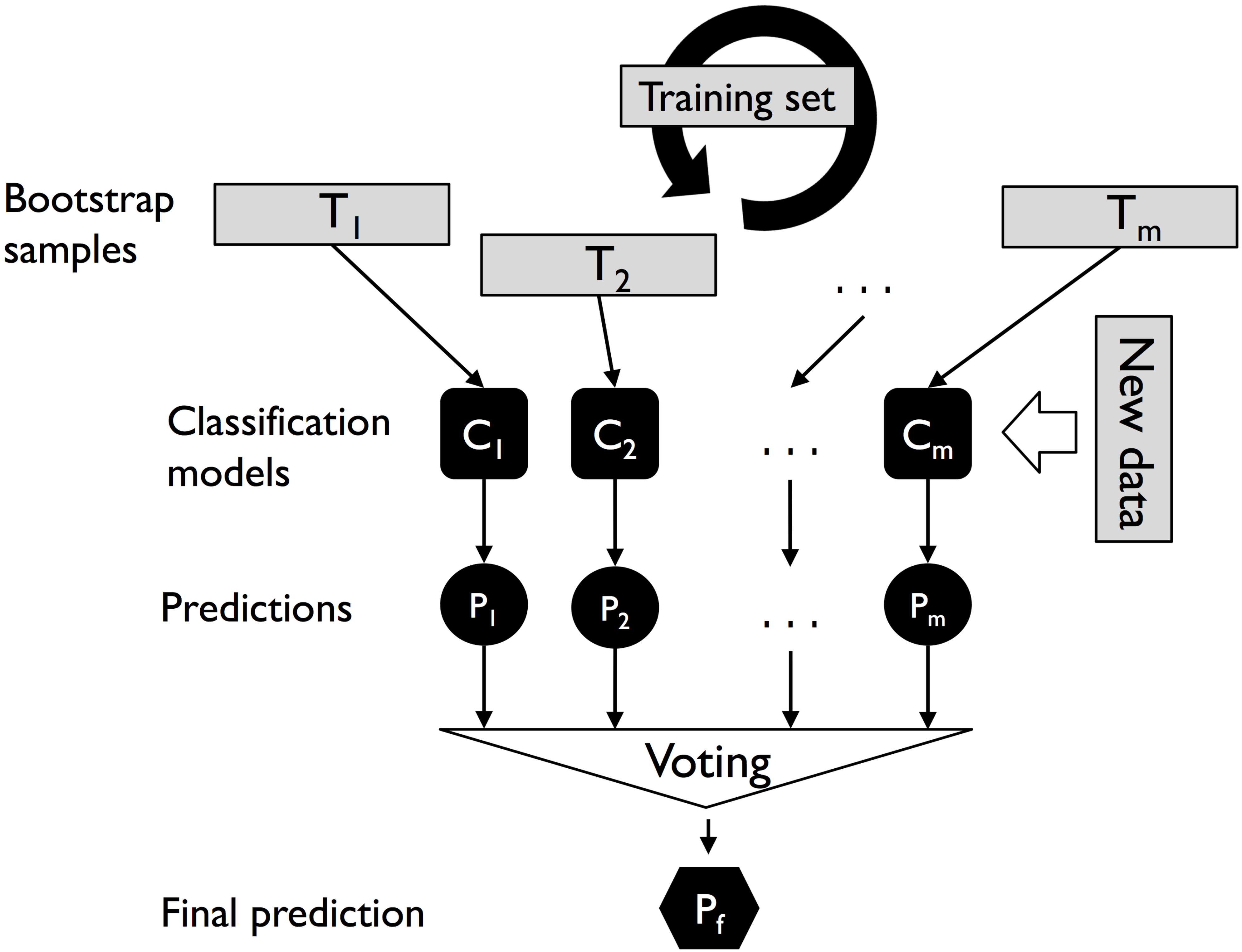
前提：總體所包含的模型不能有同樣的錯誤，亦即模型必須是不相關的

Bootstrap Sample (自助樣本)

- 假設有 n 個原始樣本的觀察值，Bootstrap的執行步驟如下：
  1. 抽取一個觀察值，記下其值後放回原始樣本集合中混合均勻，再重新抽取
  2. 重覆步驟 1 n 次，就可以得到一組Bootstrap的訓練樣本集合
  3. 重覆步驟 1 和 2 B 次，就可以得到 B 組Bootstrap的訓練樣本集合
- 假設有 10 個原始樣本觀察值 X = {13, 106, 203, 131, 160, 8, 67, 11, 301, 61}：
  - 每一組Bootstrap訓練樣本有10個抽取值。未被抽取到的觀察值則設為測試樣本，用以測試訓練樣本所建構之模型的正確率。
  - B組Bootstrap的樣本集合



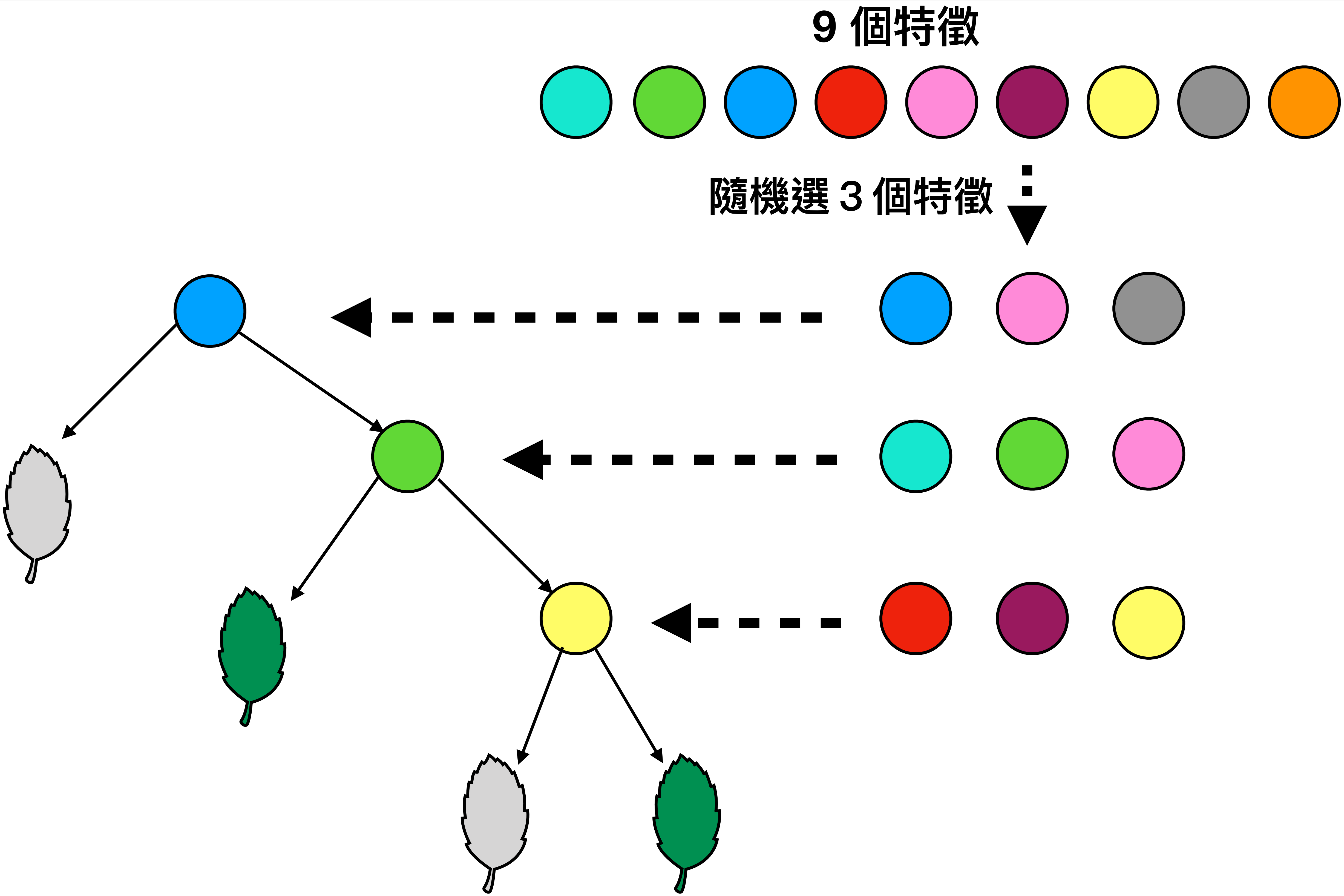
Bagging


Training set
Bootstrap samples
T_1
T_2
...
T_m
New data
Classification models
C_1
C_2
...
C_m
Predictions
P_1
P_2
...
P_m
Voting
Final prediction
P_f

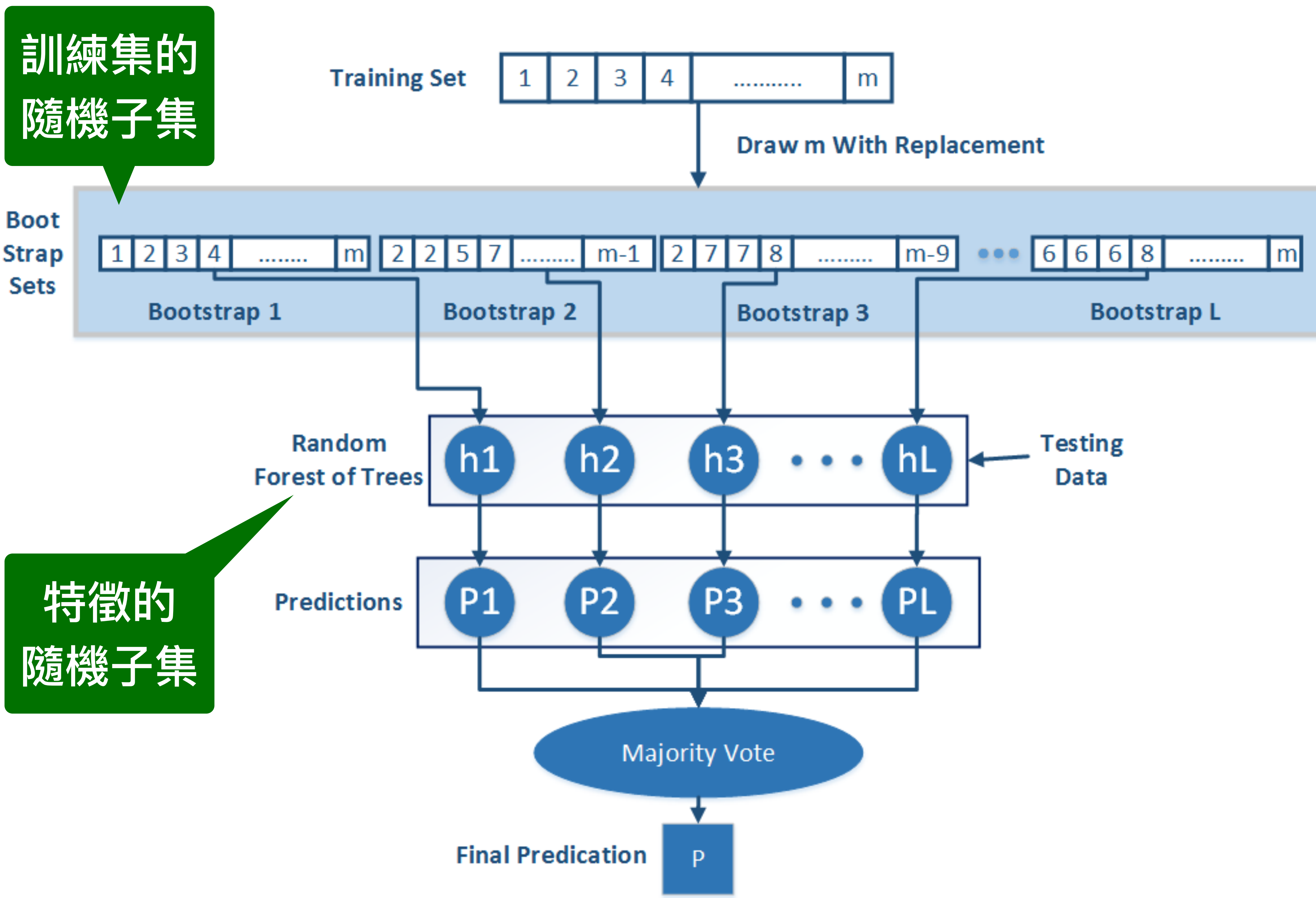
Random Forest （隨機森林）

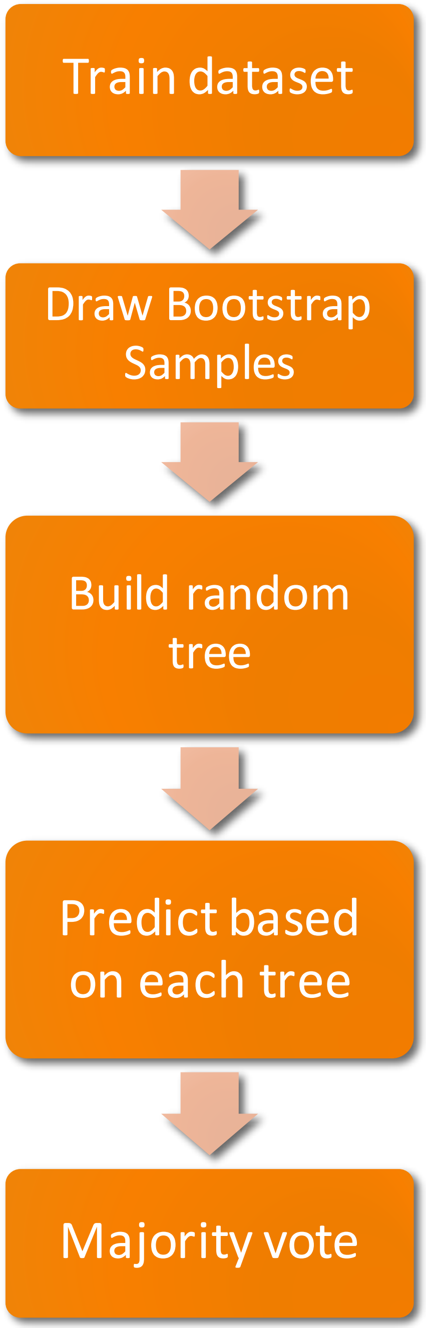
1. 定義大小為 n 的隨機(取出後放回)自助樣本 (bootstrap sample)
2. 從自助樣本中導出決策樹。對每一節點：
   (1) 隨機(取出不放回)選擇 d 個特徵
   (2) 使用特徵分割該節點，依據『目標函數』找出最佳方式，如最大化『資訊增益』
3. 重複 k 次步驟 1 和 2
4. 以**多數決 (majority voting)**的方式匯總所有決策樹的預測

Random Forest



Random Forest

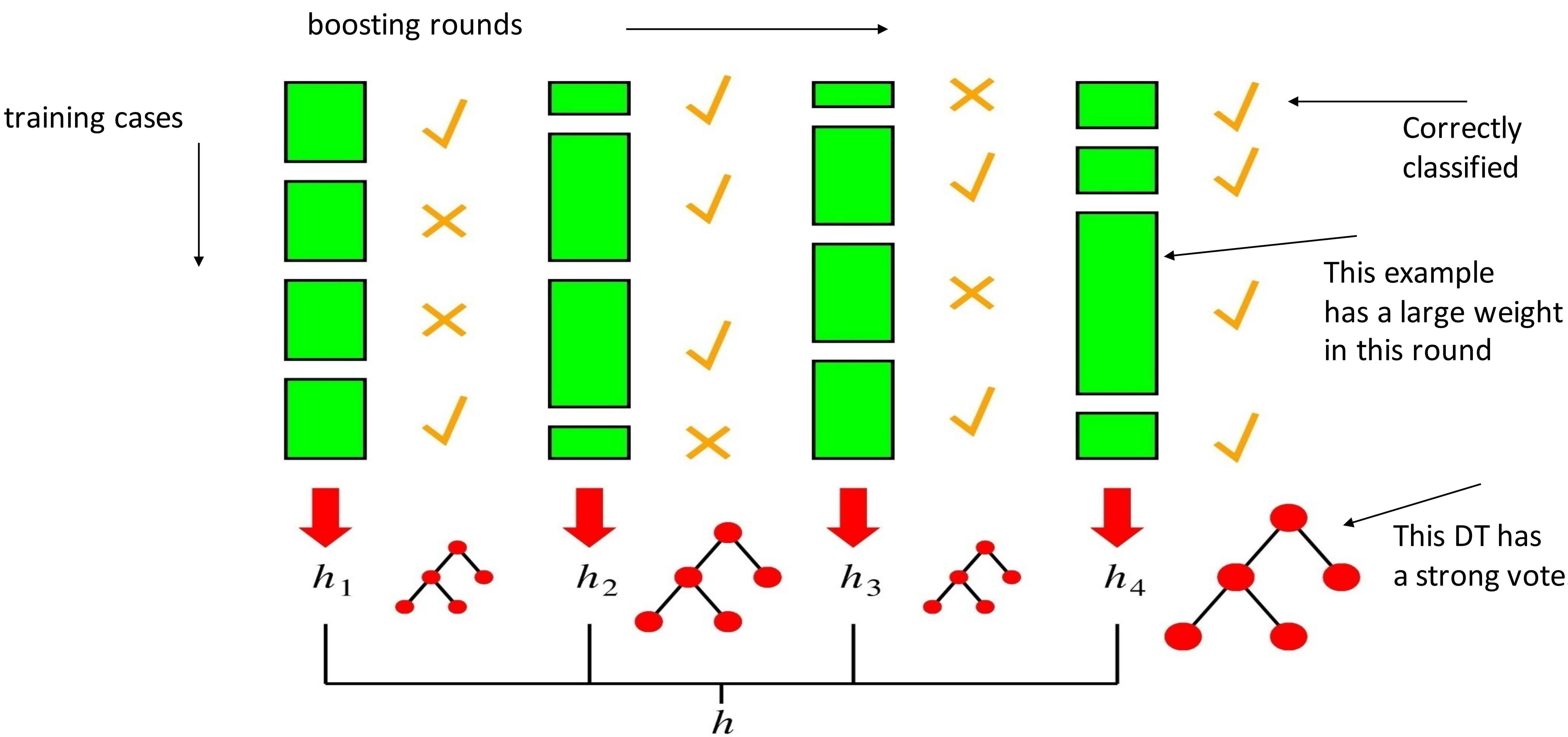




Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Records that are wrongly classified will have their weights increased
  - Records that are classified correctly will have their weights decreased

Boosting



boosting rounds
training cases
Correctly
classified
This example
has a large weight
in this round
This DT has
a strong vote.
h_1
h_2
h_3
h_4
h

AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.
Initialize: $\boxed{D_1(i) = 1/m \text{ for } i = 1, \ldots, m.}$ ← Initial Distribution of Data
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$. ← Train model
- Aim: select h_t with low weighted error:

$$\boxed{\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].}$$ ← Error of model

- Choose $\boxed{\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right).}$ ← Coefficient of model
- Update, for $i = 1, \ldots, m$:

$$\boxed{D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}}$$ ← Update Distribution

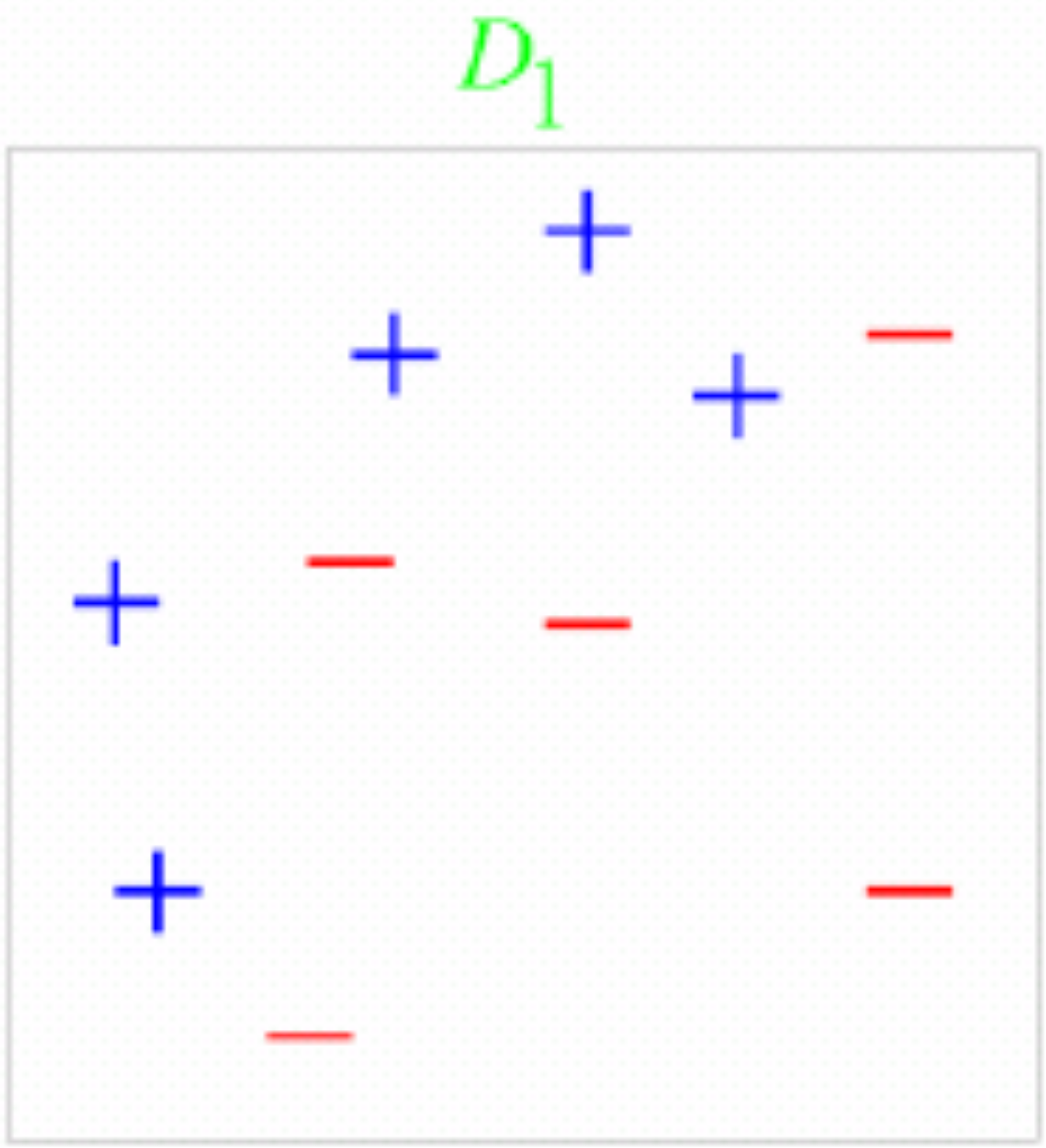
where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$\boxed{H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).}$$ ← Final average

https://www.cs.princeton.edu/~schapire/papers/explaining-adaboost.pdf

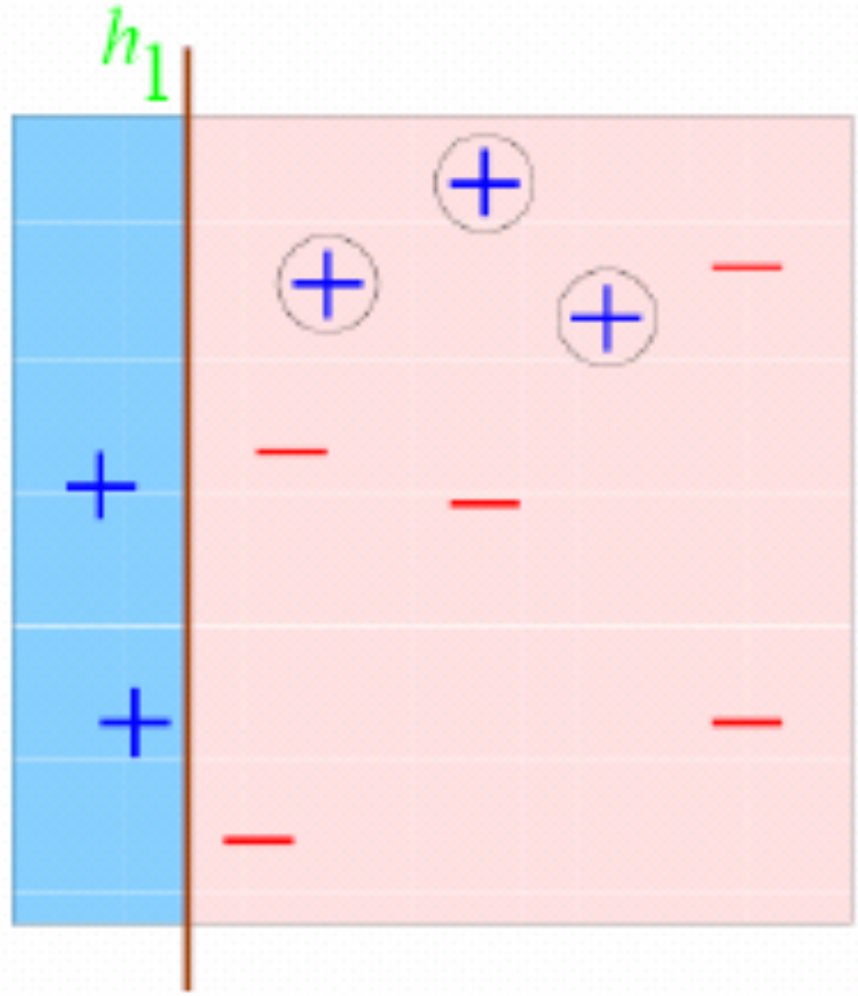
AdaBoost: Example



Training set: 10 points (represented by + or -)

Original Status: Equal Weights

AdaBoost: Example



$$\epsilon = 3/10 = 0.3$$

$$\alpha = 0.5 \cdot \ln \frac{1-0.3}{0.3} \approx 0.42$$

Round 1: Three "+" points are not correctly classified; They are given higher weights

AdaBoost: Example

預測正確 $\Rightarrow y \times h(x) > 0$
$\Rightarrow$ 降低權重

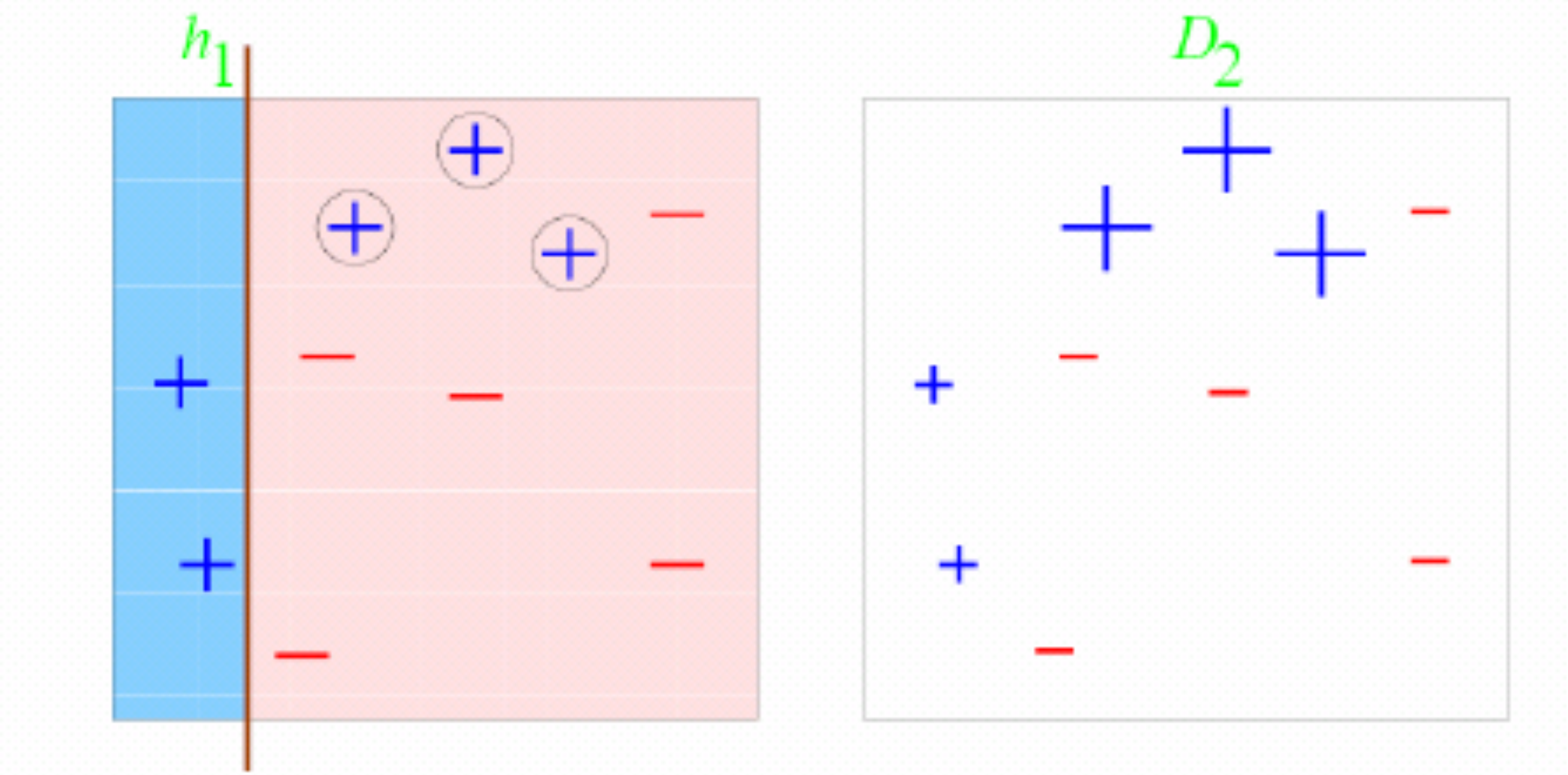
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t \boxed{y_i h_t(x_i)})}{Z_t}$$

$x^{(i)}$	$y^{(i)}$	weights	$h_1(x)$	correct?	Updated weights
1	1	0.1	1	Y	0.072
2	1	0.1	1	Y	0.072
3	-1	0.1	-1	Y	0.072
4	-1	0.1	-1	Y	0.072
5	1	0.1	-1	N	0.167
6	1	0.1	-1	N	0.167
7	-1	0.1	-1	Y	0.072
8	1	0.1	-1	N	0.167
9	-1	0.1	-1	Y	0.072
10	-1	0.1	-1	Y	0.072

分子：0.1*exp(−0.42)

分子：0.1*exp(−0.42*−1)

AdaBoost: Example

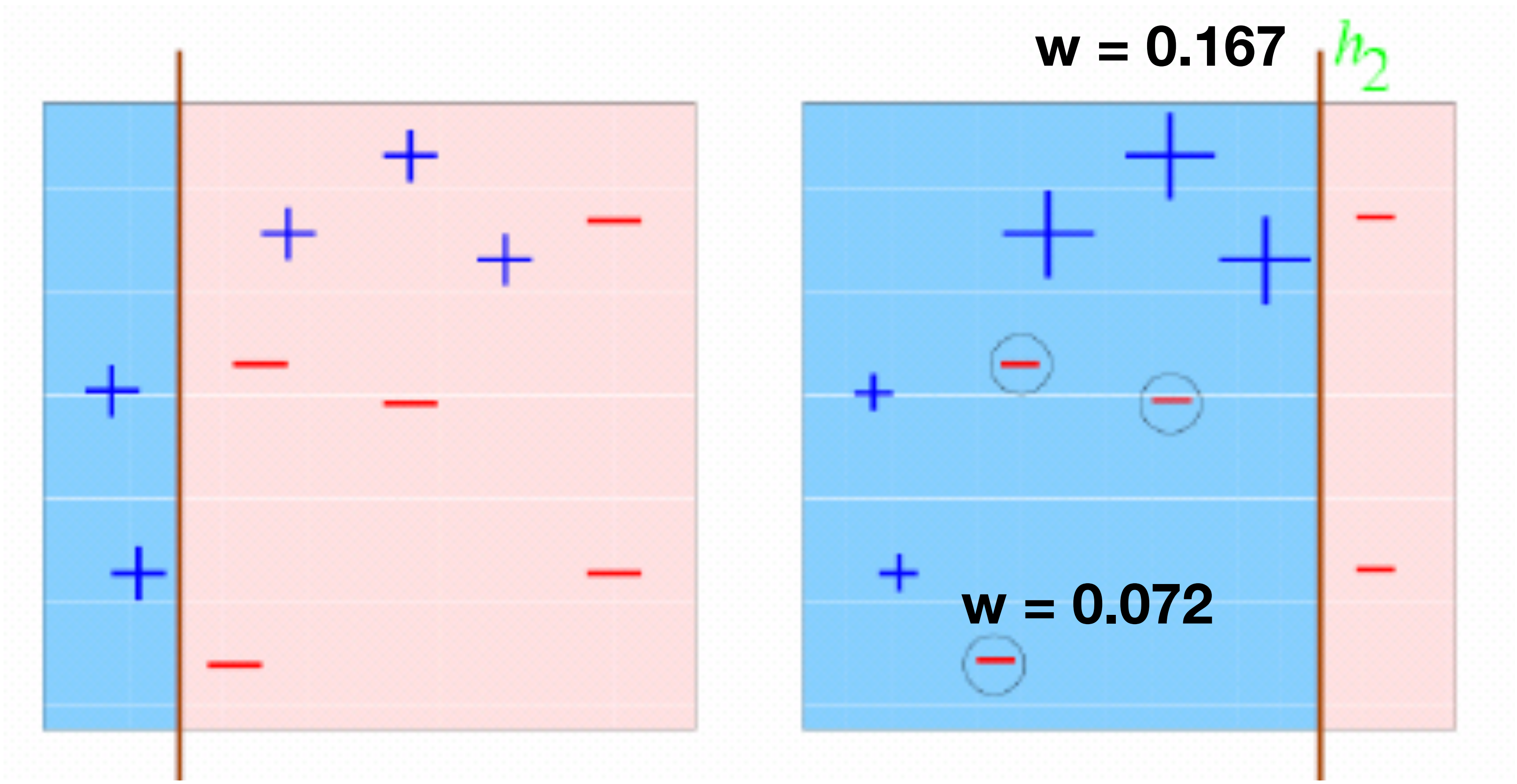


$$\epsilon = 3/10 = 0.3$$

$$\alpha = 0.5 \cdot \ln \frac{1 - 0.3}{0.3} \approx 0.42$$

Round 1: Three "+" points are not correctly classified; They are given higher weights

AdaBoost: Example

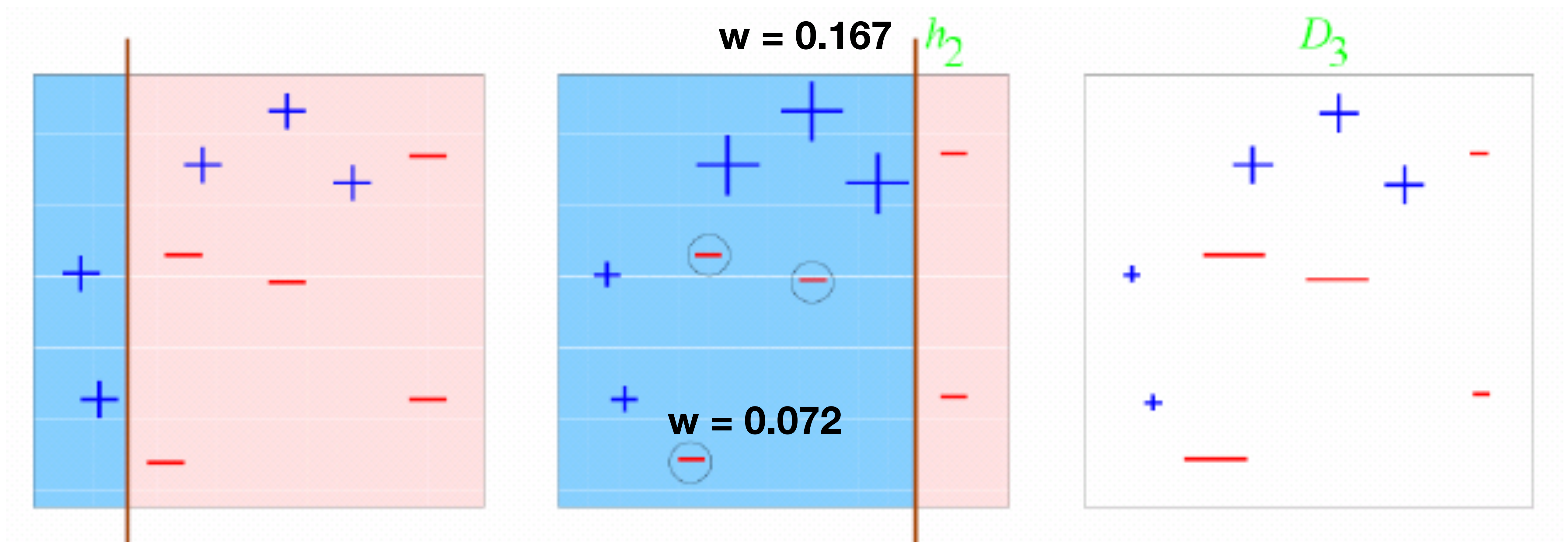


$$\epsilon = 3 * 0.072 = 0.216$$

$$\alpha \approx 0.65$$

Round 2: Three "-" points are not correctly classified; They are given higher weights

AdaBoost: Example

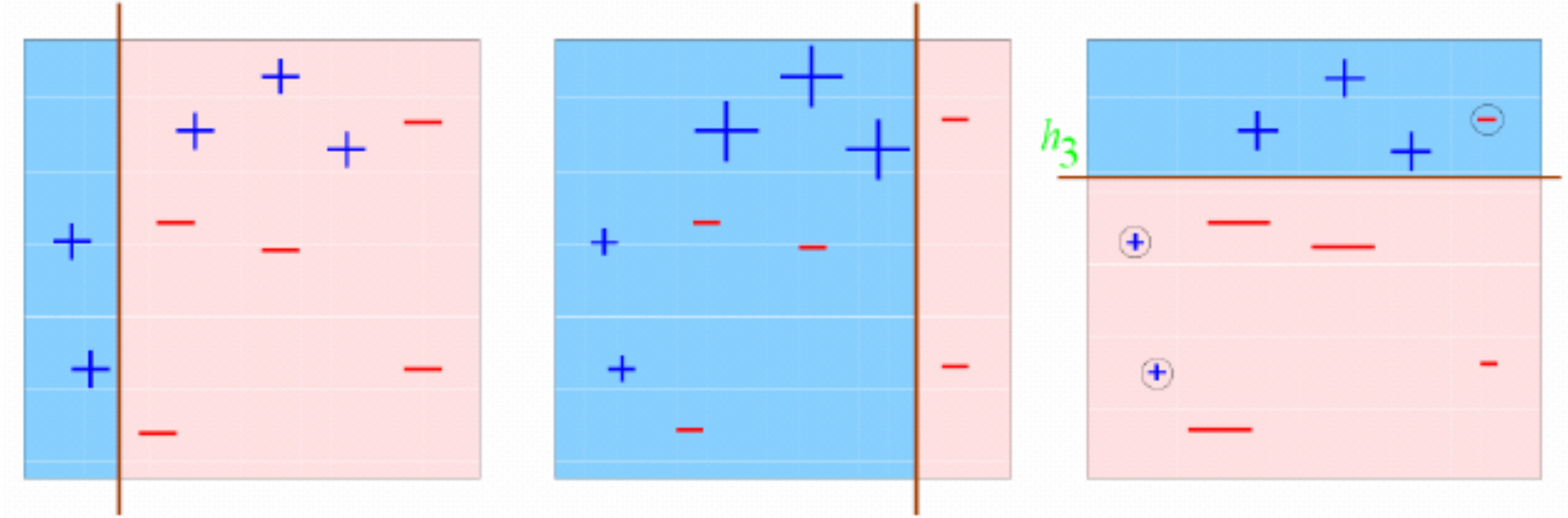


$$\epsilon = 3 * 0.072 = 0.216$$

$$\alpha \approx 0.65$$

Round 2: Three "-" points are not correctly classified; They are given higher weights

AdaBoost: Example

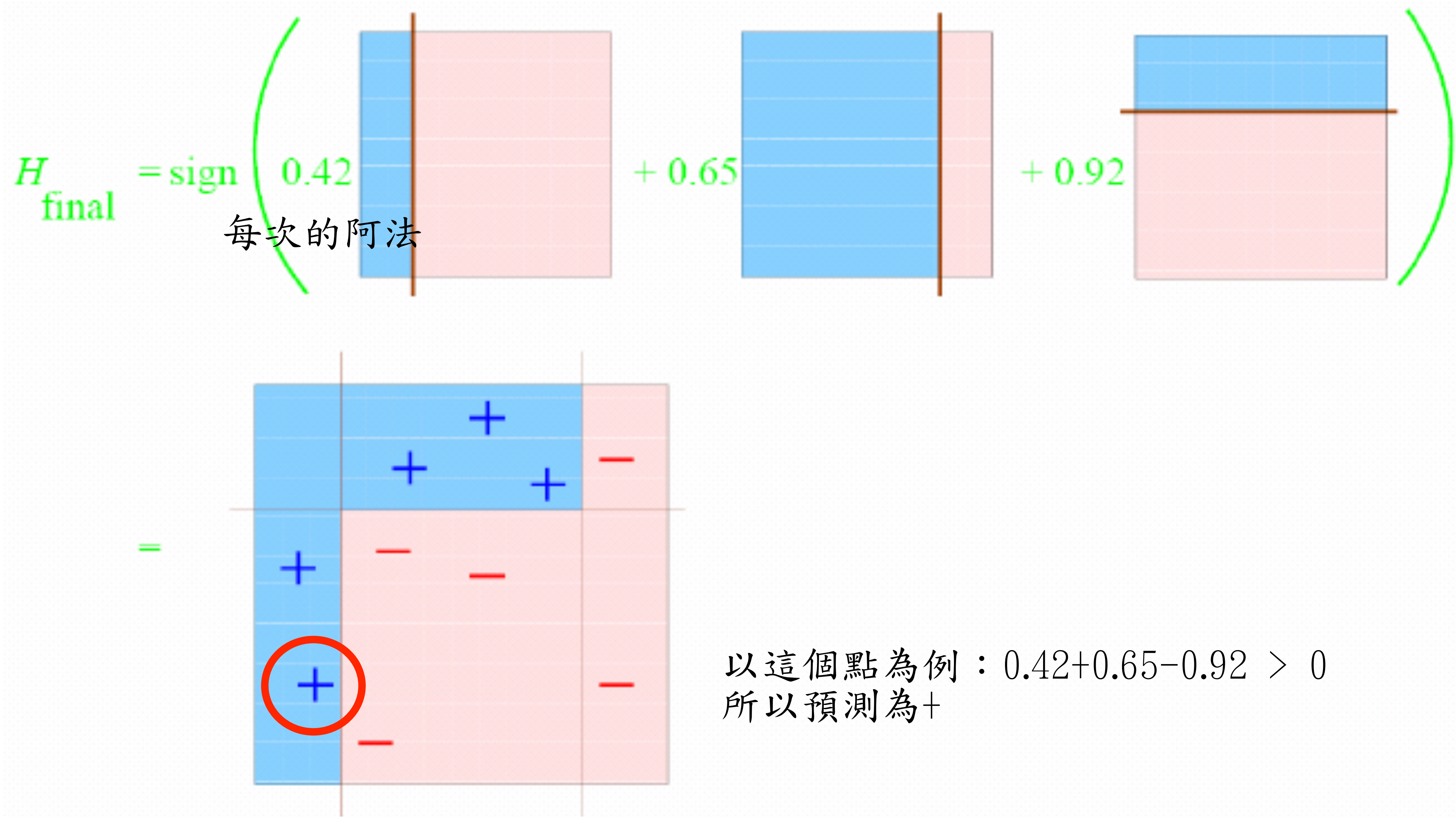


$\epsilon = 0.14$

$\alpha \approx 0.92$

Round 3: One "-" and two "+" points are not correctly classified; They are given higher weights

AdaBoost: Example



Final Classifier: integrate the three “weak” classifiers and obtain a final strong classifier

http://www.authorstream.com/Presentation/aSGuest79199-727683-animated-adaboost-example/

XGBoost

- Start from constant prediction, add a new function each time

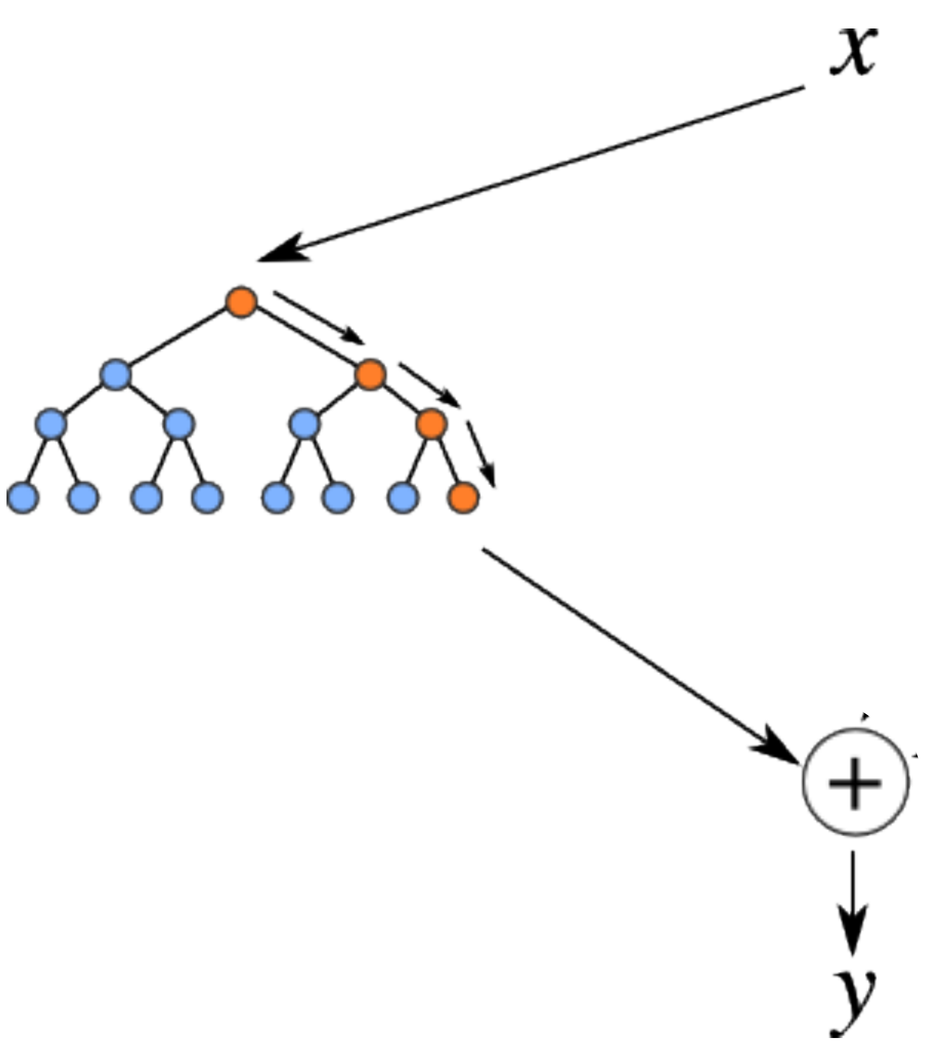
$$\hat{y}_i^{(0)} = 0$$

$$y = 0$$

XGBoost: Example

- Start from constant prediction, add a new function each time

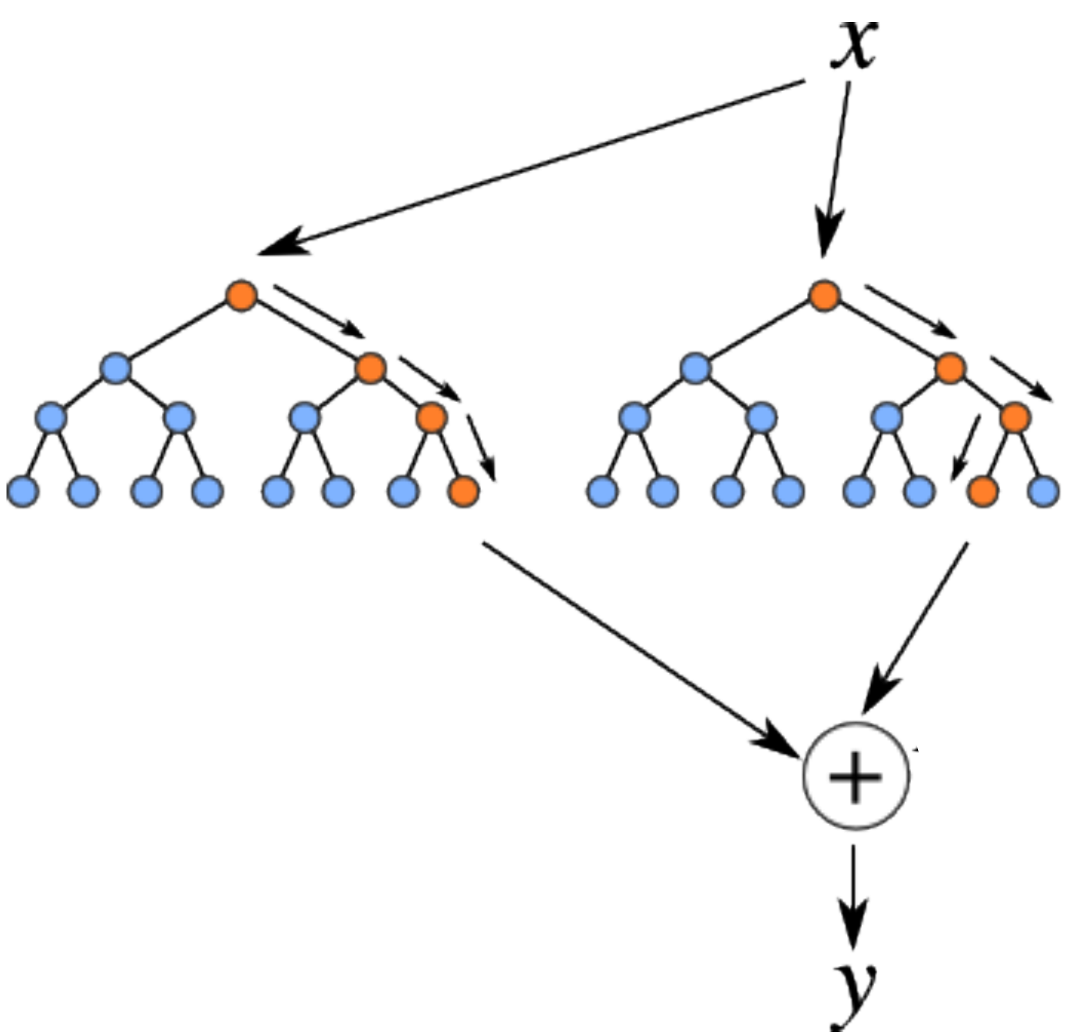
$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)
\end{aligned}
$$



XGBoost: Example

- Start from constant prediction, add a new function each time

$$\hat{y}_i^{(0)} = 0$$
$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$



XGBoost: Example

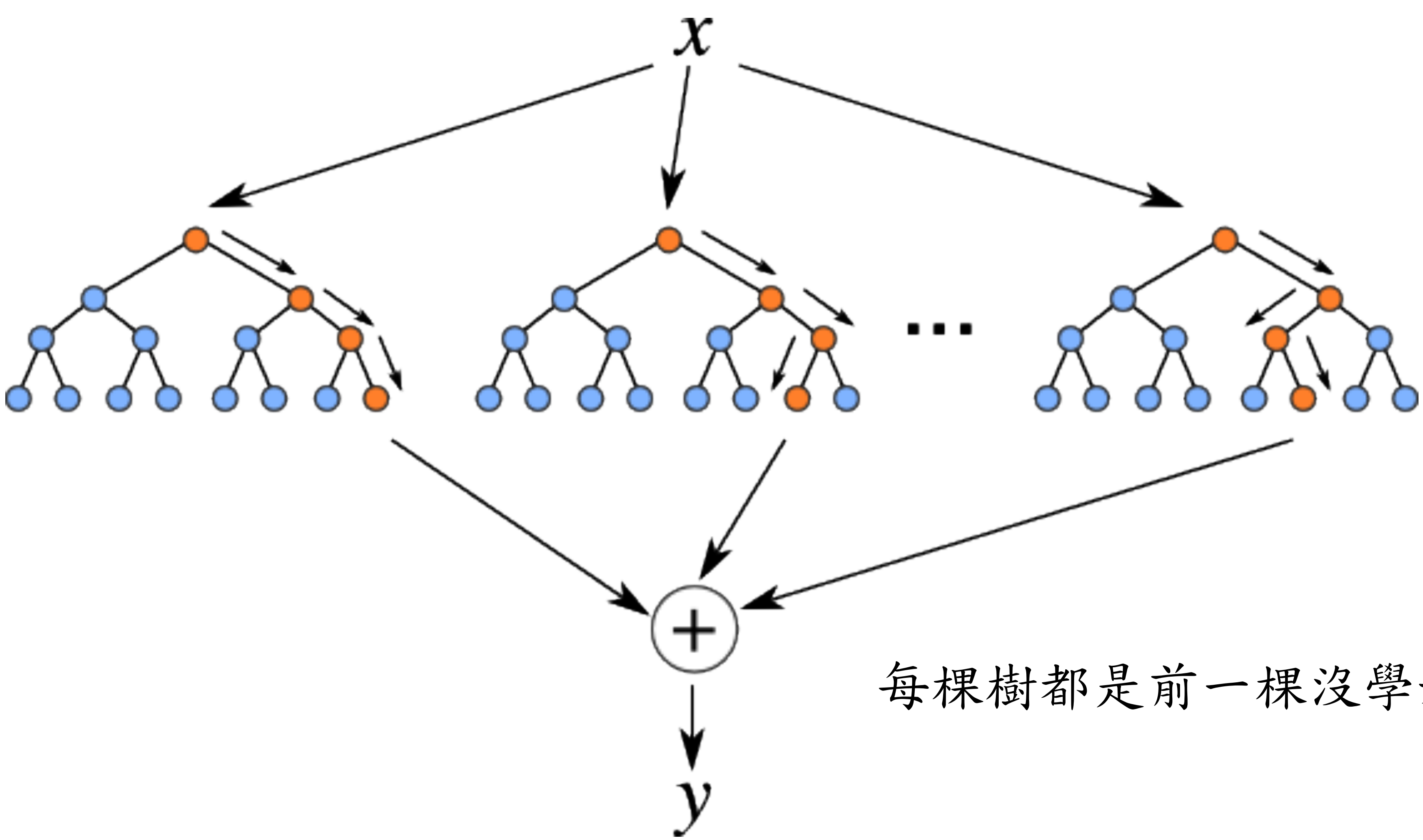
- Start from constant prediction, add a new function each time

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

New function

Model at training round t

Keep functions added in previous round



每棵樹都是前一棵沒學好的地方改進的進化版

XGBoost: Example

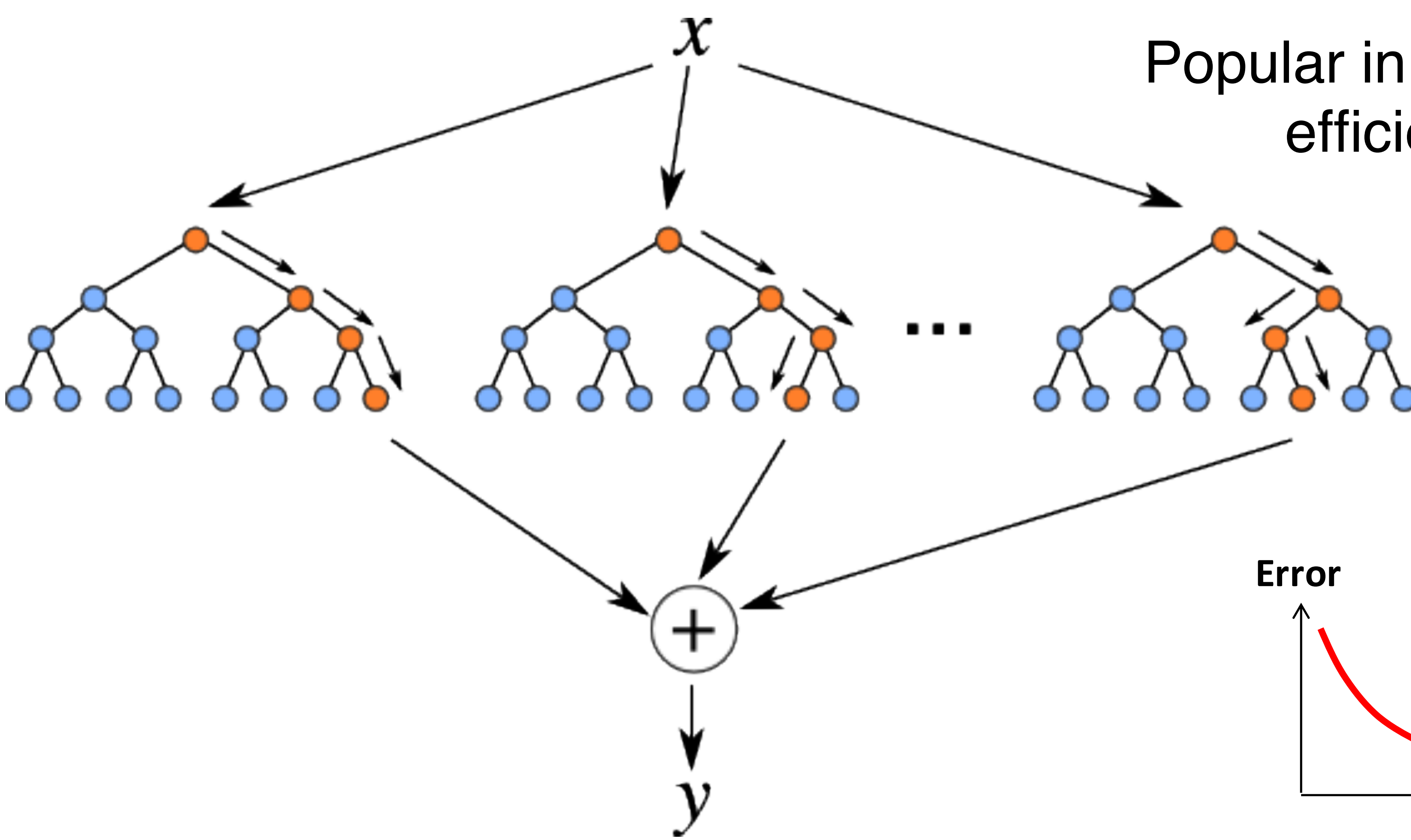
- Start from constant prediction, add a new function each time

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

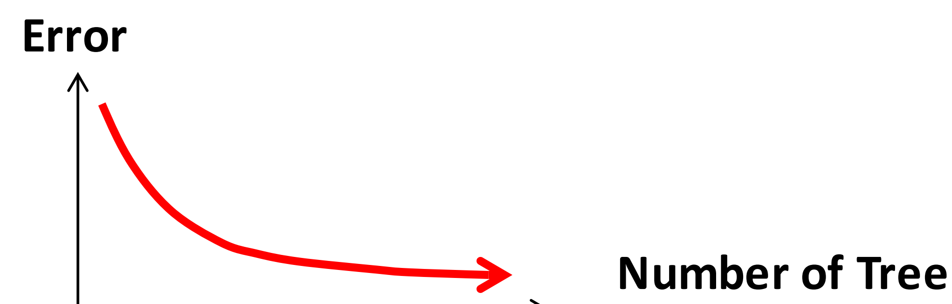
New function

Model at training round t

Keep functions added in previous round



Popular in Kaggle competitions for efficiency and accuracy



Pros & Cons

- Pros
  - Decision tree 不要求對資料進行預處理
  - random forest 可以平行處理 （參數n_job 設為CPU的個數，若不確定電腦CPU個數，可設n_job = -1）
  - Adaboost 考慮每個分類器的權重
  - 集成方法結合不同的分類模型，以抵消各自的弱點，形成一個穩定且表現良好的模型

Pros & Cons

- Cons
  - 集成方法會增加計算的複雜度，花費更多的計算成本。
  - 使用集成方法時要權衡「計算複雜度」和「效能改進」

Python code

- 訓練「隨機森林」模型

```
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier(criterion='entropy',
                                n_estimators=25,
                                random_state=1)
forest.fit(X_train, y_train)
```

隨機森林裡有
25棵決策樹

Python code

- 訓練「AdaBoost」模型

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier

tree = DecisionTreeClassifier(criterion='entropy',
                              max_depth=1,
                              random_state=1)

ada = AdaBoostClassifier(base_estimator=tree,
                         n_estimators=500,
                         learning_rate=0.1,
                         random_state=1)
ada = ada.fit(X_train, y_train)
```

單層決策樹

用500棵決策樹
單層決策樹

Python code

- 訓練「XGBoost」模型

```
from xgboost import XGBClassifier

xgbc = XGBClassifier()
xgbc.fit(X_train, y_train)
```