Boosting

Boosting 개요

❖ Boosting 아이디어

- 여러 개의 learning 모델을 순차적으로 구축하여 최종적으로 합침 (앙상블)
- 여기서 사용하는 learning 모델은 매우 단순한 모델
  - ✓ 단순한 모델: Model that slightly better than chance
- 순차적 → 모델 구축에 순서를 고려
- 각 단계에서 새로운 base learner를 학습하여 이전 단계의 base learner의 단점을 보완
- 각 단계를 거치면서 모델이 점차 강해짐 → boosting

Boosting 알고리즘 종류

- Adaptive boosting (Adaboost)
- Gradient boosting machines (GBM)
- XGboost
- Light gradient boost machines (Light GBM)
- Catboost

Adaptive Boosting (Adaboost)

❖ AdaBoost

- 각 단계에서 새로운 base learner를 학습하여 이전 단계의 base learner의 단점을 보완
- Training error가 큰 관측치의 선택 확률(가중치)을 높이고, training error가 작은 관측치의 선택 확률을 낮춤
  - ✓ 오분류한 관측치에 보다 집중!
- 앞 단계에서 조정된 확률(가중치)을 기반으로 다음 단계에서 사용될 training dataset를 구성
- 다시 첫 단계로 감
- 최종 결과물은 각 모델의 성능지표를 가중치로 하여 결합 (앙상블)

AdaBoost

❖ AdaBoost algorithm

Model, classifier

1. Set $W_i = \frac{1}{n}, i = 1,2,\ldots,n$ (impose equal weight initially)

$\log \frac{1-0.5}{0.5} = \log 1 = 0$

2. for $j = 1$ to m (m : number of classifiers)

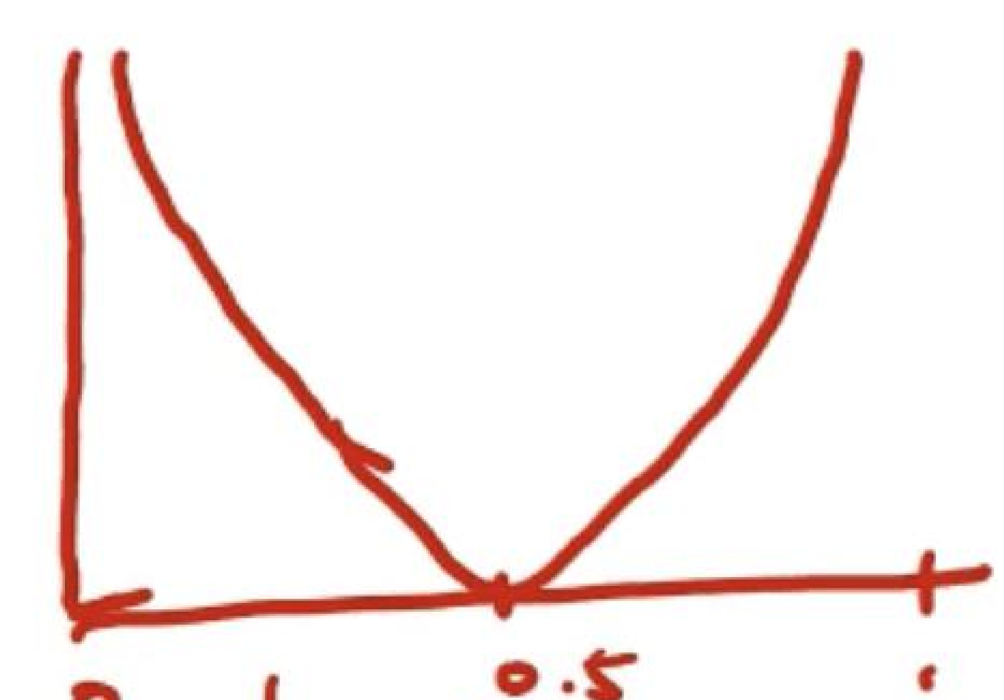
Step 1: Find $h_j(x)$ that minimizes L_j (weighted loss function)

$$L_j = \frac{\sum_{i=1}^{n} W_i\, I(y_i \neq h_j(x))}{\sum_{i=1}^{n} W_i}$$

★ $I(A)$ if A true → 1

if A false → 0

Step 2: Define the weight of a classifier: $\alpha_j = \log\left(\frac{1-L_j}{L_j}\right)$



→ 매우 헷갈린 경우

Step 3: Update weight: $W_i \leftarrow W_i\, e^{\alpha_j I(y_i \neq h_j(x))}, i = 1,2,\ldots,n$

endfor

$L_j = 0.5 \rightarrow \alpha_j = 0$

$L_j \approx 0 \rightarrow \alpha_j \rightarrow$ big $+\infty$

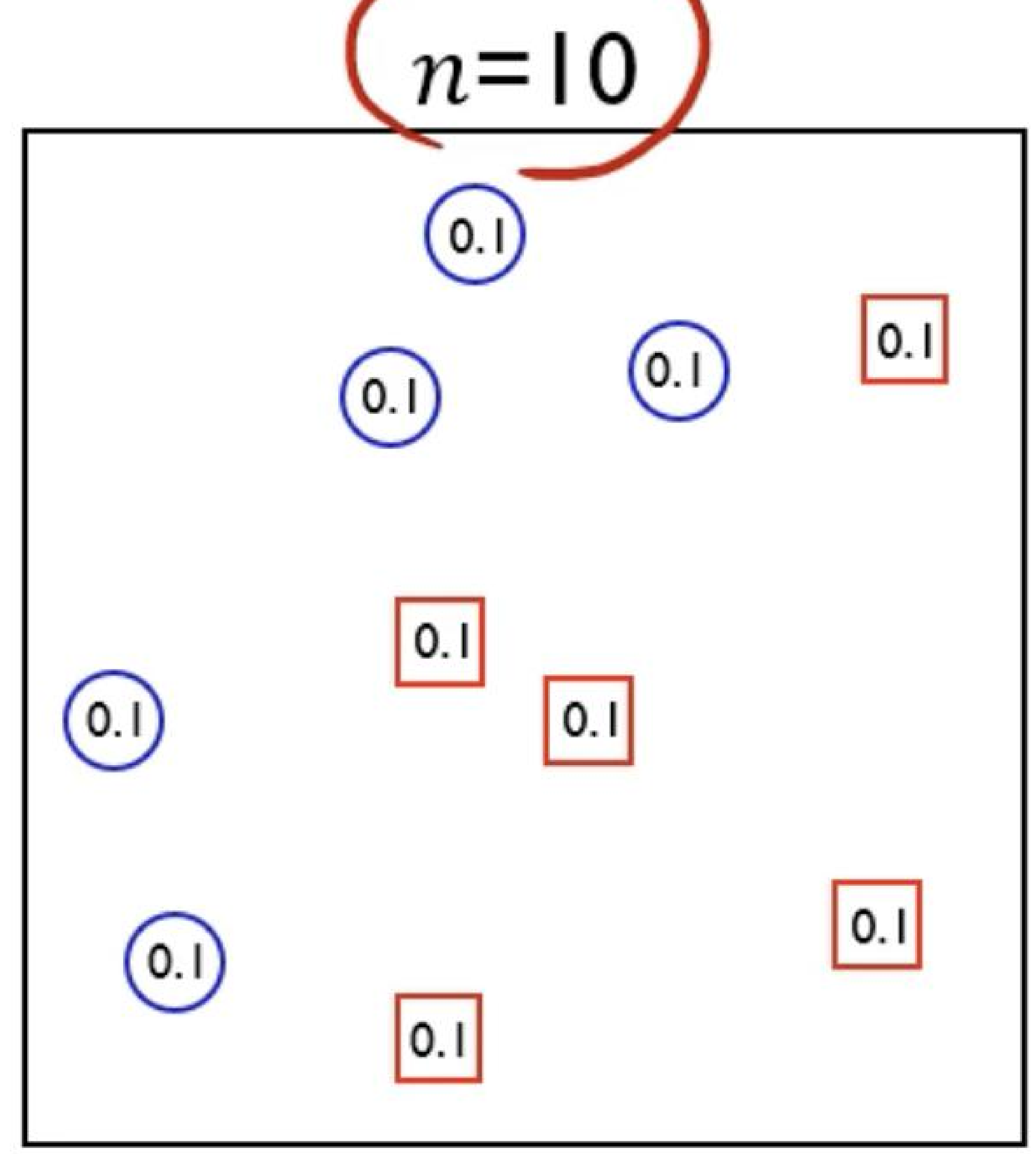
↳ perfect

3. Final boosted model: $h(x) = sign\left[\sum_{i=1}^{m} \alpha_j h_j(x)\right]$

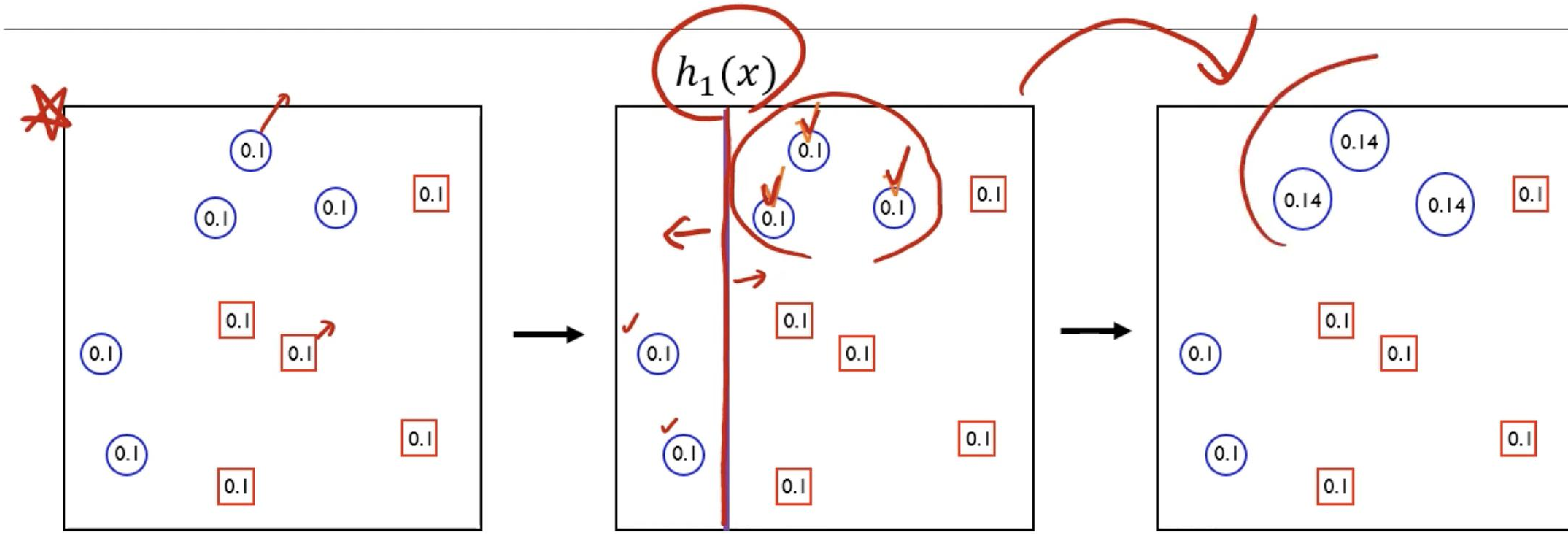
AdaBoost

Set $W_i = \frac{1}{n}, i = 1,2,\ldots,n$ (impose equal weight initially)

$wi = \frac{1}{10} = 0.1$



AdaBoost



$$L_1 = \frac{\sum_{i=1}^{n} W_i \, I(y_i \neq h_1(x))}{\sum_{i=1}^{n} W_i} = \frac{0.1 \times 3}{0.1 \times 10} = 0.3$$

10개중 3개 오분류

$$\alpha_1 = \log\left(\frac{1-L_j}{L_j}\right) = \log\left(\frac{1-0.3}{0.3}\right) \approx 0.37$$

1번째 classifier의 가중치

$$W_c = W_i \, e^{\alpha_1 I(y_i \neq h_1(x))} = 0.1e^{0.37\times 0} = 0.1$$

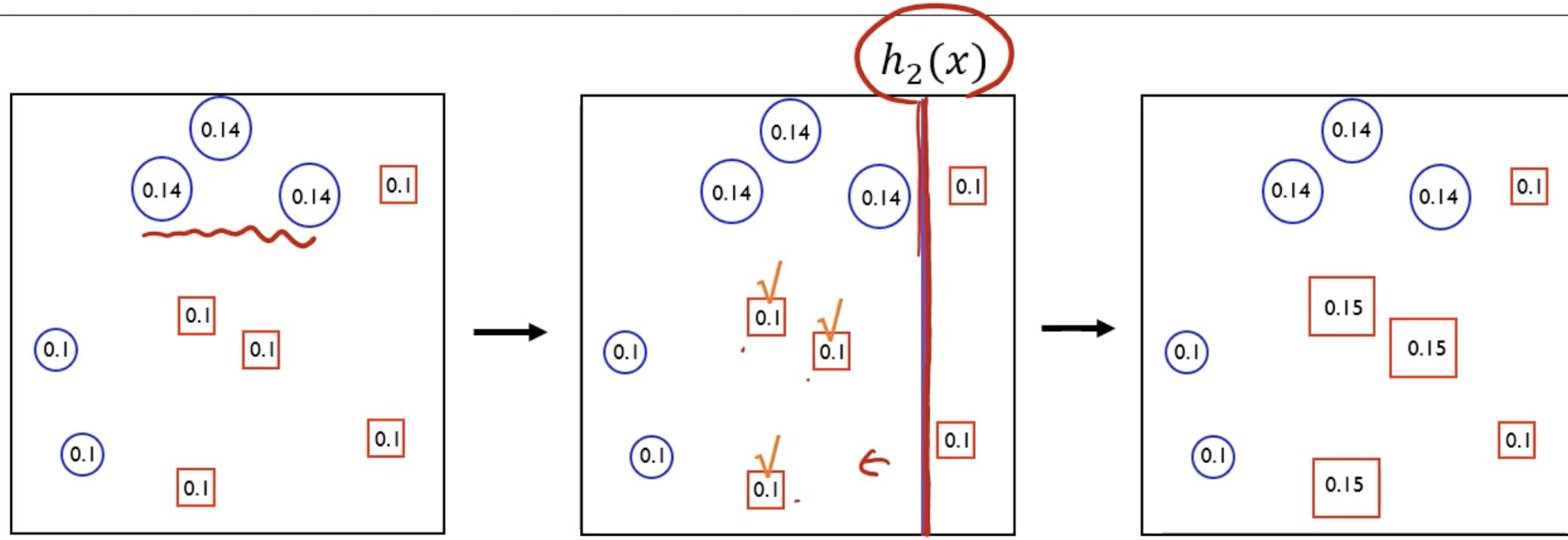
정분류 관측치 가중치

$$W_{nc} = W_i \, e^{\alpha_1 I(y_i \neq h_1(x))} = 0.1e^{0.37\times 1} = 0.14$$

오분류 √ 관측치 가중치

☺ 도형의 크기(안의 숫자)는 해당 관측치가 다음 단계의 학습데이터셋에 선택될 확률을 의미

AdaBoost



$$L_2 = \frac{\sum_{i=1}^{n} W_i \, I(y_i \neq h_2(x))}{\sum_{i=1}^{n} W_i} = \frac{0.1 \times 3}{0.1 \times 7 + 0.14 \times 3} = 0.27$$

10개중 3개 오분류

$$\alpha_2 = \log\left(\frac{1 - 0.27}{0.27}\right) \approx 0.43$$

$$W_c = W_i \, e^{\alpha_2 I(y_i \neq h_2(x))} = \{0.1 \ or \ 0.14\} \, e^{0.43 \times 0} = 0.1 \ or \ 0.14$$

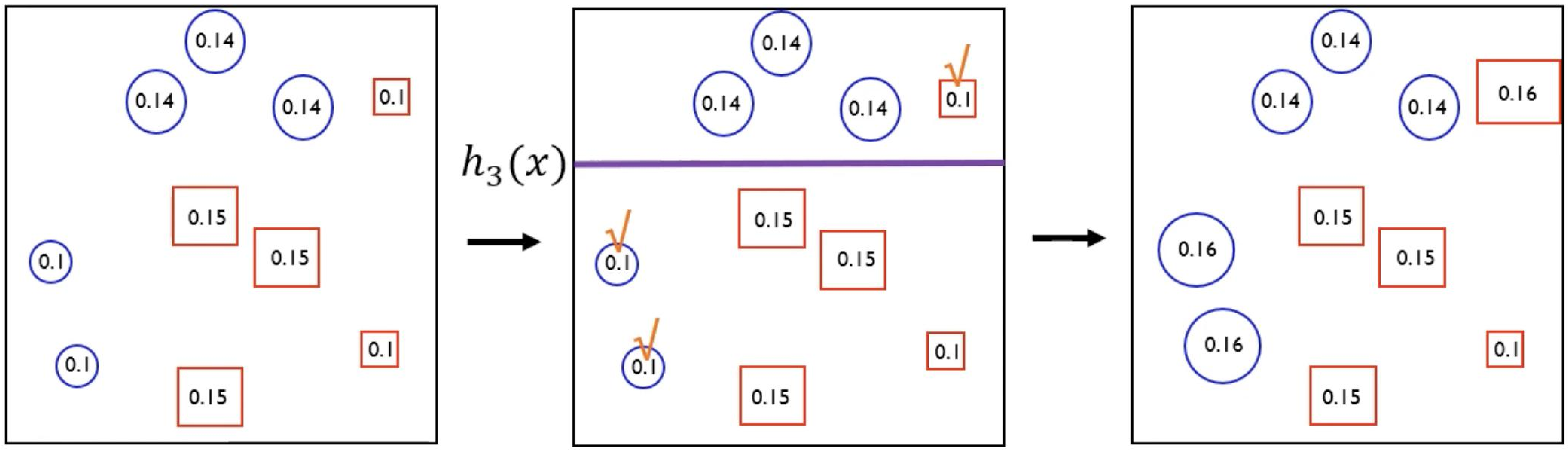
정분류 관측치 가중치

$$W_{nc} = W_i \, e^{\alpha_2 I(y_i \neq h_2(x))} = 0.1 e^{0.43 \times 1} = 0.15$$

오분류 √ 관측치 가중치

☺ 도형의 크기(안의 숫자)는 해당 관측치가 다음 단계의 학습데이터셋에 선택될 확률을 의미

AdaBoost



$$L_3 = \frac{\sum_{i=1}^{n} W_i \, I(y_i \neq h_2(x))}{\sum_{i=1}^{n} W_i} = \frac{0.1 \times 3}{0.1 \times 4 + 0.14 \times 3 + 0.15 \times 3} = 0.24$$ 10개중 3개 오분류

$$\alpha_3 = \log\left(\frac{1 - 0.24}{0.24}\right) \approx 0.5$$

$$W_c = W_i \, e^{\alpha_3 I(y_i \neq h_3(x))} = \{0.1 \; or \; 0.14 \; or \; 0.15\} \, e^{0.5 \times 0} = 0.1 \; or \; 0.14 \; or \; 0.15$$ 정분류 관측치 가중치

$$W_{nc} = W_i \, e^{\alpha_3 I(y_i \neq h_3(x))} = 0.1 e^{0.5 \times 1} = 0.16$$ 오분류 √ 관측치 가중치

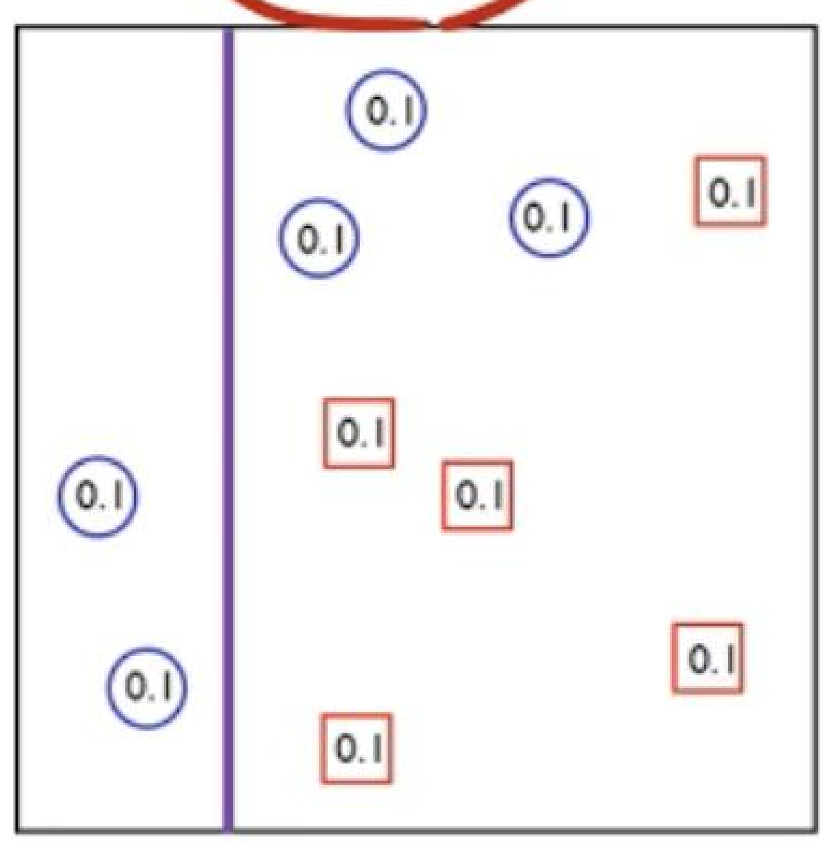
☺ 도형의 크기(안의 숫자)는 해당 관측치가 다음 단계의 학습데이터셋에 선택될 확률을 의미

AdaBoost

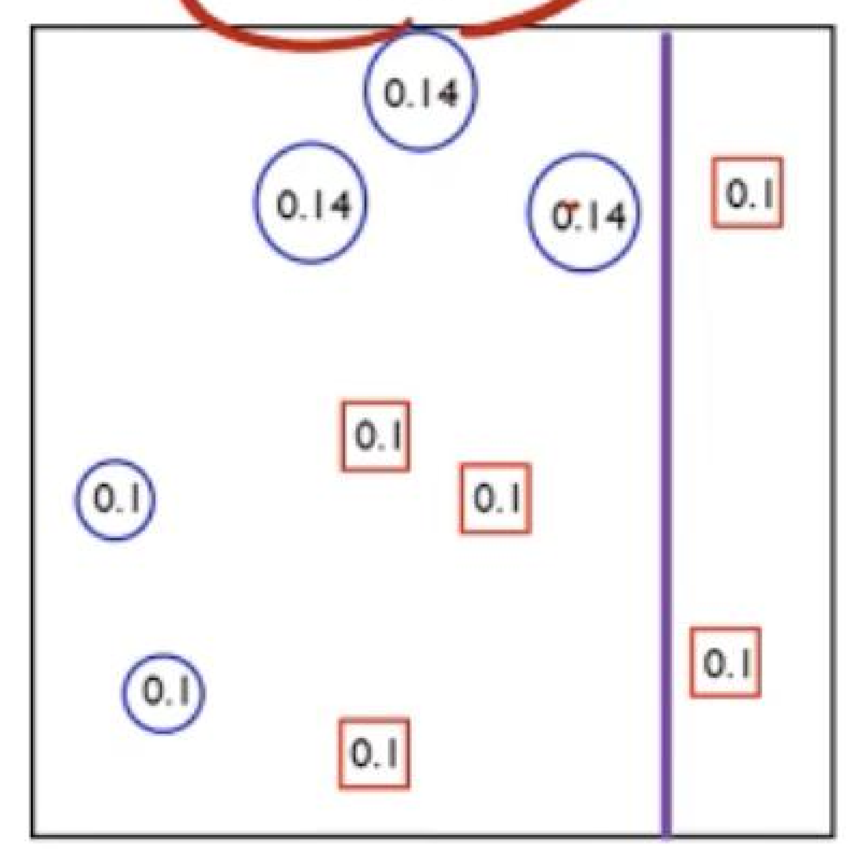
$$h(x) = sign\left[\sum_{i=1}^{m=3} \alpha_j h_j(x)\right]$$

$$h(x) = sign\left[\ \underset{\alpha_1}{0.37}\ h_1(x) \quad + \quad \underset{\alpha_2}{0.43}\ h_2(x) \quad + \quad \underset{\alpha_3}{0.5}\ h_3(x)\ \right]$$

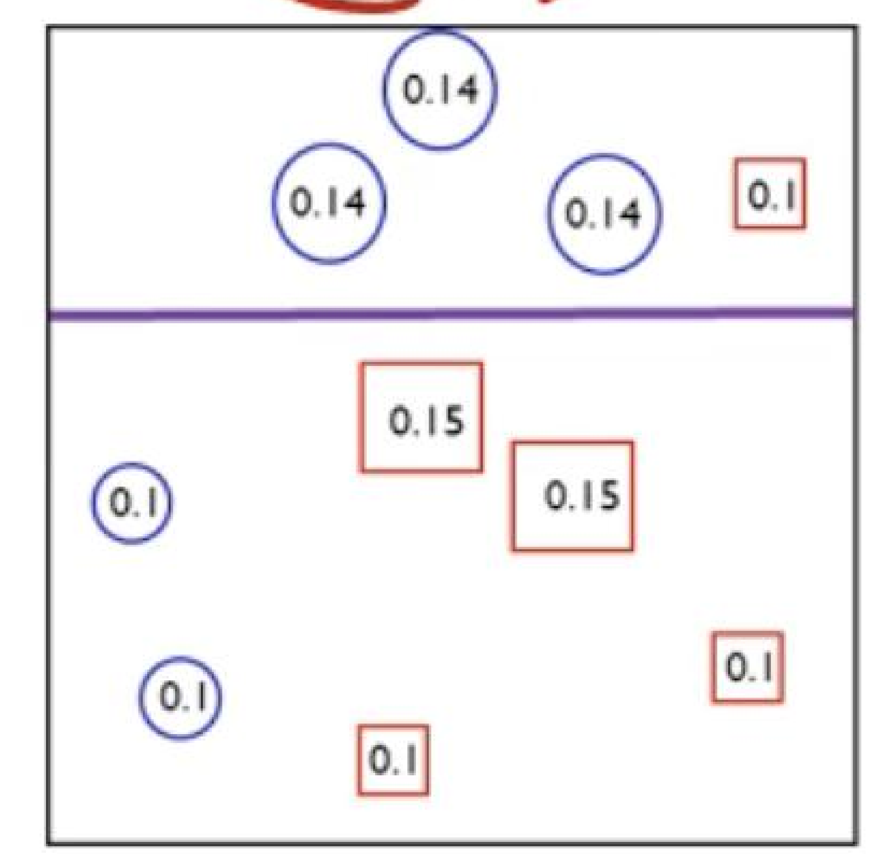
$h_1(x)$



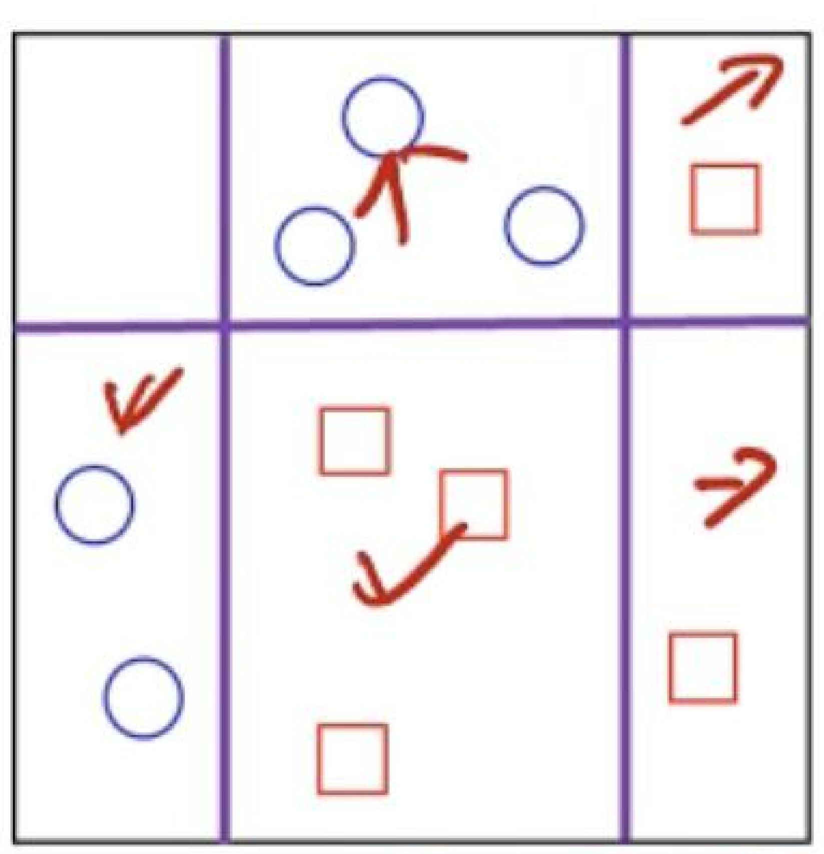
$h_2(x)$



$h_3(x)$



=



Adaboost

❖ AdaBoost algorithm

1. Set $W_i = \frac{1}{n}, i = 1,2,\dots,n$ (impose equal weight initially)

2. for $j = 1$ to m (m : number of classifiers)

   Step 1: Find $h_j(x)$ that minimizes L_j (weighted loss function)

$$L_j = \frac{\sum_{i=1}^{n} W_i\, I(y_i \neq h_j(x))}{\sum_{i=1}^{n} W_i}$$

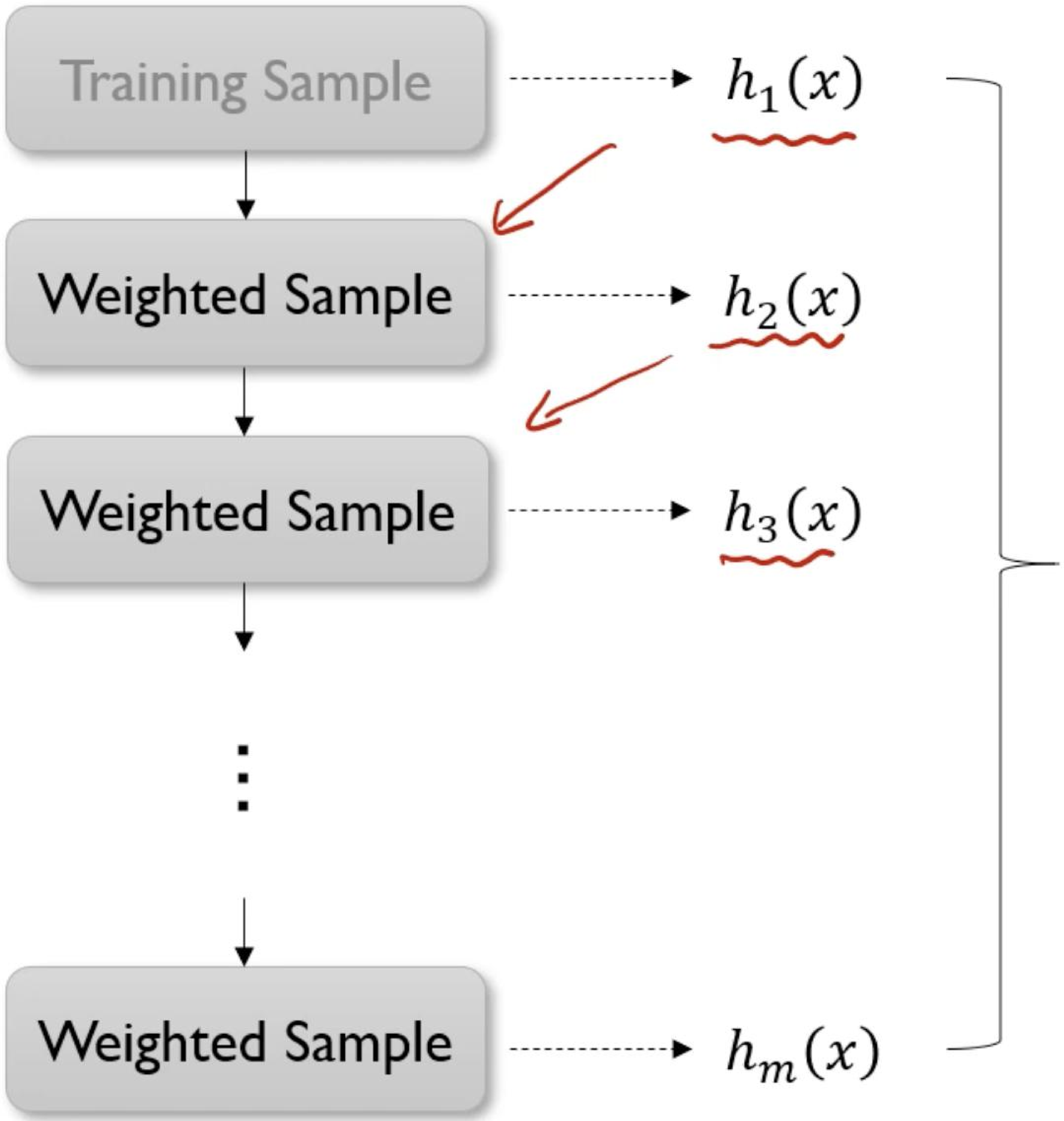
   Step 2: Define the weight of a classifier: $\alpha_j = \log\left(\frac{1-L_j}{L_j}\right)$

   Step 3: Update weight: $W_i \leftarrow W_i\, e^{\alpha_j I(y_i \neq h_j(x))}, i = 1,2,\dots,n$

endfor

3. Final boosted model: $h(x) = sign\left[\sum_{i=1}^{m} \alpha_j h_j(x)\right]$

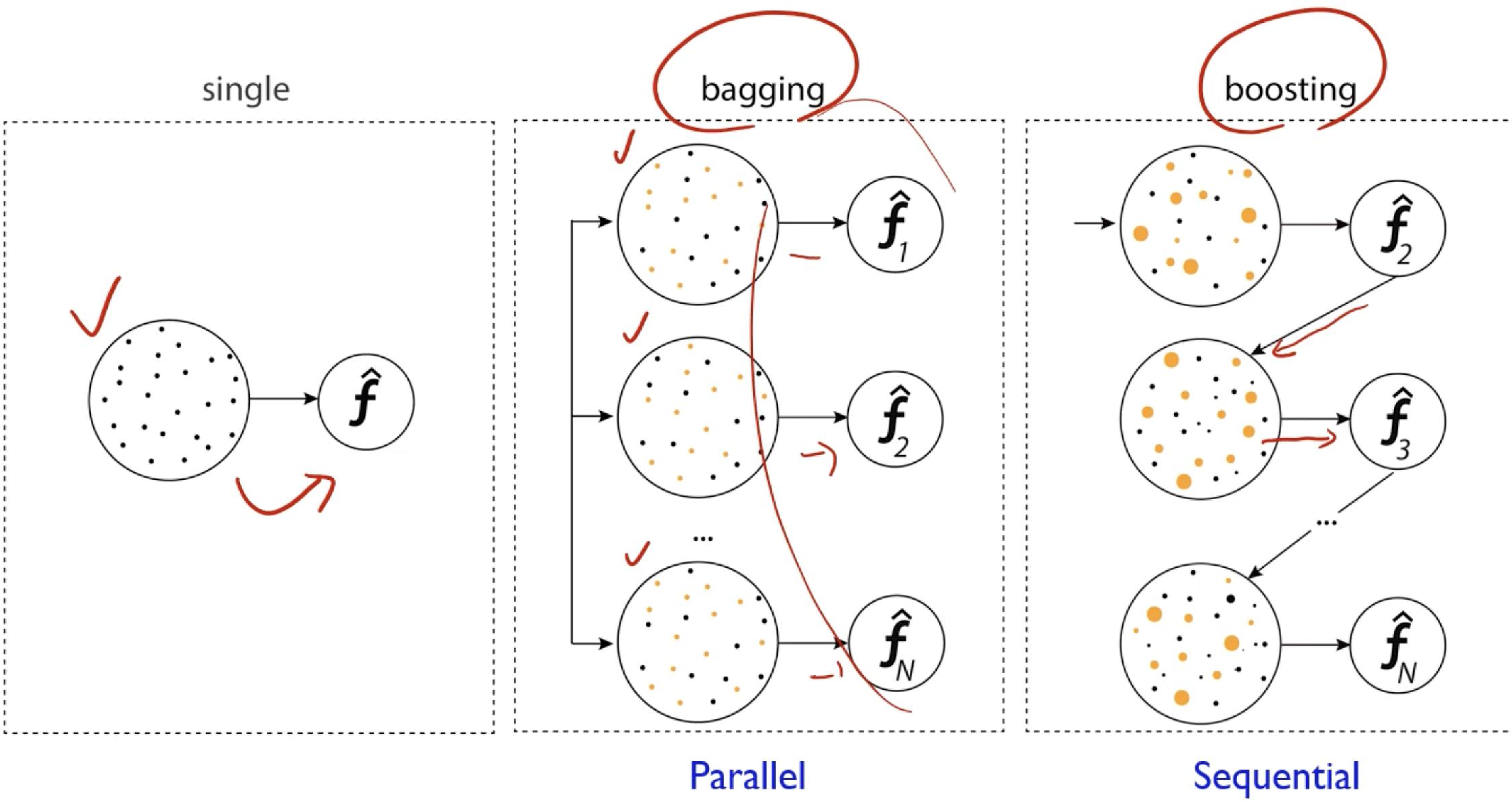
AdaBoost



$$h(x) = sign\left[\sum_{i=1}^{m} \alpha_j h_j(x)\right]$$

$h_1(x)$를 만들고, 이를 바탕으로, $h_2(x)$를 만들고, 이를 바탕으로 $h_3(x)$, ….

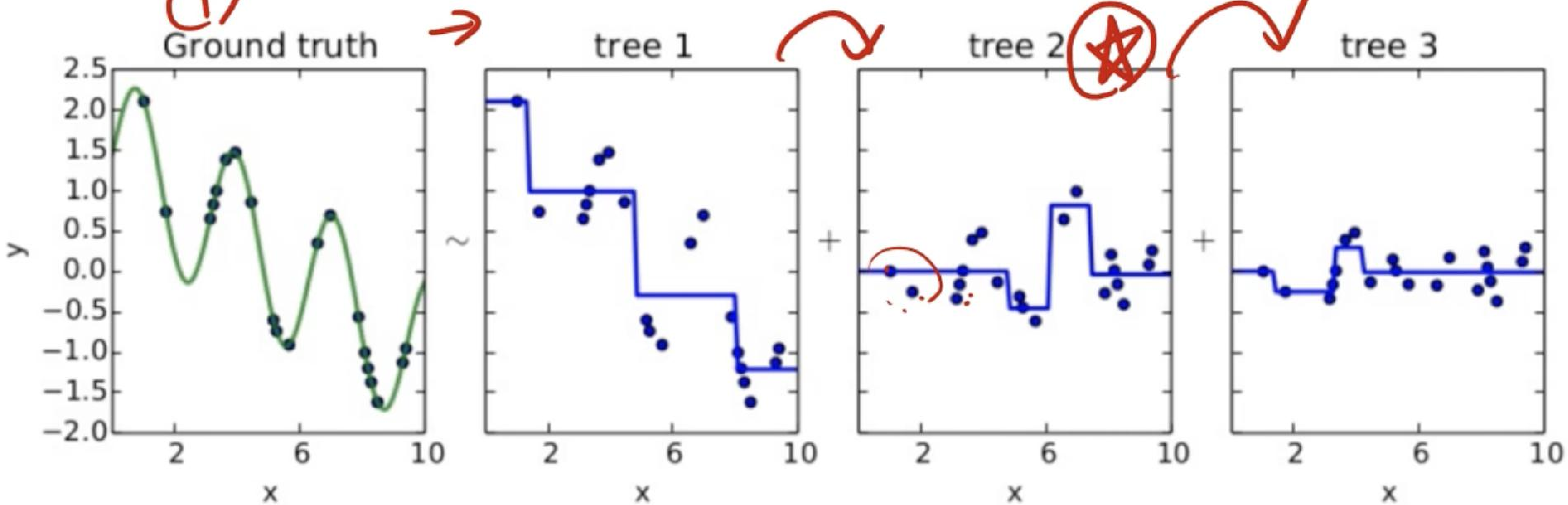
Bagging vs Boosting



Gradient Boosting Machines (GBM)

❖ GBM

- Gradient boosting = Boosting with gradient decent $y - \hat{y}$
- 첫번째 단계의 모델 tree1을 통해 Y를 예측하고, Residual을 다시 두번째 단계 모델 tree2를 통해 예측하고, 여기서 발생한 Residual을 모델 tree3로 예측
- 점차 residual 작아 짐
- Gradient boosted model = tree1 + tree2 + tree3



Gradient Boosting Machines (GBM)

❖ Why gradient?

미분

$$L = \frac{1}{2}\sum_{i=1}^{n}\{y_i - f(x_i)\}^2$$

Gradient

$$\frac{\partial L}{\partial f(x_i)} = -\{\underbrace{y_i - f(x_i)}_{\text{Residual}}\}$$

$$y_i - f(x_i) = -\frac{\partial L}{\partial f(x_i)}$$

Residual = Negative Gradient

Gradient Boosting Machines (GBM)

Y

Original Dataset

x_1	y_1
x_2	y_2
x_3	y_3
x_4	y_4
x_5	y_5
x_6	y_6
x_7	y_7
x_8	y_8
x_9	y_9
x_{10}	y_{10}

$y = f_1(x)$

Y'

Modified Dataset #1

x_1	$y_1-f_1(x_1)$
x_2	$y_2-f_1(x_2)$
x_3	$y_3-f_1(x_3)$
x_4	$y_4-f_1(x_4)$
x_5	$y_5-f_1(x_5)$
x_6	$y_6-f_1(x_6)$
x_7	$y_7-f_1(x_7)$
x_8	$y_8-f_1(x_8)$
x_9	$y_9-f_1(x_9)$
x_{10}	$y_{10}-f_1(x_{10})$

$y - f_1(x) = f_2(x)$

Y''

Modified Dataset #2

x_1	$y_1-f_1(x_1)-f_2(x_1)$
x_2	$y_2-f_1(x_2)-f_2(x_2)$
x_3	$y_3-f_1(x_3)-f_2(x_3)$
x_4	$y_4-f_1(x_4)-f_2(x_4)$
x_5	$y_5-f_1(x_5)-f_2(x_5)$
x_6	$y_6-f_1(x_6)-f_2(x_6)$
x_7	$y_7-f_1(x_7)-f_2(x_7)$
x_8	$y_8-f_1(x_8)-f_2(x_8)$
x_9	$y_9-f_1(x_9)-f_2(x_9)$
x_{10}	$y_{10}-f_1(x_{10})-f_2(x_{10})$

$y - f_1(x) - f_2(x) = f_3(x)$

Gradient
Residual

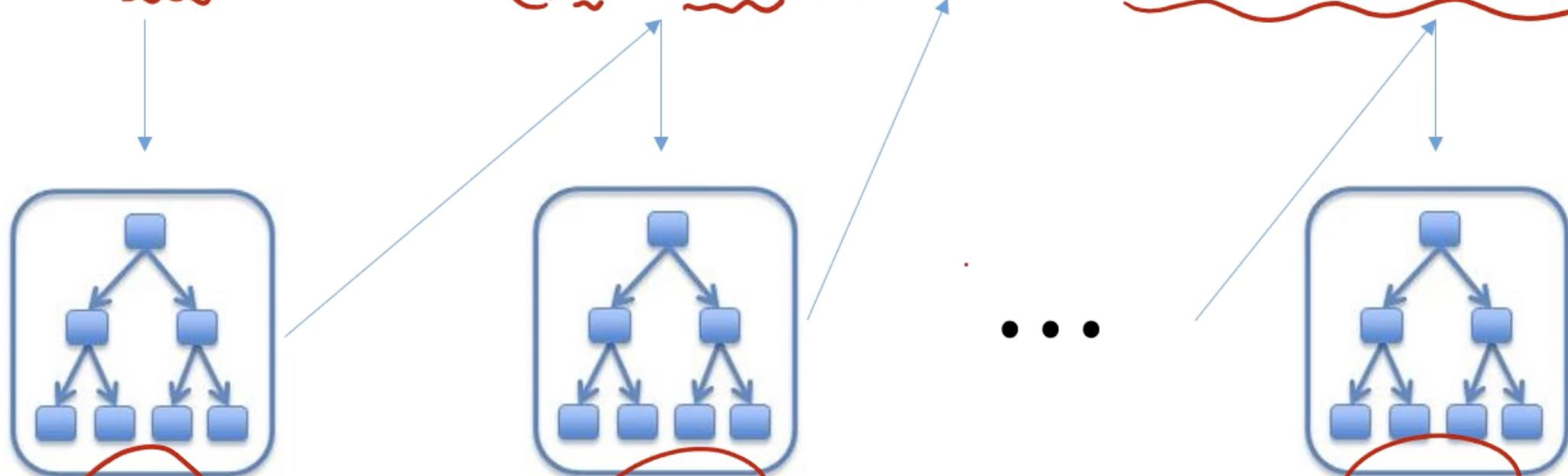
$y - f_1(x)$

$\{y - f_1(x)\} - f_2(x)$

$\{y - f_1(x) - f_2(x)\} - f_3(x)$

Gradient Boosting Machines (GBM)

$D_1 = (x_i, y_i)_{i=1}^{n}$ $\quad D_2 = \left(x_i, y_1 - f_1(x_i)\right)_{i=1}^{n}$ $\quad \cdots \quad$ $D_M = \left(x_i, y_1 - f_1(x_i) - \cdots - f_{M-1}(x_i)\right)_{i=1}^{n}$



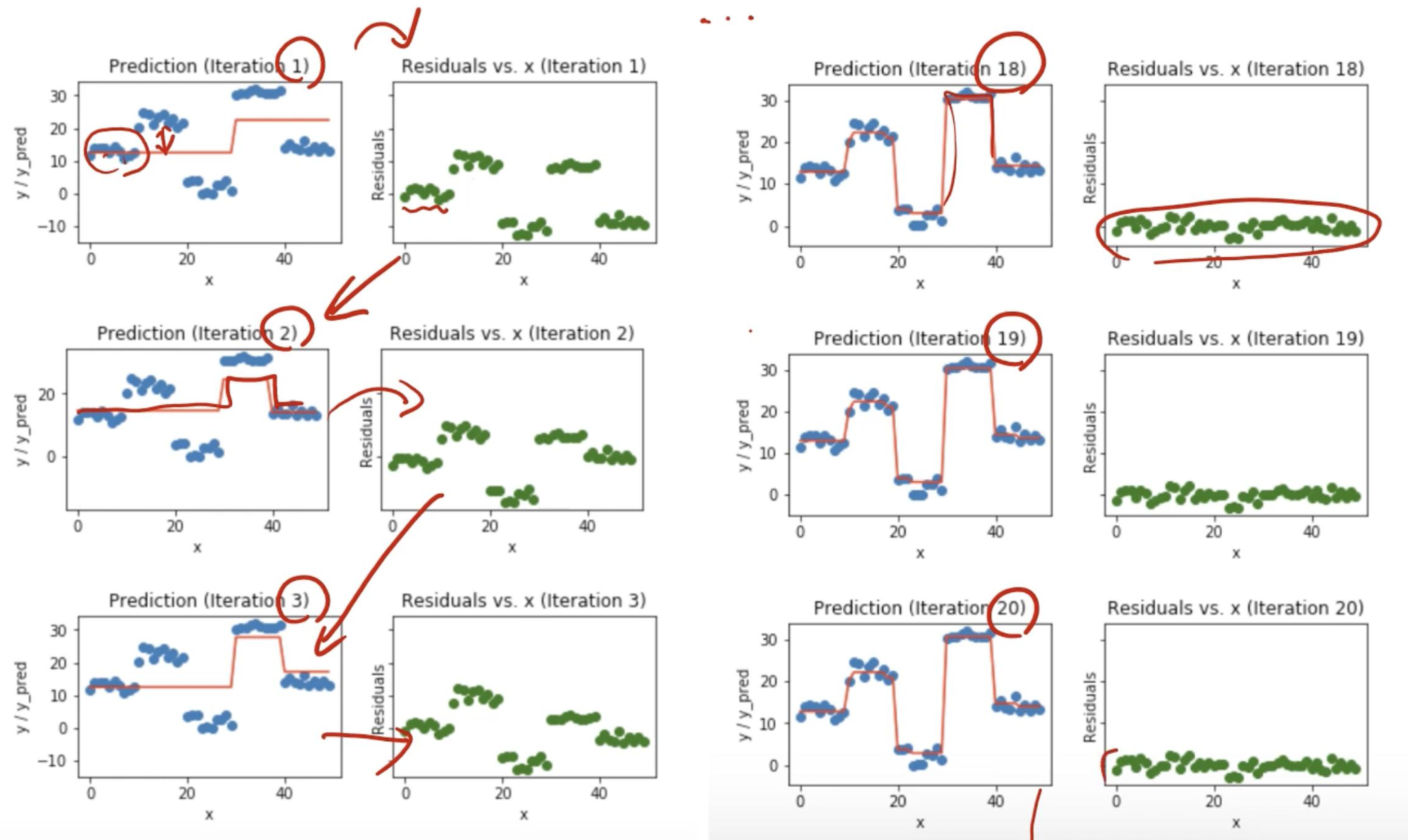
$f_1(x)$ $\quad f_2(x)$ $\quad f_M(x)$

M

$$\hat{f}(x) = f_1(x) + f_2(x) + \cdots + f_M(x)$$

https://dimensionless.in/gradient-boosting/

Gradient Boosting Machines (GBM)



https://deepai.org/machine-learning-glossary-and-terms/gradient-boosting

Gradient Boosting Machines (GBM)

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:

   (a) For $i = 1, 2, \ldots, N$ compute

   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

   $$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.