Boosting

- In bagging, we created an ensemble for reducing the variance in high-variance-low-bias (strong) base models

- Boosting is another ensemble method used for reducing the bias in high-bias-low-variance (weak) base models

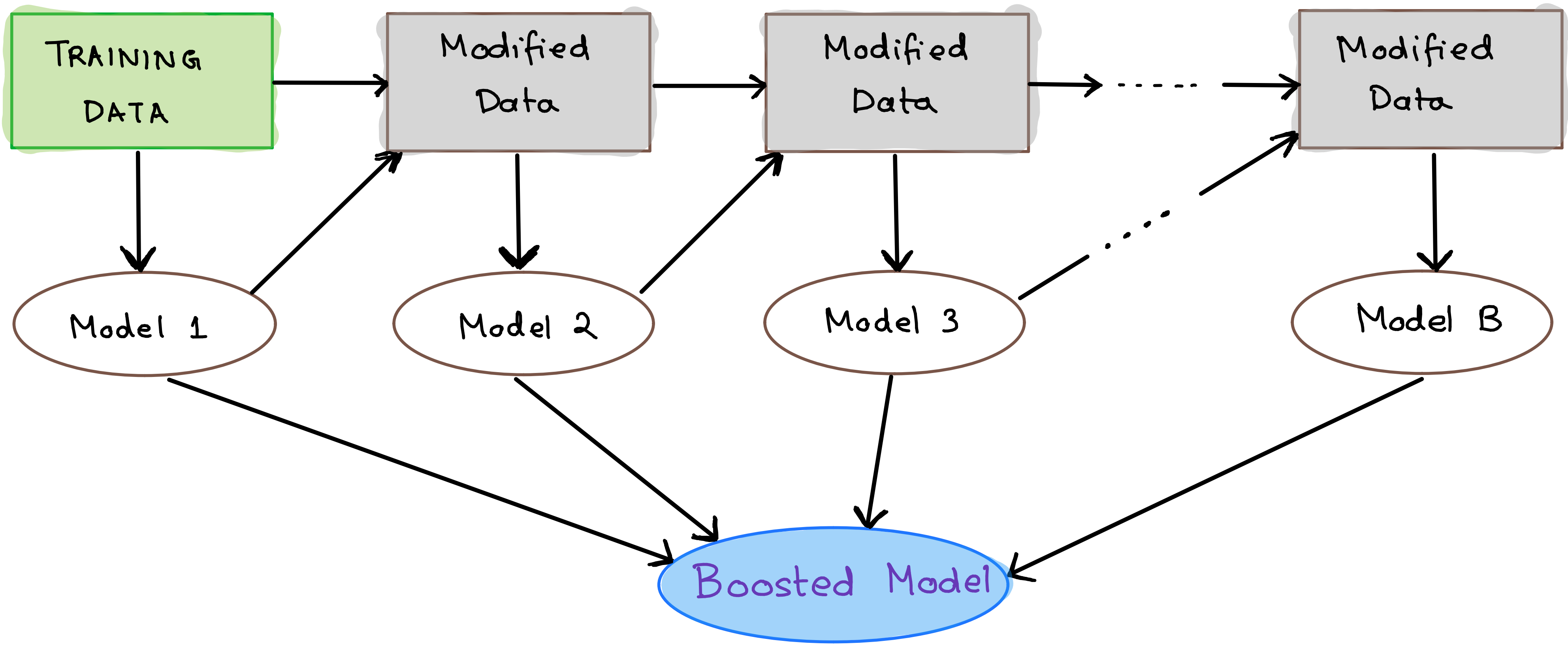
- Intuition:
  - Even a simple (weak) model can typically describe some aspects of the input-output (I/O) relationship
  - Can we then learn an ensemble of "weak models", where each weak model describes some part of the I/O relationship, and combine these models into one "strong model"?

- Boosting shares some similarities with bagging
  - Both use an ensemble of models for combining predictions
  - Both can be used with any regression or classification algorithm

- Difference between bagging and boosting lies in how the base models are being trained
  - In bagging, 'B' identically distributed models are constructed parallely
  - In boosting, the ensemble members are constructed sequentially.

Sequential Construction in Boosting

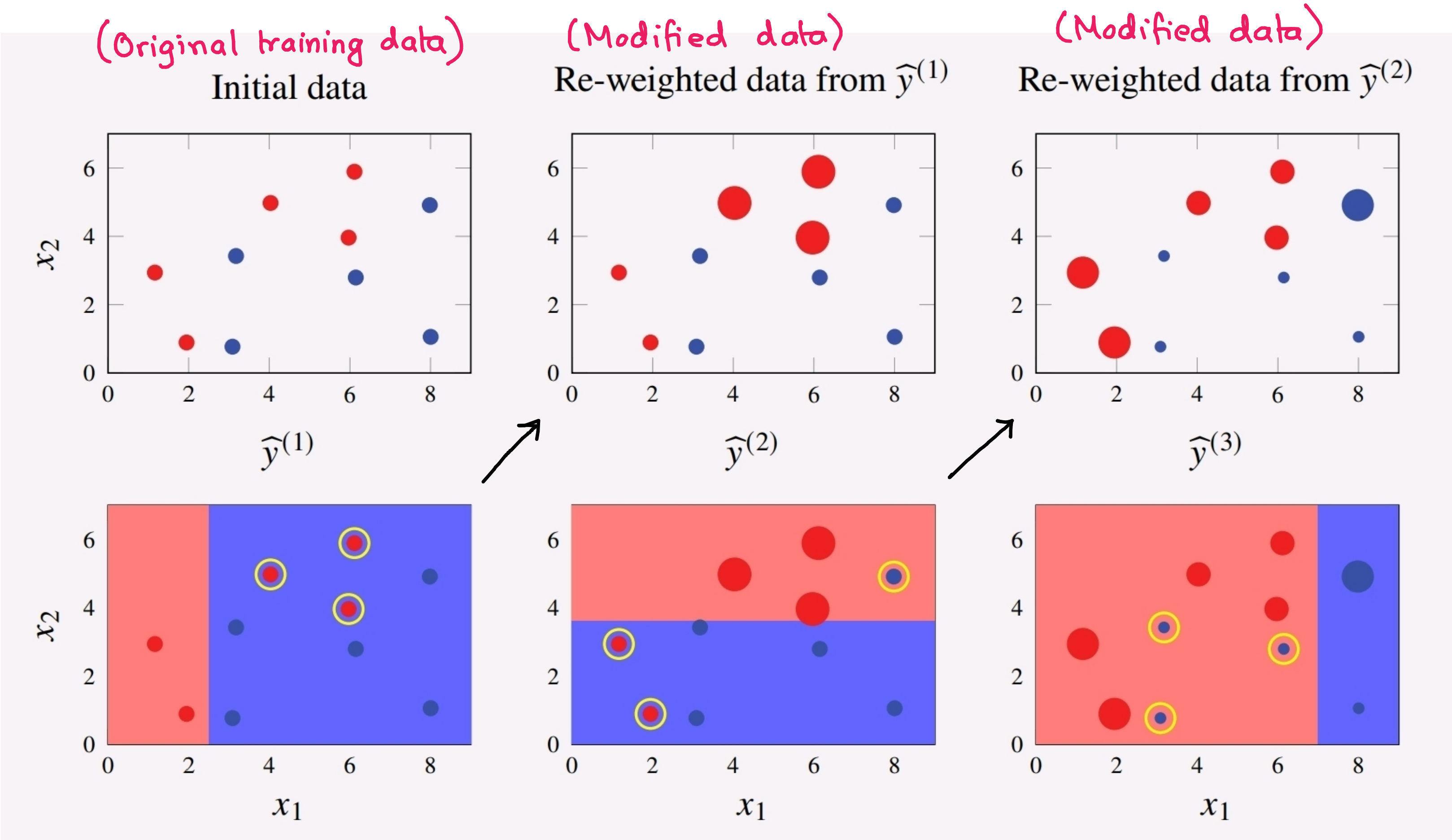
Informally, the sequential construction of ensemble members is done in such a way that each model tries to correct the mistakes made by the previous one



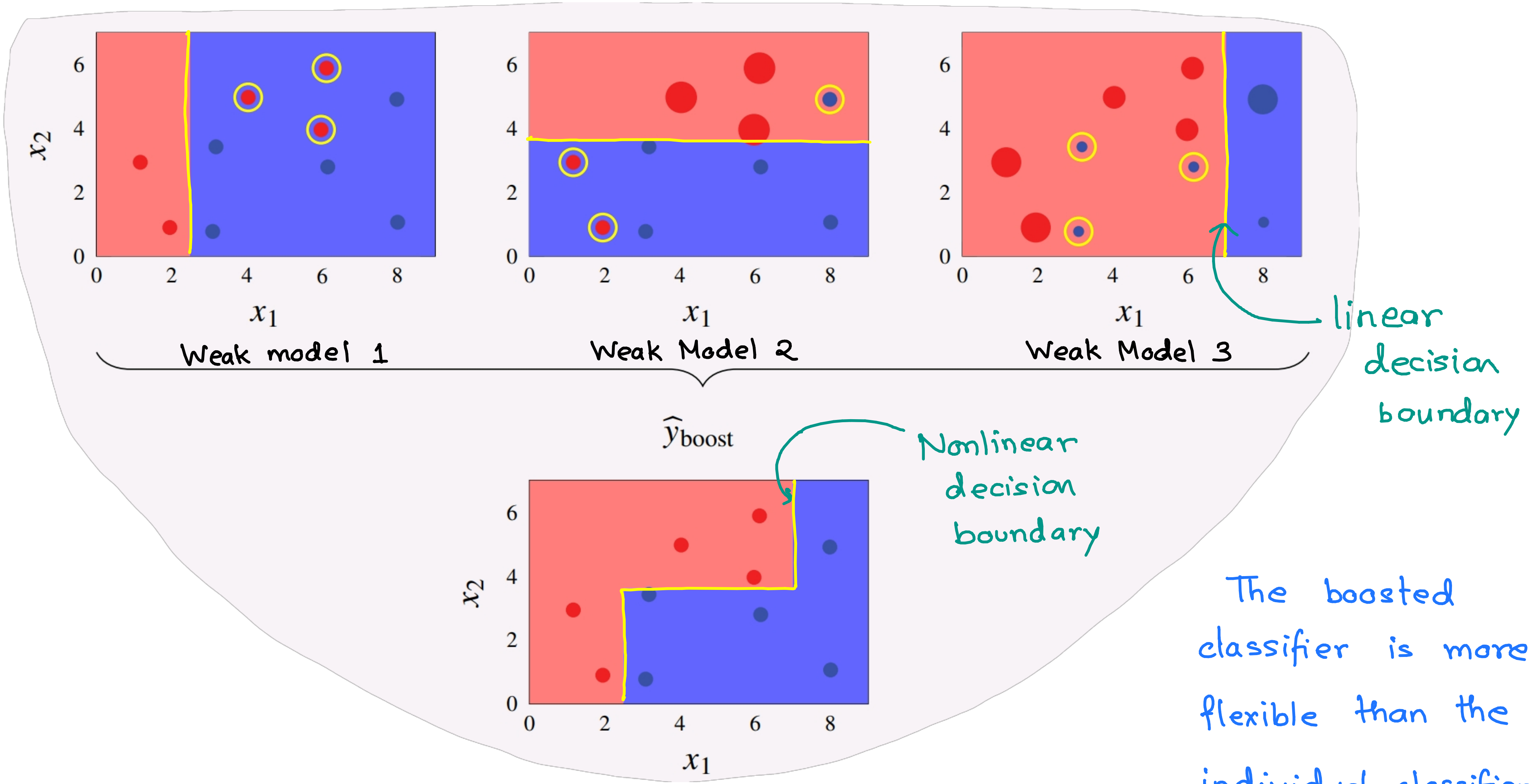
Example of sequential construction in boosting

Consider a binary classification problem with 2D input $\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

- There are $N = 10$ datapoints, 5 from each class
- A classification tree of depth one (weak model) is used as the base classifier (splits into two regions)



Weak model 1 Weak model 2 Weak model 3



Boosted model

The boosted classifier is more flexible than the individual classifiers

The final classifier $\hat{y}_{\text{boost}}(\underline{x})$ = Weighted majority vote of the three weak decision trees

Boosting Procedure (for classification)

Input: Training set $T = \{ \underline{x}_i , y_i \}_{i=1}^{N}$

Output: Boosted predictions $\hat{y}_{boost}(\underline{x})$

1. Assign weights $w_i^{(1)} = 1/N$ to all data points

2. For $b = 1$ to B

   - Train a weak classifier $\hat{y}^{(b)}(\underline{x})$ on the weighted training data $\{(\underline{x}_i, y_i, w_i^{(b)})\}_{i=1}^{N}$

   - Update the weights $\{w_i^{(b+1)}\}_{i=1}^{N}$ from $\{w_i^{(b)}\}_{i=1}^{N}$:

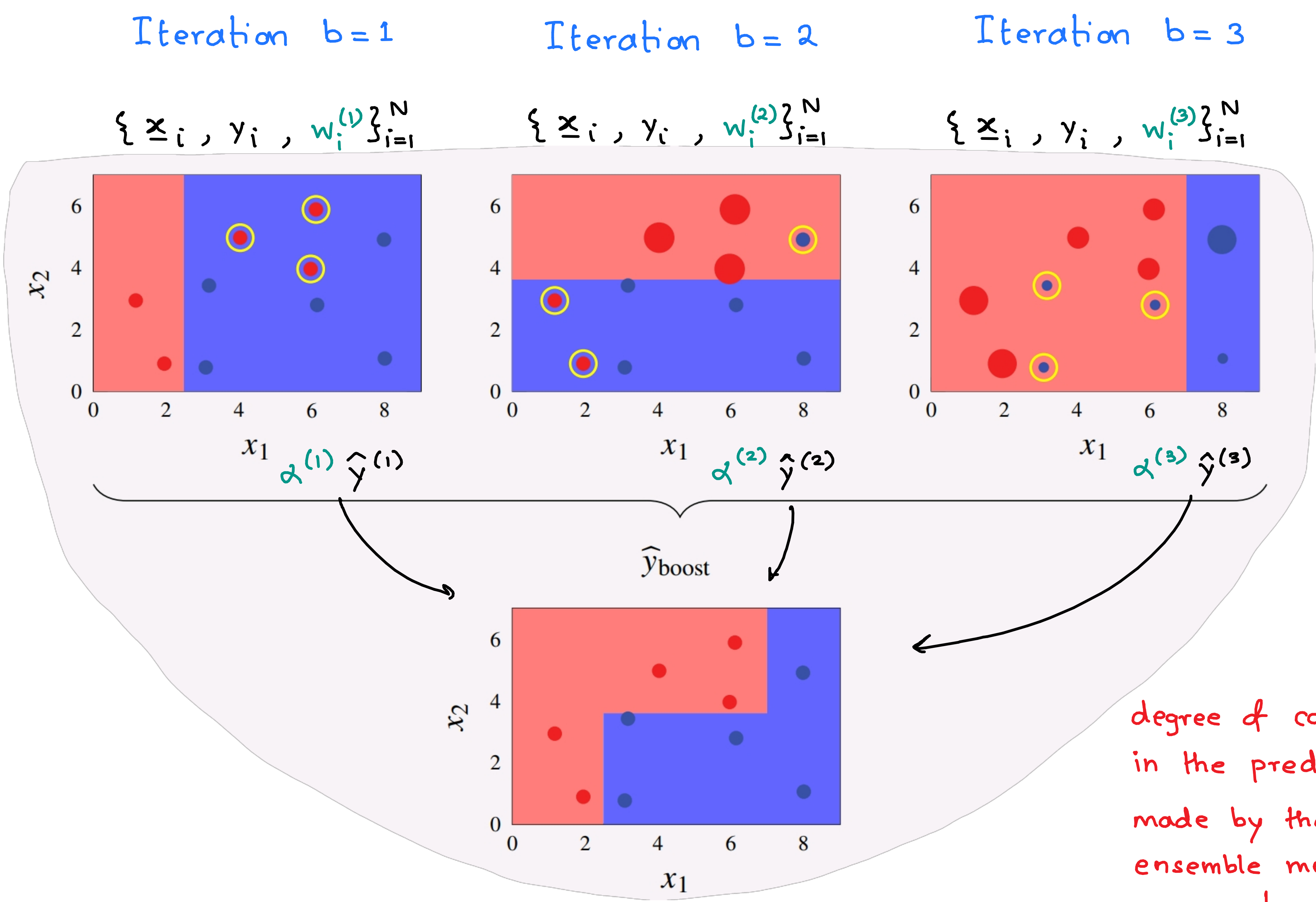
     → Increase weights for all points misclassified by $\hat{y}^{(b)}(\underline{x})$

     → Decrease weights for all points correctly classified by $\hat{y}^{(b)}(\underline{x})$

3. The predictions from the 'B' classifiers, $\hat{y}^{(1)}(\underline{x}), \hat{y}^{(2)}(\underline{x}), \ldots, \hat{y}^{(B)}(\underline{x})$, are combined using a weighted majority vote:

$$\boxed{\hat{y}_{boost}(\underline{x}) = \operatorname{sign}\left(\sum_{b=1}^{B} \alpha^{(b)} \hat{y}^{(b)}(\underline{x}) \right)}$$

$\alpha^{(b)} > 0$ always



degree of confidence in the predictions made by the 'b'th ensemble member

$$\hat{y}_{\text{boost}}(\underline{x}) = \operatorname{sign}\left(\sum_{b=1}^{3} \alpha^{(b)} \hat{y}^{(b)}(\underline{x})\right)$$

- How do we reweight the data, $w_i^{(b)}$s ?
- How are the coefficients $\alpha^{(1)}, \dots, \alpha^{(B)}$ computed ?

AdaBoost (Adaptive Boosting)

- It is the first successful implementation of the idea of boosting
- We will restrict our focus to binary classification, but boosting is also applicable to multi-class classification & regression problems
- Output of the AdaBoost classifier:

+1/−1 from individual members

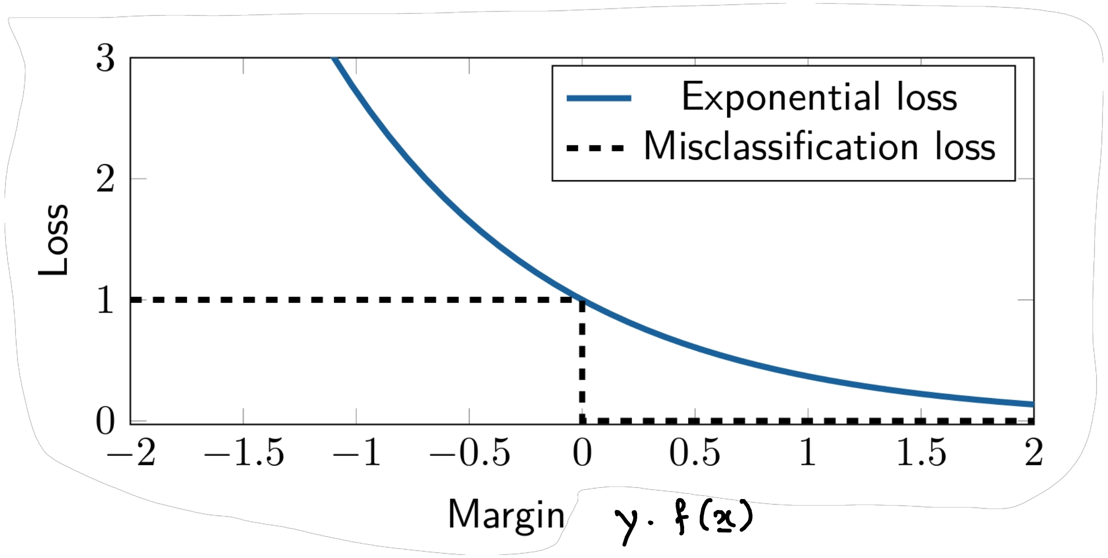
$$\hat{y}_{boost}(\underline{x}) = \text{sign}\left\{ \sum_{b=1}^{B} \alpha^{(b)} \hat{y}^{(b)}(\underline{x}) \right\}$$

- The training of an AdaBoost classifier follows the general form of a binary classifier $y = \text{sign}\{f(\underline{x})\}$
  - The class predictions are obtained by thresholding $f(\underline{x})$ at zero
  - In AdaBoost, they are obtained by thresholding the weighted sum of predictions made by all ensemble members

Exponential Loss in AdaBoost

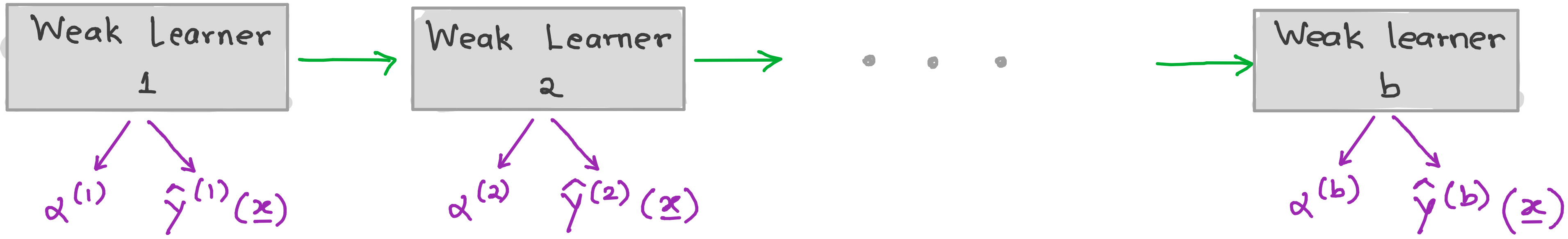
- AdaBoost uses exponential loss ← because it results in convenience in calculations

$$L(y, \hat{y}) = \exp\Big(-\underbrace{y \cdot \overbrace{f(\underline{x}; \underline{\theta})}^{\hat{y}}}_{\text{Margin}}\Big)$$



- The ensemble members are added one at a time, and when the 'b' th member is added, it is done to minimize the exponential loss of the entire ensemble constructed so far

Training of AdaBoost Classifier



Iter 1 : Model 1 prediction $\rightarrow$ $f^{(1)}(\underline{x}) = \alpha^{(1)}\hat{y}^{(1)}(\underline{x})$

Iter 2 : Model 2 prediction $\rightarrow$ $f^{(2)}(\underline{x}) = f^{(1)}(\underline{x}) + \alpha^{(2)}\hat{y}^{(2)}(\underline{x})$

Item 3 : Model 2 prediction $\rightarrow$ $f^{(3)}(\underline{x}) = f^{(2)}(\underline{x}) + \alpha^{(3)}\hat{y}^{(3)}(\underline{x})$

$\vdots \qquad \vdots \qquad \vdots$

Iter b : Model b prediction $\rightarrow$ $f^{(b)}(\underline{x}) = f^{(b-1)}(\underline{x}) + \alpha^{(b)}\hat{y}^{(b)}(\underline{x})$

Training of AdaBoost Classifier

- Predictions from boosted classifier after 'b' iterations

$$\hat{y}^{(b)}_{boost}(\underline{x}) = \text{sign}\left\{ \underbrace{\sum_{j=1}^{b} \alpha^{(j)} \hat{y}^{(j)}(\underline{x})}_{f^{(b)}(\underline{x})} \right\}$$

$$= \text{sign}\left\{ f^{(b)}(\underline{x}) \right\}$$

Iter 1   Iter 2

Model 1   Model 2 ....   Model B

$\hat{y}^{(1)}$ $\hat{y}^{(2)}$

$\alpha^{(1)}$ $\alpha^{(2)}$

$$\hat{y}^{(2)}_{boost} = \sum_{j=1}^{2} \alpha^{(j)} \hat{y}^{(j)}$$

- We can express $f^{(b)}(x)$ iteratively : $f^{(b)}(\underline{x}) = f^{(b-1)}(\underline{x}) + \alpha^{(b)} \hat{y}^{(b)}(\underline{x})$

output of 'b'th ensemble

- The ensemble members as well as the coefficients $\alpha^{(b)}$ are constructed sequentially
  - At the 'b' iteration, function $f^{(b-1)}(\underline{x})$ is known and kept fixed
  - Only $\alpha^{(b)}$ and the 'b'th model $\hat{y}^{(b)}(\underline{x})$ is learned
  - This is also called "GREEDY" construction

- $f^{(b)}(\underline{x}) = f^{(b-1)}(\underline{x}) + \alpha^{(b)} \hat{y}^{(b)}(\underline{x}) \qquad [\, f^{(0)}(\underline{x}) = 0 \,]$

- Training is done by minimizing the exponential loss of the data:

$$(\hat{\alpha}^{(b)}, \hat{y}^{(b)}(\underline{x})) = \underset{(\alpha, \hat{y})}{\arg\min} \sum_{i=1}^{N} L\left(y_i, f^{(b)}(\underline{x}_i)\right)$$

$$= \underset{(\alpha, \hat{y})}{\arg\min} \sum_{i=1}^{N} \exp\left(-y_i \cdot f^{(b)}(\underline{x}_i)\right)$$

$$= \underset{(\alpha, \hat{y})}{\arg\min} \sum_{i=1}^{N} \exp\left(-y_i \cdot \left(f^{(b-1)}(\underline{x}_i) + \underbrace{\alpha \hat{y}(\underline{x}_i)}_{\text{Unknown}} \right)\right)$$

$$= \underset{(\alpha, \hat{y})}{\arg\min} \sum_{i=1}^{N} \underbrace{\exp\left(-y_i \cdot f^{(b-1)}(\underline{x}_i)\right)}_{= w_i^{(b)}} \exp\left(-y_i \, \alpha \hat{y}(\underline{x}_i)\right)$$

- Weights for individual data points in training set for 'b' th iteration

$$w_i^{(b)} \stackrel{\text{def}}{=} \exp\left(- y_i \cdot f^{(b-1)}(\underline{x}_i)\right)$$

- Weights $w_i^{(b)} = \exp\left(-y_i \; f^{(b-1)}(\underline{x}_i)\right)$

- Note that the weights $\{w_i^{(b)}\}_{i=1}^{N}$ are independent of $\alpha^{(b)}$ & $\hat{y}^{(b)}(\underline{x})$

  - When learning $\hat{y}^{(b)}(\underline{x})$ and $\alpha^{(b)}$ by solving the loss minimization,

    we can consider $\{w_i^{(b)}\}_{i=1}^{N}$ as constants

Constant

$$(\hat{\alpha}^{(b)}, \hat{y}^{(b)}(\underline{x})) = \underset{(\alpha, \hat{y})}{\arg\min} \sum_{i=1}^{N} w_i^{(b)} \exp\left(-y_i \; \alpha \; \hat{y}(\underline{x}_i)\right)$$

- Rewrite the objective function as

Indicator function returns 0 / 1

$$\sum_{i=1}^{N} w_i^{(b)} \exp\left(-y_i \cdot \alpha \; \hat{y}(\underline{x}_i)\right) = \underbrace{e^{-\alpha} \sum_{i=1}^{N} w_i^{(b)} \, \mathbb{I}\{y_i = \hat{y}(\underline{x}_i)\}}_{= W_c}$$

correct

incorrect

$$+ \underbrace{e^{\alpha} \sum_{i=1}^{N} w_i^{(b)} \, \mathbb{I}\{y_i \neq \hat{y}(\underline{x}_i)\}}_{= W_e}$$

* $\hat{y}(\underline{x}_i)$ is the ensemble member we are to learn here

- Rewriting the objective function:

$$\underbrace{\sum_{i=1}^{N} w_i^{(b)} \exp\left(-y_i \cdot \alpha\, \hat{y}(\underline{x}_i)\right)}_{\text{"OBJ"}} = e^{-\alpha} \underbrace{W_c}_{\text{Weights for correctly classified pts}} + e^{\alpha} \underbrace{W_e}_{\text{Weights for incorrectly classified points}}$$

$$W_c = \sum_{i=1}^{N} w_i^{(b)} \mathbb{I}\{y_i = \hat{y}(\underline{x}_i)\}$$

$$W_e = \sum_{i=1}^{N} w_i^{(b)} \mathbb{I}\{y_i \neq \hat{y}(\underline{x}_i)\}$$

- Let $W = W_c + W_e$ be the total sum of weights

$$= \sum_{i=1}^{N} w_i^{(b)}$$

- The "OBJ" is minimized in two stages:
  - first w.r.t. $\hat{y}$
  - then w.r.t α

- This is possible because the argument $\hat{y}$ turns out to be independent of the actual value of α (>0)

- Let $W = W_c + W_e$ be the total sum of weights

$$= \sum_{i=1}^{N} w_i^{(b)}$$

- The "OBJ" is minimized in two stages:
  - first w.r.t. $\hat{y}$
  - then w.r.t α
- This is possible because the argument $\hat{y}$ turns out to be independent of the actual value of α (>0)
- To see this, note that we can write the "OBJ" function

$$\text{"OBJ"} = e^{-\alpha} W_c + e^{\alpha} W_e$$

$$= e^{-\alpha}(W - W_e) + e^{\alpha} W_e = e^{-\alpha} W + \underbrace{(e^{\alpha} - e^{-\alpha})}_{>0 \ \forall \alpha > 0} W_e$$

$W = \sum_{i=1}^{N} w_i^{(b)}$ ← independent of y_i

weights of incorrectly classified points

- Minimizing "OBJ" is equivalent to minimizing W_e w.r.t. $\hat{y}$

$$\hat{y}^{(b)} = \arg\min_{\hat{y}} \sum_{i=1}^{N} w_i^{(b)} \boxed{\mathbb{I}\{y_i \neq \hat{y}(\underline{x}_i)\}}$$

← misclassification loss

- Minimizing "OBJ" is equivalent to minimizing W_e w.r.t. $\hat{y}$

$$\hat{y}^{(b)} = \arg\min_{\hat{y}} \sum_{i=1}^{N} w_i^{(b)} \boxed{\mathbb{I}\{y_i \neq \hat{y}(x_i)\}}$$

weighted misclassification loss for the ith data point

- So, the 'b'th ensemble member should be trained by minimizing the weighted misclassification loss for all data points
  - This resembles standard training of classifiers, except for the weights $w_i^{(b)}$, which boils down to weighing the loss for each data point

- The intuition for weights $w_i^{(b)}$ is that, at iteration 'b', we should focus our attention on data points previously misclassified in order to "correct the mistakes" made by the ensemble of the first (b-1) classifiers

- Once the 'b' th ensemble member, $\hat{y}^{(b)}(\underline{x})$, has been trained we then need to learn coefficient $\alpha^{(b)}$

- It is done by minimizing the "OBJ" w.r.t α

$$\alpha^{(b)} = \underset{\alpha}{\arg\min} \; e^{\alpha} W + (e^{\alpha} - e^{-\alpha}) W_e$$

– Differentiate w.r.t. α and set the derivative to zero

$$\Rightarrow \quad -\alpha e^{-\alpha} W + \alpha (e^{\alpha} + e^{-\alpha}) W_e = 0$$

$$\Leftrightarrow \quad W = (e^{2\alpha} + 1) W_e$$

$$\Leftrightarrow \quad \alpha = \frac{1}{2} \ln \left(\frac{W}{W_e} - 1 \right)$$

- Optimal value of α : $$\alpha = \frac{1}{2} \ln\left(\frac{W}{W_e} - 1\right)$$

- By defining $$E_{train}^{(b)} = \frac{W_e}{W} = \sum_{i=1}^{N} \frac{w_i^{(b)}}{\sum_{j=1}^{N} w_j^{(b)}} \mathbb{I}\{y_i \neq \hat{y}^{(b)}(\underline{x}_i)\}$$

  to be the weighted misclassification error for the 'b' th classifier

  we can express the optimal value of α as:

$$\alpha^{(b)} = \frac{1}{2} \ln\left(\frac{1 - E_{train}^{(b)}}{E_{train}^{(b)}}\right)$$

- $\alpha^{(b)}$ depends upon the training error of the 'b' th ensemble member

  - Hence, $\alpha^{(b)}$ can be interpreted as the confidence in this member's prediction

- $\alpha^{(1)}, \alpha^{(2)}, \ldots \alpha^{(B)}$ are > 0

AdaBoost Algorithm

Input : Training data $\mathcal{T} = \{ \underline{x}_i , y_i \}_{i=1}^{N}$

Output : 'B' weak classifiers

1) Assign weights $w_i^{(1)} = 1/N$ to all data points

2) for $b = 1, \dots, B$ do

- Train a weak classifier $\hat{y}^{(b)}(\underline{x})$ on the weighted data $\{ \underline{x}_i , y_i , w_i^{(b)} \}_{i=1}^{N}$

- Compute $E_{train}^{(b)} = \sum_{i=1}^{N} w_i^{(b)} \mathbb{I}\{ y_i \neq \hat{y}^{(b)}(\underline{x}_i) \}$

- Compute $\alpha^{(b)} = 0.5 \ln \left(\frac{1 - E_{train}^{(b)}}{E_{train}^{(b)}} \right)$

- Compute $w_i^{(b+1)} = w_i^{(b)} \exp\left(-\alpha^{(b)} y_i \, \hat{y}^{(b)}(\underline{x}_i) \right)$

- Set $w_i^{(b+1)} \leftarrow w_i^{(b+1)} / \sum_{j=1}^{N} w_j^{(b+1)}$ for $i = 1, 2, \dots, N$