Ensemble Methods

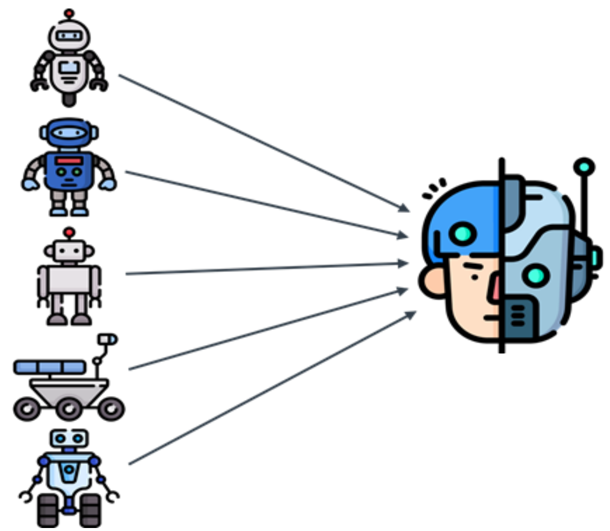
Data Intelligence and Learning (DIAL) Lab
Prof. Jongwuk Lee

Ensemble Method Basics

What is the Ensemble Method?

- **It utilizes the prediction of multiple base models to improve generalizability and robustness over a single model.**

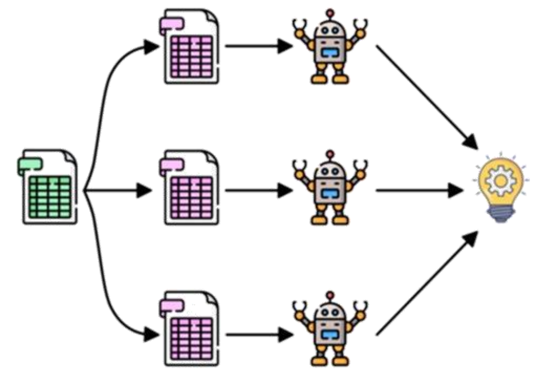
- **They must differ somehow.**
  - Trained with different data
  - Different algorithms
  - Different choices of hyperparameters



- **It is usually easy to implement.**
  - The hard part is how to design ensembles according to the goal.

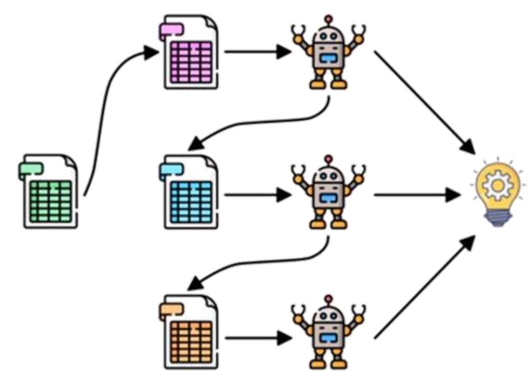
Types of Ensemble Methods

➢ **Bagging**
- It is the short form for **bootstrap aggregating**.
- Train classifiers **independently** on random subsets of training data.
  - E.g., random forest



Parallel

➢ **Boosting**
- Train classifiers **sequentially**.
- Learns from previous predictor mistakes to make better predictions.
  - E.g., AdaBoost, gradient tree boosting

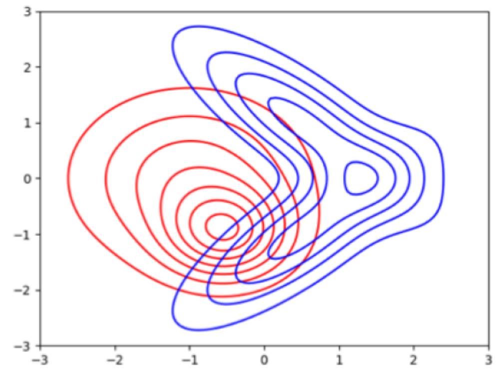


Sequential

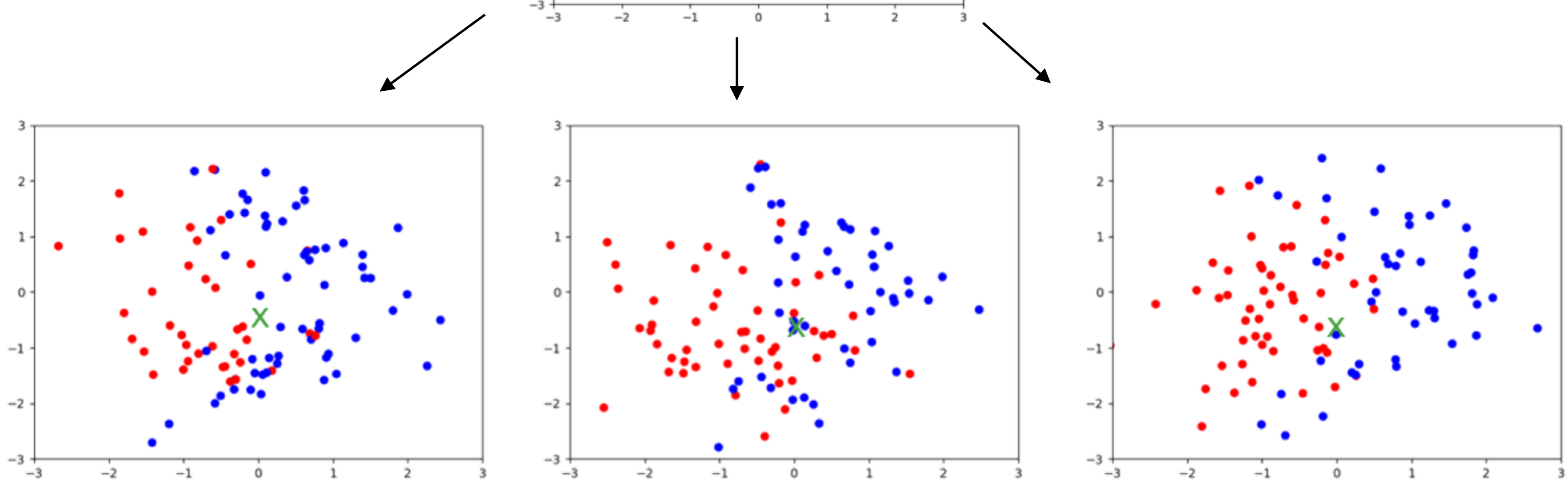
https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422

Recap: Bias-Variance Decomposition

➢ **A training set $\mathcal{D}$ consists of n pairs sampled independent and identically distributed (i.i.d.) from a data distribution p_{data}.**



Sample training sets independently.

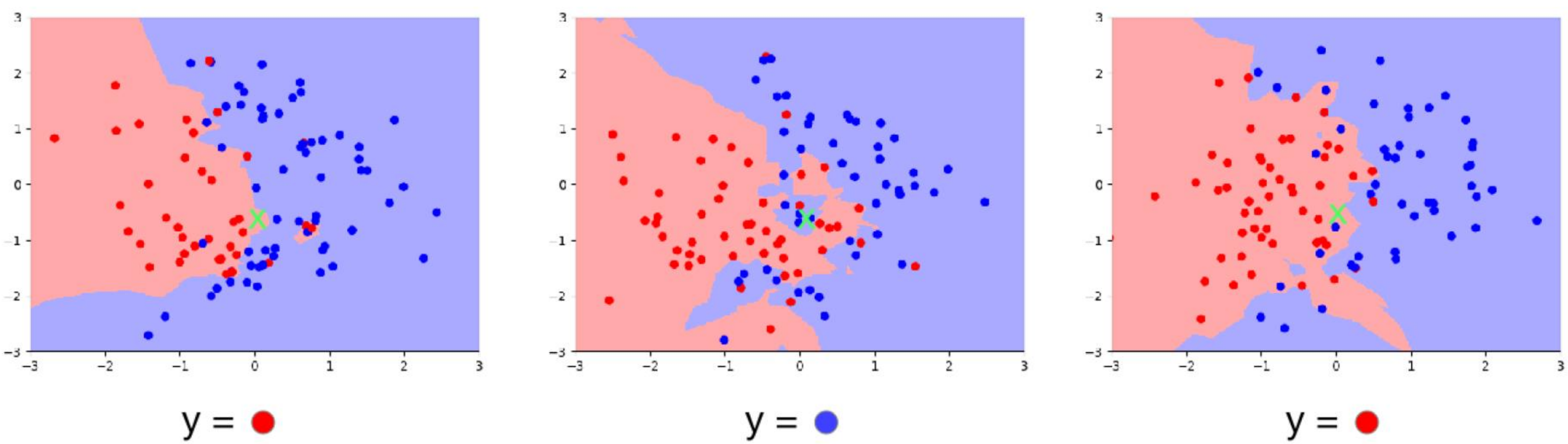


Recap: Bias-Variance Decomposition

- For each training set $\mathcal{D}_i$, we train a classifier h_i.

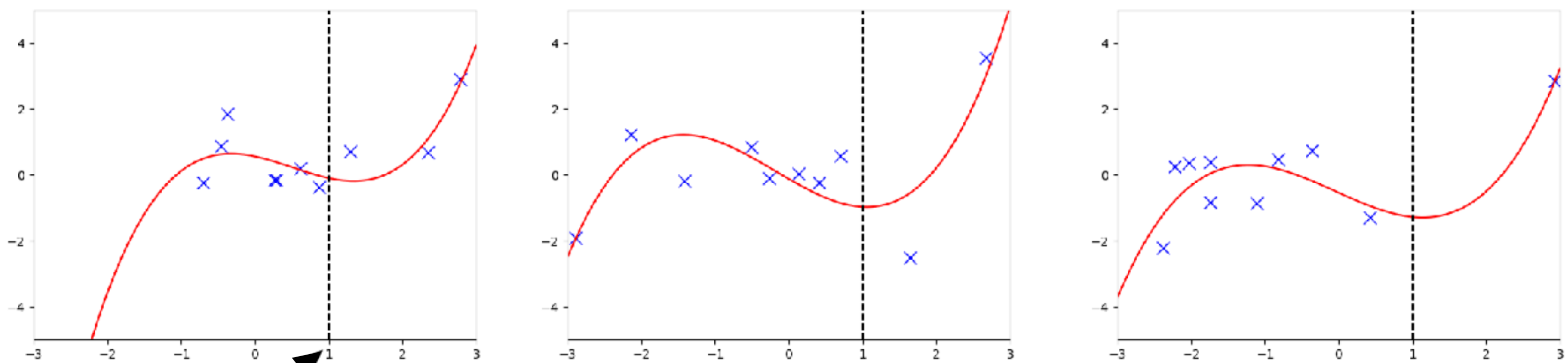
- We make prediction for each classifier $h_i(\mathbf{x})$ at a query $\mathbf{x}$.
  - Pick a fixed query sample $\mathbf{x}$ (denoted with green color).

- Note: Different results come from the choice of training sets.



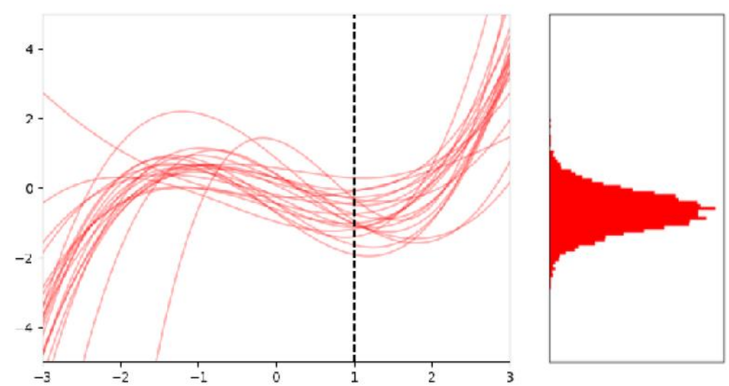
Recap: Bias-Variance Decomposition

- **It is also similar to regression.**
  - That is, $y = h_i(\mathbf{x})$ is a random variable.



Query point

Fitting lots of training sets **Histogram of y**



- **We discuss its expectation and variance over the distribution of the training set.**

Bagging

Motivation

- We sample m independent training sets $\{\mathcal{D}_i\}_{i=1}^{m}$.
- Learn a classifier h_i for each training set $\mathcal{D}_i$ and take the average for prediction.

$$y = \frac{1}{m}\sum_{i=1}^{m} y_i \quad \text{where } y_i = h_i(\mathbf{x})$$

- How does it affect bias and variance of the expected loss?

Motivation

- Bias is unchanged because the average prediction has the same expectation.

$$\mathbb{E}[y] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m}\mathbb{E}\left[\sum_{i=1}^{m} y_i\right] = \mathbb{E}[y_i]$$

- Variance is reduced because of averaging over independent samples.

$$\mathrm{Var}[y] = \mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m^2}\sum_{i=1}^{m}\mathrm{Var}[y_i] = \frac{1}{m}\mathrm{Var}[y_i]$$

Motivation

- **For i.i.d. random variables x_i with mean $\bar{x}$,**

$$\frac{1}{m}\sum_{i=1}^{m} x_i \to \bar{x} \ \text{as}\ m \to \infty$$

(Weak law of large numbers)

- **Learn a classifier h_i for each training set $\mathcal{D}_i$ and take the average.**

$$\hat{h} = \frac{1}{m}\sum_{i=1}^{m} h_i \to \bar{h} \ \text{as}\ m \to \infty$$

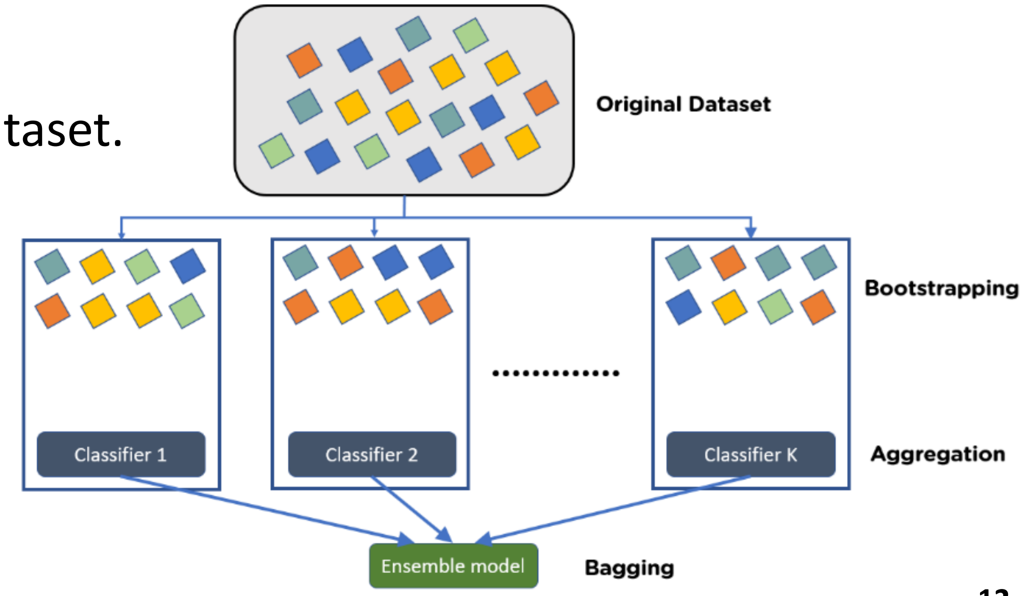
Let $\bar{h}$ denote the prediction for $\mathcal{D}$.

- **Good news: The variance vanishes.**
- **Problem: We do not have m datasets, we only have $\mathcal{D}$.**

Key Idea of Bagging

- **Bootstrap aggregating (Bagging)**
  - Use an **empirical distribution** from a training set $\mathcal{D}$.

- **Overall process**
  - Take a single dataset $\mathcal{D}$ with n samples.
  - Generate m new datasets by sampling n training examples from $\mathcal{D}$ with replacement.
  - Average the predictions of models trained on each dataset.



Bootstrap Sampling

➢ Random sampling with replacement

- Assume that the original data has 10 samples.

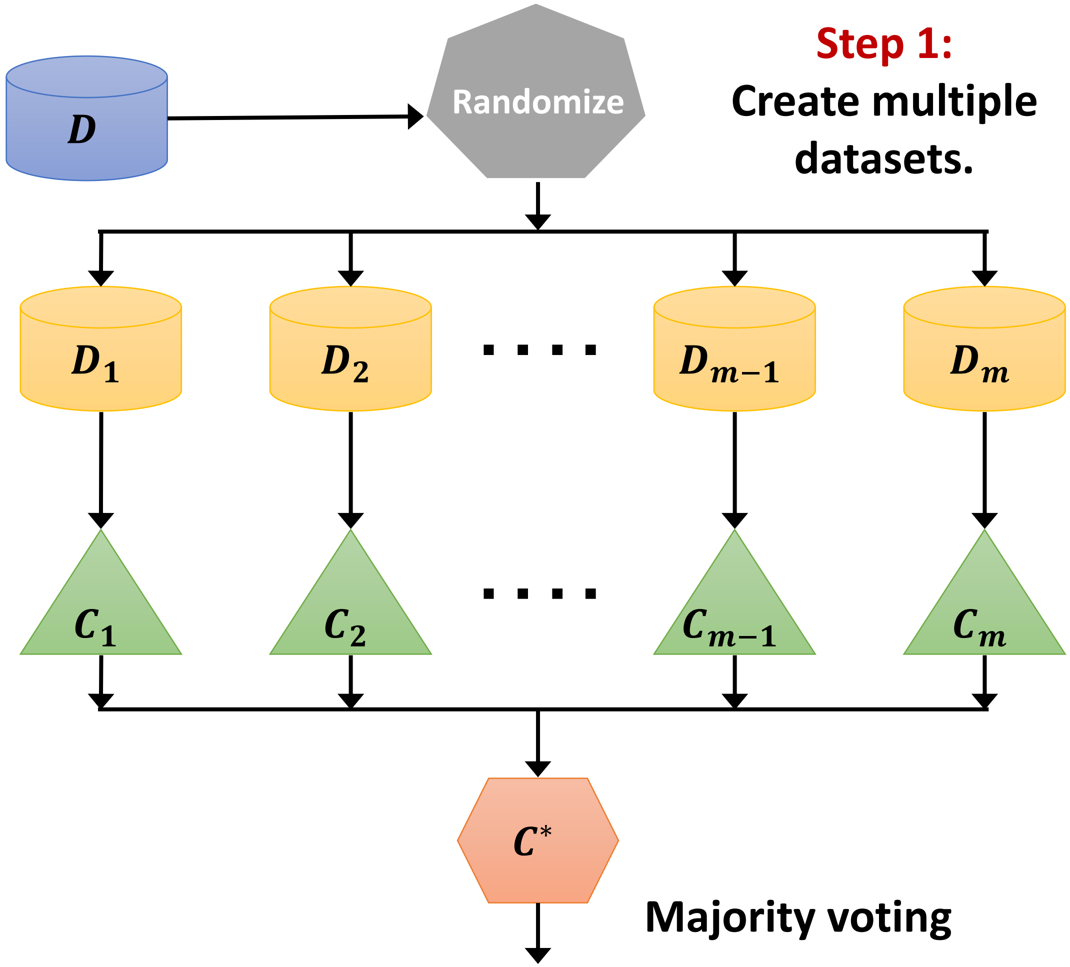
Dataset	1	2	3	4	5	6	7	8	9	10
$\mathcal{D}_1$	7	8	10	8	2	5	10	10	5	9
$\mathcal{D}_2$	1	4	9	1	2	3	2	7	3	2
$\mathcal{D}_3$	1	8	5	10	5	5	9	6	3	7

➢ Building a classifier on each dataset

- Each sample has $(1 - 1/n)^n$ probability that is not selected.
- The expected size of training data: $1 - (1 - 1/n)^n$

Overall Process of Bagging

Original Training data



Step 1: Create multiple datasets.

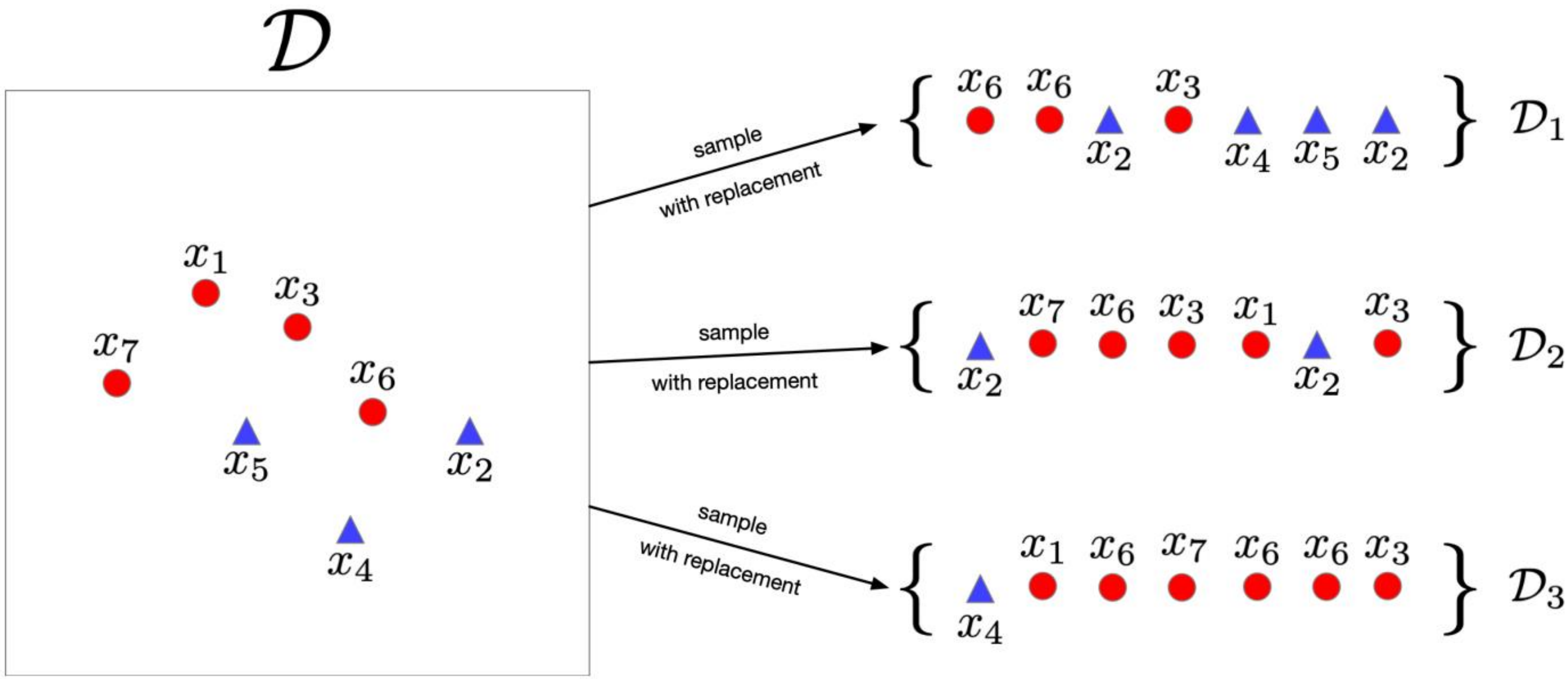
Step 2: Build multiple Classifiers.

Step 3: Combine multiple classifiers.

Majority voting

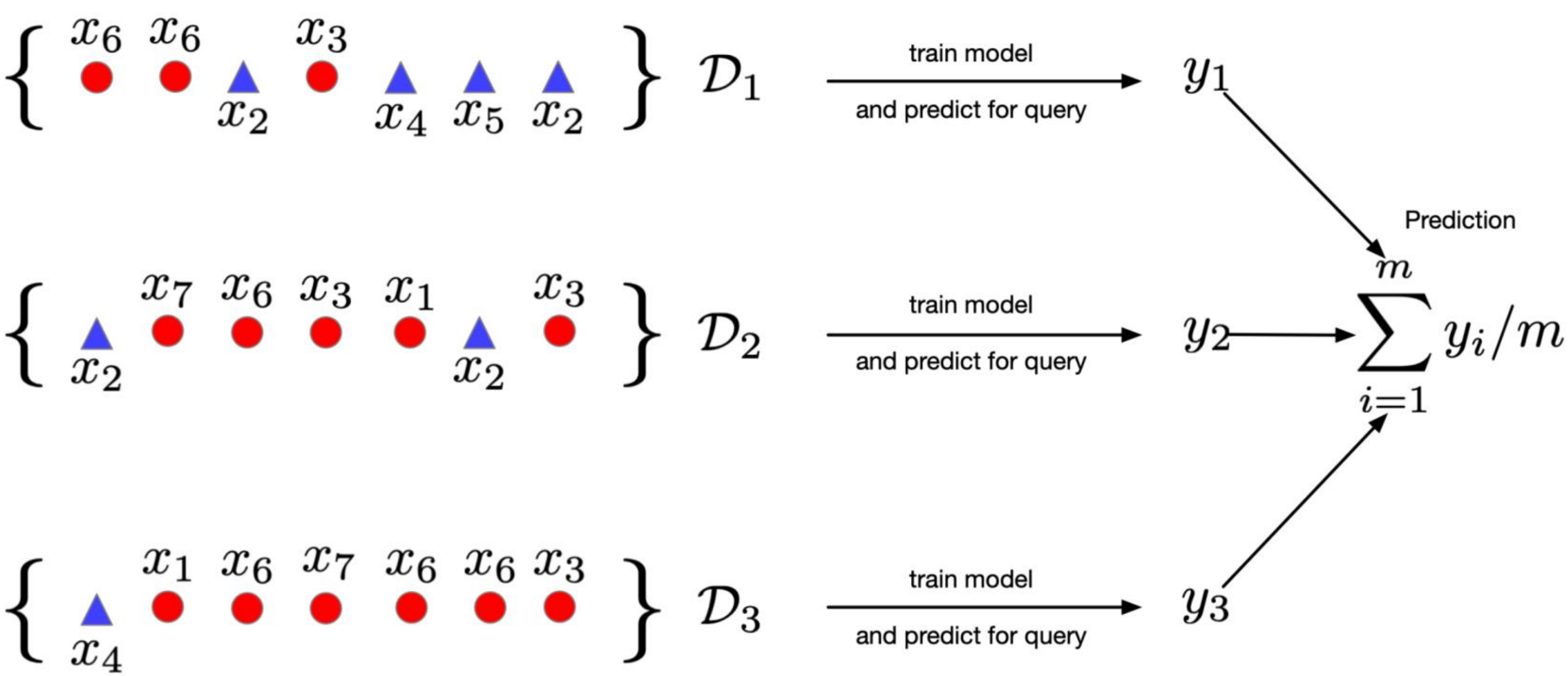
Bagging Example

- In this example, $n = 7, m = 3$



Bagging Example

- **Predicting a new sample x**



Effect of Correlation

- **Problem: The datasets are not perfectly independent, so we do not get the $1/m$ variance reduction.**
  - If the sampled predictions have variance σ^2 and correlation ρ,

$$\mathrm{Var}[y] = \mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m}(1-\rho)\sigma^2 + \rho\sigma^2$$

- **Solution: We introduce additional variability into the model to reduce sample correlation.**
  - It is helpful to use multiple configurations of the same model.

What is Random Forest?

- **It is a bagged decision tree with an extra trick to decorrelate the predictions.**

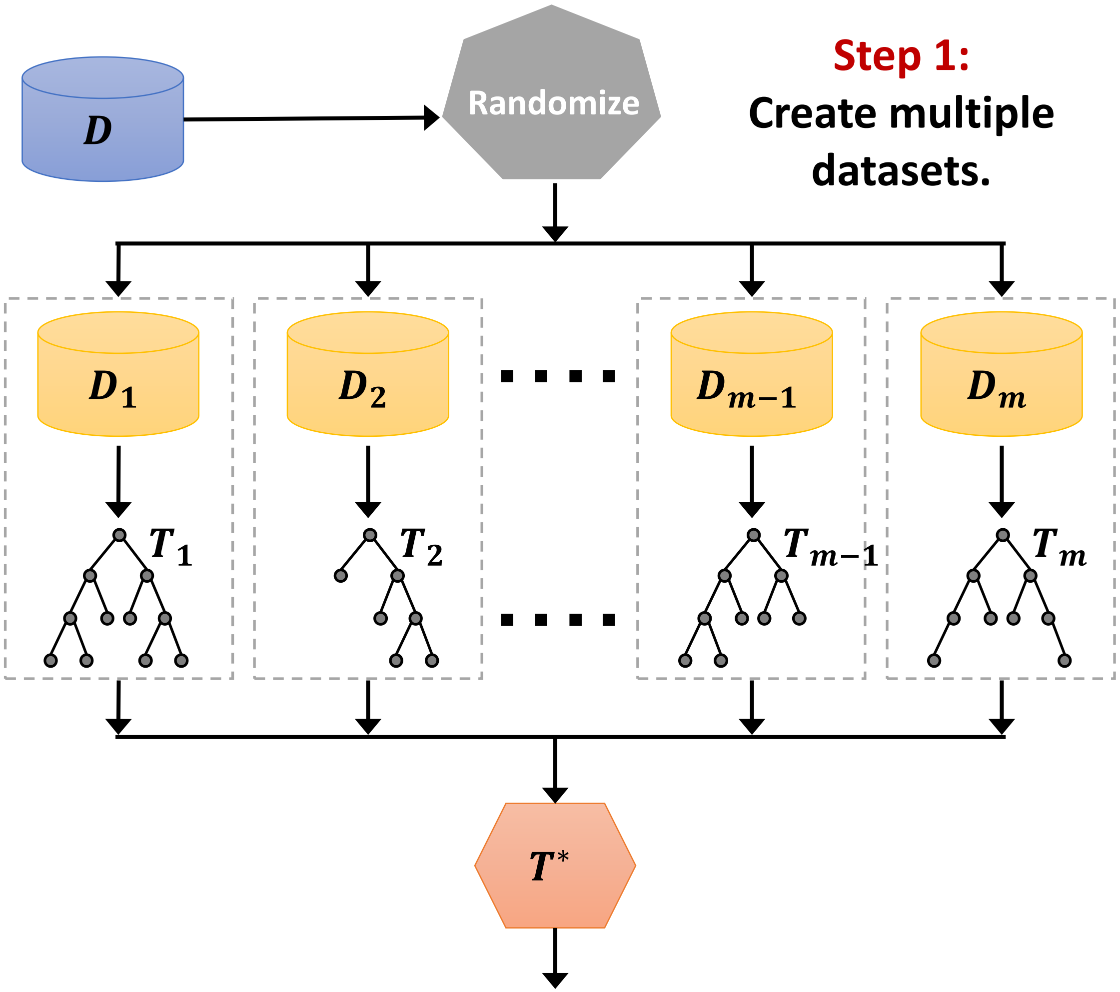
- **Introduce two sources of randomness: Bagging and random feature vectors**
  - **Random dataset**: Each tree is grown using bootstrap sampling.
  - **Random feature vectors**: **At each node, the best split is chosen from a random subset of attributes** instead of all the attributes.

- **Despite the simplicity, it often works well.**
  - It is one of the widely used models in Kaggle competitions.

Overall Process of Random Forest

Original Training data

Step 1:
Create multiple datasets.



Step 2:
Use random features to build each decision trees.

Step 3:
Combine decision trees.

Pseudo-code of Random Forest

Algorithm 1 Random Forest

Precondition: A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, features F, and number of trees in forest B.

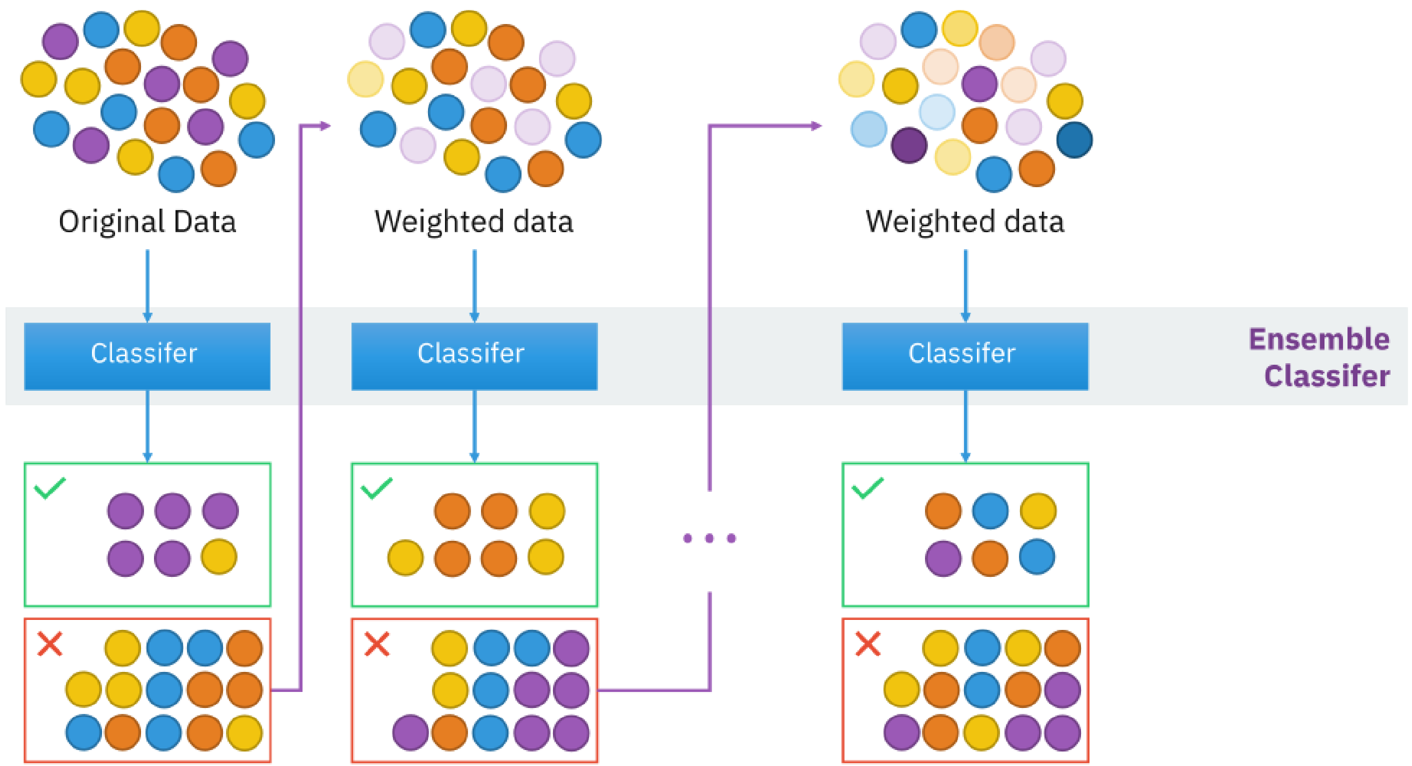
function RandomForest(S, F)
    $H \leftarrow \emptyset$
    for $i \in 1, \ldots, B$ **do**
        $S^{(i)} \leftarrow \boxed{\text{A bootstrap sample from } S}$
        $h_i \leftarrow$ RandomizedTreeLearn($S^{(i)}, F$)
        $H \leftarrow H \cup \{h_i\}$
    end for
    return H
end function
function RandomizedTreeLearn(S, F)
    At each node:
        $\boxed{f \leftarrow \text{very small subset of } F}$
        Split on best feature in f
    return The learned tree
end function

https://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/ensembles/RandomForests.pdf

Boosting

What is Boosting?

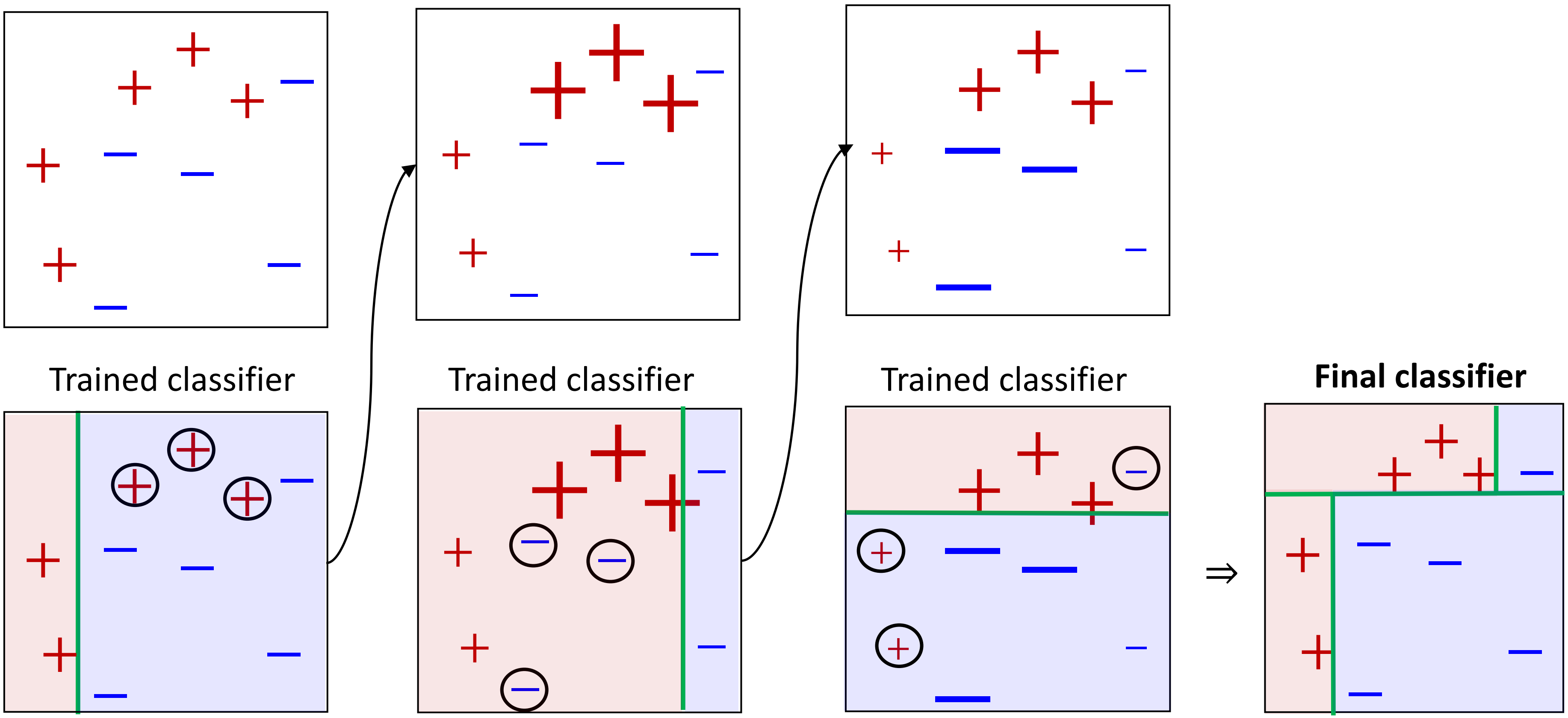
- **Train classifiers sequentially, focusing on training samples that the previous classifiers got wrong.**
  - To focus on specific samples, boosting uses a **weighted training set**.



https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting

Adaptive Boosting (AdaBoost)

- **Combine multiple weak learners.**
  - Learning a **weak learner** for a training dataset.
  - Assigning **larger weights** to **incorrect samples**.



Weighted Training Set

- **Learn a classifier using different weights for samples.**
  - The classifier **tries harder** on samples with **higher costs**.

- **How to change the cost function?**

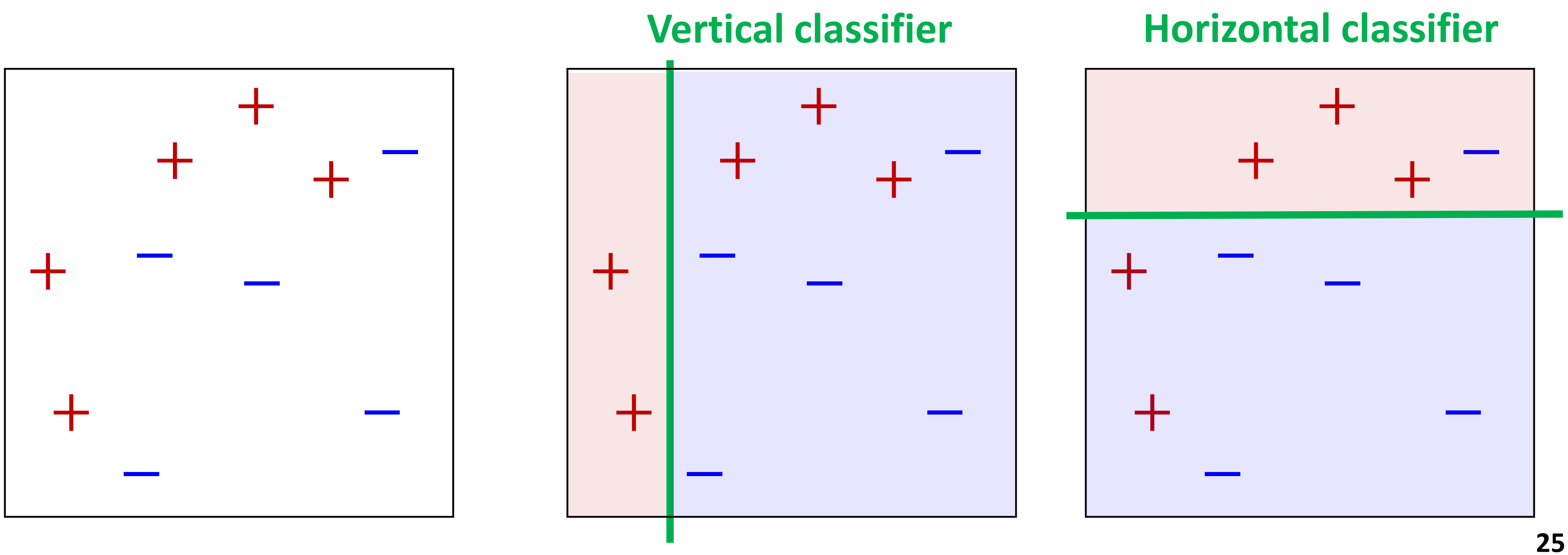
$$\sum_{i=1}^{n} \frac{1}{n} \mathbb{I}[h(x^{(i)} \neq y^{(i)}] \text{ becomes } \sum_{i=1}^{n} w^{(i)} \mathbb{I}[h(x^{(i)}) \neq y^{(i)}]$$

- **Usually, it requires $w^{(i)} > 0$ and $\sum_{i=1}^{n} w^{(i)} = 1$.**

Weak Classifier (or Learner)

- **It is a simple learning model that outputs a hypothesis that performs slightly better than a chance.**

- **Consider weak learners that are computationally efficient.**
  - E.g., decision trees
  - Even simpler: **A decision tree with a single split**.



Overall Process of AdaBoost

➢ Notations

- Input: $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}): 1 \le i \le n\}$, where $y^{(i)} \in \{-1, +1\}$
- A classifier or hypothesis function $h: \mathbf{x} \to \{-1, +1\}$
- 0-1 loss function: $\mathbb{I}[h(x^{(i)}) \neq y^{(i)}]$

➢ It learns weak learners sequentially until T times.

- For each iteration, it **learns weak learners**.
- It computes the **weighted error** and the **classifier coefficient**.
- Using the classifier coefficient, it updates **data weights**.

Overall Process of AdaBoost

- **Initialize sample weights:** $w^{(i)} = 1/n$ **for** $i = 1, \dots, n$.
- **For each iteration** $t = 1, \dots, T$
  - Fit a classifier to weighted data.

$$h_t \leftarrow \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} w^{(i)} \mathbb{I}\left[h\left(\mathbf{x}^{(i)}\right) \neq y^{(i)}\right]$$

  - Compute the weighted error and classifier coefficient.

$$\mathrm{err}_t = \frac{\sum_{i=1}^{n} w^{(i)} \mathbb{I}\left[h\left(\mathbf{x}^{(i)}\right) \neq y^{(i)}\right]}{\sum_{i=1}^{n} w^{(i)}}$$

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \mathrm{err}_t}{\mathrm{err}_t}$$

  - Update sample weights.

$$w^{(i)} \leftarrow w^{(i)} \exp\left(-\alpha_t y^{(i)} h_t\left(\mathbf{x}^{(i)}\right)\right)$$

- **Return** $H(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t \, h_t(\mathbf{x})\right)$.

Updating Sample Weights

- **Reassign higher weights for incorrect samples.**
  - Assume that the weak learner returns error $\mathrm{err}_t < 0.5$.

$$\alpha_t = \frac{1}{2}\ln\frac{1-\mathrm{err}_t}{\mathrm{err}_t} = \ln\sqrt{\frac{1-\mathrm{err}_t}{\mathrm{err}_t}}$$

- **If the sample is incorrect,**

$$w^{(i)} \leftarrow w^{(i)}\exp(\alpha_t) = w^{(i)}\sqrt{\frac{1-\mathrm{err}_t}{\mathrm{err}_t}}$$

Weights are higher.

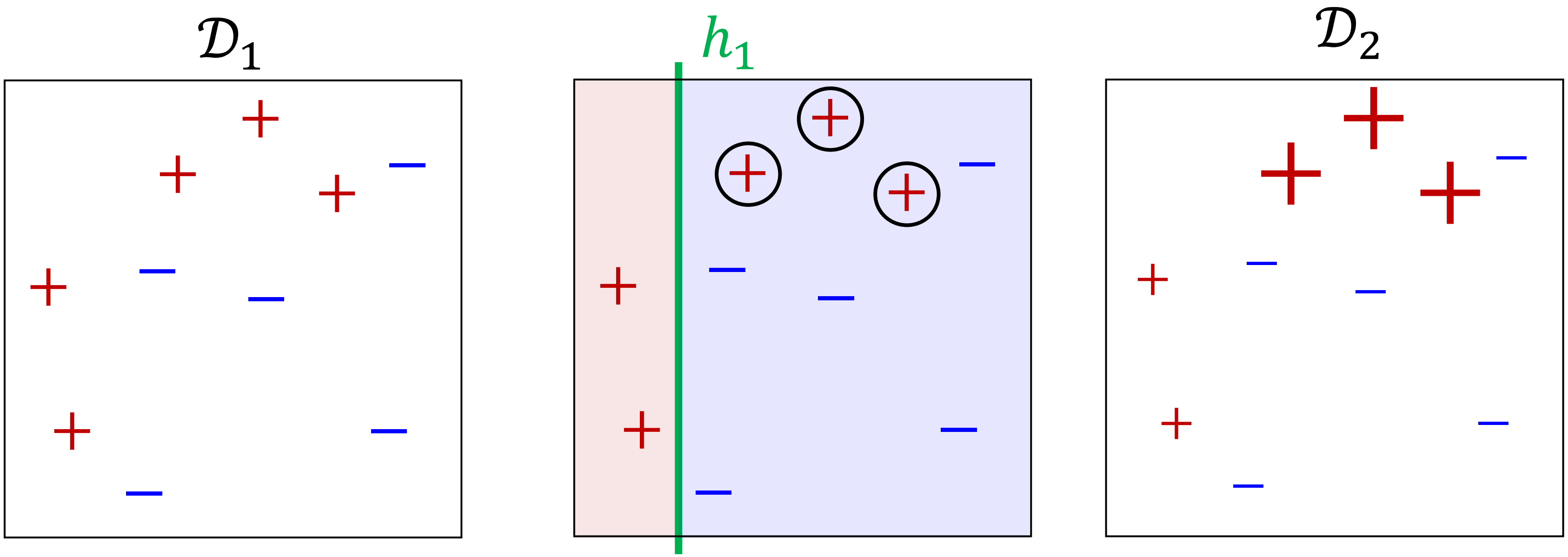
- **Otherwise,**

$$w^{(i)} \leftarrow w^{(i)}\exp(\alpha_t) = w^{(i)}\sqrt{\frac{\mathrm{err}_t}{1-\mathrm{err}_t}}$$

Weights are lower.

AdaBoost Example

- **Round 1**



$$\mathbf{w} = \left(\frac{1}{10}, \ldots, \frac{1}{10}\right)$$

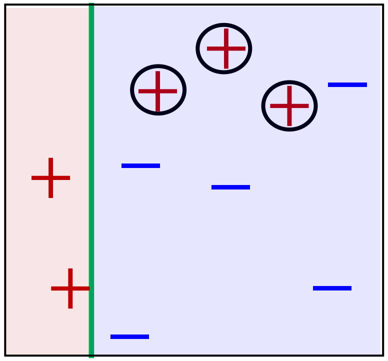
$$\mathrm{err}_1 = 0.3 \qquad \alpha_1 = \frac{1}{2}\ln\left(\frac{1-0.3}{0.3}\right) = \frac{1}{2}\ln\frac{7}{3} \approx 0.42$$

$$\mathrm{err}_t = \frac{\sum_{i=1}^{n} w^{(i)} \mathbb{I}\left[h\left(x^{(i)}\right) \neq y^{(i)}\right]}{\sum_{i=1}^{n} w^{(i)}}$$

$$\alpha_t = \frac{1}{2}\ln\frac{1-\mathrm{err}_t}{\mathrm{err}_t}$$

AdaBoost Example

- **How to update sample weights?**

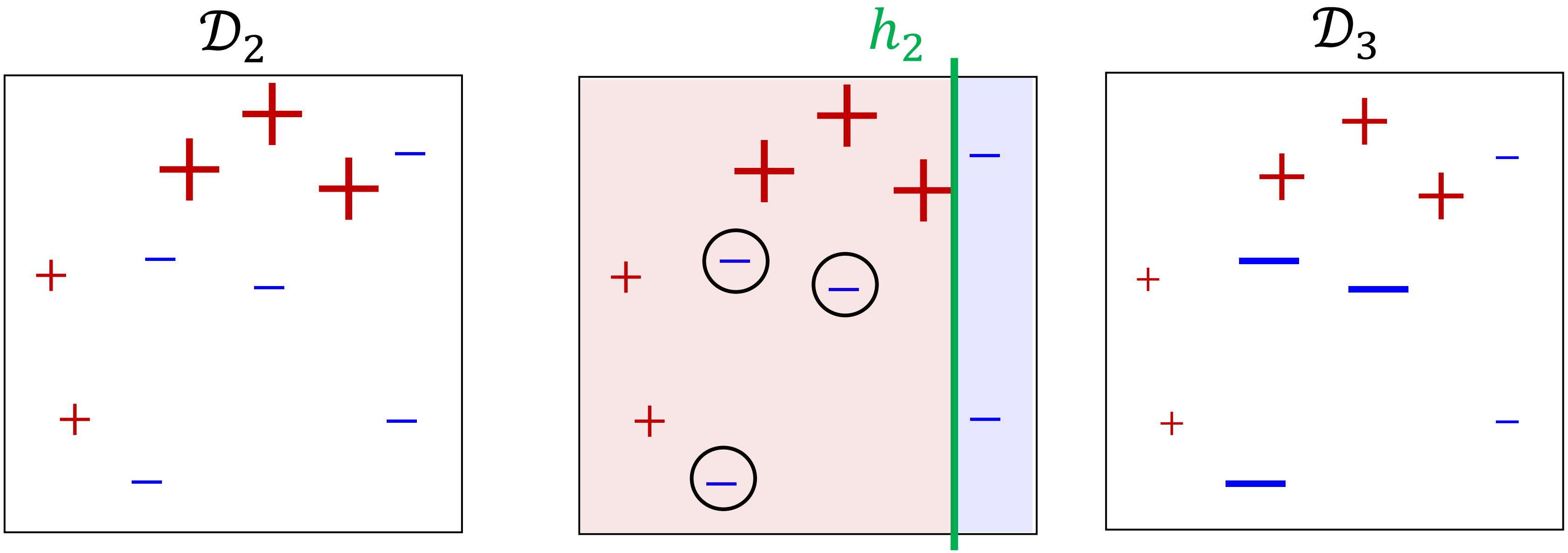


	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
y	+	+	−	−	−	+	+	+	−	−
$\hat{y}$	+	+	−	−	−	−	−	−	−	−

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
Original	$1/10$	$1/10$	$1/10$	$1/10$	$1/10$	$1/10$	$1/10$	$1/10$	$1/10$	$1/10$
Update	$1/10 * \sqrt{3/7}$	$1/10 * \sqrt{3/7}$	$1/10 * \sqrt{3/7}$	$1/10 * \sqrt{3/7}$	$1/10 * \sqrt{3/7}$	$1/10 * \sqrt{7/3}$	$1/10 * \sqrt{7/3}$	$1/10 * \sqrt{7/3}$	$1/10 * \sqrt{7/3}$	$1/10 * \sqrt{7/3}$
Update	3	3	3	3	3	7	7	7	3	3
Normalized	$1/14$	$1/14$	$1/14$	$1/14$	$1/14$	$1/6$	$1/6$	$1/6$	$1/14$	$1/14$

AdaBoost Example

- ➢ **Round 2**

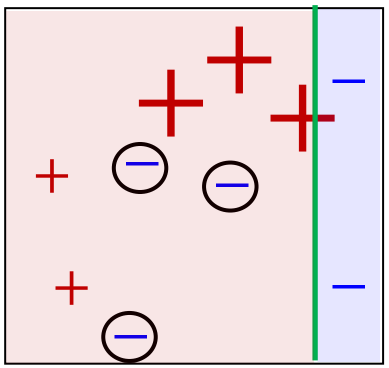


$$\text{err}_2 = 0.21$$

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-0.21}{0.21}\right) \approx 0.66$$

AdaBoost Example

- ➢ **How to update sample weights?**

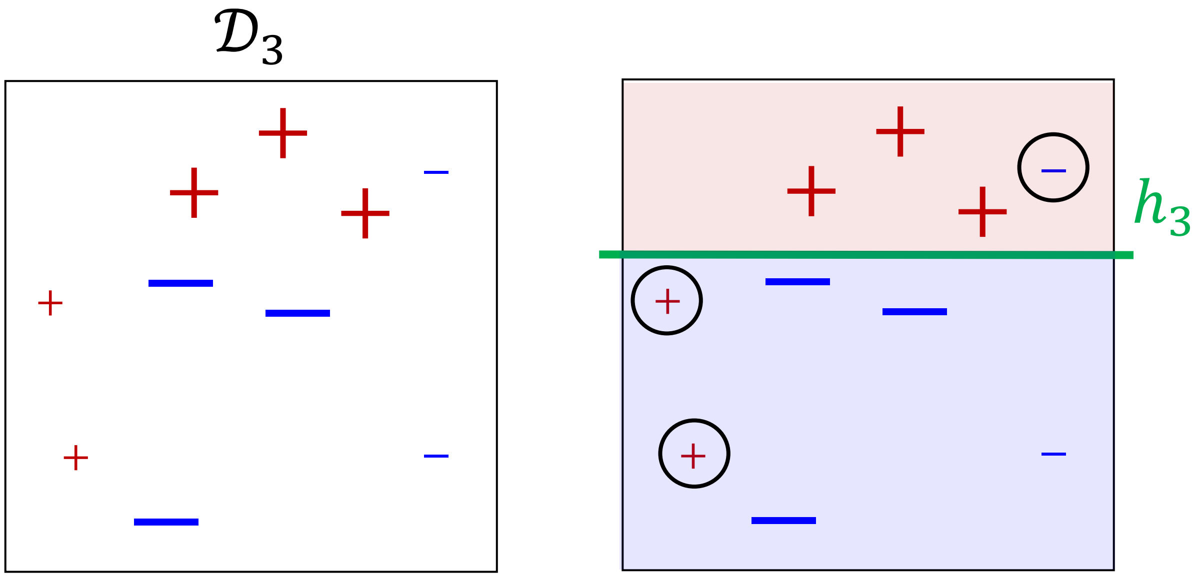


	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
y	+	+	−	−	−	+	+	+	−	−
$\hat{y}$	+	+	+	+	+	+	+	+	−	−

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
Original	$1/14$	$1/14$	$1/14$	$1/14$	$1/14$	$1/6$	$1/6$	$1/6$	$1/14$	$1/14$
Update	$1/14 * \sqrt{3/11}$	$1/14 * \sqrt{3/11}$	$1/14 * \sqrt{11/3}$	$1/14 * \sqrt{11/3}$	$1/14 * \sqrt{11/3}$	$1/6 * \sqrt{3/11}$	$1/6 * \sqrt{3/11}$	$1/6 * \sqrt{3/11}$	$1/14 * \sqrt{3/11}$	$1/14 * \sqrt{3/11}$
Normalized	0.045	0.045	0.167	0.167	0.167	0.106	0.106	0.106	0.045	0.045

AdaBoost Example

- **Round 3**

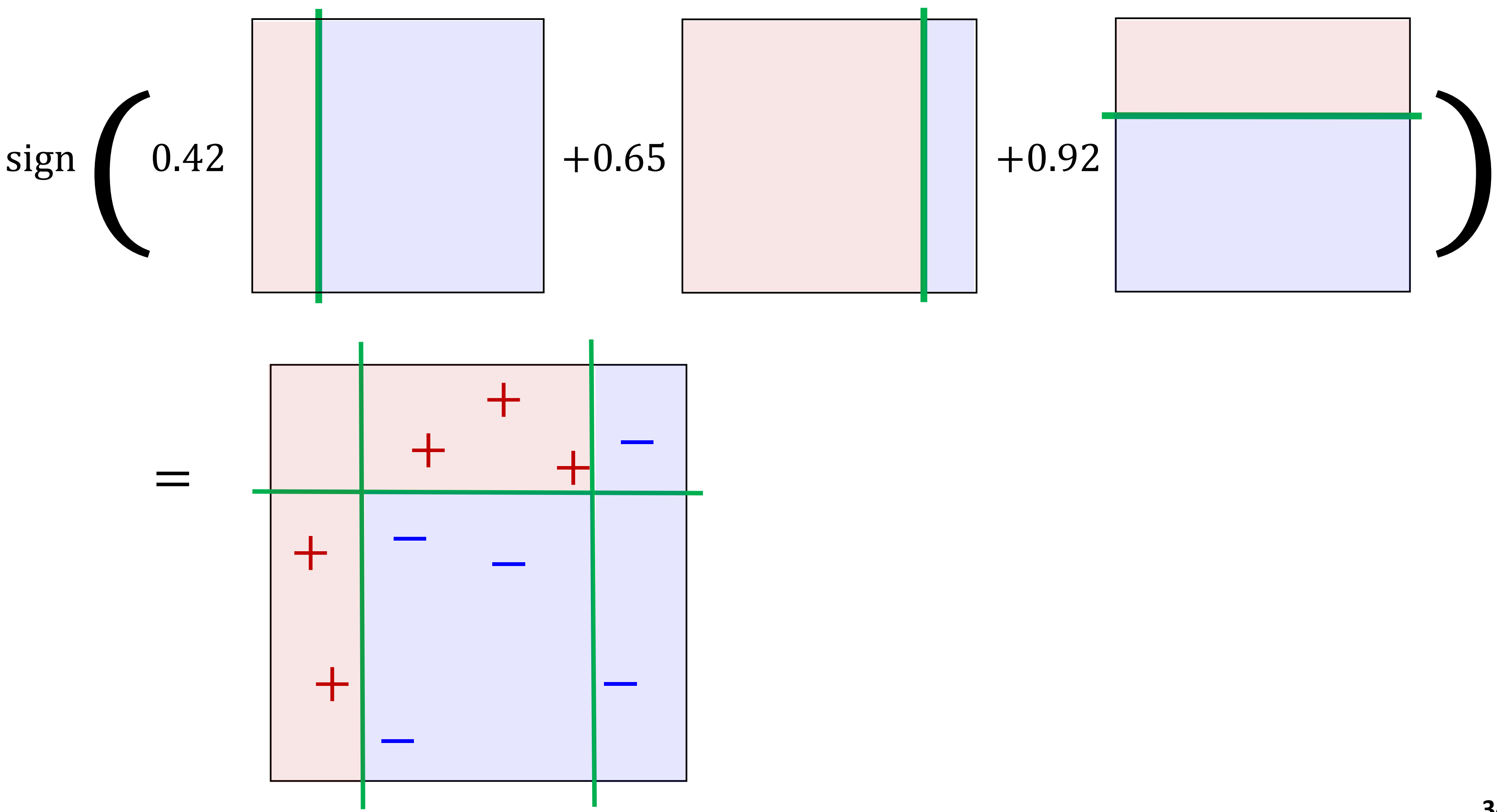


$$\text{err}_3 = 0.14$$

$$\alpha_3 = \frac{1}{2}\ln\left(\frac{1-0.14}{0.14}\right) \approx 0.92$$

AdaBoost Example

- **Final classifier**



AdaBoost Minimizes the Training Error

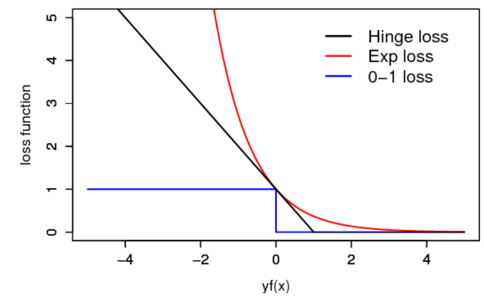
- **AdaBoost assumes that each weak learner is γ-better than a random predictor.**

Theorem

Assume that at each iteration of AdaBoost the WeakLearn returns a hypothesis with error $\text{err}_t \le \frac{1}{2} - \gamma$ for all $t = 1, \ldots, T$ with $\gamma > 0$. The training error of the output hypothesis $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$ is at most

$$L_N(H) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\{H(\mathbf{x}^{(i)}) \neq t^{(i)})\} \le \exp\left(-2\gamma^2 T\right).$$

- **It is called geometric convergence. It is fast!**
  - The error decreases with the number of iterations.

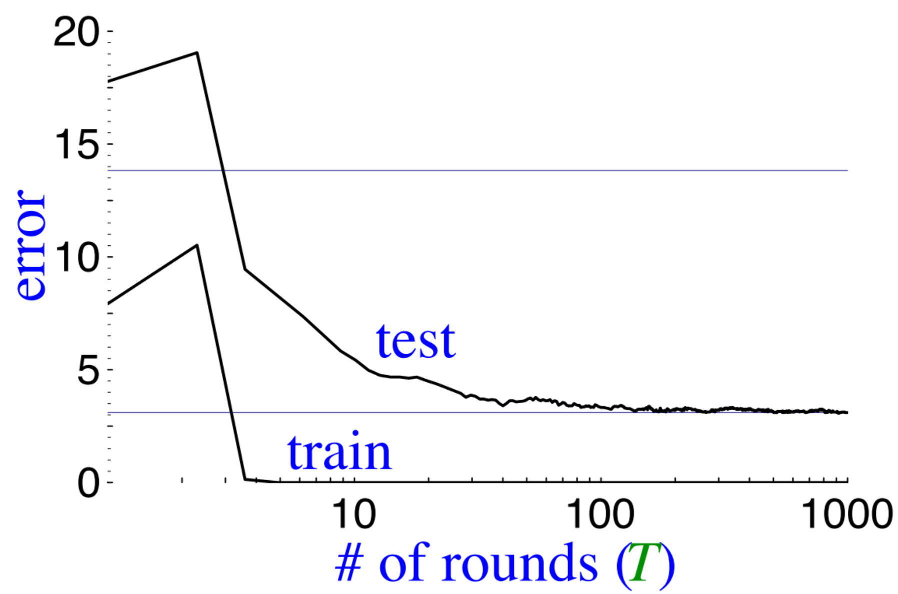


Generalization Error of AdaBoost

- **The training error of AdaBoost converges to zero.**
  - What about the test error?

- **As we add weak classifiers, the overall classifier becomes more complex.**
  - We expect that the complex classifier may overfit.

- **But often, it does not.**
  - Sometimes, the **test error decreases** even after the training error is zero!

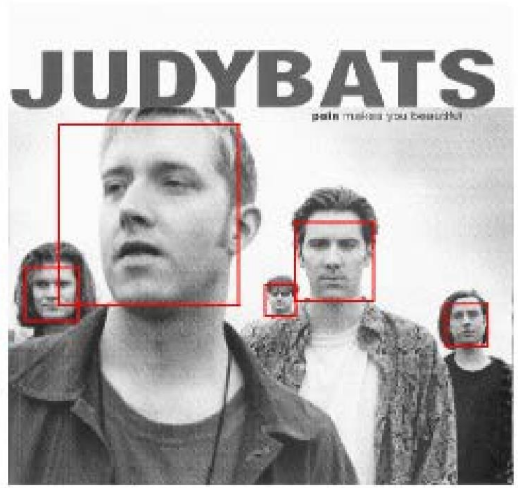


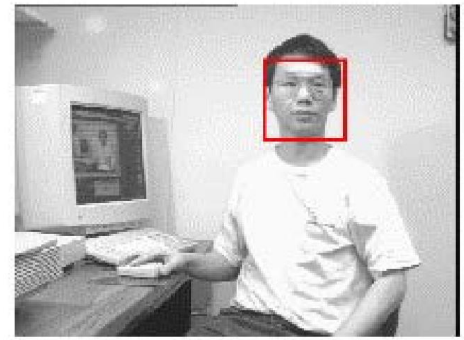
Face Detection using AdaBoost

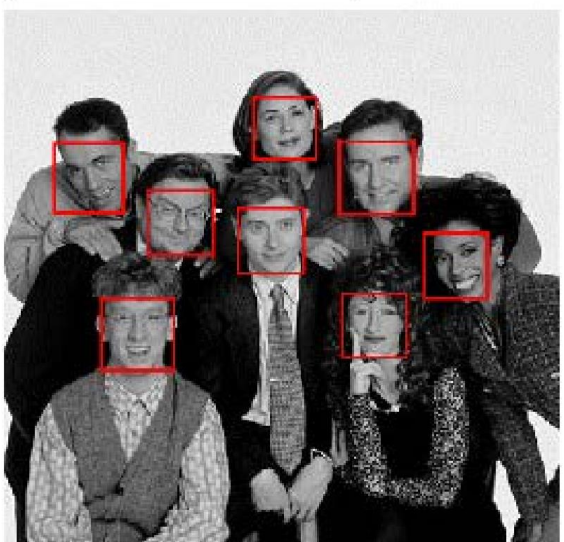
- **Famous application of boosting: detecting faces in images**
  - The weak learner compares **the total intensity in two rectangular pieces of the image**.
    - It is easy to evaluate many base classifiers, which are very fast at runtime.
    - A Smart way to make inferences in real-time (in 2001 hardware).
  - The algorithm adds classifiers **greedily** based on **the quality of the weighted training cases**.

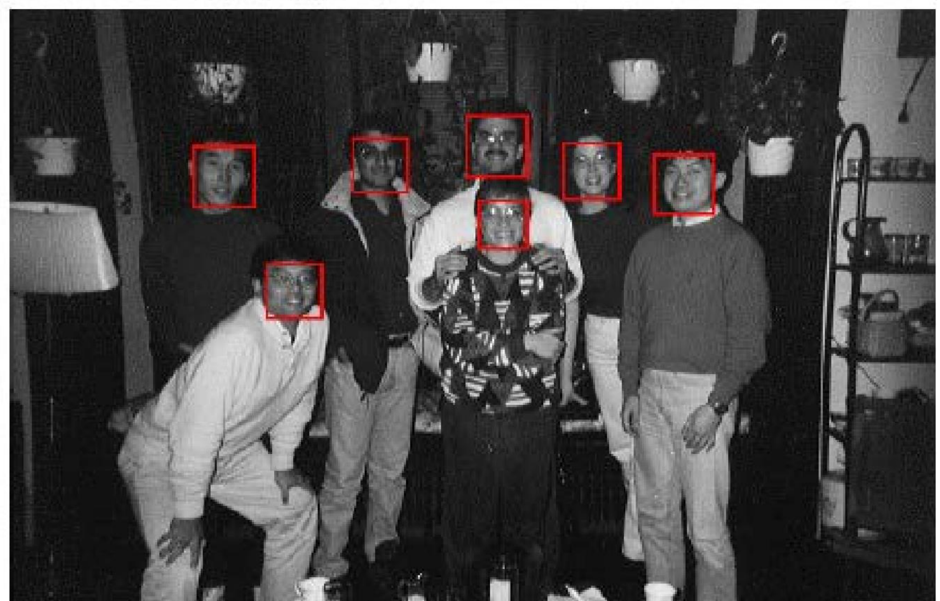
Rectangular filters

Face Detection Results using AdaBoost









Summary

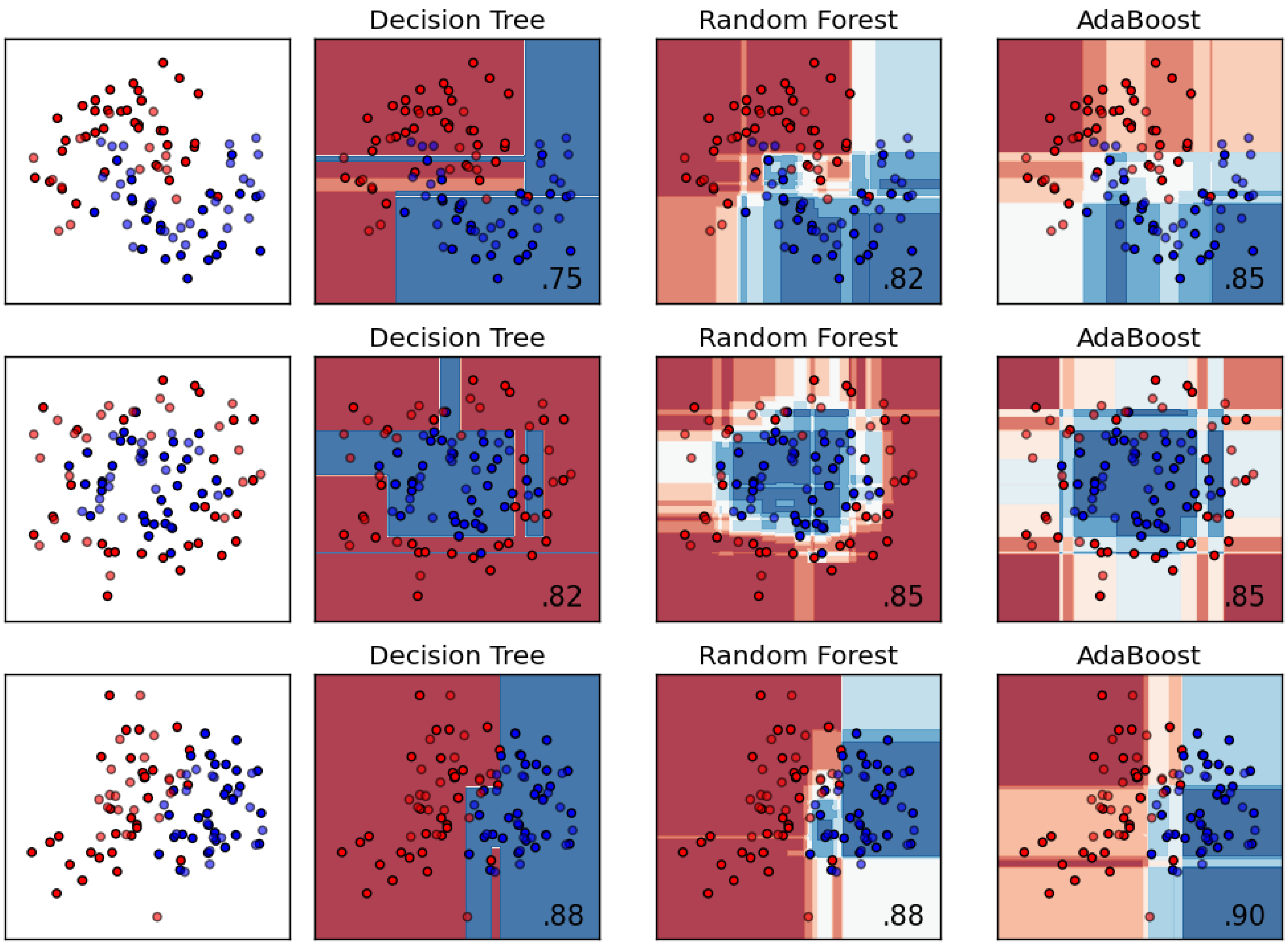
- **Ensembles combine classifiers with improving performance.**

- **Bagging**
  - Bias is not changed (much.)
  - Reduces variance (large ensemble cannot cause overfitting.)
  - Learns ensemble elements sequentially.
  - Needs to minimize the correlation between ensemble elements.

- **Boosting**
  - Reduces bias.
  - Increases variance (large ensemble can cause overfitting.)
  - Learns ensemble elements in parallel.
  - Has a high dependency on ensemble elements.

Q&A

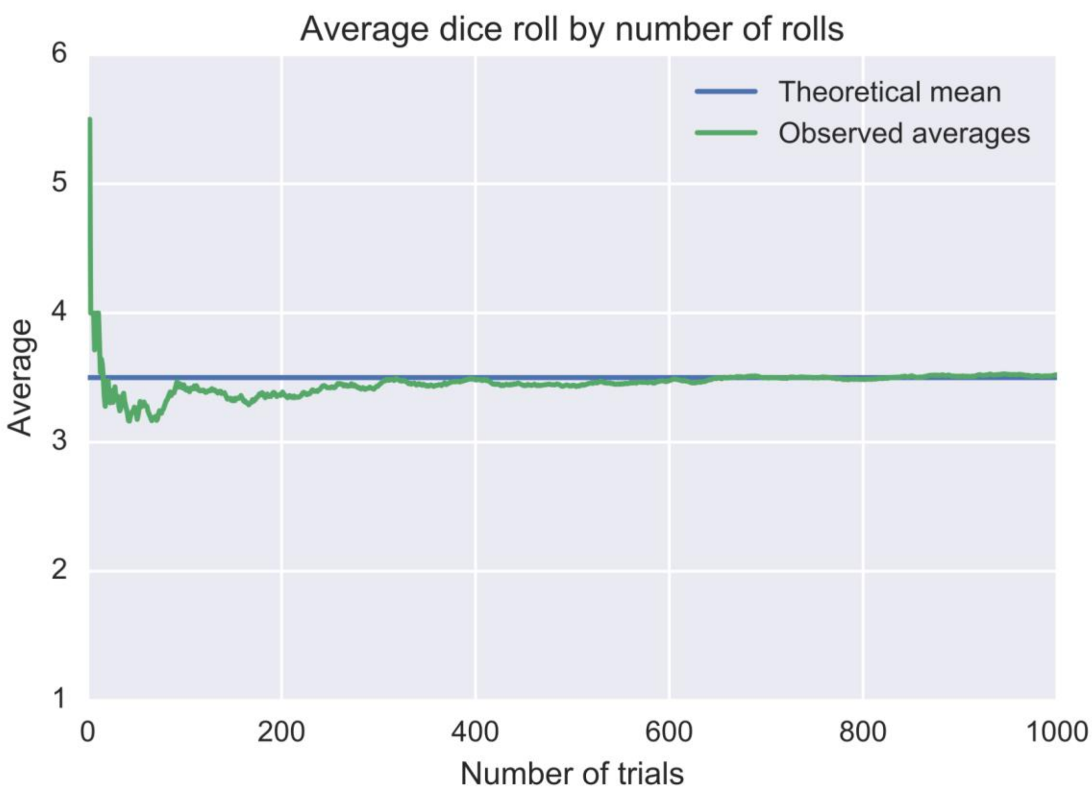
Random Forest vs. AdaBoost



https://martin-thoma.com/comparing-classifiers/

Law of Large Numbers (LLN)

- If we randomly draw independent observations from **any population** with a finite mean μ, the **sample mean $\overline{x}$** of the observed values approaches the true mean μ of the population as **the number of observations goes to ∞**.



Example: Law of Large Numbers

- We generated a normally distributed variable with mean $\mu = 100$ and standard deviation $\sigma = 4$. To obtain each plotted point, we calculate the mean up to each n.