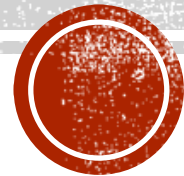


Indian Institute of Information Technology, Allahabad



Introduction to Machine Learning Ensemble Methods



By

Dr. Shiv Ram Dubey

Associate Professor

Computer Vision and Biometrics Lab

Department of Information Technology

Indian Institute of Information Technology, Allahabad

Web: <https://profile.iita.ac.in/srdubey/>

DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

Decision Tree Shortcomings

- Hard to manage bias-variance tradeoff
 - Small depth $\rightarrow$ High bias, low variance
 - Large depth $\rightarrow$ Small bias, high variance

Decision Tree Shortcomings

- Hard to manage bias-variance tradeoff
 - Small depth $\rightarrow$ High bias, low variance
 - Large depth $\rightarrow$ Small bias, high variance
- Can we manage this tradeoff in a more principled way?
- **Idea: Can we use model combination to control the trade-off more gracefully?**

General mechanism for reducing variance in a, almost always, model agnostic way

Ensemble Learning

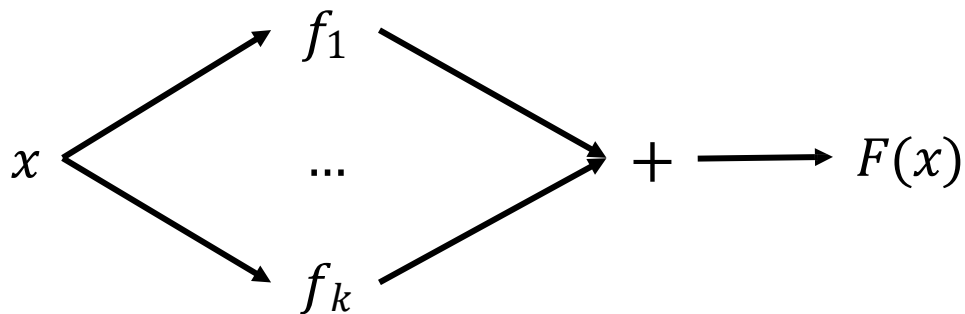
- **Step 1:** Learn a set of “base” models $f_1, \dots, f_k$
- **Step 2:** Construct model $F(x)$ that combines predictions of $f_1, \dots, f_k$

Ensemble Design Decisions

- How to learn the base models?
 - Main goal: establish diversity
- How to combine the learned base models?

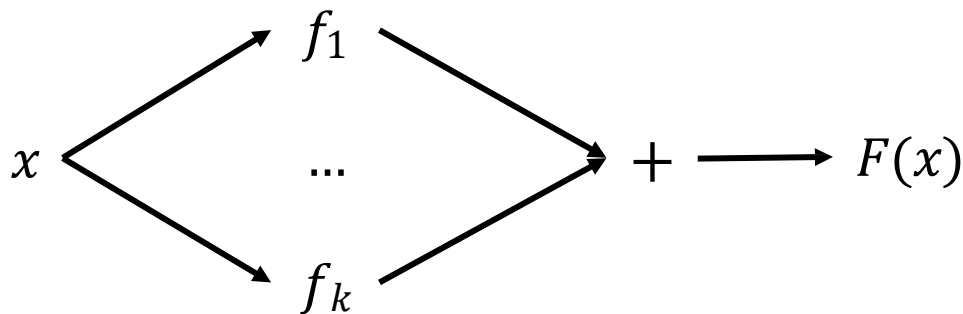
Combining Learned Base Models

- **Regression:** Average predictions $F(x) = \frac{1}{k} \sum_{i=1}^k f_i(x)$
 - Works well if the base models have similar performance



Combining Learned Base Models

- **Classification:** Majority vote $F(x) = 1 \left(\frac{1}{k} \sum_{i=1}^k f_i(x) \geq \frac{1}{2} \right)$ (for binary)
 - Can also average probabilities for classification



Combining Learned Base Models

- Can use weighted average:

$$F(x) = \sum_{i=1}^k \beta_i \cdot f_i(x)$$

- Can fit weights using linear regression on second training set
- More generally, can fit a second layer model

$$F(x) = g_{\beta}(f_1(x), \dots, f_k(x))$$

Combining Learned Base Models

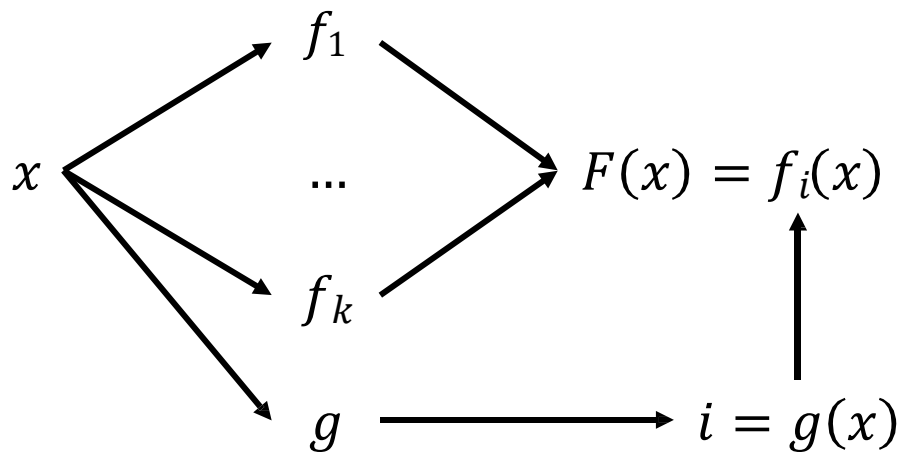
- Second model as “mixture of experts”:

$$F(x) = \sum_{i=1}^k g(x)_i \cdot f_i(x)$$

- Second stage model predicts weights over “experts” $f_i(x)$

Combining Learned Base Models

- Second model as “mixture of experts”:
 - **Special case:** $g(x)$ is one-hot
 - **Advantage:** Only need to run $g(x)$ and $f_{g(x)}(x)$



Example: Netflix Movie Recommendations

- **Goal:** Predict how a user will rate a movie based on:
 - The user's ratings for other movies
 - Other users' ratings for this movie (and others)
- **Netflix Prize (2007-2009):** \$1 million for the first team to do 10% better than the existing Netflix recommendation system
- **Winner:** BellKor's Pragmatic Chaos
 - An ensemble of 800+ rating systems

Ensemble Design Decisions

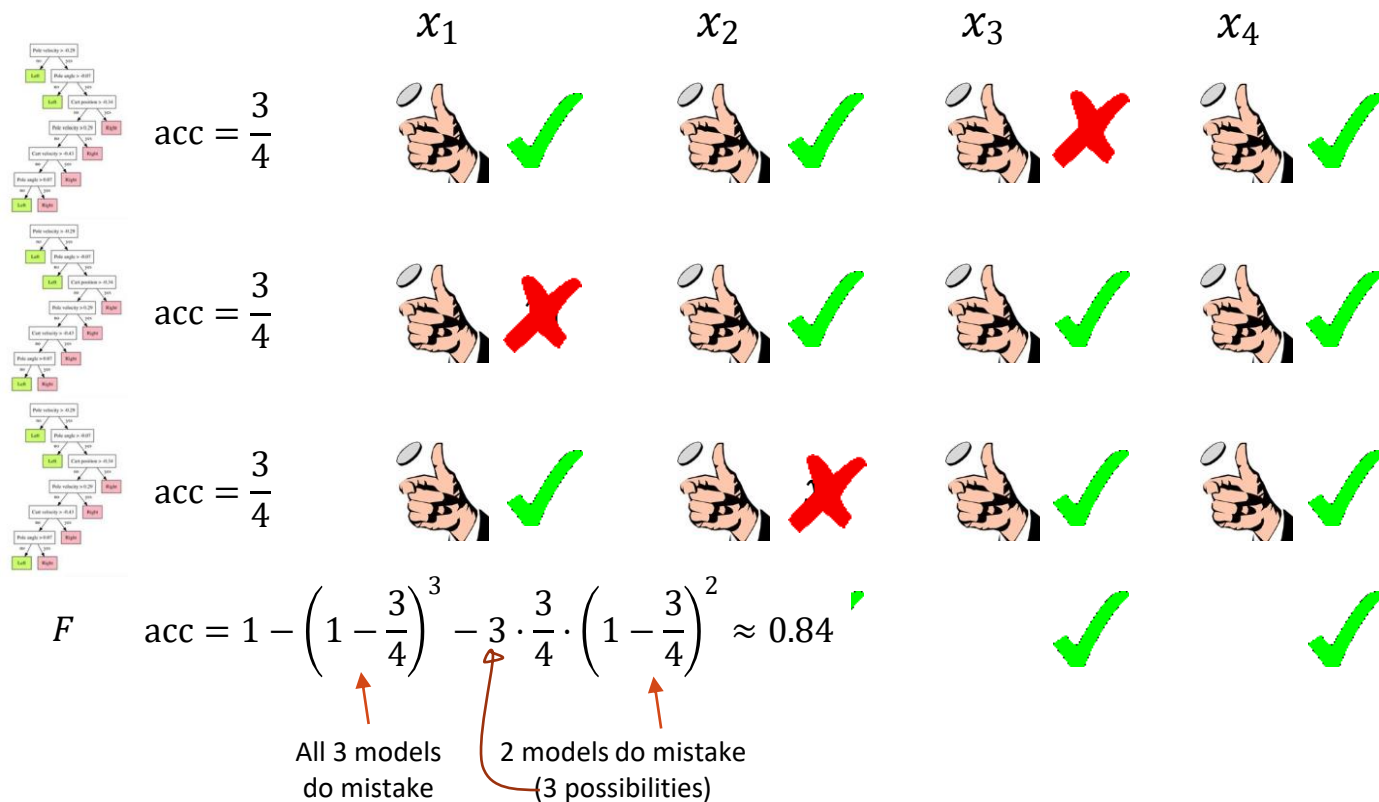
- How to learn the base models?
- How to combine the learned base models?

Learning Base Models

- Successful ensembles require **diversity**
 - Different model families
 - Different training data
 - Different features
 - Different hyperparameters
- **Intuition:** Models should make **independent** mistakes

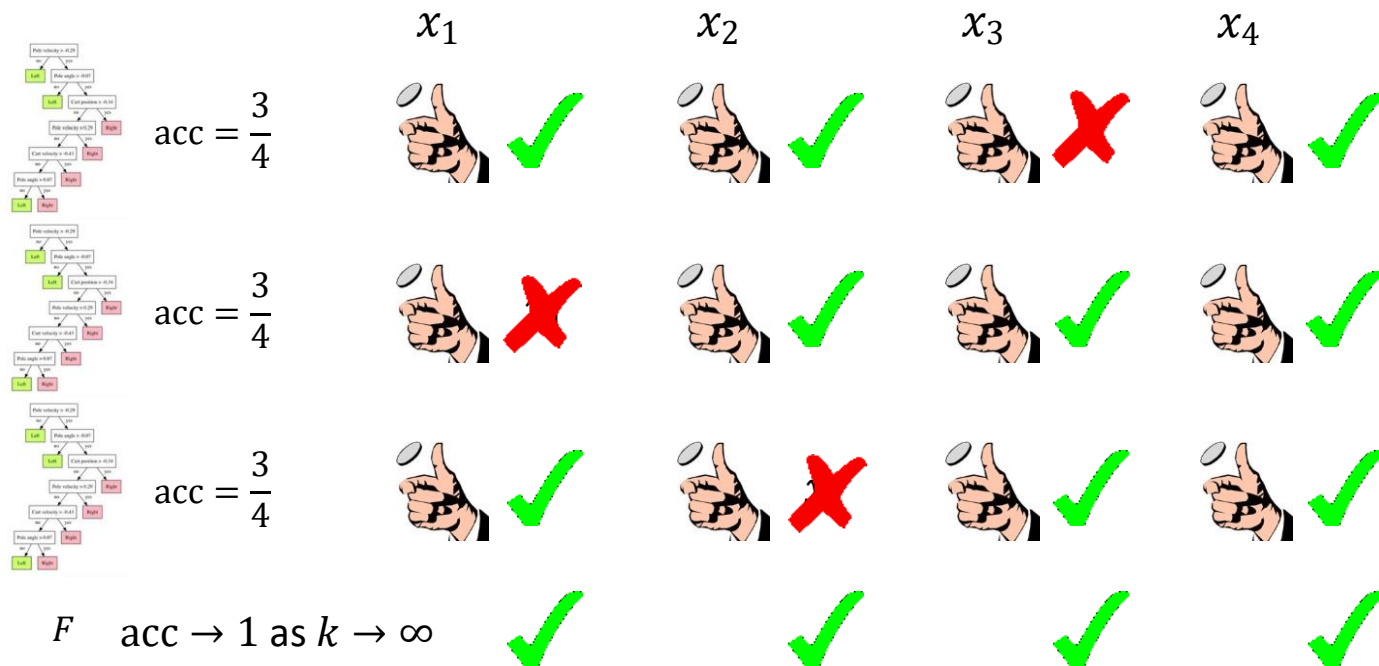
Learning Base Models

- **Intuition:** Models should make **independent** mistakes



Learning Base Models

- **Intuition:** Models should make **independent** mistakes



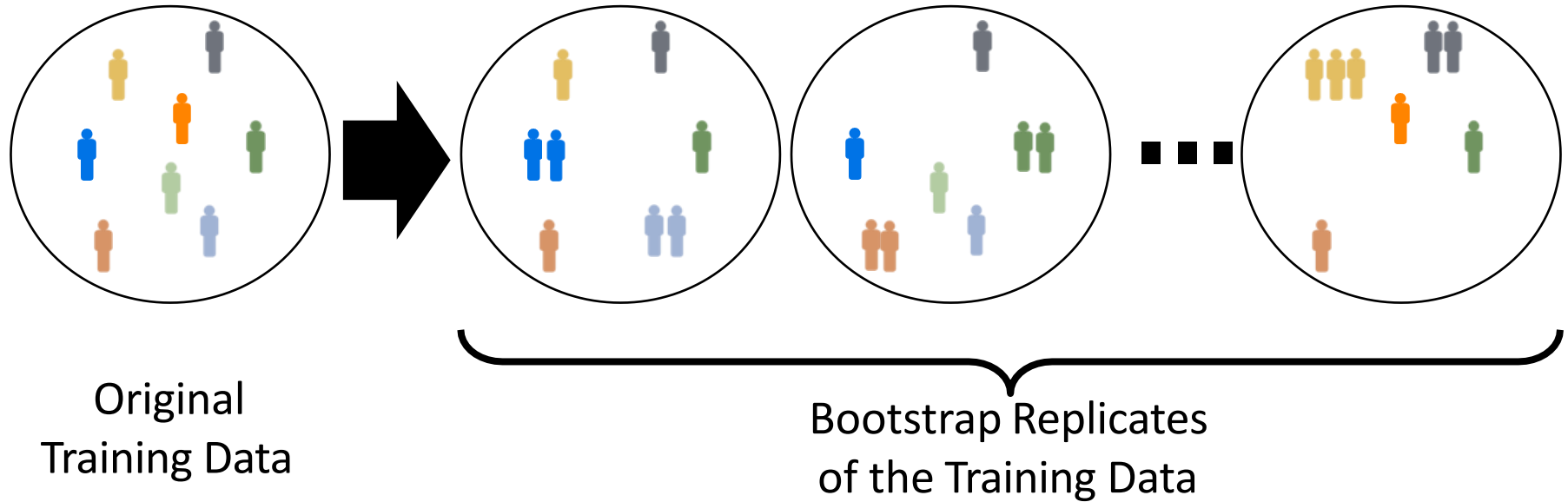
Bagging

- **Bagging:** Randomize training data (“Bootstrap Aggregating”)
 - **Random examples:** Subsample examples $\{(x, y)\}$ (obtain $X \in \mathbb{R}^{n' \times d}$)
- Meta-strategy that can build ensembles from arbitrary base learners

Bootstrap

- Subsample examples $\{(x, y)\}$ **with replacement** (obtain $X \in \mathbb{R}^{n \times d}$)
 - Each bootstrap sample will contain duplicates
- Excludes $\left(1 - \frac{1}{n}\right)^n$ of the training examples
 - Separately in each of the replicates
 - Excludes $\left(1 - \frac{1}{n}\right)^n \rightarrow \frac{1}{e} \approx 36.8\%$ (as $n \rightarrow \infty$) examples on average
- Has good statistical properties

Randomizing Learning Algorithms



Random Forests

- Train many decision trees and average them!
 - Large depth $\rightarrow$ High variance, low bias
 - Averaging many decision trees $\rightarrow$ average away “irrelevant” variance
- Very powerful model family in practice

Randomizing Decision Trees

- What if we want an ensemble of decision trees?
 - **Issue:** Decision tree learning algorithm is deterministic
 - **Solution:** Randomize the learning algorithm (may sacrifice performance)!
- Randomize decisions inside learning algorithm
 - **Example:** Randomize splits weighted (somehow) by information gain
 - **Issue:** Very specific to the algorithm
 - **Solution:** Randomize input to learning algorithm (i.e., training data)!

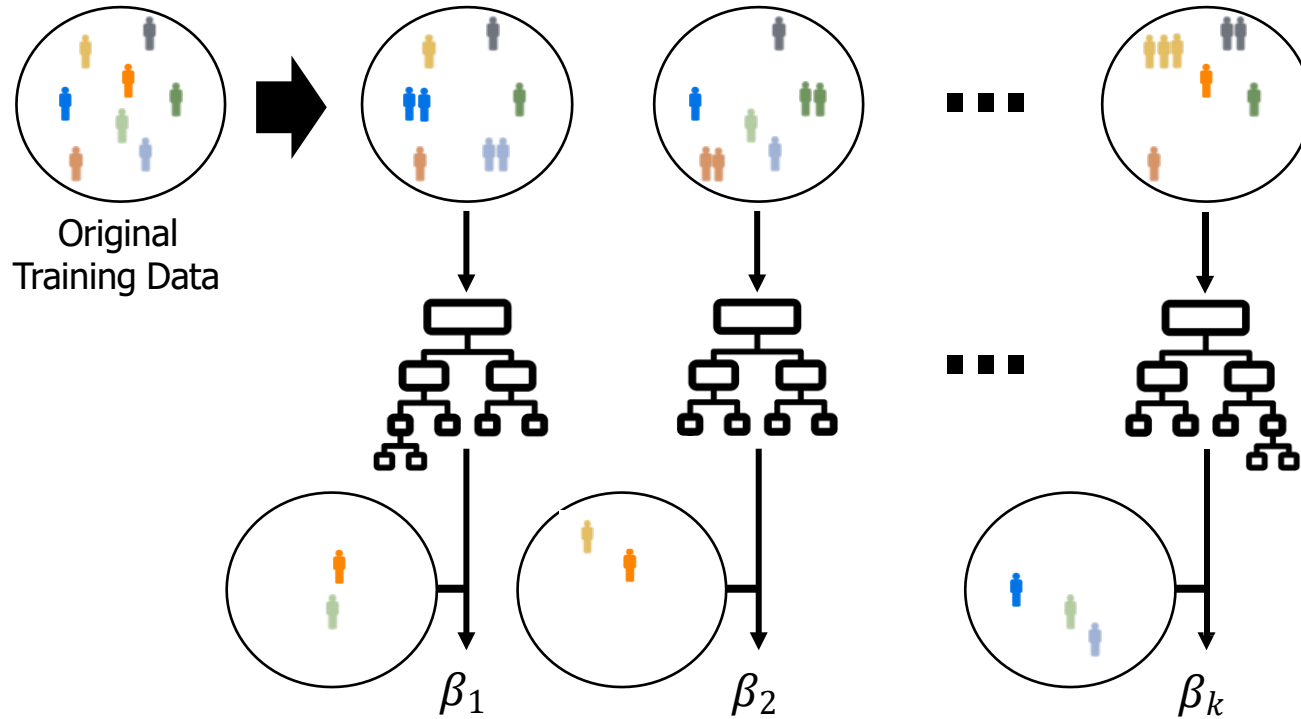
Random Forests

- Ensemble of decision trees using bagging
 - Typically use simple average
- **Intuition:**
 - Large decision trees are good nonlinear models, but high variance
 - Random forests average over many decision trees to reduce variance without increasing bias

Random Forests

- **Tweak 1:** Randomize features in learning algorithm
 - At DT node splitting step, subsample $\approx \sqrt{d}$ features
 - Allows each tree to use all features, but not at every node
 - **Aside:** If a few features are highly predictive, then they will be selected in many trees, causing the base models to be highly correlated
- **Tweak 2:** Train **unpruned** decision trees
 - Ensures base models have higher capacity
 - **Intuition:** Skipping pruning increases variance

Ensemble Learning



Bagging based Ensembles

- **Step 1:** Create bootstrap replicates of the original training dataset
- **Step 2:** Train a classifier for each replicate
- **Step 3 (Optional):** Use held-out validation set to weight models
 - Can just use average predictions

Boosting

- Can we turn weak learning algorithms into strong ones?
- Assume we have a very high bias model, can we make it better?

AdaBoost (Freund & Schapire 1997)

- Like bagging, meta-algorithm that turns base models into ensemble
 - **Provably learns** for base models achieving any error rate > 0.5
- Uses **different training example weights** (instead of different subsamples or different features) to introduce diversity
 - In particular, **upweights** currently incorrectly predicted examples
- Base models should satisfy the following:
 - High-bias/low-capacity (e.g., depth-limited decision trees, linear classifiers)
 - Able to incorporate sample weights during learning
 - **Specific to classification**

AdaBoost

- **Input**

- Training dataset Z
- Learning algorithm $\text{Train}(Z, w)$ that can handle weights w
- Hyperparameter T indicating number of models to train

- **Output**

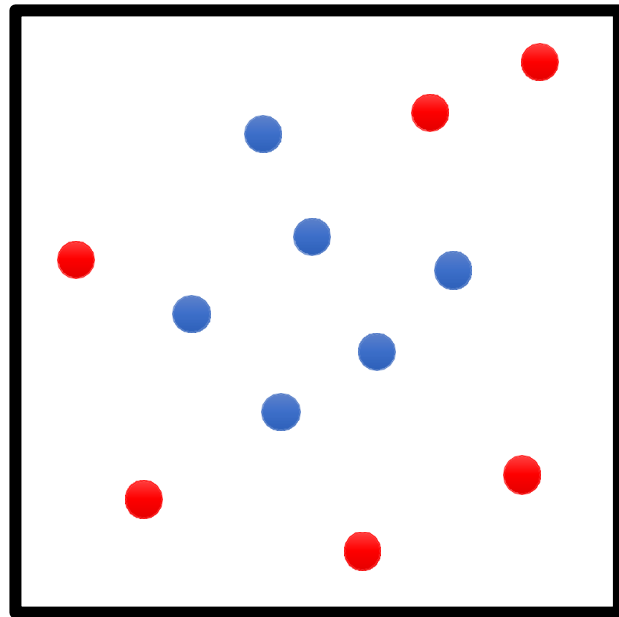
- Ensemble of models $F(x) = \sum_{t=1}^T \beta_t \cdot f_t(x)$

AdaBoost Weighting Strategy

- Iteratively learn the ensemble one by one based on past performance
- On each iteration:
 - Misclassified examples are upweighted
 - Correctly classified are downweighted
- If an example is repeatedly misclassified, it will eventually be upweighted so much that it is correctly classified
- Emphasizes “hardest” parts of the input space
 - Instances with highest weight are often outliers

AdaBoost

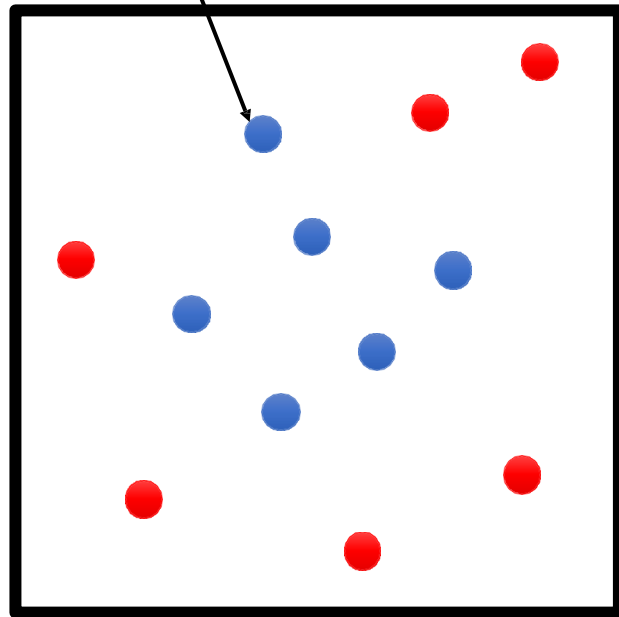
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t y_i f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



AdaBoost

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t y_i f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$

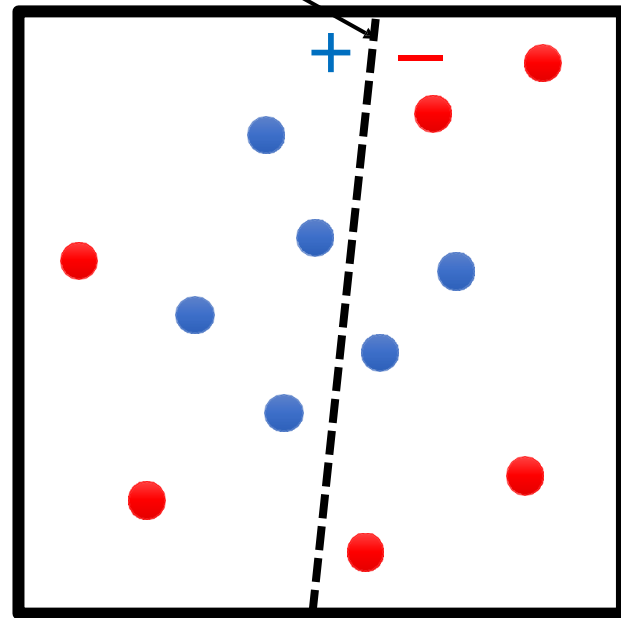
size represents weight w_i



AdaBoost

focus on linear classifiers f_t

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t y_i f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$

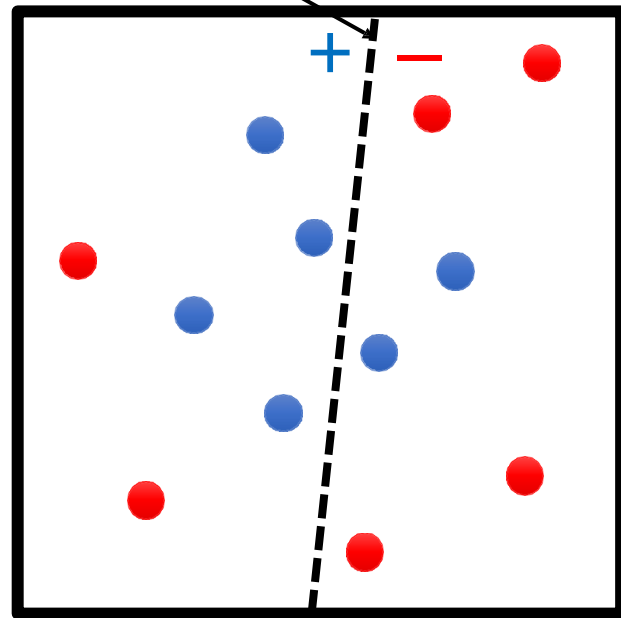


$t = 1$

AdaBoost

focus on linear classifiers f_t

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



$t = 1$

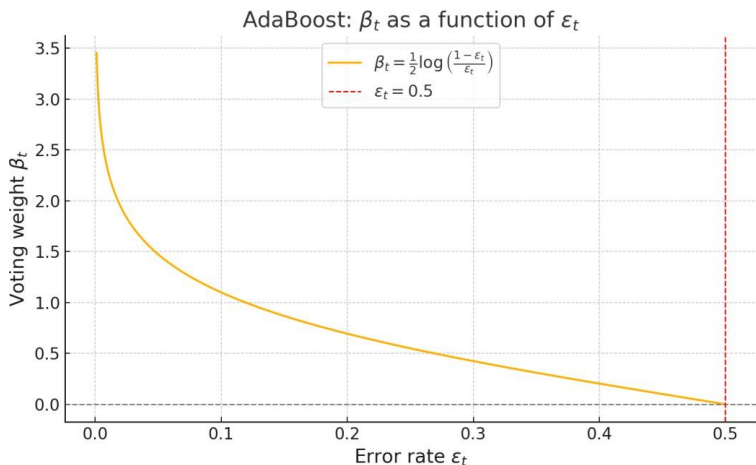
$$\epsilon_t = \text{Error}(f_t, Z, w_t) = \sum_{i=1}^n w_{t,i} \cdot \mathbb{1}[f_t(x_i) \neq y_i]$$

AdaBoost

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$

β_t becomes larger as

ϵ_t becomes smaller

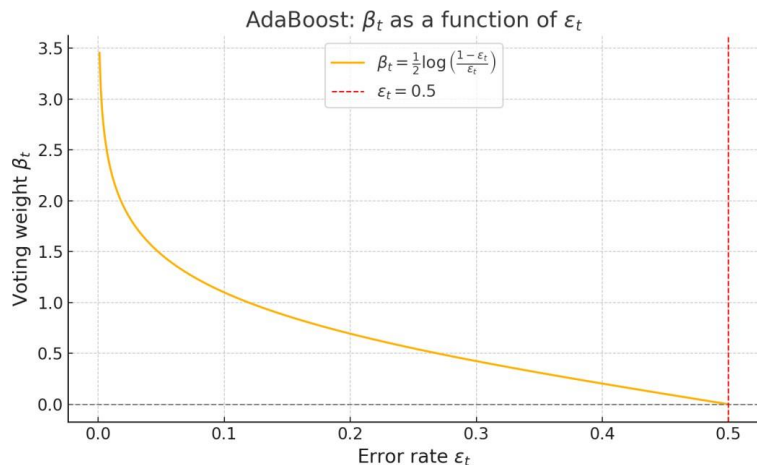


AdaBoost

- β_t measures the importance of $f_t()$
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$

β_t becomes larger as
 ϵ_t becomes smaller



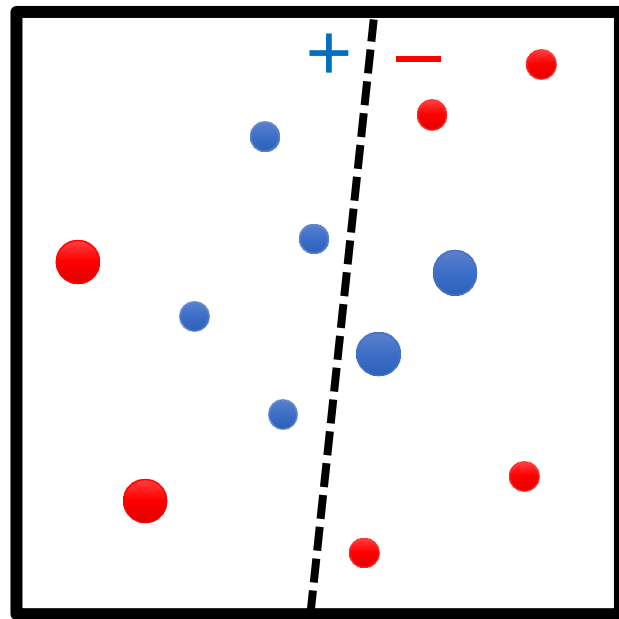
AdaBoost

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t y_i f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}\left(\sum_{t=1}^T \beta_t \cdot f_t(x)\right)$

Use convention $y_i \in \{-1, +1\}$

If incorrect ($y_i \neq f_t(x_i)$) then multiply by e^{β_t}

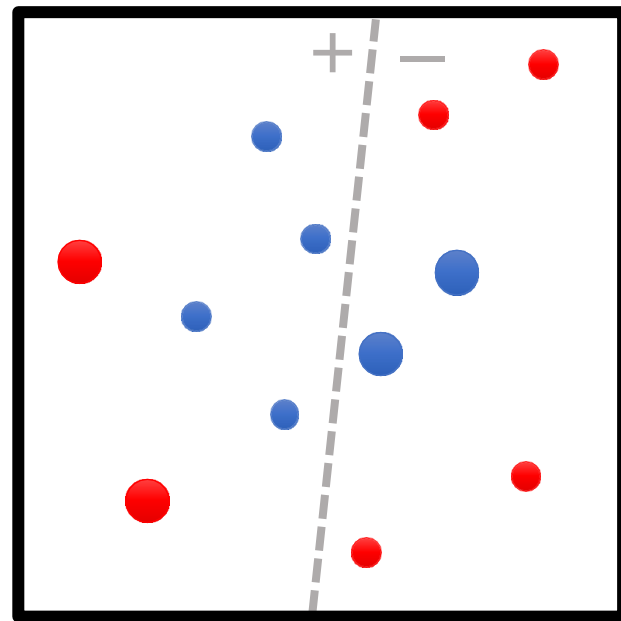
If correct ($y_i = f_t(x_i)$) then multiply by $1/e^{\beta_t}$



$t = 1$

AdaBoost

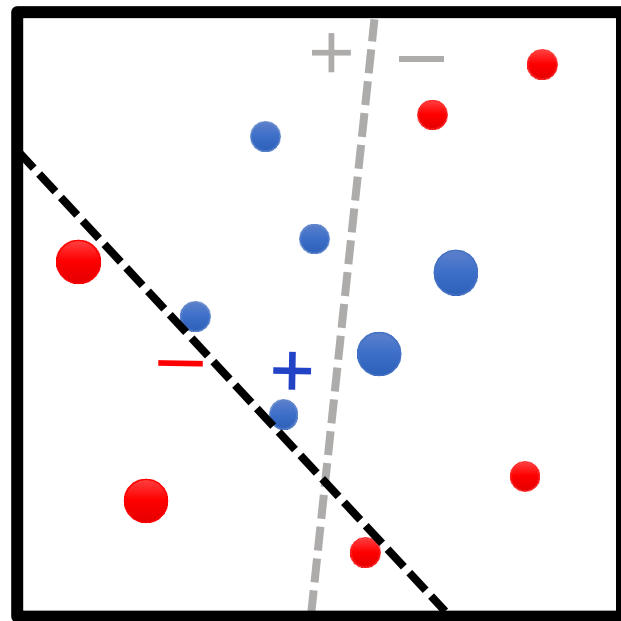
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



$t = 1$

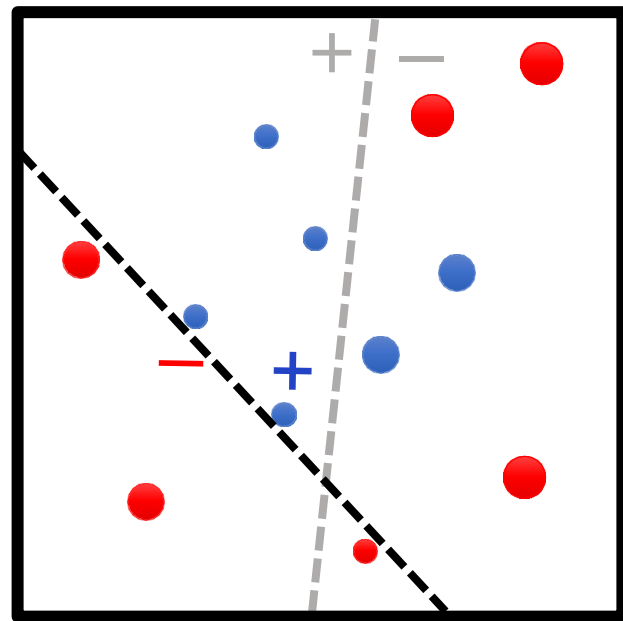
AdaBoost

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



AdaBoost

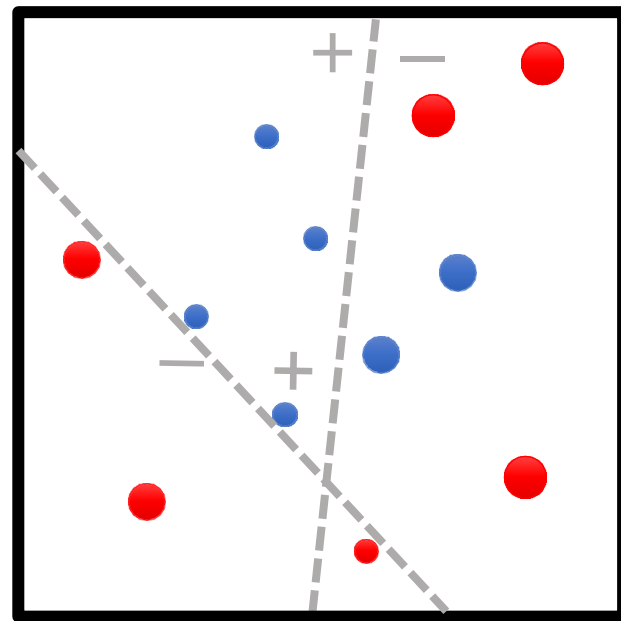
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



$t = 2$

AdaBoost

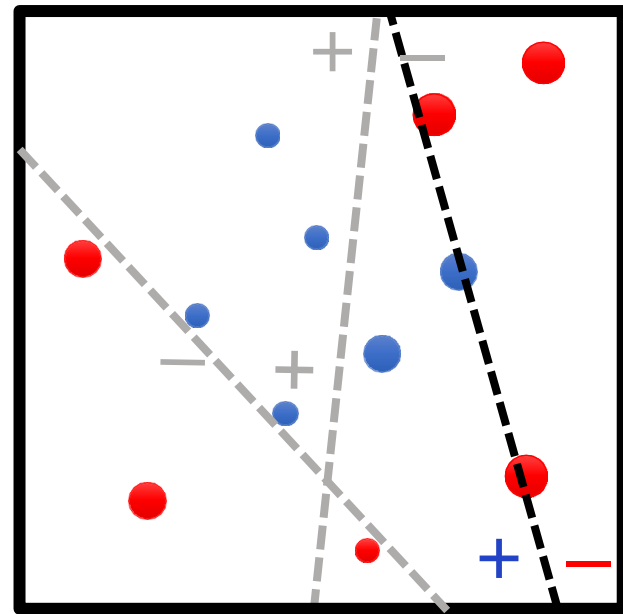
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



$t = 2$

AdaBoost

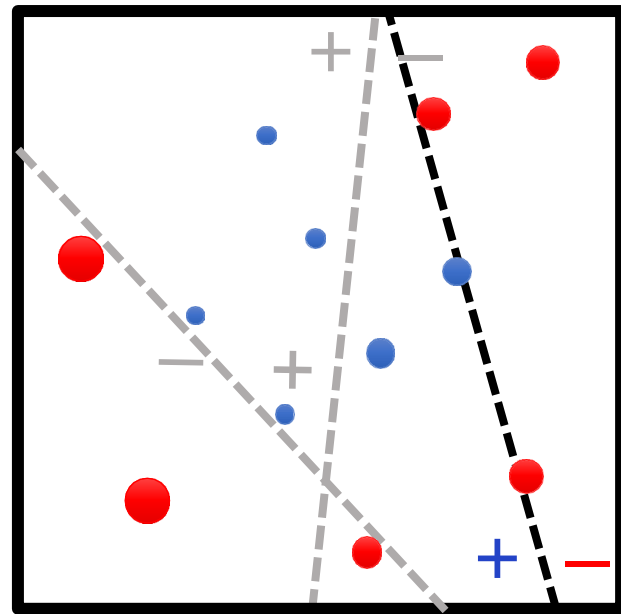
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



$t = 3$

AdaBoost

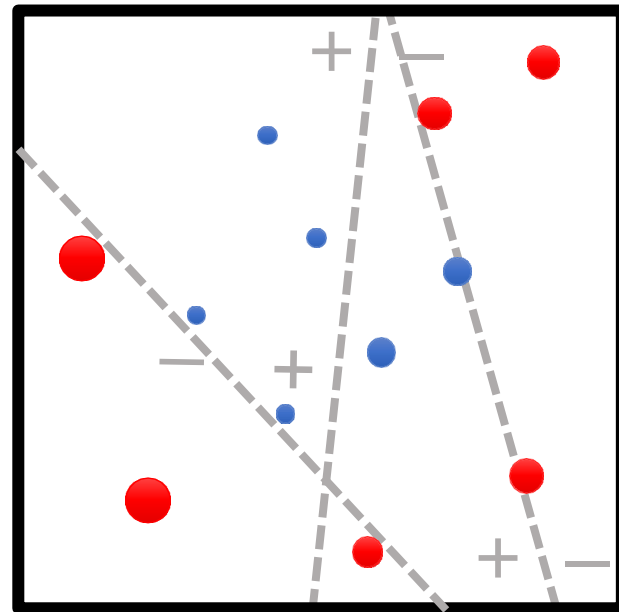
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



$t = 3$

AdaBoost

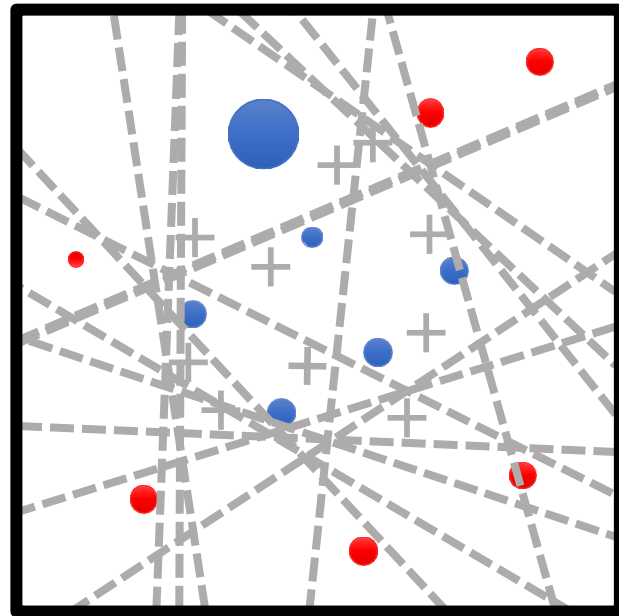
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



$t = 3$

AdaBoost

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t y_i f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$



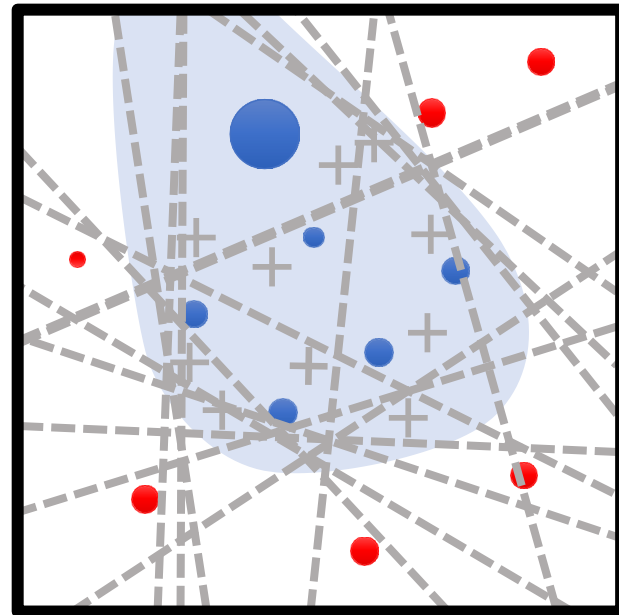
Under certain assumptions, training error
goes to zero in $O(\log n)$ iterations

→ $t = T$

AdaBoost

1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $f_t \leftarrow \text{Train}(Z, w_t)$
4. $\epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t y_i f_t(x_i)}$ (for all i)
7. **return** $F(x) = \text{sign}(\sum_{t=1}^T \beta_t \cdot f_t(x))$

final model is average of base models
weighted by their performance



AdaBoost Weighting Strategy

- On each iteration:
 - Misclassified examples are upweighted
 - Correctly classified are downweighted
- If an example is repeatedly misclassified, it will eventually be upweighted so much that it is correctly classified
- Emphasizes “hardest” parts of the input space
 - Instances with highest weight are often outliers

Aside: Learning with Weighted Examples

- Many algorithms can directly incorporate weights into the loss
- For maximum likelihood estimation:

$$\ell(\mathbf{w}; \mathbf{Z}, \beta) = \sum_{i=1}^n \beta_i \cdot \log p_{\mathbf{w}}(\mathbf{y}_i | \mathbf{x}_i)$$

- Alternatively, can subsample the data proportional to weights β_i

AdaBoost Summary

- **Strengths:**

- Fast and simple to implement
- No hyperparameters (except for T)
- Very few assumptions on base models

- **Weaknesses:**

- Can be susceptible to noise/outliers when there is insufficient data
- No way to parallelize
- Small gains over complex base models
- **Specific to classification!**

Gradient Boosting in Practice

- Gradient boosting with depth-limited decision trees (e.g., depth 3) is one of the most powerful off-the-shelf classifiers available
 - **Caveat:** Inherits decision tree hyperparameters
- XGBoost is a very efficient implementation suitable for production use
 - A popular library for gradient boosted decision trees
 - Optimized for computational efficiency of training and testing
 - Used in many competition winning entries, across many domains
 - <https://xgboost.readthedocs.io>

Recap: Bagging and Boosting

- Bagging (Bootstrap Aggregating)
 - Main goal is to reduce variance
 - Models are trained **independently** with randomized dataset
- Boosting
 - Main goal is to reduce bias
 - Models are trained **sequentially** with weighted dataset

Acknowledgement

Most of the slides are based on the ML course of:

- Mingmin Zhao and Jiatao Gu, University of Pennsylvania

THANK YOU