Lecture 19: Ensembles (Part 1)

CIS 4190/5190

Fall 2024

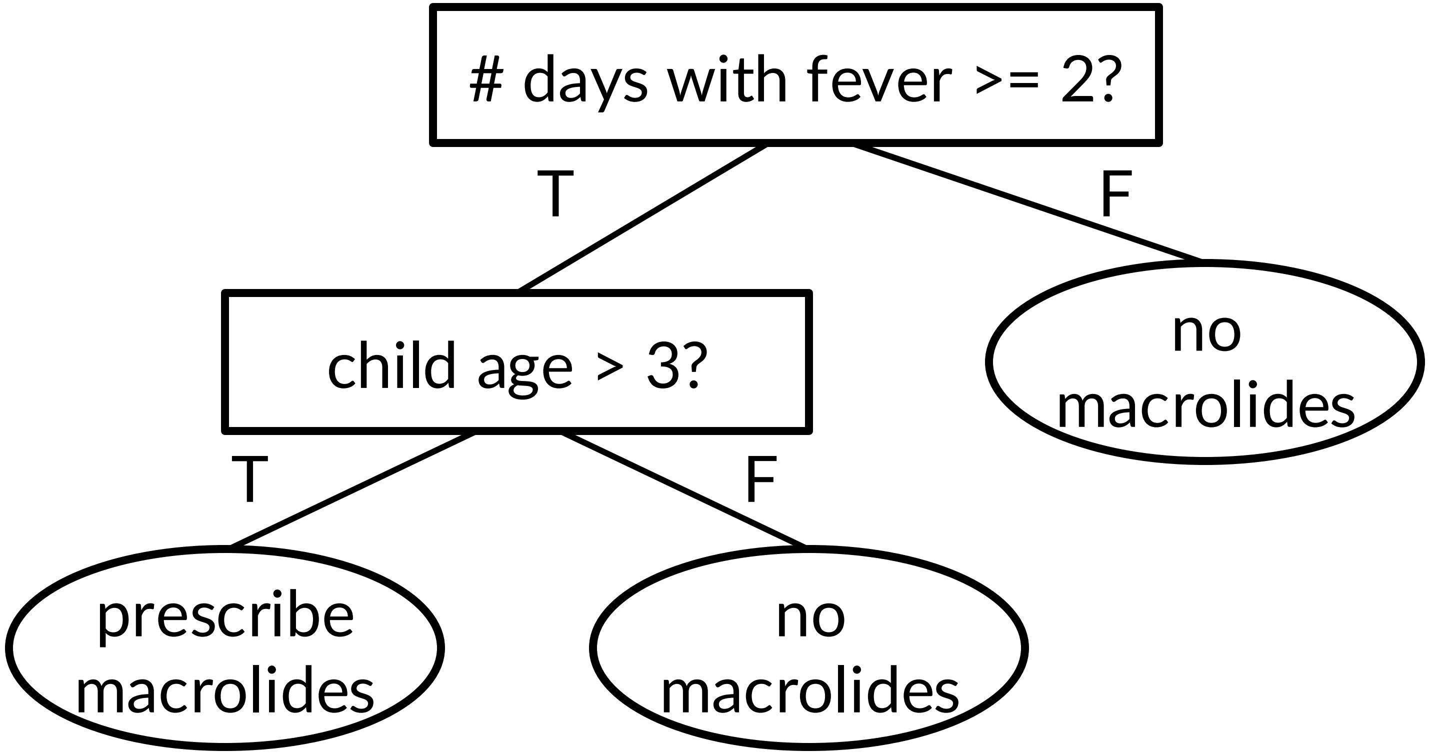
Announcements

- Projects teams
  - Img2GPS: 22 teams
  - News source classification: 38 teams
  - Audio classification: 8 teams

- HW 2 grades released
- HW 4 due on Nov 20

- Midterm 2 scheduled for 12/9
  - Cumulative
  - Similar to midterm 1 in terms of exam time, location, #questions, cheat sheet.

Decision Tree Shortcomings



Decision tree example from: Martignon and Monti. (2010). Conditions for risk assessment as a topic for probabilistic education. *Proceedings of the Eighth International Conference on Teaching Statistics* (ICOTS8).

Decision Tree Shortcomings

- Hard to manage bias-variance tradeoff
  - Small depth → High bias, low variance
  - Large depth → Small bias, high variance

Post Pruning

def PostPruneTree(T, Z_{train}, Z_{val}):

  for each internal node N of T:

    $T_N \leftarrow \text{Replace}\big(T, N, \text{LeafNode}\big(\text{Mode}(Z_{\text{train}}[N])\big)\big)$

    $g_N \leftarrow \text{Loss}(T, Z_{\text{val}}) - \text{Loss}(T_N, Z_{\text{val}})$

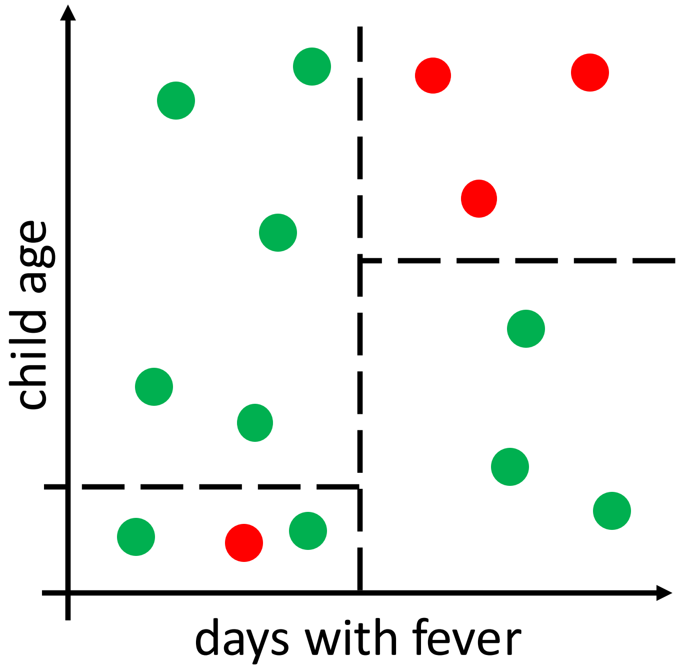
  $N_0 \leftarrow \arg\max_N g_N$

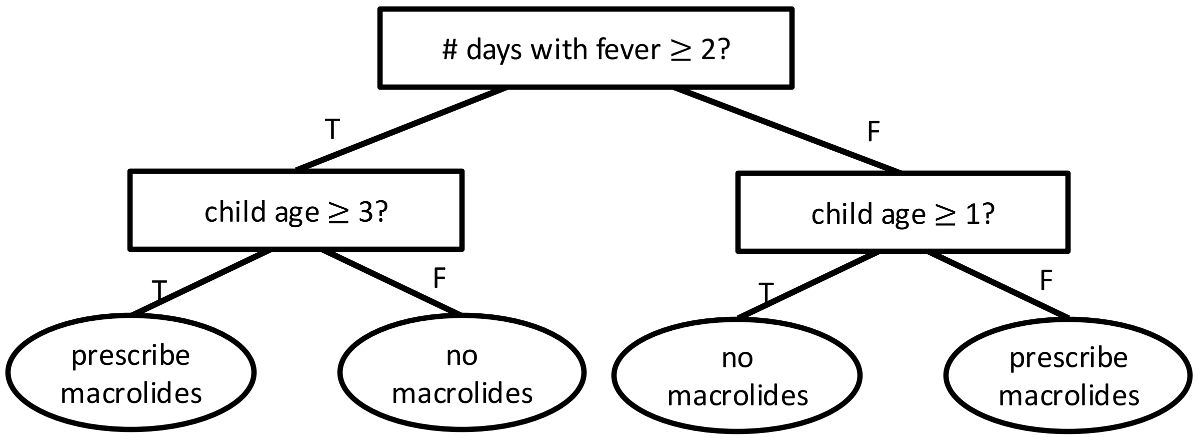
  if $g_{N_0} > 0$:

    return PostPruneTree(T_N, Z_{train}, Z_{val})

  else:

    return T





Post Pruning

def PostPruneTree($T, Z_{\text{train}}, Z_{\text{val}}$):

    for each internal node N of T:

        $T_N \leftarrow \text{Replace}\left(T, N, \text{LeafNode}\left(\text{Mode}(Z_{\text{train}}[N])\right)\right)$

        $g_N \leftarrow \text{Loss}(T, Z_{\text{val}}) - \text{Loss}(T_N, Z_{\text{val}})$

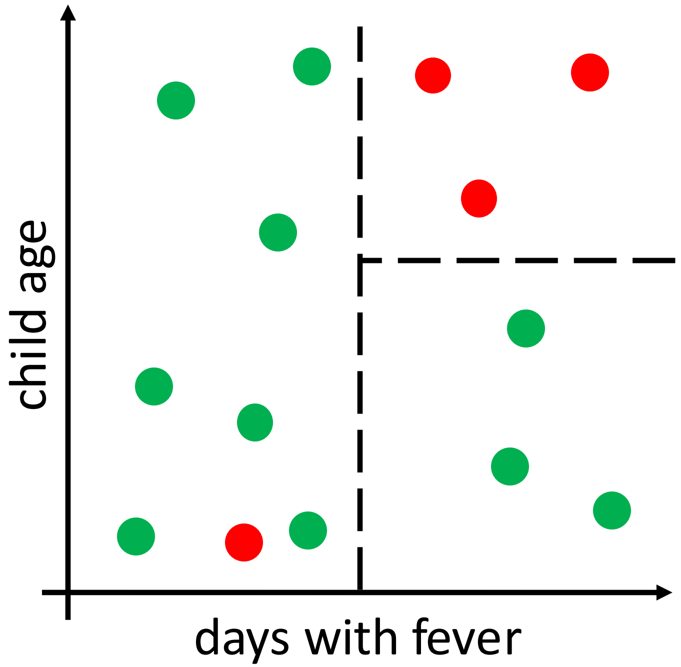
    $N_0 \leftarrow \arg\max_N g_N$

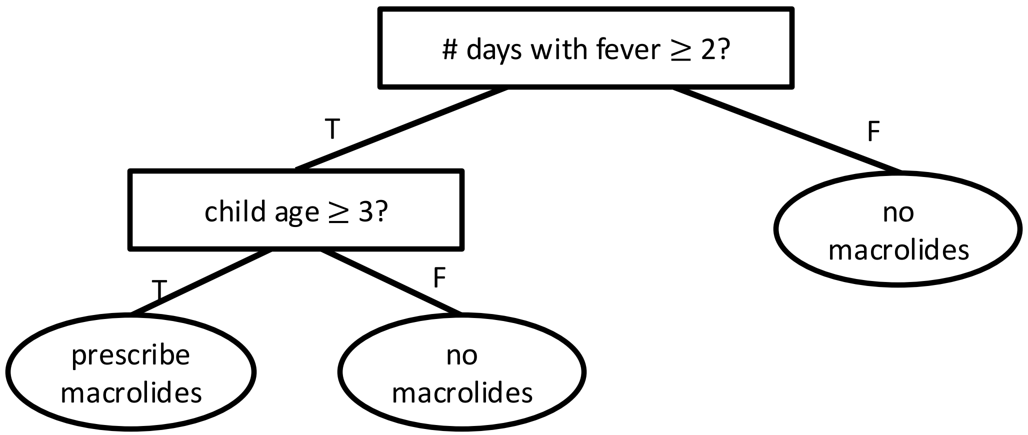
    if $g_{N_0} > 0$:

        return PostPruneTree($T_N, Z_{\text{train}}, Z_{\text{val}}$)

    else:

        return T





Decision Tree Shortcomings

- Hard to manage bias-variance tradeoff
  - Small depth → High bias, low variance
  - Large depth → 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 <u>reducing variance</u> 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 $f_1, \dots, f_k$

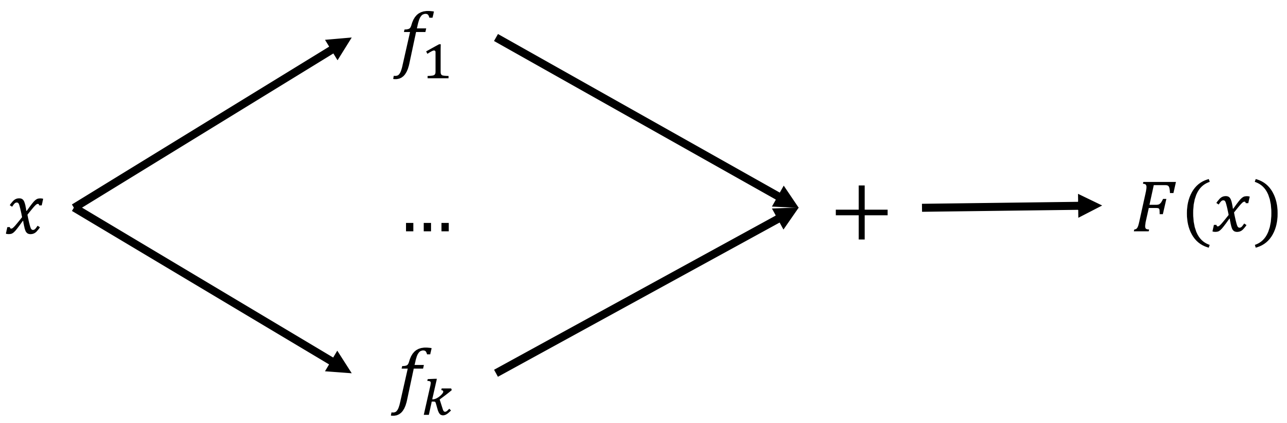
Ensemble Design Decisions

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

- <u>How to combine the learned base models?</u>

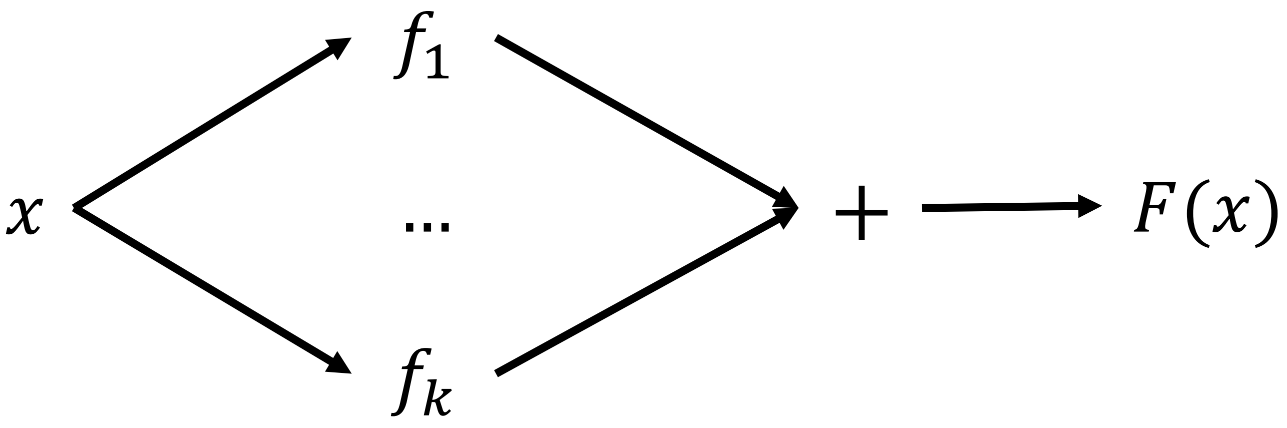
Combining Learned Base Models

- **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 $F(x) = 1\left(\sum_{i=1}^{k} f_i(x) \geq \frac{k}{2}\right)$ (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

$$F(x) = g_\beta\big(f_1(x), \ldots, f_k(x)\big)$$

Combining Learned Base Models

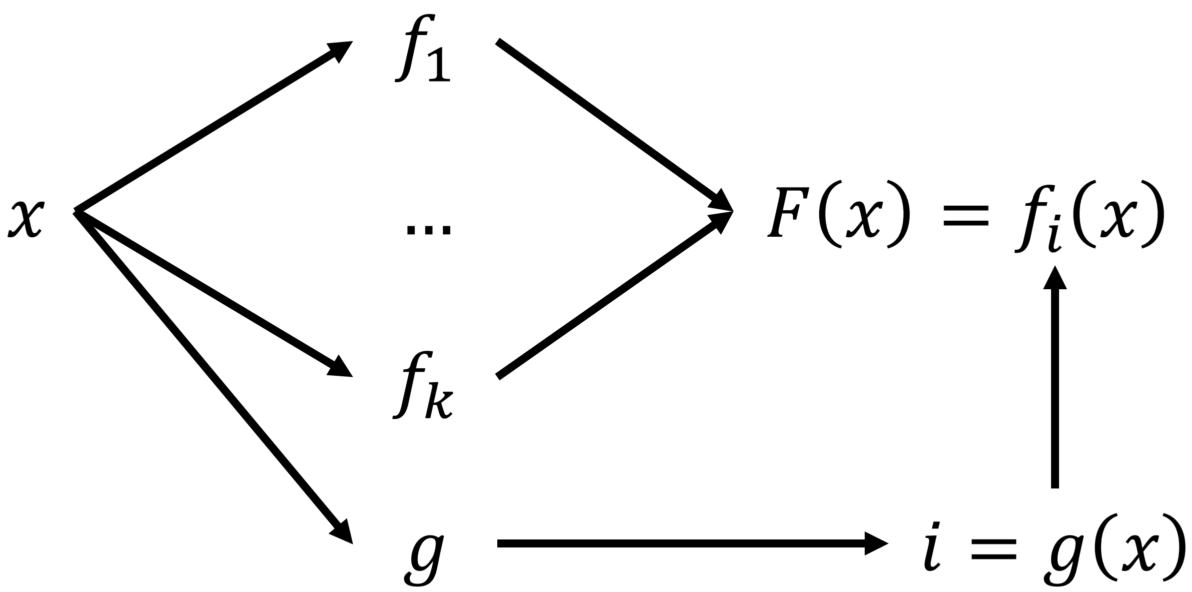
- Second model as “mixture of experts”:

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

- 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

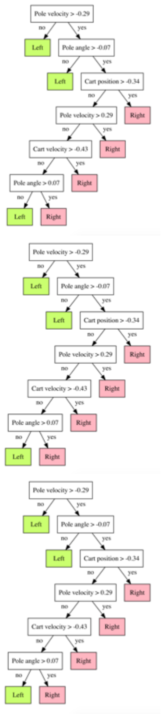
- Successful ensembles require **diversity**
  - Different model families
  - Different training data
  - Different features
  - Different hyperparameters

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

Learning Base Models

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

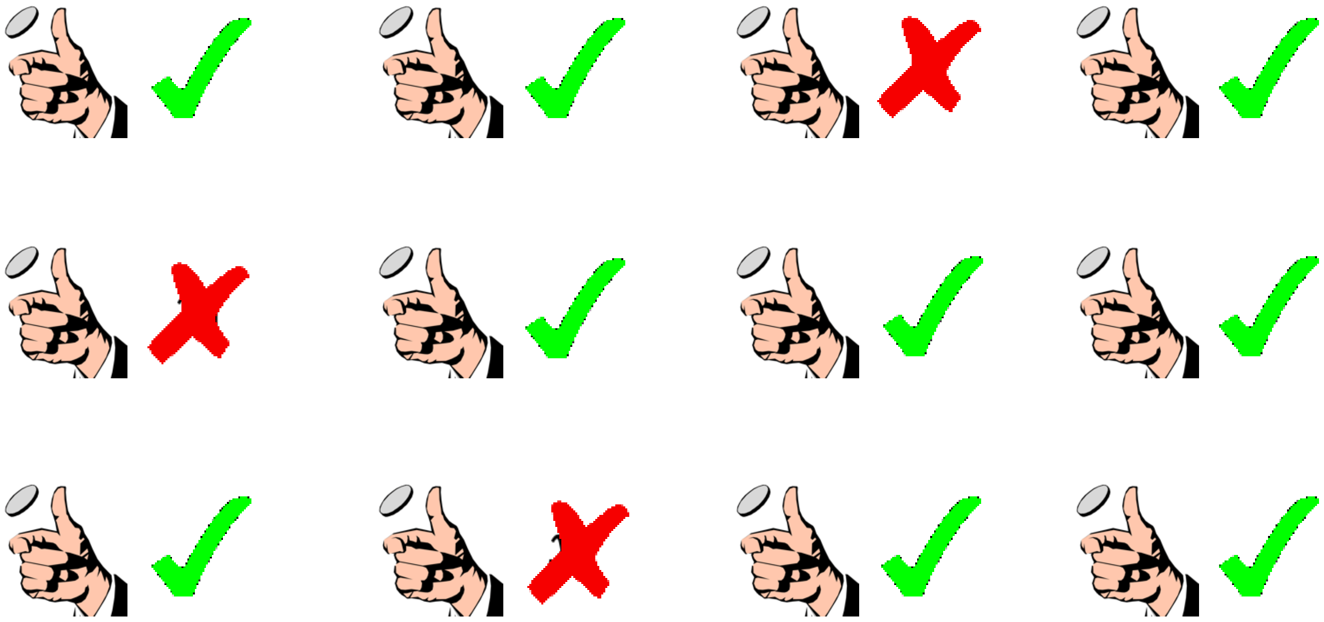
x_1 x_2 x_3 x_4



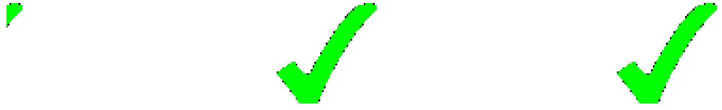
$\text{acc} = \frac{3}{4}$

$\text{acc} = \frac{3}{4}$

$\text{acc} = \frac{3}{4}$



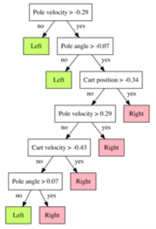
F $\quad \text{acc} = 1 - \left(1 - \frac{3}{4}\right)^3 - 3 \cdot \frac{3}{4} \cdot \left(1 - \frac{3}{4}\right)^2 \approx 0.84$



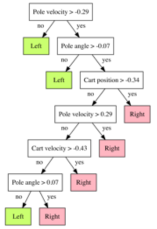
Learning Base Models

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

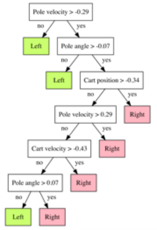
x_1 x_2 x_3 x_4



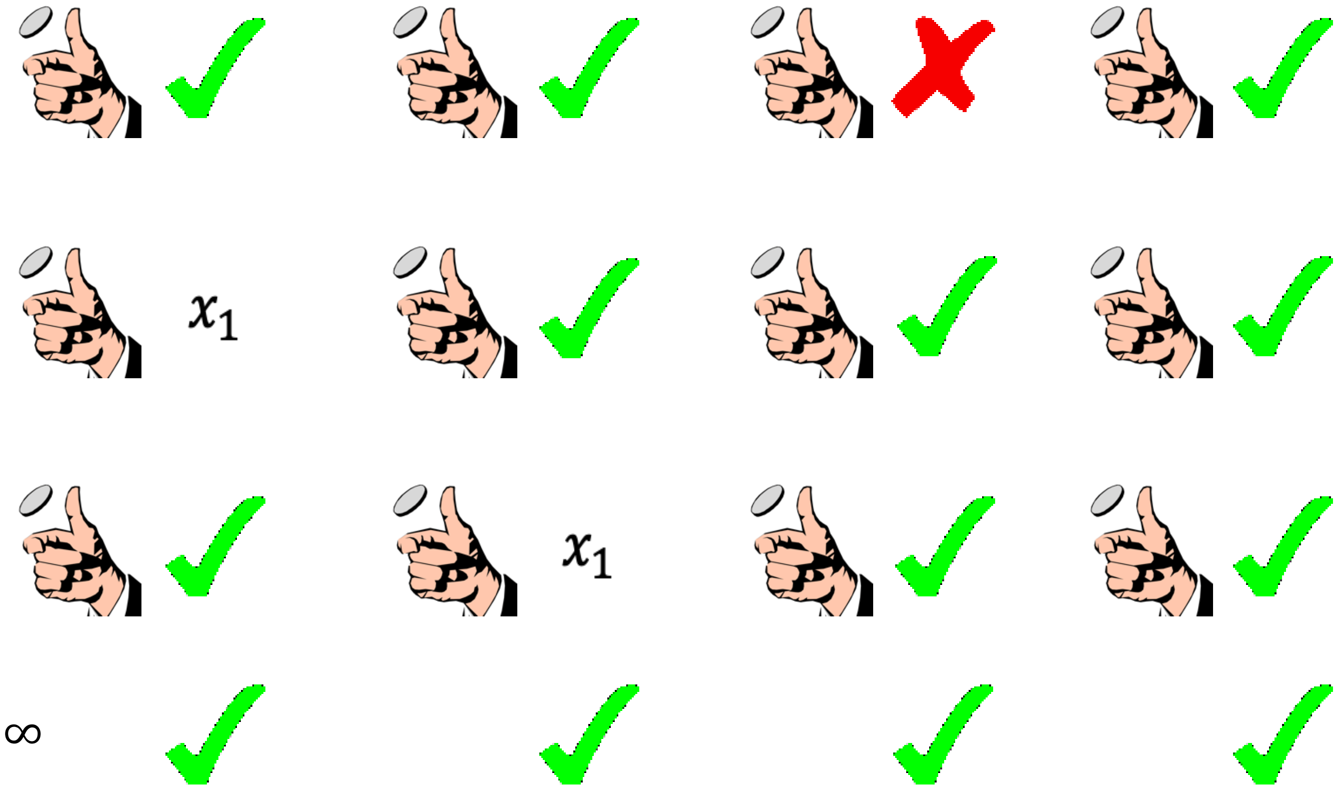
$\text{acc} = \frac{3}{4}$



$\text{acc} = \frac{3}{4}$



$\text{acc} = \frac{3}{4}$



F $\text{acc} \to 1$ as $k \to \infty$

Bagging

- **Bagging:** Randomize training data ("Boostrap 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

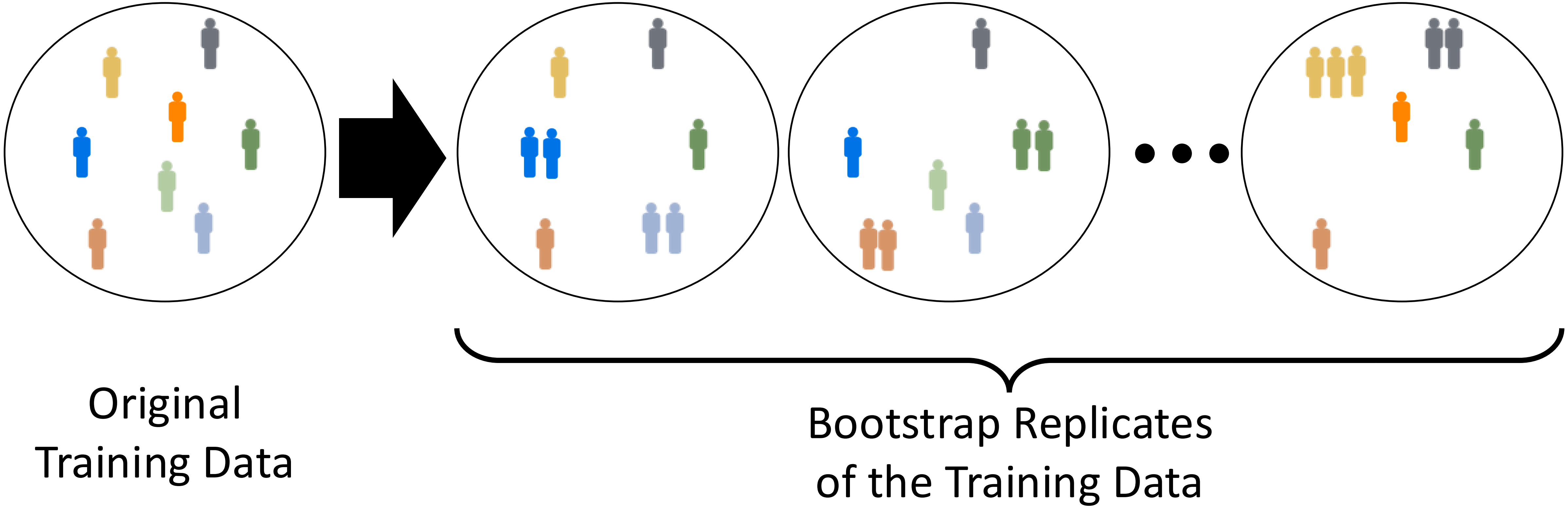
Bootstrap

- Subsample examples $\{(x, y)\}$ **with replacement** (obtain $X \in \mathbb{R}^{n\times d}$)

- Excludes $\left(1 - \frac{1}{n}\right)^n$ of the training examples
  - Separately in each of the replicates
  - As $n \to \infty$, excludes $\to \frac{1}{e} \approx 36.8\%$ examples

- Has good statistical properties

Randomizing Learning Algorithms



Random Forests

- Train many decision trees and average them!
  - Large depth → High variance, low bias
  - Averaging many decision trees → average away “irrelevant” variance

- Very powerful model family in practice

Random Forests

- Ensemble of decision trees using bagging
    - Typically use simple average

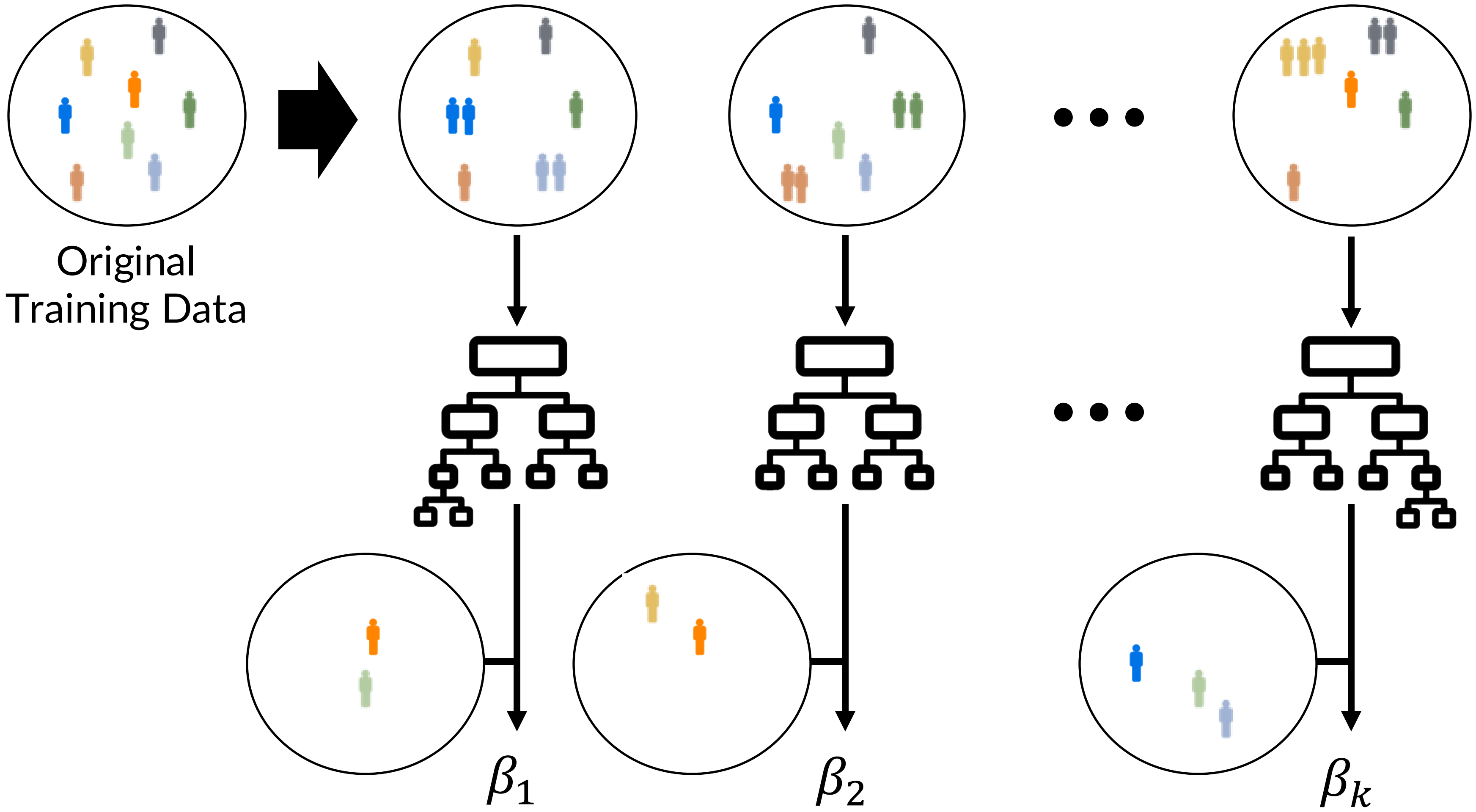
- **Intuition:**
    - Large decision trees are good nonlinear models, but high variance
    - Random forests average over many decision trees to reduce variance without increasing bias

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?

- **Provably learns** for base models achieving any error rate > 0.5

- In the context of tree, assume very short trees (depth 3-6).

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 (discuss general losses later)**

AdaBoost (Freund & Schapire 1997)

- **Input**
  - Training dataset Z
  - Learning algorithm $\mathrm{Train}(Z, w)$ 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

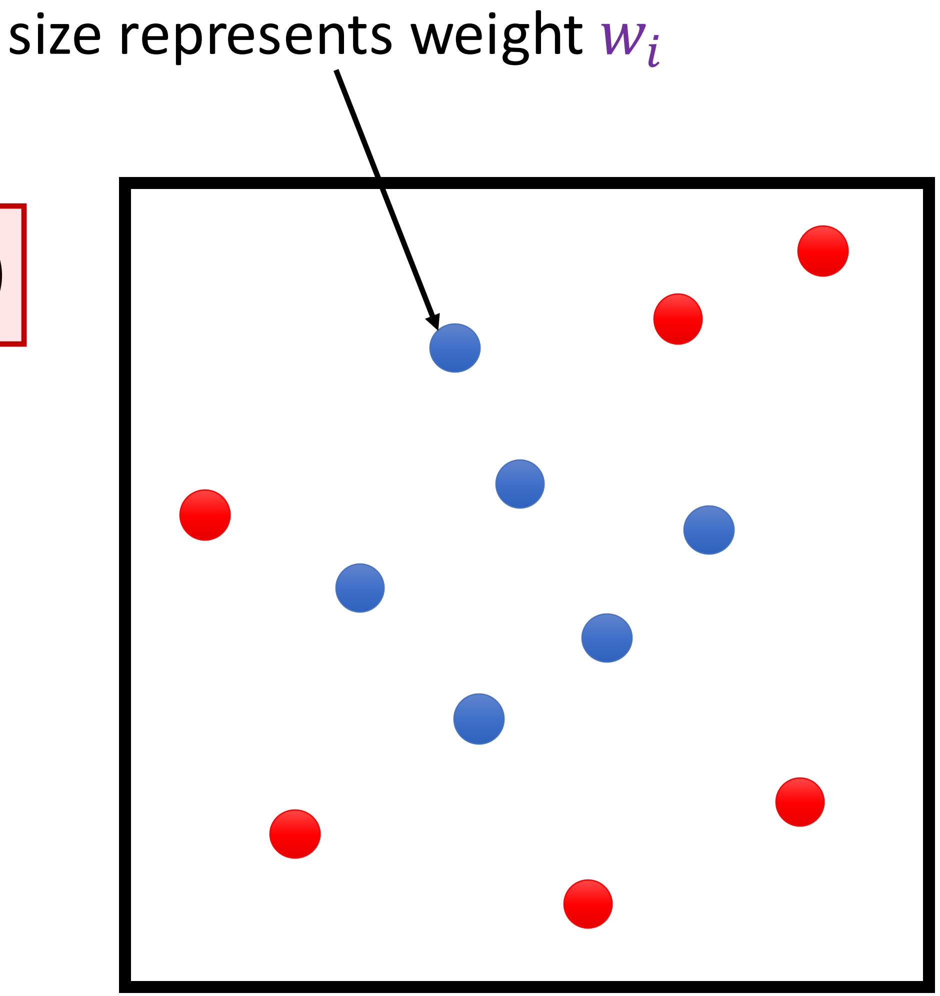
- On each iteration:
  - Misclassified examples are upweighted
  - Correctly classified are downweighted

- 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

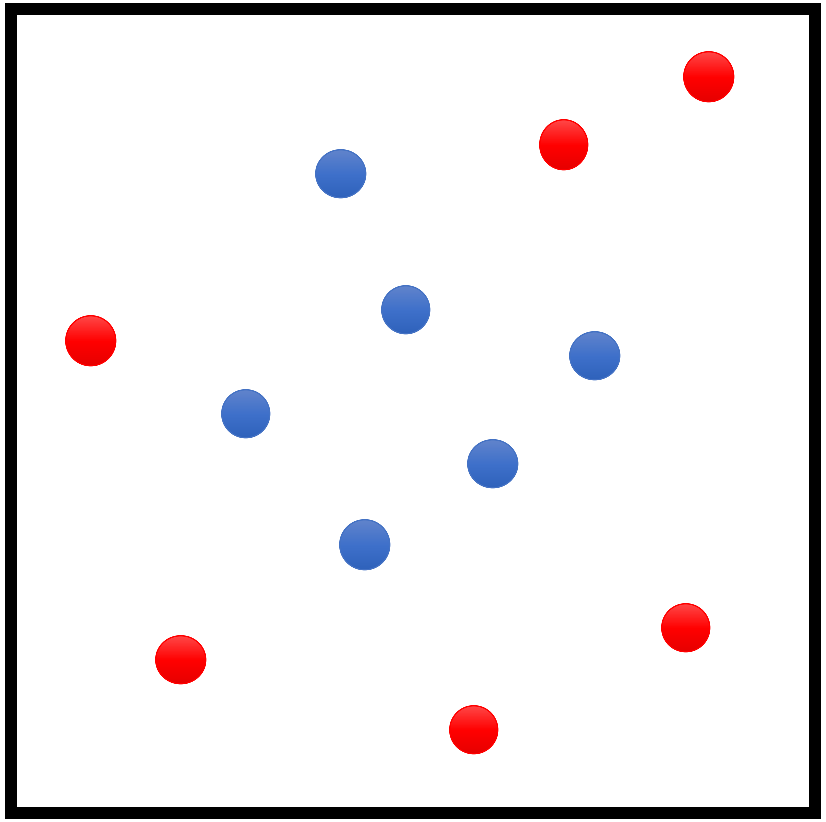
AdaBoost

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



AdaBoost

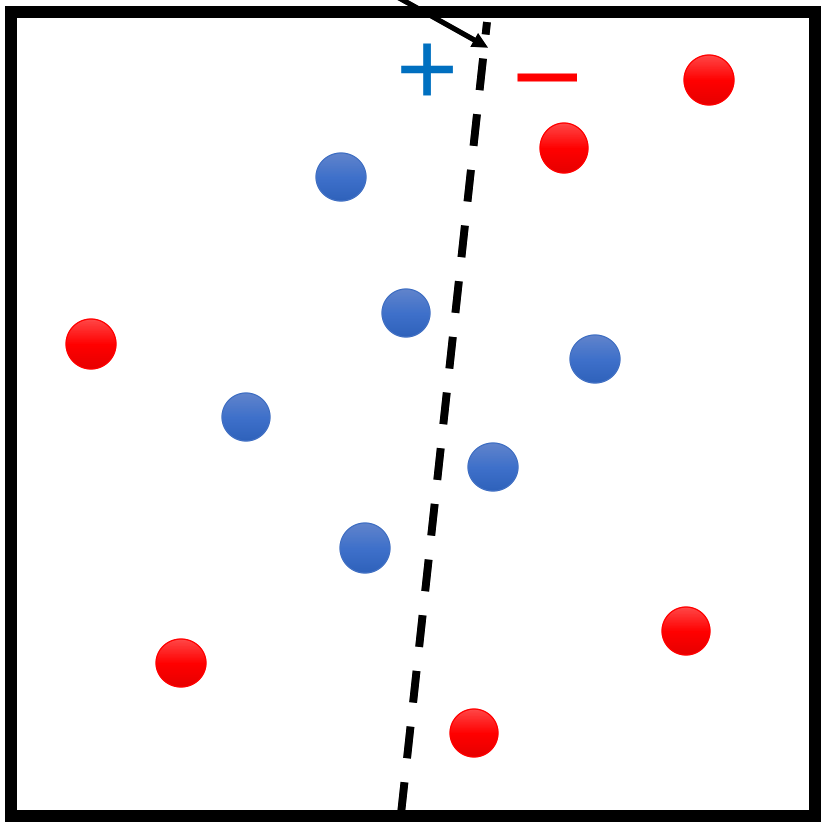
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



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

focus on linear classifiers f_t

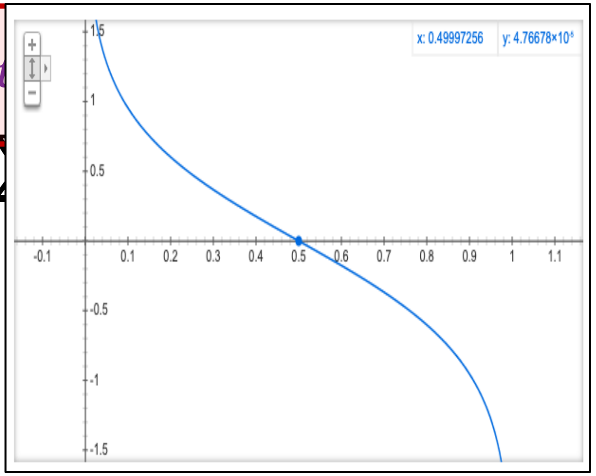


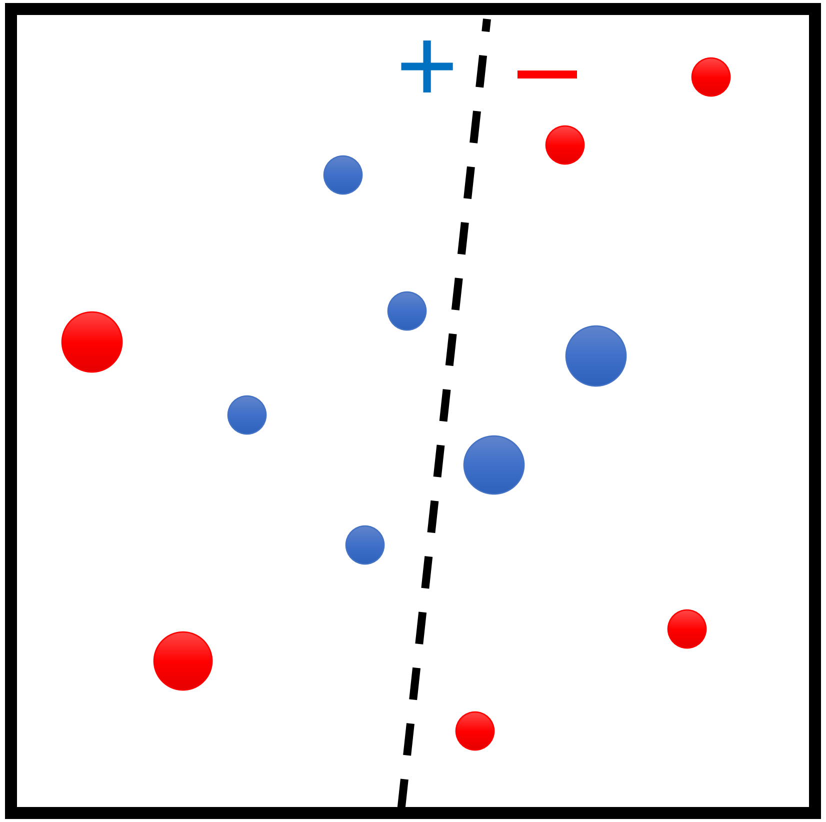
$t = 1$

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. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
6. $\quad w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t}$
7. **return** $F(x) = \mathrm{sign}(\Sigma$

β_t becomes larger as ϵ_t becomes smaller





$t = 1$

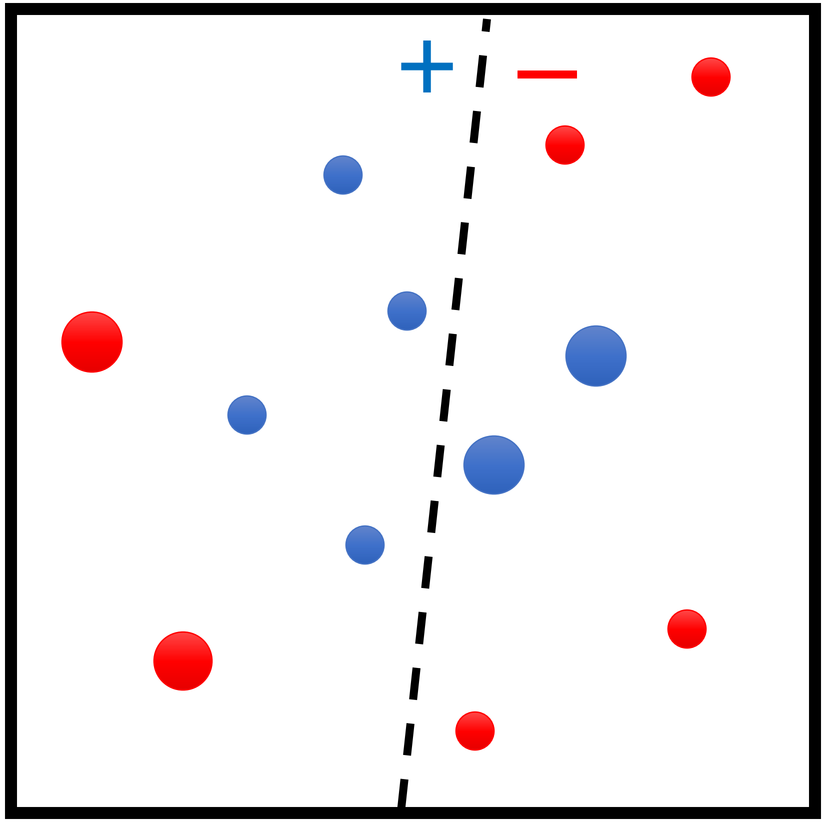
AdaBoost

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

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

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

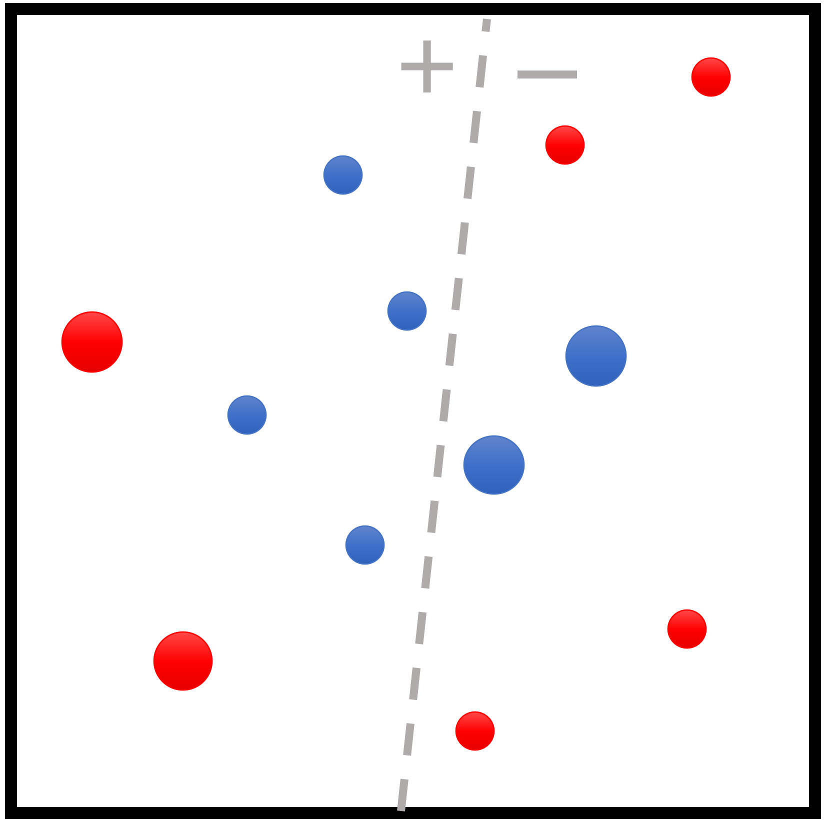
If incorrect ($y_i \neq f_t(x_i)$) then multiply by e^{β_t}



$t = 1$

AdaBoost

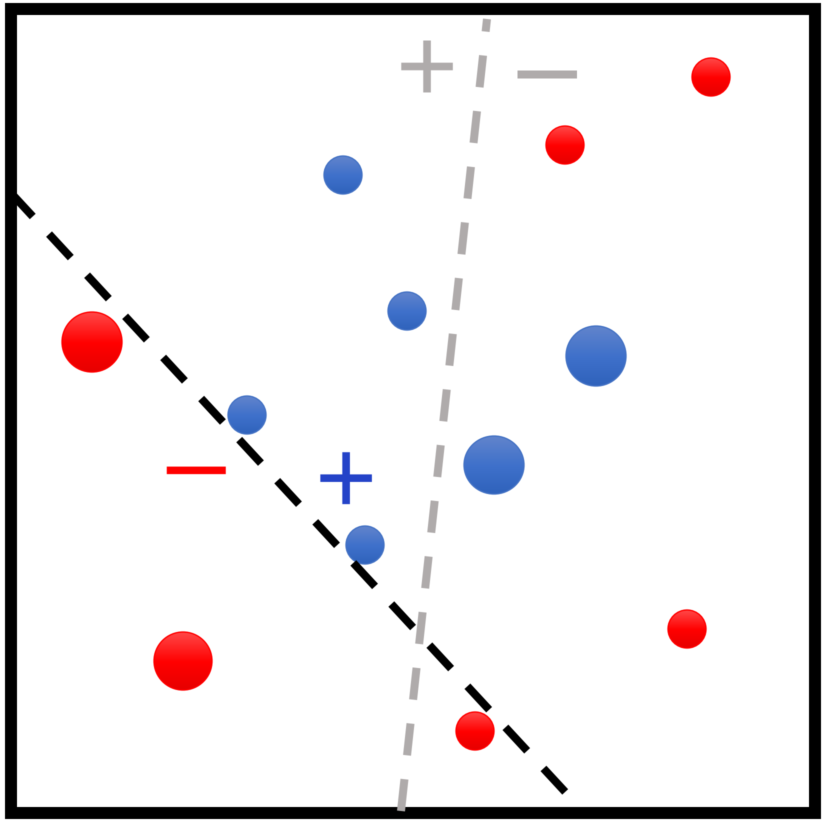
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 1$

AdaBoost

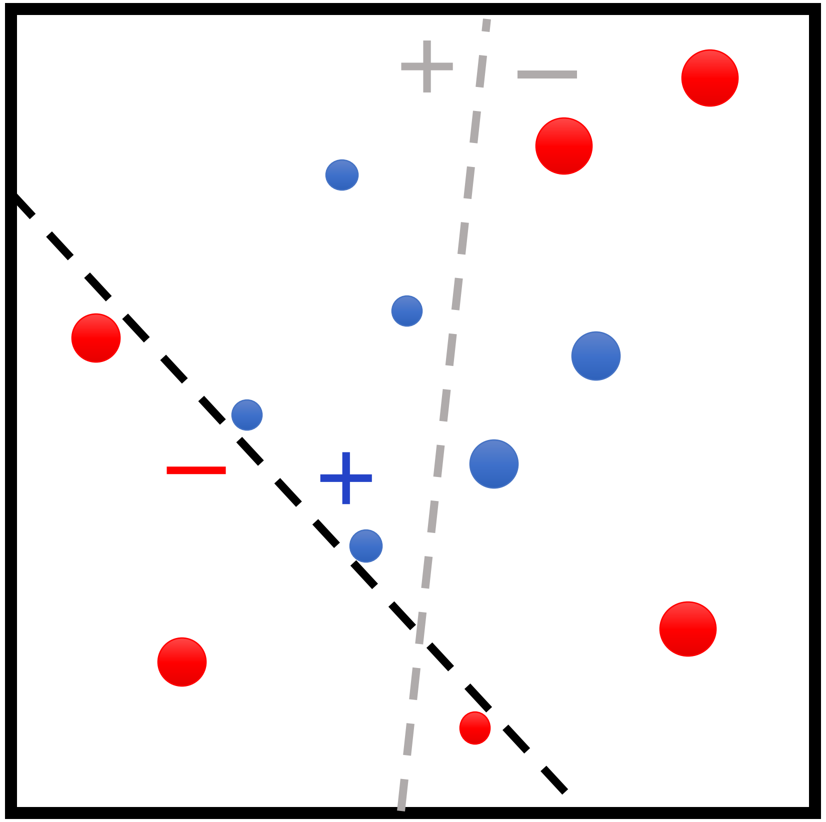
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 2$

AdaBoost

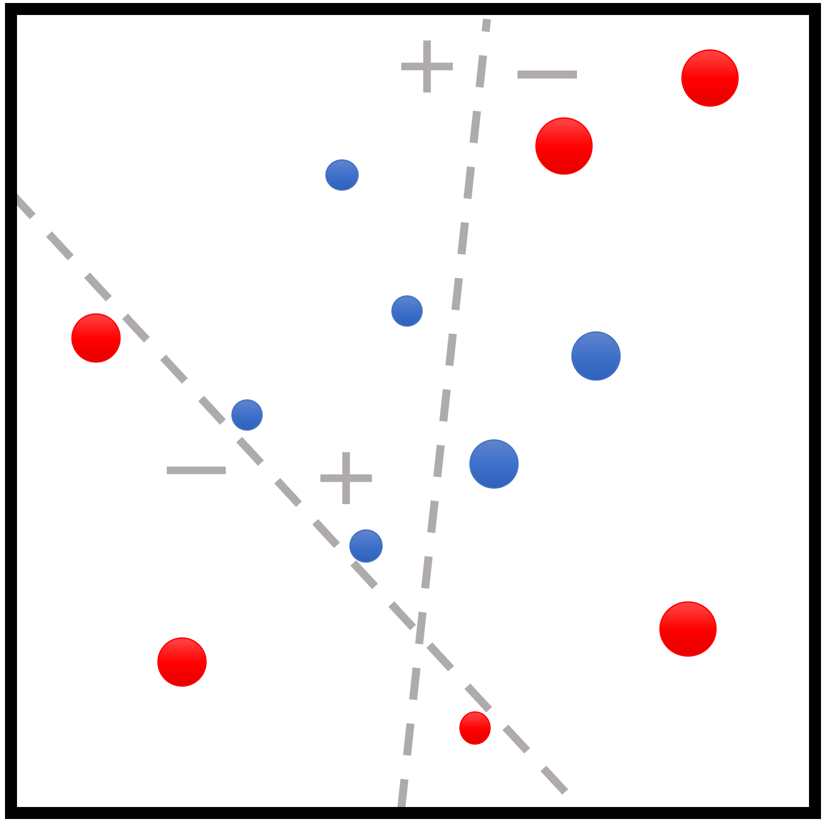
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 2$

AdaBoost

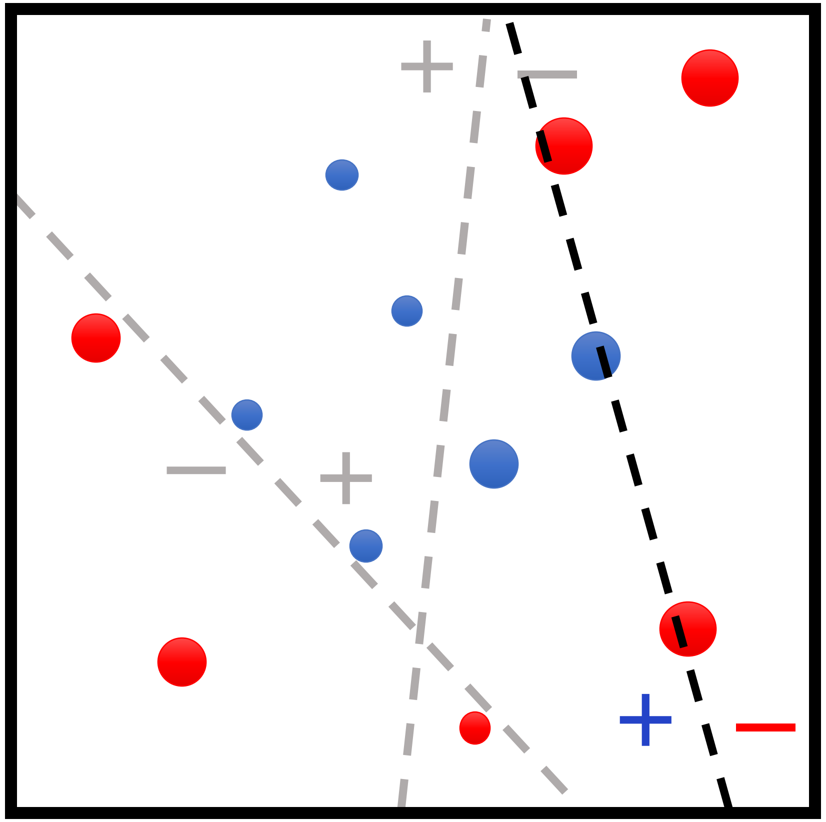
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \text{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 2$

AdaBoost

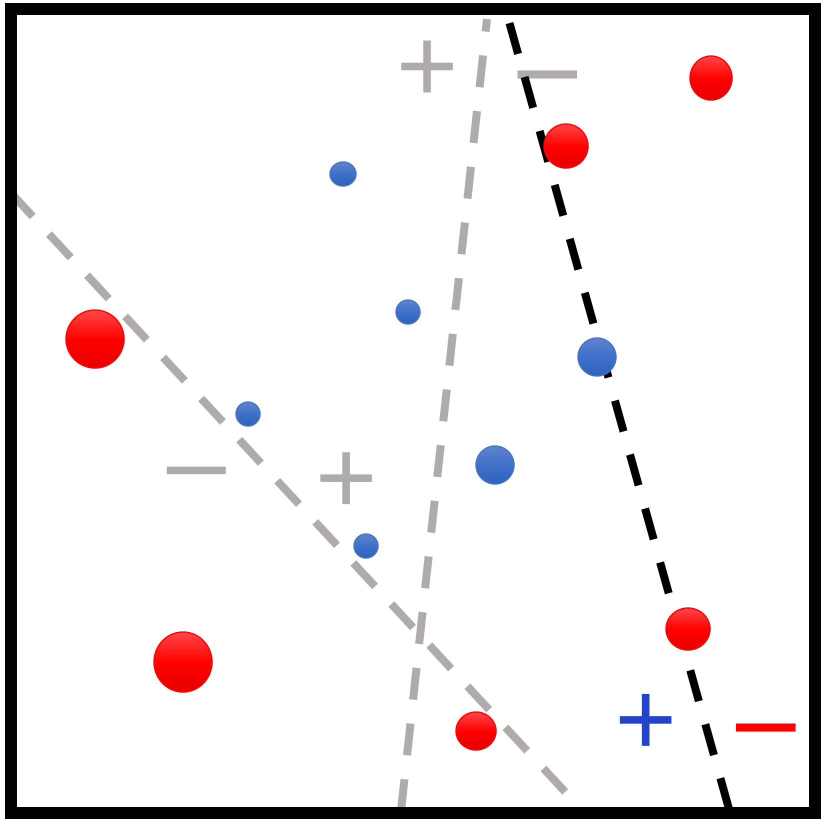
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 3$

AdaBoost

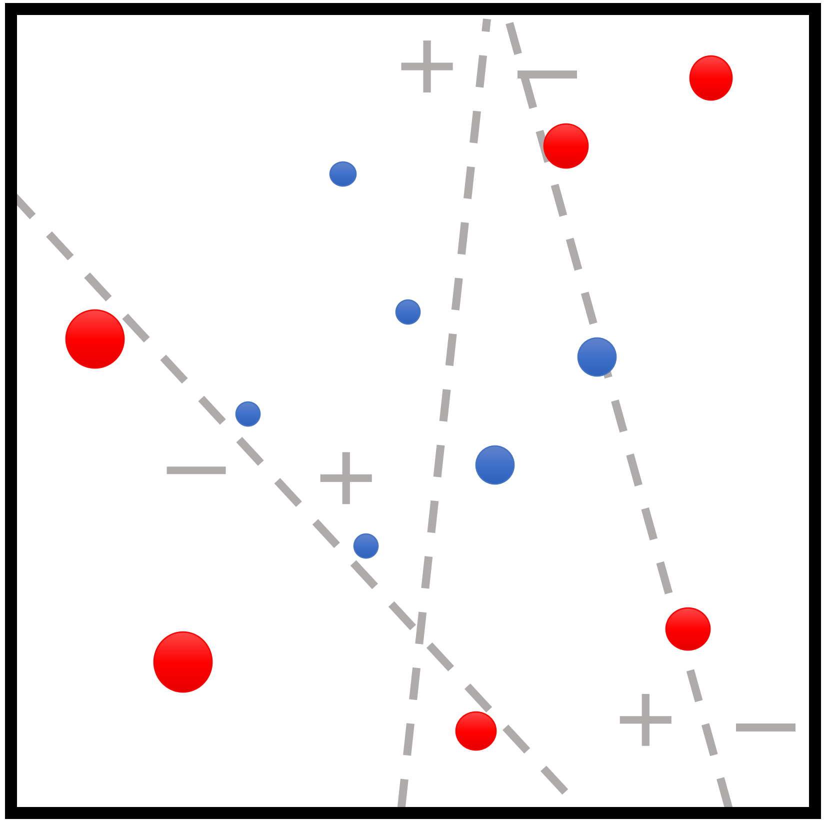
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 3$

AdaBoost

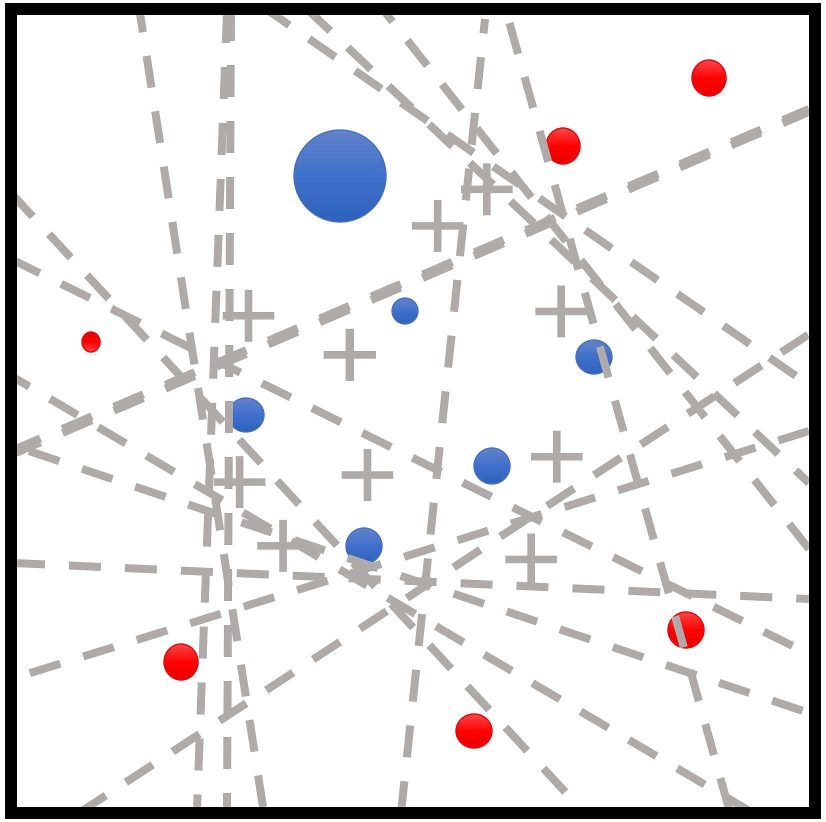
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 3$

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

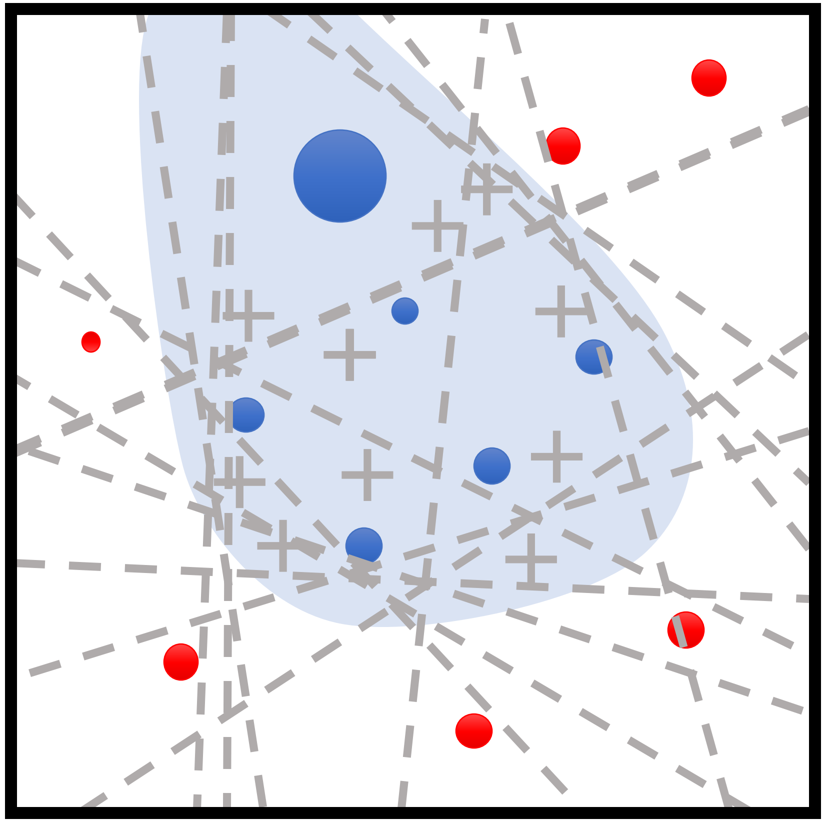


Under certain assumptions, training error goes to zero in $O(\log n)$ iterations

AdaBoost

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

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:

$$\ell(\beta; Z, w) = \sum_{i=1}^{n} w_i \cdot \log p_\beta(y_i \mid x_i)$$

- Alternatively, can subsample the data proportional to weights w_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!**

AdaBoost and Overfitting

- Basic ML theory predicts AdaBoost always overfits as $T \rightarrow \infty$
  - Hypothesis keeps growing more complex!
  - In practice, AdaBoost often does not overfit