Announcements

- HW 4 due next Wednesday (Apr 16th)

- Project Milestone 2 (Status Check-In) due in 2 weeks (Apr 23rd)
  - 3-page report

Lecture 22: Ensembles (Part 2)

CIS 4190/5190

Spring 2025

Recap: Ensemble Design Decisions

- How to learn the base models?
  - Bagging (randomize dataset)
  - Boosting (weighted dataset)

- How to combine the learned base models?
  - Averaging (regression) or majority vote (classification)

Recap: Bagging and Boosting

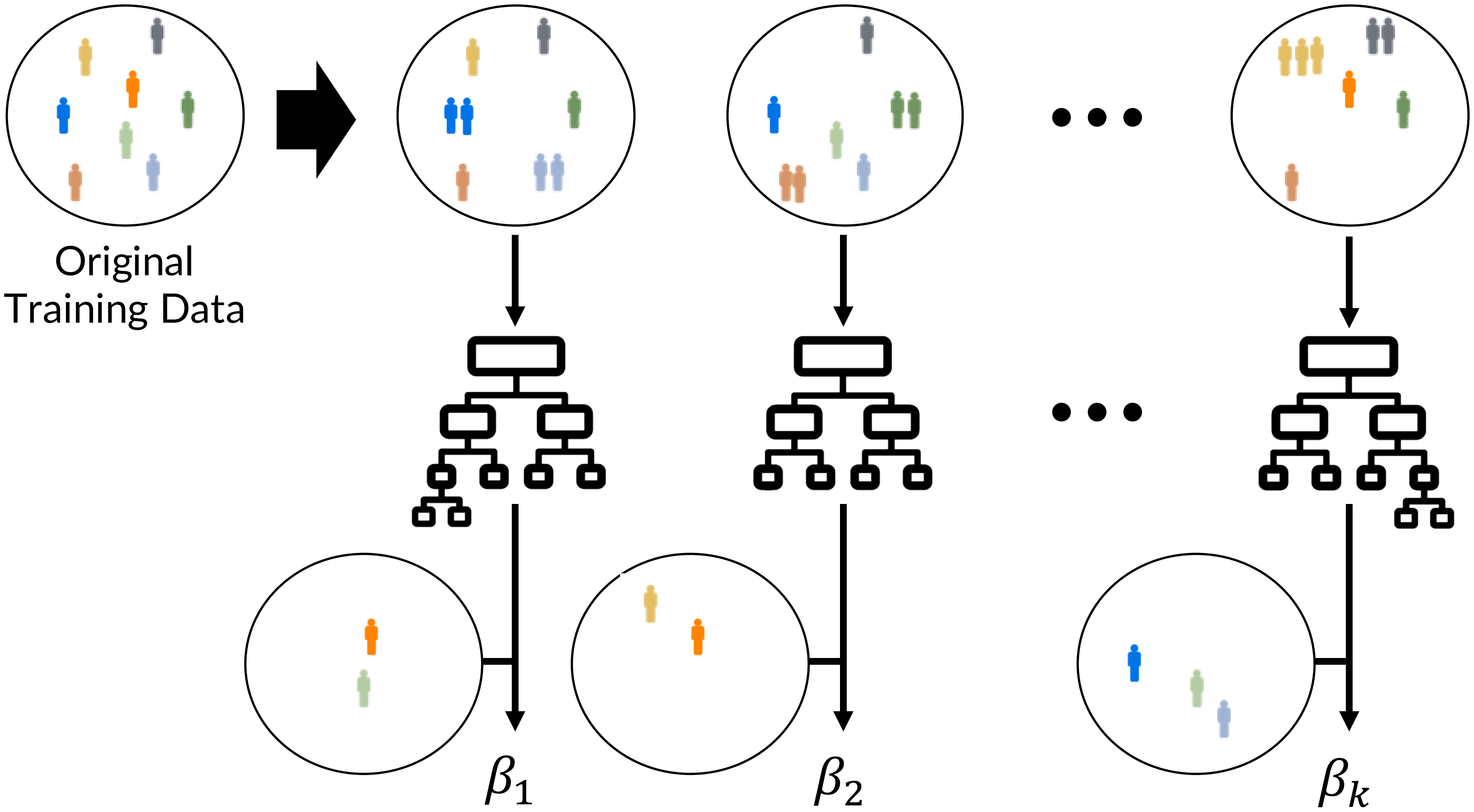
- Bagging (Boostrap Aggregating)
  - Main goal is to reduce variance
  - Models are trained independently with randomized dataset

- Boosting
  - Main goal is to reduce bias
  - Models are trained sequentially with weighted dataset

Recap: Bagging

- **Step 1:** Create bootstrap replicates of the original training dataset
  - Excludes $\left(1-\frac{1}{n}\right)^n \to \frac{1}{e}$ of the training examples ($n \to \infty$).
- **Step 2:** Train a classifier for each replicate
- **Step 3 (Optional):** Use held-out validation set to weight models
  - Can just use average predictions

Recap: Bagging



Recap: Random Forests

- Ensemble of decision trees using bagging
  - Typically use simple average (over probabilities for classification)

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

Recap: Random Forests

- **Tweak 1:** Randomize features in learning algorithm instead of bagging
  - 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

Recap: AdaBoost

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

Recap: Learning with Weighted Examples

For MSE loss:

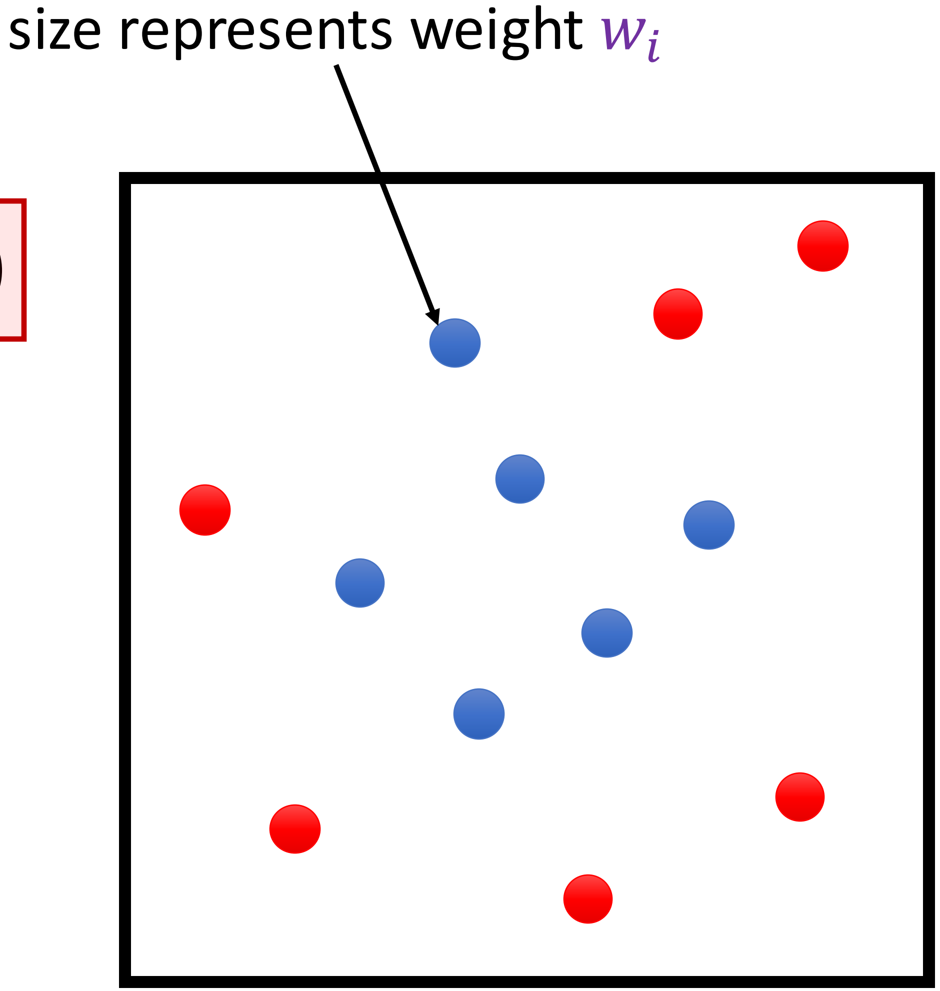
$$\ell(\beta; Z, w) = \sum_{i=1}^{n} w_i \cdot \|y_i - f_\beta(x_i)\|_2^2$$

For maximum likelihood estimation:

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

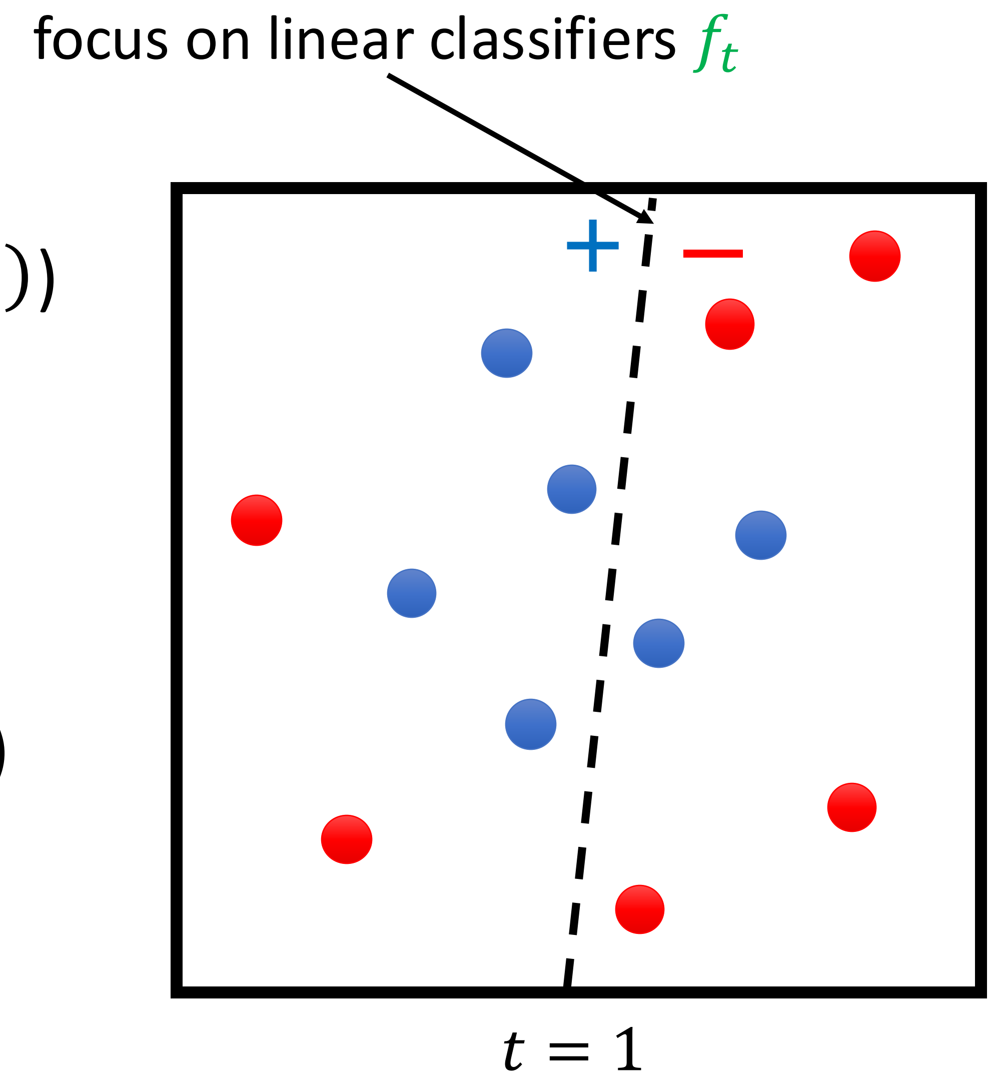
AdaBoost

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. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ 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** $F(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \beta_t \cdot f_t(x)\right)$

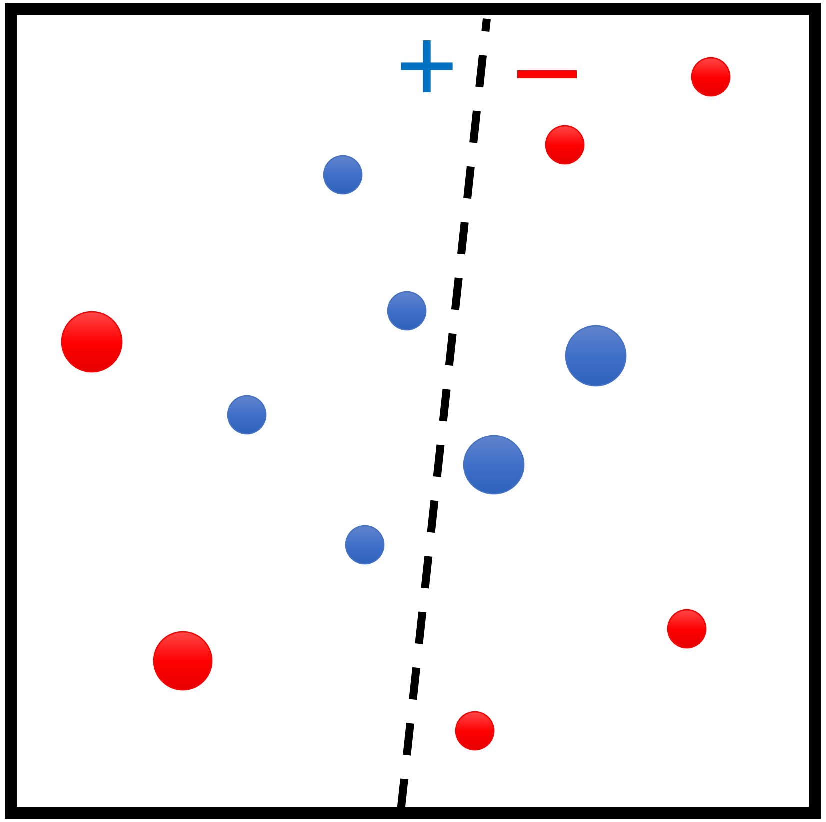


AdaBoost

- β_t measures the importance of $f_t()$
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$
  - otherwise flip h_t's predictions

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. $\quad \boxed{\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}}$
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** $F(x) = \mathrm{sign}(\sum_{t=1}^{T} \beta_t \cdot f_t(x))$

β_t becomes larger as
ϵ_t becomes smaller

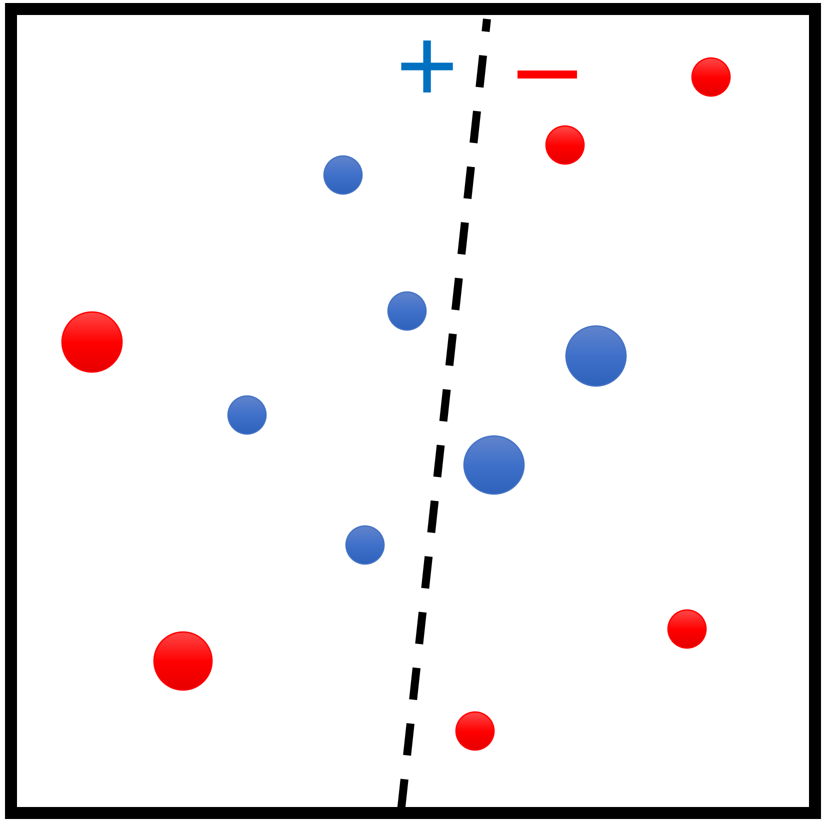


$t = 1$

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. $\quad \boxed{w_{t+1,i} \propto w_{t,i} \cdot e^{-\beta_t \cdot y_i \cdot f_t(x_i)} \text{ (for all } i\text{)}}$
7. **return** $F(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \beta_t \cdot f_t(x)\right)$

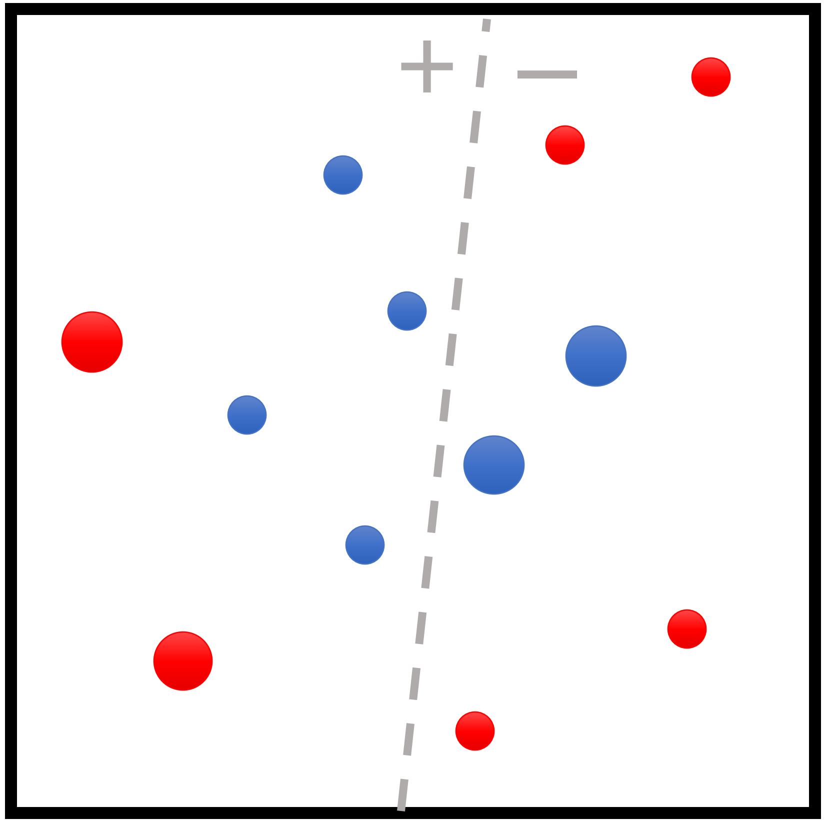
Use convention $y_i \in \{-1, +1\}$
If correct ($y_i = f_t(x_i)$) then multiply by $e^{-\beta_t}$
If incorrect ($y_i \neq f_t(x_i)$) then multiply by e^{β_t}



$t = 1$

AdaBoost

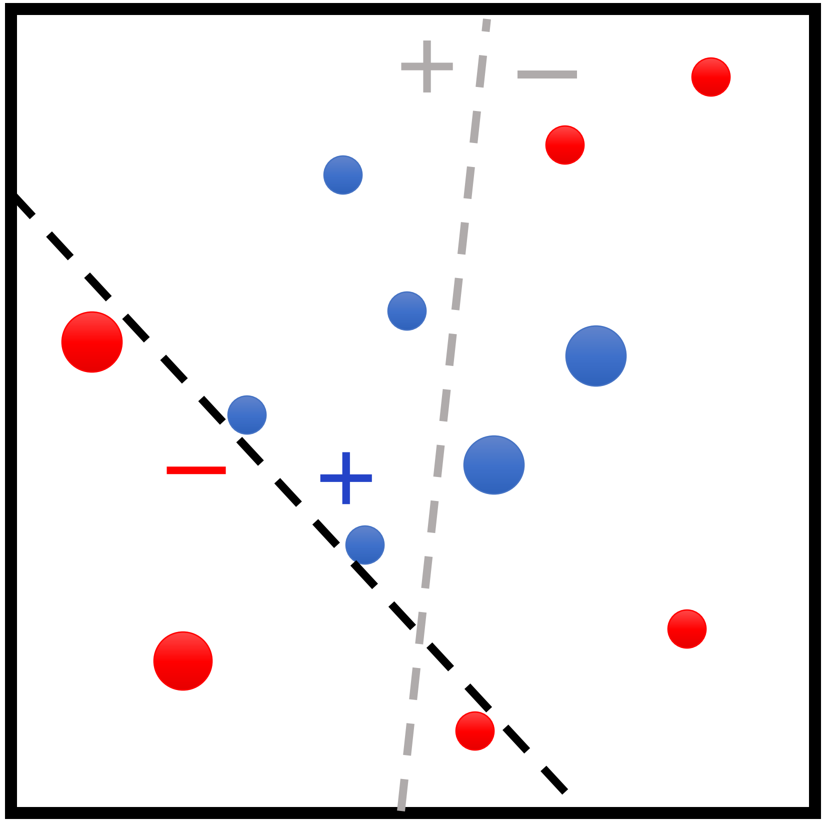
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
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

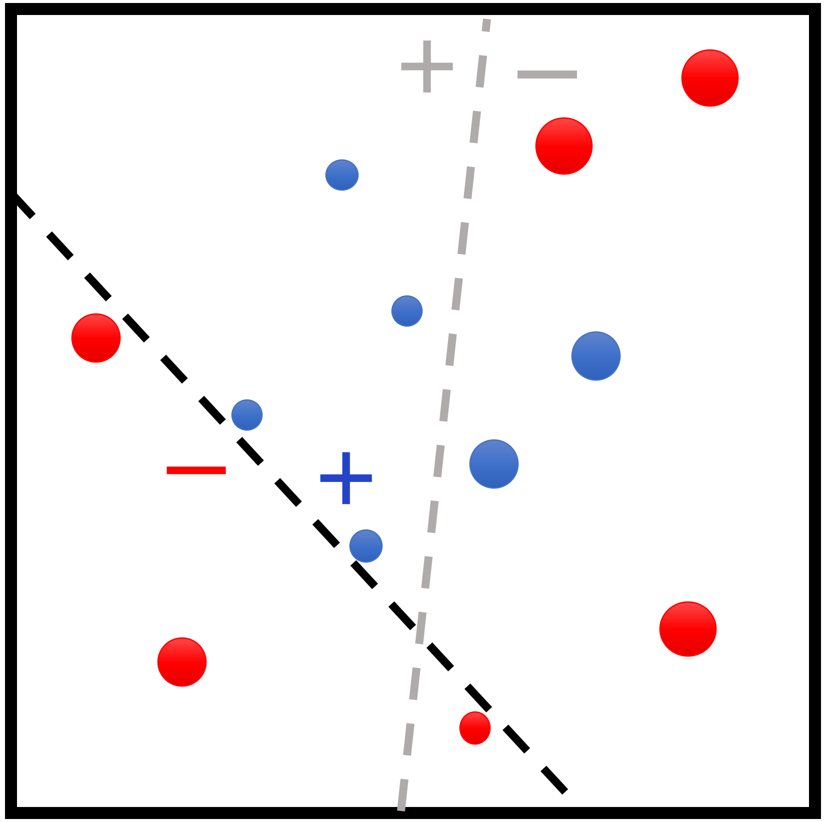
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $\boxed{f_t \leftarrow \mathrm{Train}(Z, w_t)}$
4. $\epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 2$

AdaBoost

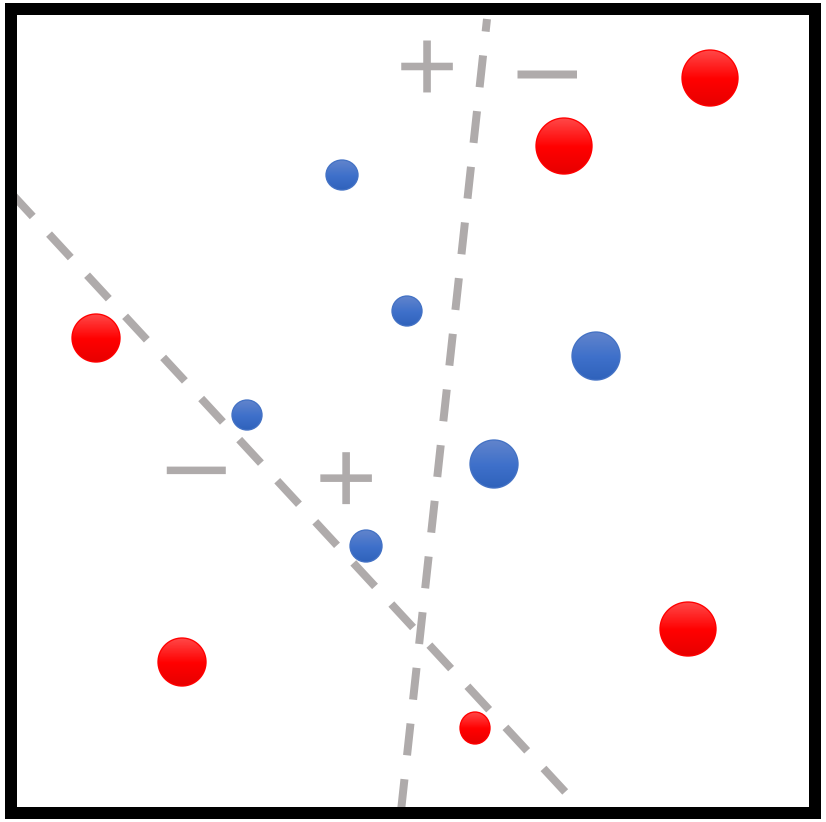
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 2$

AdaBoost

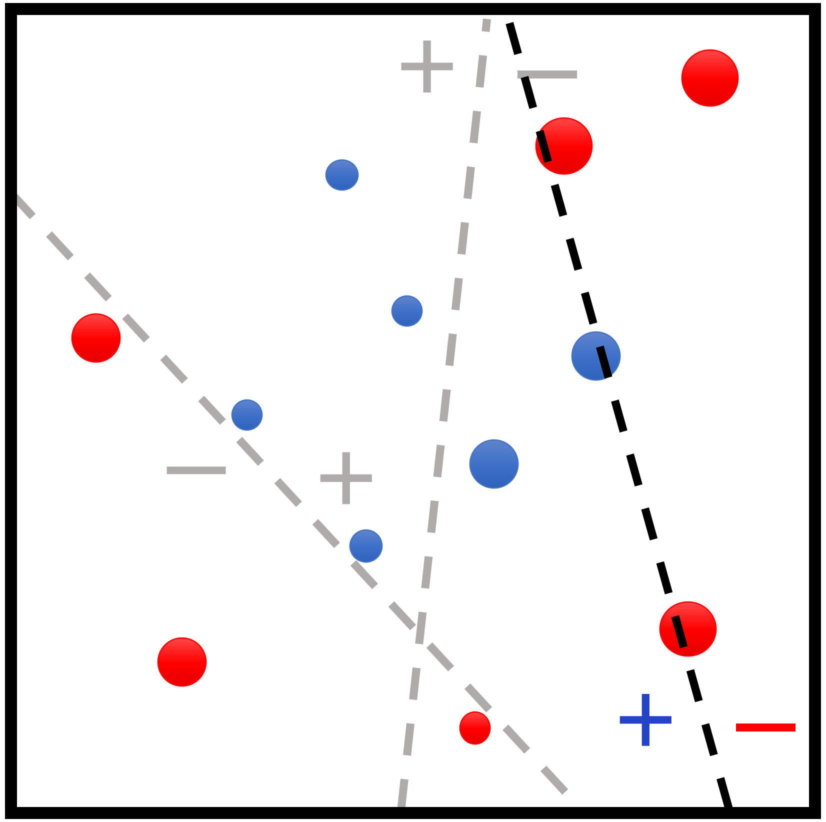
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \text{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
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

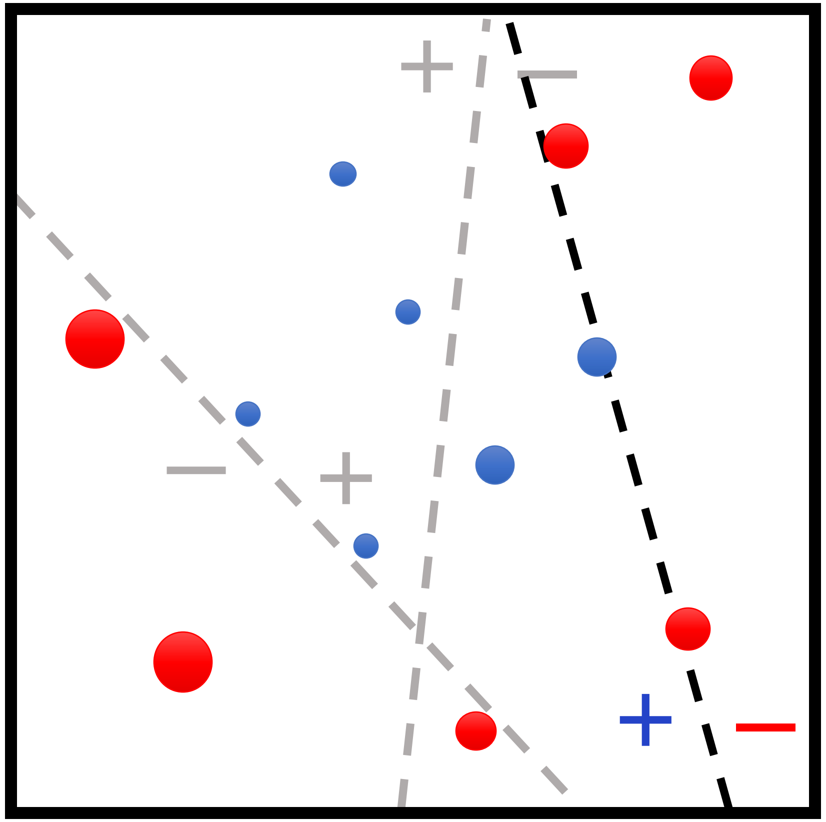
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
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

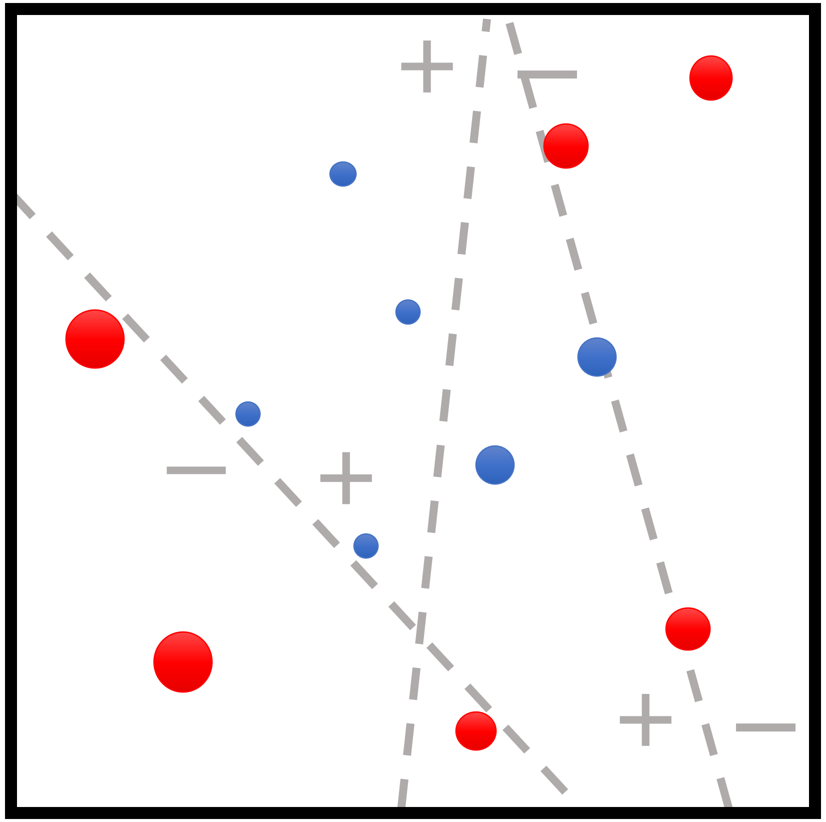
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
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

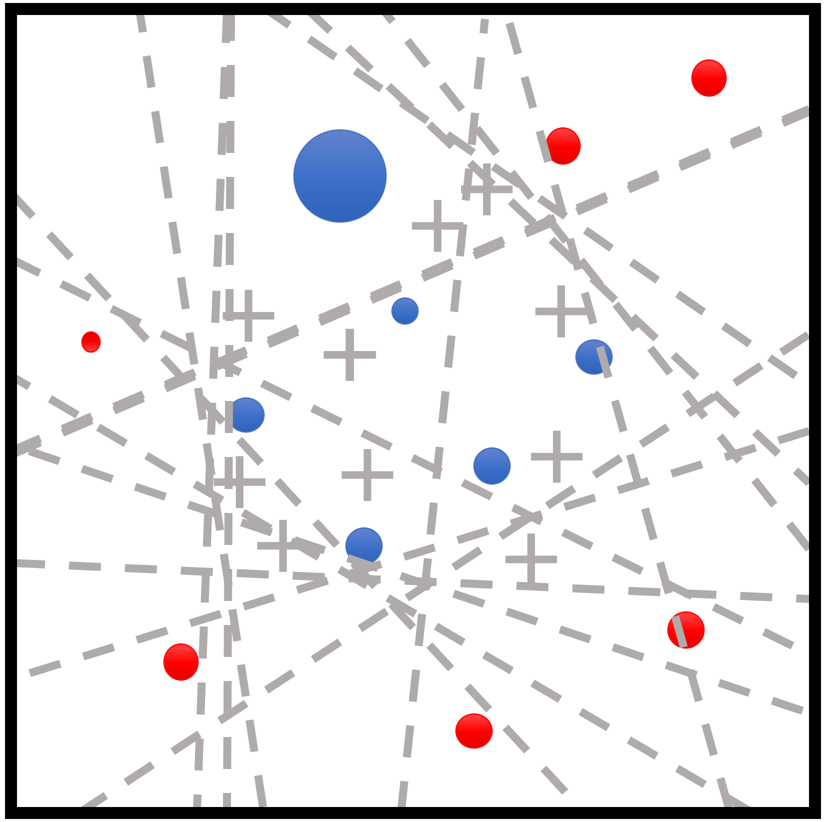
1. $w_1 \leftarrow \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \ldots, T\}$
3. $\quad f_t \leftarrow \text{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \text{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = 3$

AdaBoost

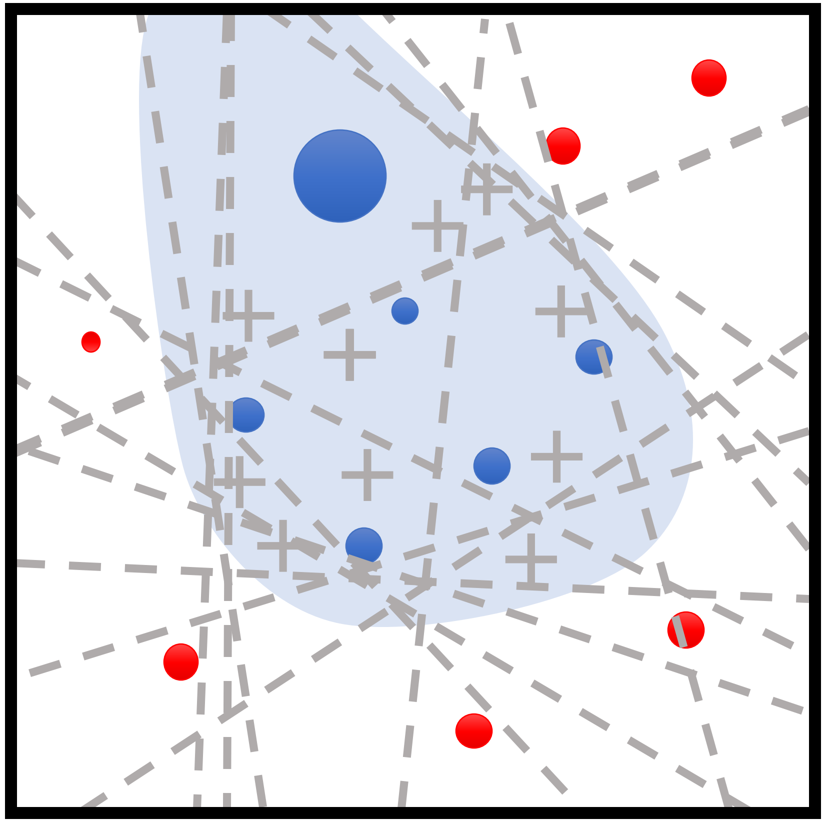
1. $w_1 \leftarrow \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ ($w_{1,i}$ weight for (x_i, y_i))
2. **for** $t \in \{1, \dots, T\}$
3. $\quad f_t \leftarrow \mathrm{Train}(Z, w_t)$
4. $\quad \epsilon_t \leftarrow \mathrm{Error}(f_t, Z, w_t)$
5. $\quad \beta_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$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



$t = T$

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** $F(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \beta_t \cdot f_t(x)\right)$



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!**

Boosting as Gradient Descent

- Both algorithms: new model $=$ old model $+$ update

- **Gradient Descent:**

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_\theta L(\theta_t; Z)$$

- **Boosting:**

$$F_{t+1}(x) = F_t(x) + \beta_{t+1} \cdot f_{t+1}(x)$$

- Here, $F_t(x) = \sum_{i=1}^{t} \beta_i \cdot f_i(x)$

Boosting as Gradient Descent

- Assuming $\beta_t = 1$ for all t, then:

$$F_t(x_i) + f_{t+1}(x_i) = F_{t+1}(x_i)$$

Boosting as Gradient Descent

- Assuming $\beta_t = 1$ for all t, then:

$$F_t(x_i) + f_{t+1}(x_i) = F_{t+1}(x_i) \approx y_i$$

- Rewriting this equation, we have

$$f_{t+1}(x_i) = F_{t+1}(x_i) - F_t(x_i) \approx \underbrace{y_i - F_t(x_i)}_{\text{“residuals”, i.e., error of the current model}}$$

Boosting as Gradient Descent

- In other words, at each step, boosting is training the next model f_{t+1} to approximate the residual:

$$f_{t+1}(x_i) \approx \underbrace{y_i - F_t(x_i)}_{\text{“residuals”, i.e., error of the current model}}$$

- **Idea:** Train f_{t+1} directly to predict residuals $y_i - F_t(x_i)$

- **This strategy works for regression as well!**

Boosting as Gradient Descent

- **Algorithm:** For each $t \in \{1, \ldots, T\}$:
  - **Step 1:** Train f_{t+1} using dataset

$$Z_{t+1} = \{(x_i, y_i - F_t(x_i))\}_{i=1}^{n}$$

  - **Step 2:** Take

$$F_{t+1}(x) = F_t(x) + f_{t+1}(x)$$

- Return the final model F_T

Boosting as Gradient Descent

- Consider losses of the form

$$L(F; Z) = \frac{1}{n} \sum_{i=1}^{n} \tilde{L}(F(x_i); y_i)$$

- In other words, sum of individual label-level losses $\tilde{L}(\hat{y}; y)$ of a prediction $\hat{y} = F(x)$ if the ground truth label is y
- For example, $\tilde{L}(\hat{y}; y) = \frac{1}{2}(y\hat{} - y)^2$ yields the MSE loss

Boosting as Gradient Descent

- Residuals are the gradient of the squared error $\tilde{L}(\hat{y}; y) = \frac{1}{2}(y - \hat{y})^2$:

$$-\frac{\partial \tilde{L}}{\partial \hat{y}}(F_t(x_i); y_i) = y_i - F_t(x_i) = \text{residual}_\text{i}$$

- For general $\tilde{L}$, instead of $\{(x_i, y_i - F_t(x_i))\}_{i=1}^{n}$ we can train f_{t+1} on

$$Z_{t+1} = \left\{\left(x_i, -\frac{\partial \tilde{L}}{\partial \hat{y}}(F_t(x_i); y_i)\right)\right\}_{i=1}^{n}$$

Boosting as Gradient Descent

- **Algorithm:** For each $t \in \{1, \ldots, T\}$:
  - **Step 1:** Train f_{t+1} using dataset

$$Z_{t+1} = \{(x_i, y_i - F_t(x_i))\}_{i=1}^{n}$$

  - **Step 2:** Take

$$F_{t+1}(x) = F_t(x) + f_{t+1}(x)$$

- Return the final model F_T

Boosting as Gradient Descent

- **Algorithm:** For each $t \in \{1, \ldots, T\}$:
  - **Step 1:** Train f_{t+1} using dataset

$$Z_{t+1} = \left\{ \left(x_i, -\frac{\partial \tilde{L}}{\partial \hat{y}} (F_t(x_i); y_i) \right) \right\}_{i=1}^{n}$$

  - **Step 2:** Take

$$F_{t+1}(x) = F_t(x) + f_{t+1}(x)$$

- Return the final model F_T

Boosting as Gradient Descent

- Casts ensemble learning in the **loss minimization framework**
  - **Model family:** Sum of base models $F_T(x) = \sum_{t=1}^{T} f_t(x)$
  - **Loss:** Any differentiable loss expressed as

$$L(F; Z) = \sum_{i=1}^{n} \tilde{L}(F(x_i), y_i)$$

- Gradient boosting is a general paradigm for training ensembles with specialized losses (e.g., most NLL losses)

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

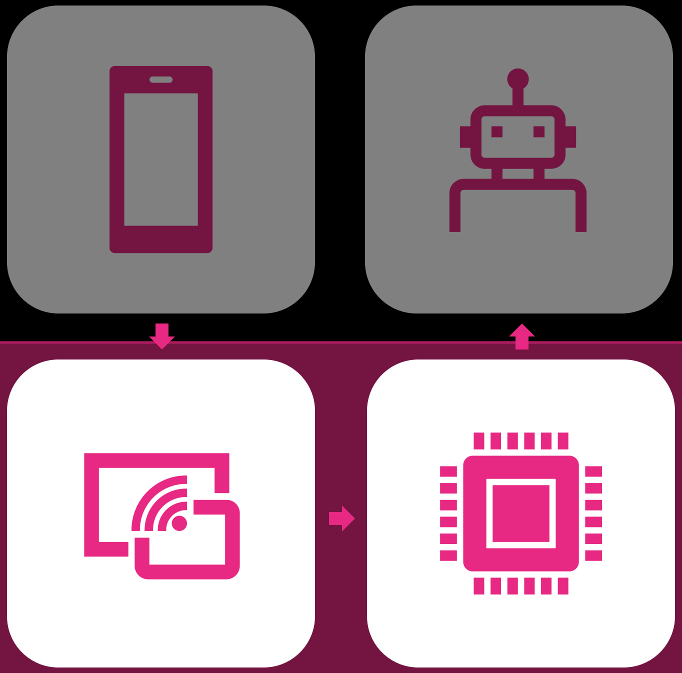
CIS 3990

Mobile and IoT Computing

Mobile and IoT Computing

The convergence of sensing, communication, and computation that allows us to:

Acquire data from the environment



Act based on the data

Aggregate data from multiple sources

Analyze data (cloud/edge) and provide insights about the world

This course

Sensing & Computing

WHAT?

Sensing Objectives

Locations

Health

Motion & Activity

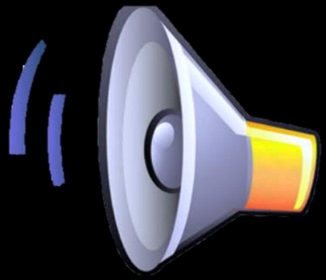
Environment

HOW?

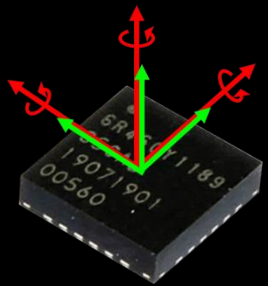
Sensing Modalities

Radio

Acoustic



Inertial



Visual

Example Mobile and IoT Systems

Through-Wall Vision

Mobile Security
Case Study: Inaudible Voice Commands

Can hack Android/Alexa using inaudible voice commands

What you are expected to learn from this class

Lectures:

- Fundamentals of Mobile and IoT Computing
- How are they applied across various industries?
- What are emerging IoT domains and what does the future of IoT look like?

Labs:

- iOS APIs, including Bluetooth, inertial, basic UI programming

Project:

- Build a physical IoT project using material learnt from class
- Collaboration

Course Projects

Course Projects: Two Options

- Project A: Image2GPS
  - Predicting camera location based on the photo.
- Project B: News Source Classification
  - Classifying the source of a news headline.

Course Projects: Two Parts

- Core Questions:
  - Describe how you address the core task (i.e., image2GPS or news source classification).
  - Describe the design, implementation and evaluation of your method/model.
- Exploratory Questions:
  - Define your own questions building on top of the core questions.
    - E.g., how much can ensembles help?
    - E.g., how much can pretrained language model help?
  - Motivation, related work, method, results, and discussions.

Course Projects: Evaluation

- Project Report
  - Document your method and results for core and exploratory questions
  - 3 pages for check-in and 5 pages for the final report
- Colab Notebook with inference results on our test data
  - Test data and code will be release later
  - Report your performance on the test data
  - Submit a Colab notebook with output of the code cells
- Summary Slides
- Demo Video (Optional)

Course Projects: Compute

You are strongly encouraged to use (relatively) small architectures.

- You should be able to use Google Colab for all evaluations
- Inference.ai: GPU compute resources (Juputer Notebook & SSH)
- You may also consider signing up for Amazon SageMaker Studio Lab
  - https://studiolab.sagemaker.aws

Welcome to Inference.ai EDU!

Setting up your first server

Course Projects: Q & A