



Ensemble Learning

These slides were assembled by Eric Eaton, with grateful acknowledgement of the many others who made their course materials freely available online. Feel free to reuse or adapt these slides for your own academic purposes, provided that you include proper attribution. Please send comments and corrections to Eric.

Ensemble Learning

Consider a set of classifiers h_1, \dots, h_L

Idea: construct a classifier $H(\mathbf{x})$ that combines the individual decisions of h_1, \dots, h_L

- e.g., could have the member classifiers vote, or
- e.g., could use different members for different regions of the instance space

Ensemble Learning

Consider a set of classifiers h_1, \dots, h_L

Idea: construct a classifier $H(\mathbf{x})$ that combines the individual decisions of h_1, \dots, h_L

- e.g., could have the member classifiers vote, or
- e.g., could use different members for different regions of the instance space
- Works well if the members each have low error rates

Successful ensembles require **diversity**

- Classifiers should make different mistakes
- Can have different types of base learners

Practical Application: Netflix Prize

Goal: predict how a user will rate a movie

- Based on the user's ratings for other movies
- and other peoples' ratings
- with no other information about the movies

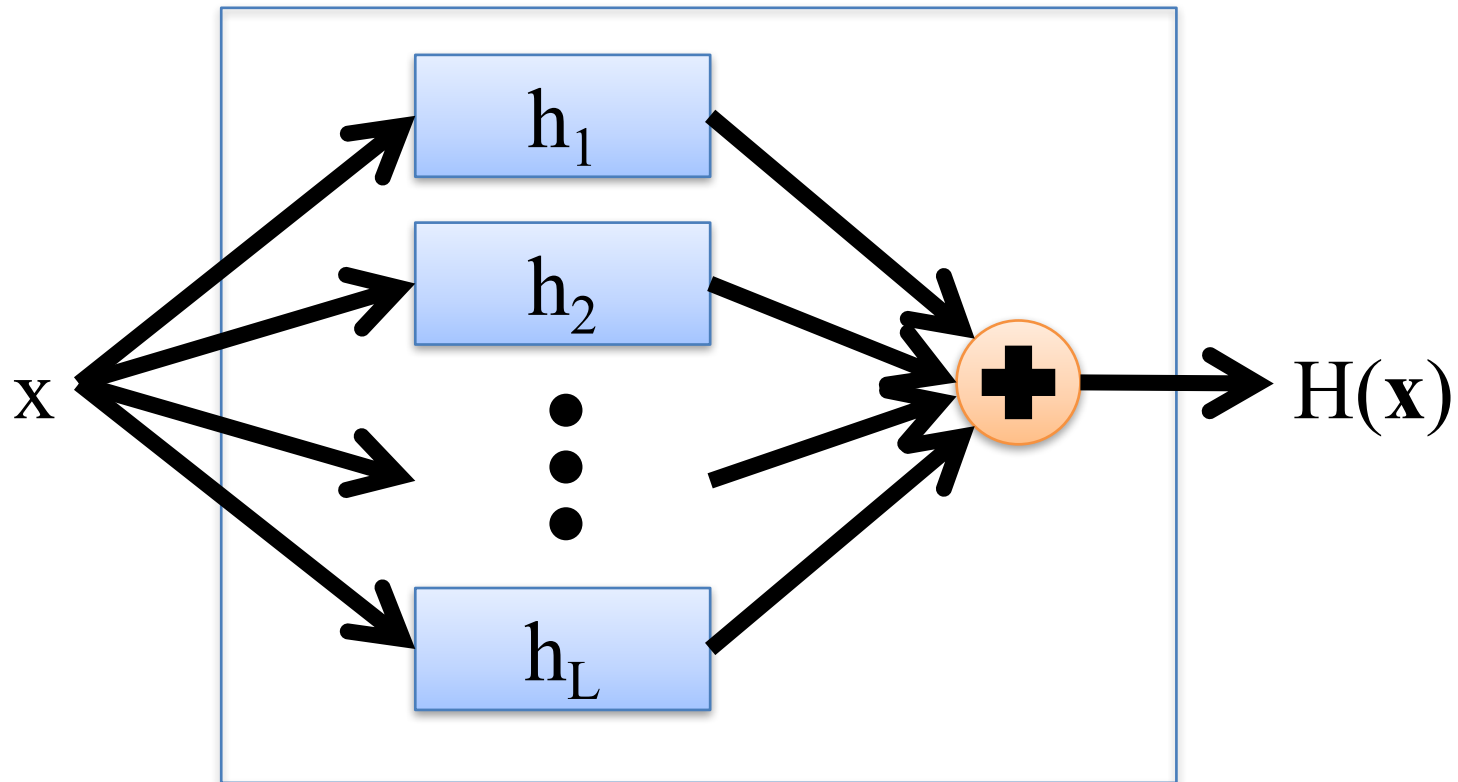


This application is called “collaborative filtering”

Netflix Prize: \$1M to the first team to do 10% better than Netflix' system (2007-2009)

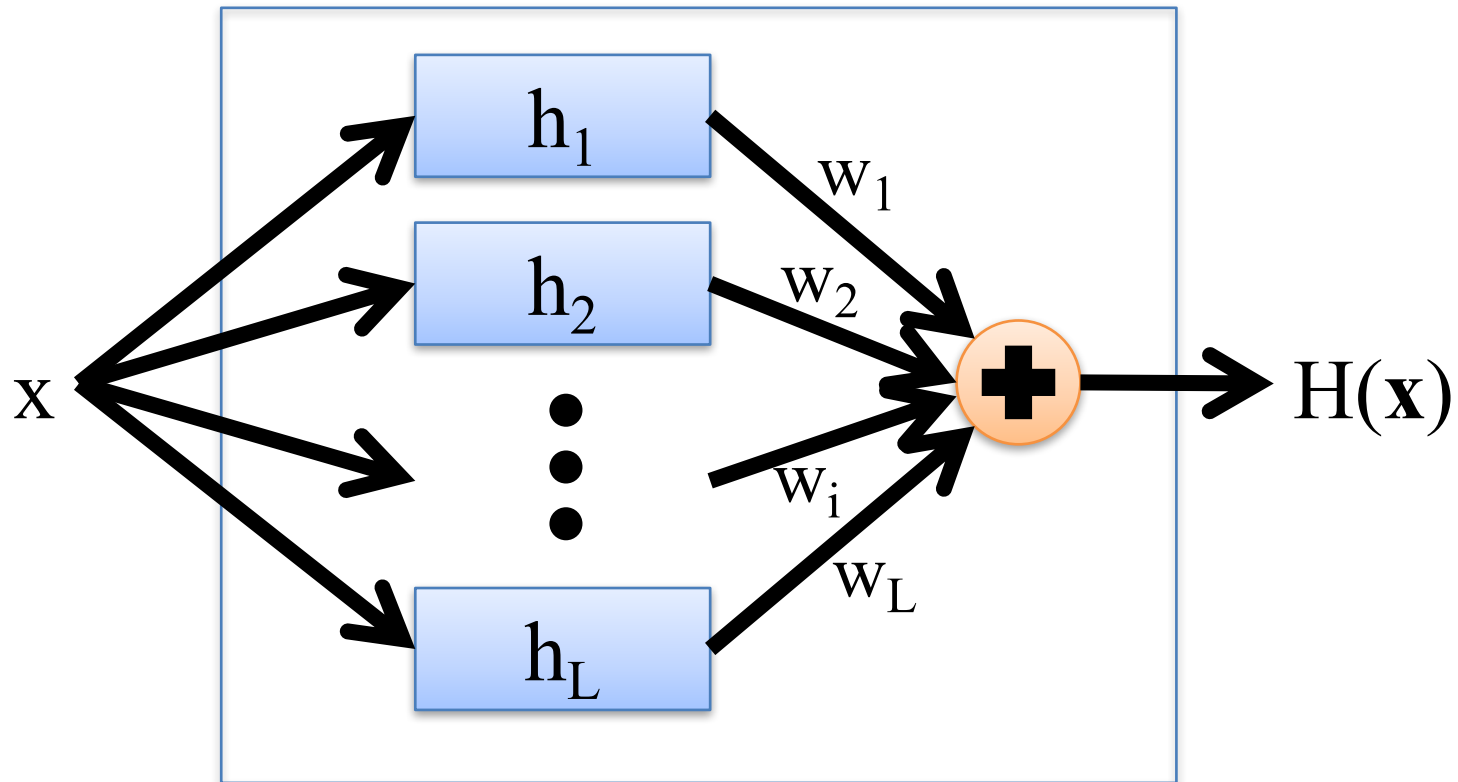
Winner: BellKor's Pragmatic Chaos – an ensemble of more than 800 rating systems

Combining Classifiers: Averaging



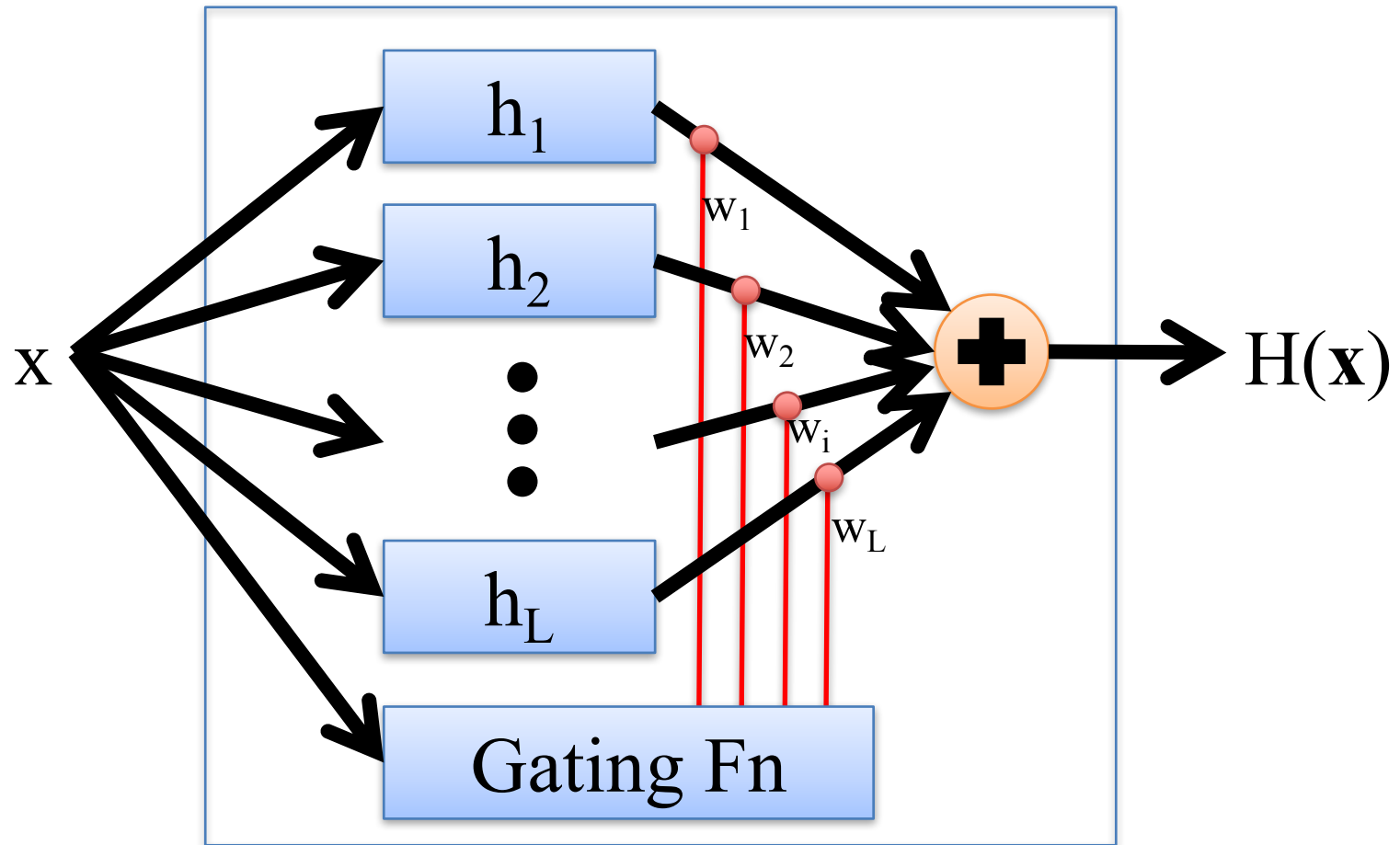
- Final hypothesis is a simple vote of the members

Combining Classifiers: Weighted Average



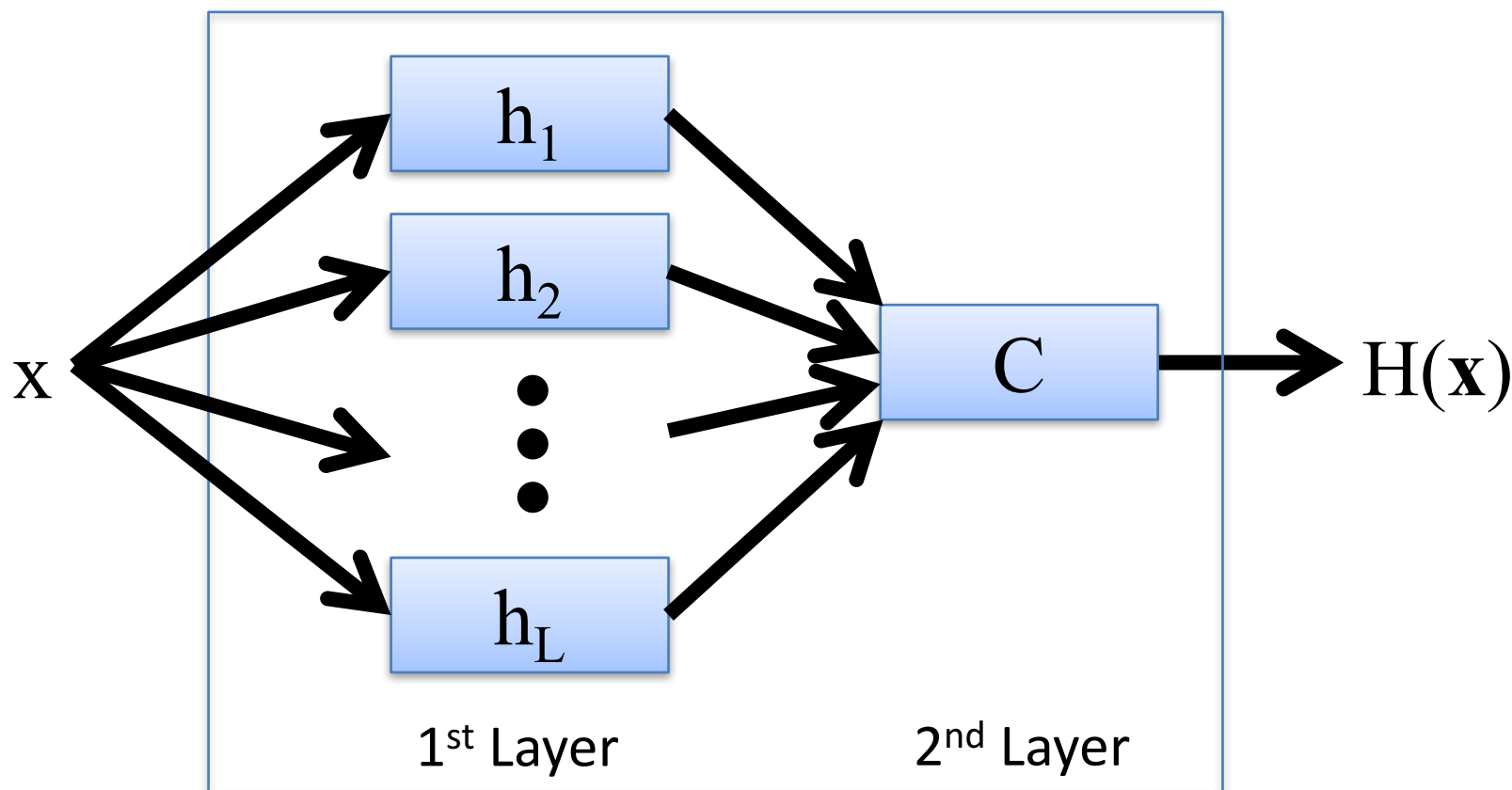
- Coefficients of individual members are trained using a validation set

Combining Classifiers: Gating



- Coefficients of individual members depend on input
- Train gating function via validation set

Combining Classifiers: Stacking



- Predictions of 1st layer used as input to 2nd layer
- Train 2nd layer on validation set

How to Achieve Diversity

Cause of the Mistake

Pattern was difficult

Overfitting

Some features are noisy

Diversification Strategy

Hopeless

Vary the training sets

Vary the set of input features

Manipulating the Training Data

Bootstrap replication:

- Given n training examples, construct a new training set by sampling n instances with replacement
- Excludes $\sim 30\%$ of the training instances

Bagging:

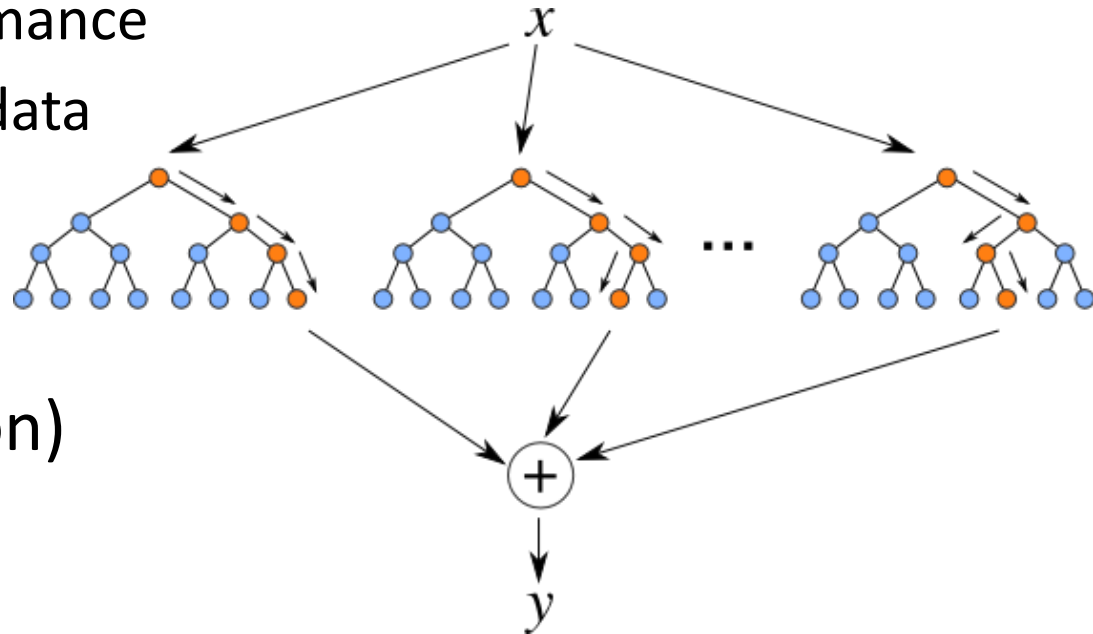
- Create bootstrap replicates of training set
- Train a classifier (e.g., a decision tree) for each replicate
- Estimate classifier performance using out-of-bootstrap data
- Average output of all classifiers

Boosting: (in just a minute...)

Manipulating the Features

Random Forests

- Construct decision trees on bootstrap replicas
 - Restrict the node decisions to a small subset of features picked randomly for each node
- Do not prune the trees
 - Estimate tree performance on out-of-bootstrap data
- Average the output of all trees (or choose mode decision)



Adaptive Boosting (AdaBoost)

AdaBoost

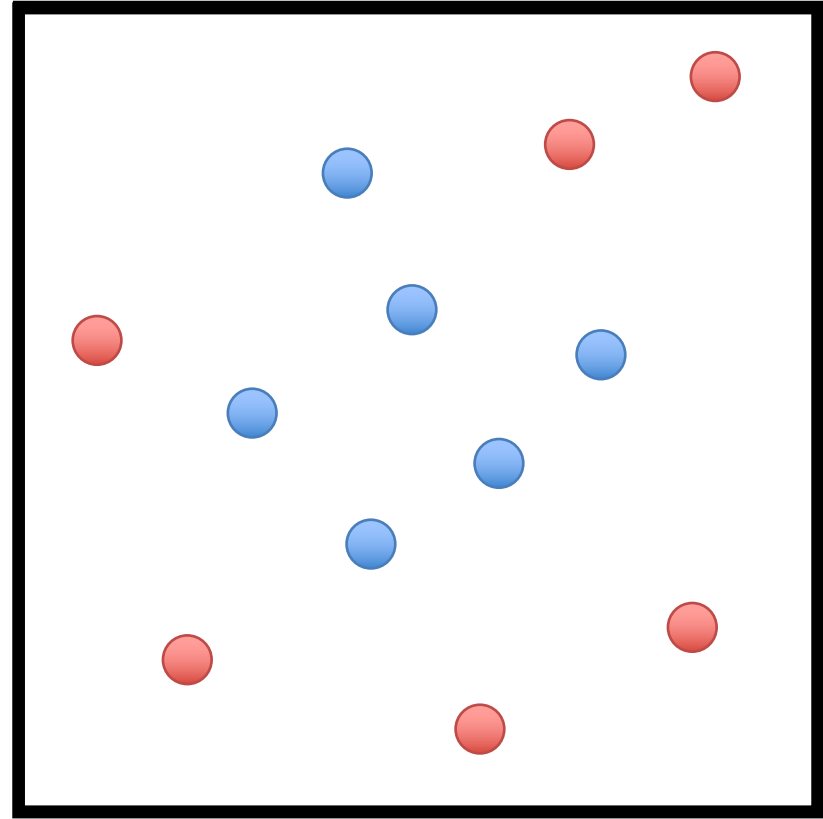
[Freund & Schapire, 1997]

- A meta-learning algorithm with great theoretical and empirical performance
- Turns a base learner (i.e., a “weak hypothesis”) into a high performance classifier
- Creates an ensemble of weak hypotheses by repeatedly emphasizing mispredicted instances

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

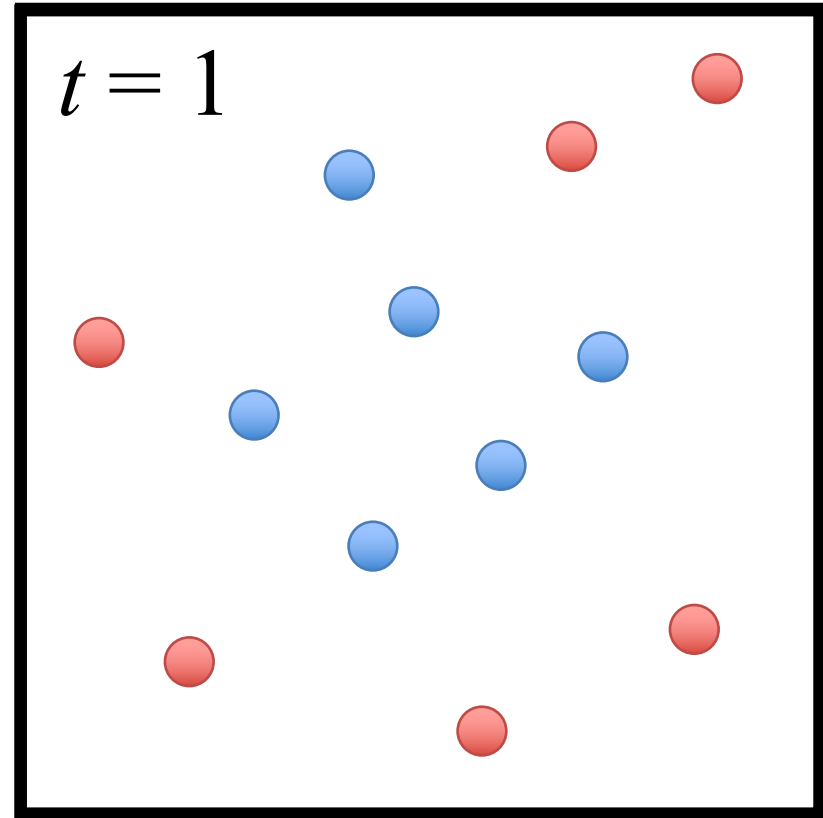


- Size of point represents the instance's weight

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

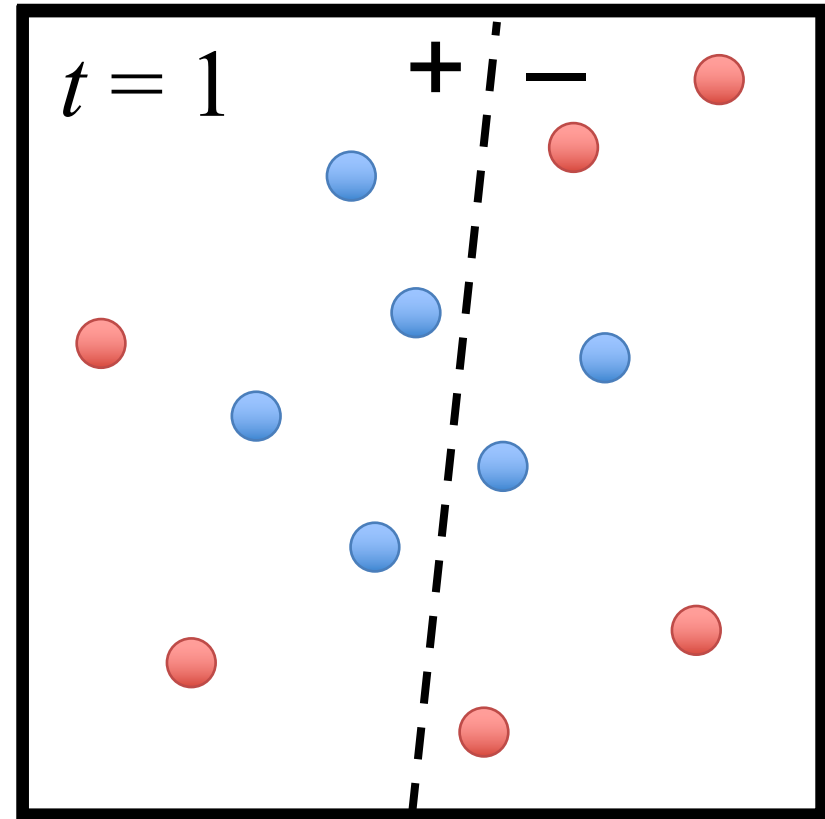
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

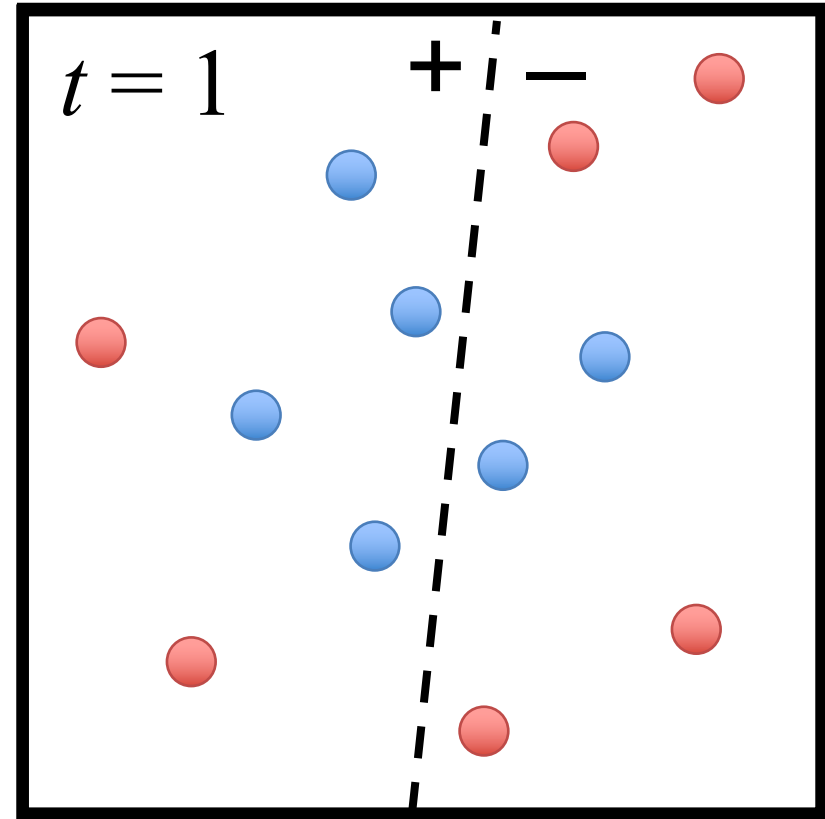
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

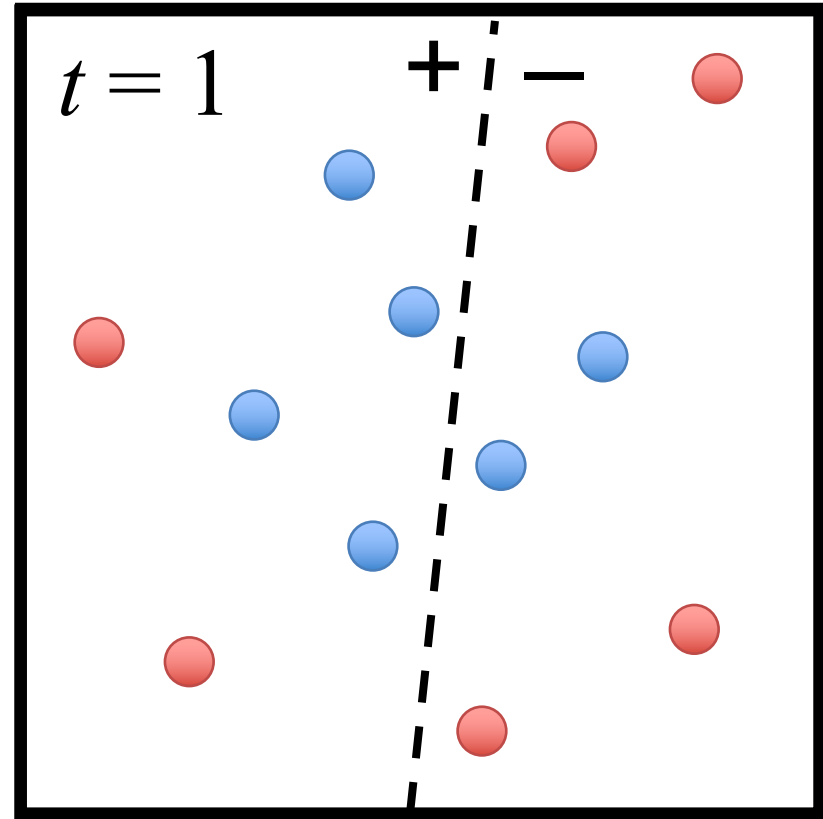


- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (can trivially guarantee)

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

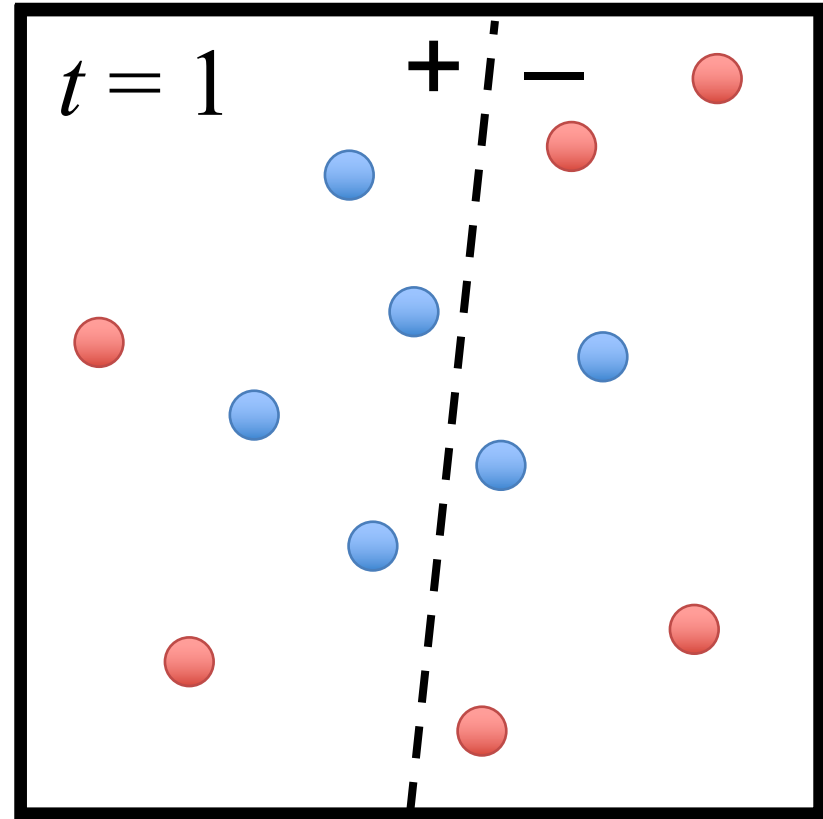


- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

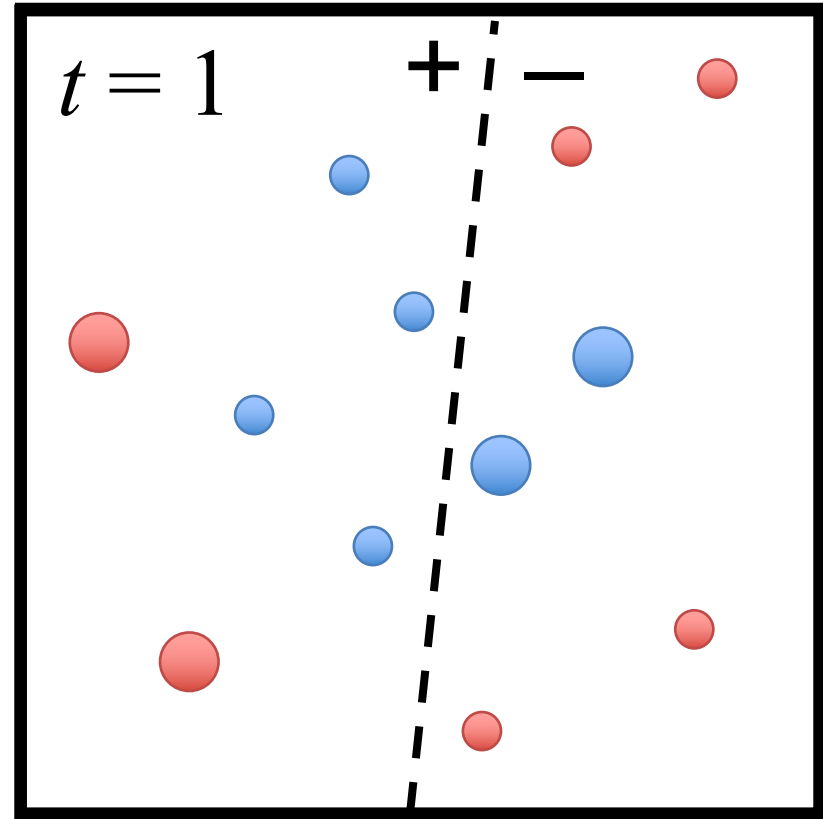


- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

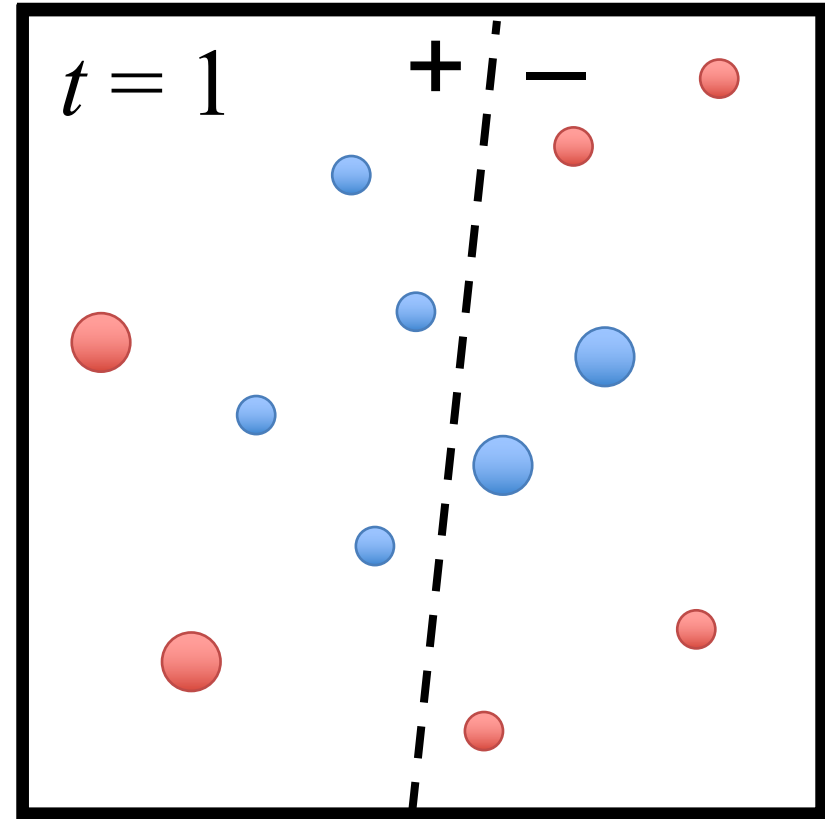


- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

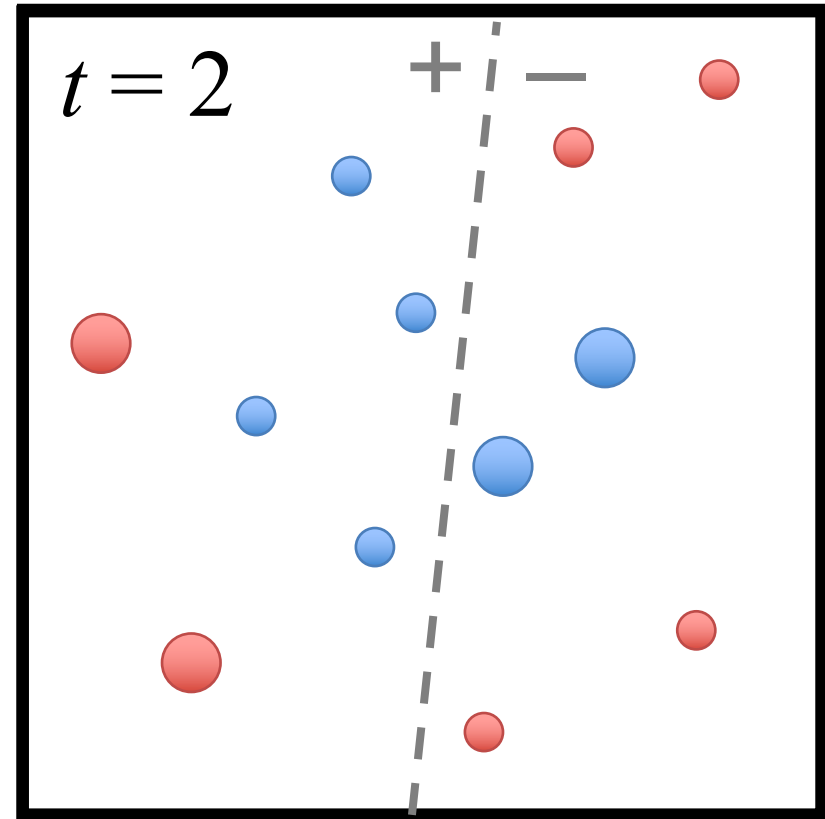


Disclaimer: Note that resized points in the illustration above are not necessarily to scale with β_t

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

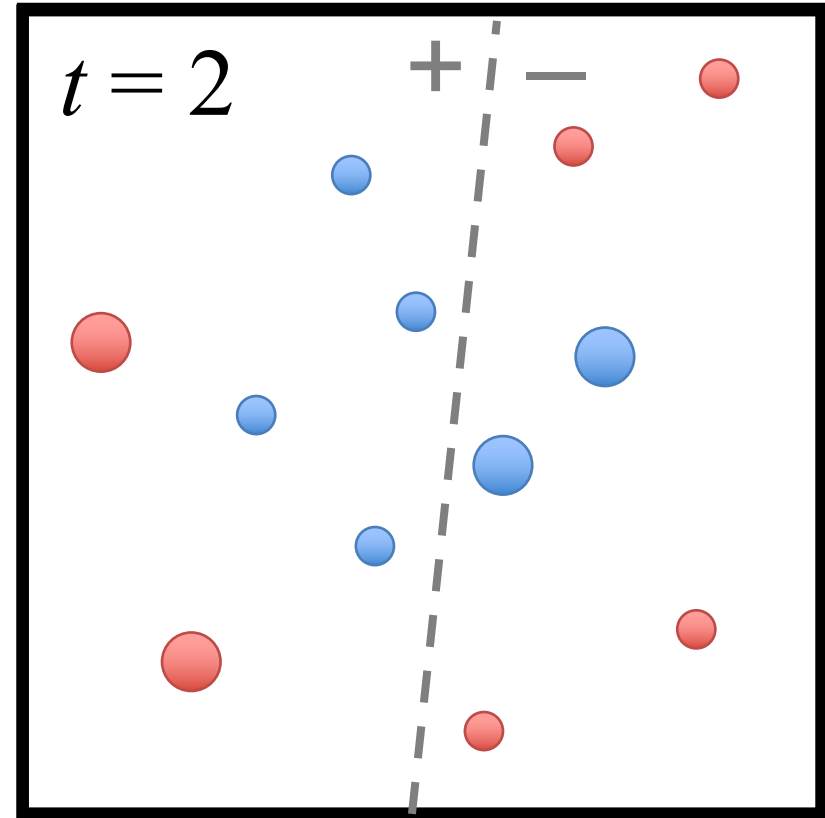
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

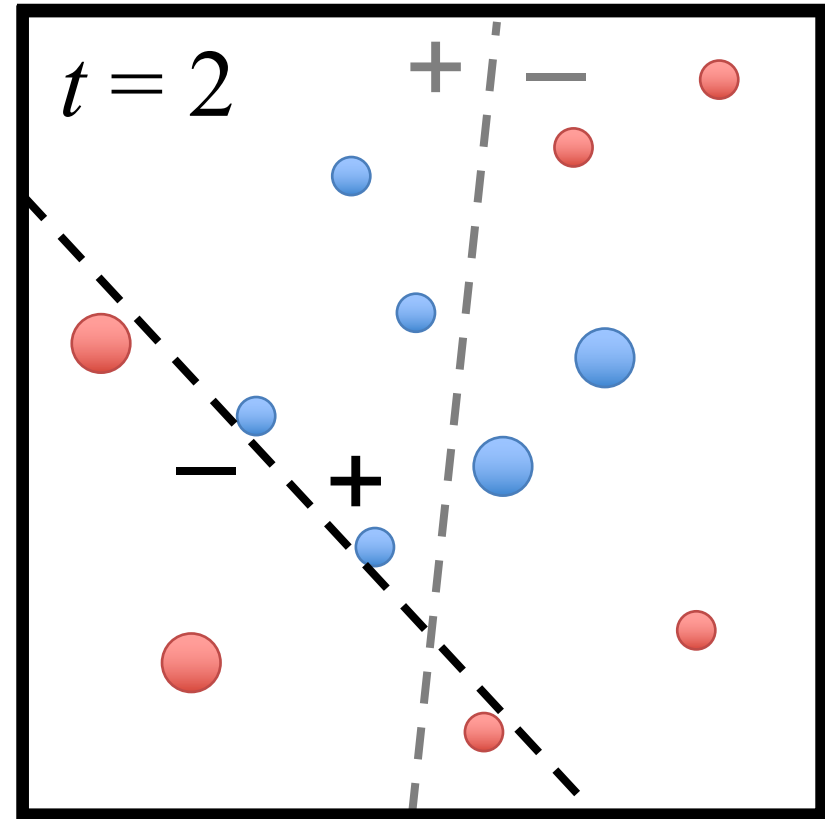
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

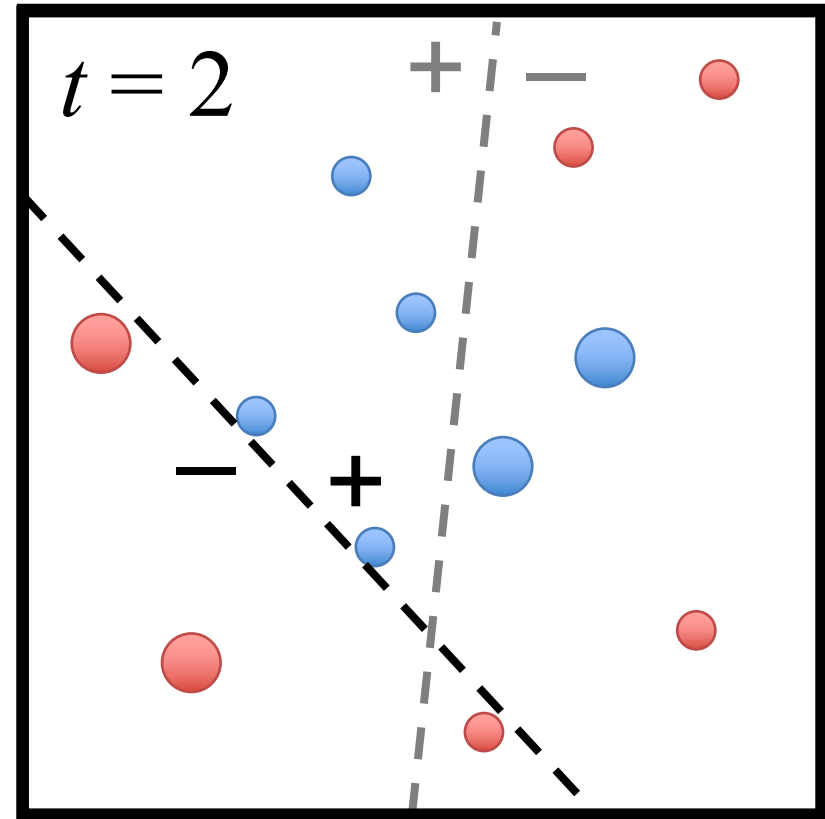
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

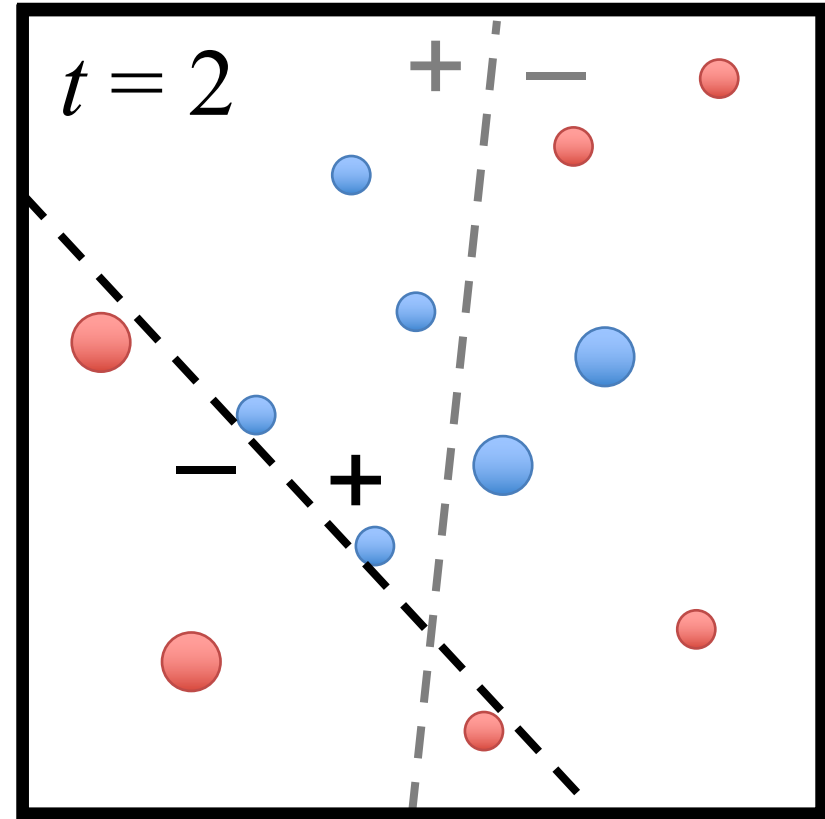


- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

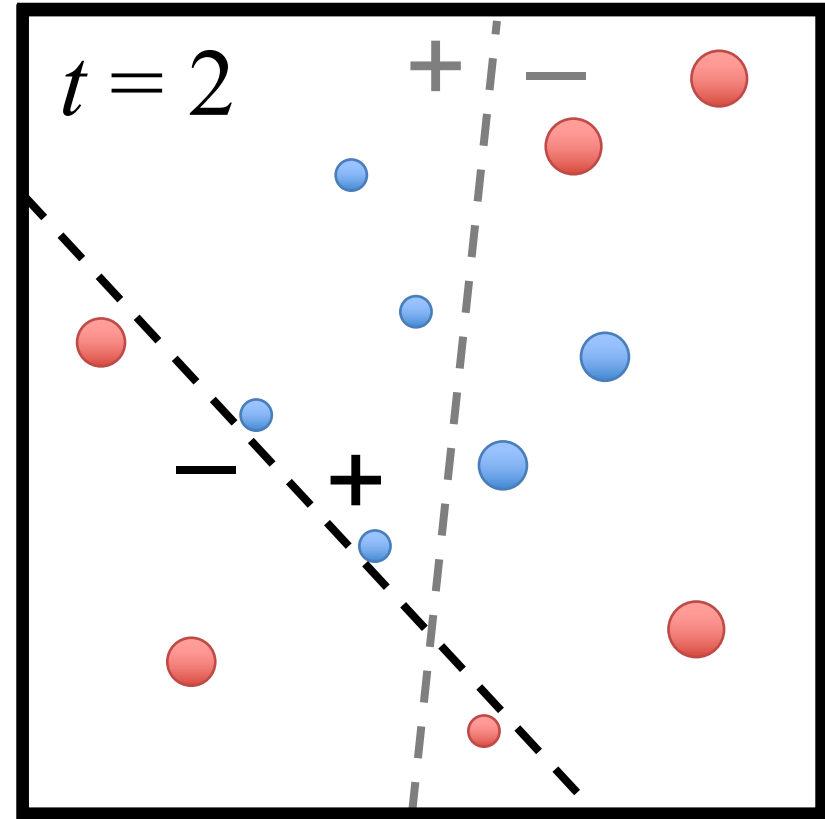


- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

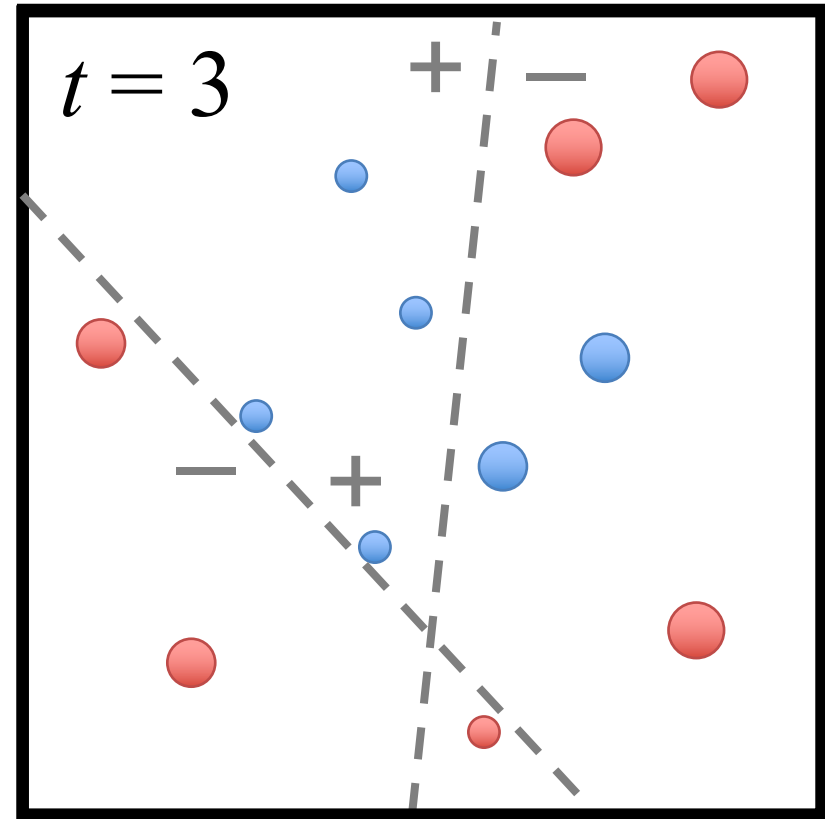


- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

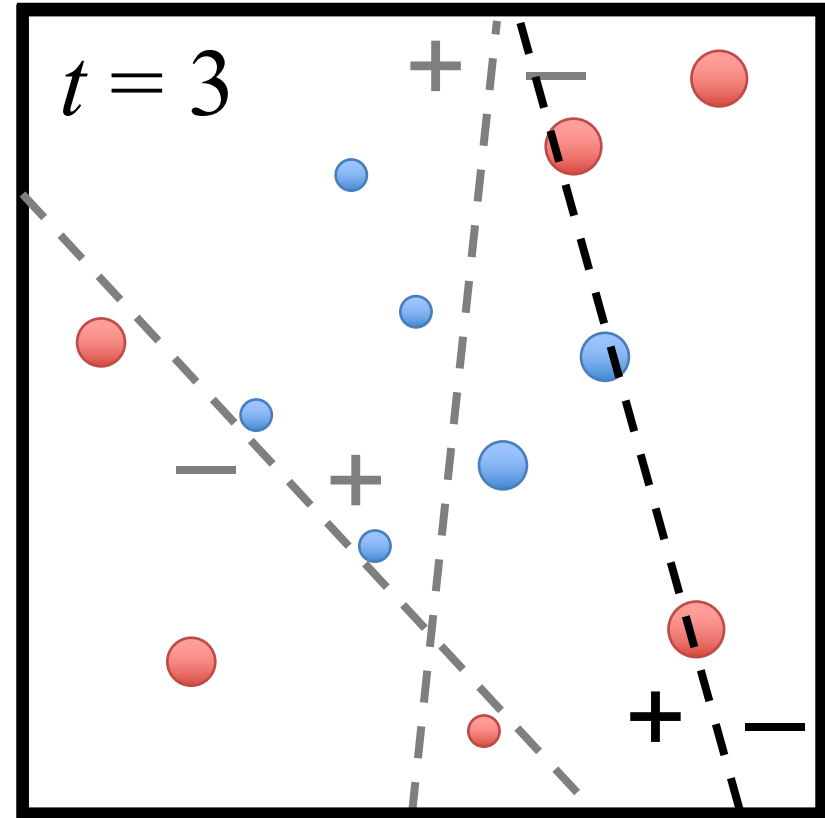
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

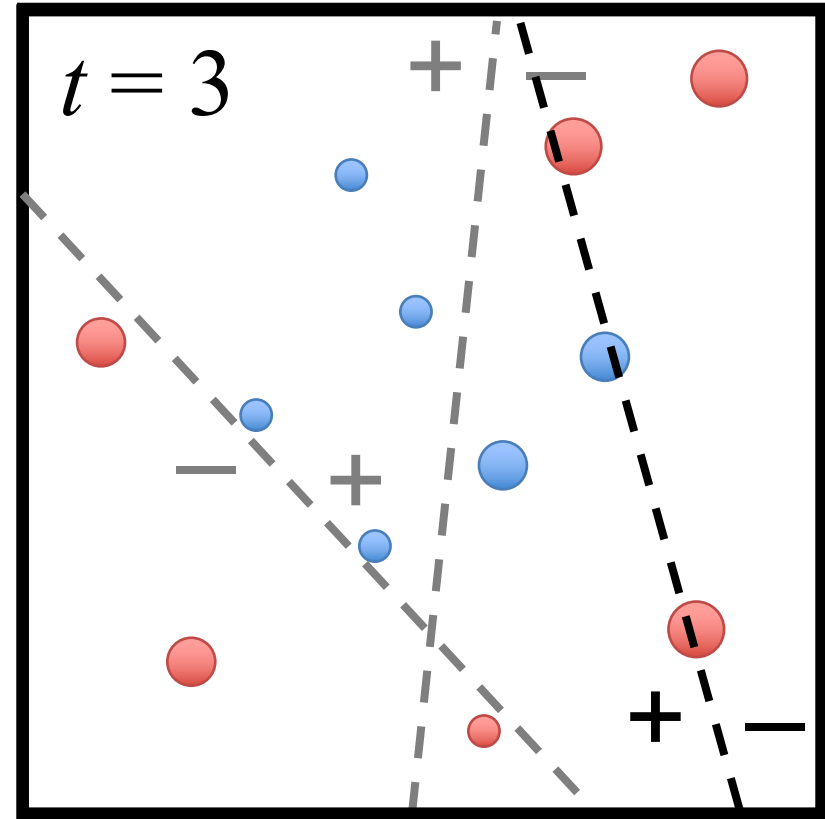
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

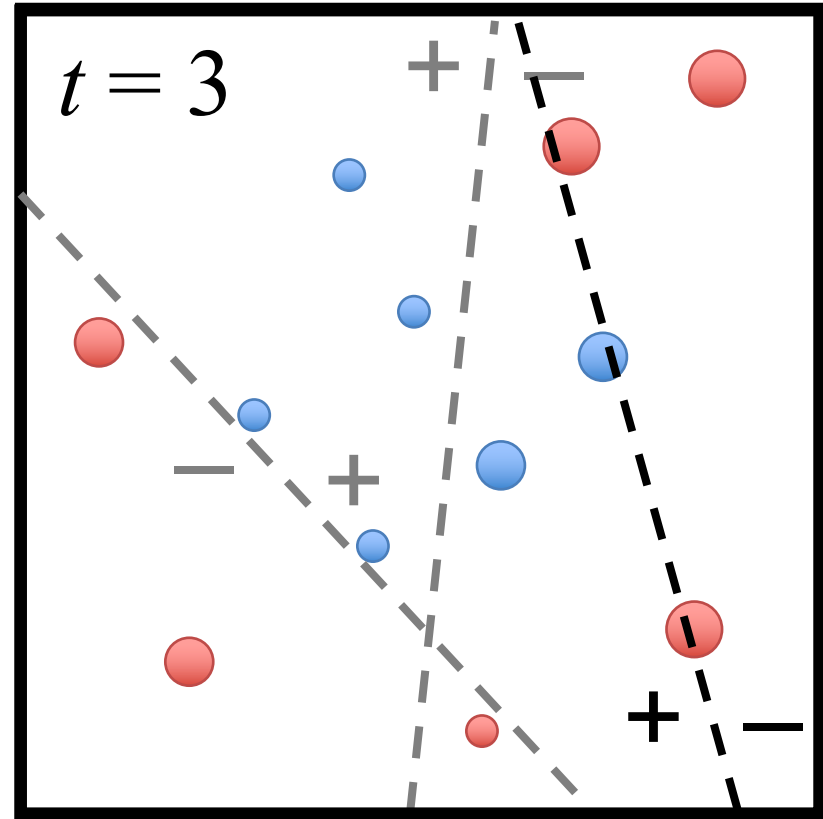


- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$ (β_t grows as ϵ_t gets smaller)

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

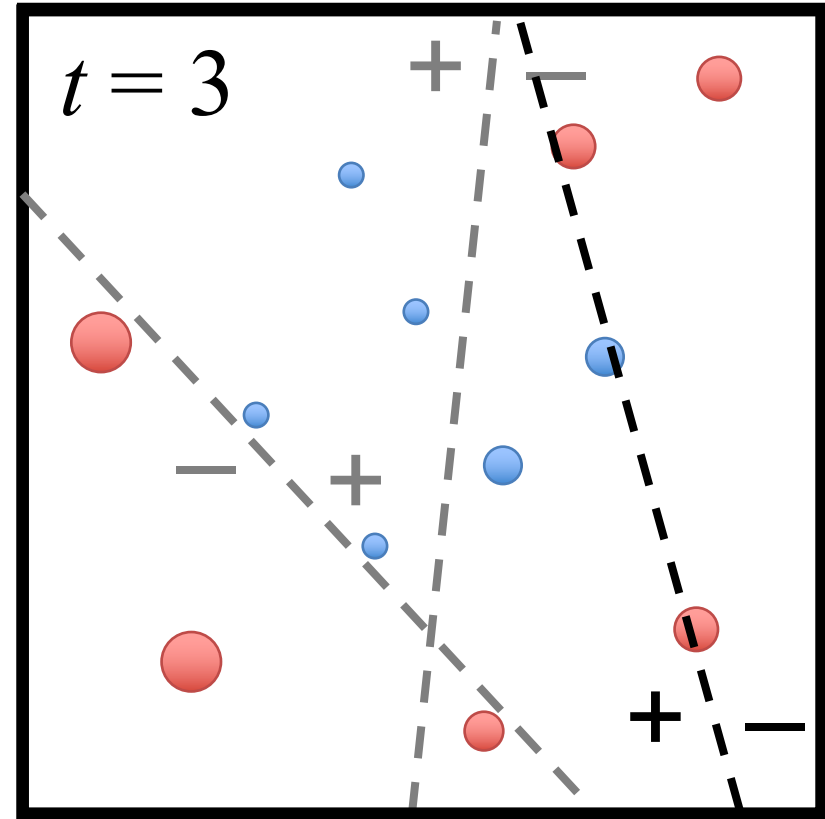


- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

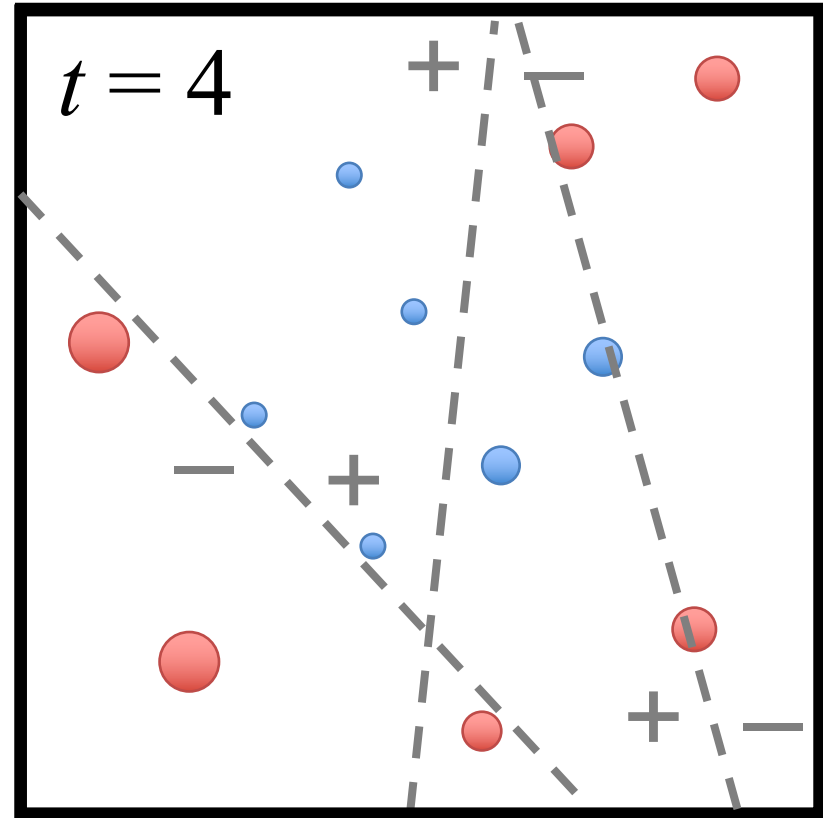


- Weights of correct predictions are multiplied by $e^{-\beta_t} \leq 1$
- Weights of incorrect predictions are multiplied by $e^{\beta_t} \geq 1$

AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

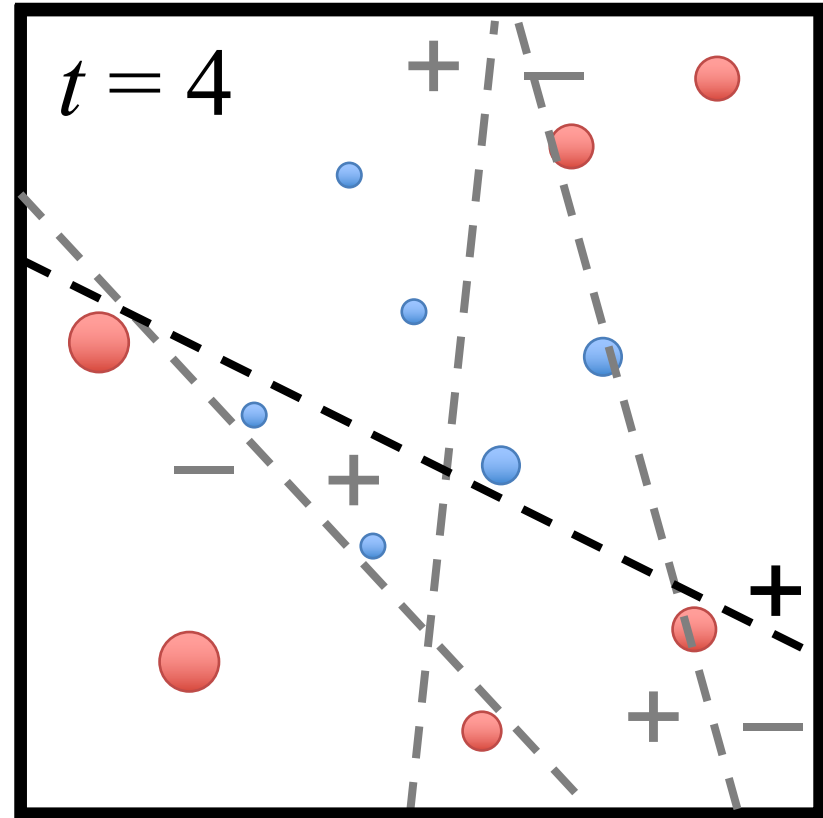
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

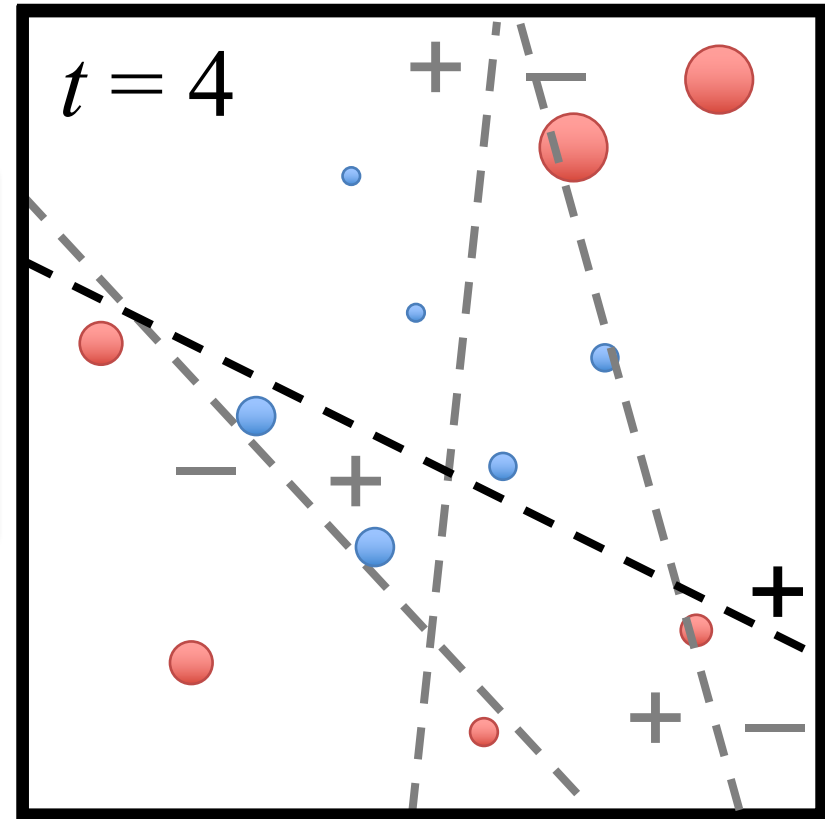
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

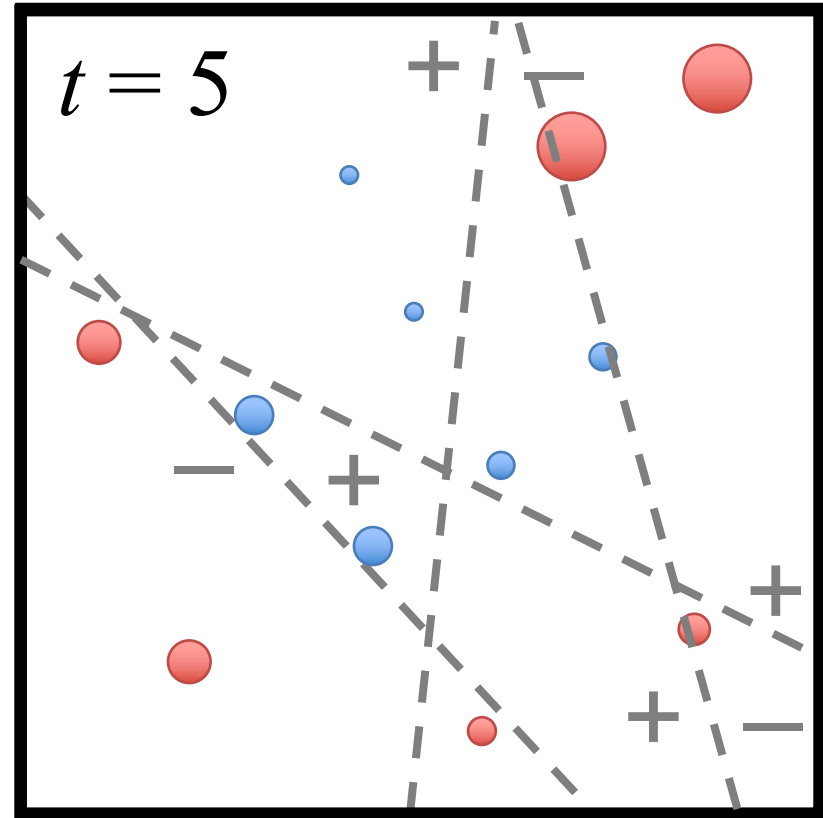
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

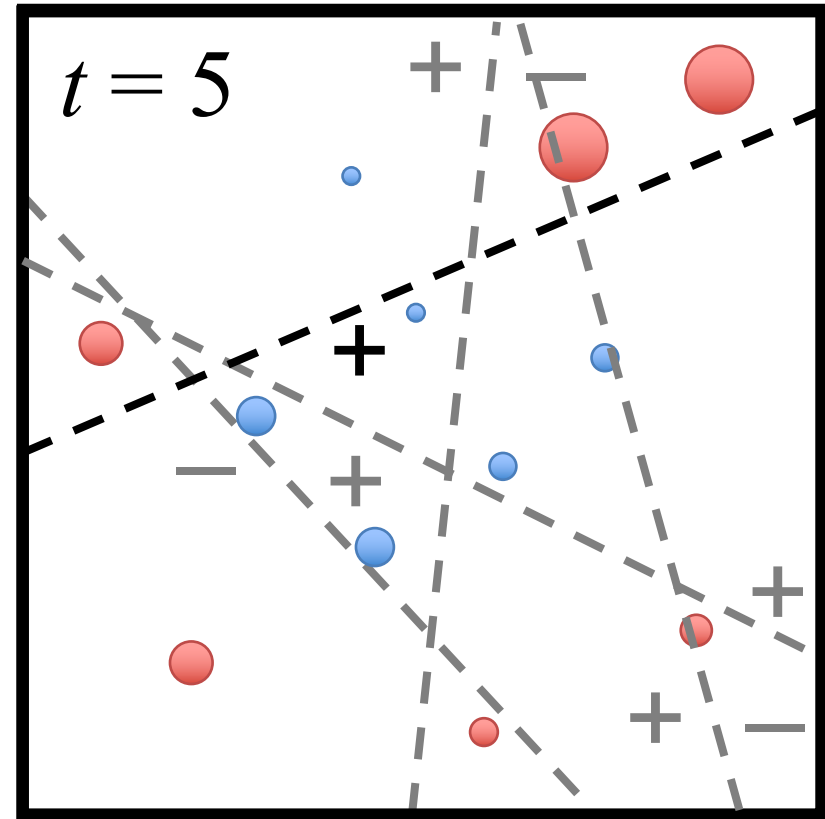
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

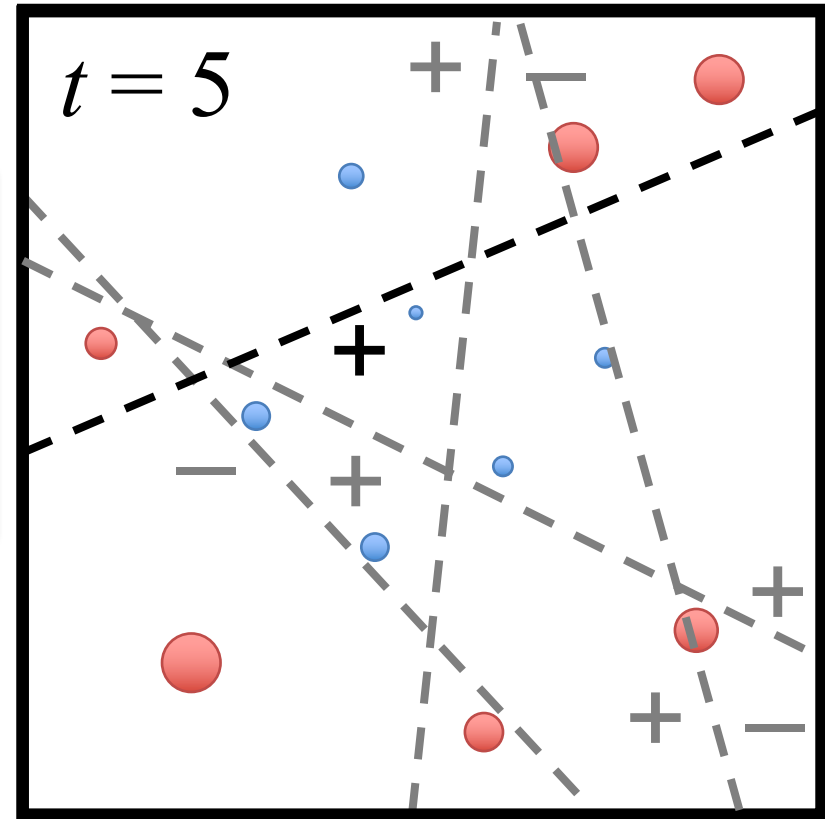
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

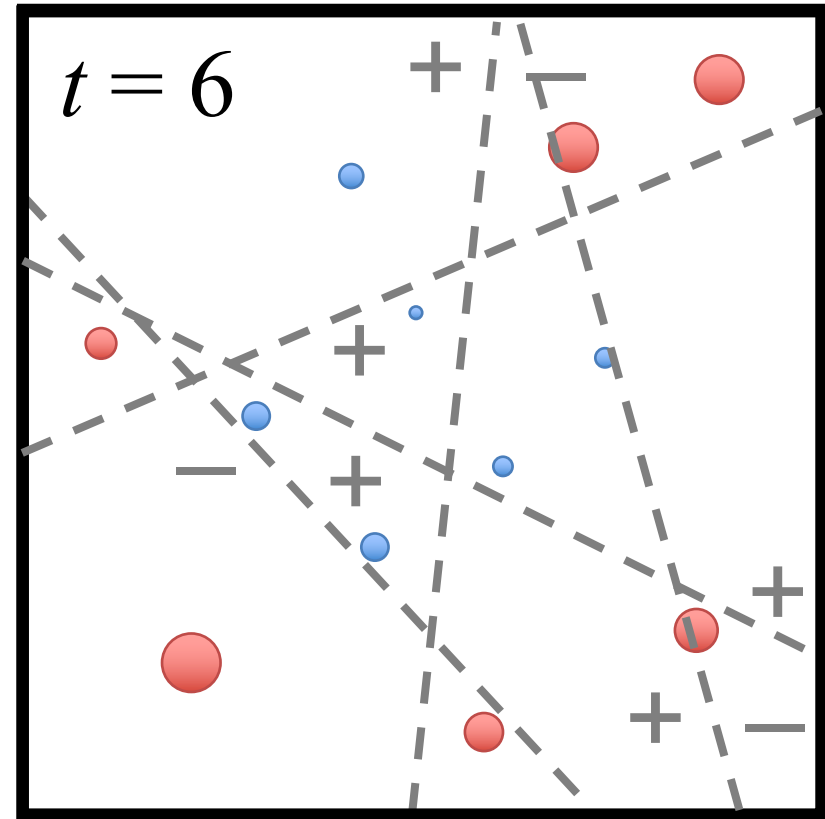
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



AdaBoost

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

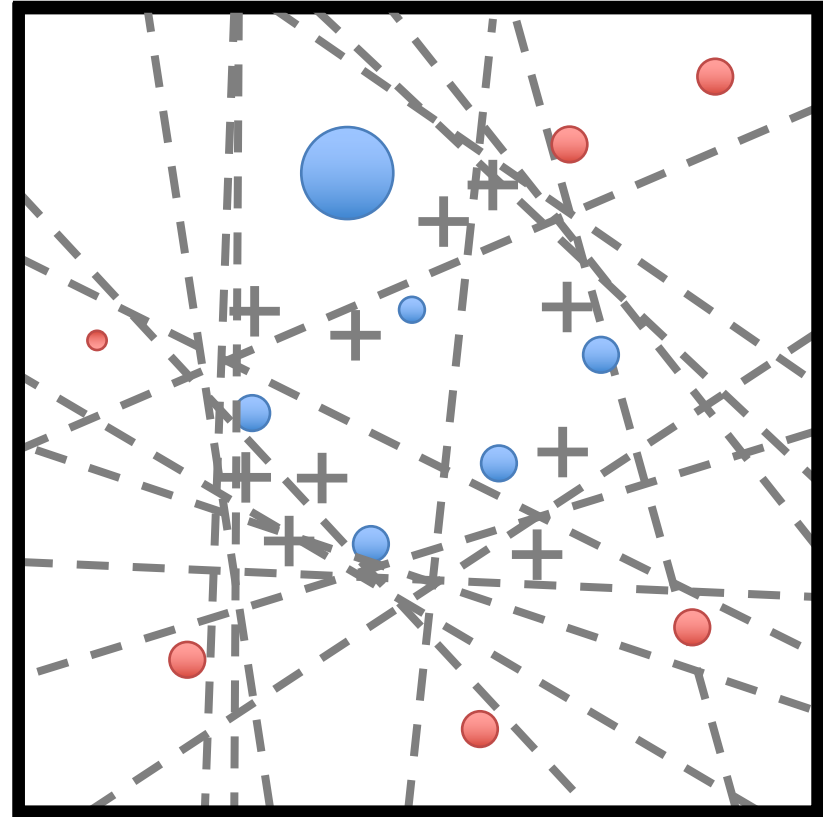


AdaBoost

$t = T$

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

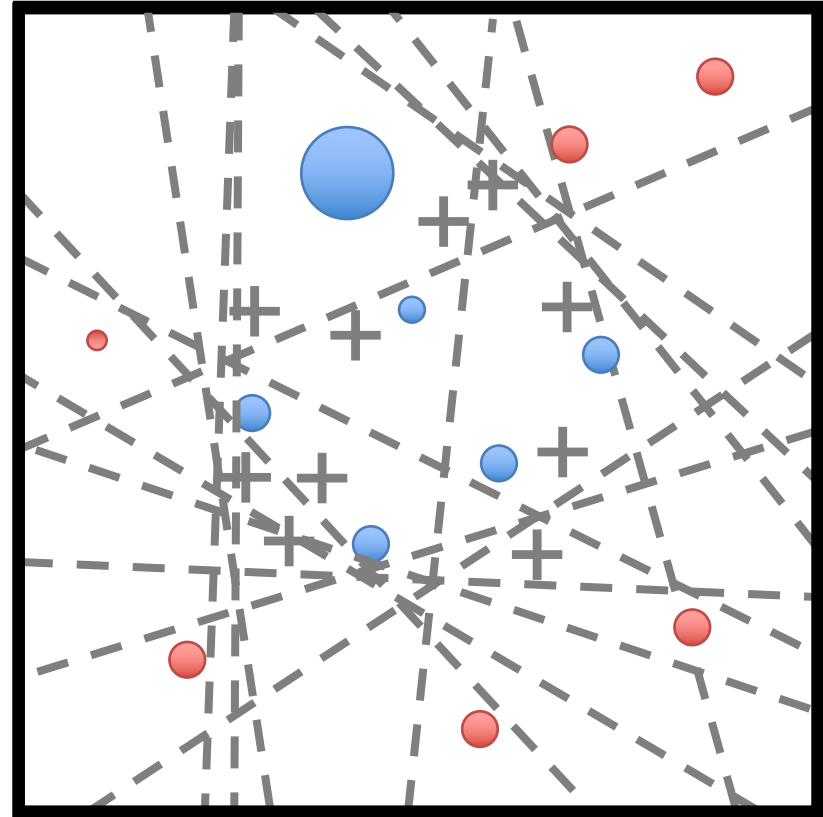


AdaBoost

$t = T$

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



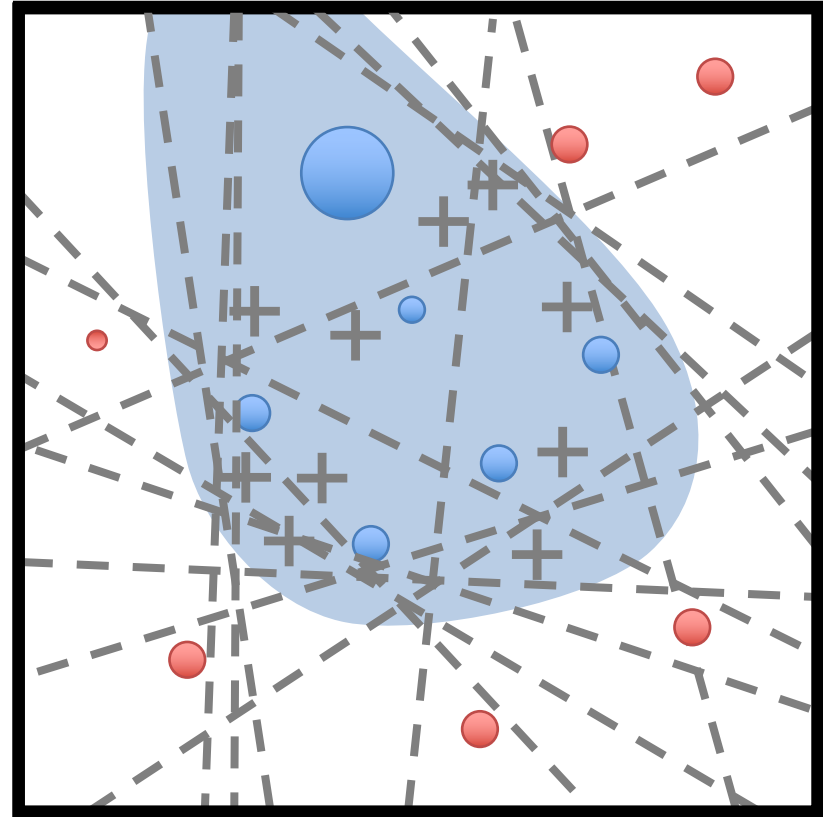
AdaBoost

$t = T$

- 1: Initialize a vector of n uniform weights \mathbf{w}_1
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with weights \mathbf{w}_t
- 4: Compute the weighted training error of h_t
- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:
 $w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$
- 7: Normalize \mathbf{w}_{t+1} to be a distribution
- 8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



- Final model is a weighted combination of members
 - Each member weighted by its importance

AdaBoost

[Freund & Schapire, 1997]

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

- 1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t on X, y with instance weights \mathbf{w}_t
- 4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

- 7: Normalize \mathbf{w}_{t+1} to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

AdaBoost

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

7: Normalize \mathbf{w}_{t+1} to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

\mathbf{w}_t is a vector of weights
over the instances at
iteration t

All points start with equal
weight

AdaBoost

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

7: Normalize \mathbf{w}_{t+1} to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

We need a way to weight instances differently when learning the model...

Training a Model with Weighted Instances

- For algorithms like logistic regression, can simply incorporate weights w into the cost function
 - Essentially, weigh the cost of misclassification differently for each instance

$$J_{\text{reg}}(\boldsymbol{\theta}) = - \sum_{i=1}^n w_i [y_i \log h_{\boldsymbol{\theta}}(\mathbf{x}_i) + (1 - y_i) \log (1 - h_{\boldsymbol{\theta}}(\mathbf{x}_i))] + \lambda \|\boldsymbol{\theta}_{[1:d]}\|_2^2$$

- For algorithms that don't directly support instance weights (e.g., ID3 decision trees, etc.), use weighted bootstrap sampling
 - Form training set by resampling instances with replacement according to w

Base Learner Requirements

- AdaBoost works best with “weak” learners
 - Should not be complex
 - Typically high bias classifiers
 - Works even when weak learner has an error rate just slightly under 0.5 (i.e., just slightly better than random)
 - Can prove training error goes to 0 in $O(\log n)$ iterations
- Examples:
 - Decision stumps (1 level decision trees)
 - Depth-limited decision trees
 - Linear classifiers

AdaBoost

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

Error is the sum the weights of all misclassified instances

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

7: Normalize \mathbf{w}_{t+1} to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

AdaBoost

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

6: Update all instance weights

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i))$$

7: Normalize \mathbf{w}_{t+1} to be a distribution

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i$$

8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

- β_t measures the importance of h_t
- If $\epsilon_t \leq 0.5$, then $\beta_t \geq 0$
 - Trivial, otherwise flip h_t 's predictions
- β_t grows as error h_t 's shrinks

AdaBoost

INPUT: training set (\mathbf{x}_i, y_i)
the number of rounds T

- 1: Initialize a vector \mathbf{w}_1 of instance weights
- 2: **for** $t = 1, \dots, T$
- 3: Train model h_t
- 4: Compute the error ϵ_t

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

- 5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$
- 6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

- 7: Normalize \mathbf{w}_{t+1} to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

- 8: **end for**
- 9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

This is the same as:

$$w_{t+1,i} = w_{t,i} \times \begin{cases} e^{-\beta_t} & \text{if } h_t(\mathbf{x}_i) = y_i \\ e^{\beta_t} & \text{if } h_t(\mathbf{x}_i) \neq y_i \end{cases}$$

will be ≤ 1

will be ≥ 1

Essentially this emphasizes misclassified instances.

AdaBoost

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

7: Normalize \mathbf{w}_{t+1} to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

Make w_{t+1} sum to 1

AdaBoost

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

2: **for** $t = 1, \dots, T$

3: Train model h_t on X, y with instance weights \mathbf{w}_t

4: Compute the weighted training error rate of h_t :

$$\epsilon_t = \sum_{i: y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$

7: Normalize \mathbf{w}_{t+1} to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$

8: **end for**

9: **Return** the hypothesis

$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$

Member classifiers with less error are given more weight in the final ensemble hypothesis

Final prediction is a weighted combination of each member's prediction

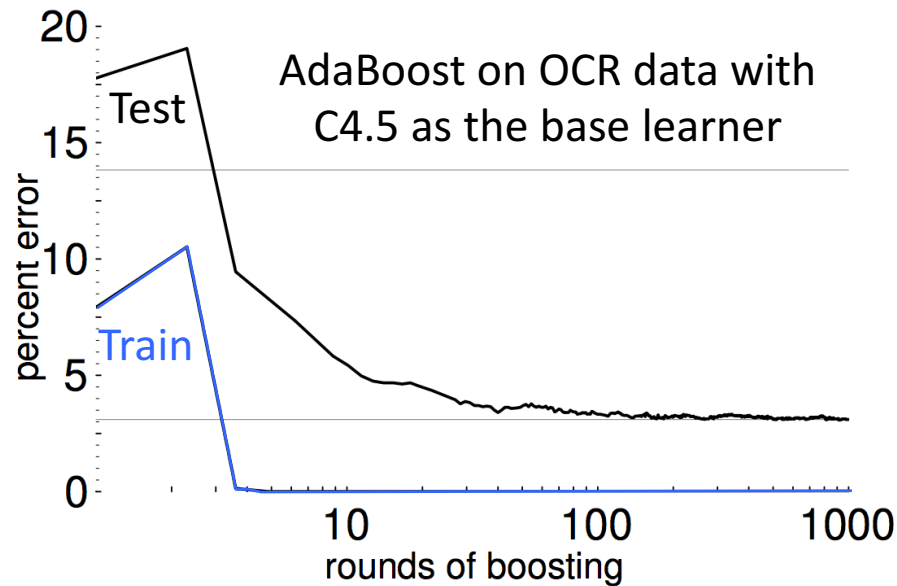
Dynamic Behavior of AdaBoost

- If a point is repeatedly misclassified...
 - Each time, its weight is increased
 - Eventually it will be emphasized enough to generate a hypothesis that correctly predicts it
- Successive member hypotheses focus on the hardest parts of the instance space
 - Instances with highest weight are often outliers

AdaBoost and Overfitting

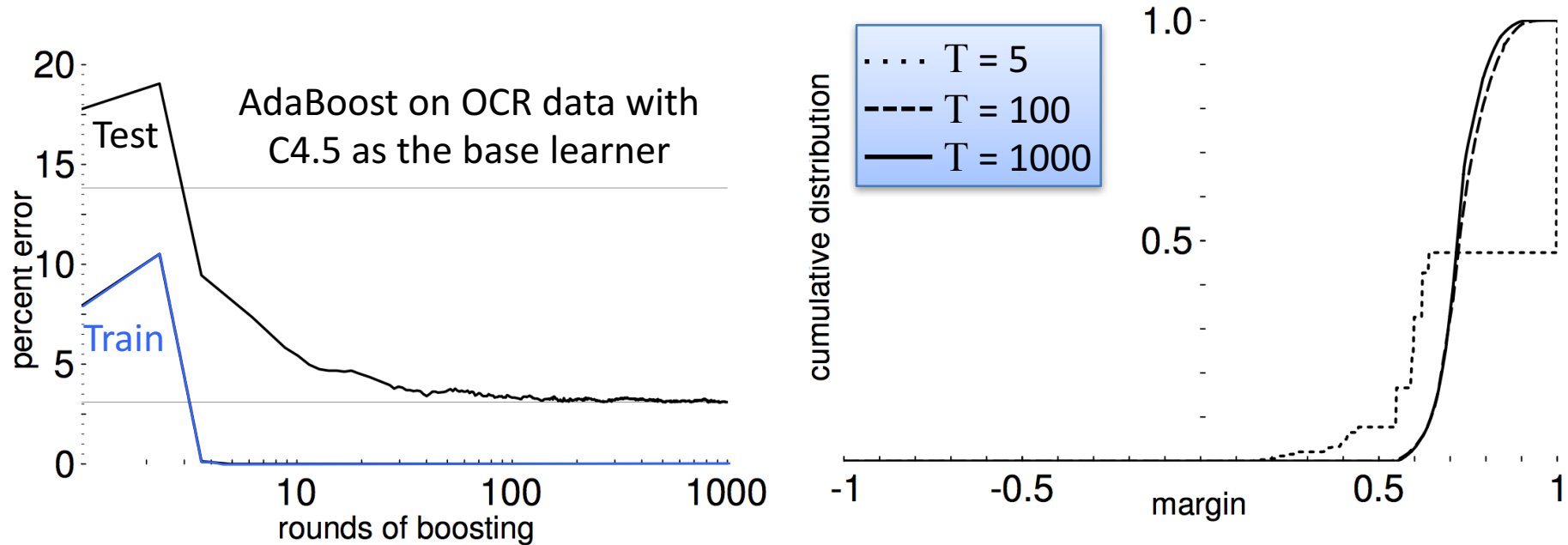
- VC Theory originally predicted that AdaBoost would always overfit as T grew large
 - Hypothesis keeps growing more complex
- In practice, AdaBoost often did not overfit, contradicting VC theory
- Also, AdaBoost does not explicitly regularize the model

Explaining Why AdaBoost Works



- Empirically, boosting resists overfitting
- Note that it continues to drive down the test error even AFTER the training error reaches zero

Explaining Why AdaBoost Works



- The “margins explanation” shows that boosting tries to increase the confidence in its predictions over time
 - Improves generalization performance
 - Effectively, boosting maximizes the margin!

AdaBoost in Practice

Strengths:

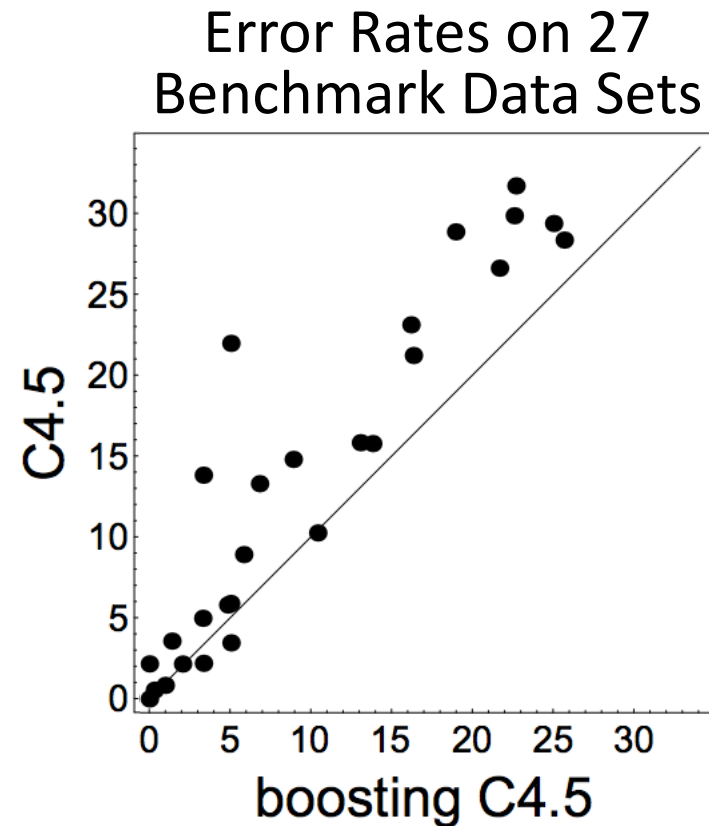
- Fast and simple to program
- No parameters to tune (besides T)
- No assumptions on weak learner

When boosting can fail:

- Given insufficient data
- Overly complex weak hypotheses
- Can be susceptible to noise
- When there are a large number of outliers

Boosted Decision Trees

- Boosted decision trees are one of the best “off-the-shelf” classifiers
 - i.e., no parameter tuning
- Limit member hypothesis complexity by limiting tree depth
- Gradient boosting methods are typically used with trees in practice



“AdaBoost with trees is the best off-the-shelf classifier in the world” -Breiman, 1996
(Also, see results by Caruana & Niculescu-Mizil, ICML 2006)