



CIS 419/519

Ensemble Methods

Ensemble Learning

Consider a set of classifiers h_1, \dots, h_L

Key Idea: construct a classifier $H(x)$ that combines the individual predictions of h_1, \dots, h_L

e.g., the member classifiers could vote,
with all votes counting equally



Works well if each of the ensemble members has a low error rate

Ensemble Learning

Consider a set of classifiers h_1, \dots, h_L

Key Idea: construct a classifier $H(x)$ that combines the individual predictions of h_1, \dots, h_L

e.g., the member classifiers could vote,
with all votes counting equally



e.g., could use different members for
different regions of the instance space



[AP Photo/Julie Marks]

Ensemble Learning

Successful ensembles require **diversity**

- Classifiers should make different (uncorrelated) mistakes
- Ensemble members can be different types of base learners
 - e.g., an ensemble of a decision tree, logistic regression, and an SVM-RBF

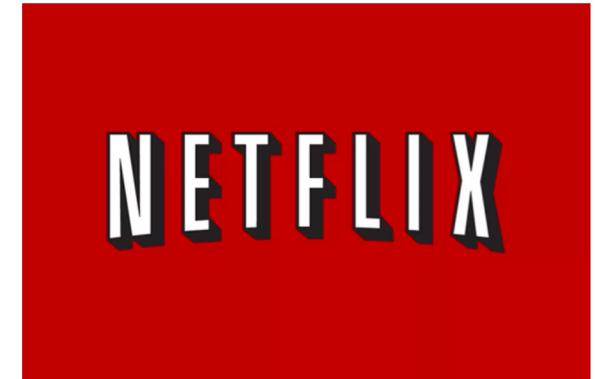


Image: Freepik.com

Application: Collaborative Filtering in Netflix

Goal: predict how a user will rate a movie, based on:

- the user's ratings for other movies, and
- other users' ratings for the same and other movies
- Given no other information about the movies

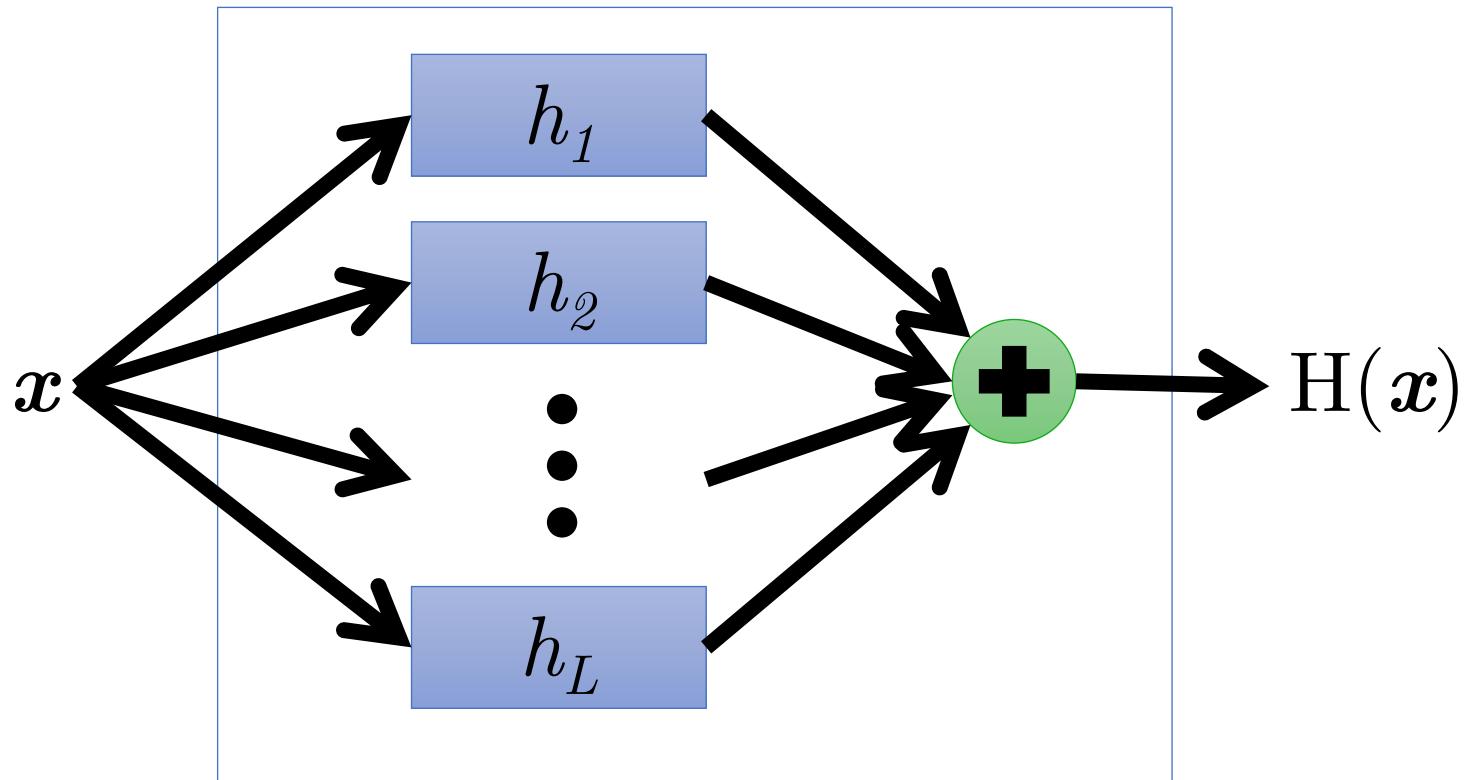


Netflix Prize: \$1M to the first team to do 10% better than Netflix' system
(2007-2009)

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

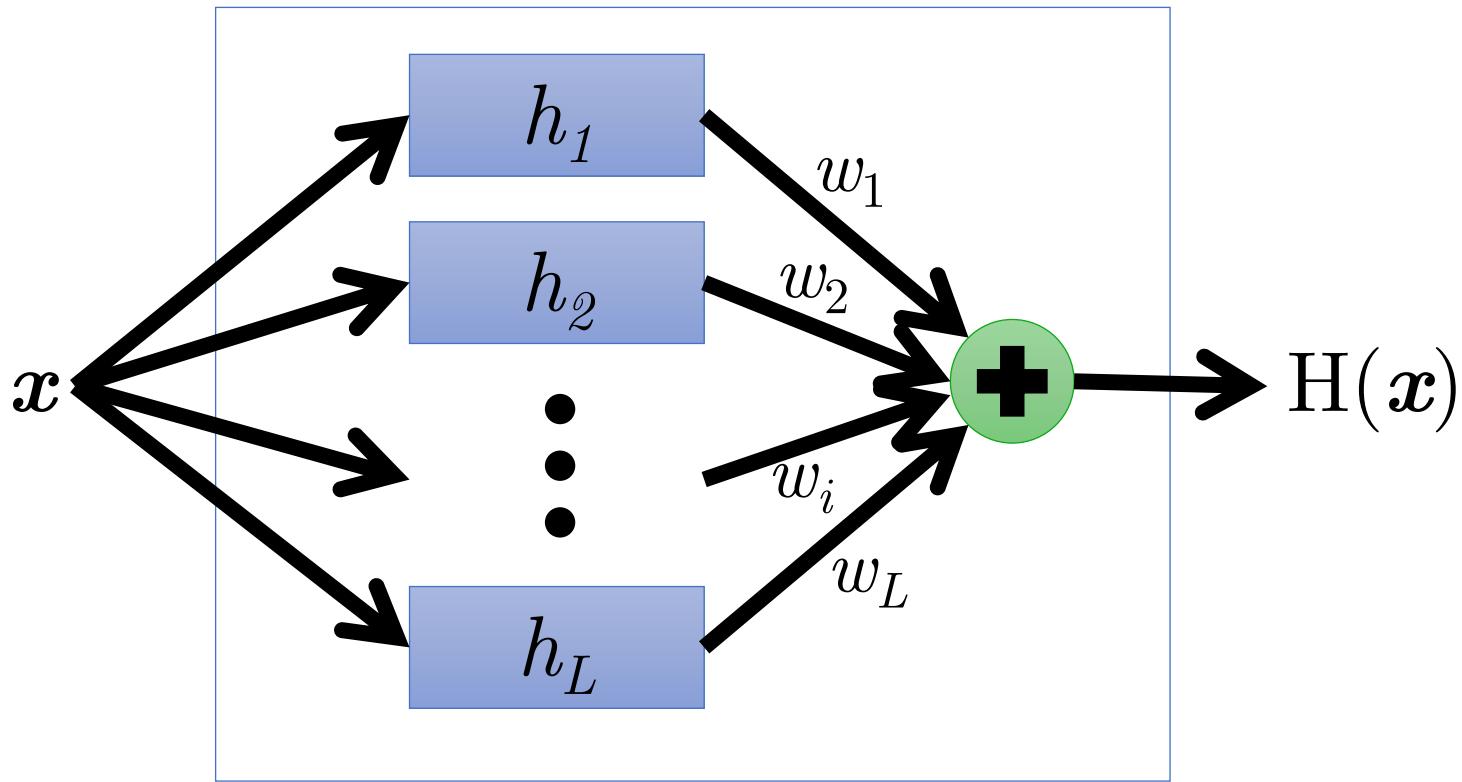
Combining Classifiers

Combining Classifiers: Averaging



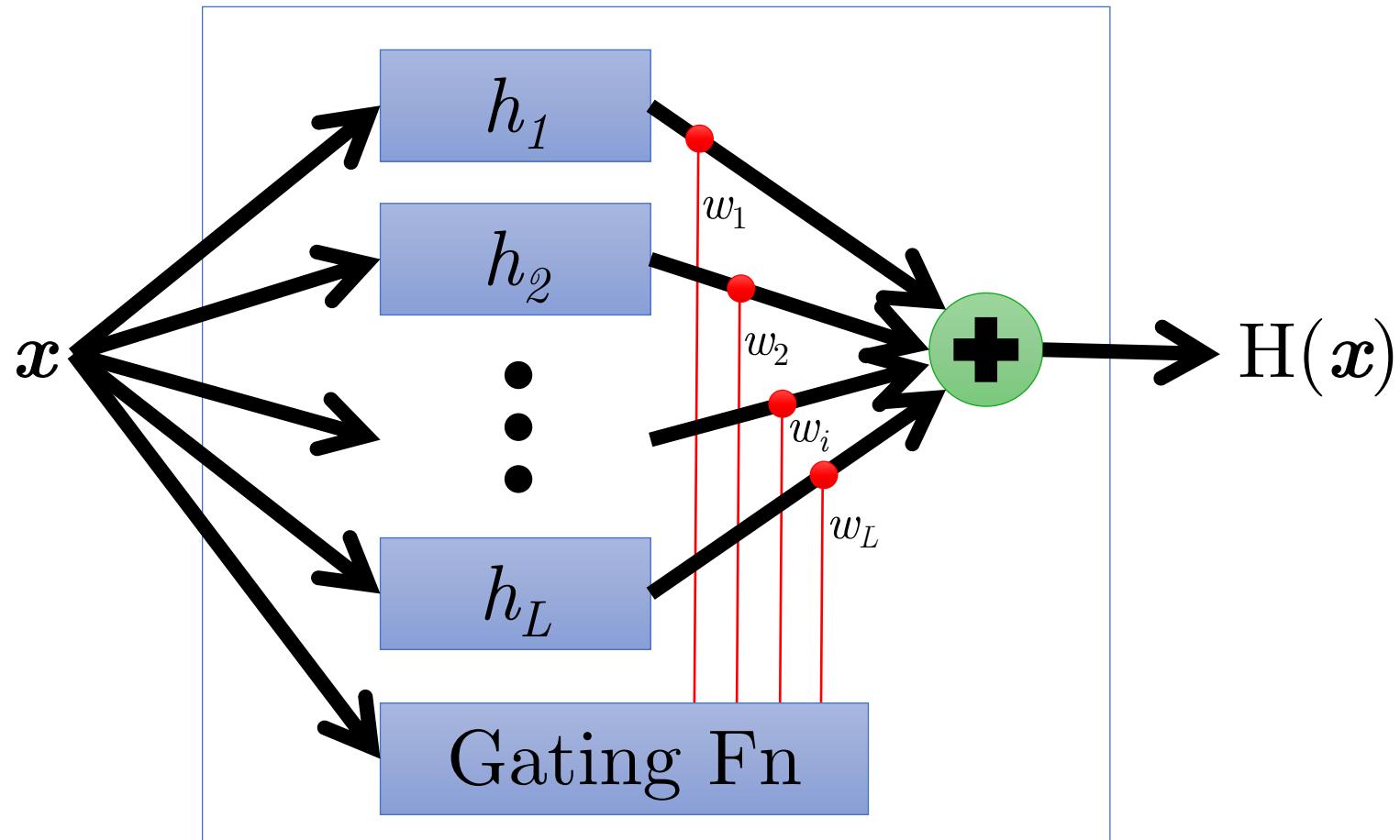
- Final hypothesis is a simple vote of the members

Combining Classifiers: Weighted Average



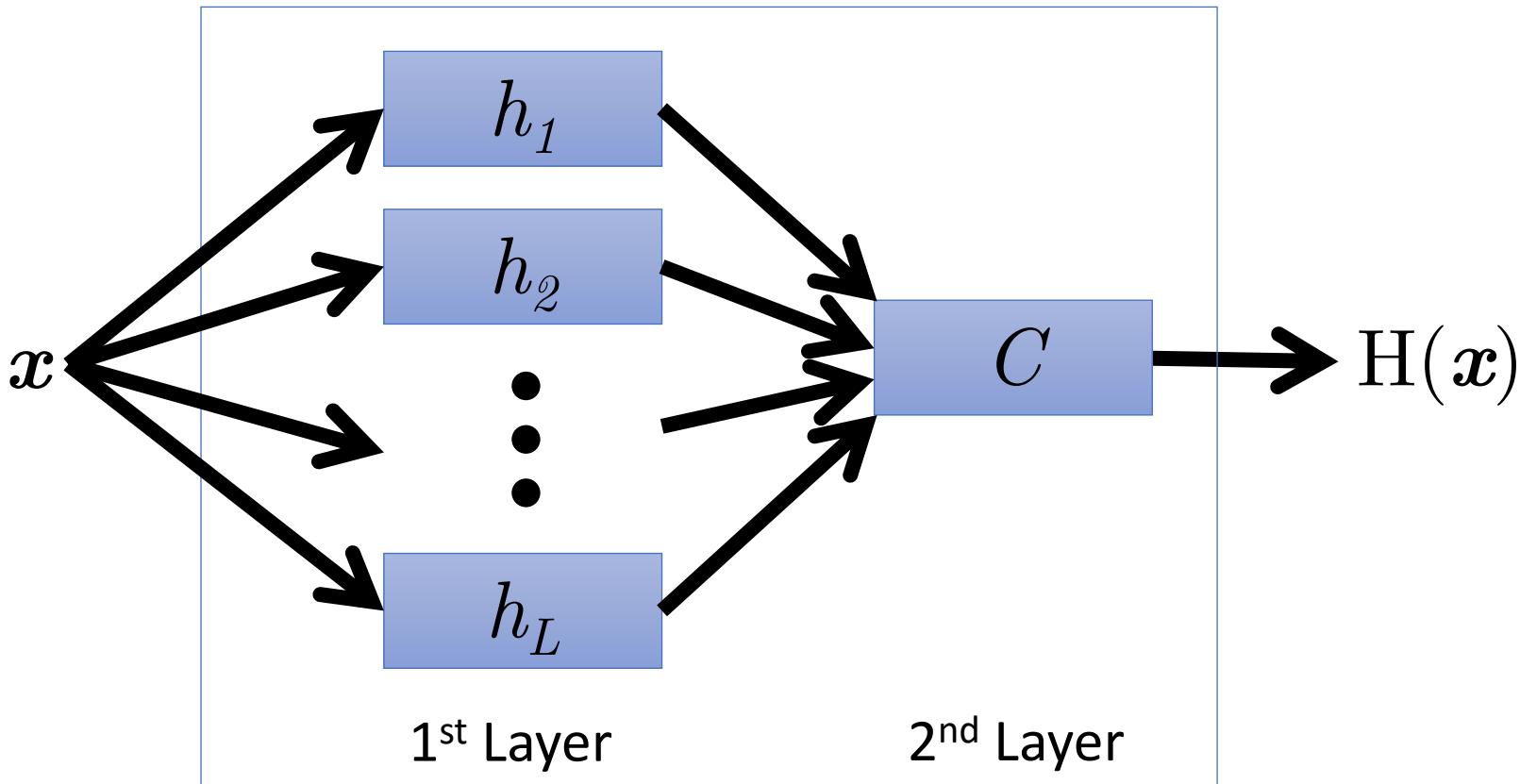
- Coefficients of individual members are trained using a validation set

Combining Classifiers: Gating



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

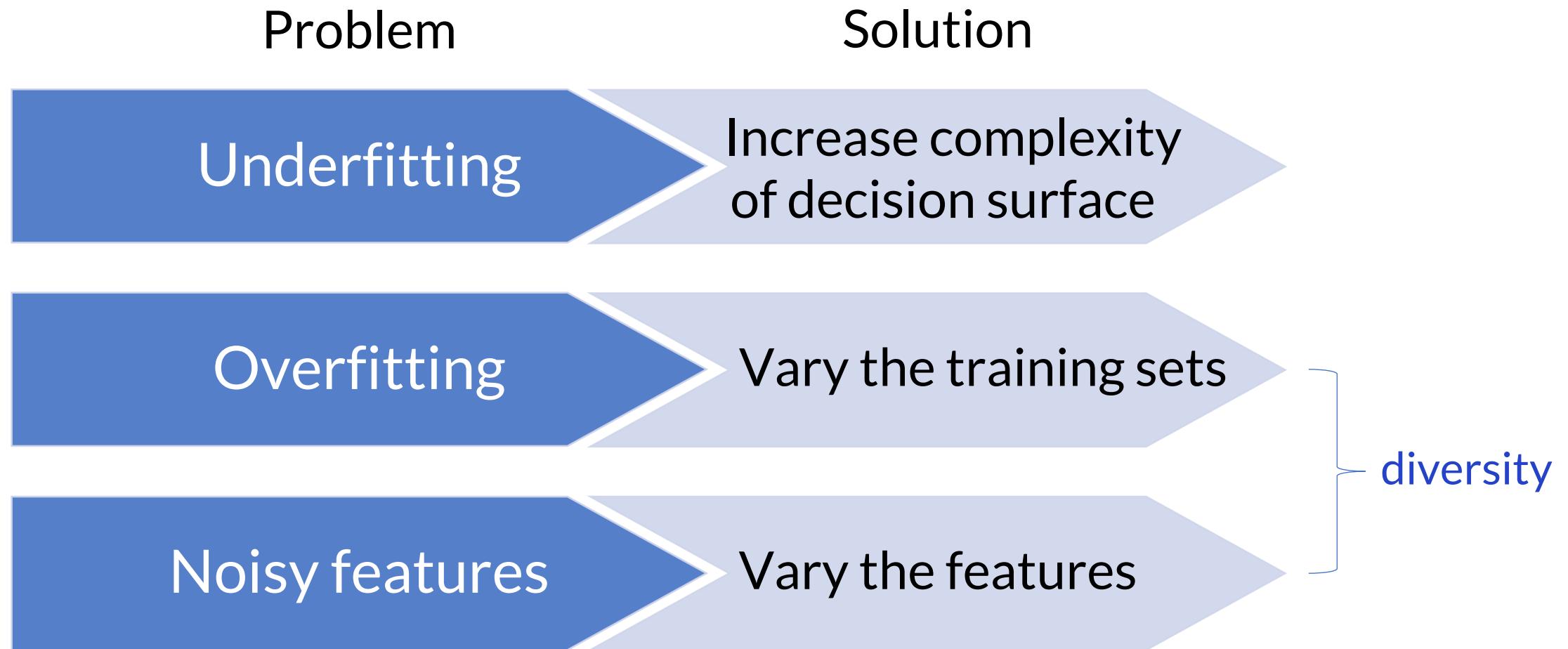
Combining Classifiers: Stacking



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

Inducing Diversity: Bootstrapping and Bagging

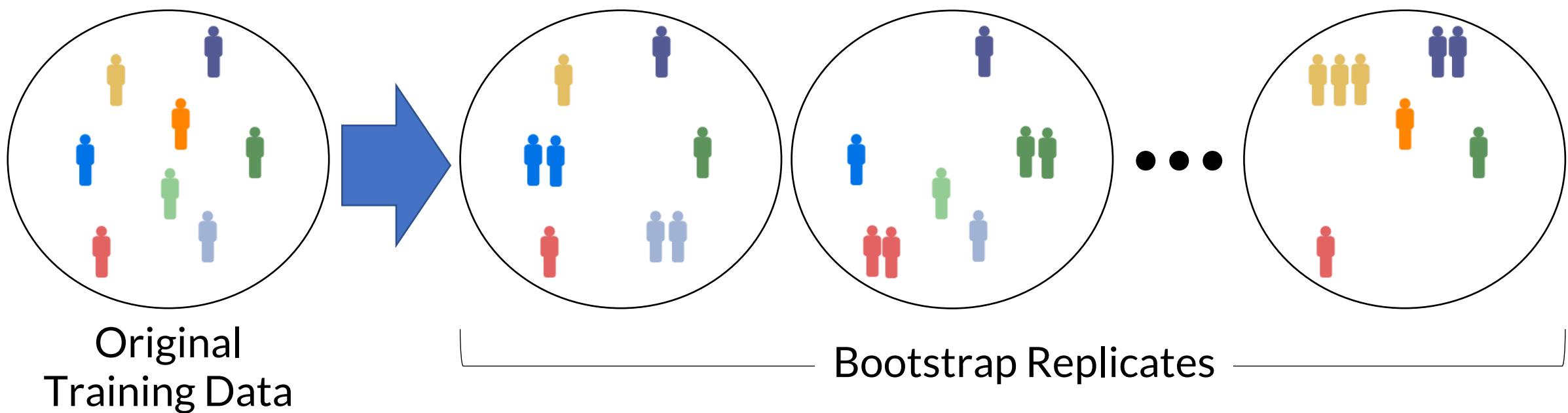
Compensating for Problems via Diversity



Varying the Training Data

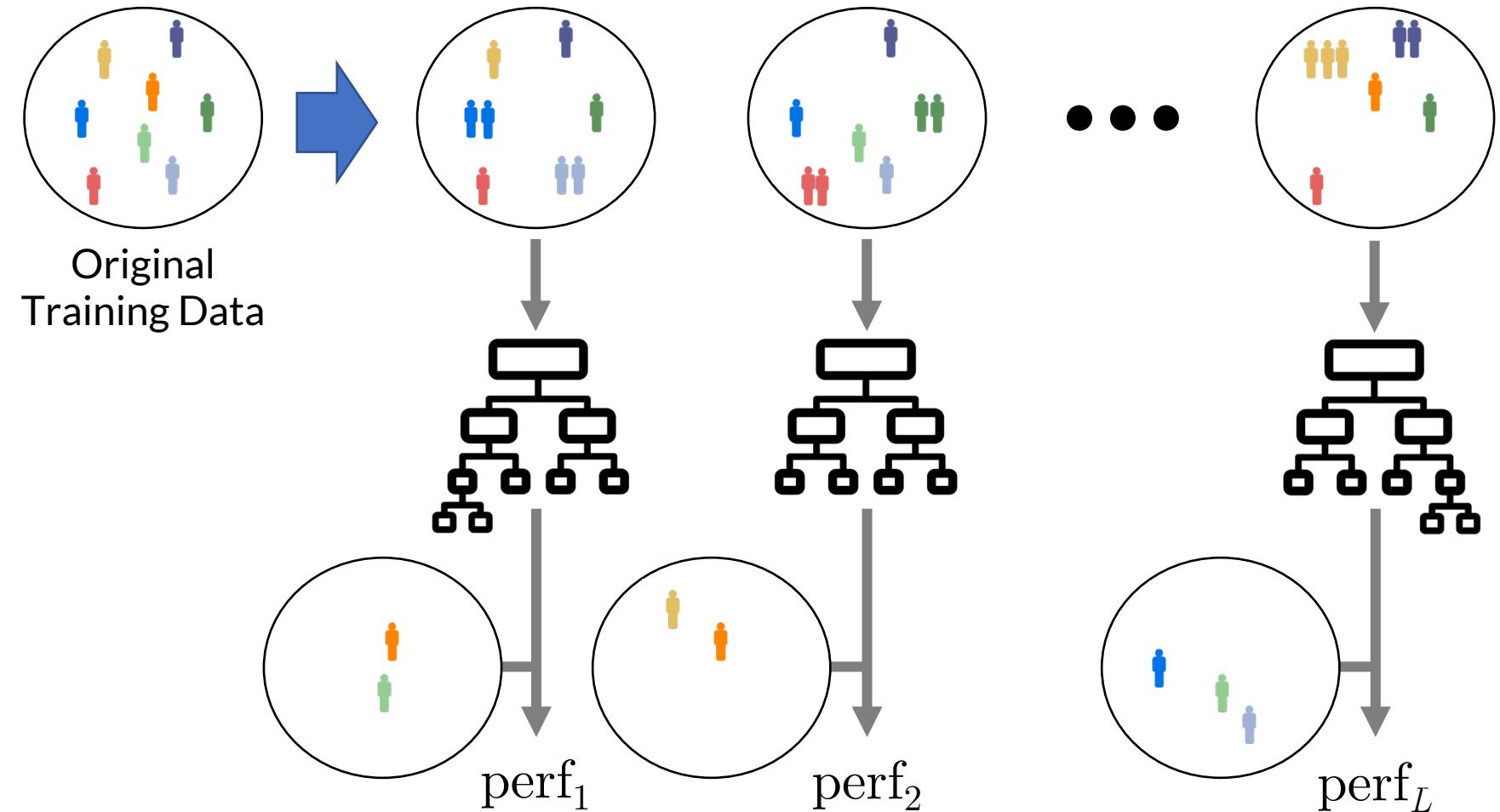
Bootstrap replication:

- Given n training instances, construct new training sets by sampling n instances with replacement
 - Excludes ~30% of the training instances in each of the replicates



Manipulating the Training Data: Bagging

1.) Create bootstrap replicates of training set



2.) Train a classifier for each replicate

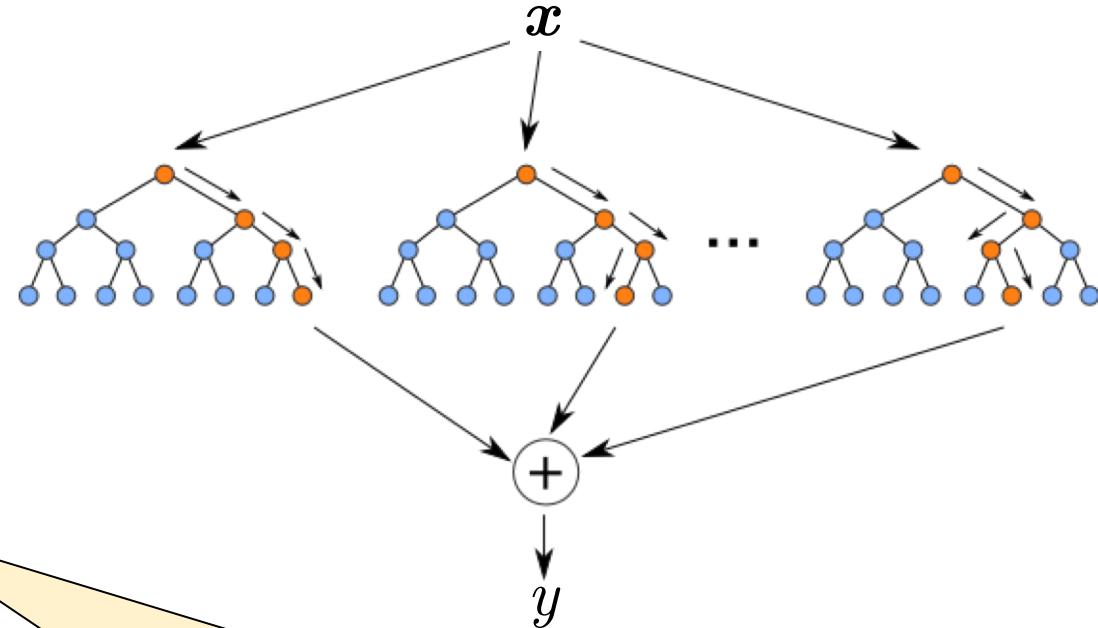
3.) Estimate classifier performance using out-of-bootstrap data

4.) Create a weighted ensemble of the classifiers, with weights based on $perf_1, \dots, perf_L$

Inducing Diversity: The Random Forest

Random Forest Classifier

- Manipulates both training data and the features to induce diversity
 - **Training data manipulation:** bagging to create an ensemble of decision trees
 - **Feature manipulation:** Each decision tree node focuses on a subset of features, chosen randomly at each node



Issue with only bootstrapping DTs

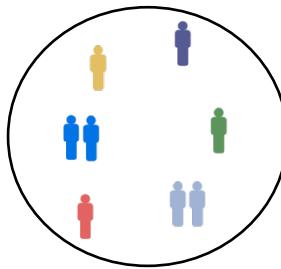
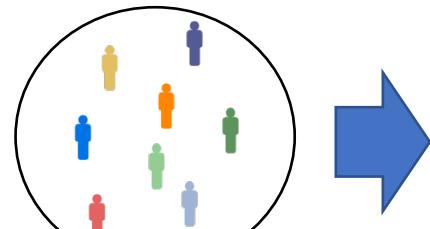
- If a few features are highly predictive, then they will be selected in many trees
- This will cause the ensemble members to become correlated

DT node creation procedure for a RF:

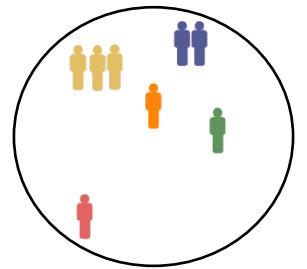
- 1.) At each node, choose \sqrt{d} features randomly to consider
- 2.) Determine split among these features

Manipulating the Features: Random Forests

- 1.) Create bootstrap replicates of training set

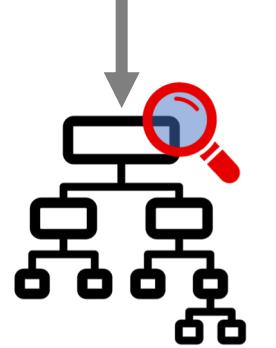
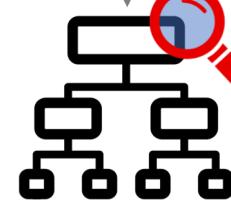
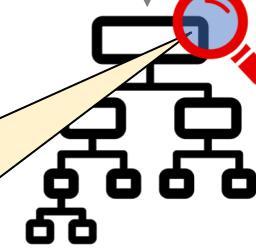


• • •



- 2.) Train an **unpruned decision tree** for each replicate

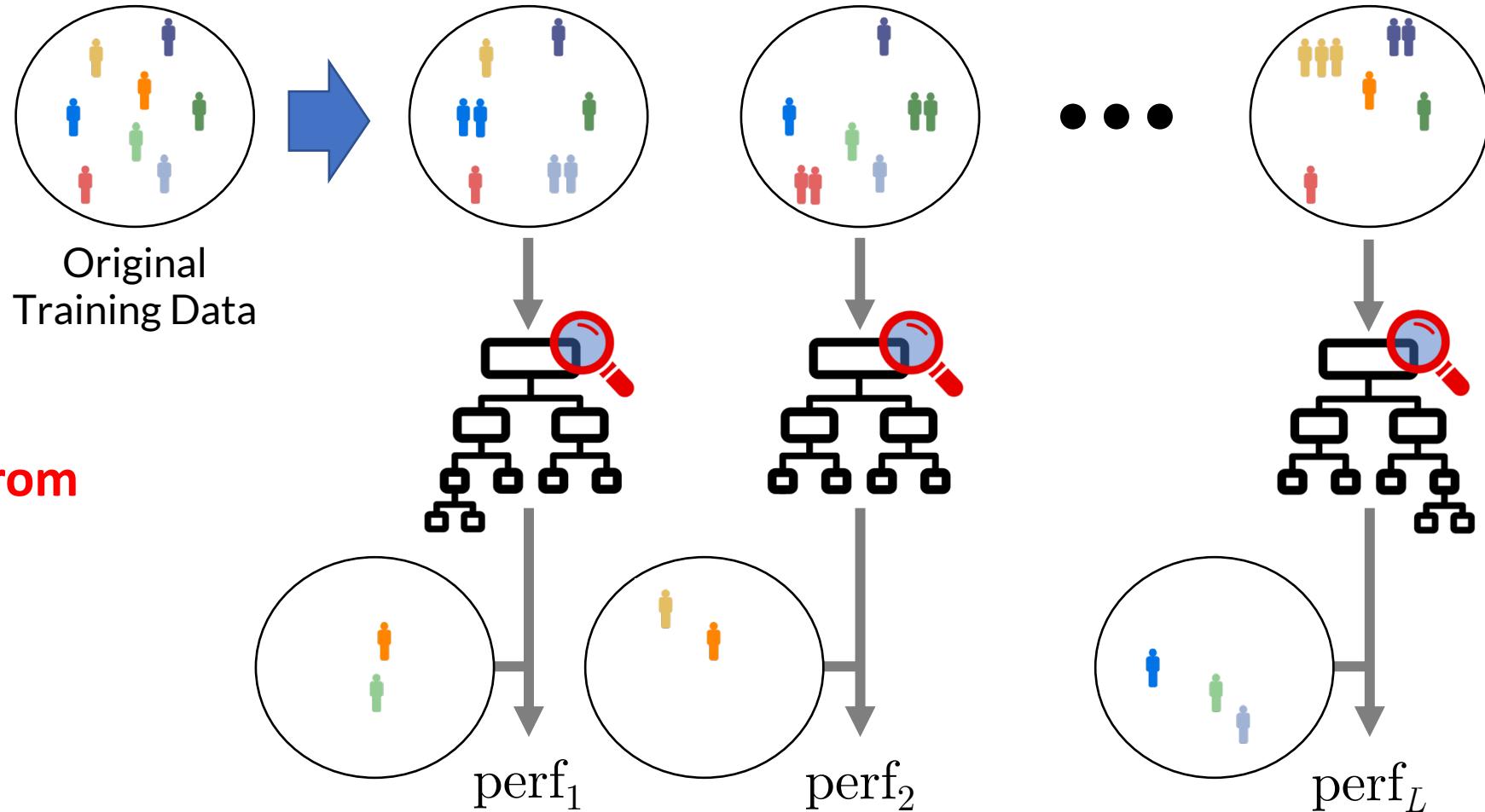
Original
Training Data



Restrict each node's decisions to only a **small subset of features**, chosen randomly for each node

Manipulating the Features: Random Forests

- 1.) Create bootstrap replicates of training set



- 4.) Create a weighted ensemble of the classifiers, with weights based on $perf_1$, ... $perf_L$

Inducing Diversity: Boosting

AdaBoost

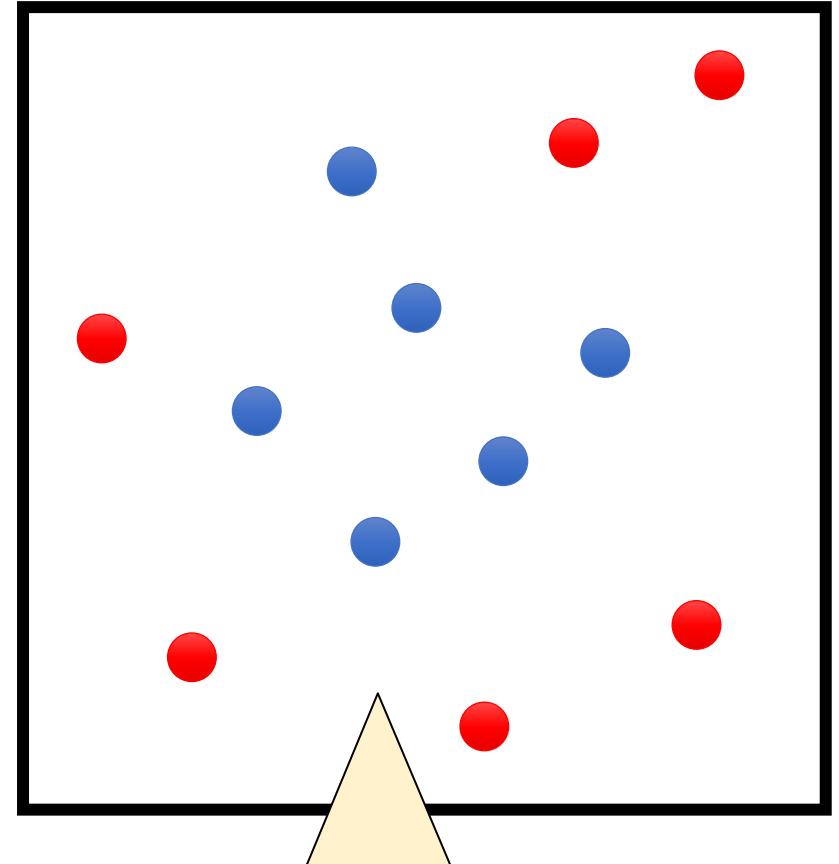
[Freund & Schapire, 1997]

- 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 misspredicted 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 instance weights \mathbf{w}_t
- 4: Compute the weighted training error ϵ_t 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

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



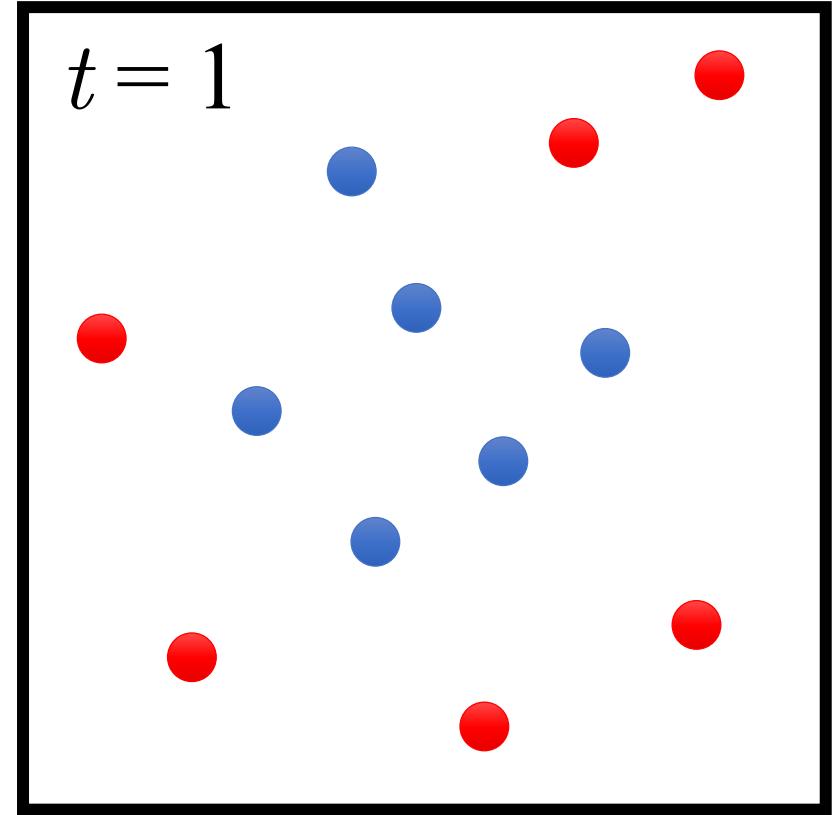
Size of point represents
the instance's weight

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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

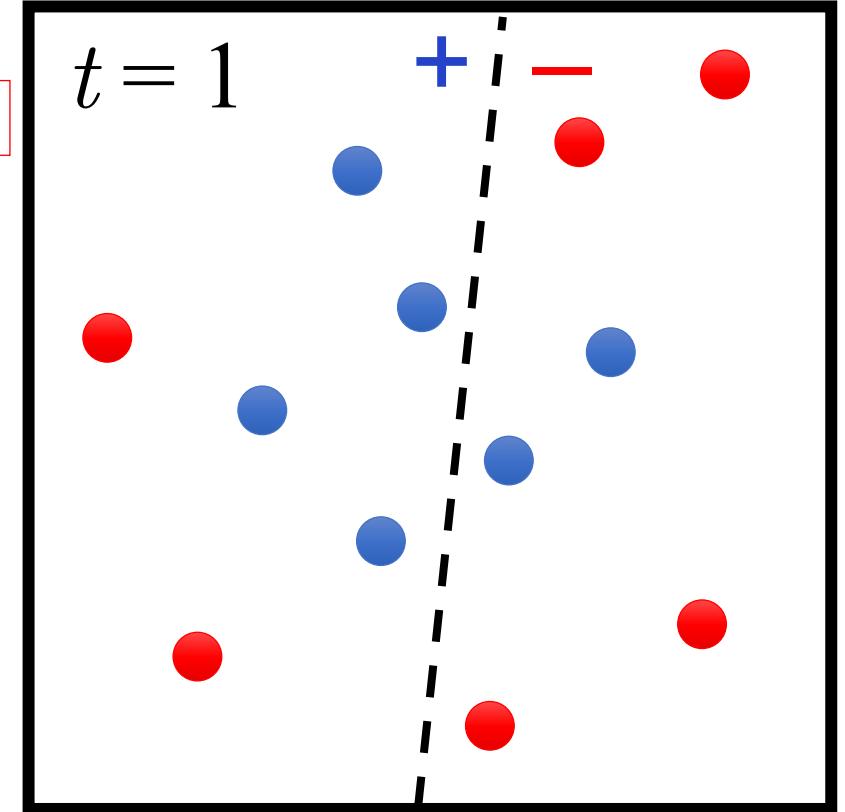


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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



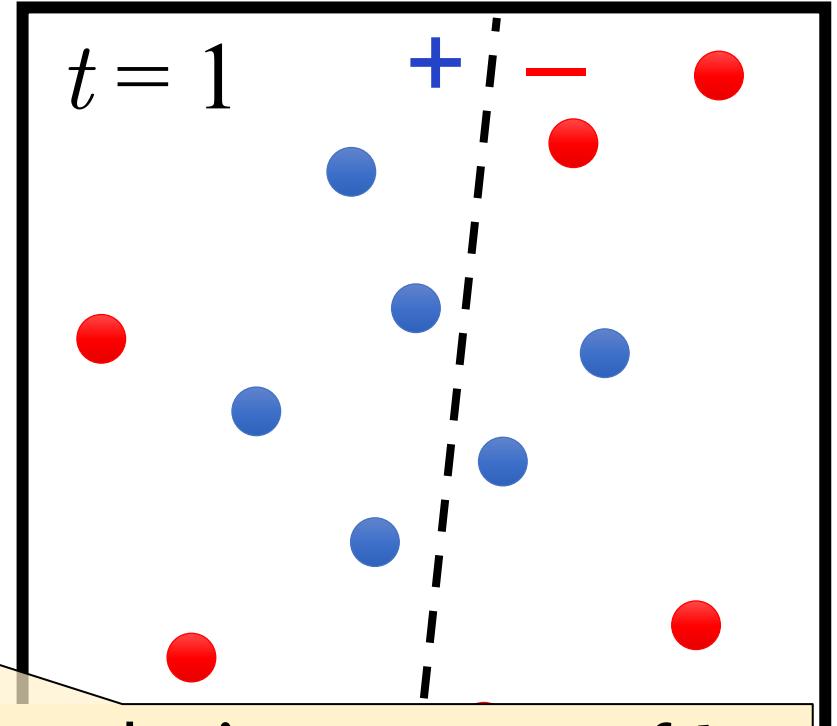
AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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

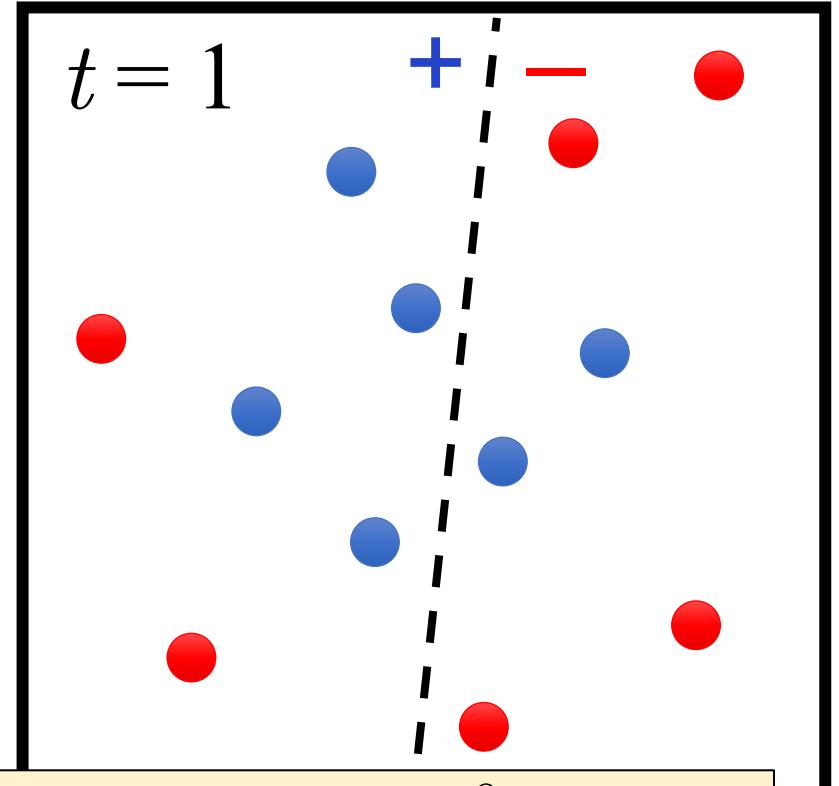
```
1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
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
```

$H(x)$

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



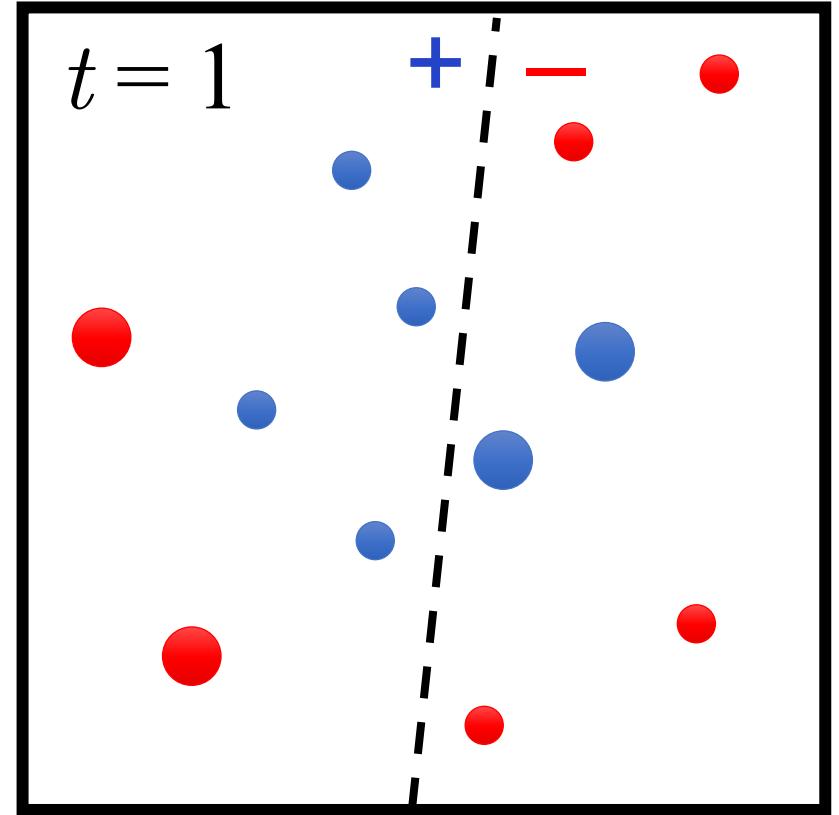
AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

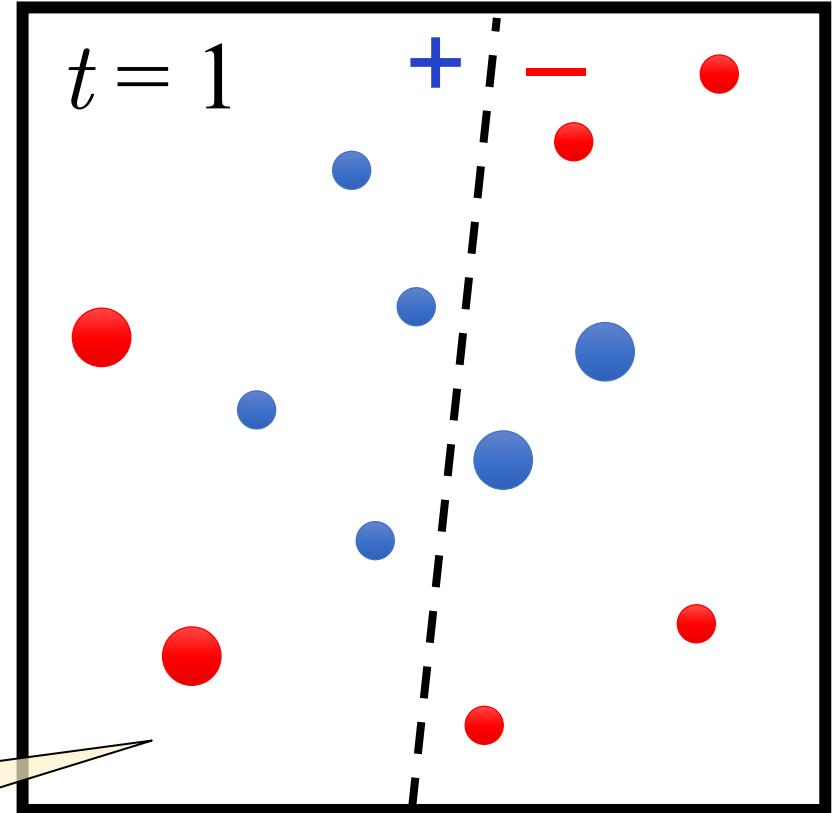
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```



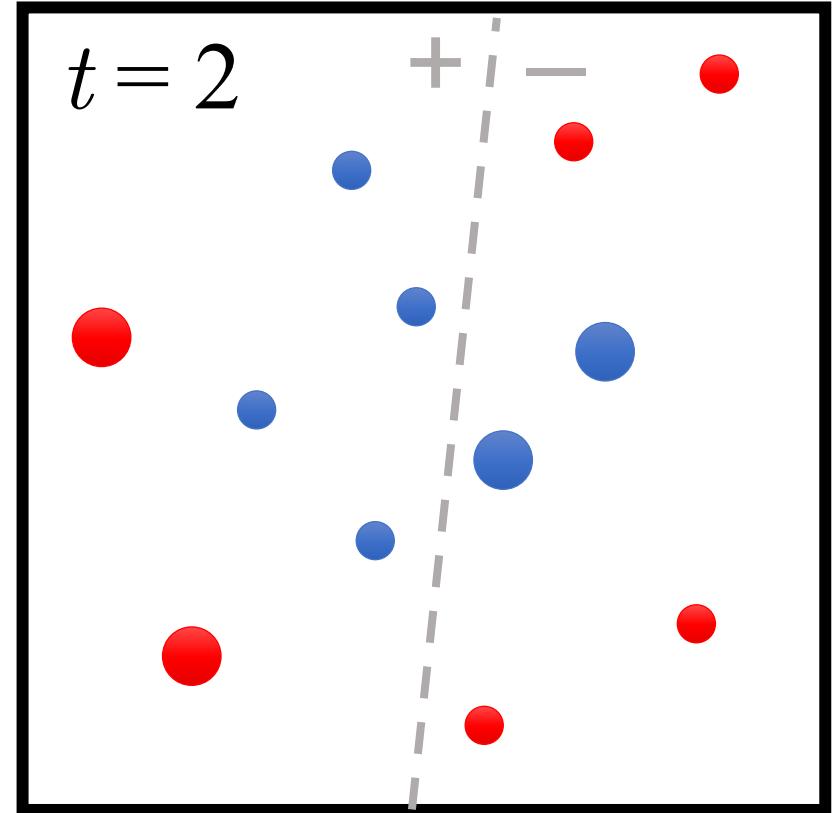
Disclaimer: Note that resized points in the illustration above are not necessarily to scale with β_t

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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

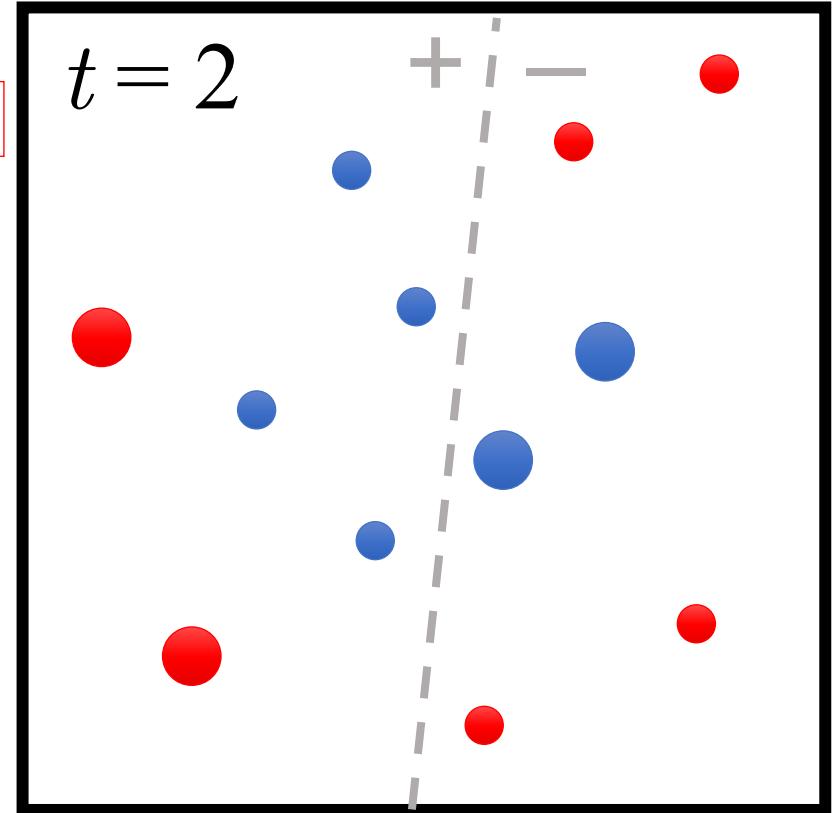


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

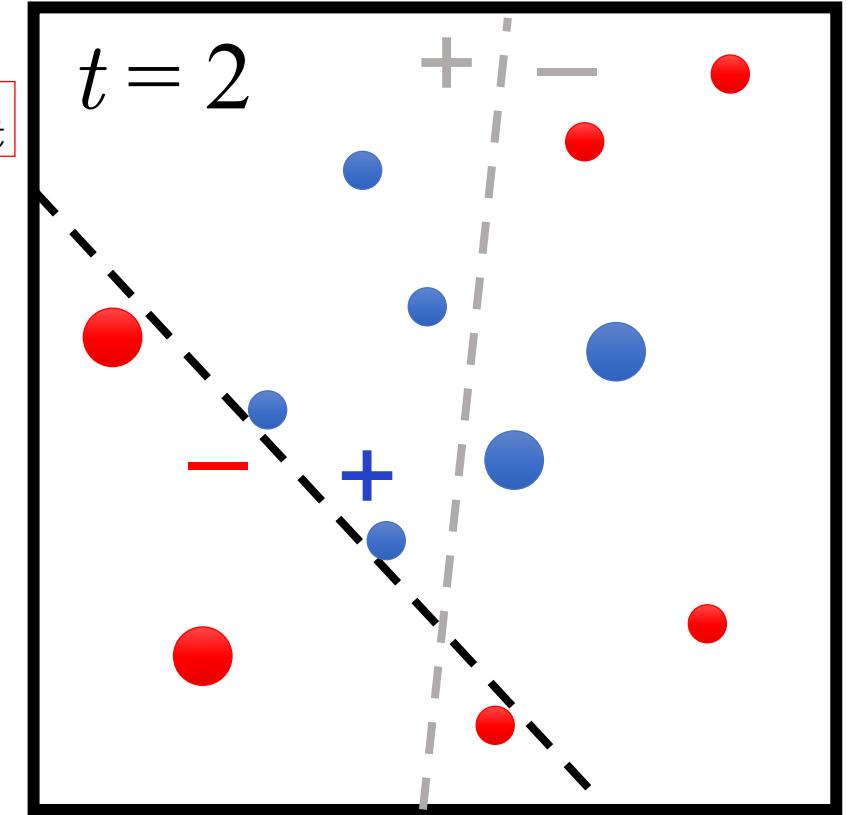


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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



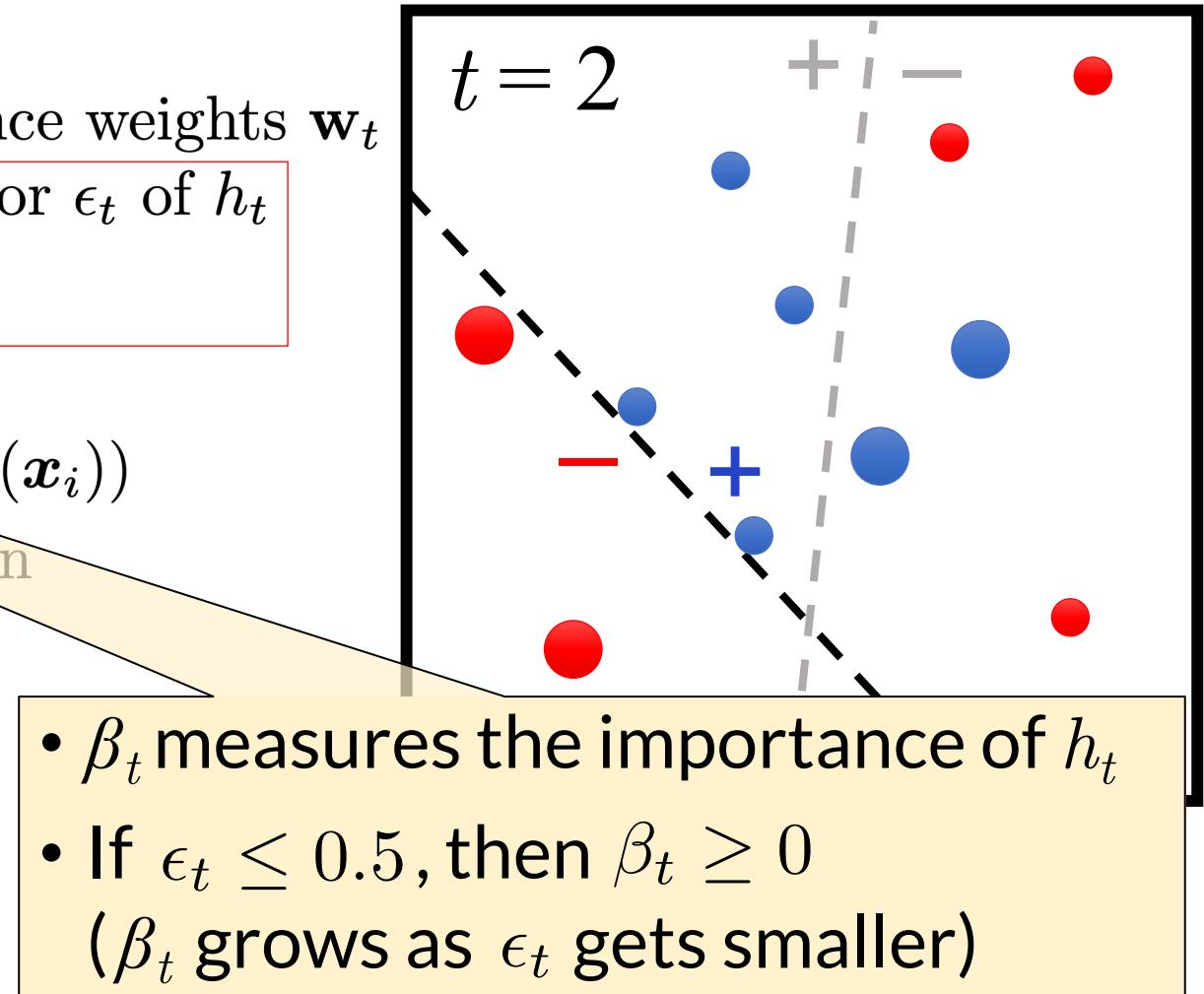
AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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

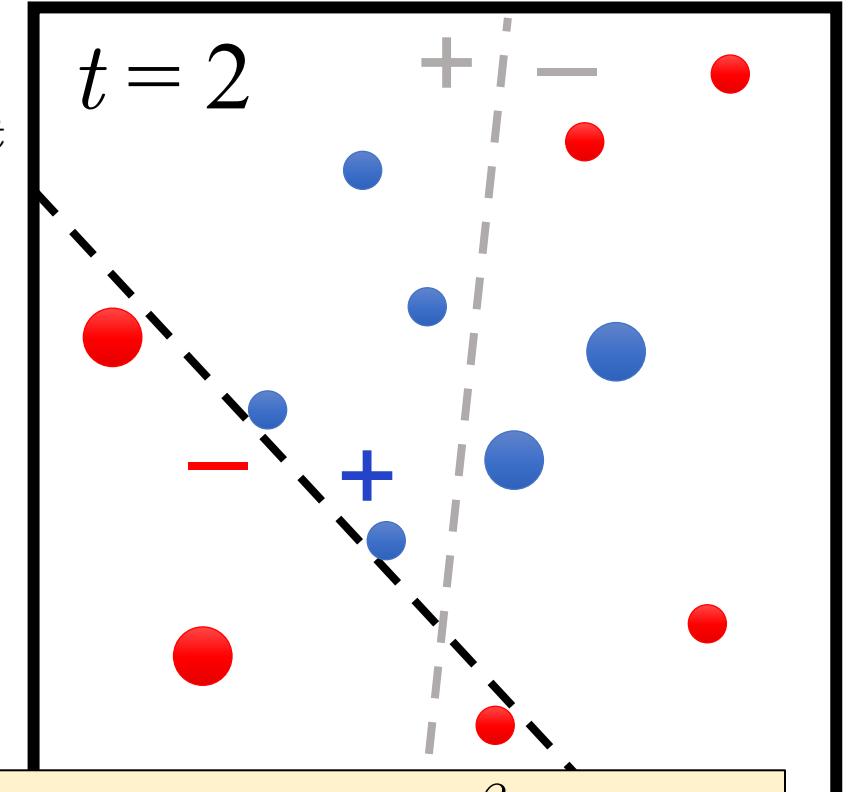
```
1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
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
```

$H(x)$

- 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

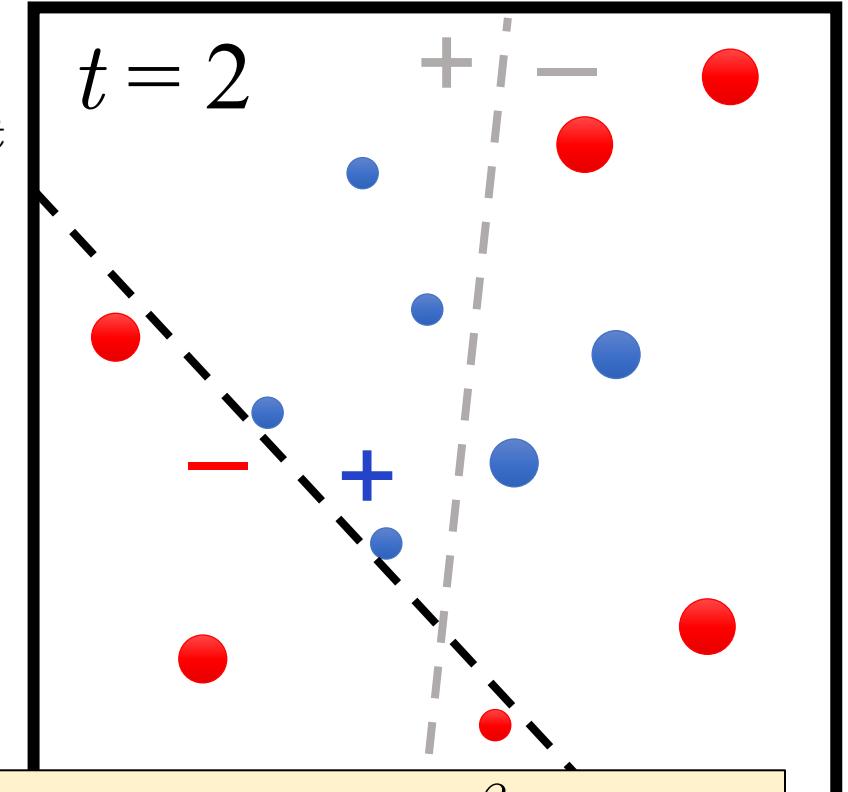
```
1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
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
```

$H(x)$

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

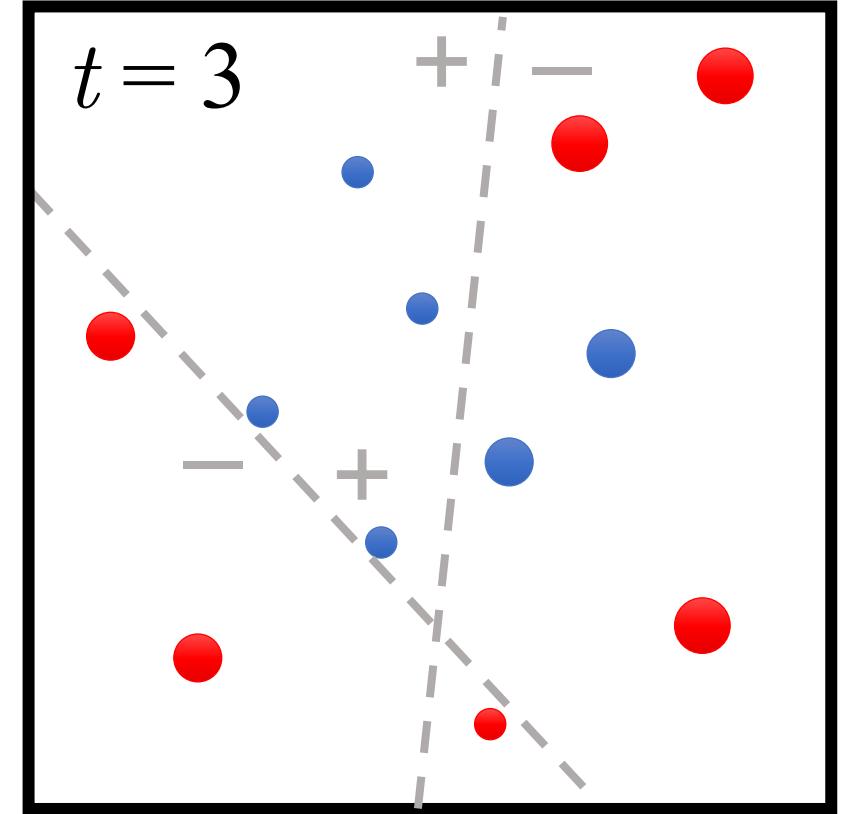


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

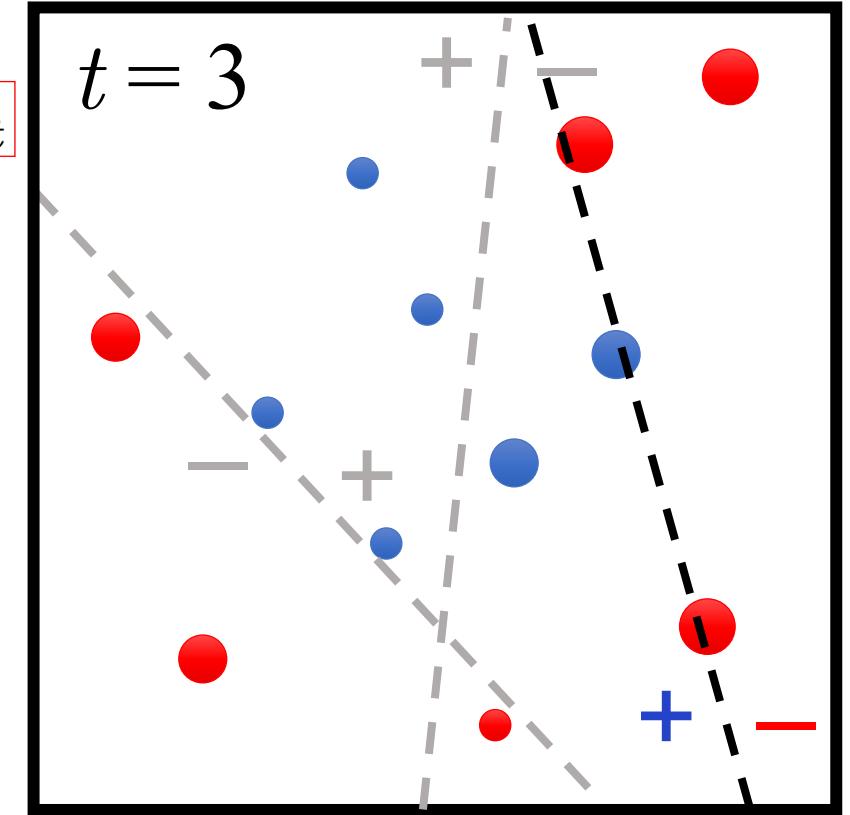


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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



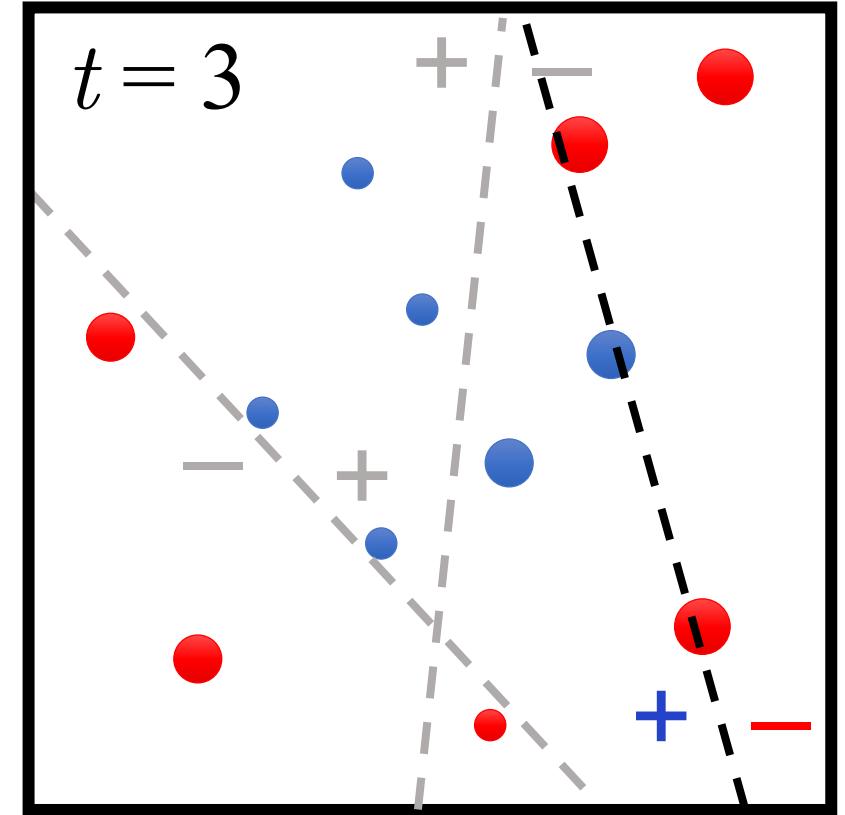
AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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

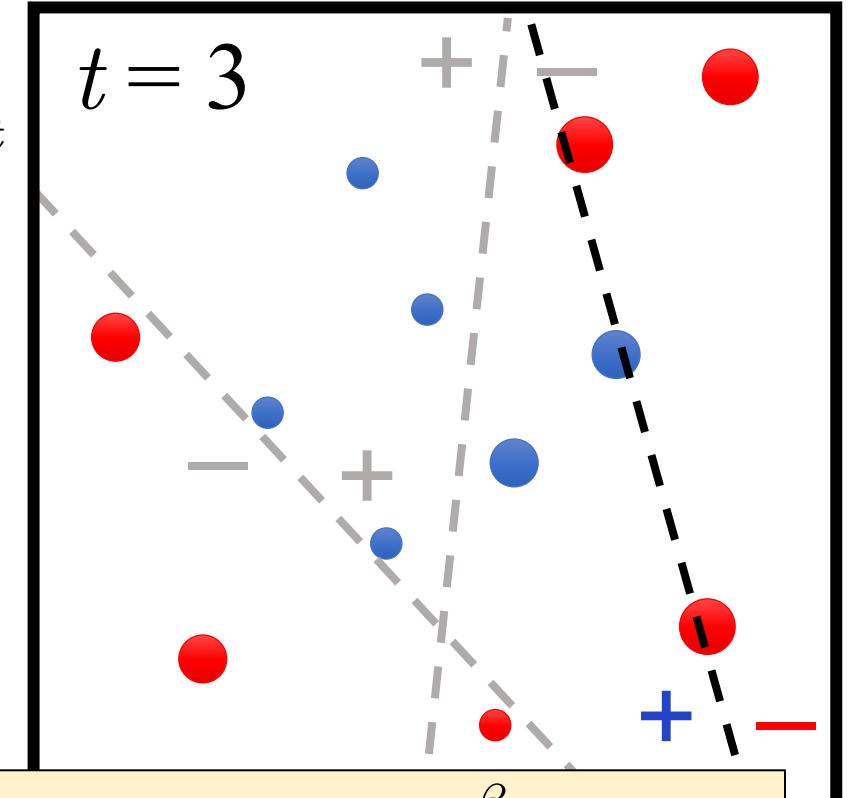
```
1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
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
```

$H(x)$

- 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

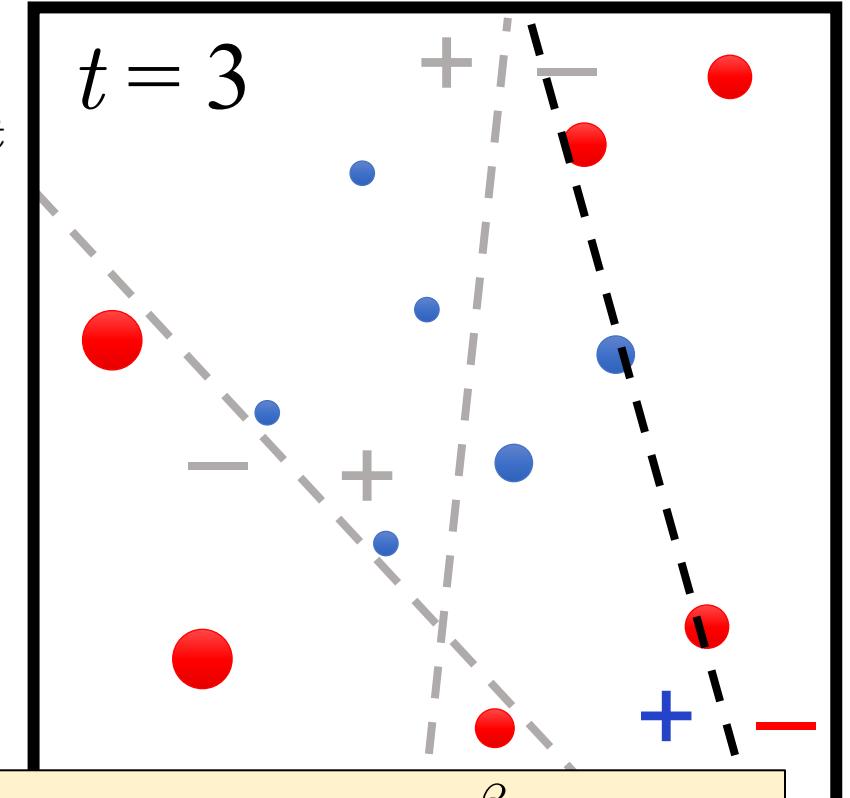
```
1: Initialize a vector of  $n$  uniform weights  $\mathbf{w}_1$ 
2: for  $t = 1, \dots, T$ 
3:   Train model  $h_t$  on  $X, y$  with instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
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
```

$H(x)$

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

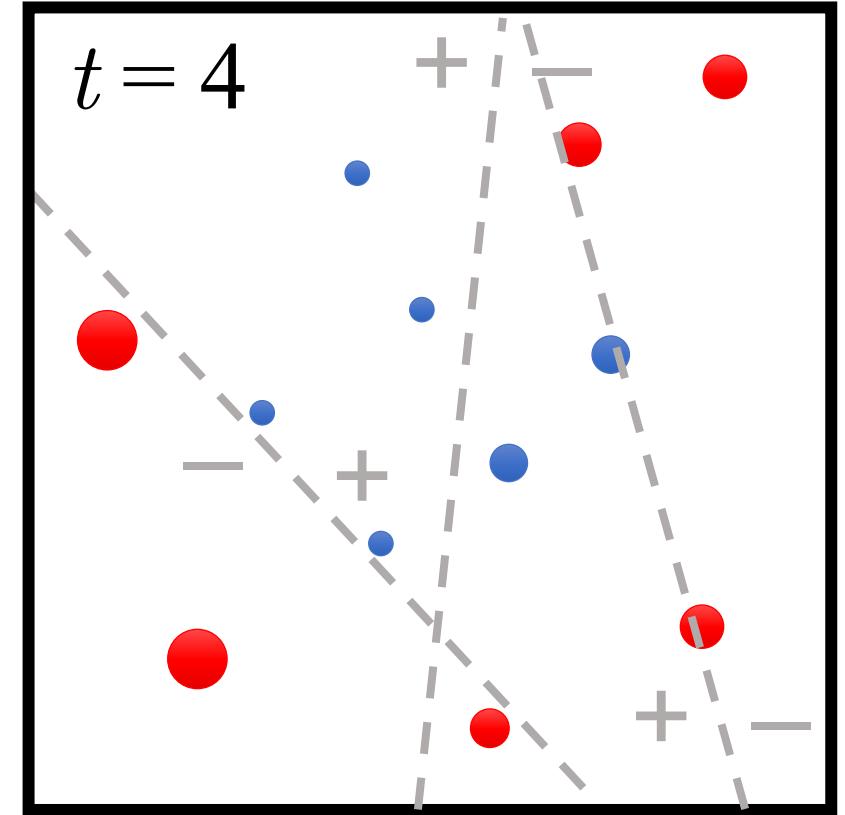


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

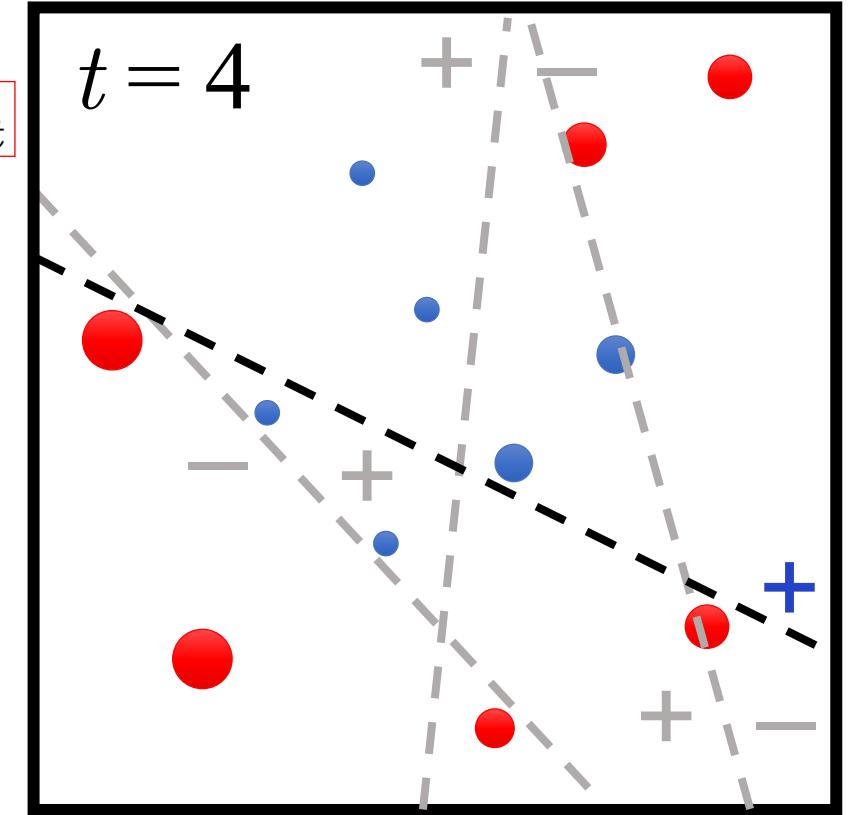


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

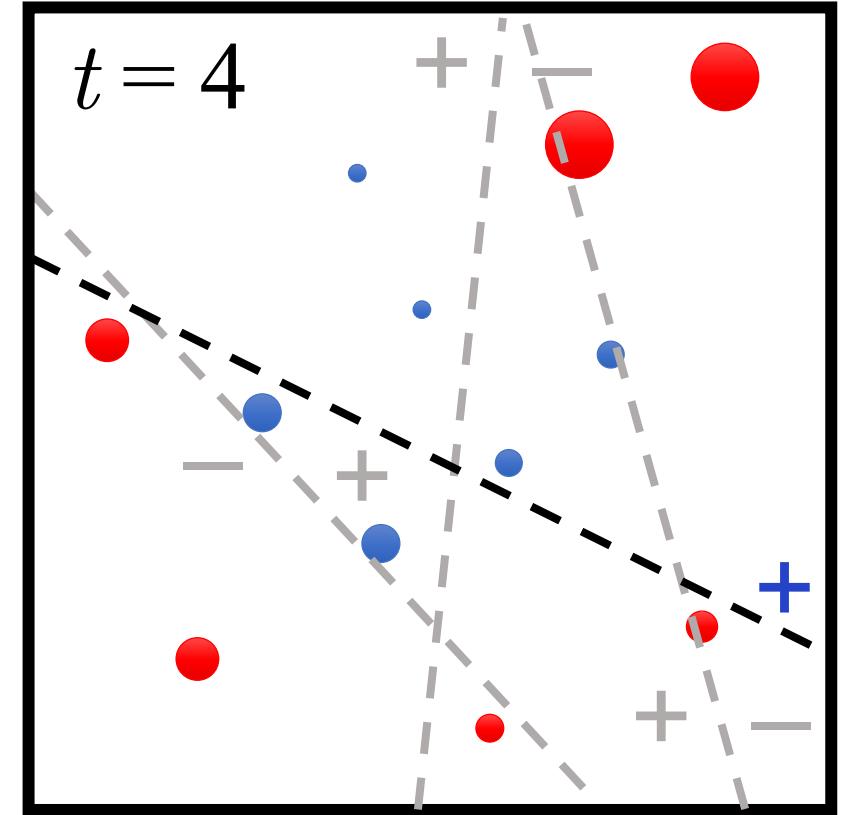


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

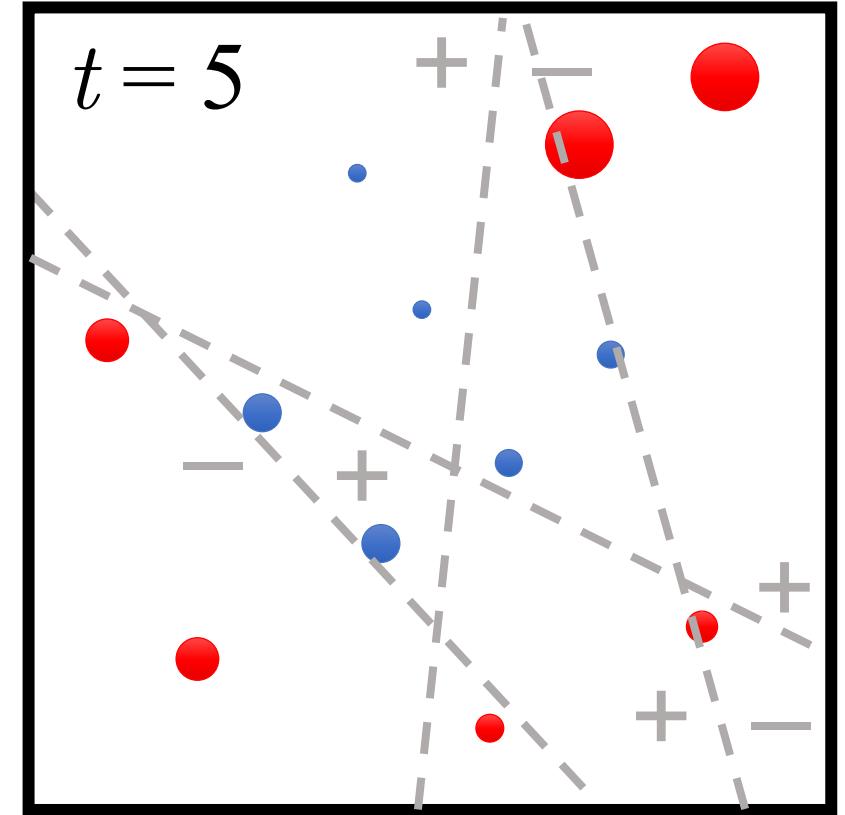


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

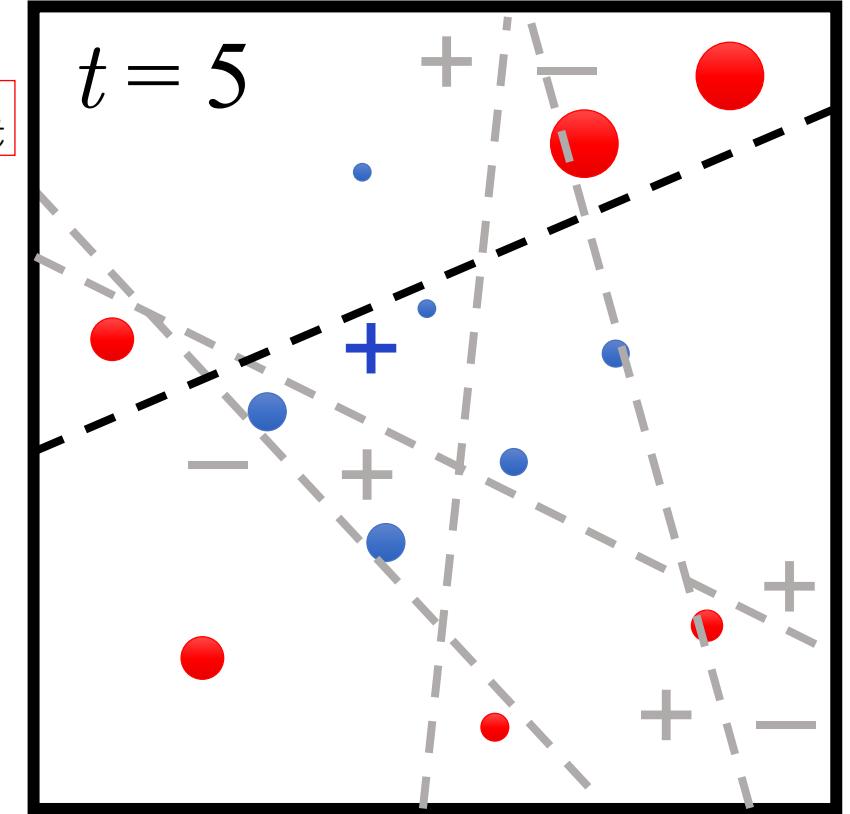


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

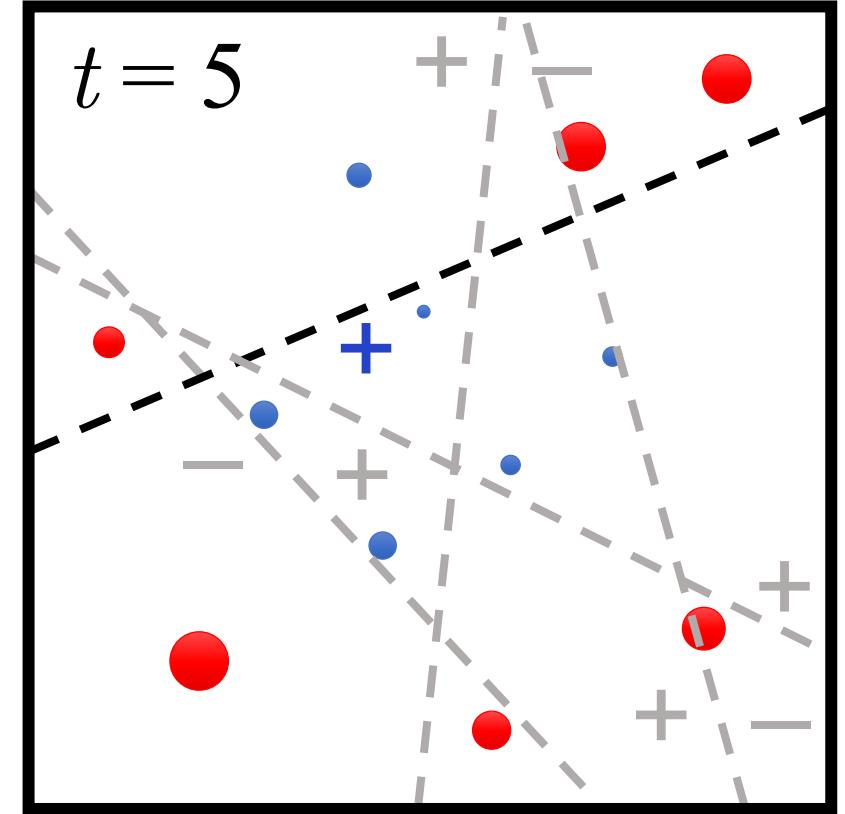


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

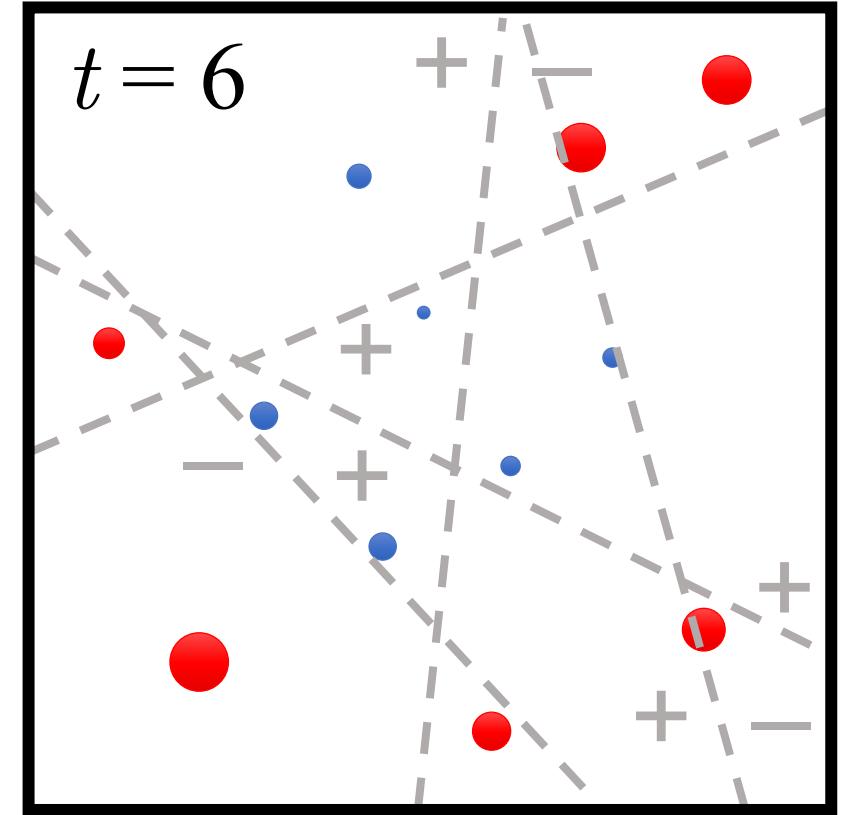


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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



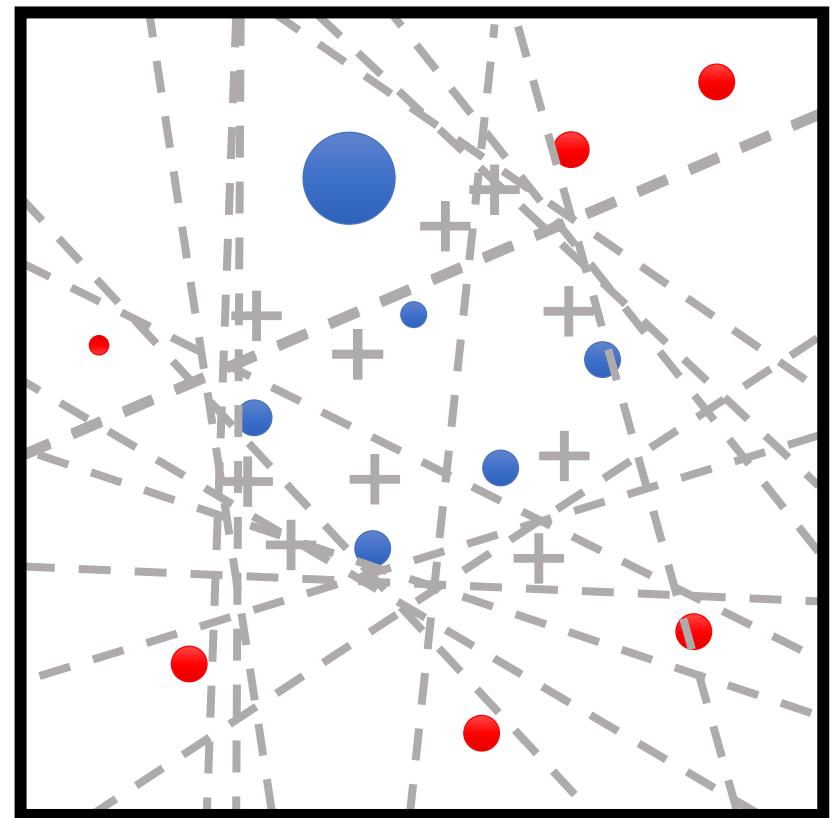
AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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
```

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

$t = T$

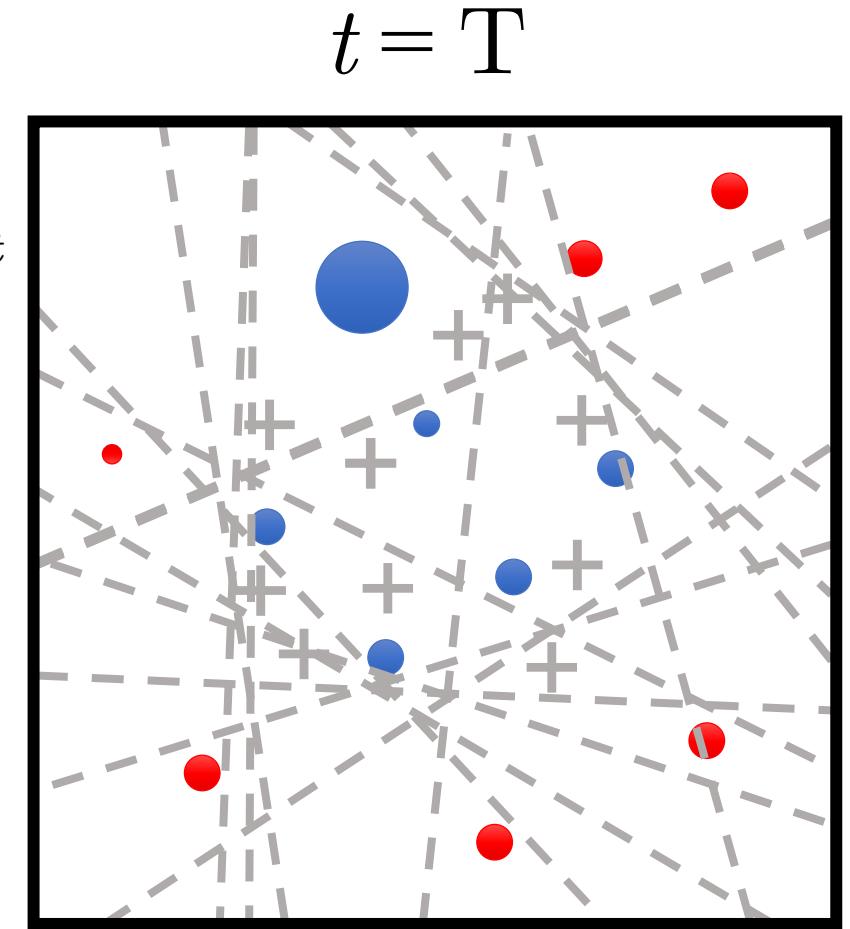


AdaBoost

```
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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  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 instance weights  $\mathbf{w}_t$ 
4:   Compute the weighted training error  $\epsilon_t$  of  $h_t$ 
5:   Choose  $\beta_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$ 
6:   Update all instance weights:
```

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

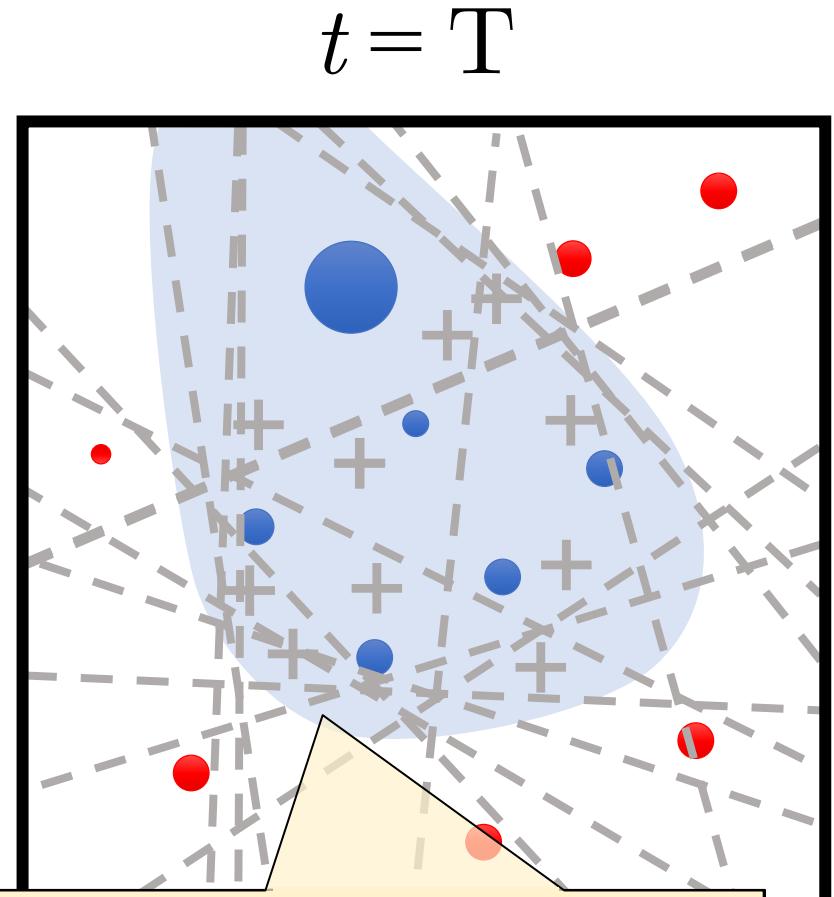
```
7:   Normalize  $\mathbf{w}_{t+1}$  to be a distribution
8: end for
```

```
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

[Freund & Schapire, 1997]

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

All points start with equal weight

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** $t = 1, \dots, T$ **do**
- 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

[Freund & Schapire, 1997]

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

- 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
 Compute the weighted training error rate of h_t :
$$\epsilon_t = \sum_{i:y_i \neq h_t(\mathbf{x}_i)} w_{t,i}$$
 - 4: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
 - 5: Update all instance weights:
$$w_{t+1,i} = w_{t,i} \exp(-\beta_t y_i h_t(\mathbf{x}_i)) \quad \forall i = 1, \dots, n$$
 - 6: Normalize \mathbf{w}_{t+1} to be a distribution:
$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \dots, n$$
 - 7: **end for**
 - 8: **Return** the hypothesis $H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$

Training a Model with Weighted Instances

- Many algorithms can simply incorporate weights w into the objective
 - Essentially, weigh the cost of misclassification differently for each instance
 - e.g., for logistic regression:

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

- 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
 - Example weak learners:
 - Decision stumps (1-level decision trees)
 - Depth-limited decision trees
 - Linear 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

AdaBoost

[Freund & Schapire, 1997]

Error is the sum the weights of all misclassified instances

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

[Freund & Schapire, 1997]

- 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

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

[Freund & Schapire, 1997]

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.

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 $\mathbf{w}_1 = [\frac{1}{n}, \dots, \frac{1}{n}]$

\dots, T

odel h_t on X, y with instance weights \mathbf{w}_t
e the weighted training error rate of h_t :

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

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

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

[Freund & Schapire, 1997]

Make \mathbf{w}_{t+1} sum to 1

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

[Freund & Schapire, 1997]

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

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

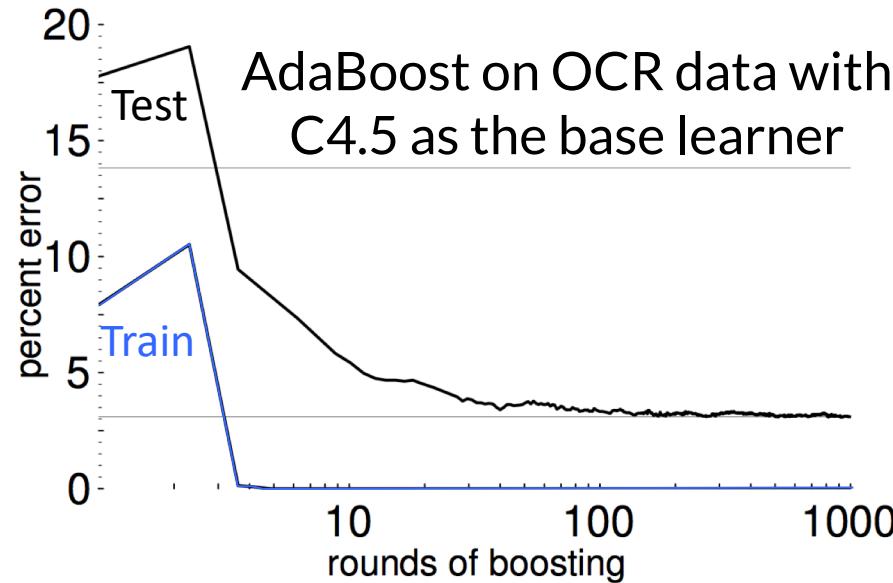
Dynamic Behavior of AdaBoost

- 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

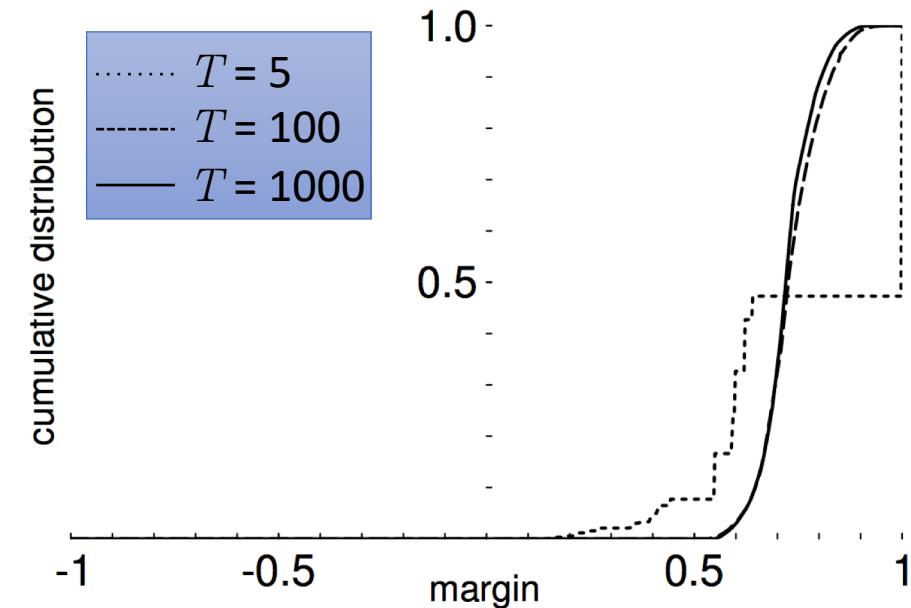
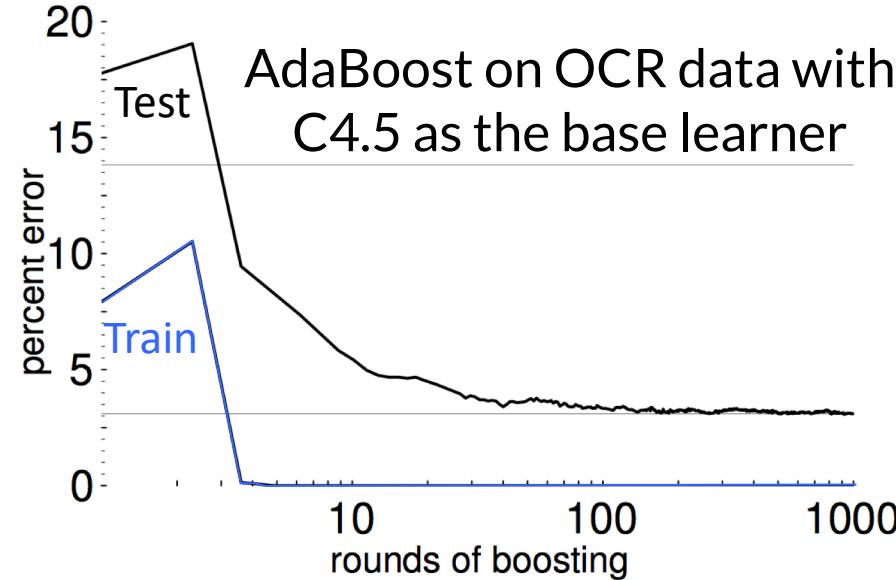
- VC Theory originally predicted that AdaBoost would always overfit as T grew large
 - Hypothesis keeps growing more complex
- In practice, AdaBoost often did not overfit, contradicting VC theory
- Also, AdaBoost does not explicitly regularize the model

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:

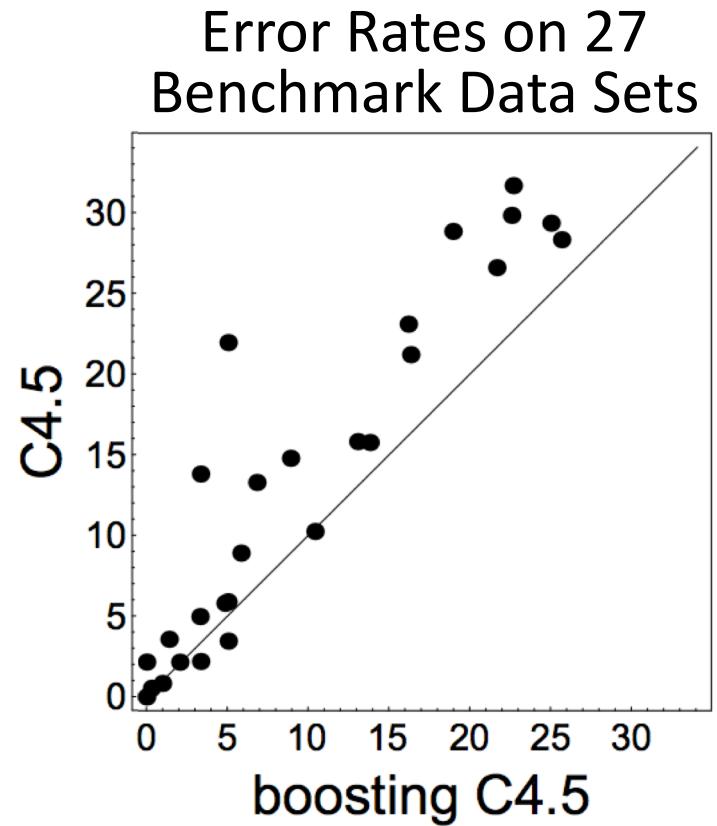
- Fast and simple to program
- No parameters to tune (besides T)
- No assumptions on weak learner

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)