

CS M146: Introduction to Machine Learning

Ensembles: Bagging and Boosting

Aditya Grover



<https://aditya-grover.github.io/>



@adityagrover_

Recap

- **Logistic Regression**
- **Perceptrons**
- **Linear Regression**
- **Decision Trees**
- **k NN**
- **Kernalized Linear Models**
- **SVMs**

Ensemble Learning

Consider a set of **base classifiers** h_1, h_2, \dots, h_L

Idea: construct an **ensemble classifier** $H(x)$ that combines the individual decisions of its members h_1, h_2, \dots, h_L . For example:

- could have the base classifiers vote, or
- could use different base for different regions of the instance space

Desiderata

- Good members: Works well if the members each have low error rates
- Successful ensembles require **diversity** in its members
 - Base classifiers should make different mistakes
 - Can have different types of base classifiers

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

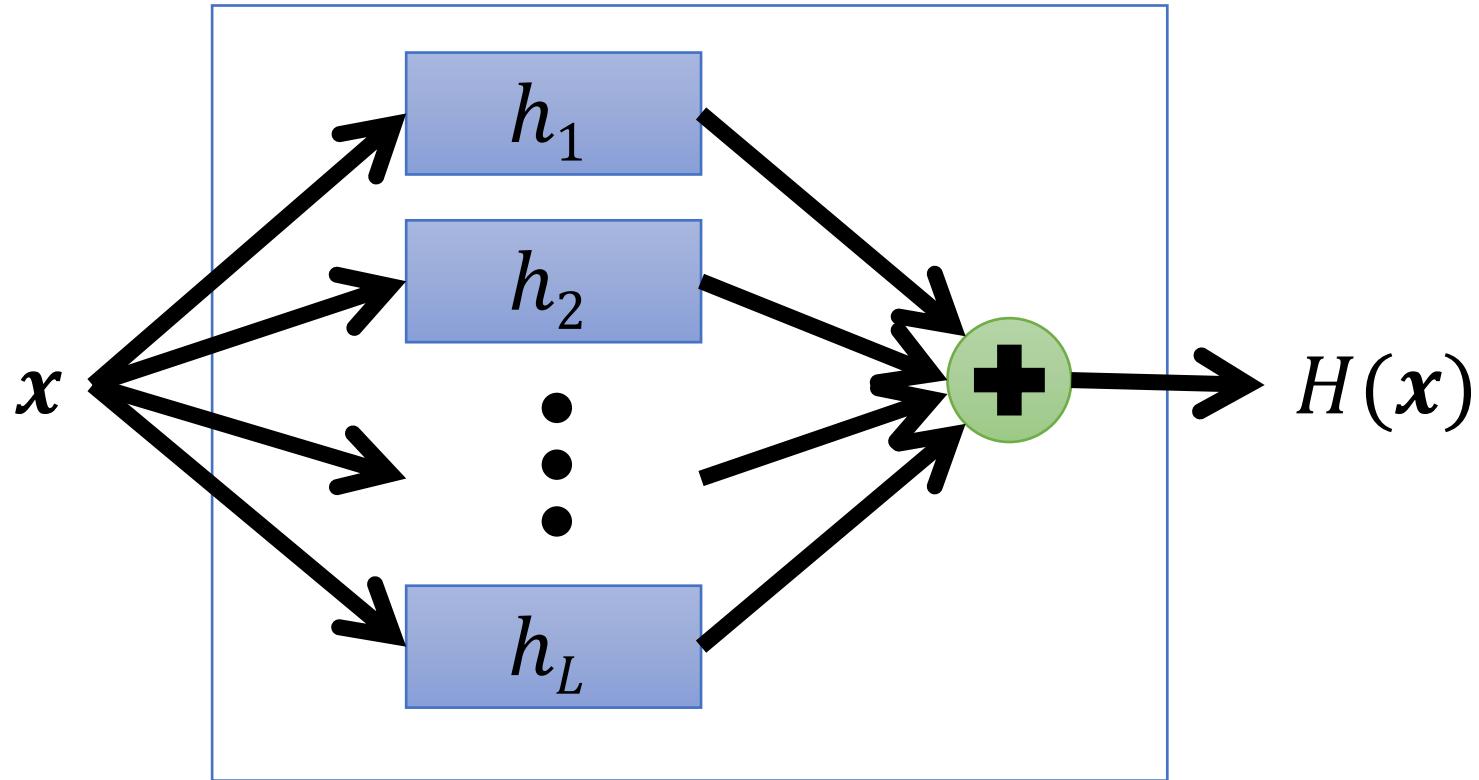


This application is called “collaborative filtering”

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

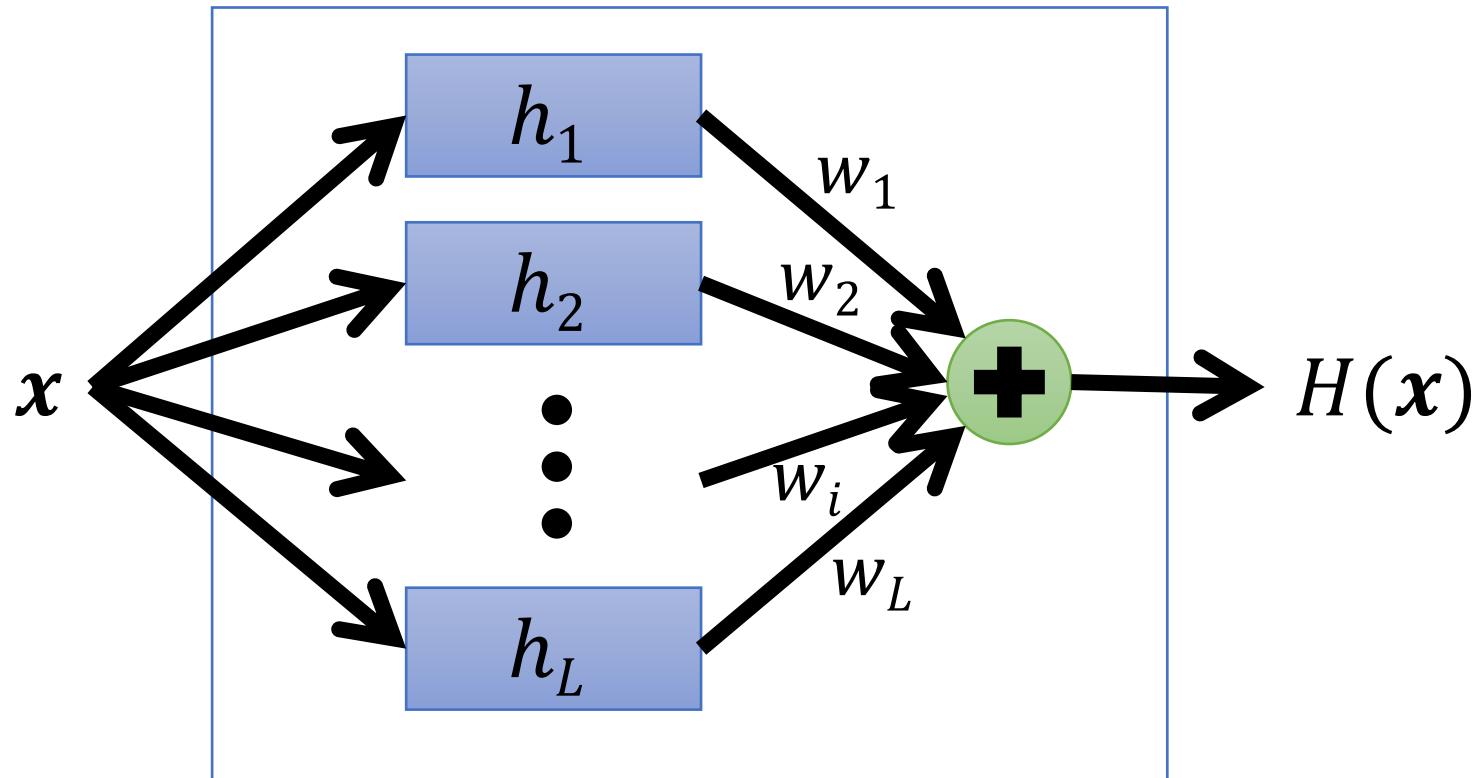
Winner: BellKor's Pragmatic Chaos – an ensemble of more than 800 rating systems

Combining Classifiers: Averaging



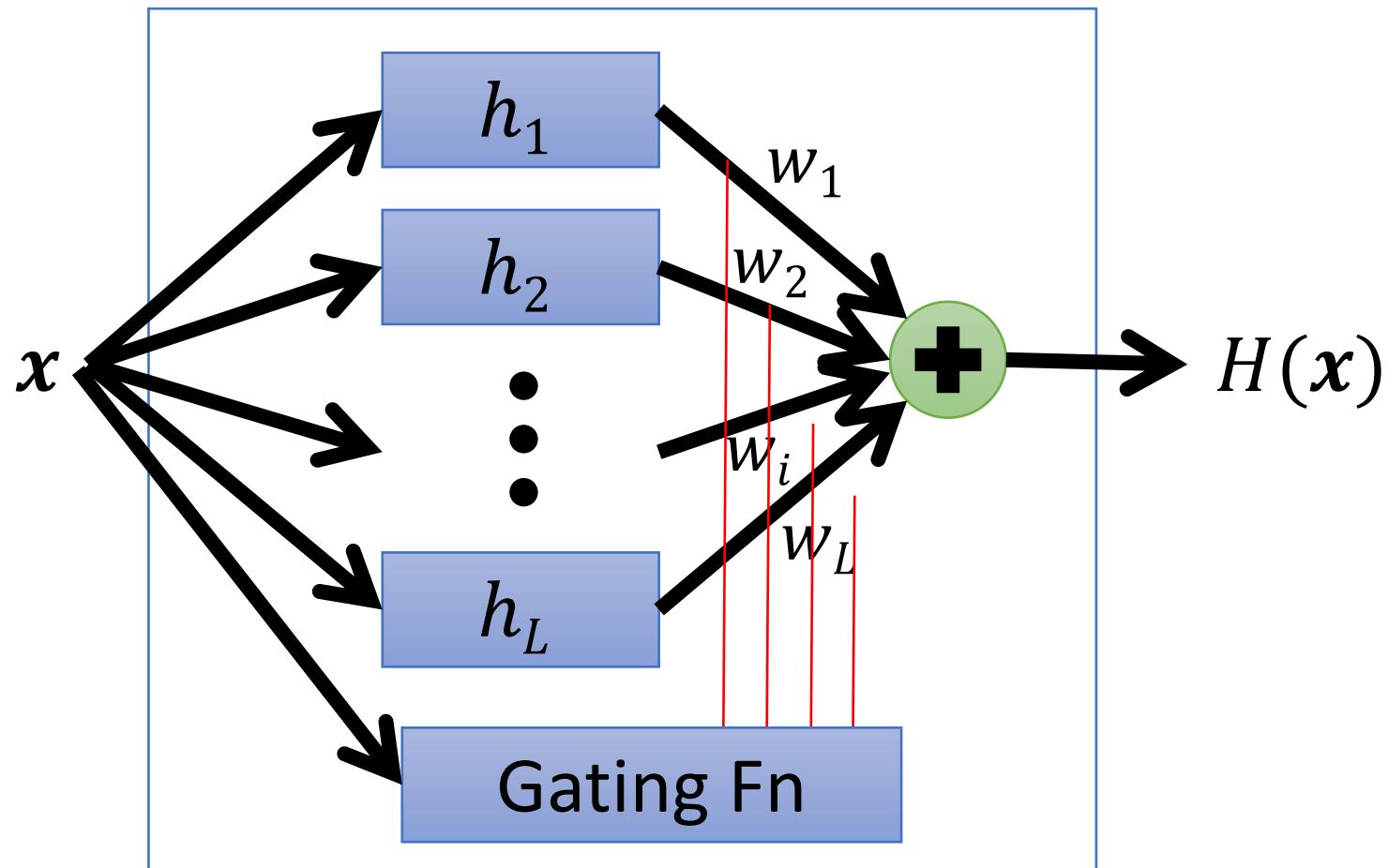
- Final hypothesis is a simple vote of the members

Combining Classifiers: Weighted Averaging



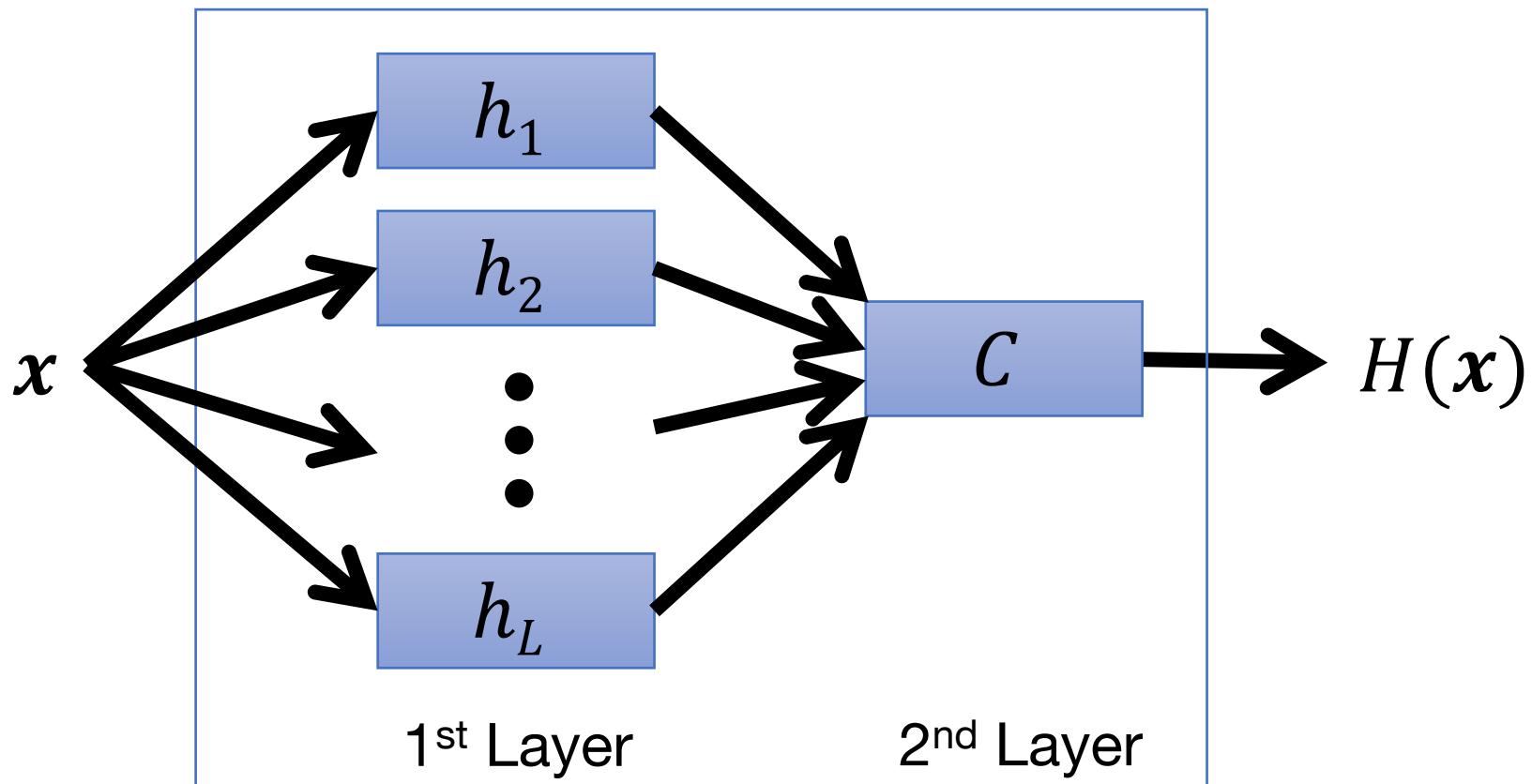
- Final hypothesis is a weighted vote of the members
- Coefficients of individual members are trained using a validation set

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

Achieving Diversity in Ensembles

Cause of the Mistake

Pattern was very difficult

Overfitting

Some features are noisy

Diversification Strategy

Hopeless

Vary the training sets

Vary the set of input features

Note: The diversification strategies are not exclusive and can be combined with each other

Bagging

Manipulating the Training Data

Bootstrap replication:

- Given n training examples, construct a new training set by sampling n instances with replacement

```
n=5; dataset_indices=np.arange(n)  
/* Sampling with replacement */  
np.random.choice(dataset_indices, size=5, replace=True)  
// Eg output: array([3, 1, 0, 0, 0])  
/* Sampling without replacement */  
np.random.choice(dataset_indices, size=5, replace=False)  
// Eg output array([1, 0, 3, 2, 4])
```

- Excludes ~30% of the training instances on average due to repeats

Manipulating the Training Data

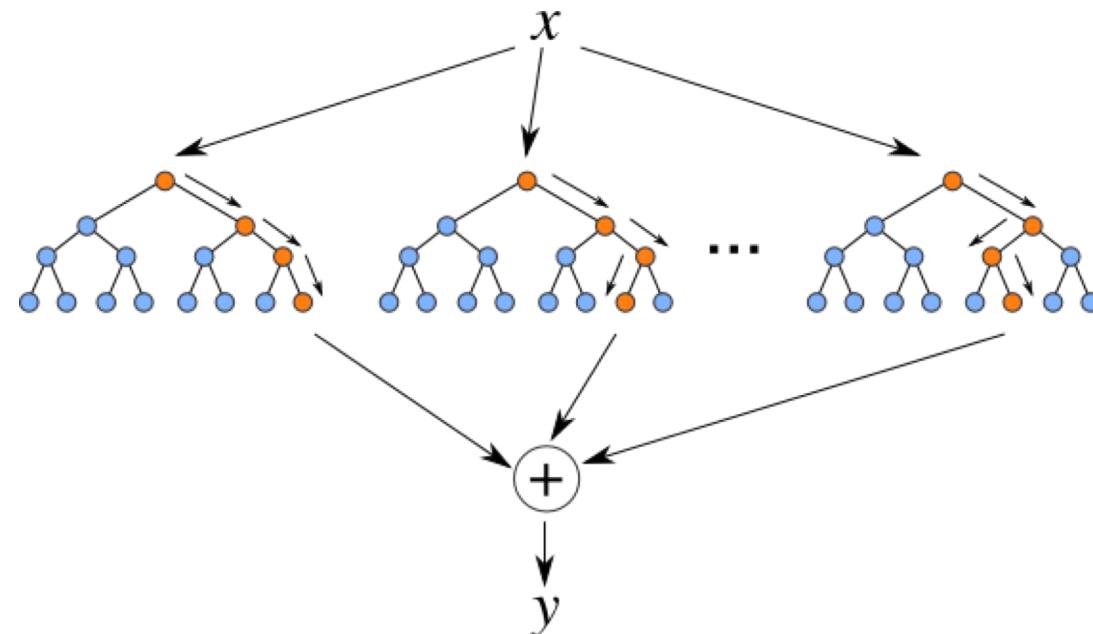
Bagging

- Create **multiple** bootstrap replicates of training set
- Train a classifier (e.g., a decision tree) for each replicate
- Estimate classifier performance using out-of-bootstrap data (i.e., indices not sampled for the training bootstrap replicate)
 - Can use for hyperparameter validation, assigning weights to classifiers
- Average output of all classifiers as final hypothesis

Manipulating the Features

Random Forests

- Extension of bagging that manipulates both data & feature sets
- 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



Boosting

AdaBoost

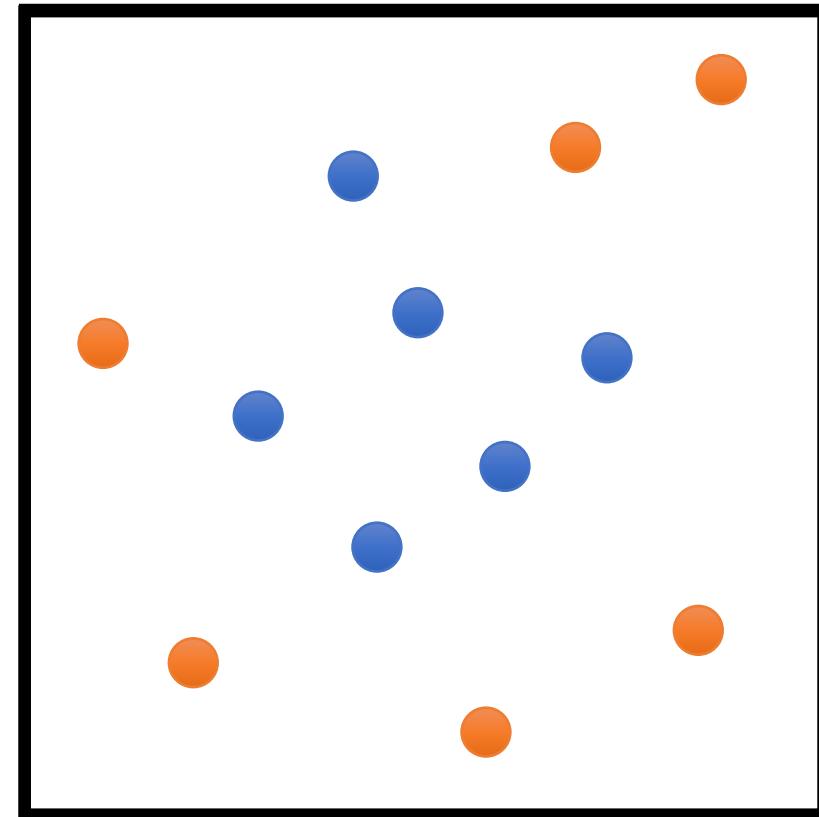
Kearns and Valiant (1988, 1989): "Can a **set of weak learners** create a **single strong learner**?"

- One of the most profound questions in history of ML.
Adaboost [Freund & Schapire, 1997] answers it in the affirmative
- A meta-learning algorithm with great theoretical and empirical performance
- Turns a base learner (i.e., a “weak hypothesis”) into a high performance classifier 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
```

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

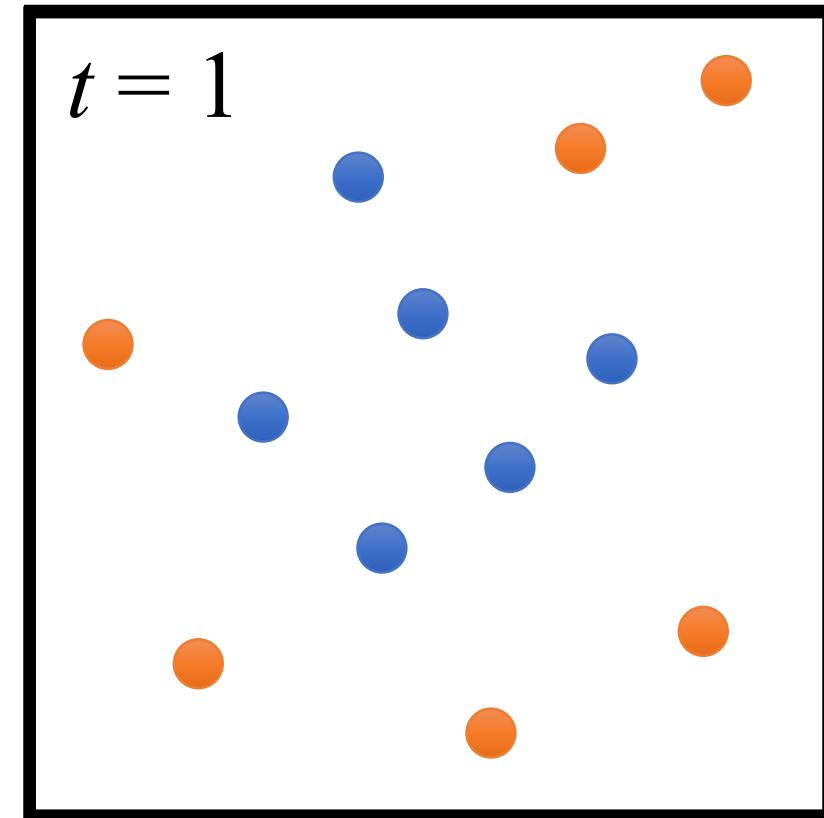


- Note: Here $\mathbf{w}_i \in \mathbb{R}^n$ is an n-dimensional vector with entries given as $w_{i,j}$
- In the figure, size of point represents the instance's weight at current round t
- **Note: Do not confuse instance weights with weight parameters (from prior lectures)**
 - AdaBoost is only concerned with the former.

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
```

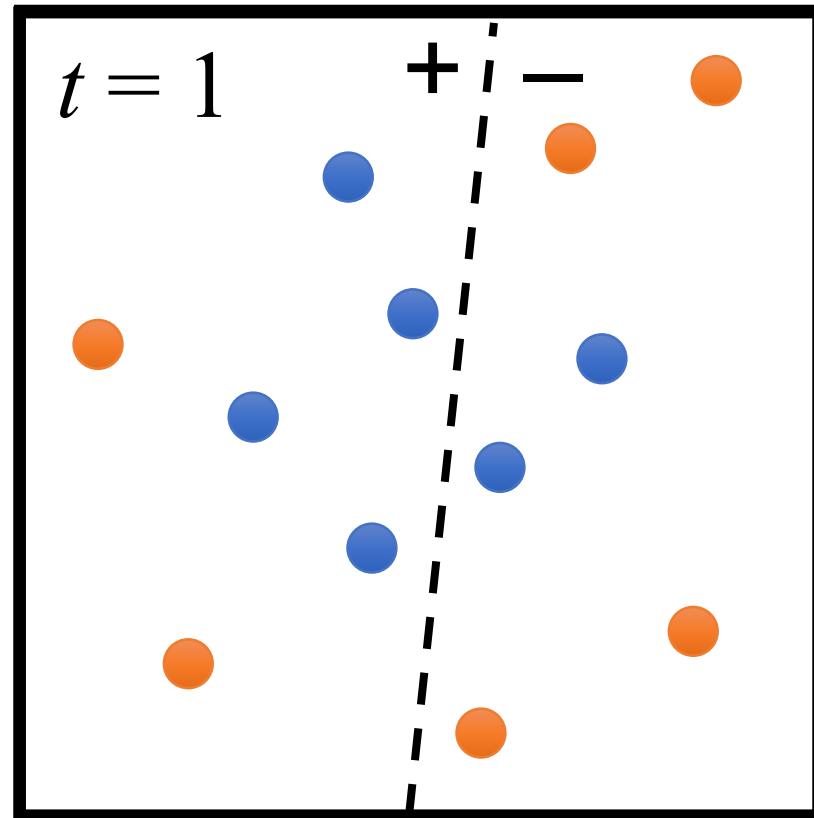
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

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



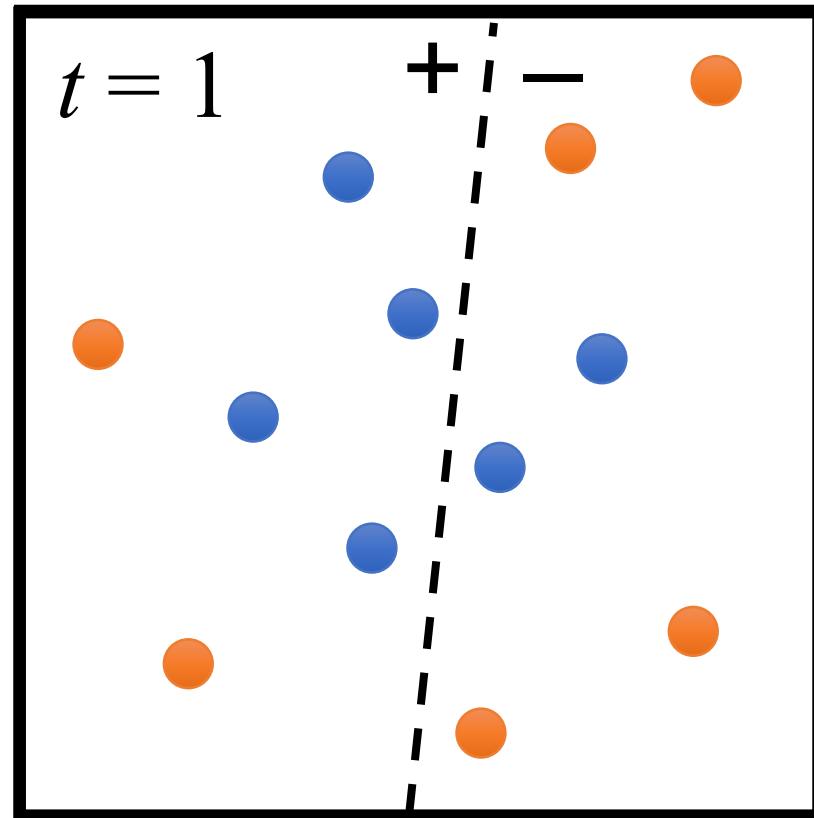
AdaBoost

```
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)

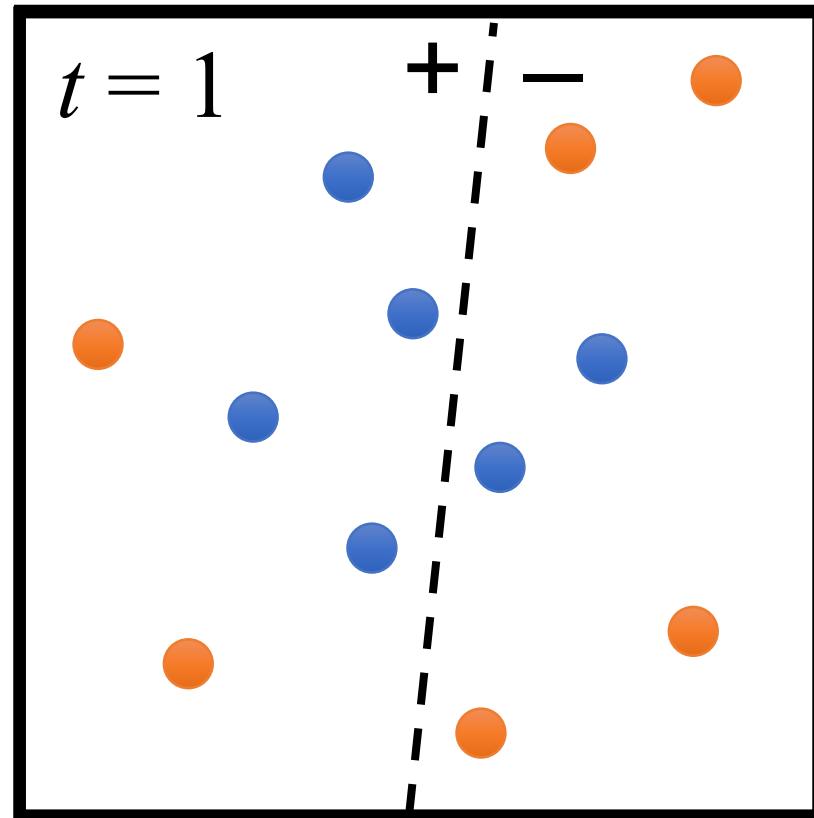
AdaBoost

```
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)

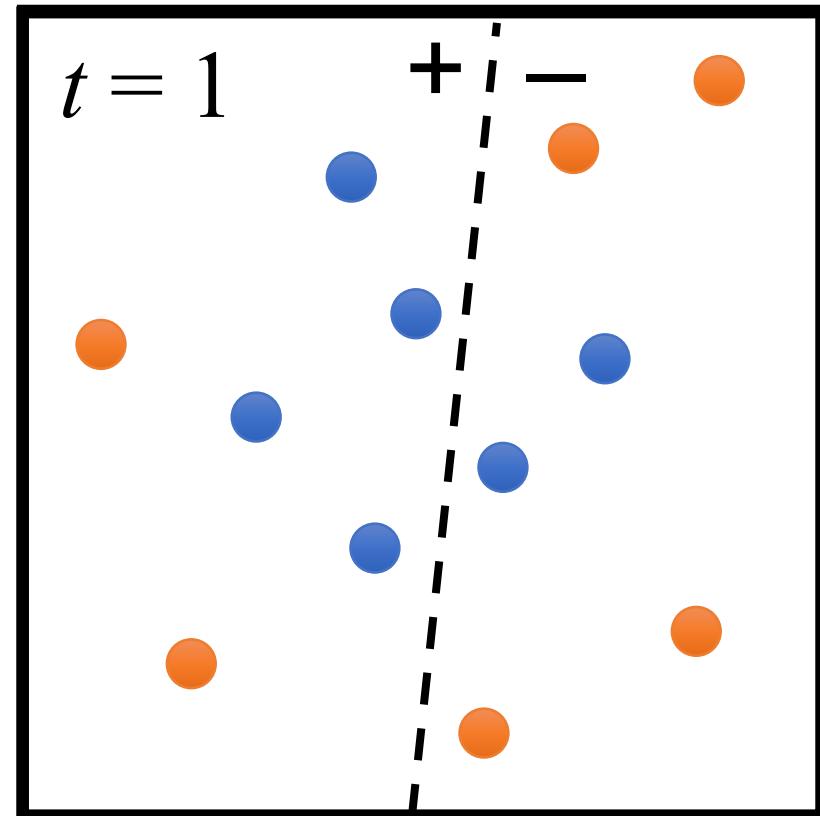
AdaBoost

```
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
```

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

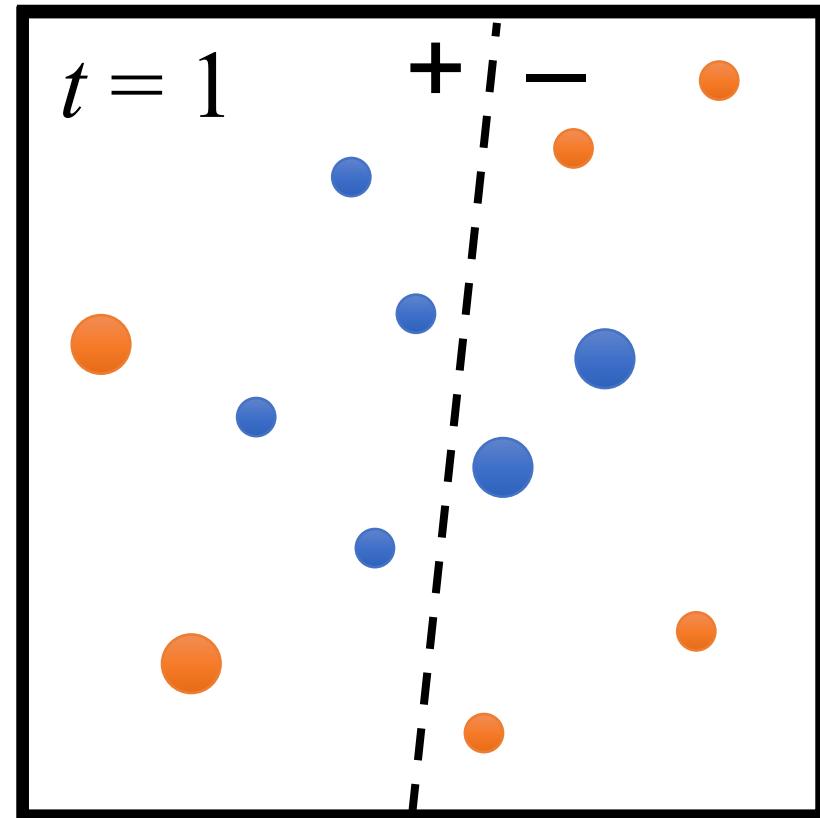


- Weights of correct predictions are multiplied by $\exp(-\beta_t) \leq 1$
- Weights of incorrect predictions are multiplied by $\exp \beta_t \geq 1$

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
```

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

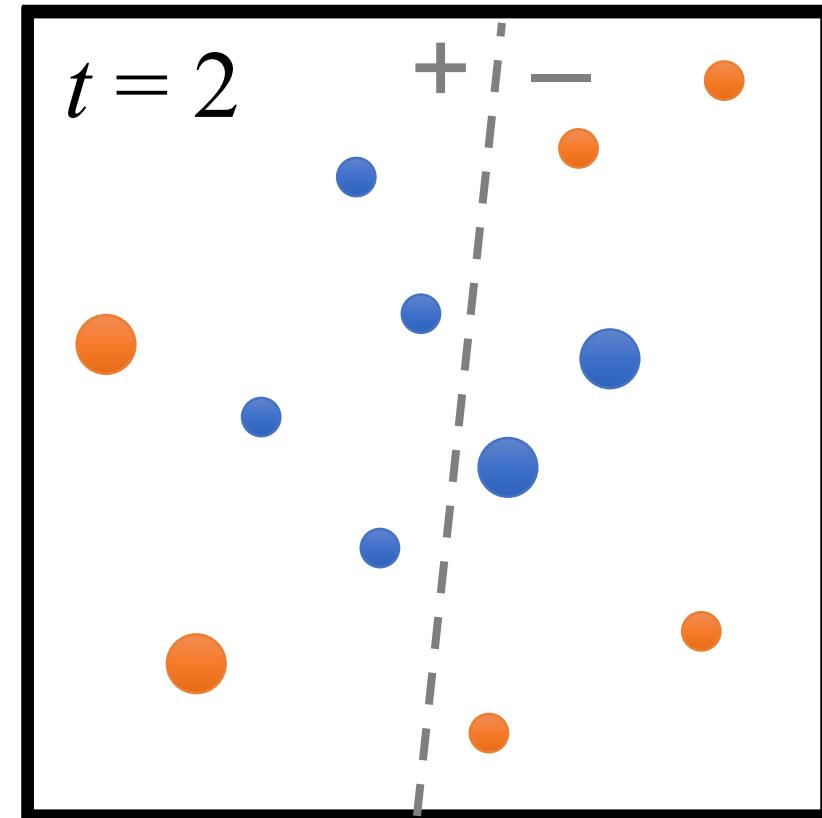


Disclaimer: Note that resized points in the illustration above are not necessarily to scale with β_t

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
```

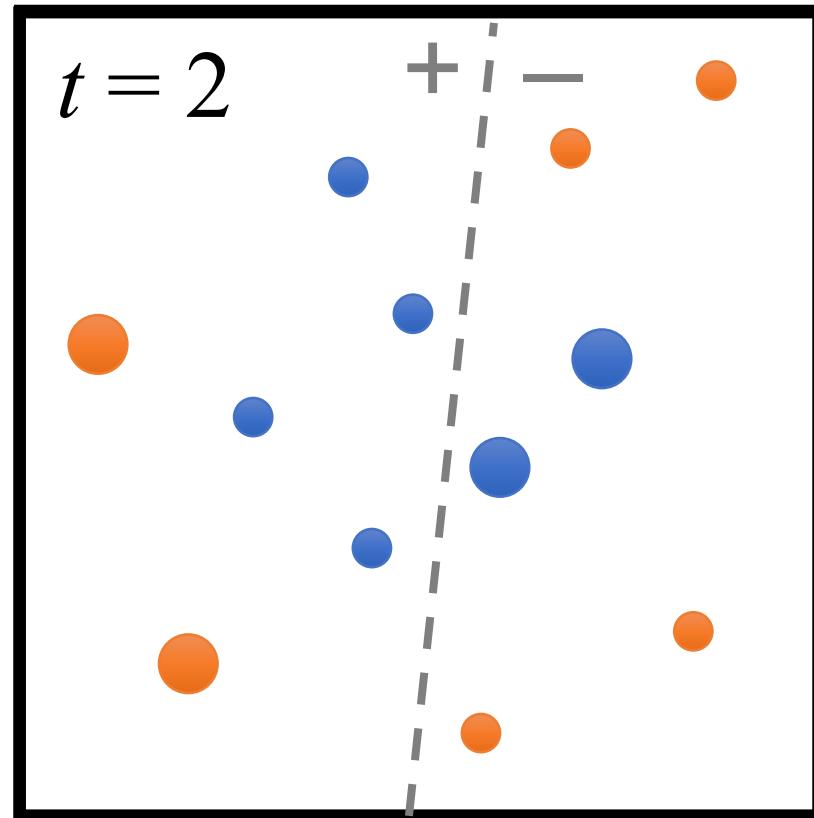
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

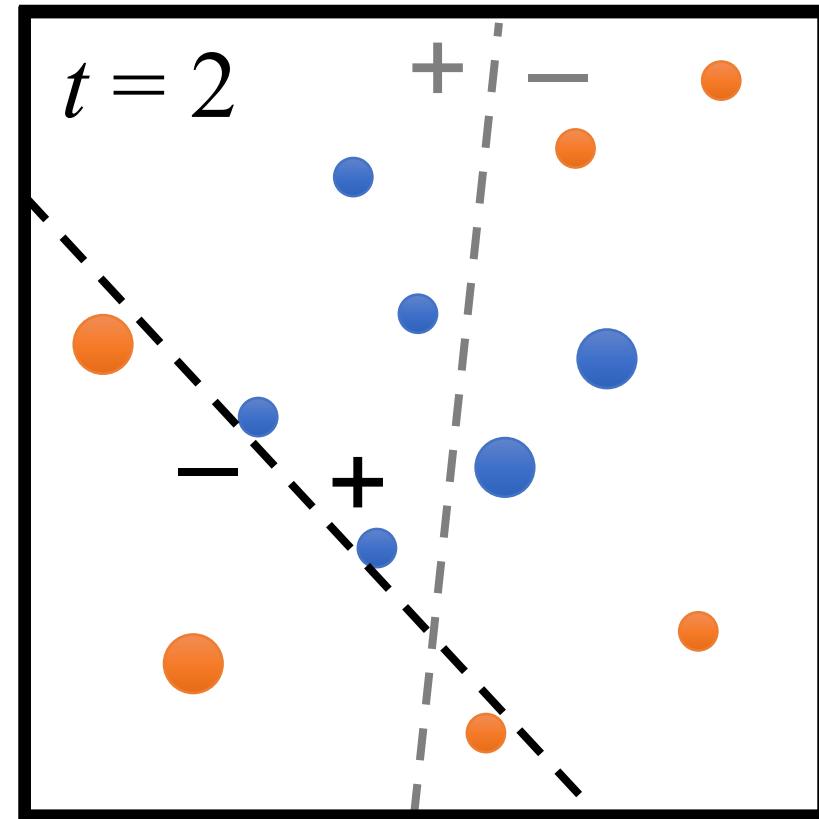
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

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



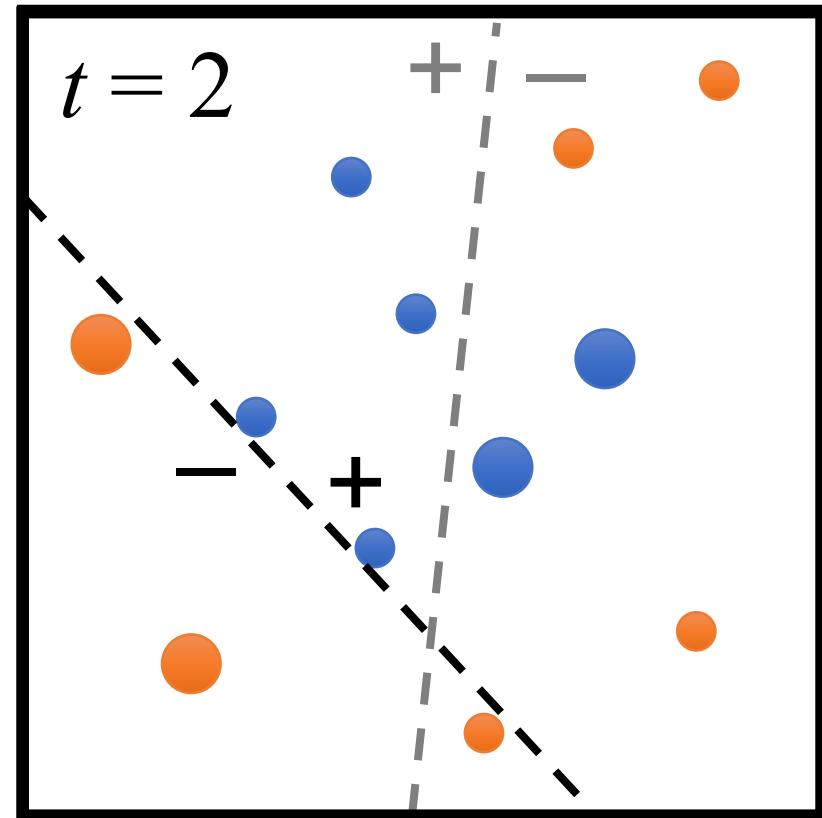
AdaBoost

```
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)

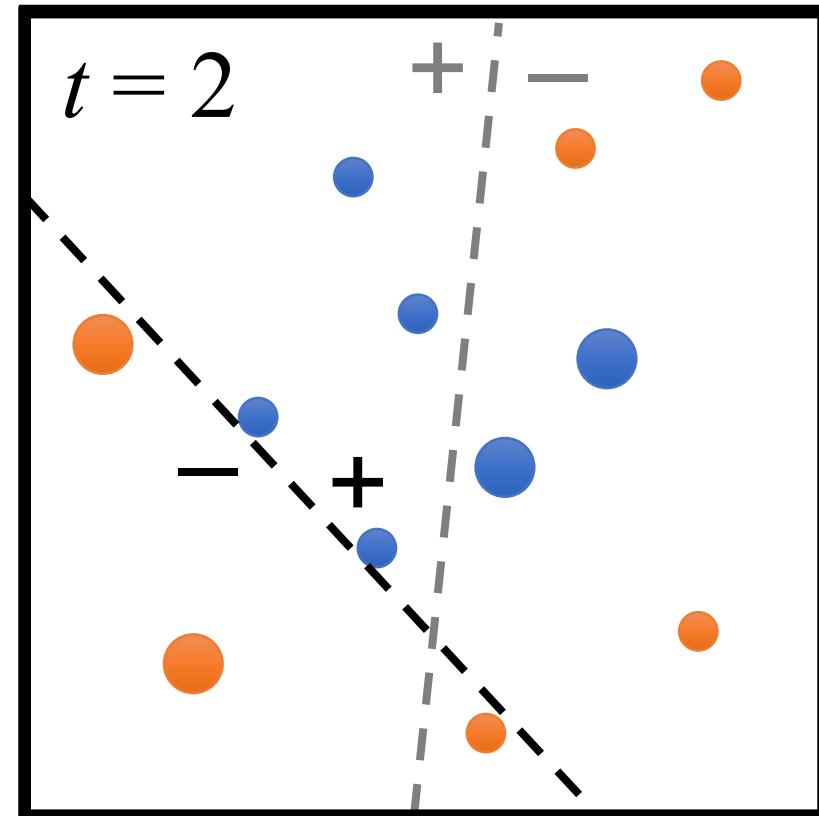
AdaBoost

```
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
```

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

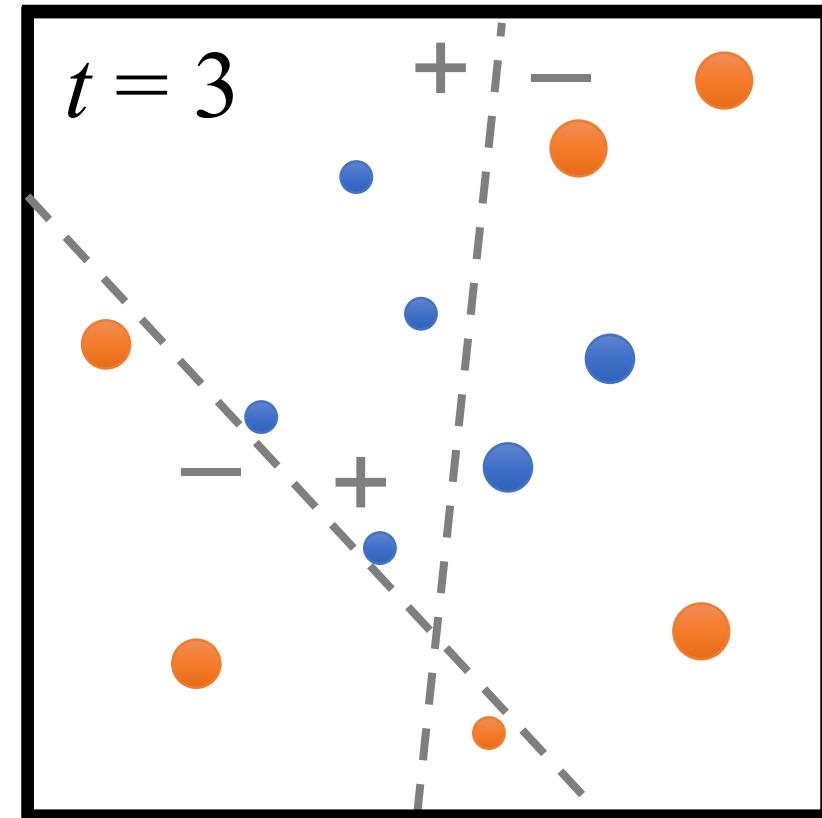


- Weights of correct predictions are multiplied by $\exp(-\beta_t) \leq 1$
- Weights of incorrect predictions are multiplied by $\exp \beta_t \geq 1$

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
```

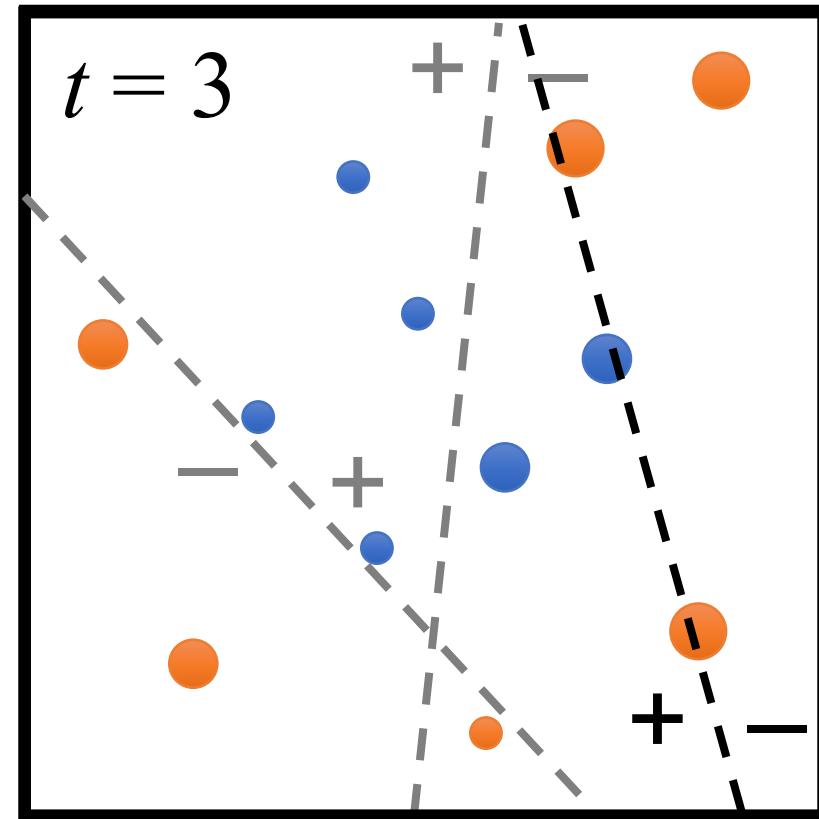
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

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



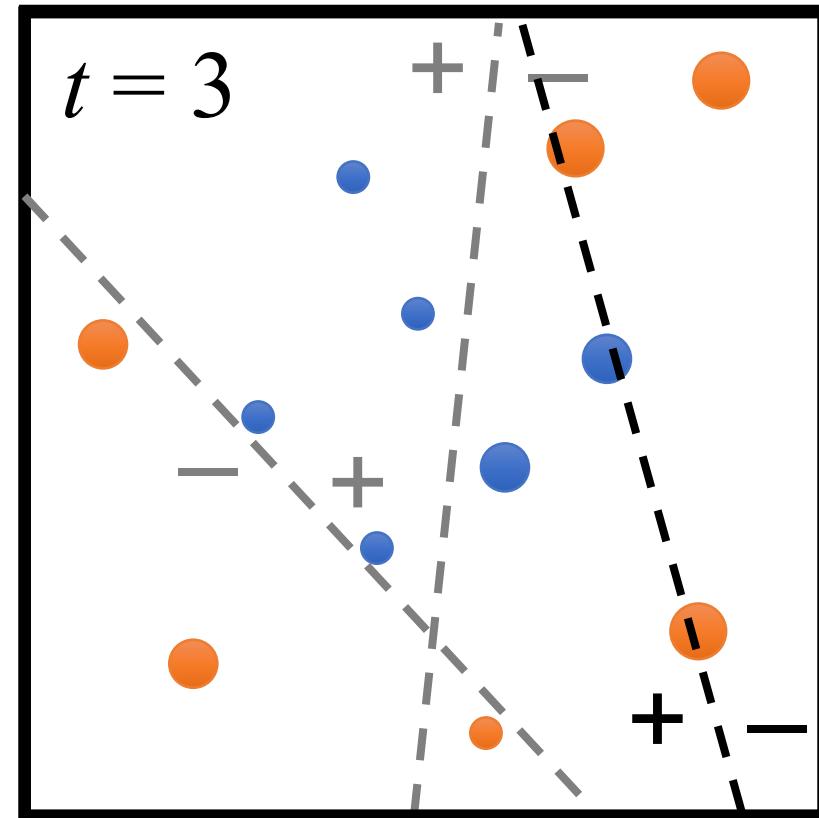
AdaBoost

```
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)

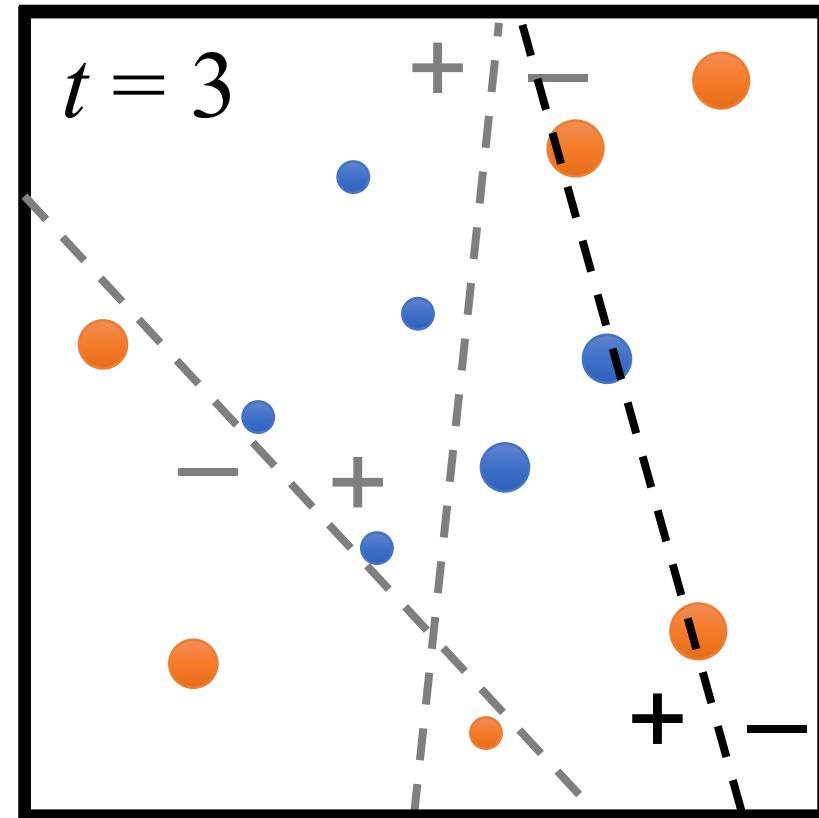
AdaBoost

```
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 $\exp(-\beta_t) \leq 1$
- Weights of incorrect predictions are multiplied by $\exp \beta_t \geq 1$

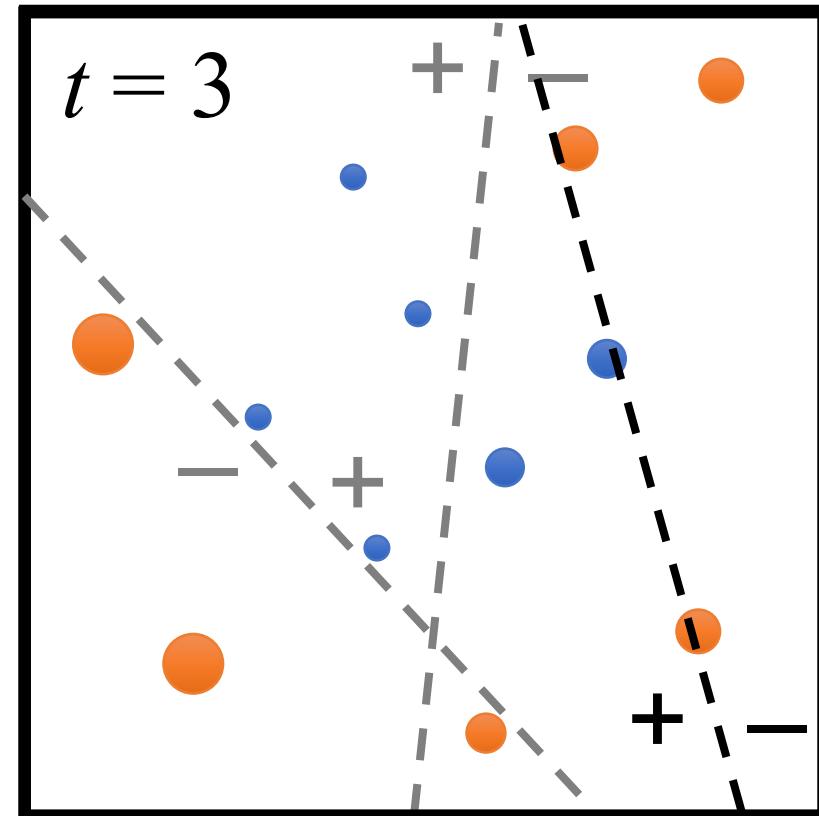
AdaBoost

```
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
```

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

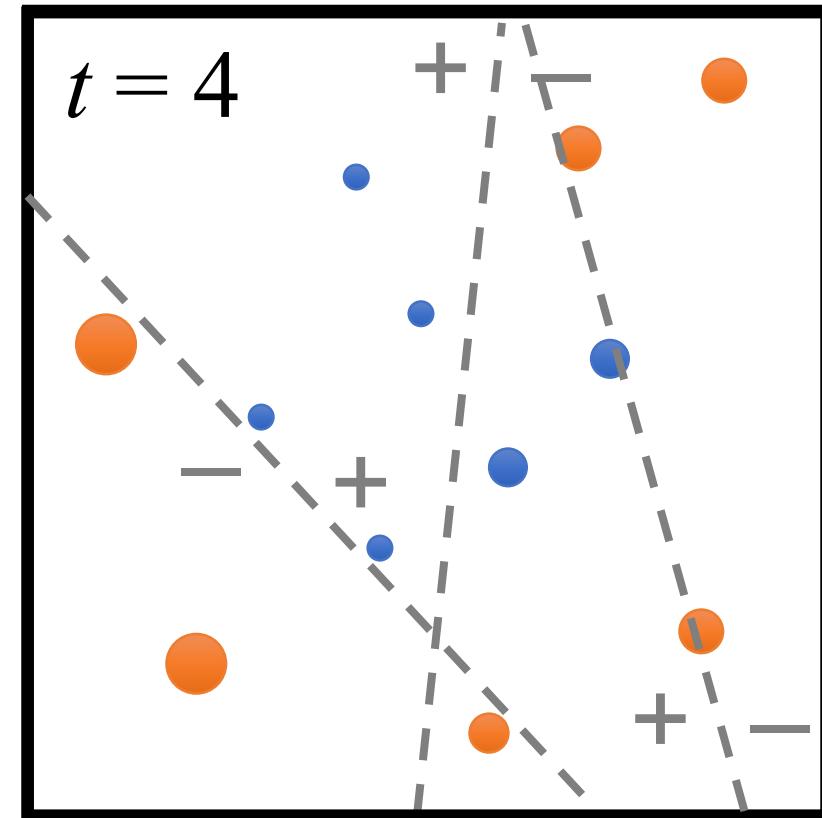


- Weights of correct predictions are multiplied by $\exp(-\beta_t) \leq 1$
- Weights of incorrect predictions are multiplied by $\exp \beta_t \geq 1$

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
```

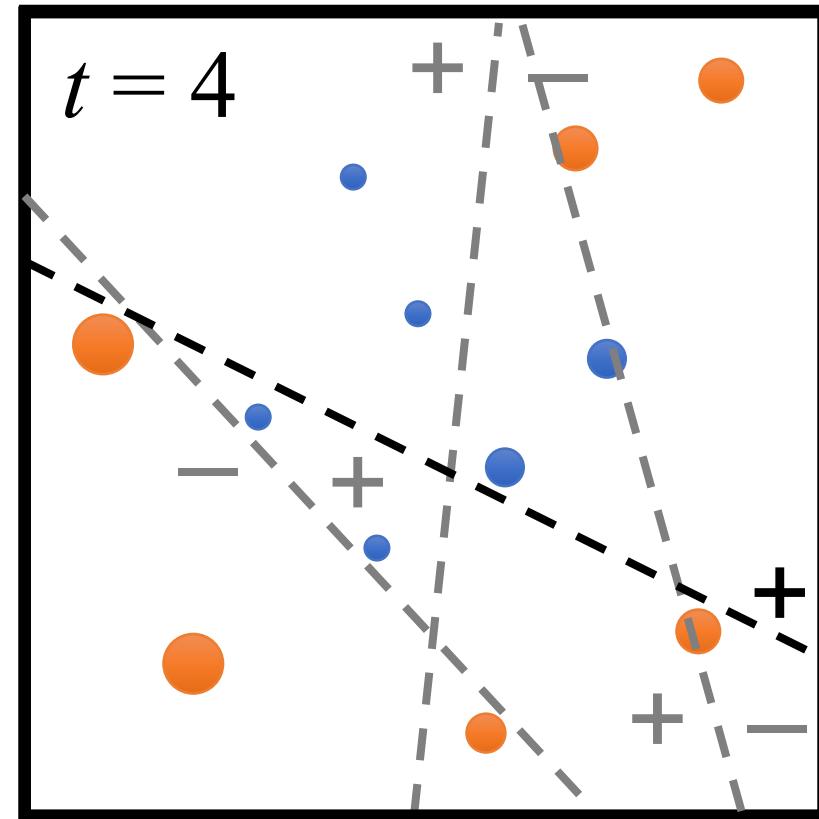
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

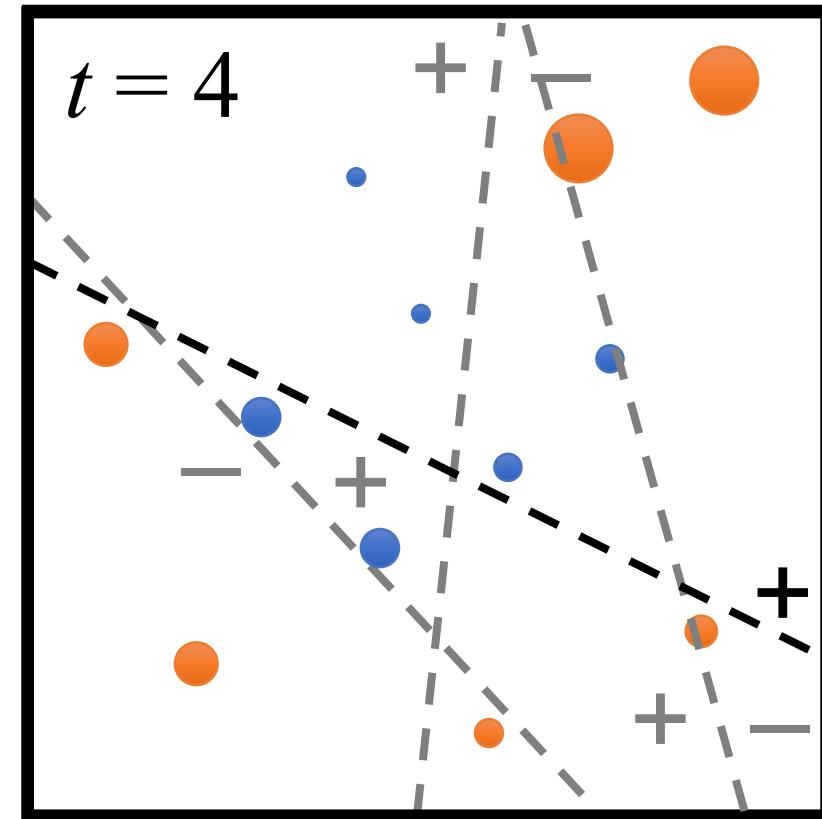
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

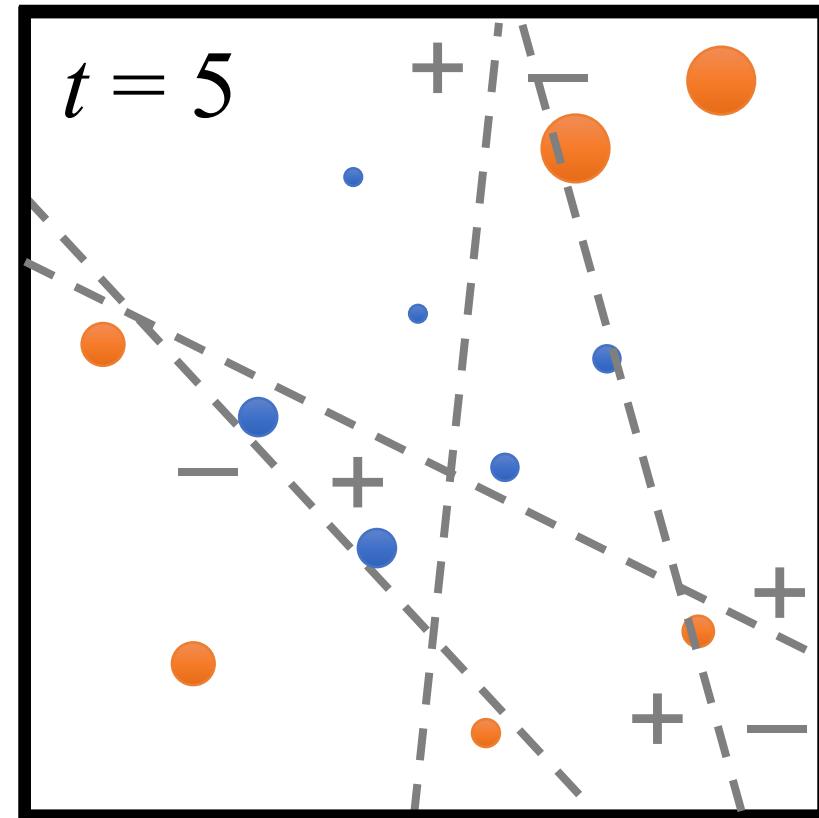
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

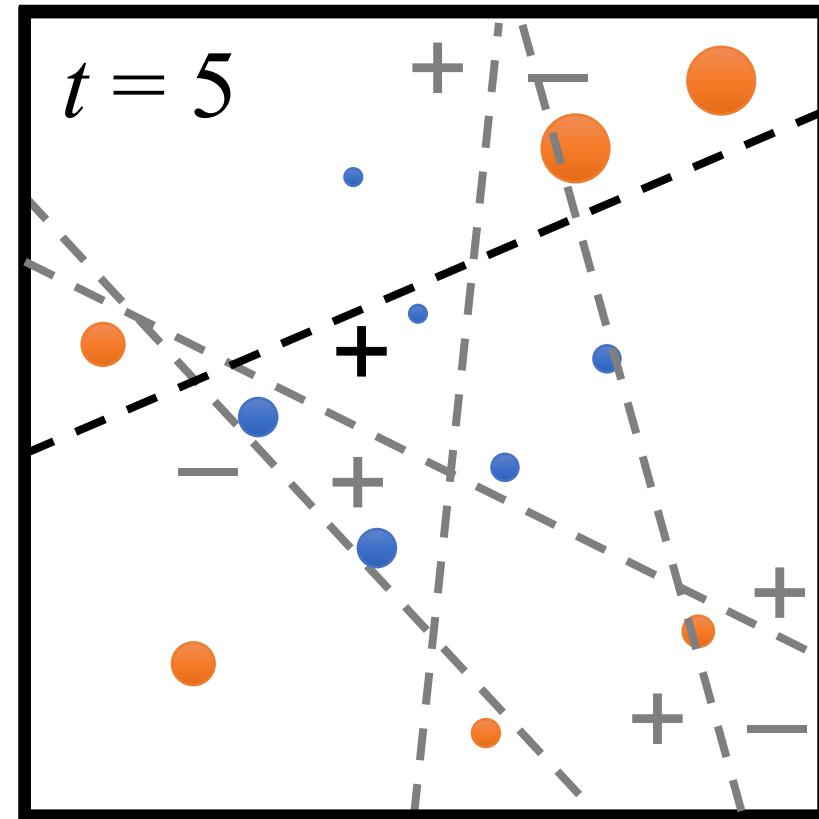
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

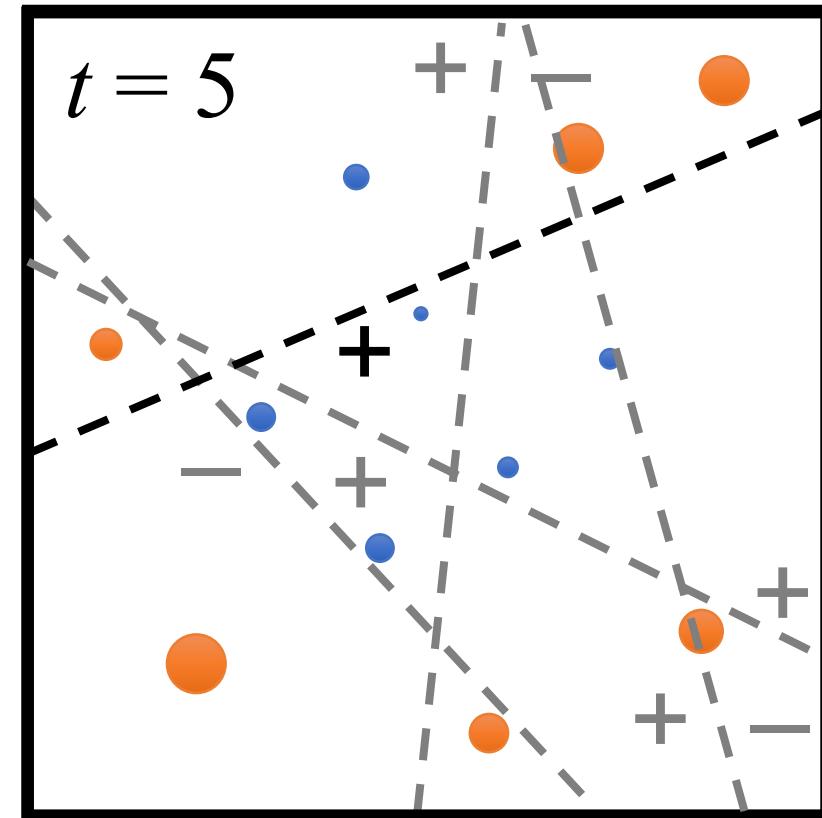
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

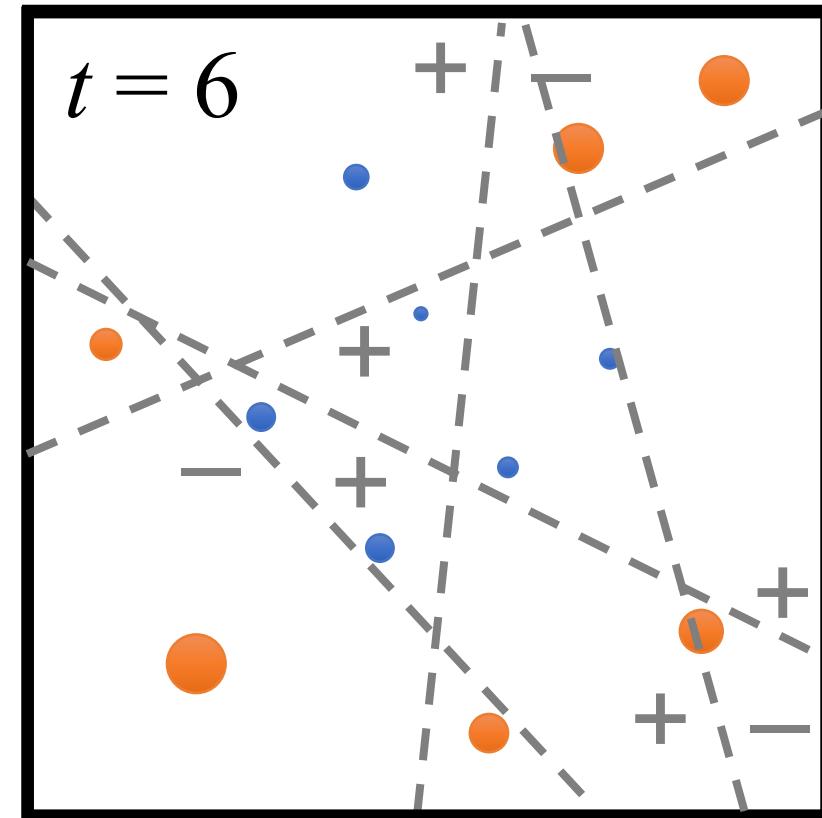
$$H(\mathbf{x}) = \text{sign} \left(\sum_{t=1}^T \beta_t h_t(\mathbf{x}) \right)$$



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
```

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

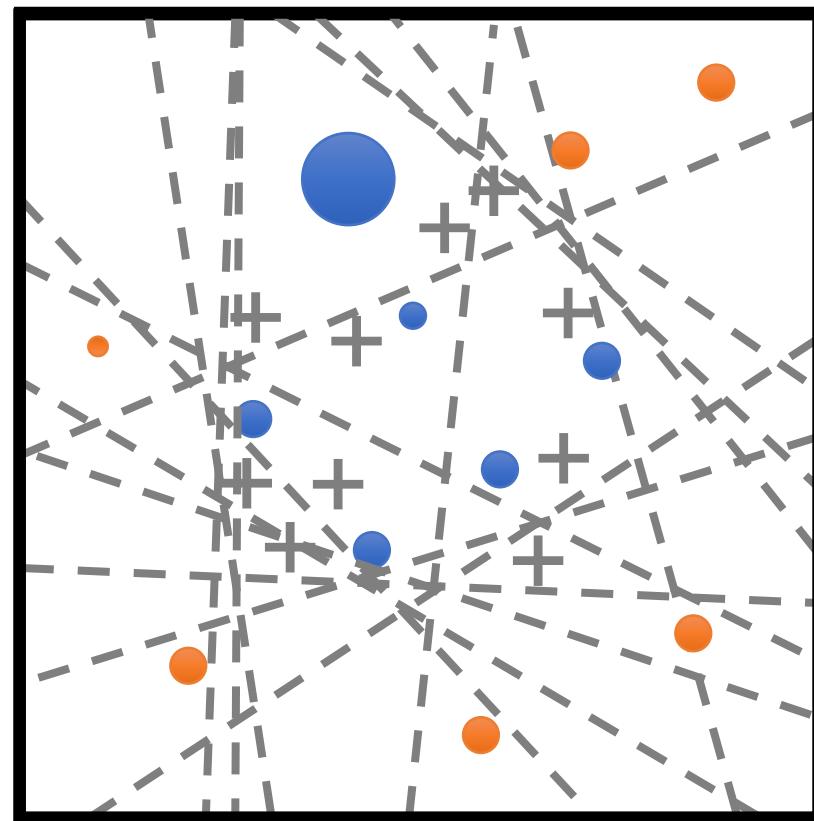


AdaBoost

```
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
```

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

$t = T$

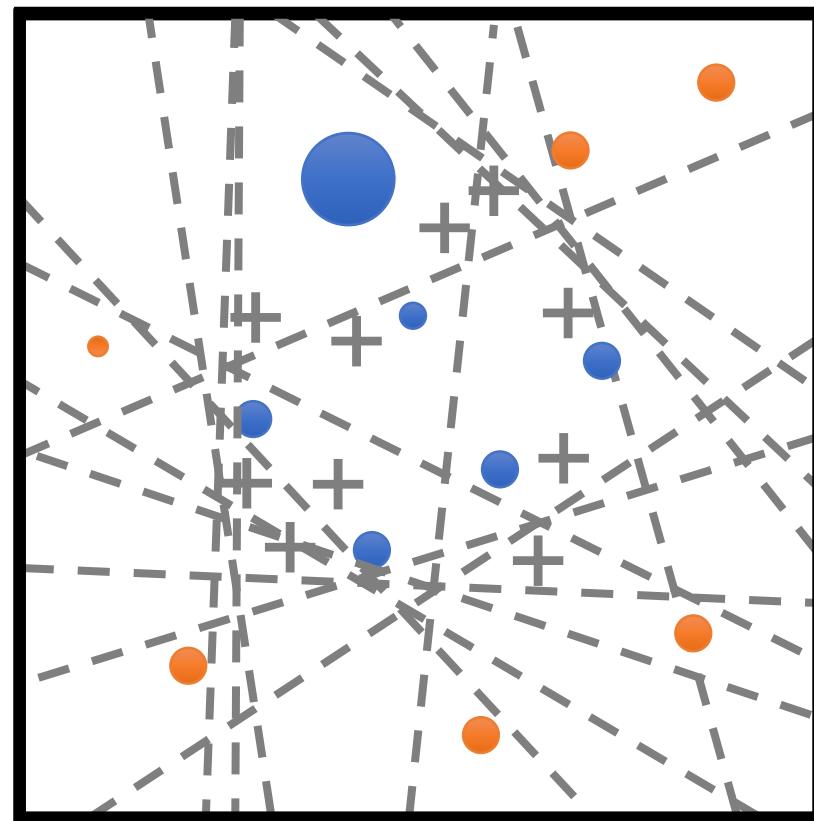


AdaBoost

```
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 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
```

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

$t = T$

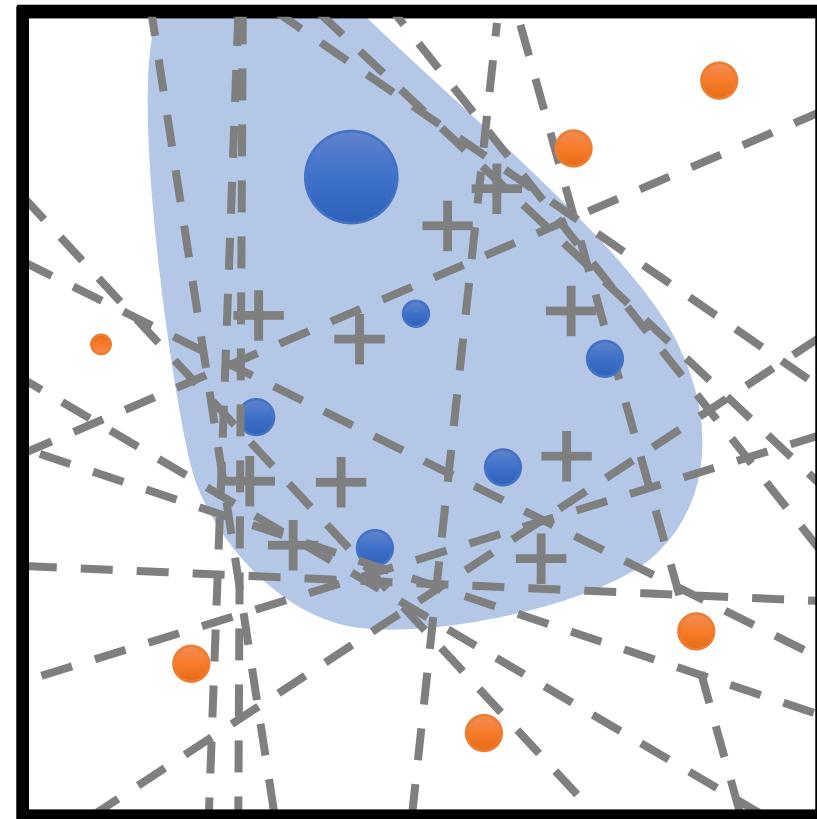


AdaBoost

```
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 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
```

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

$t = T$



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

AdaBoost In Detail

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 :

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

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

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

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

Training a Model with Weighted Instances

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

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

Base Learner Requirements

- AdaBoost works best with “weak” learners
 - Should not be complex – easy and fast to train, more diverse
 - Typically high bias classifiers are used (overfit less)
 - 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) – HW3
 - 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 of 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

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

- 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

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 $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$
the number of iterations T

- 1: Initialize a vector $\mathbf{w}_0 = \mathbf{1}_n$
- 2: **for** $t = 1, \dots, T$ **do**
- 3: Train model $h_t(\mathbf{x})$
- 4: Compute the error ϵ_t

$$\epsilon_t = \sum_{i:y_i \neq h_t(\mathbf{x}_i)} 1$$

- 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 \mathbf{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
 - Can stop adding more members when validation error stops decreasing

Summary

- Ensemble Methods

Combine a diverse set of base learners to create a strong ensemble

- Bagging

Train multiple base learners on bootstrap replicas of the training dataset e.g., Random Forests, which additional uses random features for each trees

- Boosting

Adaboost: Train a sequence of weak learners that fix mistakes of previous learners by instance reweighting