

# Ensemble Methods

**Prof. Mingkui Tan**

SCUT Machine Intelligence Laboratory (SMIL)



# Contents

1 Introduction of Ensemble Learning

2 Decision Tree

3 AdaBoost

4 GBDT (Gradient Boosting Decision Trees)

# Contents

1 Introduction of Ensemble Learning

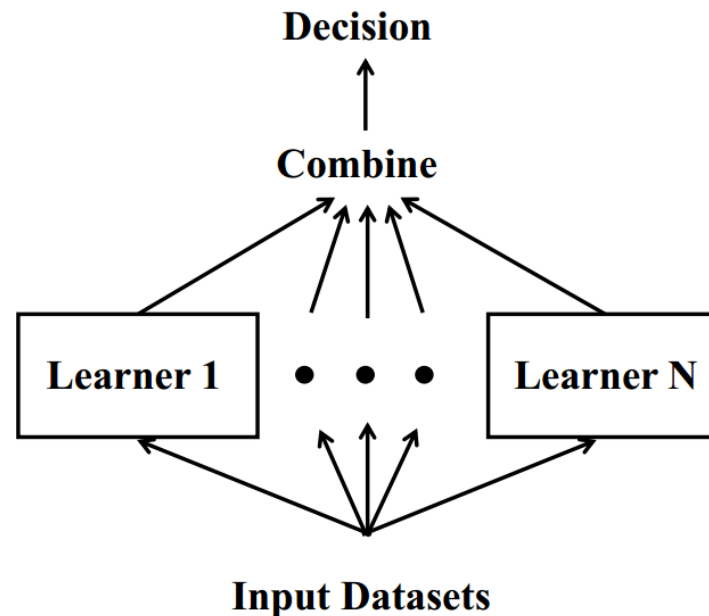
2 Decision Tree

3 AdaBoost

4 GBDT (Gradient Boosting Decision Trees)

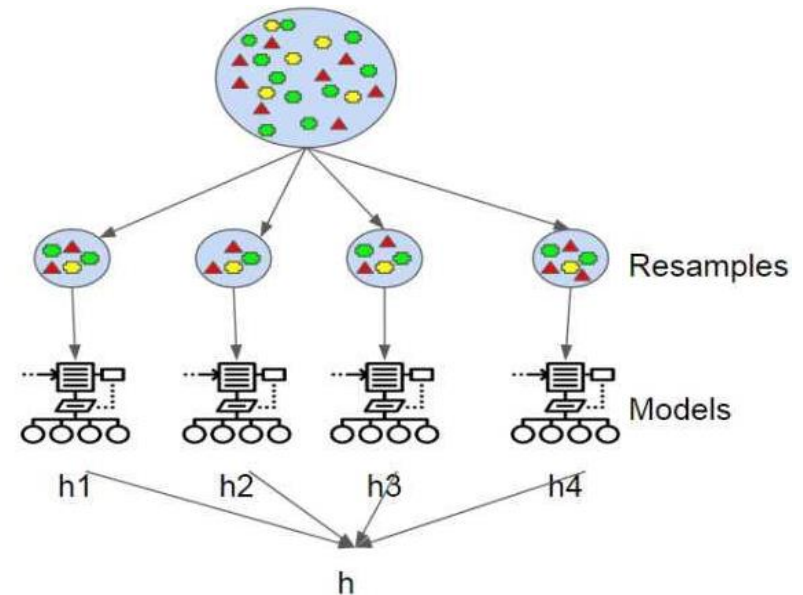
# Ensemble Learning

- Ensemble learning: combine several base models to produce a better predictive model
- Main methods: Bagging, Boosting



# Bagging

- Get  $T$  sampling sets through bootstrap sampling
- Train  $T$  base learners through the sampling sets respectively



- For classification:

The class with the most votes becomes the final class

- For regression:

The final output is the average output of every base learner

# Random Forest

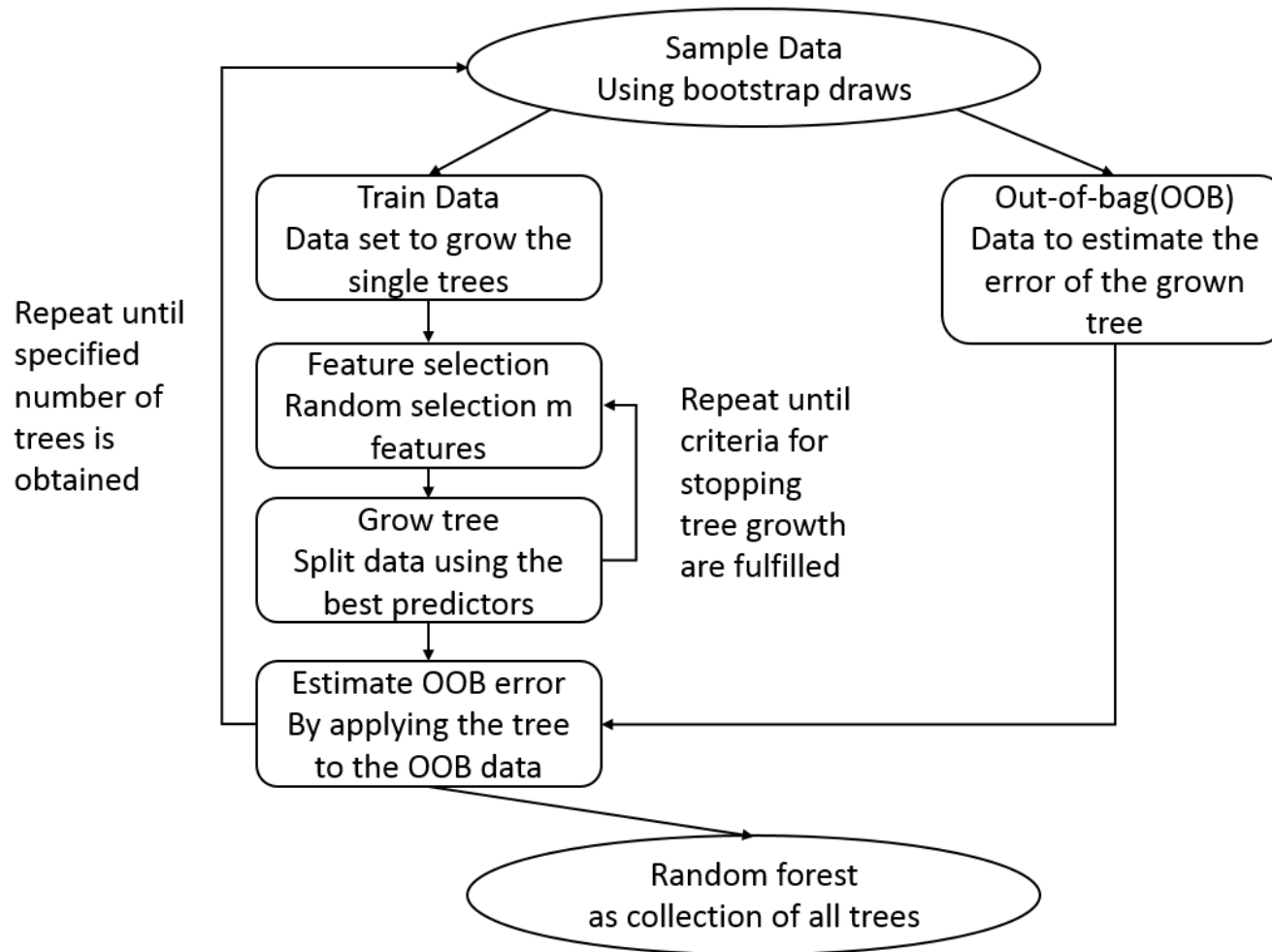
- Random forest is an extension of bagging using decision tree as base learner
- Randomly select  $m$  out of  $p$  features to get the optimal partition feature

## Comparison between bagging and random forest

- The training efficiency of random forest is better than bagging
- Bagging uses decision tree with **definite structure**
- Random forest uses decision tree with **random structure**

# Random Forest

- An example of the process flow is depicted below:



# Contents

1 Introduction of Ensemble Learning

2 Decision Tree

3 AdaBoost

4 GBDT(Gradient Boosting Decision Trees)



# Decision Tree Example

## ■ Learning the ‘Play Tennis Decision Tree’

4 Attributes

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

# Decision Tree Example

Play Tennis: Yes or No?

- Learned function is a tree
- Each branch of the tree represents a possible decision, outcome, or reaction
- The farthest branches (leaf nodes) on the tree represent the end results

# Algorithm: Decision Tree

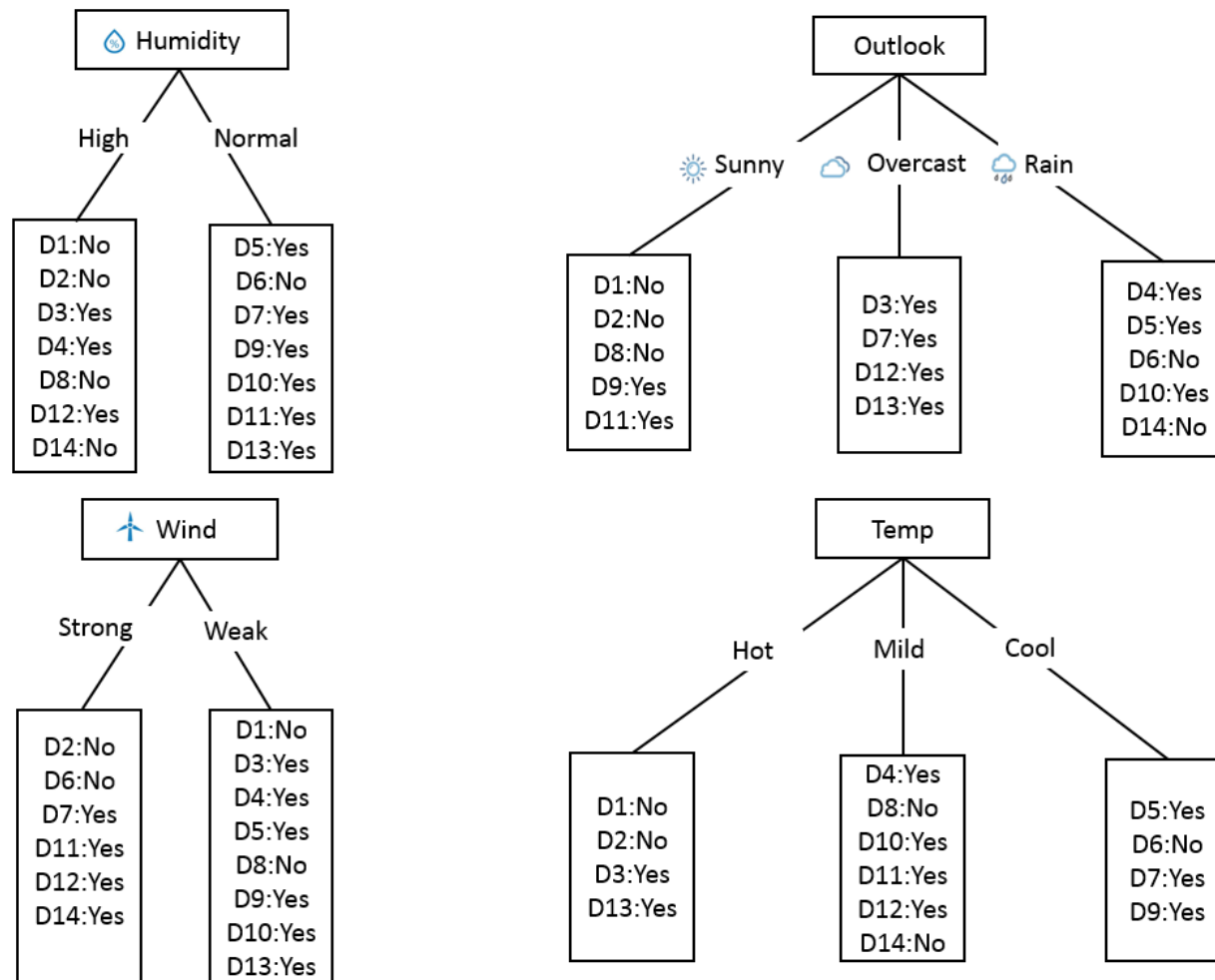
**Input:** Training set  $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ ; **Attribute set**  $A = \{a_1, a_2, \dots, a_d\}$ .

**Procedure:** Function TreeGenerate( $D, A$ ). **Output:** A decision tree.

---

```
1  Generate node;
2  if all the samples in  $D$  belong to class  $C$  then
3    |   mark this node as class  $C$  leaf node; return
4  end
5  if  $A = \emptyset$  OR samples in  $D$  have the same value on  $A$  then
6    |   mark this node as leaf node, and the class should be the most frequent
        |   occurrence class; return
7  end
8  Select the best partition attribute  $a_*$  from  $A$ ;
9  for each value  $a_*^v$  of attribute  $a_*$  do
10   |    $D_v$  is the sample subset of  $D$  with  $a_* = a_*^v$ ;
11   |   if  $D_v = \emptyset$  then
12     |   |   mark this node as the leaf node, and the class should be the most frequent
        |   |   occurrence class; return
13   |   else
14     |   |   generate a branch for this node, TreeGenerate( $D_v, A \setminus \{a_*\}$ )
15   |   end
16 end
```

# Decision Tree Example



Which is the best partition attribute?

# Decision Tree Example

An attribute is good when:

- For one value we get all instances as positive
- For other value we get all instances as negative

An attribute is poor when:

- It provides no discrimination
- Attribute is immaterial to the decision
- Same number of positive and negative instances for each value

# Measure of Homogeneity of Examples

- **Entropy** characterizes the (im)purity of an arbitrary collection of examples
- Given a collection  $D$  of positive and negative examples
- **Entropy of  $D$**  relative to boolean classification is

$$\text{Entropy}(D) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

where  $p_+$  is proportion of positive examples

$p_-$  is proportion of negative examples

# Measure Homogeneity of Examples

## Illustration:

- $D$  is a collection of 14 examples with 9 positive and 5 negative examples

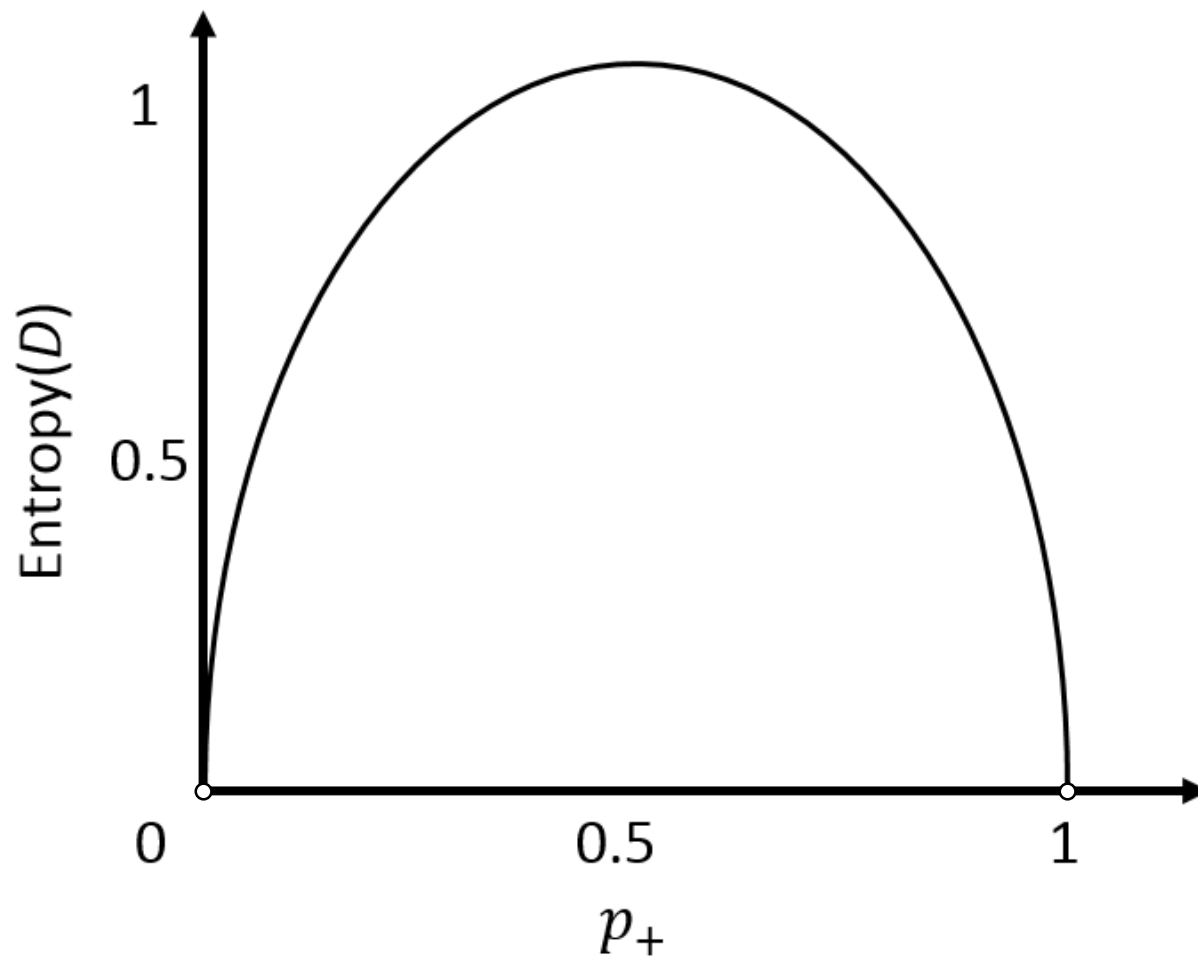
- Entropy of  $D$  relative to the boolean classification:

$$\text{Entropy}(9 +, 5 -) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940$$

- Entropy is zero if all members of  $D$  belong to the same class

# Measure Homogeneity of Examples

Entropy Function Relative to a Boolean Classification





# Entropy for Multi-valued Target function

- If the target attribute can take on  $c$  different values, the entropy of  $D$  relative to this  $c$ -wise classification is:

$$\text{Entropy}(D) = \sum_{i=1}^c -p_i \log_2 p_i$$

# Information Gain

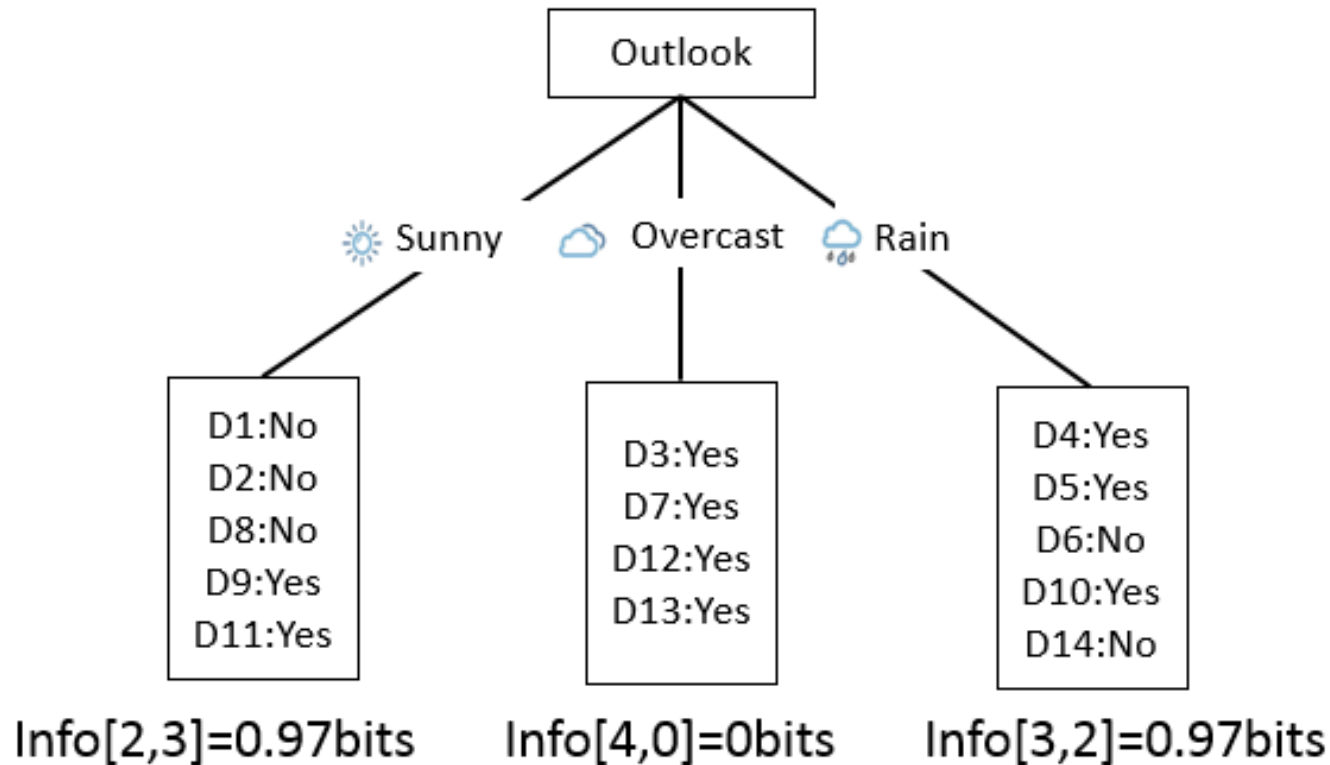
- Entropy measures the (im)purity of a collection
- Information gain of attribute  $A$  is the reduction in entropy caused by partitioning the set of examples  $D$

$$\text{Gain}(D, A) \equiv \text{Entropy}(D) - \sum_{v \in \text{Values}(A)} \frac{|D_v|}{|D|} \text{Entropy}(D_v)$$

where  $\text{Values}(A)$  is the set of all possible values for attribute  $A$

$D_v$  is the subset of  $D$  for which attribute  $A$  has value  $v$

# Measure of Purity: Information

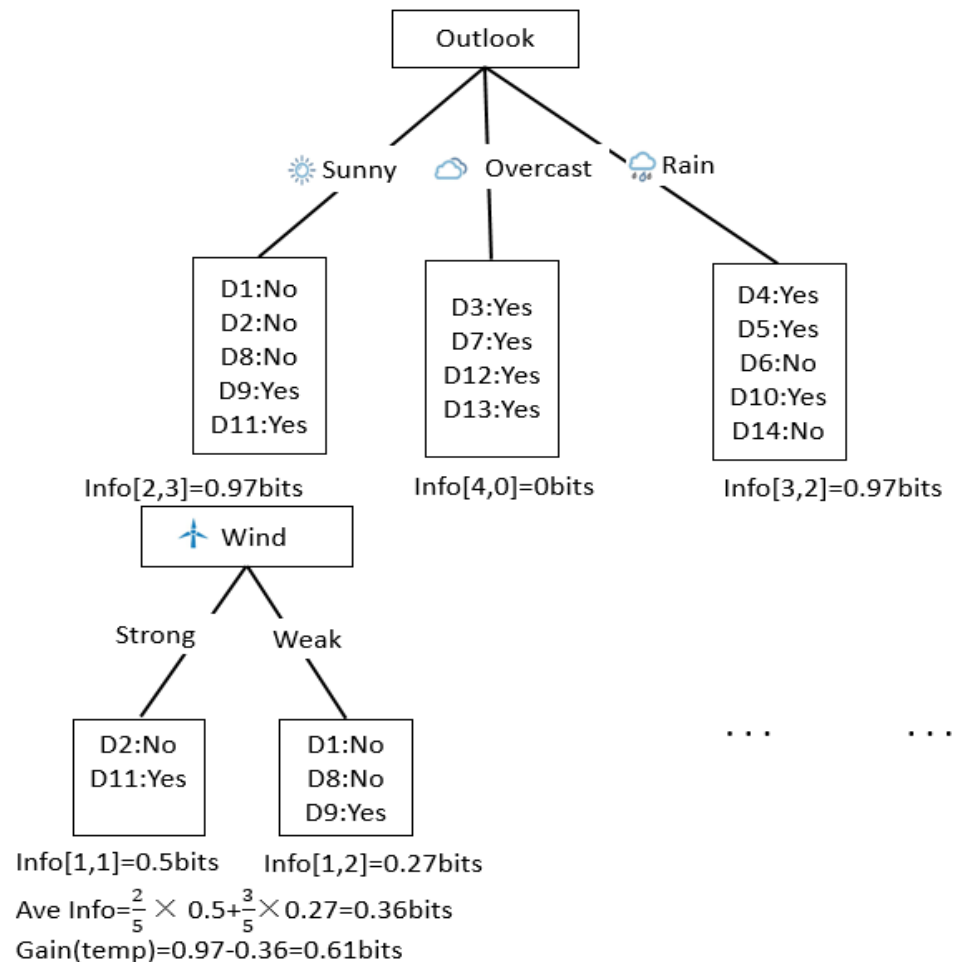


$$\text{Info}[2,3] = \text{entropy}(2,3) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.97\text{bits}$$

# Information Gain for Each Attribute

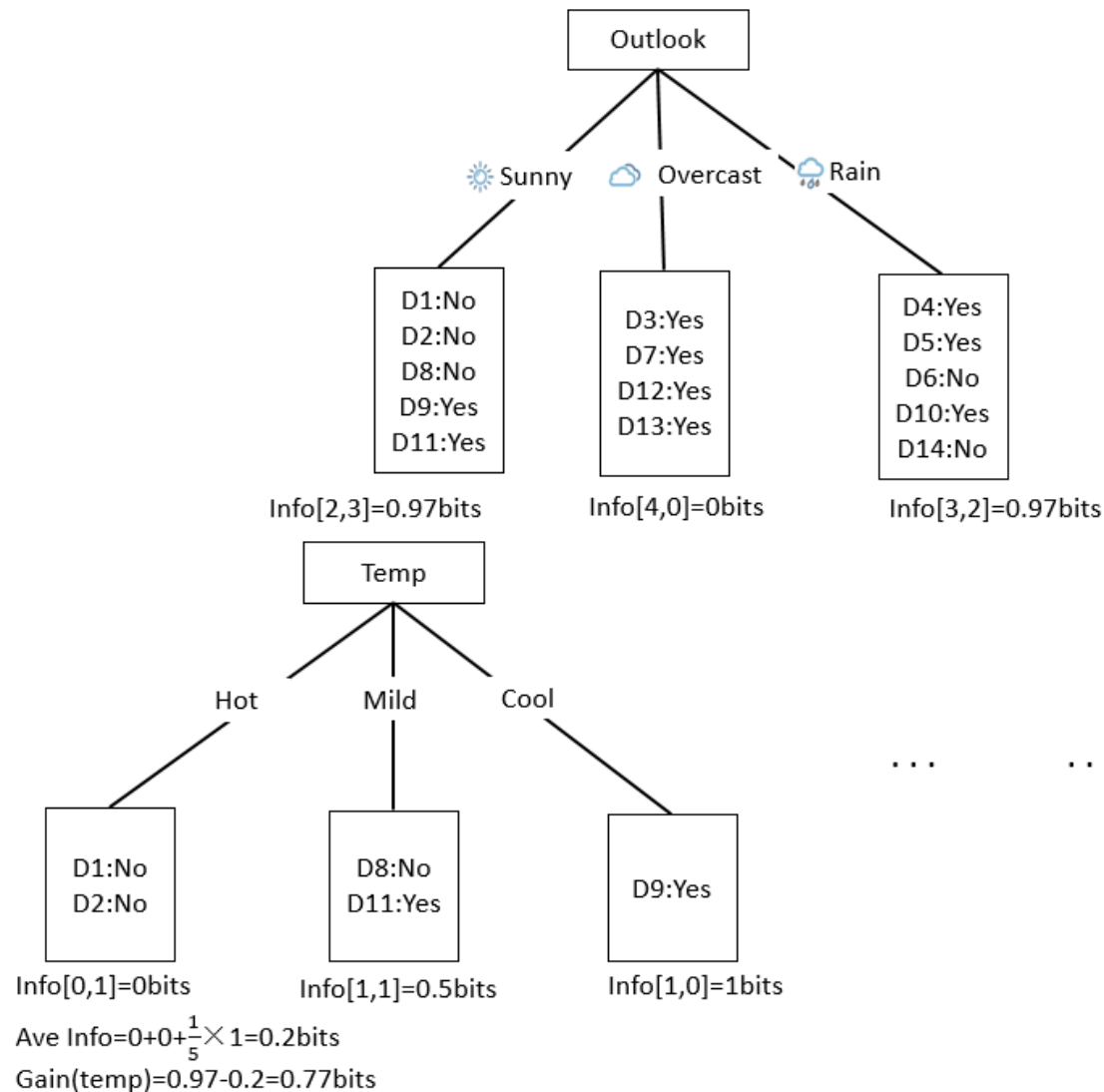
- $\text{Gain}(\text{outlook}) = 0.94 - 0.693 = 0.247$
- $\text{Gain}(\text{temperature}) = 0.94 - 0.911 = 0.029$
- $\text{Gain}(\text{humidity}) = 0.94 - 0.788 = 0.152$
- $\text{Gain}(\text{windy}) = 0.94 - 0.892 = 0.048$
- $\text{argmax}_A\{0.247, 0.029, 0.152, 0.048\} = \text{outlook}$
- Select outlook as the splitting attribute of tree

# Expanded Tree: Play Tennis for Outlook = Sunny

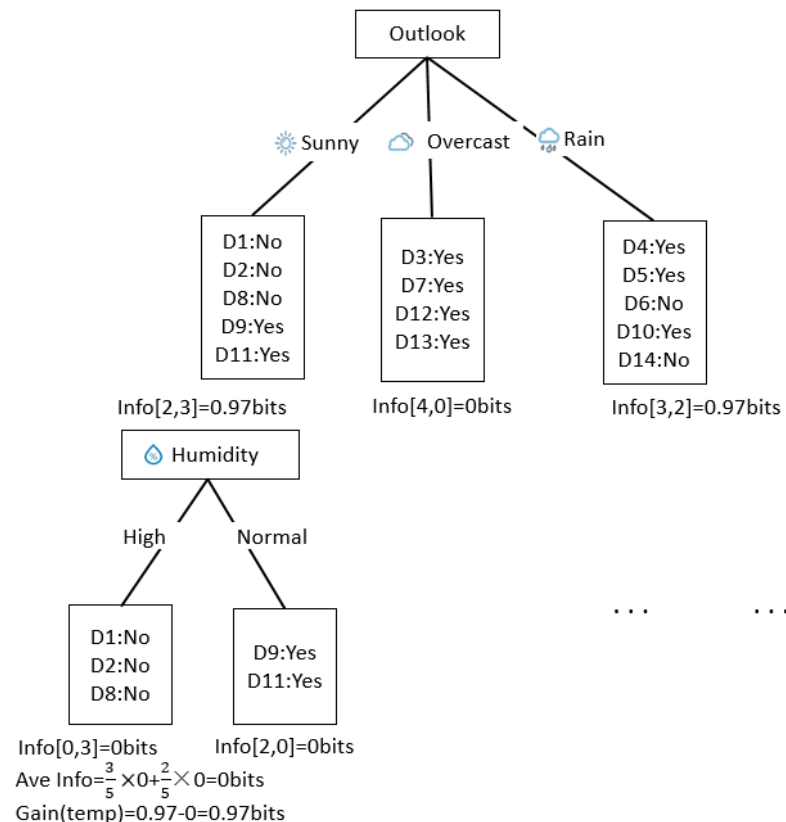


- Where *temp* contains wind, humidity and temperature

# Expanded Tree: Play Tennis for Outlook = Sunny

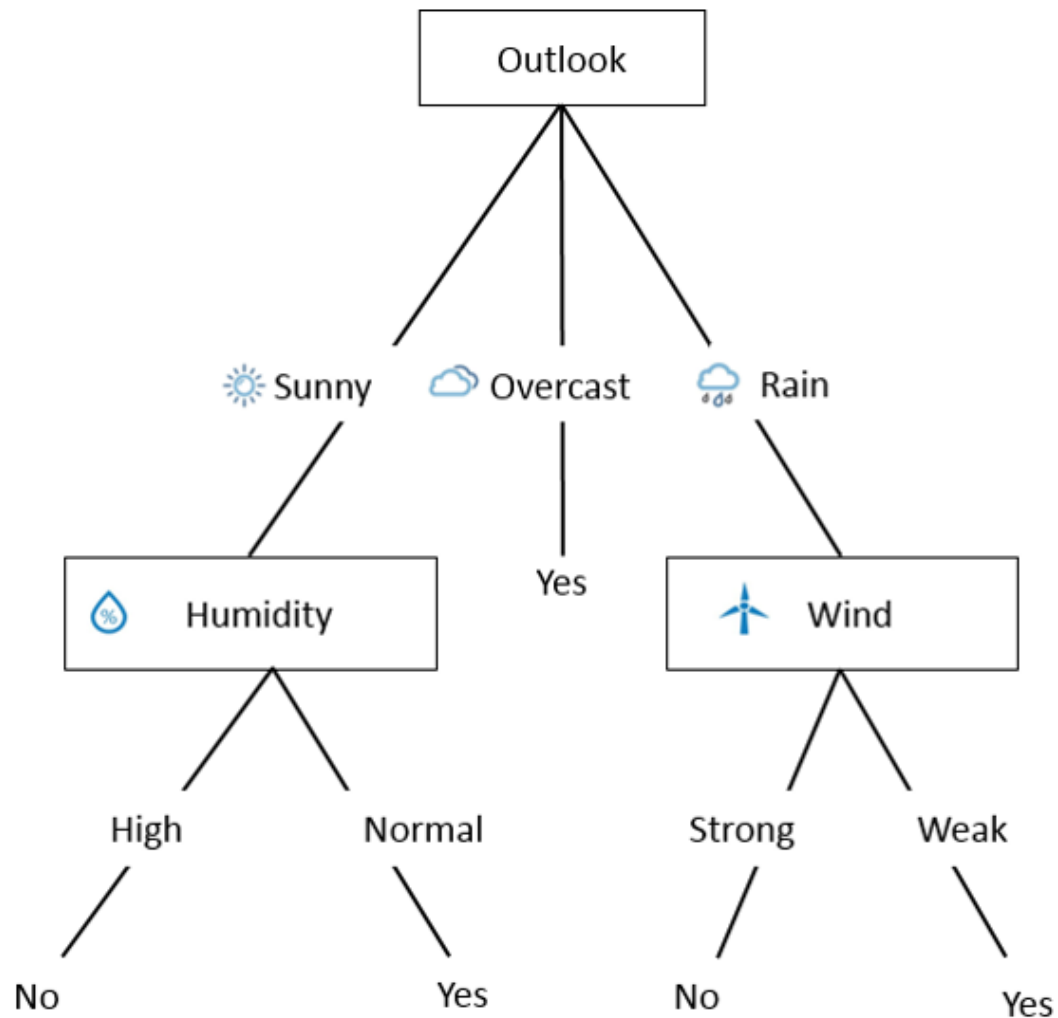


# Expanded Tree: Play Tennis for Outlook = Sunny



- Since  $\text{Gain}(\text{humidity})$  is the highest, select humidity as splitting attribute
- No need to split further

# Decision Tree for Weather Data





# Contents

1 Introduction of Ensemble Learning

2 Decision Tree

3 AdaBoost

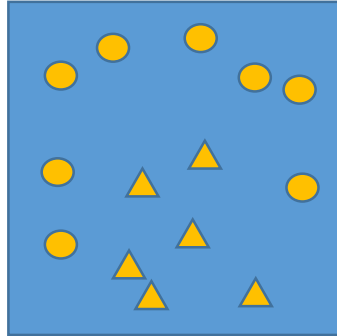
4 GBDT (Gradient Boosting Decision Trees)

# AdaBoost: What is AdaBoost?

- Stands for Adaptive Boosting
- Combines base learners linearly
- Iteratively adapts to the errors made by base learners in previous iterations
- Re-weighting scheme:
  - Higher weight assigned to incorrectly classified data points
  - Lower weight assigned to correctly classified data points

# AdaBoost: Initialization of the algorithm

Training data



- Assigning same weight  $w_1(i)$  to all data points

$$w_1(i) = \frac{1}{n}, i = 1, 2, \dots, n$$

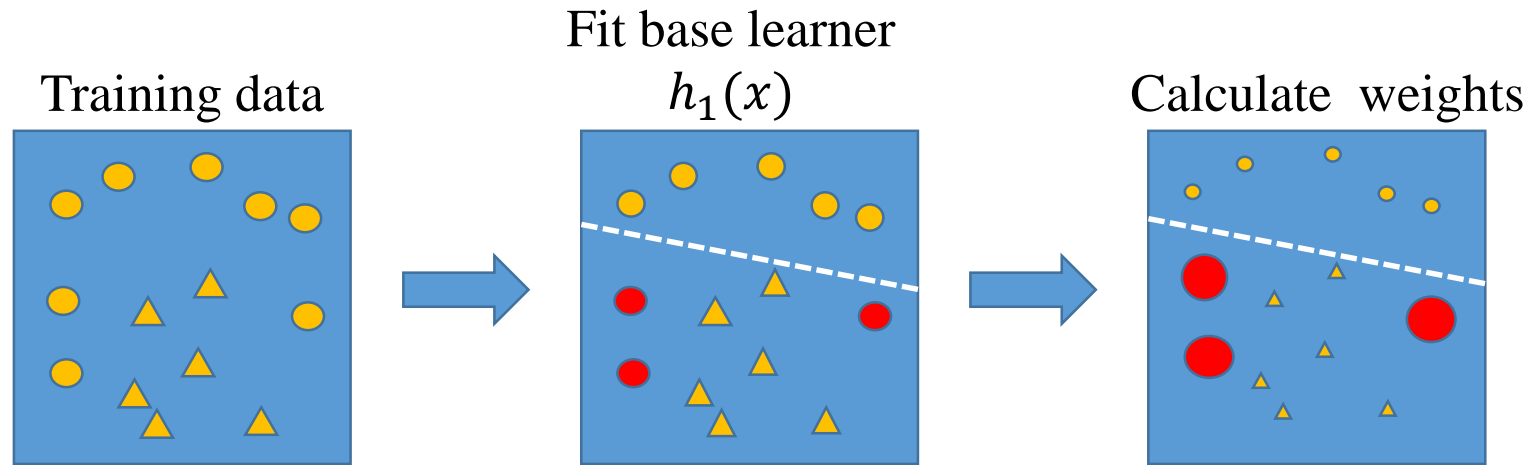
where  $n$  is the number of data points

# AdaBoost: Iterations of Algorithm

For  $t = 1, \dots, T$  or until low enough error is achieved:

- Fit base learner  $h_t(x)$  to data points
- Calculate the classification error rate  $e_t$  of  $h_t(x)$
- Calculate the weight  $\alpha_t$  of  $h_t(x)$
- Update the weights of each data point

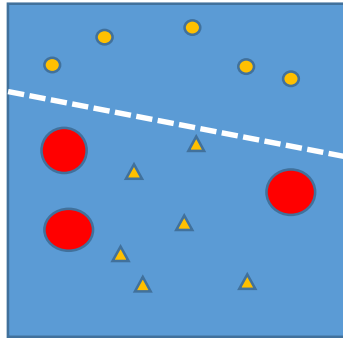
# AdaBoost: Iteration 1



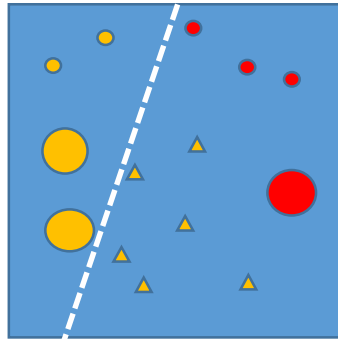
- Initially all data points have the same weight  $1/n$
- The class correctly classified will be given less weights in the next iteration, and higher weights for misclassified class

# AdaBoost: Iteration 2

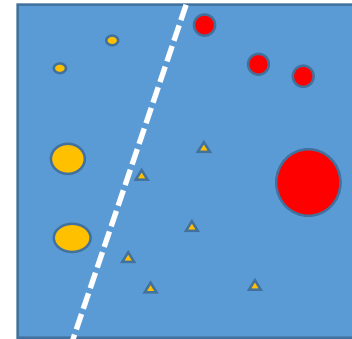
Weights after 1<sup>st</sup>  
iteration



Fit base learner  
 $h_2(x)$



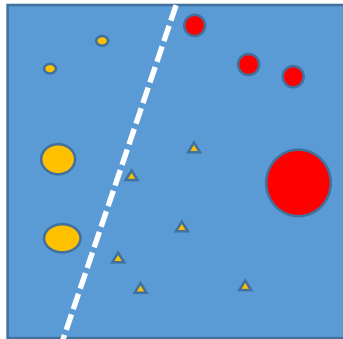
Calculate weights



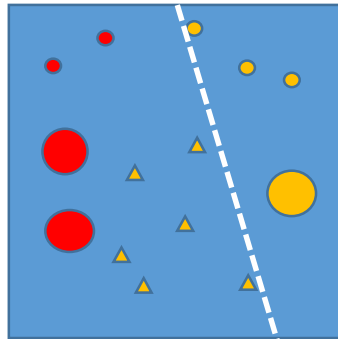
- Base learner forms a decision boundary which classifies the data points better with higher weights

# AdaBoost: Iteration 3

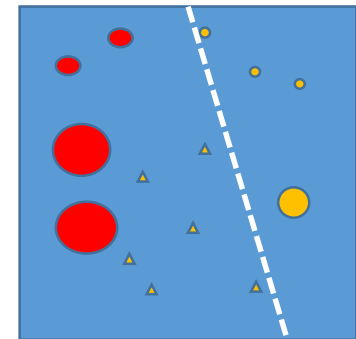
Weights after 2<sup>nd</sup>  
iteration



Fit base learner  
 $h_3(x)$



Calculate weights

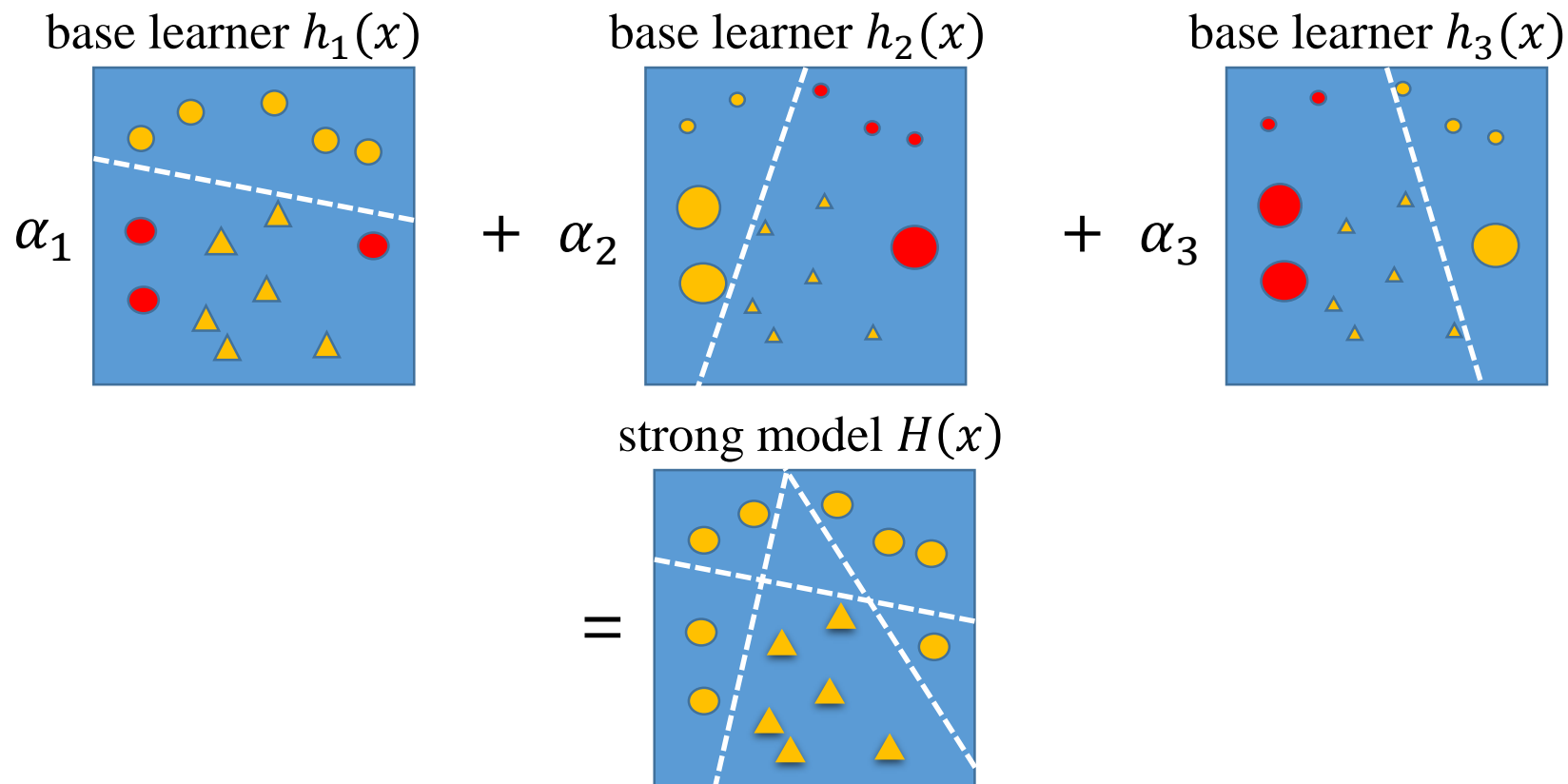


...continue iterating until either:

- Sufficiently low training error is achieved (with enough iterations, the algorithm can reach 100% accuracy)
- A pre-defined number of base learners was added

# AdaBoost: Final Model

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





# AdaBoost: More Details

- Fit base learner  $h_t(x)$  to data points, we define base classifier  $h_t(x)$  as

$$h_t(x) \in \{-1, +1\}$$

- Calculate the classification error rate  $e_t$  of  $h_t(x)$  on the training data set

$$e_t = p(h_t(x) \neq y_i) = \sum_{i=1}^n w_t(i) \mathbb{I}(h_t(x_i) \neq y_i)$$

For binary classification case,  $\mathbb{I}(X = x_i) = \begin{cases} 1, & h_t(x_i) \neq y_i \\ 0, & h_t(x_i) = y_i \end{cases}$

# AdaBoost: More Details

- Calculate the weight  $\alpha_t$  of  $h_t(x)$  in the final classifier

$$\alpha_t = \frac{1}{2} \ln \frac{1 - e_t}{e_t}$$

Here, when  $e_t \leq 0.5$ ,  $\alpha_t \geq 0$ , with the decreasing of  $e_t$ ,  $\alpha_t$  will be larger

Note: classifier with smaller error rate would be more important

# AdaBoost: More Details

## ■ Update the weights

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} e^{-\alpha_t y_i h_t(x_i)}$$

Here,  $z_t = \sum_{i=1}^n w_t(i) e^{-\alpha_t y_i h_t(x_i)}$  is normalization term,  
and makes  $w_t(i)$  a probability distribution

To simplify,  $w_{t+1}(i) = \begin{cases} \frac{w_t(i)}{z_t} e^{-\alpha_t}, & \text{for right predictive sample} \\ \frac{w_t(i)}{z_t} e^{\alpha_t}, & \text{for wrong predictive sample} \end{cases}$

# AdaBoost: More Details

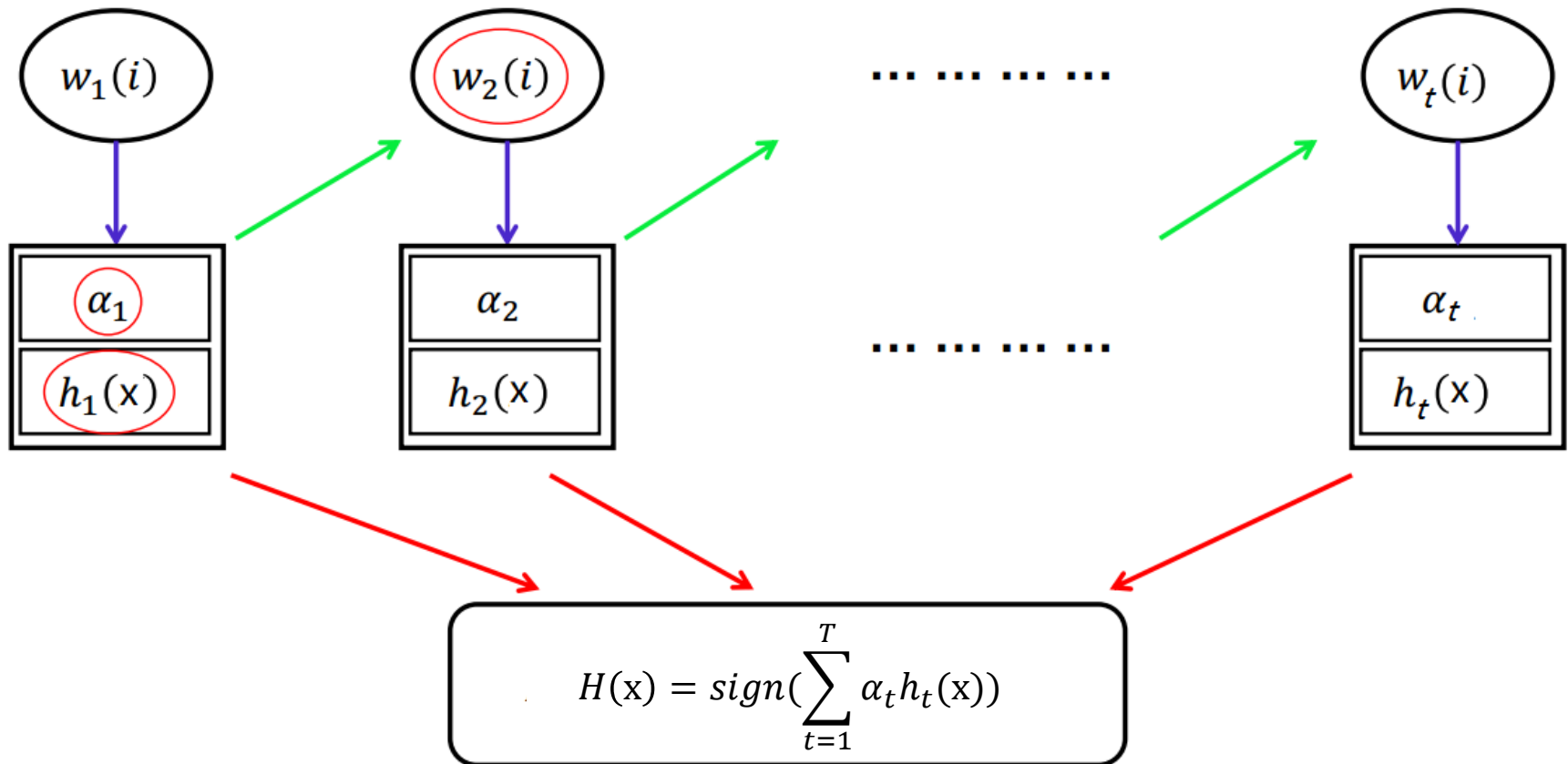
- Ensemble learner

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

Note:  $h_t(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$  is a nonlinear function, so AdaBoost can deal with nonlinear problem

# AdaBoost: More Details

Every iteration generates a new base learner  $h_t(x)$  with weight  $\alpha_t$



# AdaBoost: Example

Given the following training samples:

Index	1	2	3	4	5	6	7	8	9	10
$x$	0	1	2	3	4	5	6	7	8	9
$y$	-1	-1	-1	1	1	1	-1	-1	-1	1

- Initialize the same weights for data points at the beginning,  $w_1(1), w_1(2), \dots, w_1(10) = 0.1$
- Assume the threshold of  $x$  is 2.5, when  $x \leq 2.5, h_1(x) = -1$ , when  $x > 2.5, h_1(x) = 1$
- So the right samples are Index 1, 2, 3, 4, 5, 6, 10, and the wrong samples are Index 7, 8, 9 (three data points)
- The error rate  $e_1 = 3 * 0.1 = 0.3$
- For all samples, the threshold is chosen to be 2.5, which has the lowest error rate  $e_1$

# AdaBoost: Example

- Now, the first base classifier is

$$h_1(x) = \begin{cases} 1, x \leq 2.5 \\ -1, x > 2.5 \end{cases}$$

- Then, for  $h_1(x)$ ,  $\alpha_1$  is

$$\alpha_1 = \frac{1}{2} \ln \frac{1 - e_1}{e_1} = 0.424$$

- For the first round, the weight of  $h_1(x)$  is 0.424, it is used to update the weights of data points of the next round

$$w_{t+1}(i) = \frac{w_t(i)}{Z_t} e^{-\alpha_t y_i h_t(x_i)}$$

- $w_2(i), i = 1, \dots, 10$  are as shown in the table below:

index	1	2	3	4	5	6	7	8	9	10
$x$	0	1	2	3	4	5	6	7	8	9
$y$	-1	-1	-1	1	1	1	-1	-1	-1	1
$w_2$	0.0714	0.0714	0.0714	0.0714	0.0714	0.0714	0.1667	0.1667	0.1667	0.0714

# AdaBoost: Why Use AdaBoost?

- Needs only a simple classifier as a base learner
- Can achieve prediction similar to powerful classifiers
- Can combine with any learning algorithms
- Requires little parameter tuning
- Extended to problems beyond binary classification



# AdaBoost: Summary

- AdaBoost (i.e. Adaptive boosting) is one of the most popular and powerful ensemble methods
- AdaBoost focuses on the data points that are erroneous
- Simple to implement and depend on the base learner
- However, vulnerable to noisy data

# Contents

1 Introduction of Ensemble Learning

2 Decision Tree

3 AdaBoost

4 GBDT (Gradient Boosting Decision Trees)

# Gradient Boosting Decision Trees

Gradient Boosting Decision Trees (GBDT) is a decision tree algorithm with iteration

Example: What is the difference between regression tree and GBDT?

Suppose: There are 4 people A, B, C and D, whose age are 14, 16, 24, 26 respectively

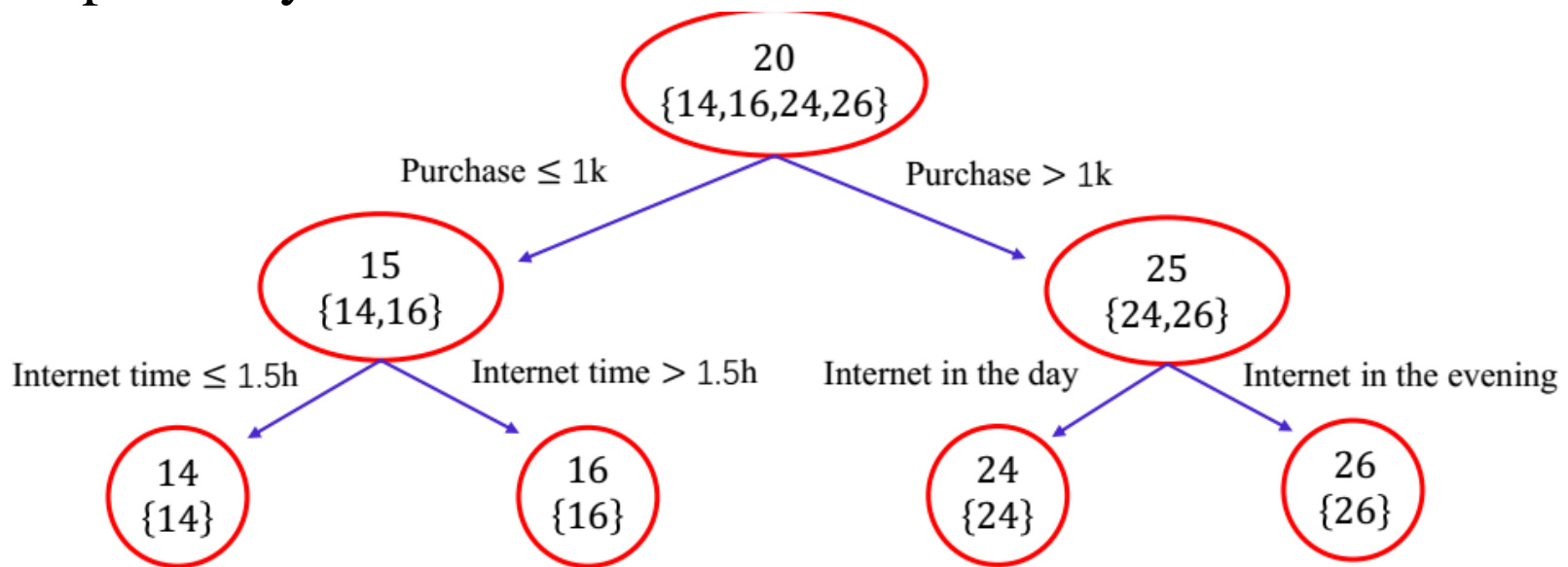
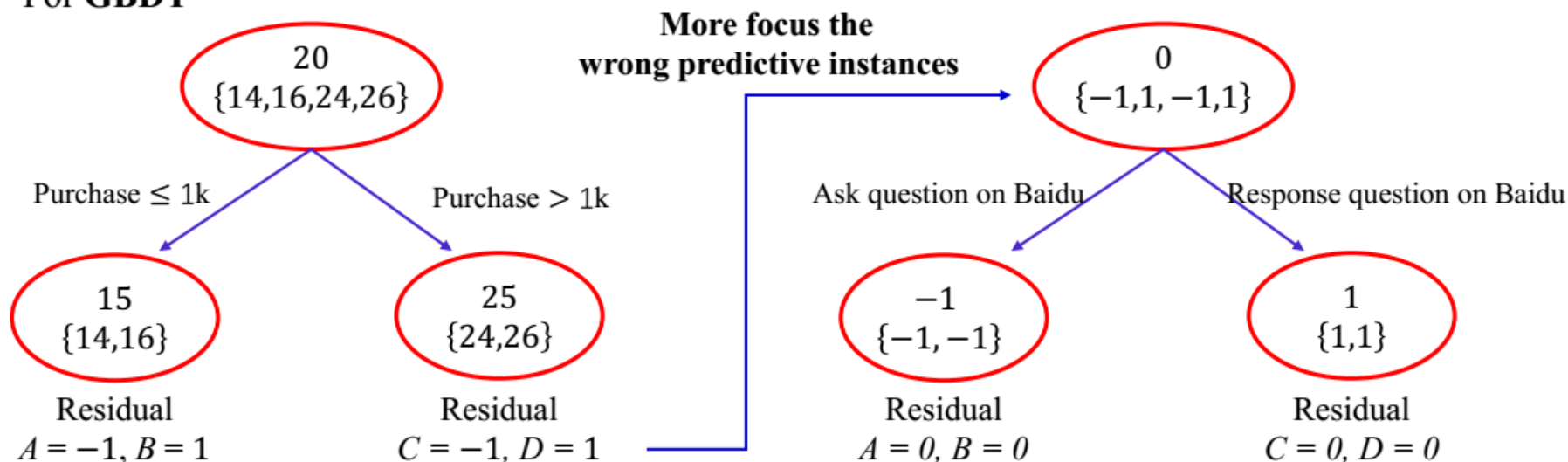


Figure: Single regression tree

# GBDT

- The key of GBDT is that trees learn all the results and residuals of all trees before
- The residual is the difference of predictive value and real value, so the predictive value is the sum of all results of trees

For **GBDT**



- So,  $A=15+(-1)=14$ ,  $B=15+1=16$ ,  $C=25+(-1)=24$ ,  $D=25+1=26$

# Gradient Boosting Decision Trees: Questions

## Q1: Why do we need GBDT?

- The motivation of this algorithm
  - Every calculation of residual is to increase the weight of wrong predictive samples
  - The residual of right predictive sample is zero
- In the next iteration, the model can concentratively address these wrong predictive samples
- Another function is to prevent overfitting

# Gradient Boosting Decision Trees: Questions

Q2: Where does this algorithm reflect gradient boosting?

- Residual is the gradient descent direction, which is the derivation of mean square error (MSE)
- MSE is the loss function of CART regression tree

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$ y_i - f(x_i) $	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $ y_i - f(x_i)  \leq \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $ y_i - f(x_i)  > \delta_m$ where $\delta_m = \alpha \text{th-quantile}\{ y_i - f(x_i) \}$
Classification	Deviance	$k$ th component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

# Algorithm: GBDT

**Input:**  $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , where  $x_i \in \mathcal{X}, y_i \in \{-1, 1\}$

**Initialize:**  $f_0(x) = \operatorname{argmin}_{\mu} \sum_{i=1}^n \mathcal{L}(y_i, \mu)$

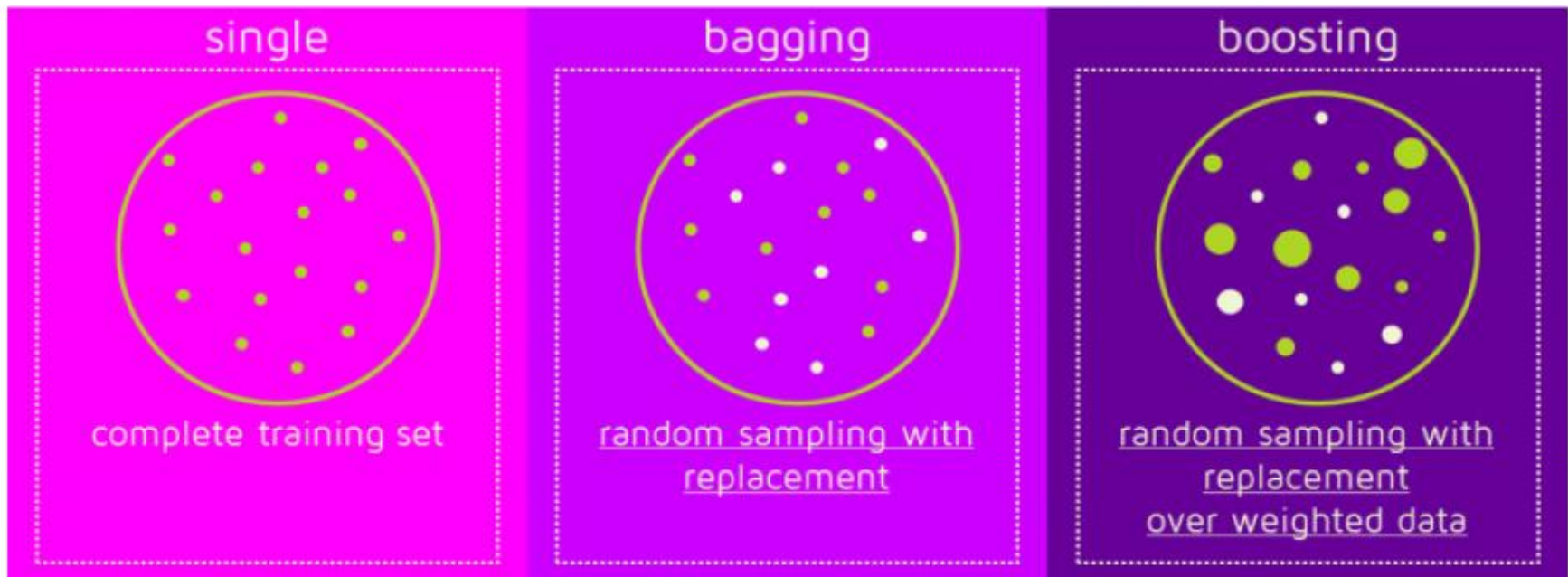
**Output:**  $\hat{f}(x) = f_M(x)$

---

```
1.  for  $m = 1, 2, \dots, M$  do
2.      for  $i = 1, 2, \dots, n$  do
3.           $r_{im} = -\left[\frac{\partial \mathcal{L}(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)}\right]$ 
4.      end
5.      Fit a regression tree to targets  $r_{im}$  giving terminal regions
6.       $R_{jm}, j = 1, 2, \dots, J_m$ 
7.      for  $j = 1, 2, \dots, J_m$  do
8.           $\mu_{jm} = \operatorname{argmin}_{\mu} \sum_{x_i \in R_{jm}} \mathcal{L}(y_i, f_{m-1}(x_i) + \mu)$ 
9.      end
10.     Update  $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \mu_{jm} I(x \in R_{jm})$ 
11. end
```

# Summary

- Instead of training different models on same data, training same model will
  - multiple times on different data sets
  - combine different models
- We can use some simple/weak model as the base model
- How do we get multiple training data sets (in practice, we only have one data set at training time)?





Thank You