

决策树与集成学习

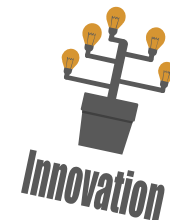
Decision Tree and Ensemble learning

郝 奇

南京大学 天文与空间科学学院

**Application of
Machine Learning
in Astronomy**

机器学习在天文中的应用





Contents

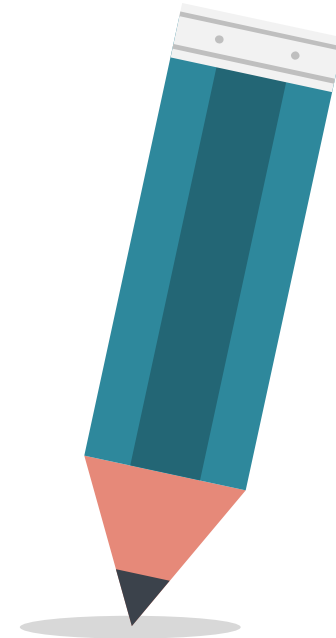
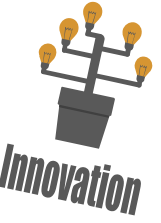


01 Decision Tree

02 Ensemble learning

Decision Tree

01



1. Decision Tree

Recursion / Recursive algorithms

```
def some_fun(x):  
    if x == "":  
        return 0  
    else:  
        return 1 + some_fun(x[1:])
```

What dose this function do ?

1. Decision Tree

Divide-and-Conquer algorithm

```
def quicksort(array):  
    if len(array) < 2:  
        return array  
    else:  
        pivot = array[0]  
        smaller, bigger = [], []  
        for ele in array[1:]:  
            if ele <= pivot:  
                smaller.append(ele)  
            else:  
                bigger.append(ele)  
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```

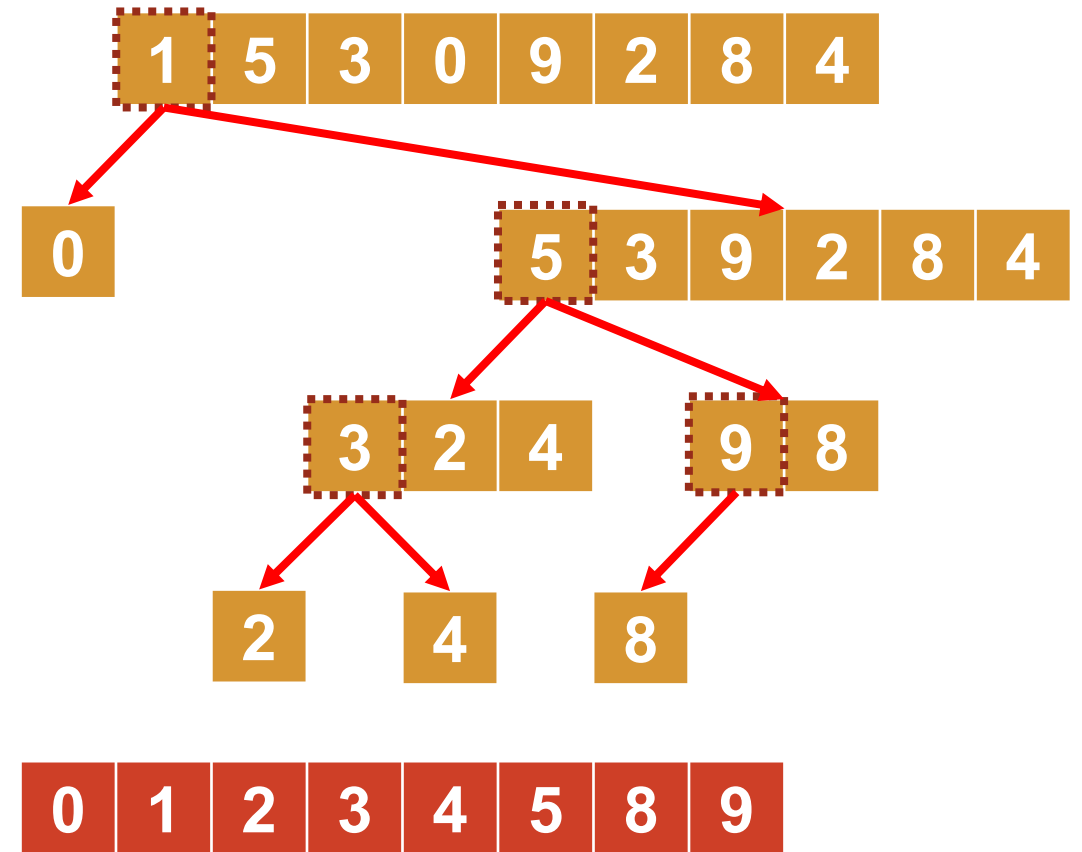
1. Decision Tree

Divide-and-Conquer algorithm

```
def quicksort(array):  
    if len(array) < 2:  
        return array  
    else:  
        pivot = array[0]  
        smaller, bigger = [], []  
        for ele in array[1:]:  
            if ele <= pivot:  
                smaller.append(ele)  
            else:  
                bigger.append(ele)  
        return quicksort(smaller) + [pivot] + quicksort(bigger)
```

Time complexity (average): ?

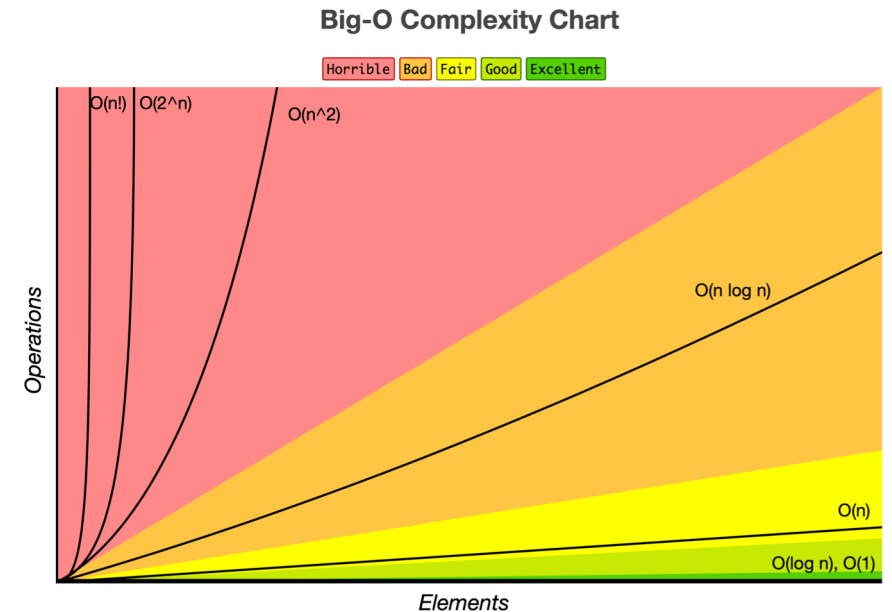
$O(n \log n)$



1. Decision Tree

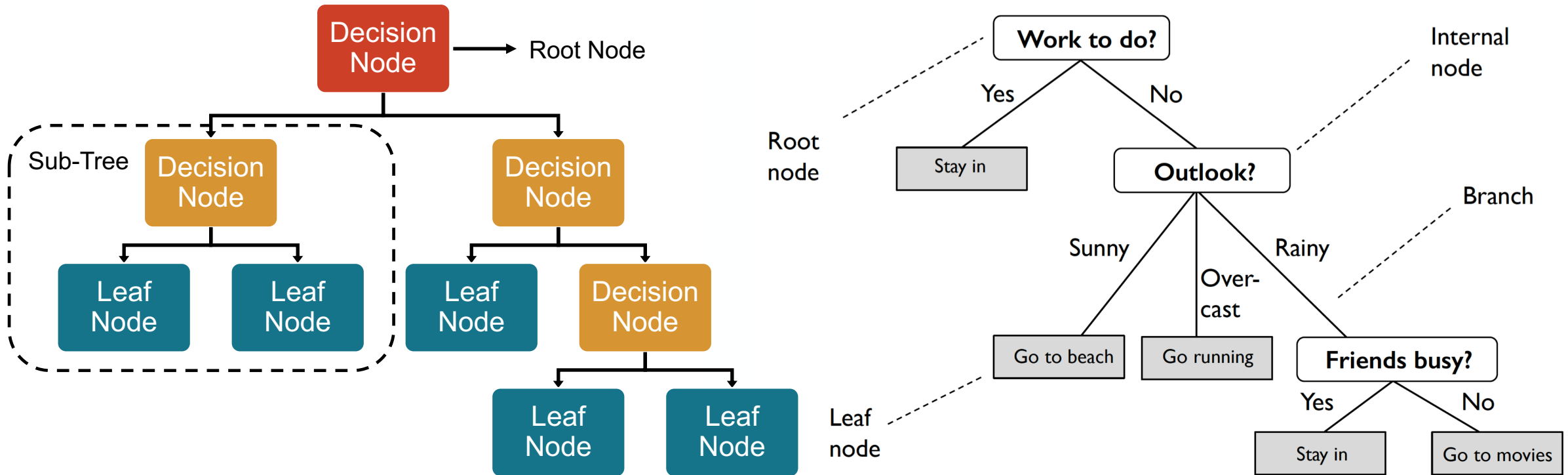
Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
<u>Quicksort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n^2)$	$O(\log(n))$
<u>Mergesort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Timsort</u>	$\Omega(n)$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$
<u>Heapsort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n \log(n))$	$O(1)$
<u>Bubble Sort</u>	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Insertion Sort</u>	$\Omega(n)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Selection Sort</u>	$\Omega(n^2)$	$\theta(n^2)$	$O(n^2)$	$O(1)$
<u>Tree Sort</u>	$\Omega(n \log(n))$	$\theta(n \log(n))$	$O(n^2)$	$O(n)$
<u>Shell Sort</u>	$\Omega(n \log(n))$	$\theta(n(\log(n))^2)$	$O(n(\log(n))^2)$	$O(1)$
<u>Bucket Sort</u>	$\Omega(n+k)$	$\theta(n+k)$	$O(n^2)$	$O(n)$
<u>Radix Sort</u>	$\Omega(nk)$	$\theta(nk)$	$O(nk)$	$O(n+k)$
<u>Counting Sort</u>	$\Omega(n+k)$	$\theta(n+k)$	$O(n+k)$	$O(k)$
<u>Cubesort</u>	$\Omega(n)$	$\theta(n \log(n))$	$O(n \log(n))$	$O(n)$



1. Decision Tree

1.1 Decision tree terminology



- A nested sequence of “if-else” decisions based on the features.
- A class label as a return value at the end of each sequence.

1. Decision Tree

1.1 Decision tree terminology

Decision Tree in Pseudocode

TreeGenerate($\mathcal{D}$):

if $y = 1, \forall \langle X, y \rangle \in \mathcal{D}$ or $y = 0, \forall \langle X, y \rangle \in \mathcal{D}$:

return Tree

else:

Pick best feature x_j :

$\mathcal{D}_0$ at Child₀ : $x_j = 0 \forall \langle X, y \rangle \in \mathcal{D}$

$\mathcal{D}_1$ at Child₁ : $x_j = 1 \forall \langle X, y \rangle \in \mathcal{D}$

return Node(x_j , TreeGenerate($\mathcal{D}_0$), TreeGenerate($\mathcal{D}_1$))

1. Decision Tree

1.1 Decision tree terminology

TreeGenerate(D, T, A)

D : set of training examples;

T : target attribute;

A : set of descriptive attributes;

Create a Root node for the tree.

Decision Tree Algorithm

If T have the same target attribute value t_i

Then Return the single-node tree, (i.e., Root or Leaf node), with target attribute = t_i ;

If $A = \emptyset$ (i.e. there is no descriptive attributes available)

Then Return the single-node tree, (i.e., Root or Leaf node), with most common value of target in T ;

Otherwise

Select attribute a^* from A that **best classify D on splitting measure**

Set a^* the attribute for Root

For each value of a^* , v_i , do

Add a branch below Root, corresponding to $a^* = v_i$;

Let D_v be the subset of D that have $a^* = v_i$;

If $D_v = \emptyset$

Then add a leaf node below the branch with target value = most common value of target in T ;

else

Add the subtree below the branch learned by TreeGenerate($D_v, T, A - \{a^*\}$)

Return (Root)

1. Decision Tree

1.1 Decision tree terminology

■ When to stop ?

- if leaf nodes contain only examples of the same class;
- feature values are all the same for all examples;
- statistical significance test;
- User-defined maximum depth.

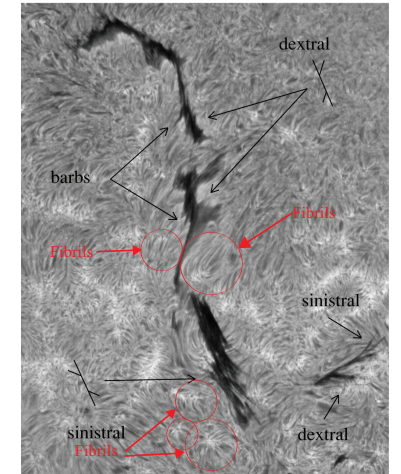
■ How to split ?

- what measurement/criterion as measure of goodness;
- binary vs multi-category split;

1. Decision Tree

1.2 Example: simple training data set

ID	Grayscale (Intensity)	Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament
1	gray	high	northern hemisphere	small	many	Yes
2	dark gray	high	northern hemisphere	medium	many	Yes
3	dark gray	high	northern hemisphere	medium	many	Yes
4	gray	high	southern hemisphere	medium	many	Yes
5	light gray	middle	southern hemisphere	small	some	Yes
6	gray	middle	southern hemisphere	medium	some	yes
7	dark gray	middle	southern hemisphere	medium	some	No
8	dark gray	middle	northern hemisphere	large	few	No
9	light gray	middle	northern hemisphere	medium	few	No
10	gray	high	southern hemisphere	medium	some	No
11	light gray	low	northern hemisphere	medium	few	No
12	dark gray	middle	southern hemisphere	medium	many	No
13	gray	low	southern hemisphere	large	few	No



1. Decision Tree

1.2 Example: simple training data set

Problem setting

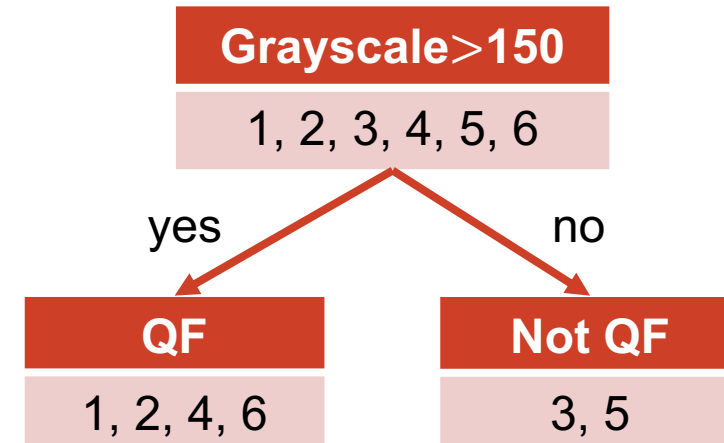
- **Set of possible samples X**
 - Each sample x in X is a feature vector;
 - e.g., <Grayscale=gray, Latitude=high, ..., Barb number=many>;
- **Unknown target function $f: X \rightarrow Y$**
 - $Y = 1$ if the filament is quiescent filament, else $Y = 0$;
- **Set of function hypotheses $H = \{h|h: X \rightarrow Y\}$**
 - Each hypothesis h is a decision tree;
 - Trees sorts x to leaf, which assigns y .

1. Decision Tree

1.3 Learning a decision stump

Decision stump: simple decision tree with 1 splitting rule based on thresholding 1 feature.

ID	Grayscale (Intensity)	Latitude	Barb Number	Quiescent Filament
1	200	60	10	1
2	185	70	8	1
3	145	65	12	1
4	195	30	2	0
5	120	60	2	0
6	175	35	1	0



How to find the best “rule” (best classify measurement/criterion) ?

1. Define a “**score**” for the rule;
2. **Search** for the rule with the best score.

Most intuitive score: **classification accuracy**

If we use this rule, how many examples do we label correctly?

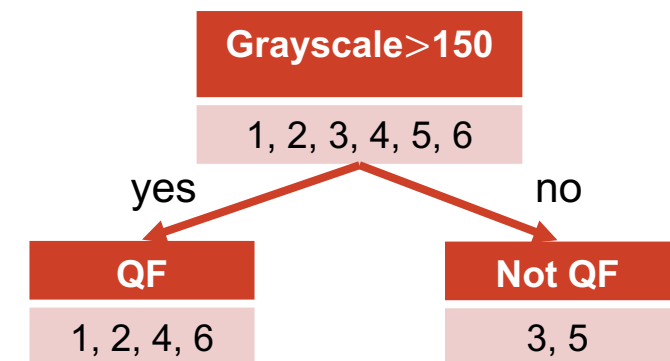
1. Decision Tree

1.3 Learning a decision stump

Computing classification accuracy for the rule (grayscale>150):

- Find **most common labels** if we use this rule:
 - when (grayscale>150), we get “yes” 2 times out of 4;
 - when (grayscale≤150), we get “no” 1 times out of 2.
- Compute accuracy:
 - the accuracy (“score”) of the rule (grayscale=dark gray) is 3 times out of 6.

ID	Grayscale (Intensity)	Latitude	Barb Number	Quiescent Filament
1	200	60	10	1
2	185	70	8	1
3	145	65	12	1
4	195	30	2	0
5	120	60	2	0
6	175	35	1	0



This “**score**” evaluates quality of a rule.

We “learn” a decision stump by **find the rule with the best score**.

1. Decision Tree

1.1 Learning a decision stump

Search for the decision stump maximizing classification **score**:

- “baseline rule” of predicting mode (no split):
 - 3/6 accuracy
- If (grayscale>150):
 - predict “yes” (2/4) else predict “no” (1/2): 1/2 accuracy
- If (latitude>40):
 - predict “yes” (3/4) else predict “no” (2/2): 5/6 accuracy
- If (Barb>5):
 - predict “yes” (3/3) else predict “no” (3/3): 6/6 accuracy

ID	Grayscale (Intensity)	Latitude	Barb Number	Quiescent Filament
1	200	60	10	1
2	185	70	8	1
3	145	65	12	1
4	195	30	2	0
5	120	60	2	0
6	175	35	1	0

- Highest-scoring rule: (Barb>5) with leaves “yes” and “no”.
- Notice we **only need to test feature thresholds that happen** in the data:
 - There is no point in testing the rule (Barb>10), it gets the “baseline” score;
 - There is no point in testing the rule (Barb>0.5), it gets the (Barb>1) score;
 - We do not need to test “<”, since it would give equivalent rules.

1. Decision Tree

1.1 Learning a decision stump

How much does this cost?

- Assume we have:
 - “ n ” examples (filaments that we detected);
 - “ d ” features (filaments features that we detected);
 - “ k ” thresholds (>0 , >10 , ...) for each feature.
- Computing the score of one rule costs $O(n)$:
 - We need to go through all “ n ” examples to find most common labels;
 - We need to go through all “ n ” examples again to compute the accuracy.
- We compute score for up to “ $k * d$ ” rules.
- So the total cost of $O(ndk)$.

1. Decision Tree

1.1 Learning a decision stump

Is a cost of $O(ndk)$ good?

- Assume we have:
 - “ n ” examples (filaments that we detected);
 - “ d ” features (filaments features that we detected);
 - “ k ” thresholds (>0 , >10 , ...) for each feature.
- Size of the input data is $O(nd)$:
 - If “ k ” is small then the cost is roughly the same as loading the data;
 - E.g., if all features are binary then $k=1$, just test (feature >0). The cost of fitting decision stump is is $O(nd)$, so we can fit huge dataset.
 - If “ k ” is large then this could be too slow for large datasets;
 - E.g., if all features are numerical with unique values then $k = n$. The cost of fitting decision stump is is $O(n^2d)$.

1. Decision Tree

Shouldn't we just use accuracy score?

For leaves: yes, just maximize accuracy.

For internal nodes: ?

Which **score function should
a decision tree use?**

1. Decision Tree

1.2 The splitting criterion

Previous Lecture

Entropy as Measure of Randomness

For a categorical variable that can take “ k ” values, entropy is defined by:

$$\text{entropy} = -\sum_{c=1}^k p_c \log p_c$$

where p_c is the proportion of times you have value “ c ”.

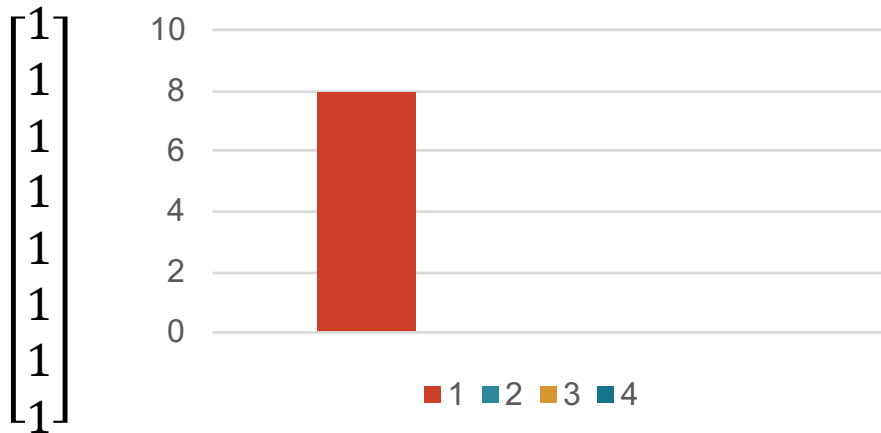
- Roughly, it's another measure of the “spread” of values.
- Minimum value is 0, maximum value is $\log(k)$.
 - We use the convention that $0 \log 0 = 0$.

1. Decision Tree

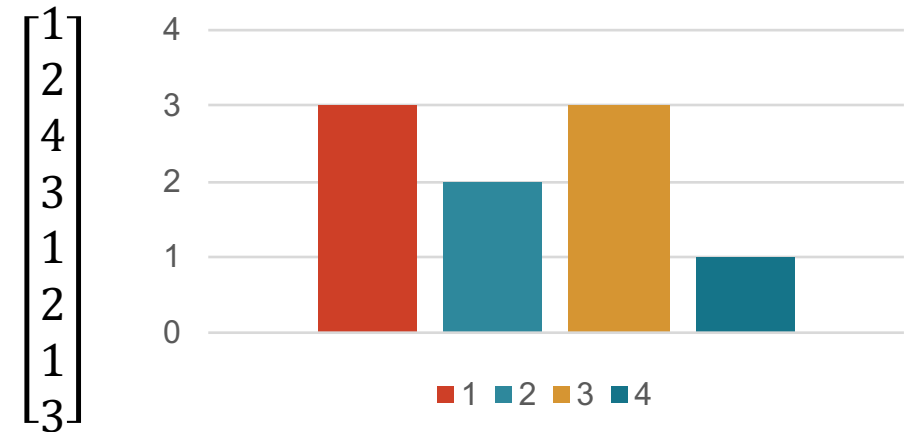
1.2 The splitting criterion

Previous Lecture

Entropy as Measure of Randomness



Low entropy means “very predictable”



High entropy means “very random”

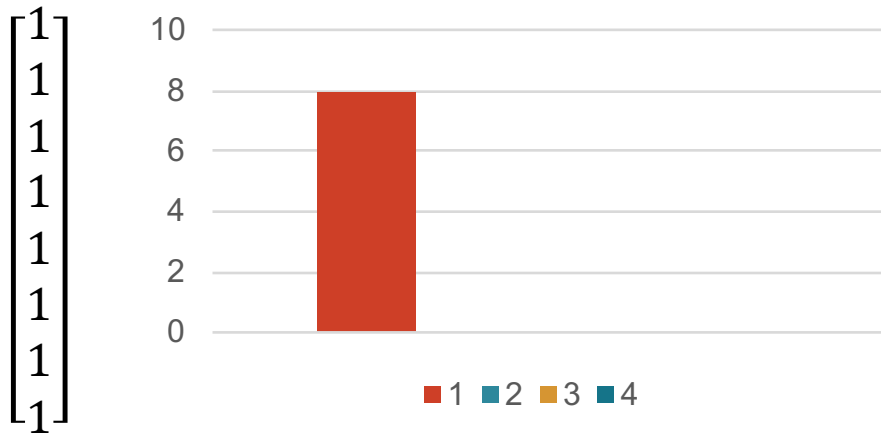
For categorical features: uniform distribution has highest entropy.

For continuous densities with fixed mean and variance: Normal distribution has highest entropy.

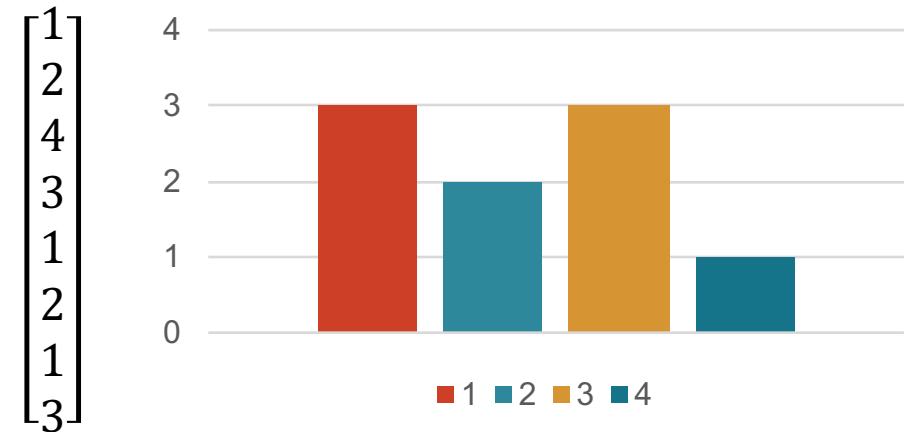
1. Decision Tree

1.2 The splitting criterion

Information Entropy as measure of **Purity**



Low entropy means “**high purity**”



High entropy means “**low purity**”

$$\text{Ent}(D) = - \sum_{k=1}^K p_k \log_2 p_k, k = 1, 2, \dots, K$$

1. Decision Tree

1.2 The splitting criterion

ID3 (Iterative Dichotomizer)

Information Entropy

$$\text{Ent}(D) = - \sum_{k=1}^K p_k \log_2 p_k, k = 1, 2, \dots, K$$

Information Gain

$$\text{Gain}(D) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v), v = 1, 2, \dots, V$$

$$\mathbf{A} = \{a_1, a_2, \dots, a_u\}, a_* = \{a_*^1, a_*^2, \dots, a_*^V\}$$

1. Decision Tree

1.2 The splitting criterion

ID	Grayscale (Intensity)	Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament
1	gray	high	northern hemisphere	small	many	Yes
2	dark gray	high	northern hemisphere	medium	many	Yes
3	dark gray	high	northern hemisphere	medium	many	Yes
4	gray	high	southern hemisphere	medium	many	Yes
5	light gray	middle	southern hemisphere	small	some	Yes
6	gray	middle	southern hemisphere	medium	some	yes
7	dark gray	middle	southern hemisphere	medium	some	No
8	dark gray	middle	northern hemisphere	large	few	No
9	light gray	middle	northern hemisphere	medium	few	No
10	gray	high	southern hemisphere	medium	some	No
11	light gray	low	northern hemisphere	medium	few	No
12	dark gray	middle	southern hemisphere	medium	many	No
13	gray	low	southern hemisphere	large	few	No

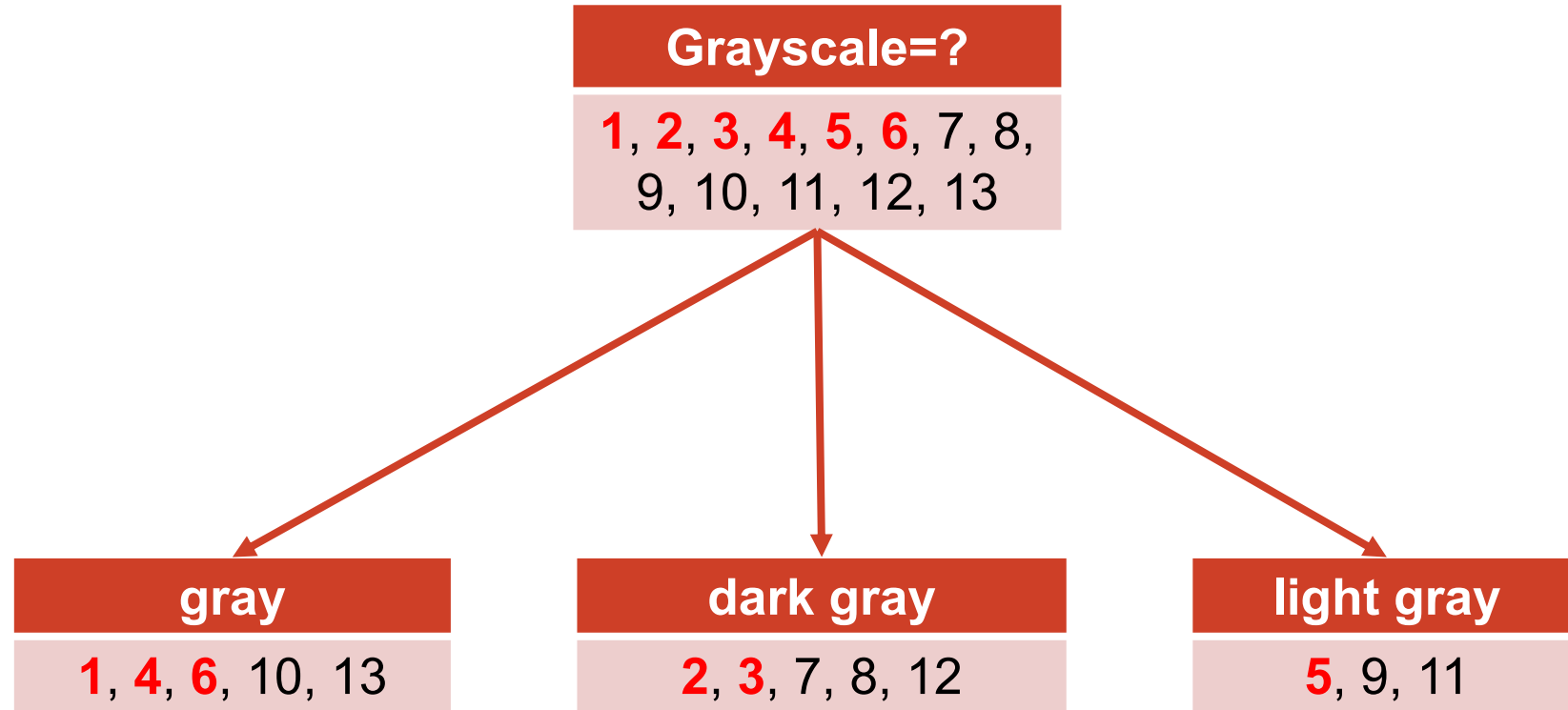
1. Decision Tree

1.2 The splitting criterion

ID	Grayscale (Intensity)	Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament
1	gray	high	northern hemisphere	small	many	Yes
4	gray	high	southern hemisphere	medium	many	Yes
6	gray	middle	southern hemisphere	medium	some	yes
10	gray	high	southern hemisphere	medium	some	No
13	gray	low	southern hemisphere	large	few	No
2	dark gray	high	northern hemisphere	medium	many	Yes
3	dark gray	high	northern hemisphere	medium	many	Yes
7	dark gray	middle	southern hemisphere	medium	some	No
8	dark gray	middle	northern hemisphere	large	few	No
12	dark gray	middle	southern hemisphere	medium	many	No
5	light gray	middle	southern hemisphere	small	some	Yes
9	light gray	middle	northern hemisphere	medium	few	No
11	light gray	low	northern hemisphere	medium	few	No

1. Decision Tree

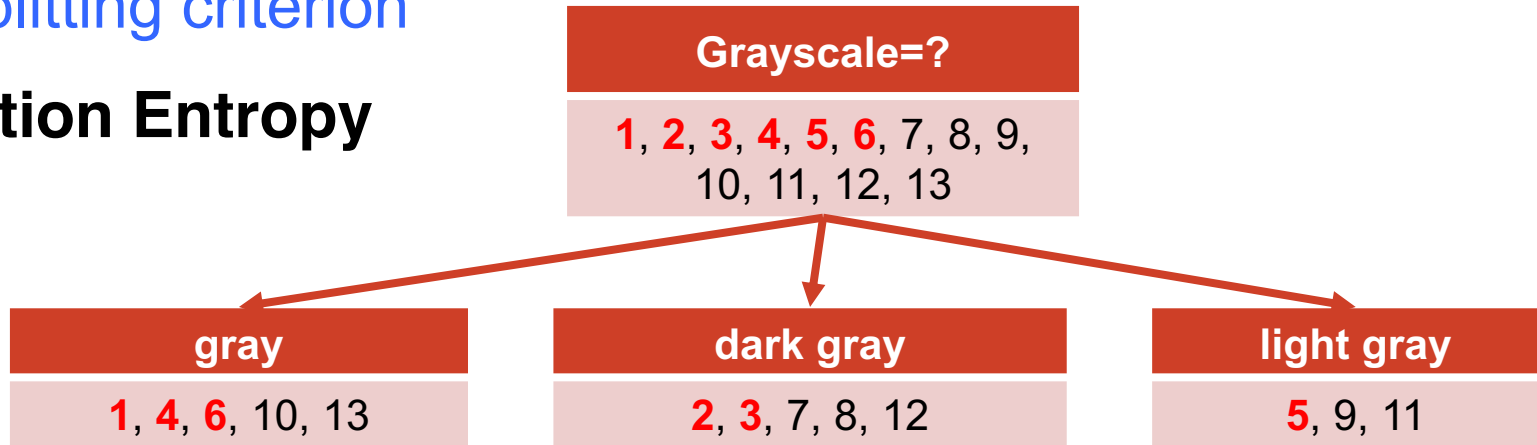
1.2 The splitting criterion



1. Decision Tree

1.2 The splitting criterion

Information Entropy



$$\text{For } D^1 = \{1, 4, 6, 10, 13\}, p_1 = \frac{3}{5}, p_2 = \frac{2}{5}, \quad \text{Ent}(D^1) = - \sum_{k=1}^2 p_k \log_2 p_k = - \left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5} \right) = 0.971$$

$$\text{For } D^2 = \{2, 3, 7, 8, 12\}, p_1 = \frac{2}{5}, p_2 = \frac{3}{5}, \quad \text{Ent}(D^2) = - \sum_{k=1}^2 p_k \log_2 p_k = - \left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5} \right) = 0.971$$

$$\text{For } D^3 = \{5, 9, 11\}, p_1 = \frac{1}{3}, p_2 = \frac{2}{3}, \quad \text{Ent}(D^3) = - \sum_{k=1}^2 p_k \log_2 p_k = - \left(\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3} \right) = 0.918$$

1. Decision Tree

1.2 The splitting criterion

Information Gain

$$\text{Gain}(D) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v), v = 1, 2, \dots, V$$

$$\text{Ent}(D) = -\left(\frac{6}{13} \log_2 \frac{6}{13} + \frac{7}{13} \log_2 \frac{7}{13}\right) = 0.996 \quad \text{Ent}(D^1) = -\left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5}\right) = 0.971$$

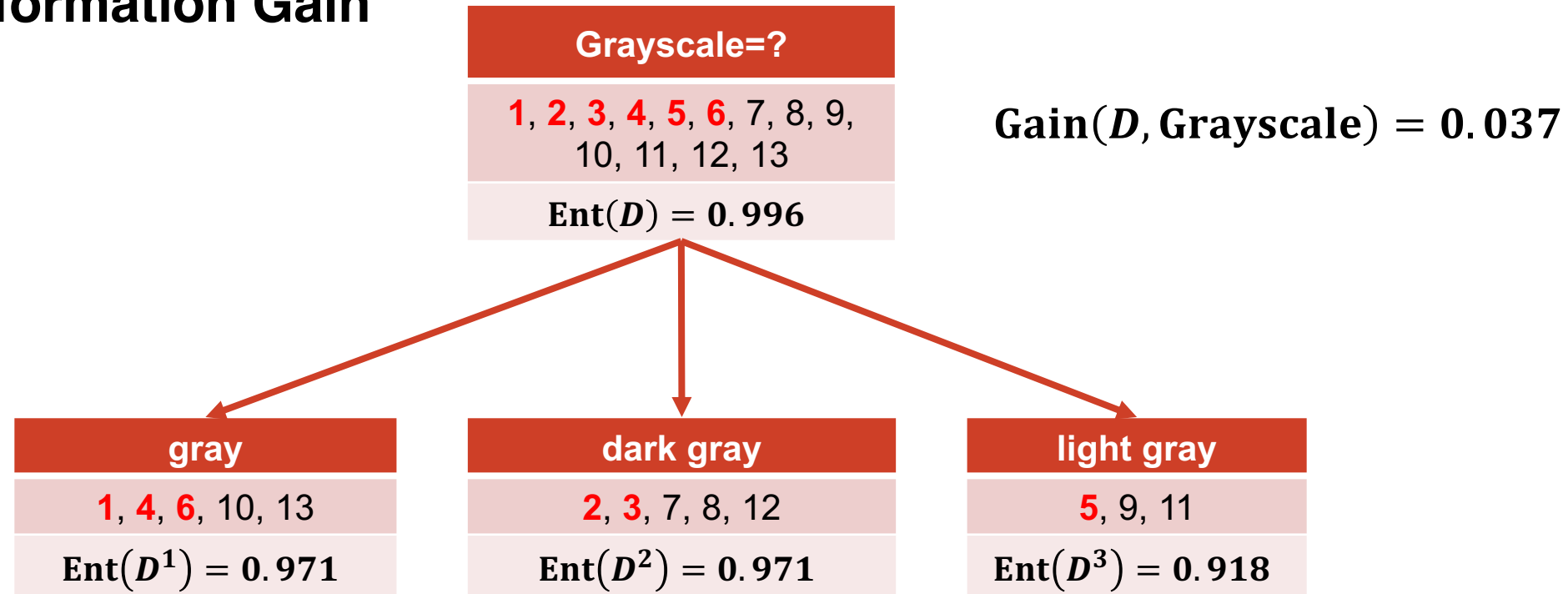
$$\text{Ent}(D^2) = -\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5}\right) = 0.971 \quad \text{Ent}(D^3) = -\left(\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3}\right) = 0.918$$

$$\begin{aligned} \text{Gain}(D, \text{Grayscale}) &= \text{Ent}(D) - \sum_{v=1}^3 \frac{|D^v|}{|D|} \text{Ent}(D^v) \\ &= 0.996 - \left(\frac{5}{13} \times 0.971 + \frac{5}{13} \times 0.971 + \frac{3}{13} \times 0.918\right) = 0.037 \end{aligned}$$

1. Decision Tree

1.2 The splitting criterion

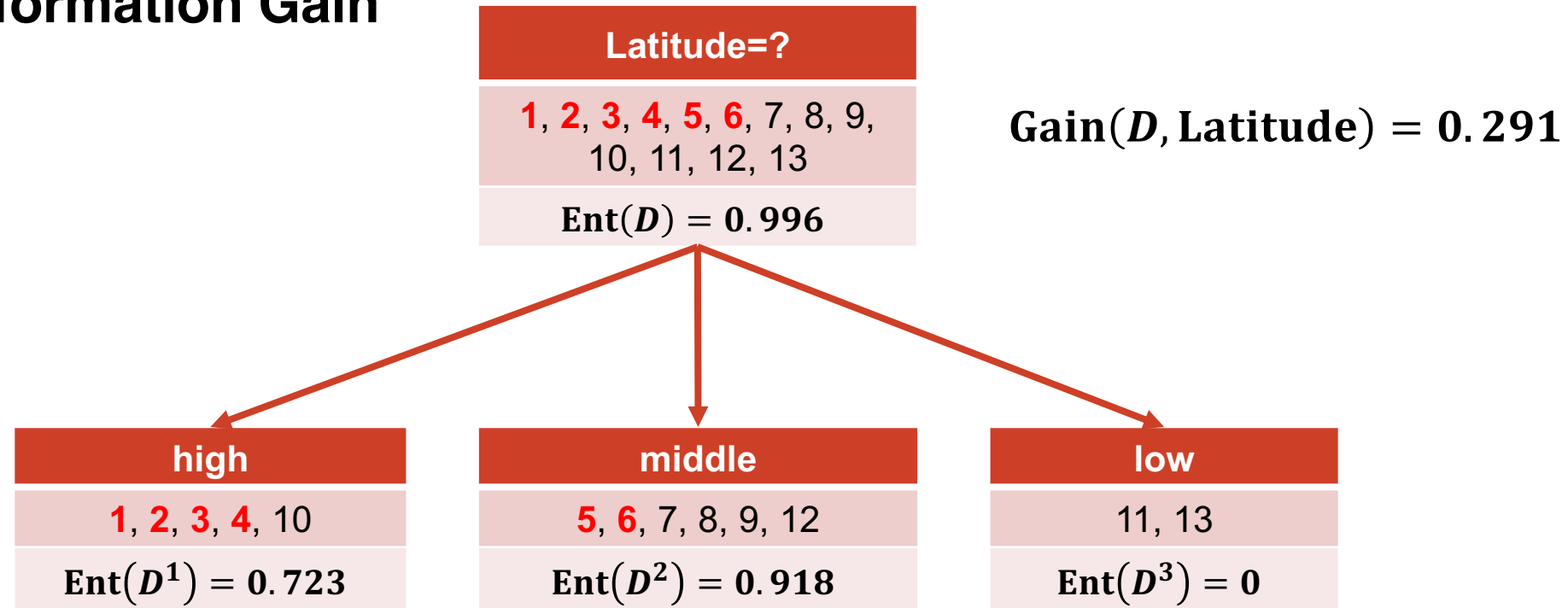
Information Gain



1. Decision Tree

1.2 The splitting criterion

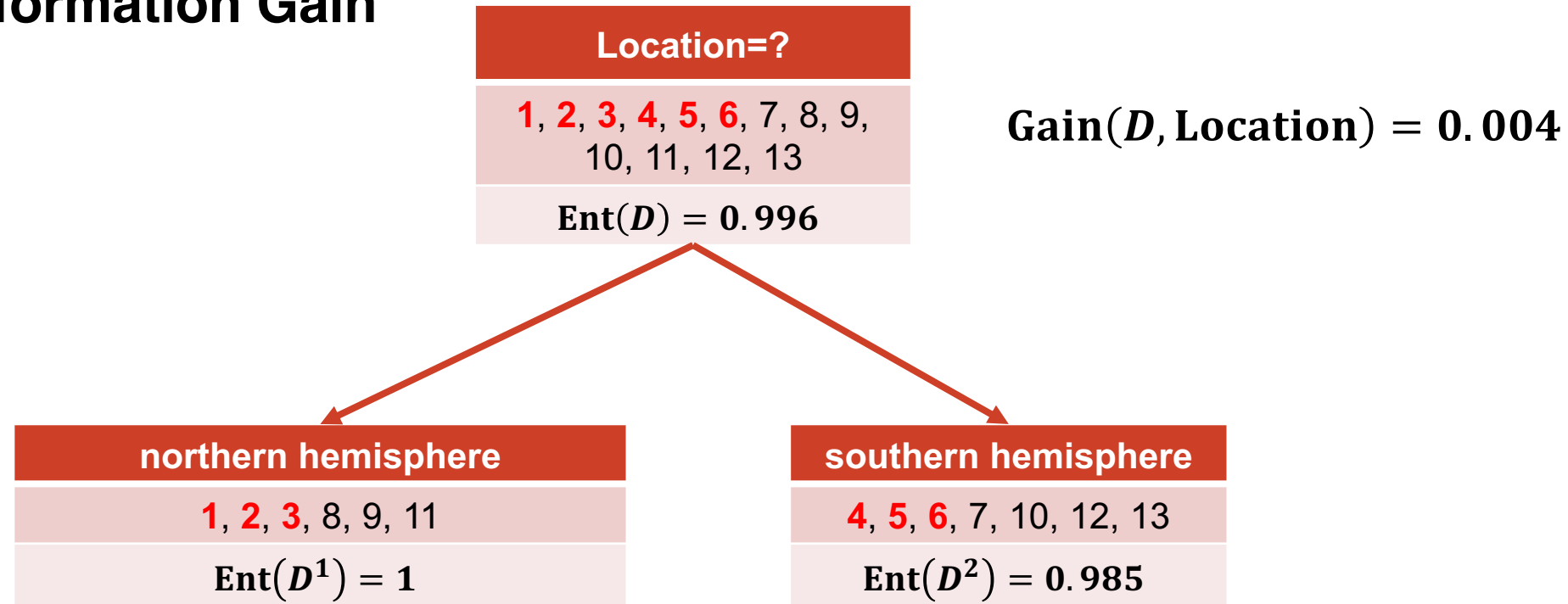
Information Gain



1. Decision Tree

1.2 The splitting criterion

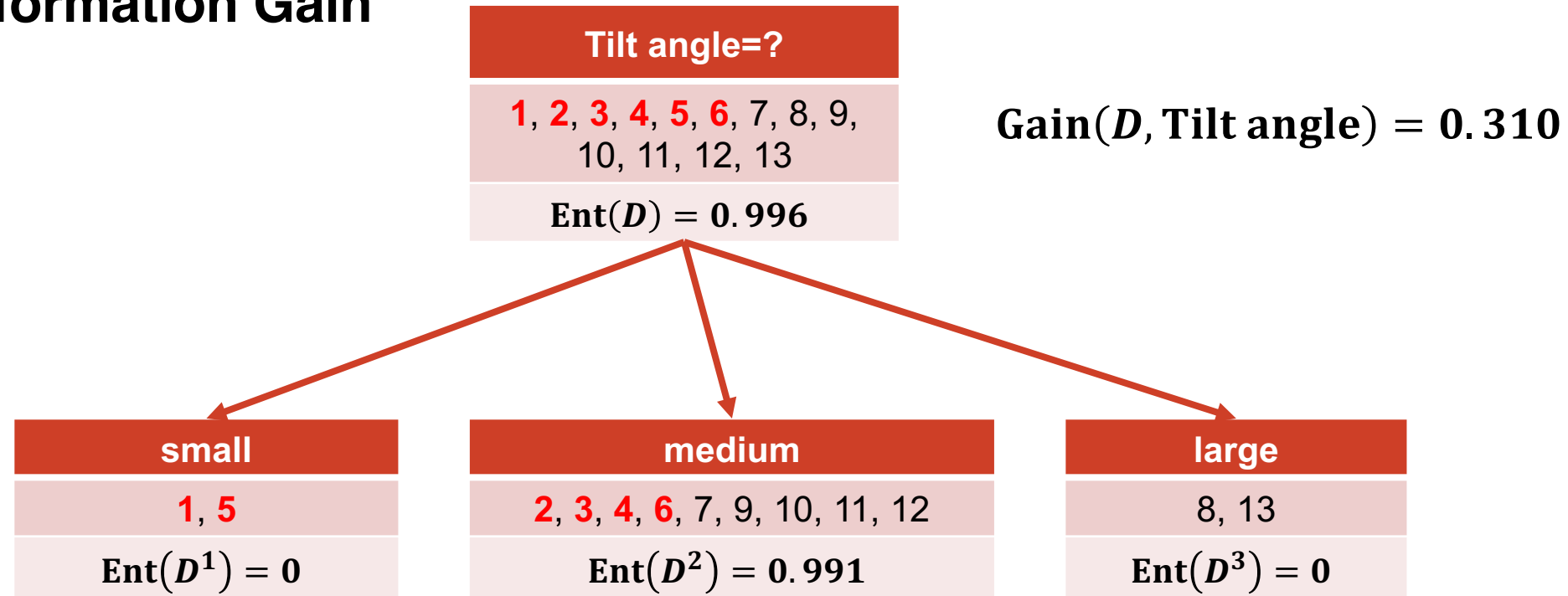
Information Gain



1. Decision Tree

1.2 The splitting criterion

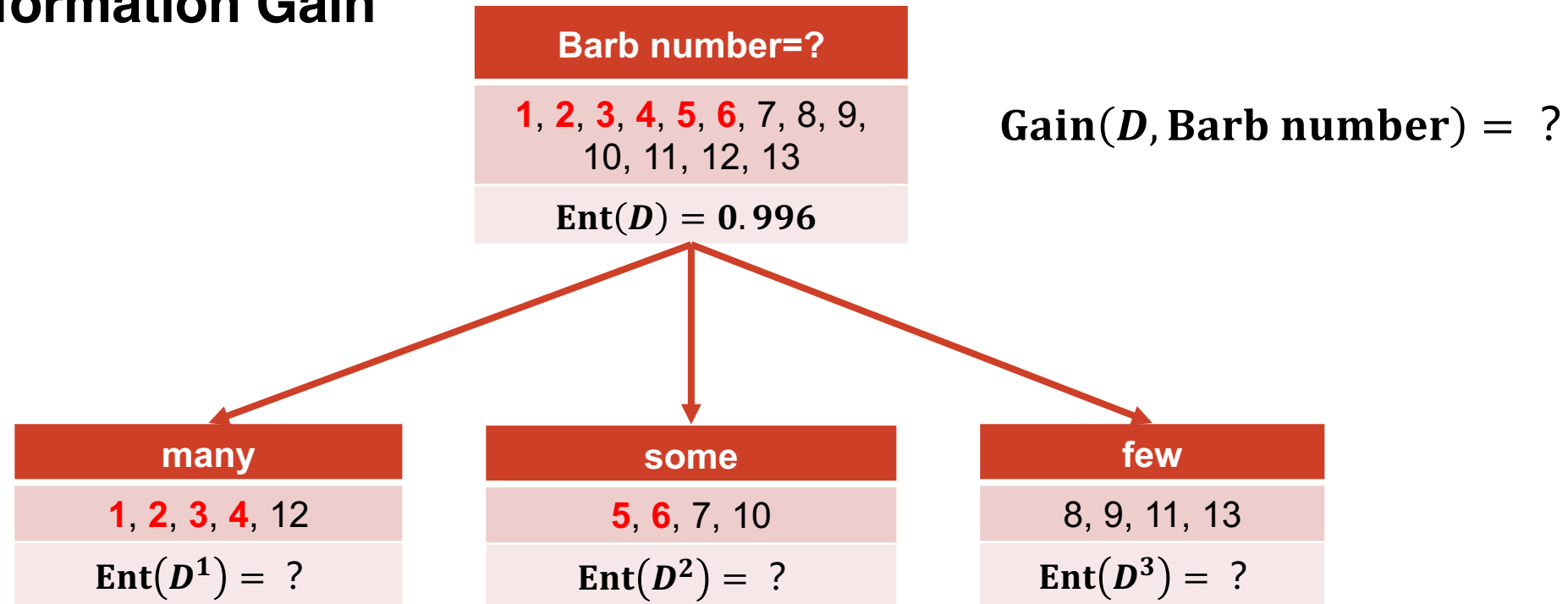
Information Gain



1. Decision Tree

1.2 The splitting criterion

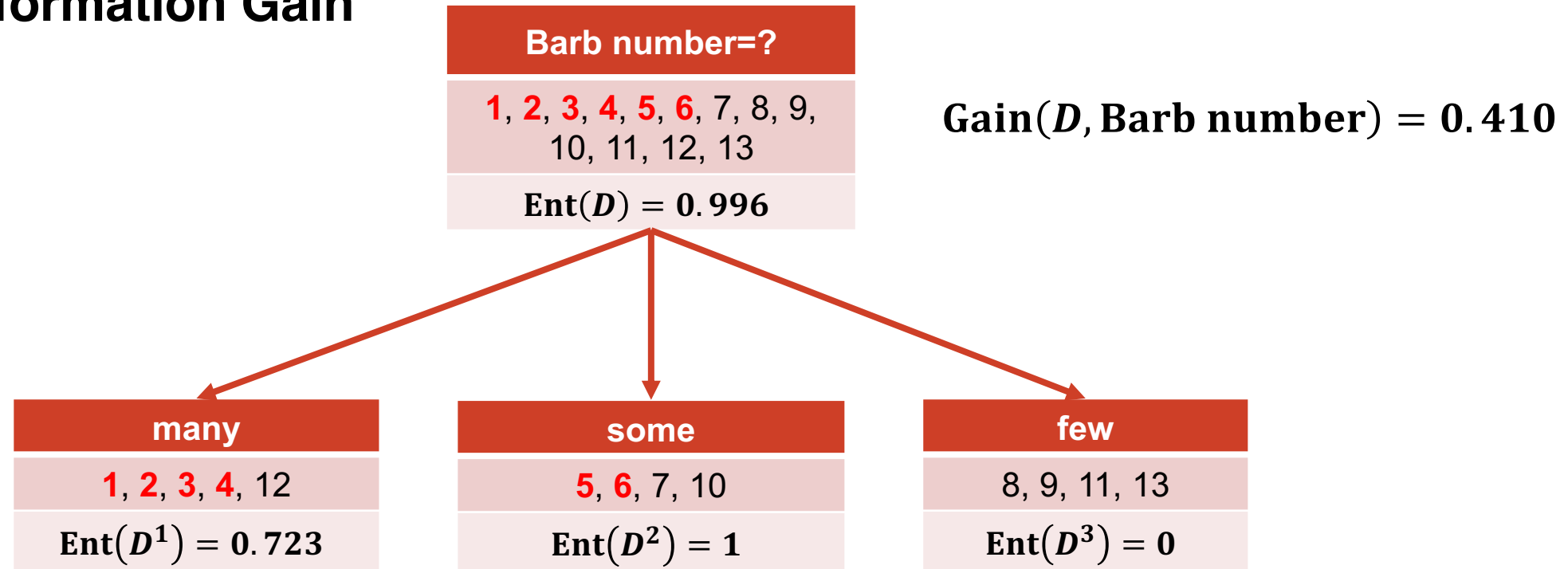
Information Gain



1. Decision Tree

1.2 The splitting criterion

Information Gain



1. Decision Tree

1.2 The splitting criterion

Information Gain

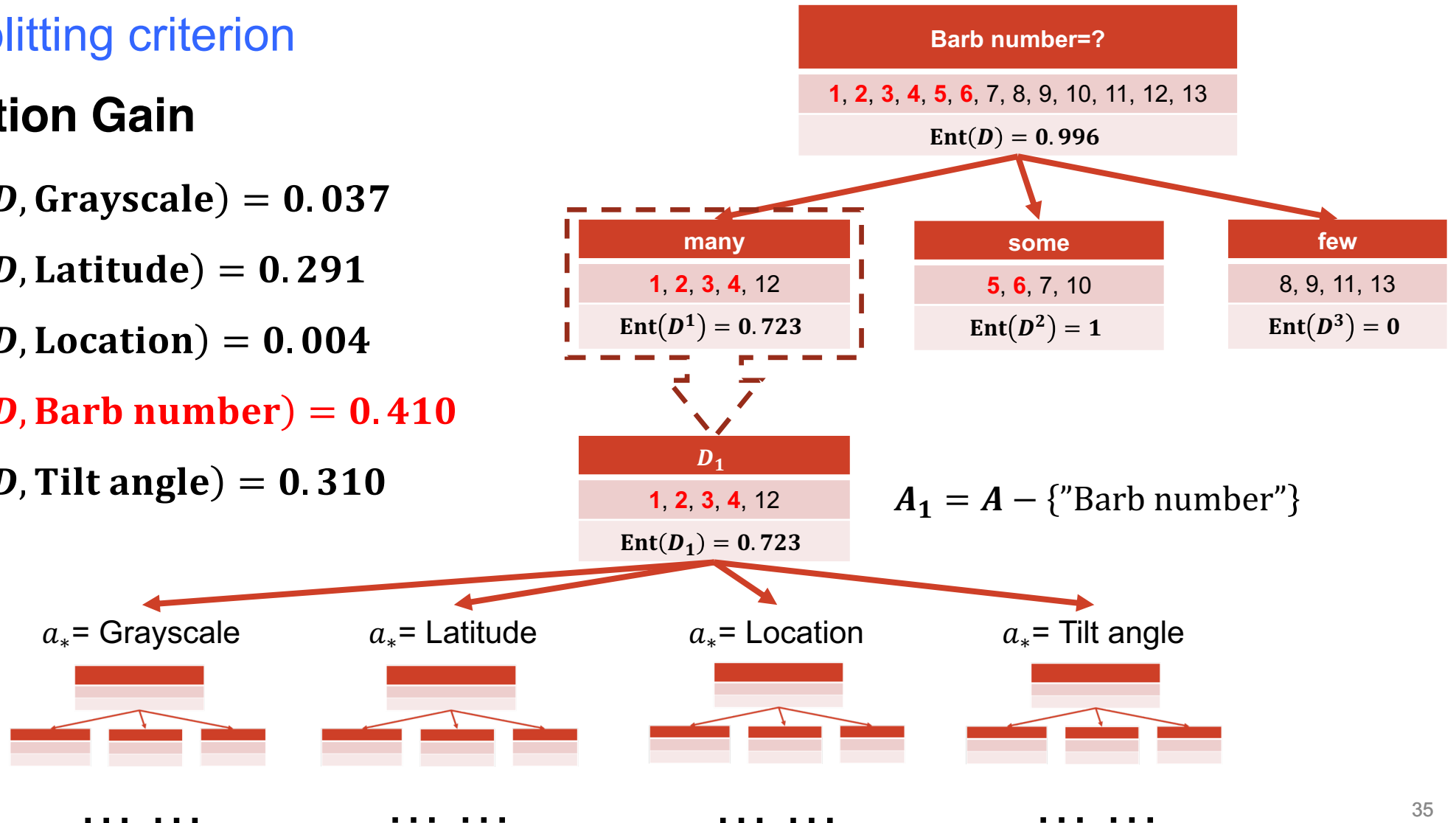
$$\text{Gain}(D, \text{Grayscale}) = 0.037$$

$$\text{Gain}(D, \text{Latitude}) = 0.291$$

$$\text{Gain}(D, \text{Location}) = 0.004$$

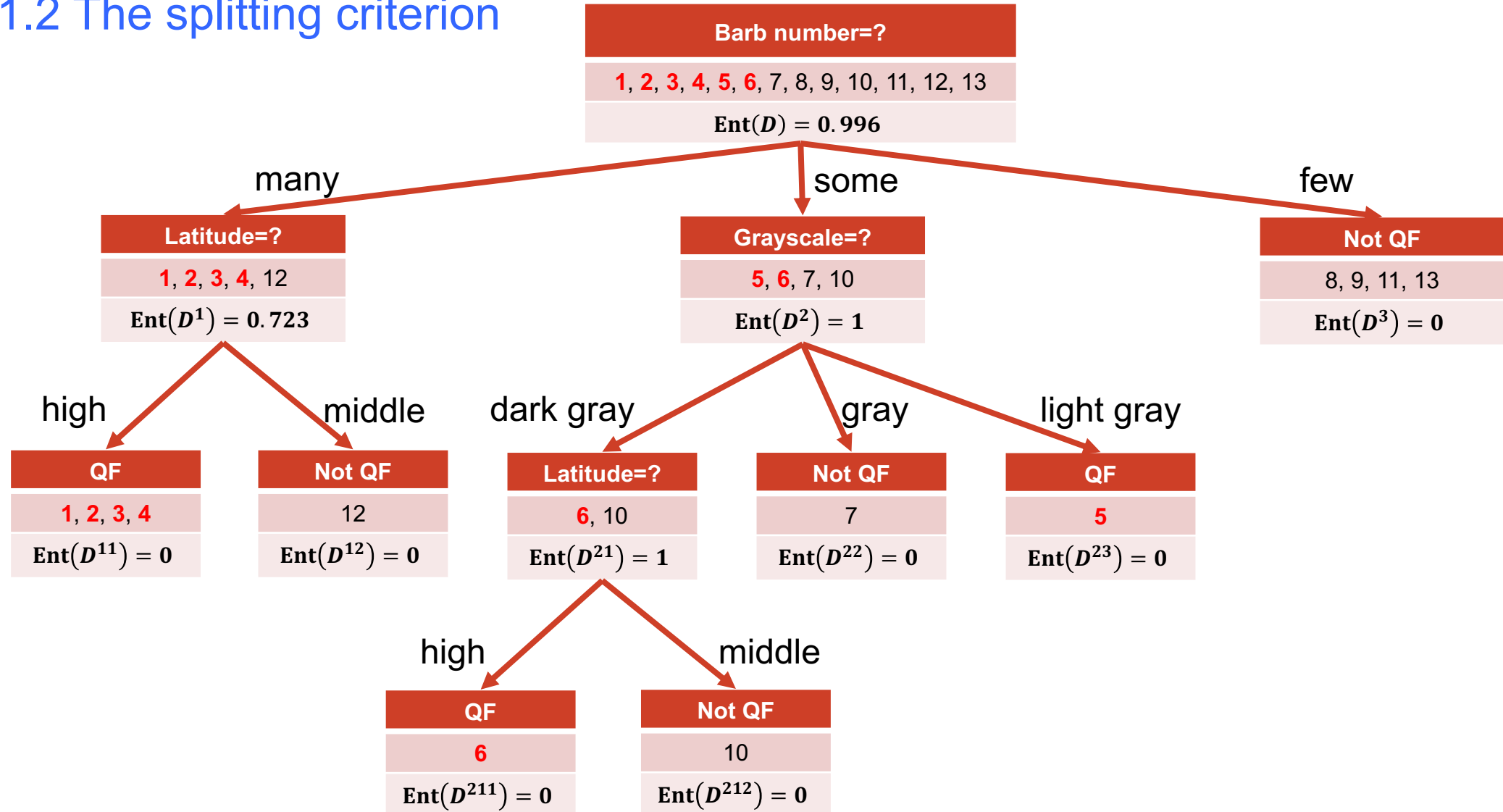
$$\text{Gain}(D, \text{Barb number}) = 0.410$$

$$\text{Gain}(D, \text{Tilt angle}) = 0.310$$



1. Decision Tree

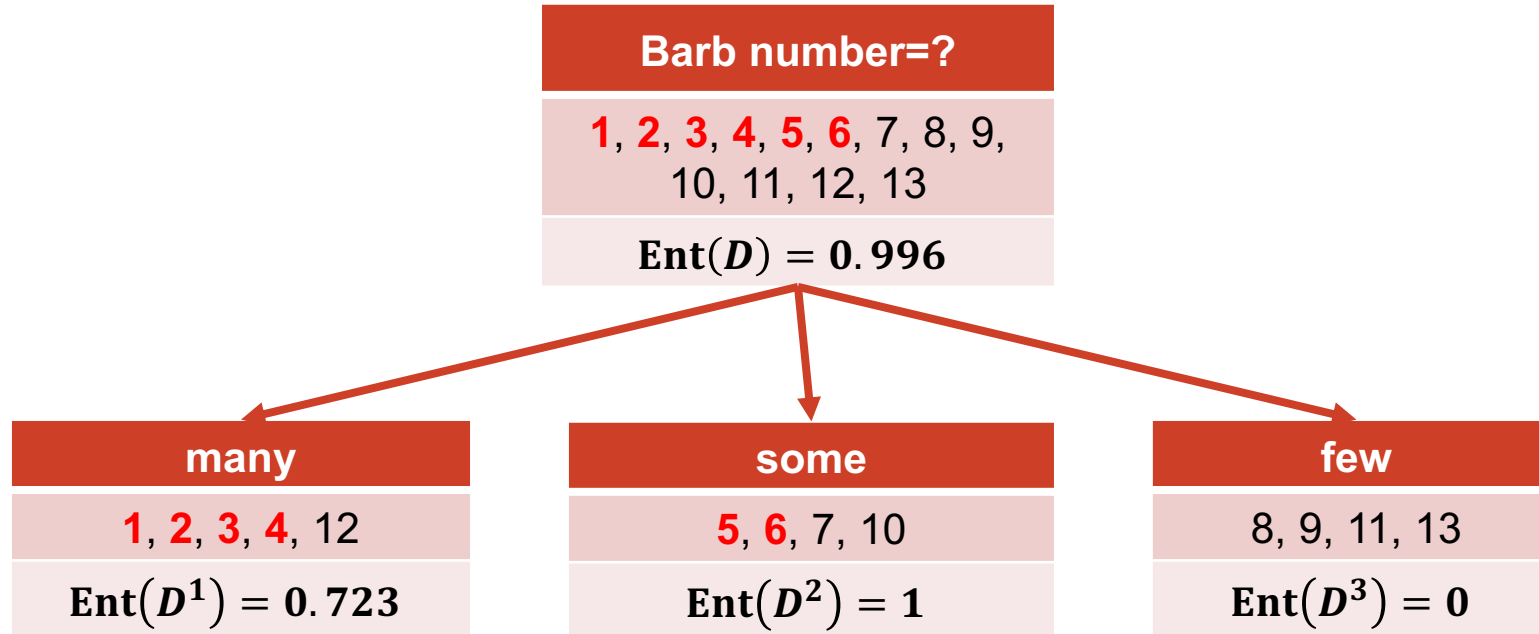
1.2 The splitting criterion



1. Decision Tree

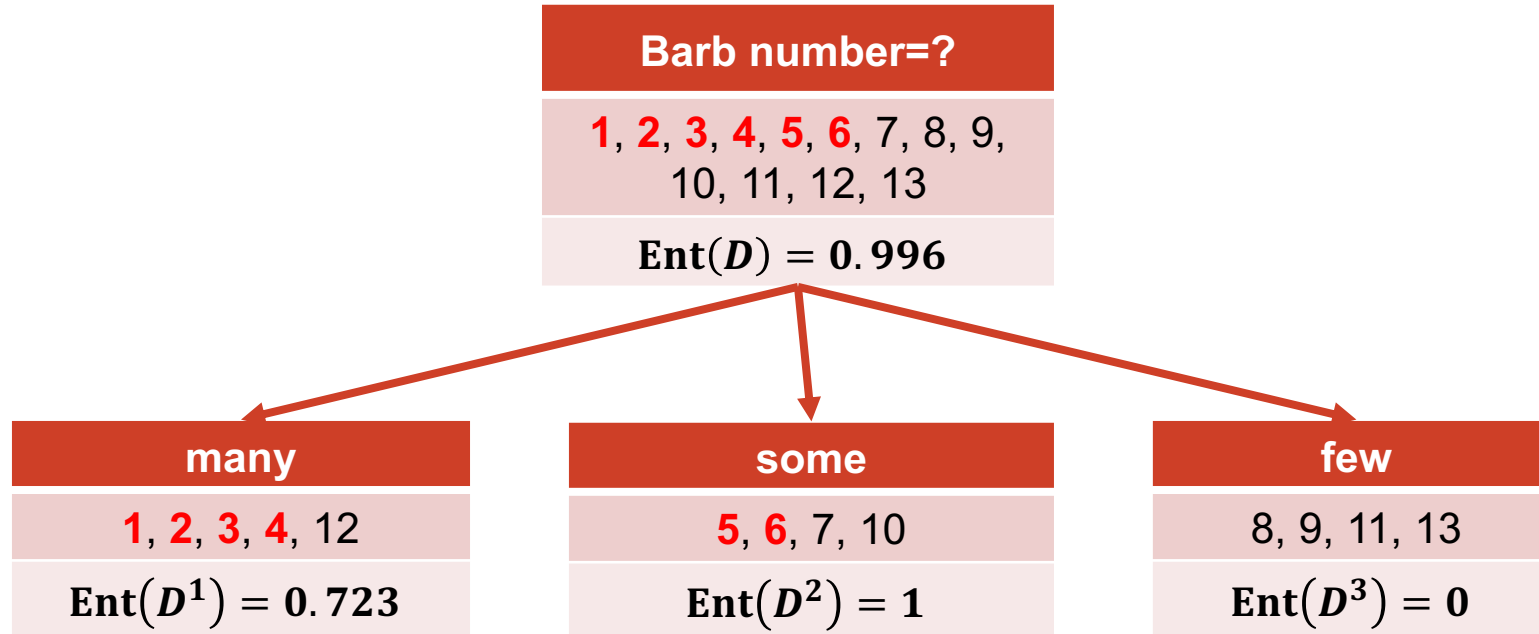
Why growing decision trees via **entropy instead of **misclassification error** (or **classification accuracy**)?**

1. Decision Tree



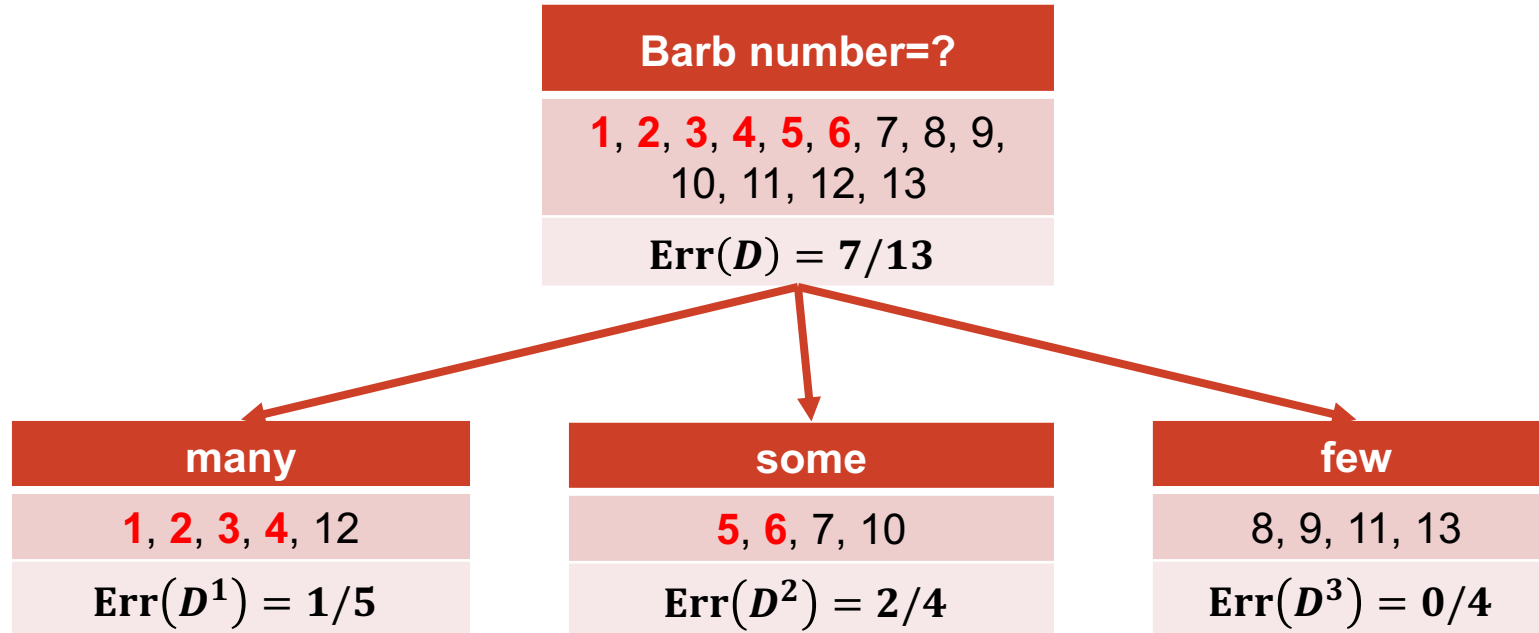
$$\text{Gain}(D) = I(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} I(D^v)$$

1. Decision Tree



$$\begin{aligned}\text{Gain}(D, \text{Barb number}) &= \text{Ent}(D) - \sum_{v=1}^3 \frac{|D^v|}{|D|} \text{Ent}(D^v) \\ &= 0.996 - \left(\frac{5}{13} \times 0.723 + \frac{4}{13} \times 1 + \frac{4}{13} \times 0 \right) = \mathbf{0.410}\end{aligned}$$

1. Decision Tree



$$\begin{aligned}\text{Gain}(D, \text{Barb number}) &= \text{Err}(D) - \sum_{v=1}^3 \frac{|D^v|}{|D|} \text{Err}(D^v) \\ &= \frac{7}{13} - \left(\frac{5}{13} \times \frac{1}{5} + \frac{4}{13} \times \frac{2}{4} + \frac{4}{13} \times 0 \right) = 0\end{aligned}$$

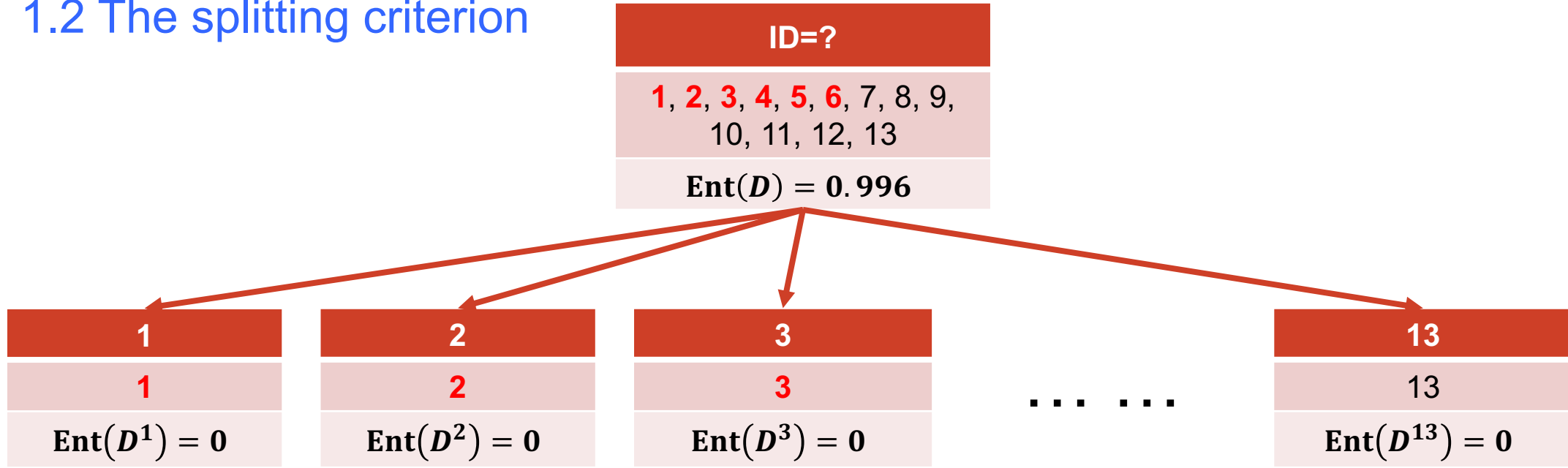
1. Decision Tree

1.2 The splitting criterion

ID	Grayscale (Intensity)	Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament
1	gray	high	northern hemisphere	small	many	Yes
2	dark gray	high	northern hemisphere	medium	many	Yes
3	dark gray	high	northern hemisphere	medium	many	Yes
4	gray	high	southern hemisphere	medium	many	Yes
5	light gray	middle	southern hemisphere	small	some	Yes
6	gray	middle	southern hemisphere	medium	some	yes
7	dark gray	middle	southern hemisphere	medium	some	No
8	dark gray	middle	northern hemisphere	large	few	No
9	light gray	middle	northern hemisphere	medium	few	No
10	gray	high	southern hemisphere	medium	some	No
11	light gray	low	northern hemisphere	medium	few	No
12	dark gray	middle	southern hemisphere	medium	many	No
13	gray	low	southern hemisphere	large	few	No

1. Decision Tree

1.2 The splitting criterion



$$\begin{aligned}\text{Gain}(D, \text{Grayscale}) &= \text{Ent}(D) - \sum_{v=1}^{13} \frac{D^v}{D} \text{Ent}(D^v) \\ &= 0.996 - \left(\frac{1}{13} \times 0 + \frac{1}{13} \times 0 + \dots + \frac{1}{13} \times 0 + \frac{1}{13} \times 0 \right) = 0.996\end{aligned}$$

1. Decision Tree

1.2 The splitting criterion

C4.5

Information Gain

$$\text{Gain}(D) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v), v = 1, 2, \dots, V$$
$$A = \{a_1, a_2, \dots, a_u\}, a_* = \{a_*^1, a_*^2, \dots, a_*^V\}$$

Gain Ratio

$$\text{GainRatio}(D) = \frac{\text{Gain}(D)}{\text{SplitInfo}(D)}$$

$$\text{SplitInfo}(D) = - \sum_{v=1}^V \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}, v = 1, 2, \dots, V$$

1. Decision Tree

1.2 The splitting criterion

CART (Classification and Regression Tree)

Gini Impurity

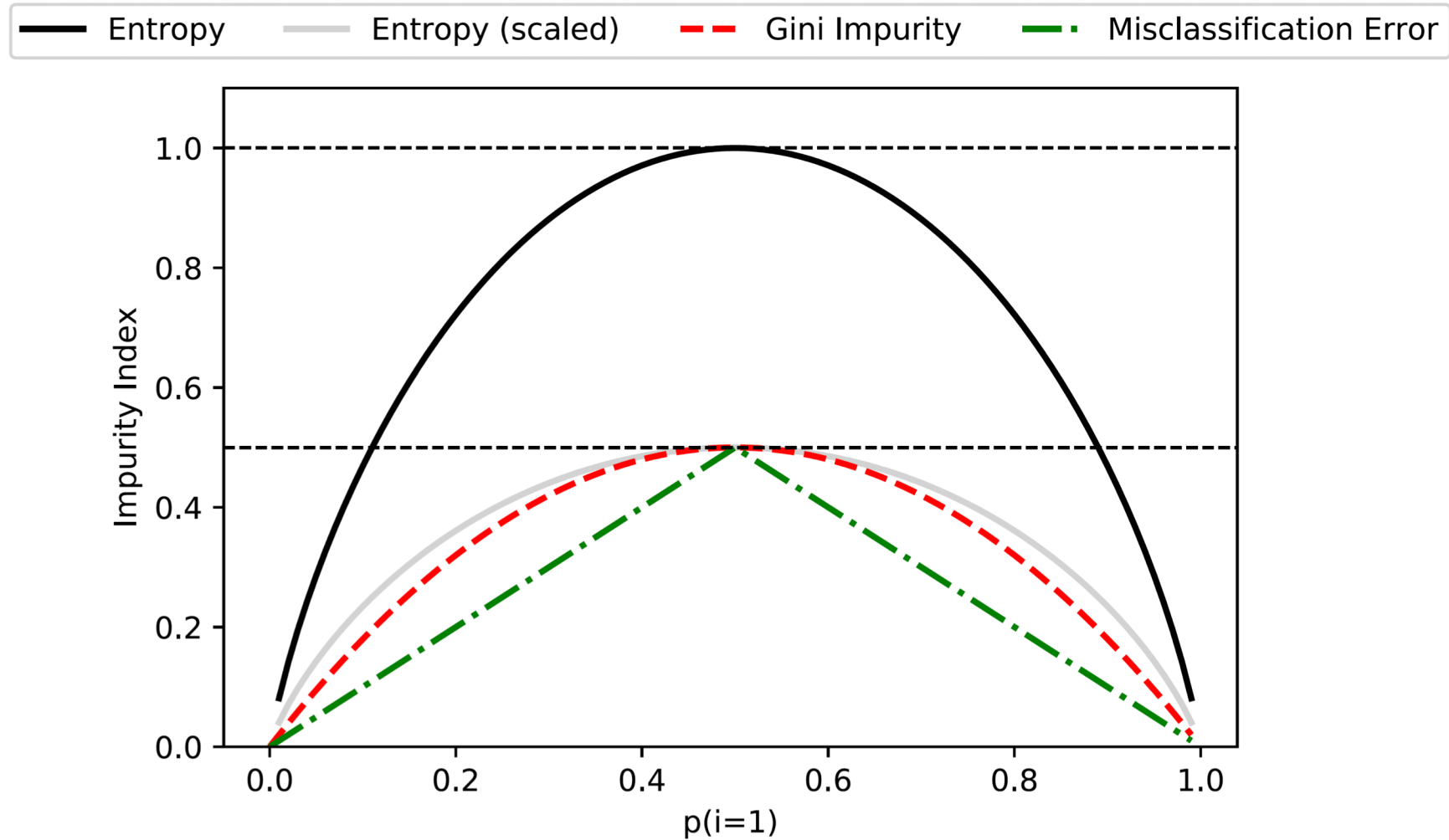
$$\text{Gini}(D) = \sum_{k=1}^K \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^K p_k^2, k = 1, 2, \dots, K$$

Gini Index

$$\text{GiniIndex}(D) = \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Gini}(D^v), v = 1, 2, \dots, V$$
$$\mathbf{A} = \{a_1, a_2, \dots, a_u\}, a_* = \{a_*^1, a_*^2, \dots, a_*^V\}$$

1. Decision Tree

1.2 The splitting criterion



1. Decision Tree

1.3 Pruning

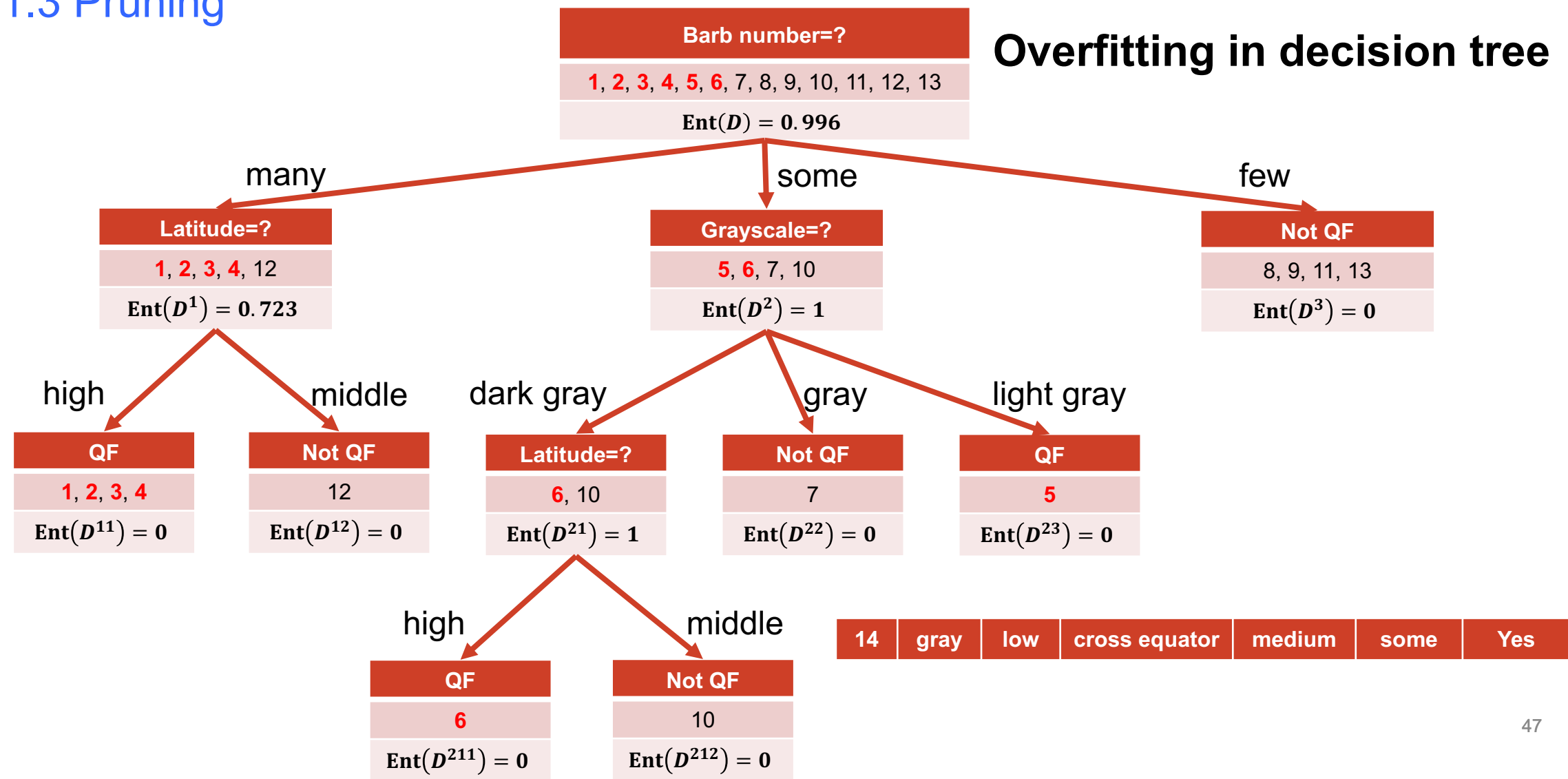
ID	Grayscale (Intensity)	Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament
1	gray	high	northern hemisphere	small	many	Yes
2	dark gray	high	northern hemisphere	medium	many	Yes
...	...	...	...	...	...	...
11	light gray	low	northern hemisphere	medium	few	No
12	dark gray	middle	southern hemisphere	medium	many	No
13	gray	low	southern hemisphere	large	few	No
14	gray	low	cross equator	medium	some	Yes

What effect on earlier tree?

1. Decision Tree

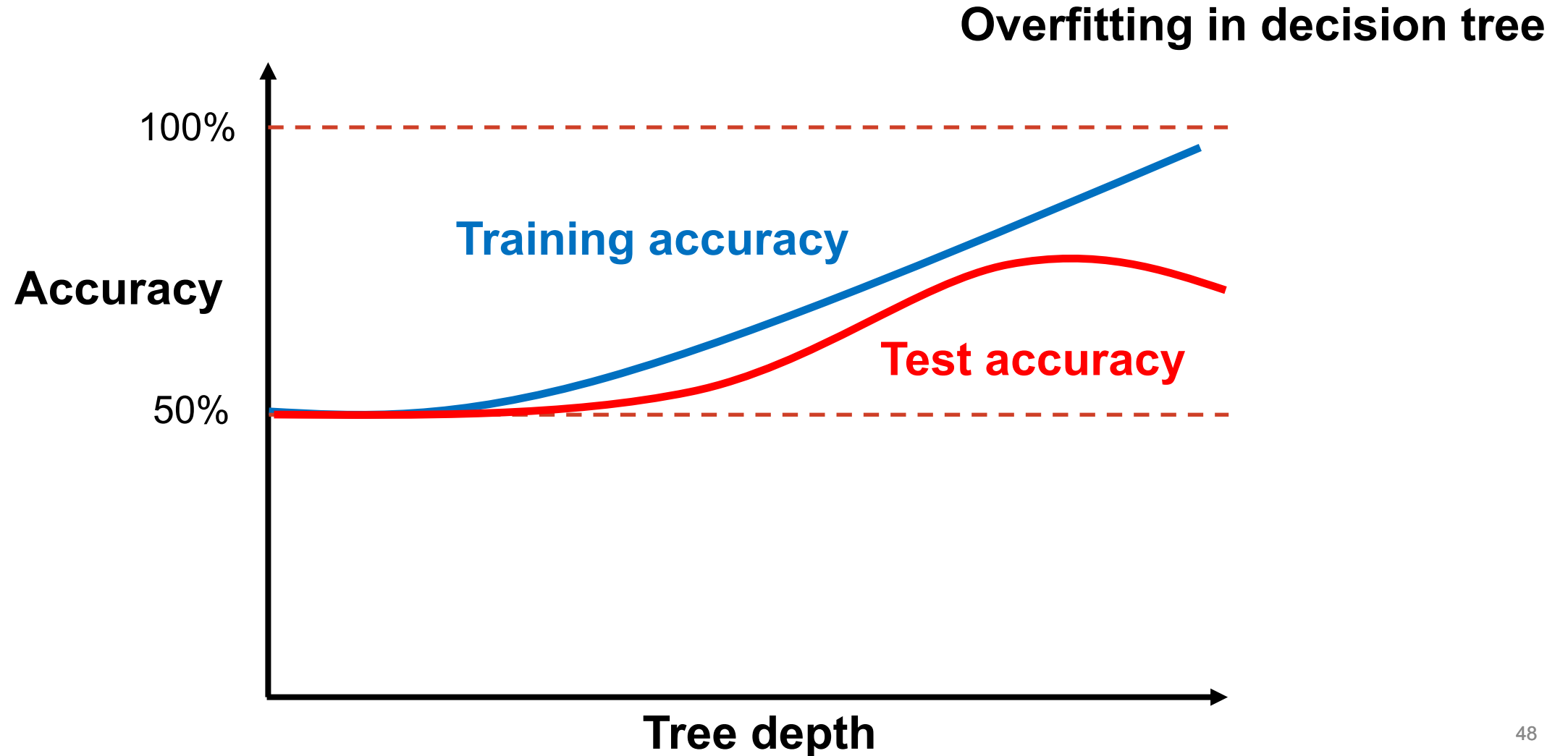
1.3 Pruning

Overfitting in decision tree



1. Decision Tree

1.3 Pruning



1. Decision Tree

1.3 Pruning

Split data into training and validation set

ID	Grayscale (Intensity)	Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament																																																		
1	gray	high	northern hemisphere	small	many	Yes																																																		
3	dark gray	high	northern hemisphere	medium	many	Yes																																																		
5	light gray	middle	southern hemisphere	small	some	Yes																																																		
6	gray	middle	southern hemisphere	medium	some	Yes																																																		
7	dark gray	middle	southern hemisphere	medium	some	No																																																		
9	light gray	<table><tr><th>ID</th><th>Grayscale (Intensity)</th><th>Latitude</th><th>Location</th><th>Tilt Angle</th><th>Barb Number</th><th>Quiescent Filament</th></tr><tr><td>2</td><td>dark gray</td><td>high</td><td>northern hemisphere</td><td>medium</td><td>many</td><td>Yes</td></tr><tr><td>4</td><td>gray</td><td>high</td><td>southern hemisphere</td><td>medium</td><td>many</td><td>Yes</td></tr><tr><td>6</td><td>gray</td><td>middle</td><td>southern hemisphere</td><td>medium</td><td>some</td><td>yes</td></tr><tr><td>8</td><td>dark gray</td><td>middle</td><td>northern hemisphere</td><td>large</td><td>few</td><td>No</td></tr><tr><td>10</td><td>gray</td><td>high</td><td>southern hemisphere</td><td>medium</td><td>some</td><td>No</td></tr><tr><td>12</td><td>dark gray</td><td>middle</td><td>southern hemisphere</td><td>medium</td><td>many</td><td>No</td></tr></table>	ID	Grayscale (Intensity)	Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament	2	dark gray	high	northern hemisphere	medium	many	Yes	4	gray	high	southern hemisphere	medium	many	Yes	6	gray	middle	southern hemisphere	medium	some	yes	8	dark gray	middle	northern hemisphere	large	few	No	10	gray	high	southern hemisphere	medium	some	No	12	dark gray	middle	southern hemisphere	medium	many	No					
ID	Grayscale (Intensity)		Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament																																																	
2	dark gray		high	northern hemisphere	medium	many	Yes																																																	
4	gray		high	southern hemisphere	medium	many	Yes																																																	
6	gray		middle	southern hemisphere	medium	some	yes																																																	
8	dark gray		middle	northern hemisphere	large	few	No																																																	
10	gray	high	southern hemisphere	medium	some	No																																																		
12	dark gray	middle	southern hemisphere	medium	many	No																																																		
11	light gray																																																							
13	gray																																																							

1. Decision Tree

1.3 Pruning

Pre-pruning

- Set a depth cut-off (maximum tree depth) a priori;
- Cost-complexity pruning:
 - where is an impurity measure,
 - is a tuning parameter,
 - is the total number of nodes;
- **Stop growing if split is not statistically significant (e.g. χ^2);**
- Set a minimum number of data points for each node.

1. Decision Tree

1.3 Pruning

Post-pruning

- **Grow full tree first, then remove nodes**, in C4.5
- Reduced-error pruning, remove nodes via validation set:
 - Greedily remove the node that most improves validation set accuracy
 - Problematic for limited data
- Can also convert trees to rules first and then prune the rules:
 - Convert tree to equivalent set of rules;
 - Prune each rule independently of others;
 - Sort final rules into desired sequence for use.

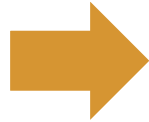
1. Decision Tree

1.4 Extensions to decision tree

Previous Lecture

Continuous Valued Attributes

Spine length
50.08
59.09
103.15
38.06
170.25
47.07
74.11
103.15
42.06



Spine length		
< 50	>=50, <150	>=150
0	1	0
0	1	0
0	1	0
1	0	0
0	0	1
1	0	0
0	1	0
0	1	0
1	0	0

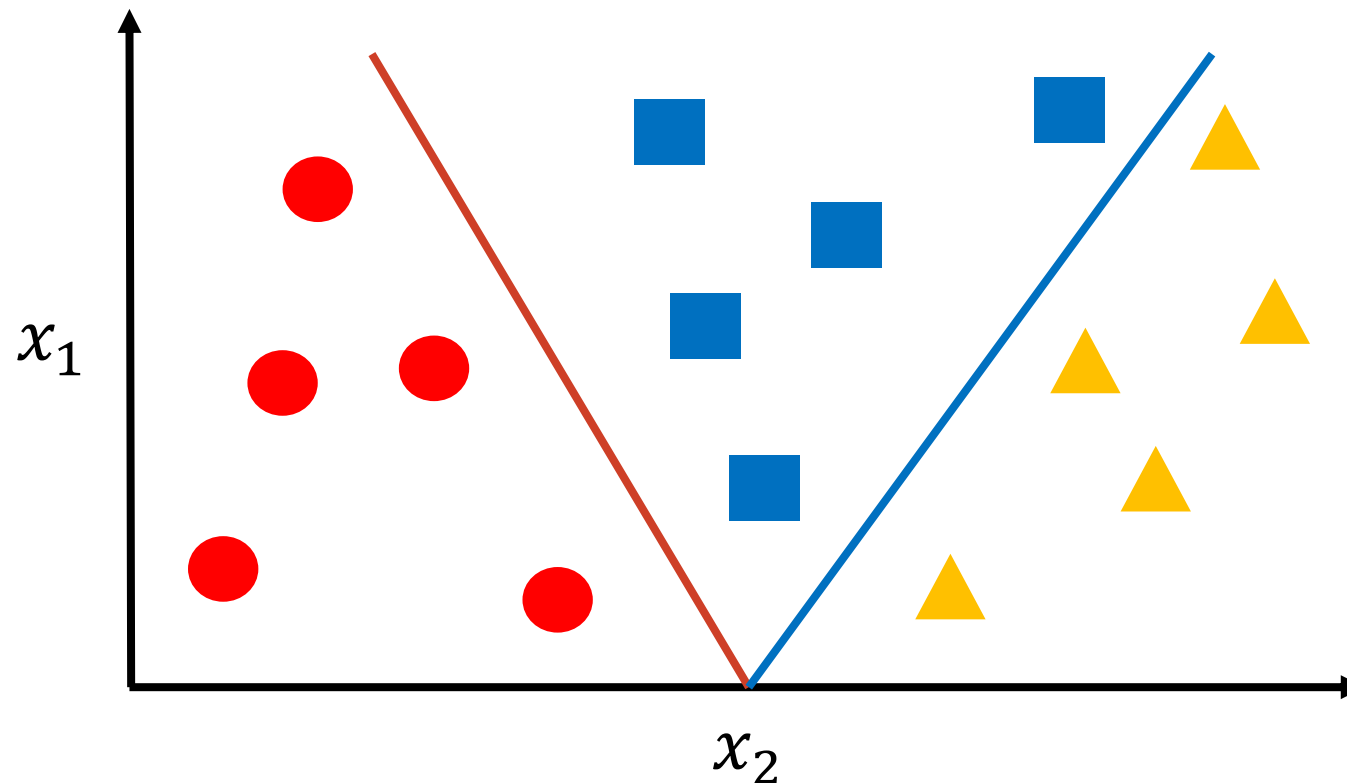
Turn numerical data into categorical: Only need consider 3 values.

1. Decision Tree

1.4 Extensions to decision tree

Multivariate decision tree

How would the decision tree split look like this?



1. Decision Tree

1.4 Extensions to decision tree

Regression trees

1. Decision Tree

1.4 Extensions to decision tree

Unknown/missing Attribute Values

Use training example anyway, sort through tree:

- If node n tests A , assign most common value of A among other examples sorted to node n ;
- Assign most common value of A among other examples with same target value;
- Assign probability to each possible value of A : assign fraction of example to each descendant in tree;

Classify new examples in same fashion.

1. Decision Tree

1.5 Pros and Cons

ID3 – Iterative Dichotomizer 3

One of the earliest decision tree algorithms.

Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar, 1986, 81-106).

- Cannot handle numeric features;
- No pruning, prone to overfitting;
- Short and wide trees (compared to CART);
- Maximizing information gain or minimizing entropy;
- Discrete features, binary and multi-category features.

1. Decision Tree

1.5 Pros and Cons

C4.5

Quinlan, J. R. 1993. C4.5: Programming for machine learning. Morgan Kauffmann, 38, 48.

- Continuous and discrete features;
- Continuous is very expensive, because must consider all possible ranges, bi-partition;
- Can handle missing attributes (ignores them in gain compute);
- Post-pruning (bottom-up pruning);
- Adopt gain ratio.

1. Decision Tree

1.5 Pros and Cons

CART– Classification and Regression Tree

Breiman, L. et al, 1984, Classification and regression trees, Calif: Wadsworth Interational Group.

- Continuous and discrete features;
- Strictly binary splits (taller trees than ID3 and C4.5);
- Binary splits can generate better trees than C4.5, but tend to be larger and harder to interpret;
- Variance reduction in regression trees;
- Adopt Gini impurity.
- Cost complexity pruning.

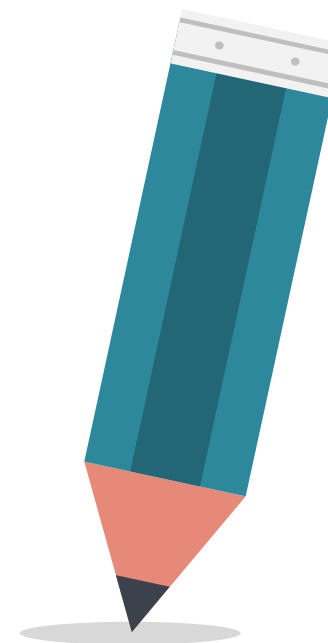
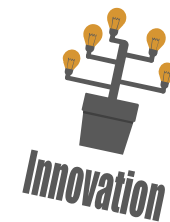
1. Decision Tree

1.5 Pros and Cons

- 😊 Easy to interpret and communicate;
- 😊 Can represent “complete” hypothesis space;
- 😞 Easy to overfit;
- 😞 Elaborate pruning required;
- 😞 Expensive to just fit a “diagonal line”;
- 😞 Output range is bounded (depend on training example) in regression trees.

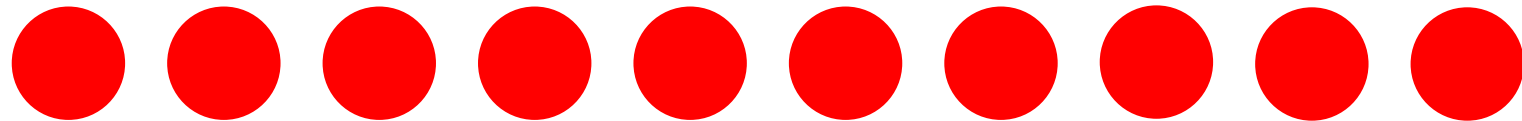
Ensemble learning

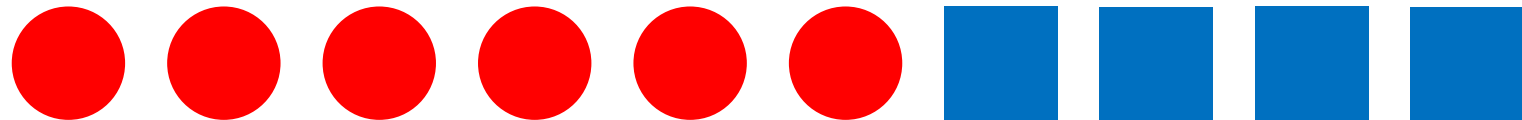
02



2. Ensemble learning

2.1 Voting

 **Unanimity**

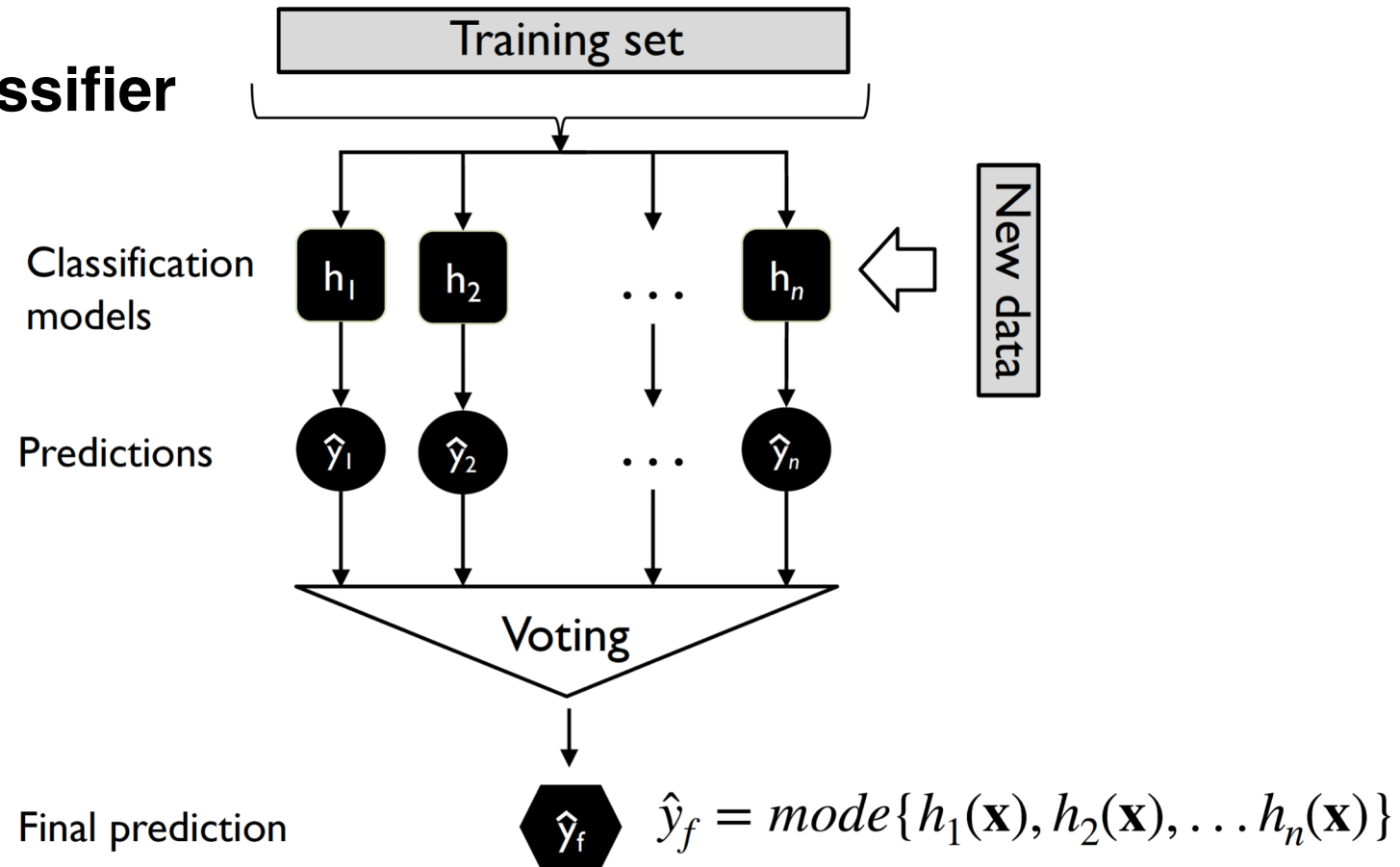
 **Majority**

 **Plurality**

2. Ensemble learning

2.1 Voting

Majority Vote Classifier



where $h_i(\mathbf{x}) = \hat{y}_i$

2. Ensemble learning

2.1 Voting

Majority Vote Classifier

	T ₁	T ₂	T ₃
h ₁	✓	✓	X
h ₂	X	✓	✓
h ₃	✓	X	✓
Ensemble	✓	✓	✓

	T ₁	T ₂	T ₃
h ₁	✓	✓	X
h ₂	✓	✓	X
h ₃	✓	✓	X
Ensemble	✓	✓	X

	T ₁	T ₂	T ₃
h ₁	✓	X	X
h ₂	X	✓	X
h ₃	X	X	✓
Ensemble	X	X	X

Example 1

Example 2

Example 3

2. Ensemble learning

2.1 Voting

Why Majority Vote ?

- Assume n independent classifiers with a base error rate ϵ ;
 - Independent means that the errors are uncorrelated;
- Assume a binary classification task;
- Assume the error rate is better than random guessing;
 - i.e., lower than 0.5 for binary classification.
- The probability that we make a wrong prediction via the ensemble if k classifiers predict the same class label:

$$P(k) = \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k}, k > \lceil n/2 \rceil$$

2. Ensemble learning

2.1 Voting

Why Majority Vote ?

- The probability that we make a wrong prediction via the ensemble if k classifiers predict the same class label:

$$P(k) = \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k}, k > \lceil n/2 \rceil$$

- Ensemble error:

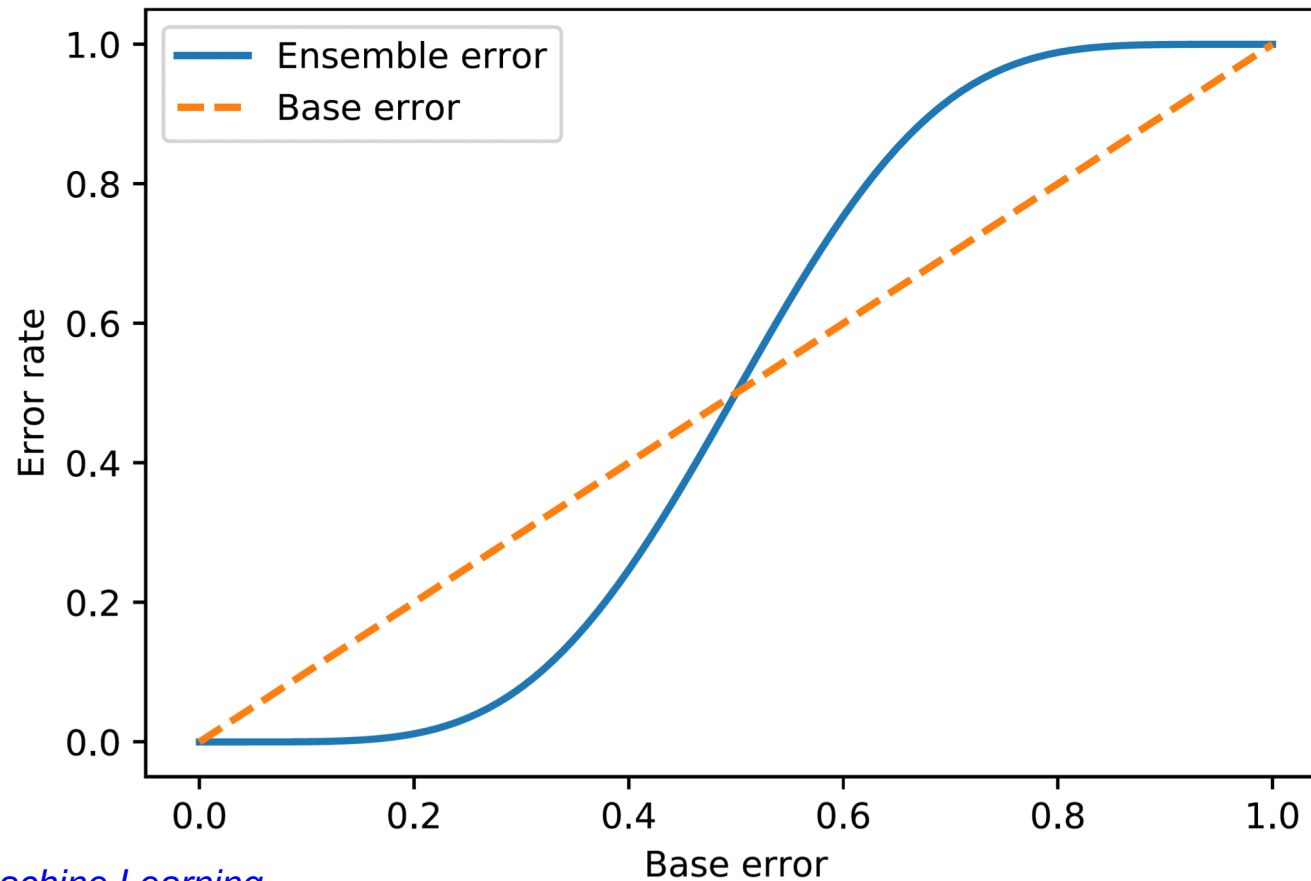
$$\epsilon_{ens} = \sum_k \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k}, k > \lceil n/2 \rceil$$

2. Ensemble learning

2.1 Voting

Why Majority Vote ?

$$\epsilon_{ens} = \sum_k^n \binom{n}{k} \epsilon^k (1 - \epsilon)^{n-k}, k > \lceil n/2 \rceil$$



2. Ensemble learning

2.1 Voting

Hard voting

$$\hat{y} = \arg \max_j \sum_{i=1}^n h_i^j(\mathbf{x})$$

$$H(\mathbf{x}) = \{h_1, h_2, \dots, h_n\}, h_i^j(\mathbf{x}) \in \{0, 1\}$$

$h_i^j(\mathbf{x})$ is the predicted class membership of the i th classifier for class label j

2. Ensemble learning

2.1 Voting

soft voting

Use only for well-calibrated classifiers!

$$\hat{y} = \arg \max_j \sum_{i=1}^n w_i h_i^j(\mathbf{x})$$

$$H(\mathbf{x}) = \{h_1, h_2, \dots, h_n\}, h_i^j(\mathbf{x}) \in [0, 1]$$

$h_i^j(\mathbf{x})$ is the predicted class membership probability of the i th classifier for class label j ; w_i is the optional weighting parameter, $w_i \geq 0, \sum_{i=1}^n w_i = 1$.

2. Ensemble learning

2.1 Voting

soft voting

Use only for well-calibrated classifiers!

$H(x)$	$j = 0$	$j = 1$	w_i
h_1	0.9	0.1	0.2
h_2	0.8	0.2	0.3
h_3	0.3	0.7	0.5

$$\hat{y} = \arg \max_j \sum_{i=1}^n w_i h_i^j(x)$$

$$j \in \{0, 1\}, i \in \{1, 2, 3\}$$

$$p(j = 0|\mathbf{x}) = 0.2 \times 0.9 + 0.3 \times 0.8 + 0.3 \times 0.5 = 0.57$$

$$p(j = 1|\mathbf{x}) = 0.2 \times 0.1 + 0.3 \times 0.2 + 0.3 \times 0.7 = 0.28$$

$$\hat{y} = \arg \max_j \sum_{i=1}^n w_i h_i^j(x) = \arg \max_j \{p(j = 0|\mathbf{x}), p(j = 1|\mathbf{x})\} = 0$$

2. Ensemble learning

2.2 Bagging (Bootstrap Aggregating)

Bagging in Pseudocode

Let n be the number of bootstrap samples

for $i = 1$ to n **do**

 Draw bootstrap sample of size m , D_i ;

 Train base classifier h_i on D_i ;

$\hat{y} = \text{mode}\{h_1, h_2, \dots, h_n\}$

2. Ensemble learning

2.2 Bagging (Bootstrap Aggregating)

Bootstrap Sampling



2. Ensemble learning

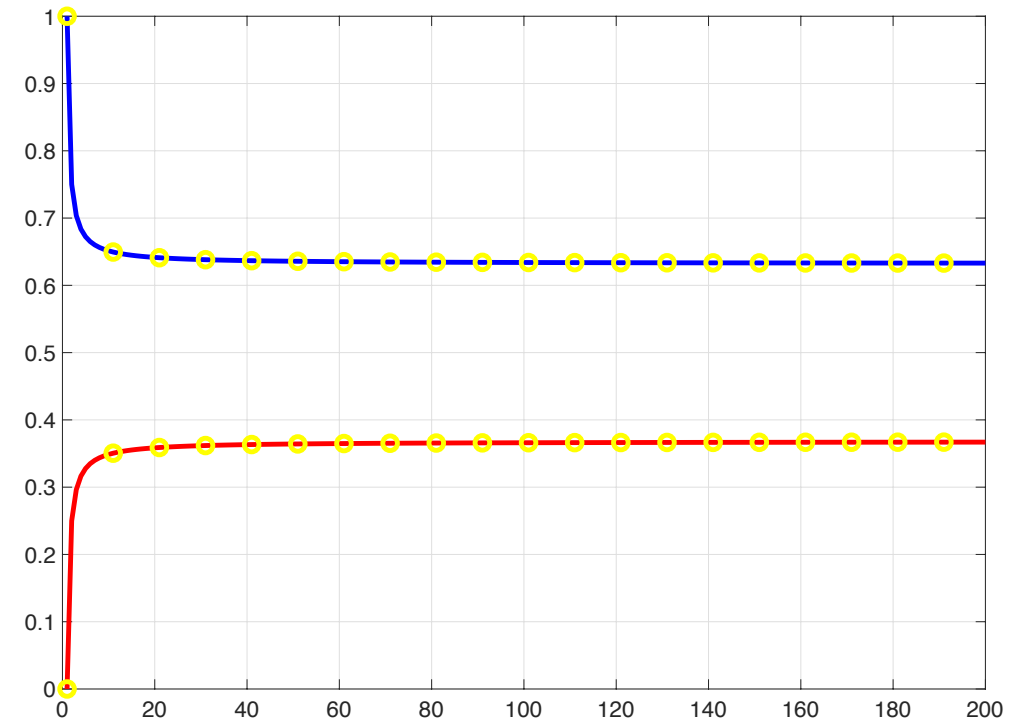
2.2 Bagging (Bootstrap Aggregating)

Bootstrap Sampling

$$P(\text{not chosen}) = \left(1 - \frac{1}{n}\right)^n,$$

$$\frac{1}{e} \approx 0.368, \quad n \rightarrow \infty.$$

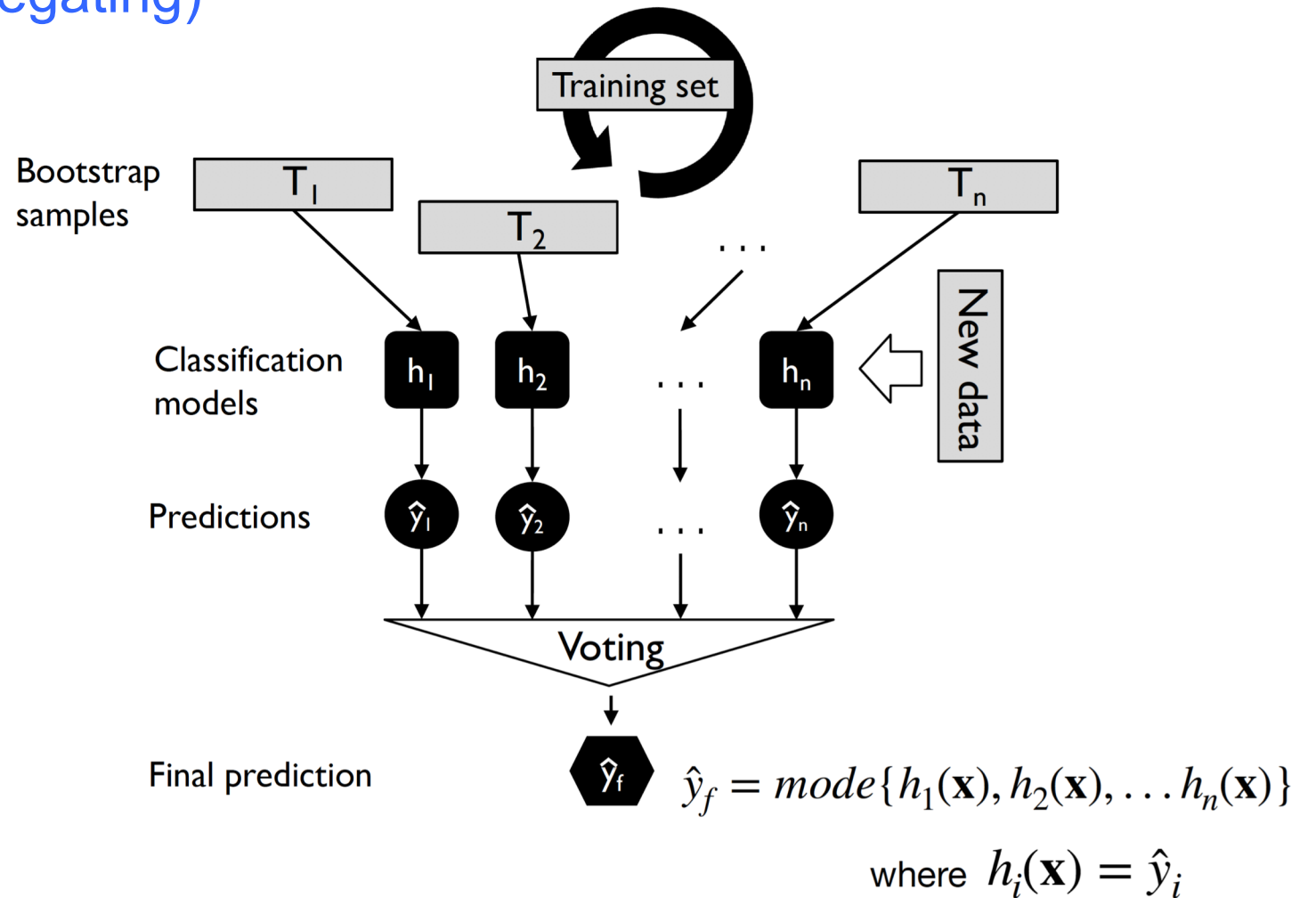
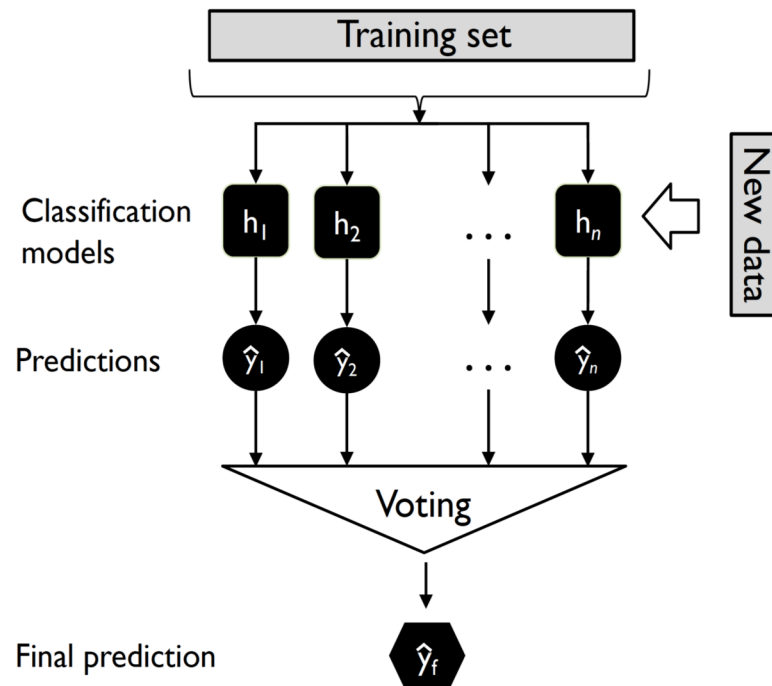
$$P(\text{chosen}) = 1 - \left(1 - \frac{1}{n}\right)^n \approx 0.632$$



2. Ensemble learning

2.2 Bagging (Bootstrap Aggregating)

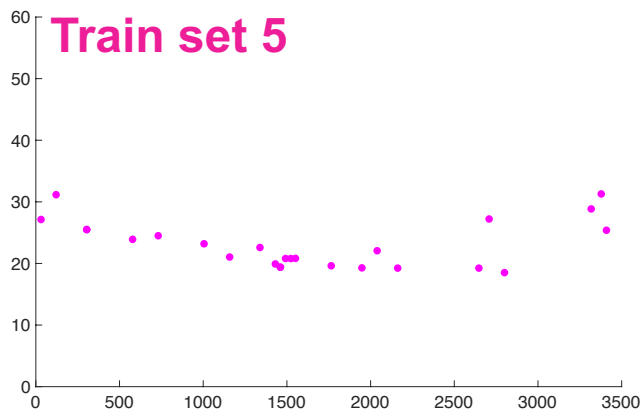
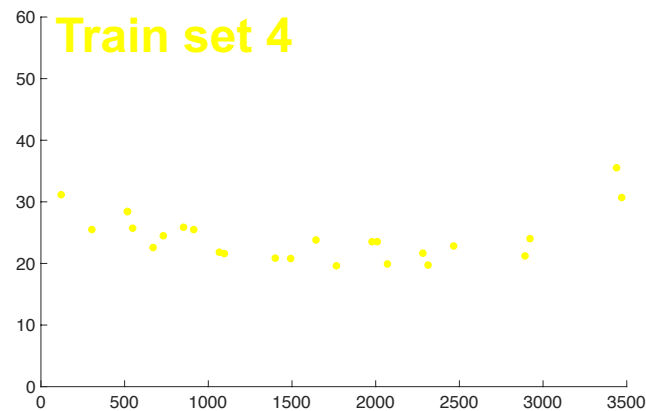
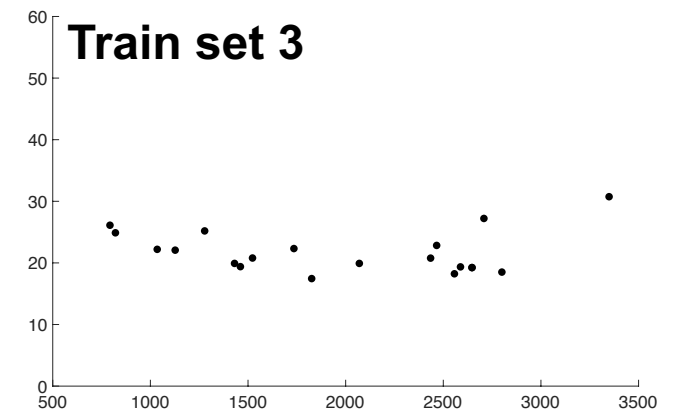
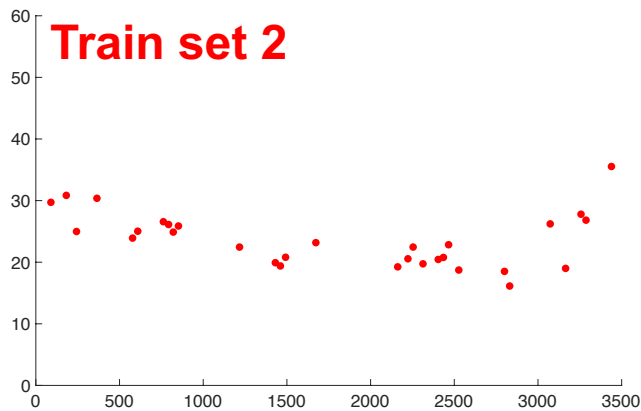
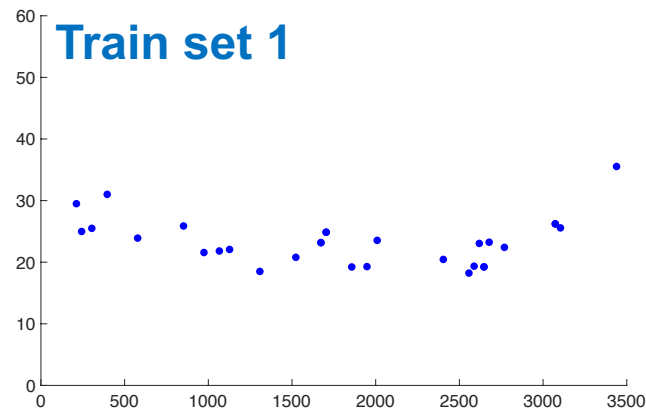
Bagging classifier



2. Ensemble learning

2.2 Bagging (Bootstrap Aggregating)

Previous Lecture

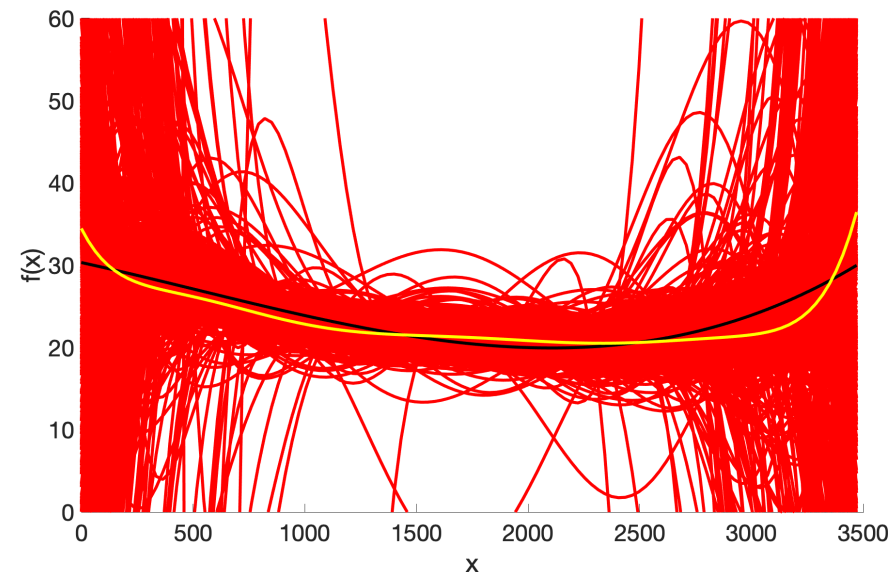
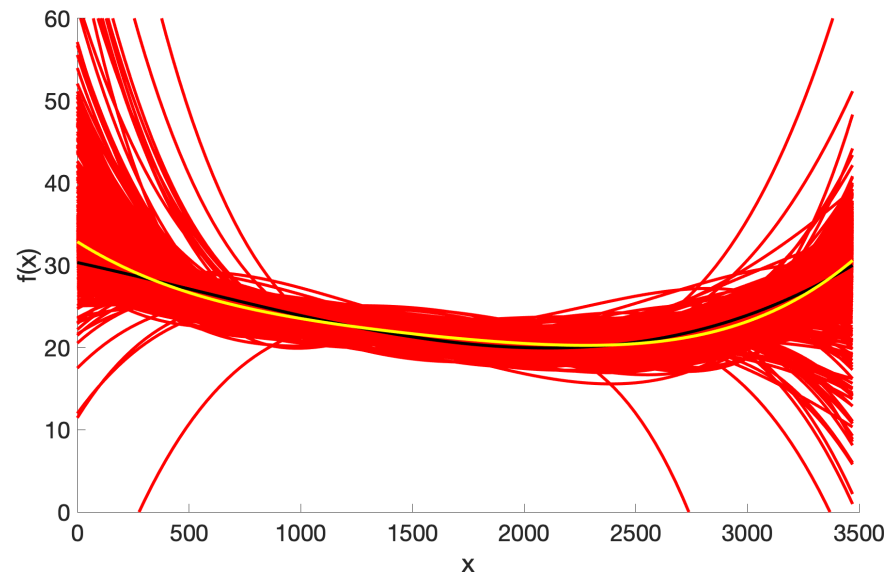


2. Ensemble learning

2.2 Bagging (Bootstrap Aggregating)

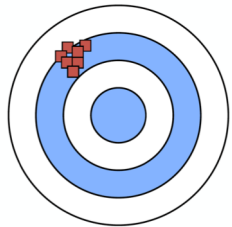
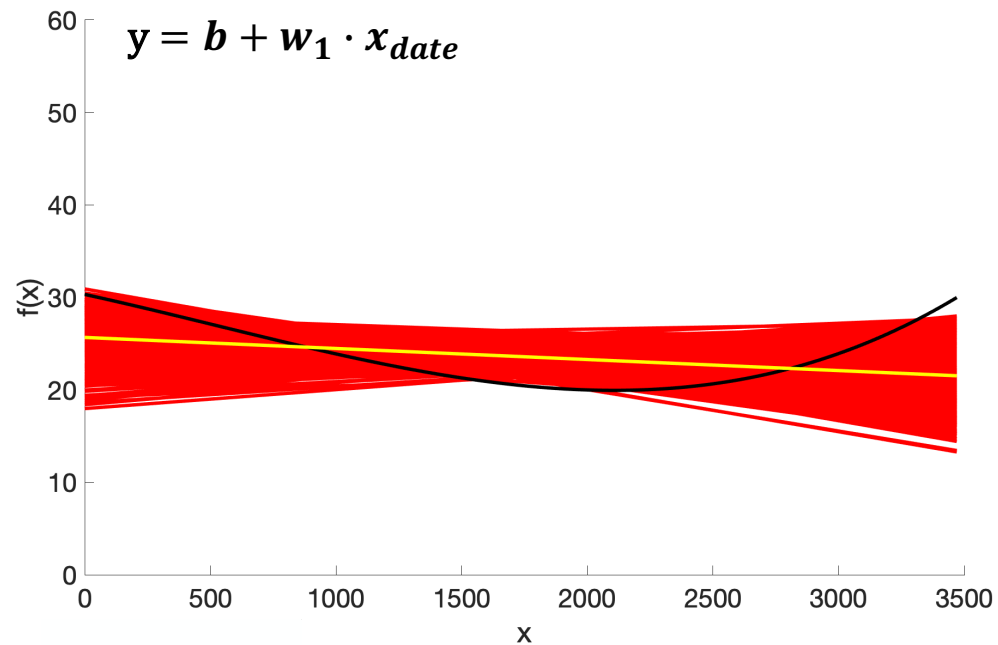
Previous Lecture

- **Black** curve: the true function $\hat{f}$
- **Red** curves: f^* in 1000 training sets
- **Yellow** curve: the average of 1000 $f^* = \bar{f}$



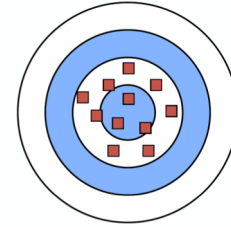
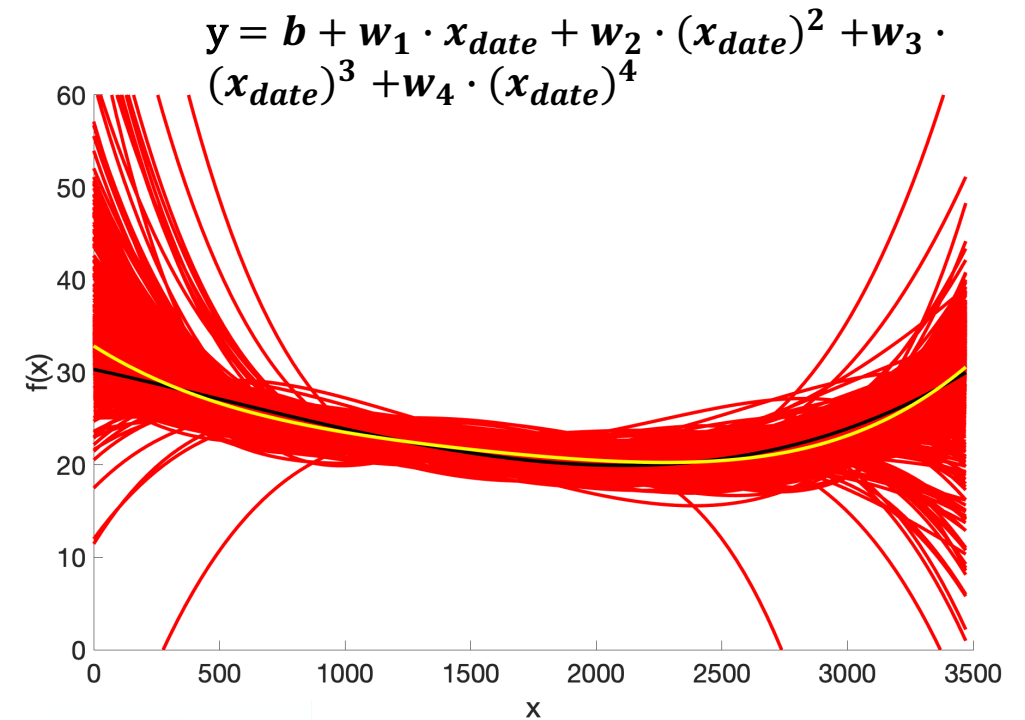
2. Ensemble learning

2.2 Bagging (Bootstrap Aggregating)



Small Variance

Previous Lecture

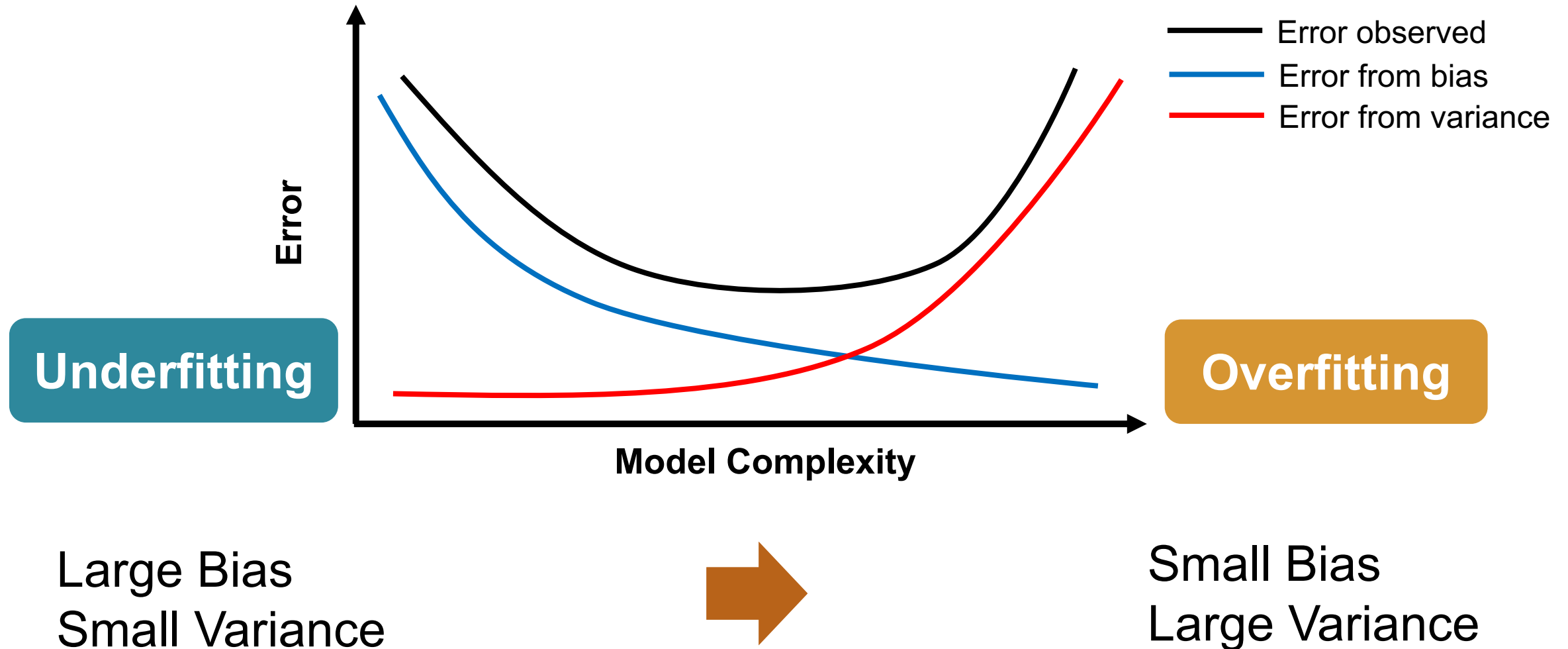


Large Variance

2. Ensemble learning

2.2 Bagging (Bootstrap Aggregating)

Previous Lecture



2. Ensemble learning

2.3 Boosting

- Guarantee :
 - If your ML algorithm can produce classifier with error rate smaller than 50% (accuracy larger than 50%) on training data;
 - You can obtain 0% error rate classifier after boosting.

2. Ensemble learning

2.3 Boosting

General Boosting

- Obtain the first classifier $f_1(x)$;
- Find another function $f_2(x)$ to help $f_1(x)$;
 - However, if $f_2(x)$ is similar to $f_1(x)$, it will not help a lot;
 - We want $f_2(x)$ to be complementary with $f_1(x)$; **How?**
- Obtain the second classifier $f_2(x)$;
- Repeat until the stopping criterion; **When?**
- Finally, combining all the classifiers. **How?**

The classifiers are learned sequentially.

2. Ensemble learning

2.3 Boosting

How to obtain different classifier ?

- Training on different training data sets.
- How to have different training data sets ?
 - Re-sampling your training data to form a new set;
 - Re-weighting your training data to form a new set.

$$(x^1, \hat{y}^1, w^1) \quad w^1 = \cancel{1} \quad 0.5$$

$$(x^2, \hat{y}^2, w^2) \quad w^2 = \cancel{1} \quad 2$$

$$(x^3, \hat{y}^3, w^3) \quad w^3 = \cancel{1} \quad 1.2$$

$$L(f) = \sum_n l(f(x^n), \hat{y}^n)$$



$$L(f) = \sum_n w^n l(f(x^n), \hat{y}^n)$$

In real implementation, you only have to change the cost (objective) function.

2. Ensemble learning

2.3 Boosting

- **Adaptive Boosting**

- e.g., AdaBoost

- **Gradient Boosting**

- LightGBM, XGBoost, scikit-learn's GradientBoostingClassifier

- **Differ mainly in terms of how**

- Weights are updated -- have different training data sets
 - Classifiers are combined

2. Ensemble learning

2.3 Boosting

AdaBoost **Idea : Training $f_2(x)$ on the new training set that fails $f_1(x)$**

ε_1 : the error rate of $f_1(x)$ on its training data,

$$\varepsilon_1 = \frac{\sum_{i=1}^n w_1^i \delta(f_1(x^i) \neq \hat{y}^i)}{Z_1}, Z_1 = \sum_{i=1}^n w_1^i, \varepsilon_1 < 0.5$$

Change the example weights from w_1^i to w_2^i by

$$\frac{\sum_{i=1}^n w_2^i \delta(f_1(x^i) \neq \hat{y}^i)}{Z_2} = 0.5$$

Training $f_2(x)$ based on the new weights w_2^i .

The performance of f_1 for new weights would be random.

2. Ensemble learning

2.3 Boosting

AdaBoost **Idea : Training $f_2(x)$ on the new training set that fails $f_1(x)$**

$$(x^1, \hat{y}^1, w^1) \quad w^1 = 1 \quad \checkmark$$

$$(x^2, \hat{y}^2, w^2) \quad w^2 = 1 \quad \times$$

$$(x^3, \hat{y}^3, w^3) \quad w^3 = 1 \quad \checkmark$$

$$(x^4, \hat{y}^4, w^4) \quad w^4 = 1 \quad \checkmark$$

$$\epsilon_1 = 0.25$$

$f_1(x)$

$$w^1 = 1/\sqrt{3} \quad \checkmark$$

$$w^2 = \sqrt{3} \quad \times$$

$$w^3 = 1/\sqrt{3} \quad \checkmark$$

$$w^4 = 1/\sqrt{3} \quad \checkmark$$

$$\epsilon_1 = 0.5$$

$f_2(x)$

$$\epsilon_2 < 0.5$$

2. Ensemble learning

2.3 Boosting

AdaBoost **Idea : Training $f_2(x)$ on the new training set that fails $f_1(x)$**

If x^i misclassified by f_1 ($f_1(x^i) \neq \hat{y}^i$)

$w_2^i \leftarrow w_1^i$ multiplying d_1

Increase

If x^i correctly classified by f_1 ($f_1(x^i) = \hat{y}^i$)

What is the value of d_1 ?

$w_2^i \leftarrow w_1^i$ divided by d_1

Decrease

f_2 will be learned based on sample weights w_2^i .

2. Ensemble learning

2.3 Boosting

AdaBoost **Idea : Training $f_2(x)$ on the new training set that fails $f_1(x)$**

$$\varepsilon_1 = \frac{\sum_{i=1}^n w_1^i \delta(f_1(x^i) \neq \hat{y}^i)}{Z_1}, Z_1 = \sum_{i=1}^n w_1^i, \varepsilon_1 < 0.5$$

$$\frac{\sum_{i=1}^n w_2^i \delta(f_1(x^i) \neq \hat{y}^i)}{Z_2} = 0.5, Z_2 = \sum_{i=1}^n w_2^i$$

$f_1(x^i) \neq \hat{y}^i, w_2^i \leftarrow w_1^i$ multiplying d_1

$f_1(x^i) = \hat{y}^i, w_2^i \leftarrow w_1^i$ divided by d_1

$$\sum_{i=1}^n w_2^i \delta(f_1(x^i) \neq \hat{y}^i)$$

$$= \sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1$$

$$Z_2 = \sum_{i=1}^n w_2^i = \sum_{f_1(x^i) \neq \hat{y}^i} w_2^i + \sum_{f_1(x^i) = \hat{y}^i} w_2^i$$

$$= \sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1 + \sum_{f_1(x^i) = \hat{y}^i} w_1^i / d_1$$

2. Ensemble learning

2.3 Boosting

AdaBoost **Idea : Training $f_2(x)$ on the new training set that fails $f_1(x)$**

$$\frac{\sum_{i=1}^n w_2^i \delta(f_1(x^i) \neq \hat{y}^i)}{Z_2} = 0.5, \quad \frac{\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1}{\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1 + \sum_{f_1(x^i) = \hat{y}^i} w_1^i / d_1} = 0.5$$

$$\frac{\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1 + \sum_{f_1(x^i) = \hat{y}^i} w_1^i / d_1}{\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1} = 2, \quad \frac{\sum_{f_1(x^i) = \hat{y}^i} w_1^i / d_1}{\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1} = 1$$

$$\sum_{f_1(x^i) = \hat{y}^i} w_1^i / d_1 = \sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1, \quad \frac{1}{d_1} \sum_{f_1(x^i) = \hat{y}^i} w_1^i = d_1 \sum_{f_1(x^i) \neq \hat{y}^i} w_1^i,$$

$$\varepsilon_1 = \frac{\sum_{i=1}^n w_1^i \delta(f_1(x^i) \neq \hat{y}^i)}{Z_1} = \frac{\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i}{Z_1}, \quad \sum_{f_1(x^i) \neq \hat{y}^i} w_1^i = Z_1 \varepsilon_1$$

2. Ensemble learning

2.3 Boosting

AdaBoost **Idea : Training $f_2(x)$ on the new training set that fails $f_1(x)$**

$$\sum_{f_1(x^i)=\hat{y}^i} w_1^i / d_1 = \sum_{f_1(x^i) \neq \hat{y}^i} w_1^i d_1, \quad \frac{1}{d_1} \boxed{\sum_{f_1(x^i)=\hat{y}^i} w_1^i} = d_1 \boxed{\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i}$$

$$\varepsilon_1 = \frac{\sum_{i=1}^n w_1^i \delta(f_1(x^i) \neq \hat{y}^i)}{Z_1} = \frac{\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i}{Z_1},$$

$$\sum_{f_1(x^i) \neq \hat{y}^i} w_1^i = Z_1 \varepsilon_1 \qquad \sum_{f_1(x^i)=\hat{y}^i} w_1^i = Z_1 (1 - \varepsilon_1)$$

$$Z_1 (1 - \varepsilon_1) / d_1 = Z_1 \varepsilon_1 d_1 \qquad d_1 = \sqrt{(1 - \varepsilon_1) / \varepsilon_1} > 1$$

2. Ensemble learning

2.3 Boosting

AdaBoost Algorithm

Initialize k : the number of AdaBoost rounds

Initialize D : the training dataset, $D = \{(x^1, \hat{y}^1), (x^2, \hat{y}^2) \dots (x^n, \hat{y}^n)\}$;

Initialize w_1^i : $w_1^i = \frac{1}{n}, i = 1, 2, \dots, n$;

For $r = 1$ to k **do**

For all i : $w_r^i = \frac{w_r^i}{\sum_i w_r^i}$ Normalize weights

$f_r(x) := \text{FitWeakLearner}(D, w_r)$;

$\varepsilon_r := \sum_i w_r^i \delta(f_r(x^i) \neq \hat{y}^i)$; Compute error

If $\varepsilon_r > 1/2$ **then stop**;

$d_r = \sqrt{(1 - \varepsilon_r)/\varepsilon_r}$; Small if error is large and vice versa

$w_{r+1} := w_r \times \begin{cases} d_r & \text{if } f_r(x^i) \neq \hat{y}^i \\ \frac{1}{d_r} & \text{if } f_r(x^i) = \hat{y}^i \end{cases}$

$$\alpha_r = \frac{1}{2} \ln \sqrt{(1 - \varepsilon_r)/\varepsilon_r};$$

$$w_{r+1} := w_r \times \begin{cases} e^{\alpha_r} & \text{if } f_r(x^i) \neq \hat{y}^i \\ e^{-\alpha_r} & \text{if } f_r(x^i) = \hat{y}^i \end{cases}$$

$$w_{r+1} = w_r \times \exp(-f_r(x^i) \hat{y}^i \alpha_r)$$

Predict: $H(x) = \text{sign}(\sum_r^k \alpha_r f_r(x))$

2. Ensemble learning

2.3 Boosting

■ Stopping Criteria

- Stop if classification is perfect;
- Stop if the error is at least as bad as random guessing (e.g. $\frac{1}{2}$ for binary classification);
- Stop if the number of rounds meets k ;

■ combining all the classifiers

- Uniform weight: $H(x) = \text{sign}(\sum_{r=1}^k f_r(x))$;
- Non-uniform weight: $H(x) = \text{sign}(\sum_{r=1}^k \alpha_r f_r(x))$.

2. Ensemble learning

2.3 Boosting

AdaBoost

Smaller error ε , larger weight for final voting

$$\varepsilon = 0.1 \rightarrow \alpha \approx 0.55$$

$$\varepsilon = 0.4 \rightarrow \alpha \approx 0.10$$

ID	Grayscale (Intensity)	Quiescent Filament	Weight	Updated weight
1	200	1	0.2	0.18
2	185	1	0.2	0.18
3	145	1	0.2	0.18
4	195	0	0.2	0.22
5	120	0	0.2	0.22

$$\varepsilon_r = 0.2 \times 0 + 0.2 \times 0 + 0.2 \times 0 + 0.2 \times 1 + 0.2 \times 1 = \frac{2}{5} = 0.4, \quad r = 1$$

$$\alpha_r = \frac{1}{2} \ln \sqrt{(1 - \varepsilon_r) / \varepsilon_r} = \frac{1}{2} \ln \sqrt{(1 - 0.4) / 0.4} \approx 0.1$$

$$\text{if } f_r(x^i) \neq \hat{y}^i \quad w_{r+1} = w_r e^{\alpha_r} = 0.2 \times e^{0.1} \approx 0.22$$

$$\text{if } f_r(x^i) = \hat{y}^i \quad w_{r+1} = w_r e^{-\alpha_r} = 0.2 \times e^{-0.1} \approx 0.18$$

$$\sum_i w_r^i = 3 \times 0.18 + 2 \times 0.22 = 0.98 \quad \frac{0.18}{0.98} \approx 0.18 \quad \frac{0.22}{0.98} = 0.22$$

2. Ensemble learning

2.3 Boosting

AdaBoost

Decision stump: simple decision tree with 1 splitting rule based on thresholding 1 feature.

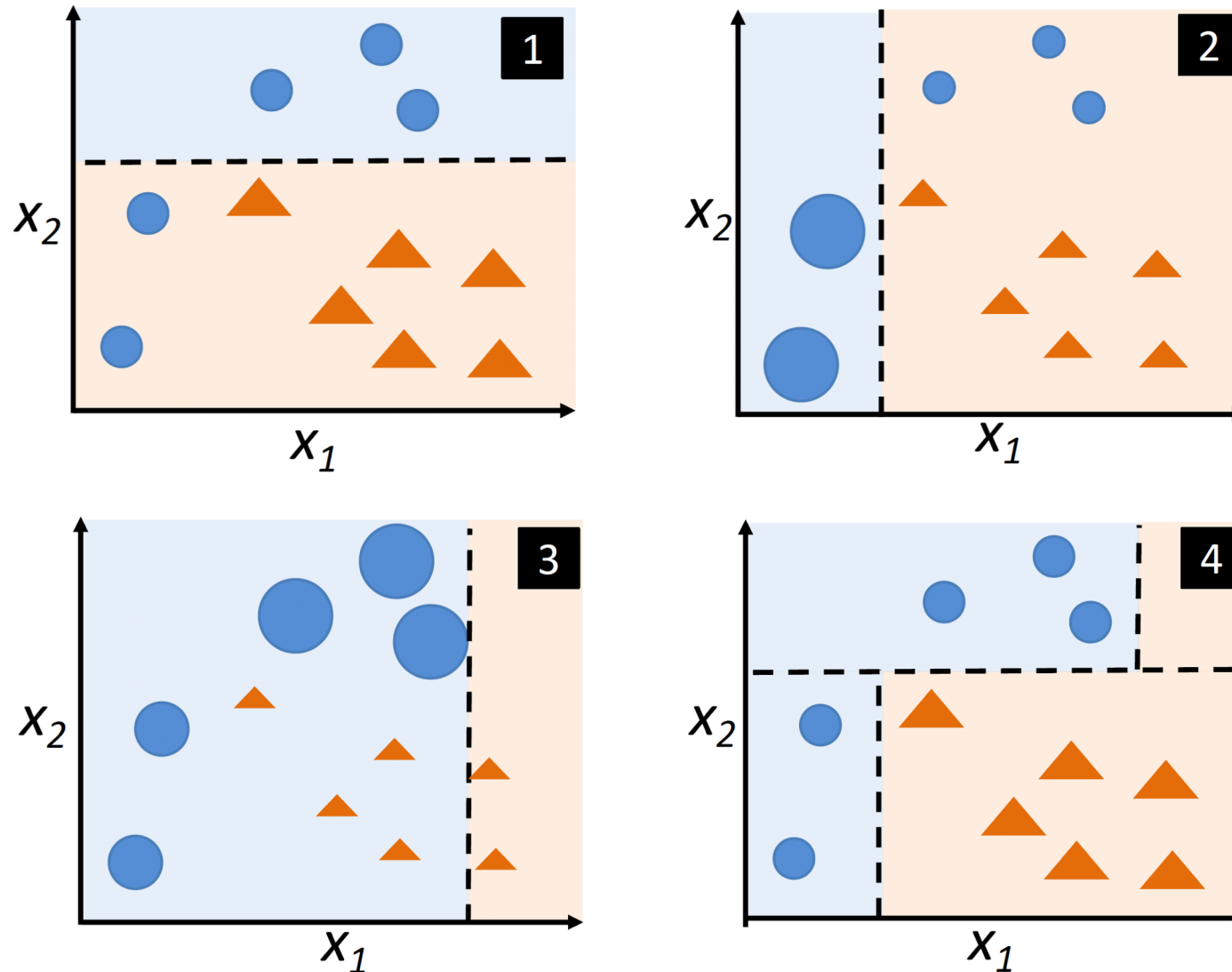
Weak classifier :

Considering decision tree stump for binary classification problem with labels -1, 1

2. Ensemble learning

2.3 Boosting

AdaBoost



2. Ensemble learning

2.3 Boosting

Conceptual overview based on decision trees

Gradient Boosting

- **Step 1:** Construct a base tree (just the root node)
- **Step 2:** Build next tree based on errors of the previous tree
- **Step 3:** Combine tree from step 1 with trees from step 2. Go back to step 2.

2. Ensemble learning

2.3 Boosting

Conceptual overview based on decision trees

Gradient Boosting

X_0 = ID	X_1 = Grayscale	X_2 = Length	X_3 = Barb Number	Y= Latitude
1	200	500	10	60
2	185	450	8	70
3	145	620	12	65
4	195	150	2	30

Step 1: Construct a base tree (just the root node):

$$y_1^* = \frac{1}{n} \sum_{i=1}^n y^i = \frac{1}{4} (60 + 70 + 65 + 30) = 56.25$$

2. Ensemble learning

2.3 Boosting

Conceptual overview based on decision trees

Gradient Boosting

Step 2: Build next tree based on errors of the previous tree:

First, compute (pseudo) residuals, suppose $r_1 = y_1 - y_1^*$

X_0 = ID	X_1 = Grayscale	X_2 = Length	X_3 = Barb Number	Y = Latitude	r_1 = residual
1	200	500	10	60	$60 - 56.25 = 3.75$
2	185	450	8	70	$70 - 56.25 = 13.75$
3	145	620	12	65	$65 - 56.25 = 8.75$
4	195	150	2	30	$30 - 56.75 = -26.75$

2. Ensemble learning

2.3 Boosting

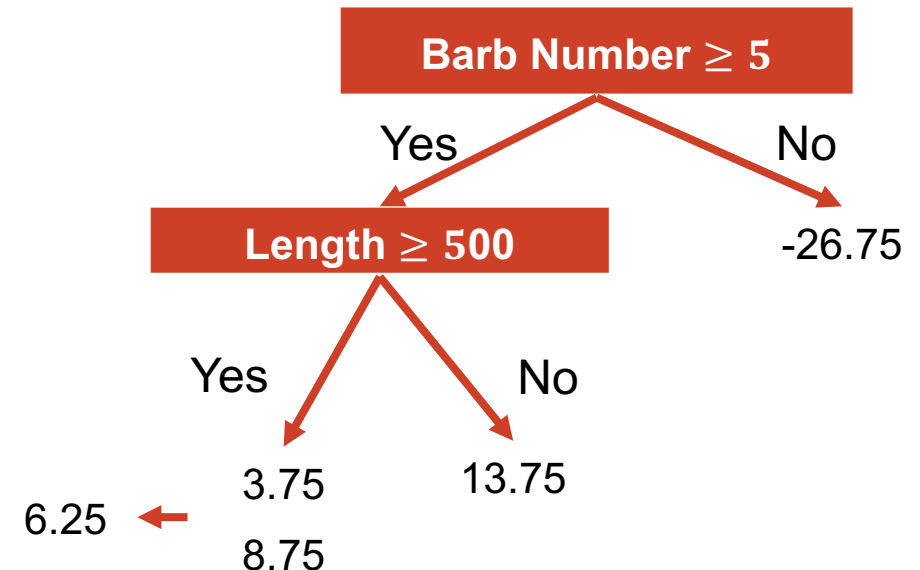
Conceptual overview based on decision trees

Gradient Boosting

Step 2: Build next tree based on errors of the previous tree:

Then, create a tree based on $x_1, x_2, \dots, x_m$ to fit the residuals

X_0 = ID	X_1 = Grayscale	X_2 = Length	X_3 = Barb Number	Y = Latitude	r_1 = residual
1	200	500	10	60	3.75
2	185	450	8	70	13.75
3	145	620	12	65	8.75
4	195	150	2	30	-26.75



2. Ensemble learning

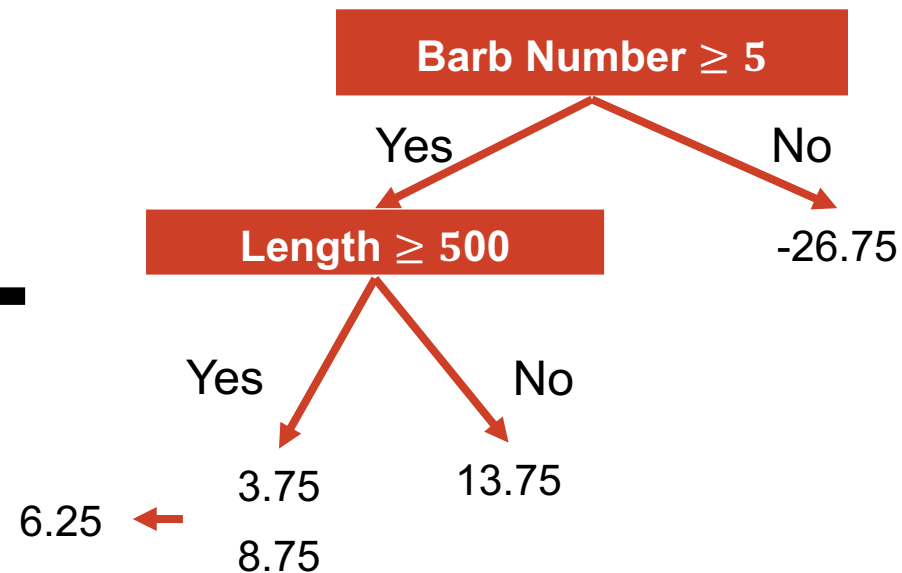
2.3 Boosting

Conceptual overview based on decision trees

Gradient Boosting

Step 3: Combine tree from step 1 with tree:

$$y_1^* = \frac{1}{n} \sum_{i=1}^n y^i = 56.25 \quad +$$



E.g. predict ID=3: $56.25 + \alpha \times 6.25$

where α learning rate between 0 and 1 (If $\alpha = 1$, low bias but high variance)

2. Ensemble learning

2.3 Boosting

Gradient Boosting

Initial function $g_0(x) = 0$

For $t = 1$ **to** T **do**

Find a function $f_t(x)$ and α_t to improve $g_{t-1}(x)$

$$g_{t-1}(x) = \sum_{i=1}^{t-1} \alpha_i f_i(x)$$

$$g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$$

Predict: $H(x) = \text{sign}(g_T(x))$

What is the learning target of $g(x)$?

Maximize $\hat{y}^n g(x^n)$

Minimize
$$L(g) = \sum_n l(\hat{y}^n, g(x^n)) = \sum_n \exp(-\hat{y}^n g(x^n))$$

2. Ensemble learning

2.3 Boosting

Gradient Boosting

Find $g(x)$, minimize $L(g) = \sum_n \exp(-\hat{y}^n g(x^n))$

If we already have $g(x) = g_{t-1}(x)$, how to update $g(x)$?

Gradient Descent:

$$g_t \leftarrow g_{t-1} \boxed{- \eta \nabla L(g_{t-1})}$$

$$- \sum_n \exp(-\hat{y}^n g_{t-1}(x^n)) (-\hat{y}^n)$$

$$g_t(x) = g_{t-1}(x) \boxed{+ \alpha_t f_t(x)}$$

2. Ensemble learning

2.3 Boosting

Gradient Boosting

$$f_t(x) \xleftrightarrow{\text{Same direction}} \sum_n \exp(-\hat{y}^n g_t(x^n)) (\hat{y}^n)$$

We want to find $f_t(x)$ maximizing

$$\sum_n \underbrace{\exp(-\hat{y}^n g_{t-1}(x^n))}_{\text{example weight } u_t^n} \underbrace{(\hat{y}^n) f_t(x^n)}_{\text{Same Sign}}$$

$$w_{r+1} = w_r \times \exp(-f_r(x^i) \hat{y}^i \alpha_r)$$

$$u_t^n = \exp(-\hat{y}^n g_{t-1}(x^n)) = \exp\left(-\hat{y}^n \sum_{i=1}^{t-1} \alpha_i f_i(x^n)\right) = \prod_{i=1}^{t-1} \exp(-\hat{y}^n \alpha_i f_i(x^n))$$

2. Ensemble learning

2.3 Boosting

Gradient Boosting

Find $g(x)$, minimize $L(g) = \sum_n \exp(-\hat{y}^n g(x^n))$

$$g_t(x) = g_{t-1}(x) + \alpha_t f_t(x) \quad \alpha_t \text{ is something like learning rate}$$

Find α_t minimizing $L(g)$

$$\begin{aligned} L(g) &= \sum_n \exp(-\hat{y}^n g(x^n)) = \sum_n \exp(-\hat{y}^n (g_{t-1}(x^n) + \alpha_t f_t(x^n))) = \\ &= \sum_n \exp(-\hat{y}^n (g_{t-1}(x^n))) \exp(-\hat{y}^n (\alpha_t f_t(x^n))) = \\ &= \sum_{f_t(x^n) \neq \hat{y}^n} \exp(-\hat{y}^n g_{t-1}(x^n)) \exp(\alpha_t) + \sum_{f_t(x^n) = \hat{y}^n} \exp(-\hat{y}^n g_{t-1}(x^n)) \exp(-\alpha_t) \end{aligned}$$

Let $\partial L(g) / \partial \alpha_t = 0$

$$\alpha_t = \ln \sqrt{(1 - \varepsilon_r) / \varepsilon_r}$$

2. Ensemble learning

2.4 Random Forests

Trees VS Forests

- A single tree may over fit to the training data;
- Instead train multiple trees and combine their predictions;
- Each tree can differ in both training data and node tests;
- Achieve this by injecting randomness into training algorithm.

2. Ensemble learning

2.4 Random Forests

Random Forests =
Bagging of decision trees +
random feature subsets

2. Ensemble learning

2.4 Random Forests

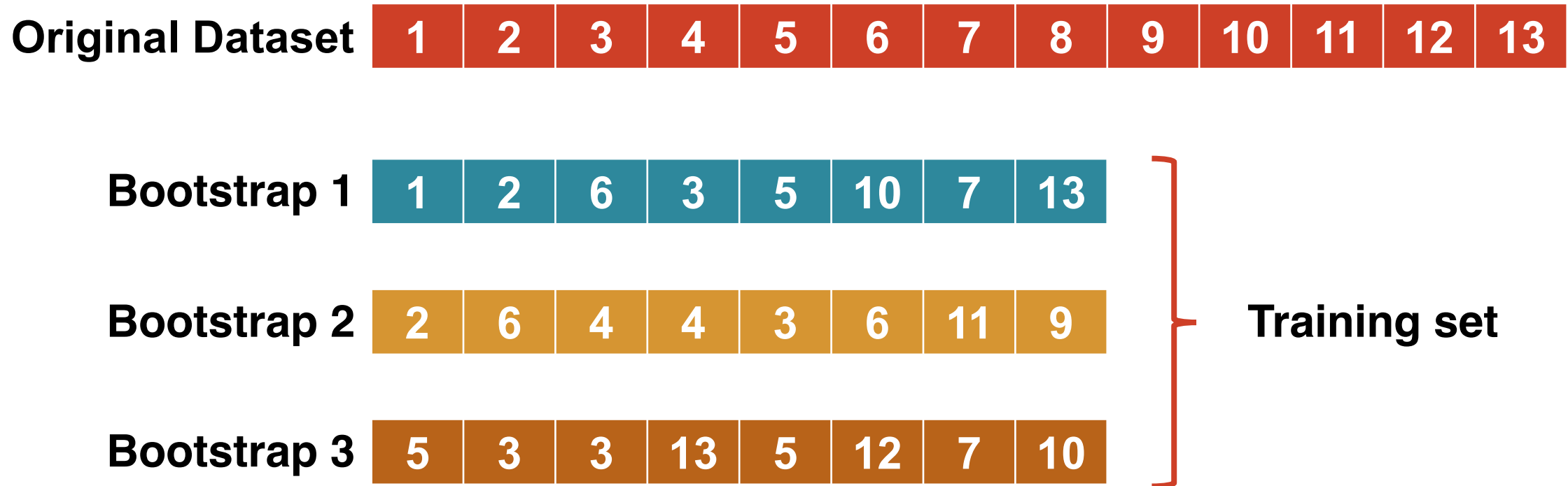
Bootstrap Sampling

ID	Grayscale (Intensity)	Latitude	Location	Tilt Angle	Barb Number	Quiescent Filament
1	gray	high	northern hemisphere	small	many	Yes
2	dark gray	high	northern hemisphere	medium	many	Yes
3	dark gray	high	northern hemisphere	medium	many	Yes
4	gray	high	southern hemisphere	medium	many	Yes
5	light gray	middle	southern hemisphere	small	some	Yes
6	gray	middle	southern hemisphere	medium	some	yes
7	dark gray	middle	southern hemisphere	medium	some	No
8	dark gray	middle	northern hemisphere	large	few	No
9	light gray	middle	northern hemisphere	medium	few	No
10	gray	high	southern hemisphere	medium	some	No
11	light gray	low	northern hemisphere	medium	few	No
12	dark gray	middle	southern hemisphere	medium	many	No
13	gray	low	southern hemisphere	large	few	No

2. Ensemble learning

2.4 Random Forests

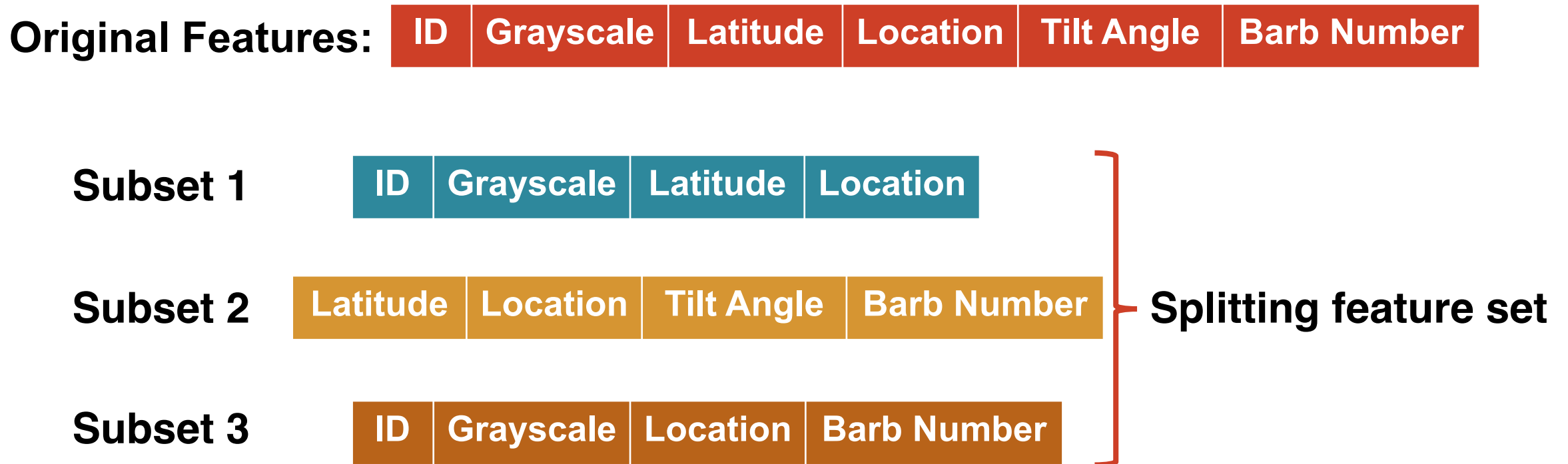
Bootstrap Sampling



2. Ensemble learning

2.4 Random Forests

Random feature subset for each tree of node:



Number of features in subset= $\log_2 m + 1$, where m is the number of input features. 106

2. Ensemble learning

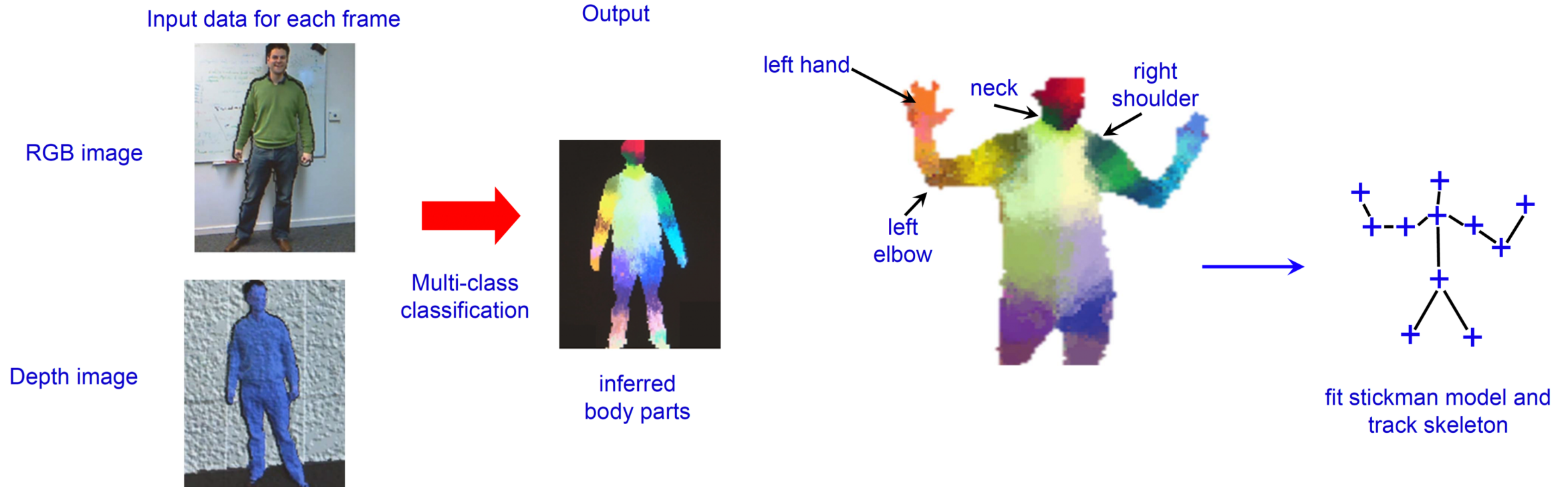
2.4 Random Forests

- 😊 Random forests average a set of deep decision trees;
 - Tend to be one of the best “out of the box” classifiers;
 - Often close to the best performance of any method on the first run;
- 😊 Predictions are very fast;
 - E.g. compare with Bagging considering all features, random forest only considering a subset;
- 😞 Many parameters: depth of tree, number of trees, type of node tests, random sampling;
- 😞 Requires a lot of training data and large memory footprint.

2. Ensemble learning

2.4 Random Forests

Application--Body tracking in Microsoft Kinect for XBox 360

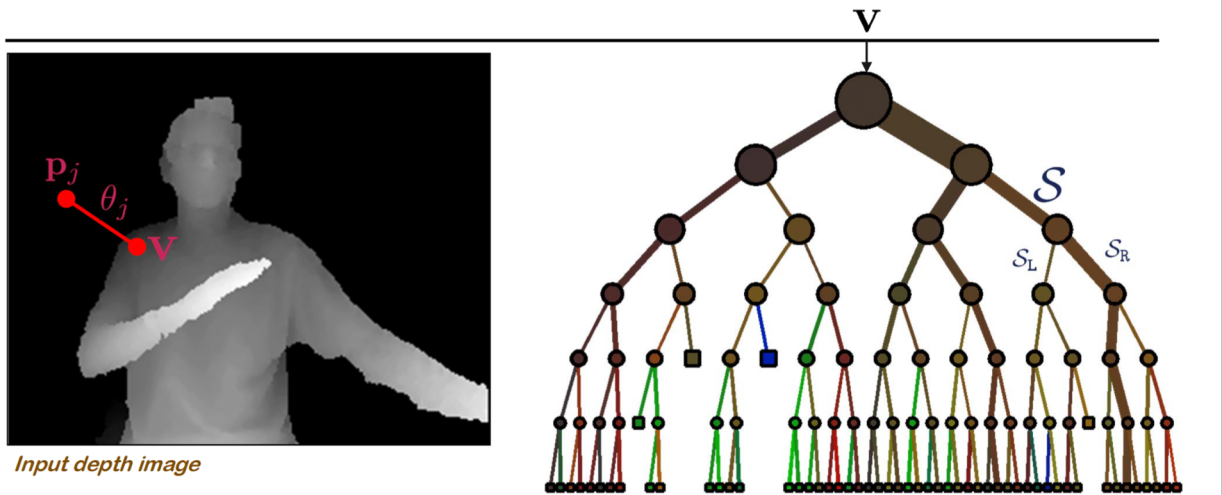


2. Ensemble learning

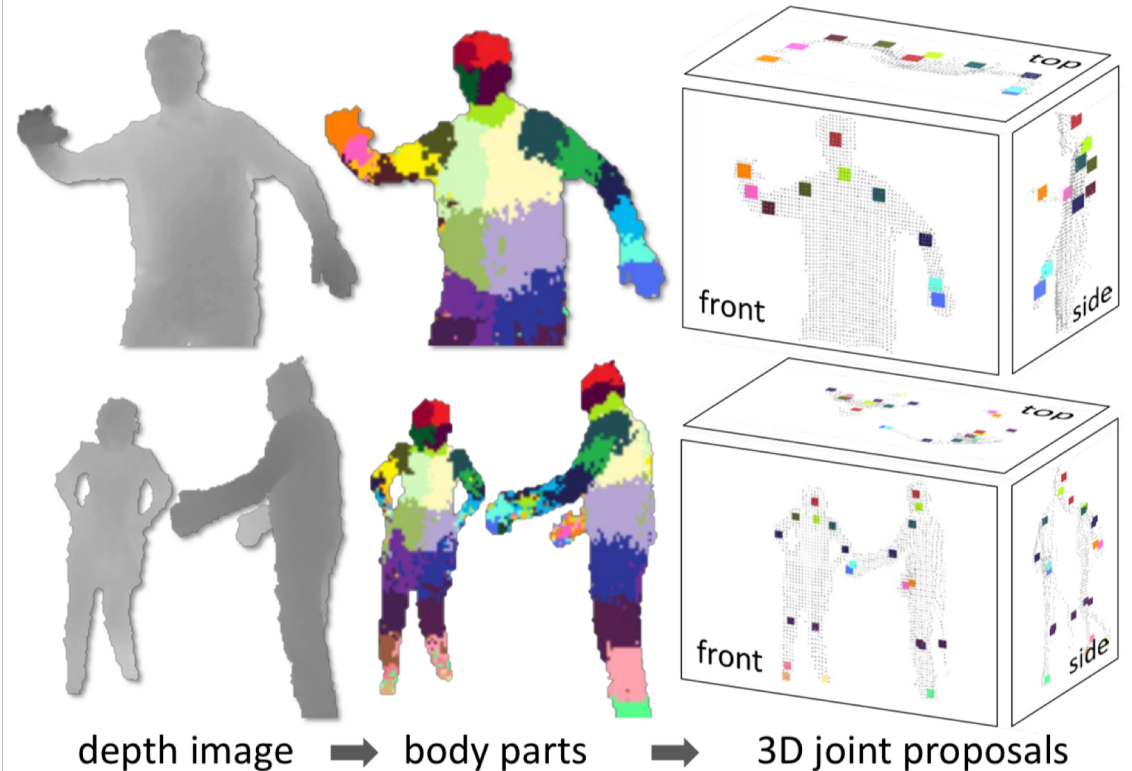
2.4 Random Forests

Application – Body tracking in Microsoft Kinect for XBox 360

Node tests



Input data point	$\mathbf{v} = (x \ y)$	(pixel position in 2D image)
Output	$c \in \{\text{left - hand, right - hand, head, r. - foot, r. - shoulder...}\}$	
Feature response	$f(\mathbf{v}, \theta_j) = I(\mathbf{v}) - I(\mathbf{p}_j)$	(depth difference between two points)



2. Ensemble learning

2.4 Random Forests

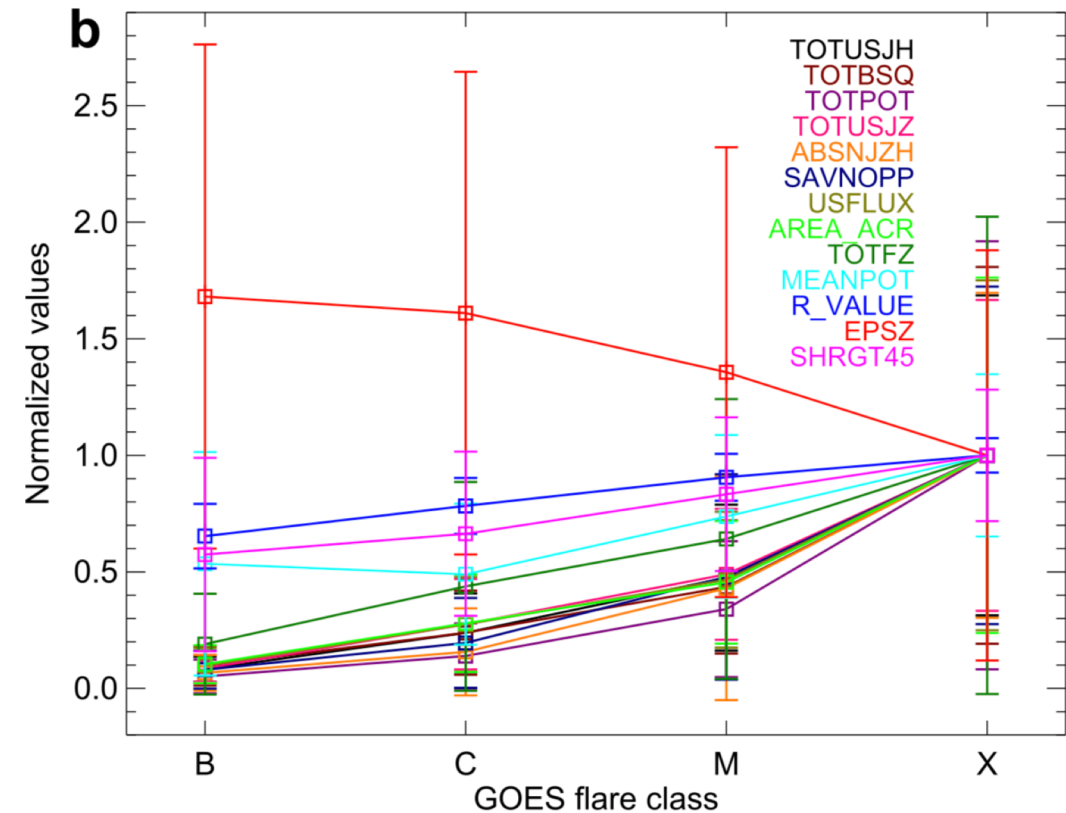
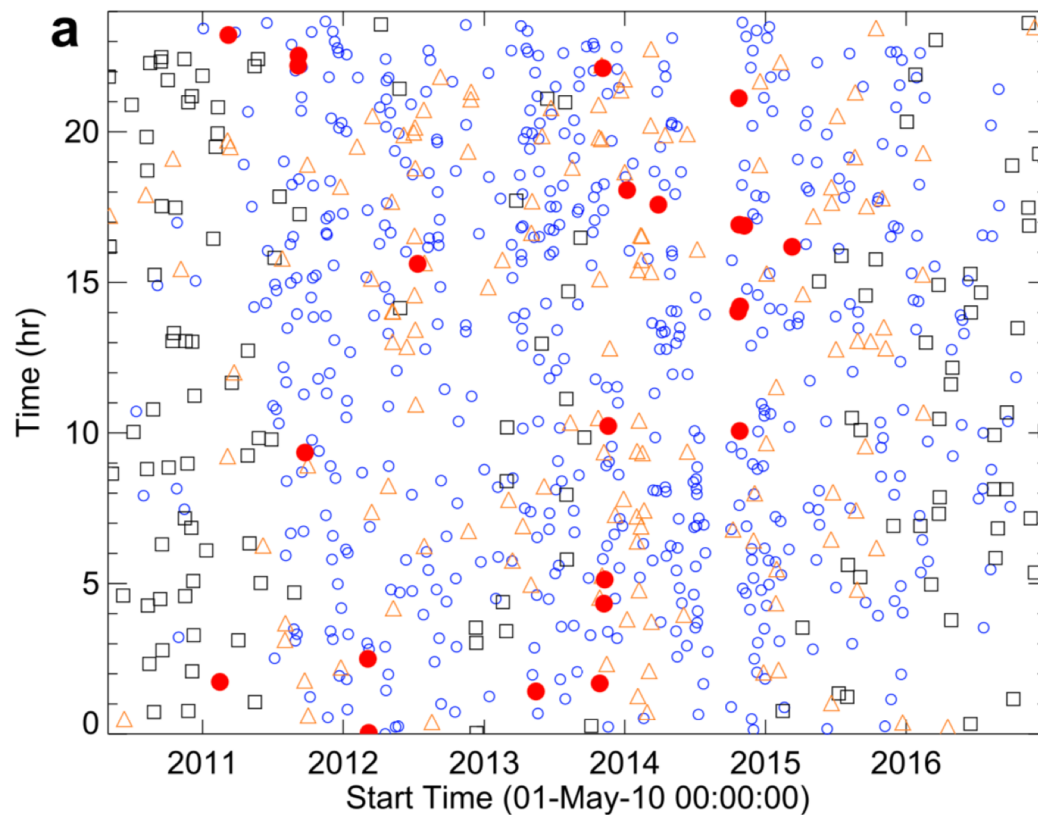
Application—Predict solar flares

SHARP Keyword ^a	Formula	Unit	RF Importance ^b	B Class ($n = 128$)	C Class ($n = 552$)	M Class ($n = 142$)	X Class ($n = 23$)
TOTUSJH	$H_{\text{c total}} \propto \sum B_z \cdot J_z $	$10^2 \text{ G}^2 \text{ m}^{-1}$	37.4	4.8 ± 3.1	13.9 ± 9.9	27.7 ± 18.2	58.3 ± 40.0
TOTBSQ	$F \propto \sum B^2$	10^{10} G^2	17.9	1.0 ± 0.9	2.6 ± 1.9	4.6 ± 3.0	10.7 ± 8.6
TOTPOT	$\rho_{\text{tot}} \propto \sum (\mathbf{B}^{\text{Obs}} - \mathbf{B}^{\text{Pot}})^2 dA$	$10^{23} \text{ ergs cm}^{-3}$	21.1	1.0 ± 1.4	2.7 ± 2.7	6.7 ± 5.7	19.6 ± 18.0
TOTUSJZ	$J_{\text{z total}} = \sum J_z dA$	10^{12} A	50.6	9.5 ± 6.4	30.3 ± 21.4	53.9 ± 30.9	110.0 ± 73.4
ABSNJZH	$H_{\text{c abs}} \propto \sum B_z \cdot J_z $	$10 \text{ G}^2 \text{ m}^{-1}$	19.9	6.1 ± 7.0	14.3 ± 17.0	39.2 ± 43.8	91.2 ± 63.6
SAVNCPP	$J_{\text{z sum}} \propto \sum B_z^+ J_z dA + \sum B_z^- J_z dA $	10^{12} A	24.6	2.7 ± 2.7	6.5 ± 6.4	15.8 ± 14.6	33.1 ± 24.0
USFLUX	$\Phi = \sum B_z dA$	10^{21} Mx	14.2	7.1 ± 5.5	19.9 ± 14.7	33.7 ± 21.0	72.2 ± 54.2
AREA_ACR	Area = $\sum$ Pixels	10^2 pixels	23.7	3.0 ± 2.4	8.2 ± 6.1	13.3 ± 7.7	29.2 ± 22.3
TOTFZ	$F_z \propto \sum (B_x^2 + B_y^2 - B_z^2) dA$	-10^{23} dyne	13.9	1.2 ± 1.3	2.7 ± 2.7	3.9 ± 3.7	6.1 ± 6.2
MEANPOT	$\bar{\rho} \propto \frac{1}{N} \sum (\mathbf{B}^{\text{Obs}} - \mathbf{B}^{\text{Pot}})^2$	$10^3 \text{ ergs cm}^{-3}$	19.8	6.5 ± 5.8	5.9 ± 3.7	8.9 ± 4.2	12.1 ± 4.2
R_VALUE	$\Phi = \sum B_{\text{LOS}} dA$ within R mask	Mx	31.4	3.2 ± 0.7	3.8 ± 0.6	4.4 ± 0.5	4.9 ± 0.4
EPSZ	$\delta F_z \propto \frac{\sum (B_x^2 + B_y^2 - B_z^2)}{\sum B^2}$	-10^{-1}	15.4	2.1 ± 1.3	2.0 ± 1.3	1.7 ± 1.2	1.2 ± 1.1
SHRGT45	Area with shear $> 45^\circ$ / Total Area	...	12.7	0.23 ± 0.17	0.27 ± 0.14	0.34 ± 0.13	0.40 ± 0.11

2. Ensemble learning

2.4 Random Forests

Application—Predict solar flares



2. Ensemble learning

2.4 Random Forests

Application—Predict solar flares

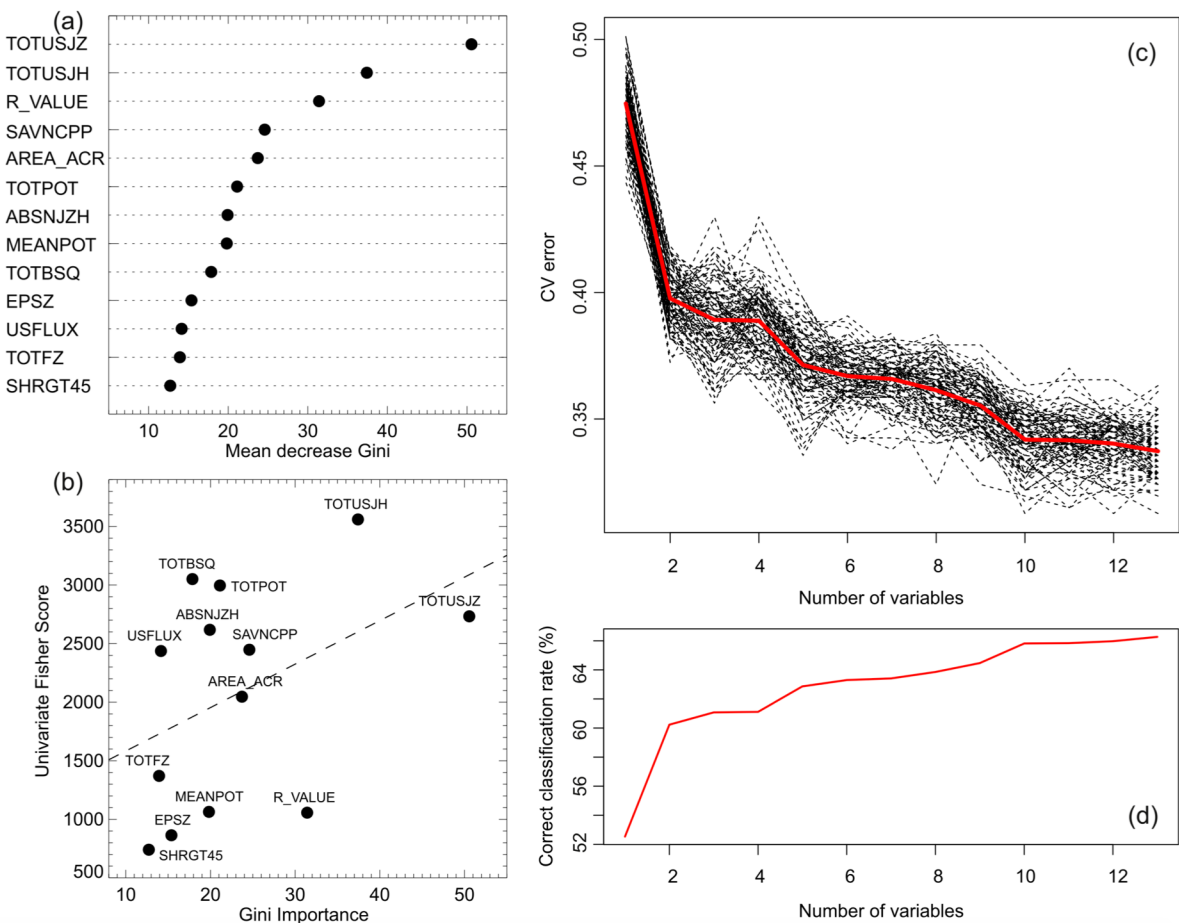


Table 4
RF Binary-class Flare Prediction Results (within 24 hr) Using 13 *SDO*/HMI Parameters and Comparison to Other Studies

Prediction ↓ Observation →	B/C Class (<i>n</i> = 165)	M/X Class (<i>n</i> = 165)
B/C Class	129.536 ± 4.025	41.722 ± 3.370
M/X Class	35.464 ± 4.025	123.278 ± 3.370
Recall: This work	0.785 ± 0.036	0.747 ± 0.030
Bloomfield et al. (2012)	N/A	0.704
Ahmed et al. (2013)	N/A	0.523
Bobra & Couvidat (2015)	N/A	0.832 ± 0.042
Nishizuka et al. (2017)	N/A	0.716
Precision: This work	0.756 ± 0.033	0.777 ± 0.033
Bloomfield et al. (2012)	N/A	0.146
Ahmed et al. (2013)	N/A	0.740
Bobra & Couvidat (2015)	N/A	0.417 ± 0.037
Nishizuka et al. (2017)	N/A	0.969
Accuracy: This work	0.766 ± 0.023	0.766 ± 0.021
Bloomfield et al. (2012)	N/A	0.830
Ahmed et al. (2013)	N/A	0.963
Bobra & Couvidat (2015)	N/A	0.924 ± 0.007
Nishizuka et al. (2017)	N/A	0.990
TSS: This work	0.532 ± 0.036	0.532 ± 0.030
Bloomfield et al. (2012)	N/A	0.539
Ahmed et al. (2013)	N/A	0.512
Bobra & Couvidat (2015)	N/A	0.761 ± 0.039
Nishizuka et al. (2017)	N/A	0.71 ± 0.002

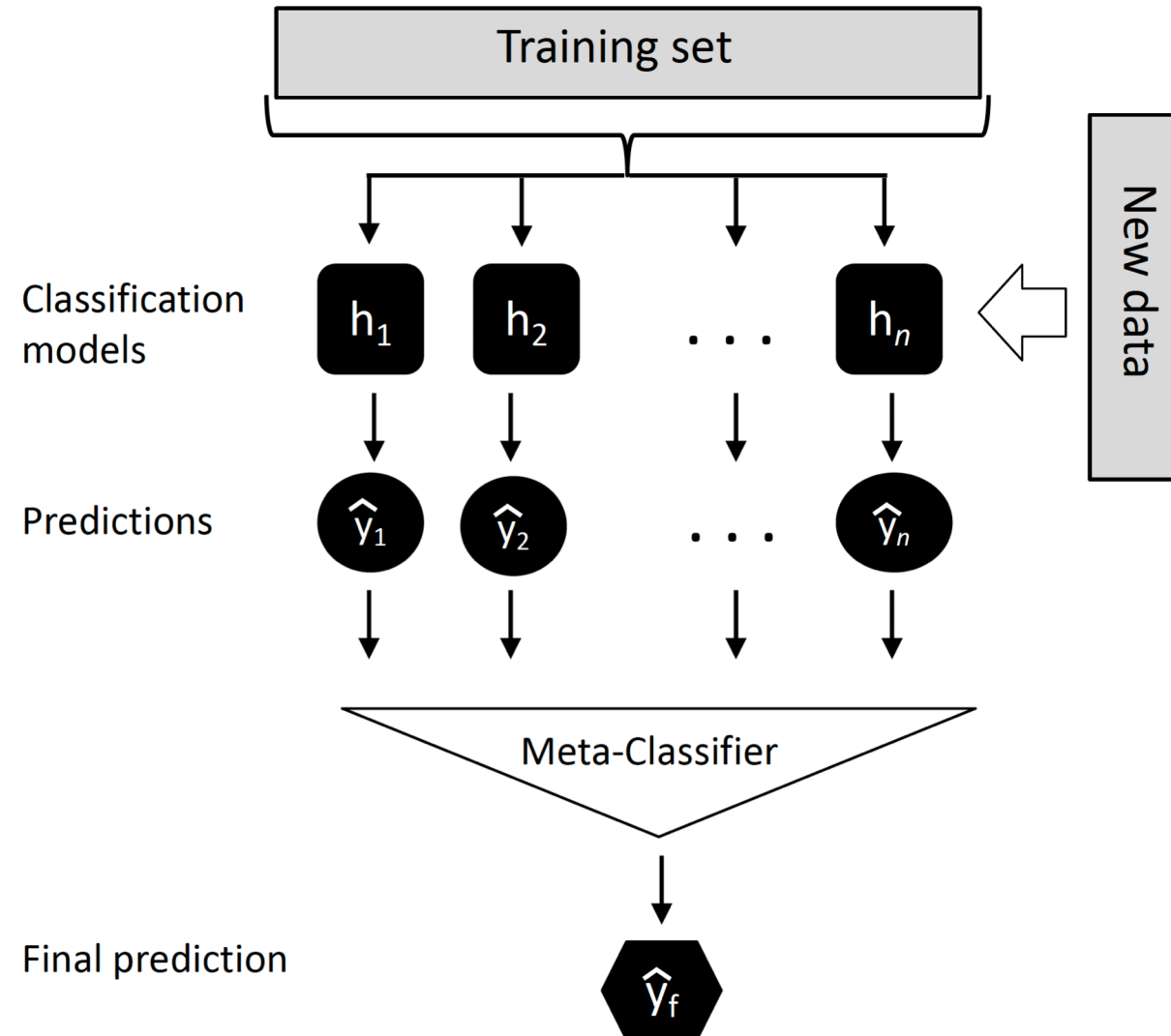
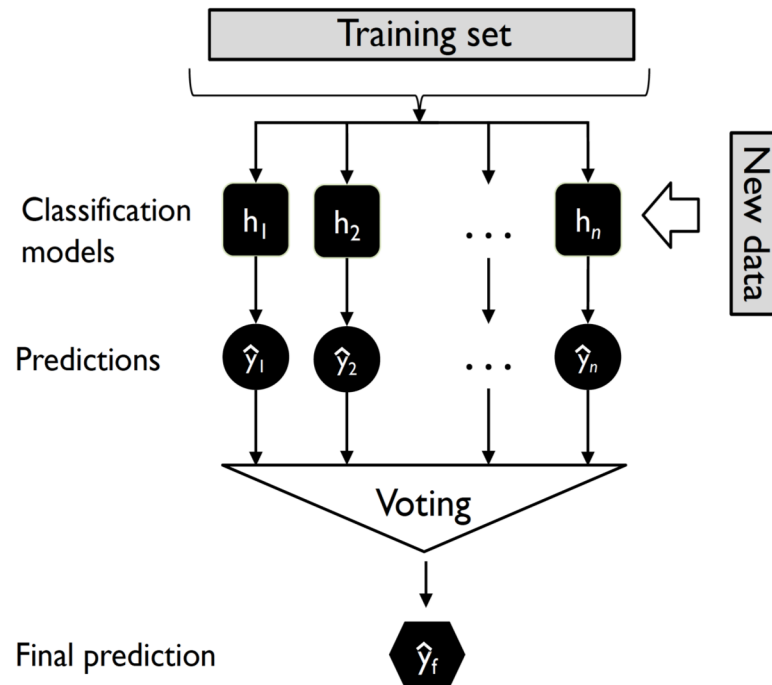
2. Ensemble learning

2.5 Stacking

**Fit another classifier that uses
the predictions as features.**

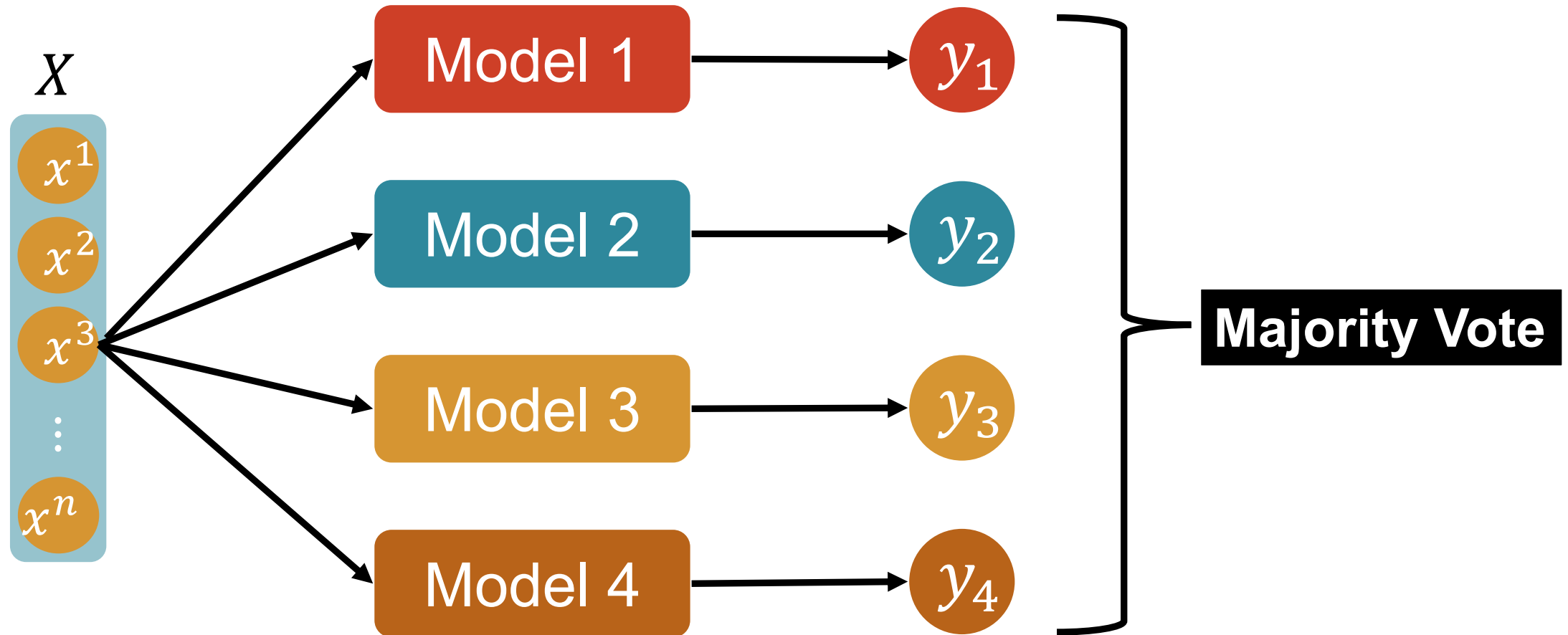
2. Ensemble learning

2.5 Stacking



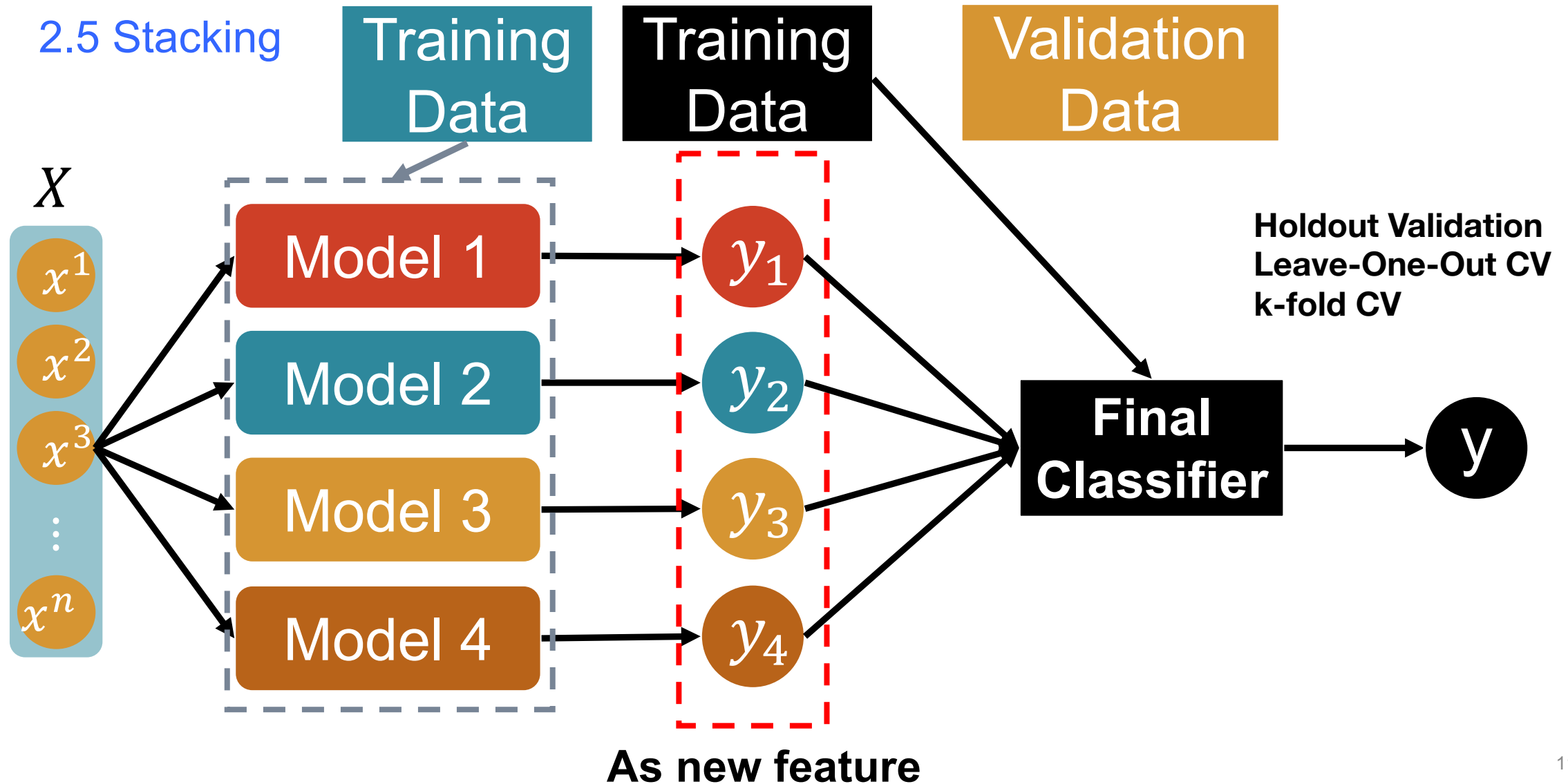
2. Ensemble learning

2.5 Stacking



2. Ensemble learning

2.5 Stacking



To learn more

Mlxtend (machine learning extensions)

<http://rasbt.github.io/mlxtend/#examples>
<http://rasbt.github.io/mlxtend/installation/>
<https://anaconda.org/conda-forge/mlxtend>



Open Terminal:
>>Anaconda prompt
>>conda install -c conda-forge mlxtend

conda-forge / packages / mlxtend 0.17.3

Machine Learning Library Extensions

Conda	Files	Labels	Badges
License: BSD v3 143495 total downloads Last upload: 3 months and 21 days ago			

Installers

Info: This package contains files in non-standard labels.

conda install ?

Platform	Version
linux-64	v0.12.0
win-32	v0.12.0
noarch	v0.17.3
osx-64	v0.12.0
win-64	v0.12.0

To install this package with conda run one of the following:

```
conda install -c conda-forge mlxtend
conda install -c conda-forge/label/gcc7 mlxtend
conda install -c conda-forge/label/cf201901 mlxtend
conda install -c conda-forge/label/cf202003 mlxtend
```

```
haoqi ~ -bash — 129x36
~ — jupyter_mac.command — -bash
Last login: Mon Oct 12 17:39:56 on ttys002
(base) haoqi@Mac-Pro:~ haoqi$ Anaconda Prompt
usage: Anaconda [-h] [--disable-ssl-warnings] [--show-traceback] [-v] [-q]
               [-V] [-t TOKEN] [-s SITE]
...
Anaconda error: argument : invalid choice: 'Prompt' (choose from 'auth', 'label', 'channel', 'config', 'copy', 'download', 'groups', 'login', 'logout', 'move', 'notebook', 'package', 'remove', 'search', 'show', 'upload', 'whoami')
(base) haoqi@Mac-Pro:~ haoqi$ conda install -c conda-forge mlxtend
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /opt/anaconda3

added / updated specs:
- mlxtend

The following packages will be downloaded:

package | build | size | channel
-----|-----|-----|-----
ca-certificates-2020.6.20 | hecda079_0 | 146 KB | conda-forge
certifi-2020.6.20 | py37h2987424_2 | 151 KB | conda-forge
conda-4.9.0 | py37h2987424_1 | 3.0 MB | conda-forge
mlxtend-0.17.3 | pyh9f0ad1d_0 | 1.2 MB | conda-forge
openssl-1.1.1h | haf1e3a3_0 | 1.9 MB | conda-forge
-----|-----|-----|-----
Total: | 6.4 MB

The following NEW packages will be INSTALLED:

mlxtend conda-forge/noarch:mlxtend-0.17.3-pyh9f0ad1d_0

The following packages will be UPDATED:
```

To learn more

Mlxtend (machine learning extensions)

```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from mlxtend.plotting import plot_decision_regions

tree = DecisionTreeClassifier(criterion='entropy',
                             max_depth=2,
                             random_state=1)
tree.fit(X_train, y_train)

plot_decision_regions(X_train, y_train, tree)

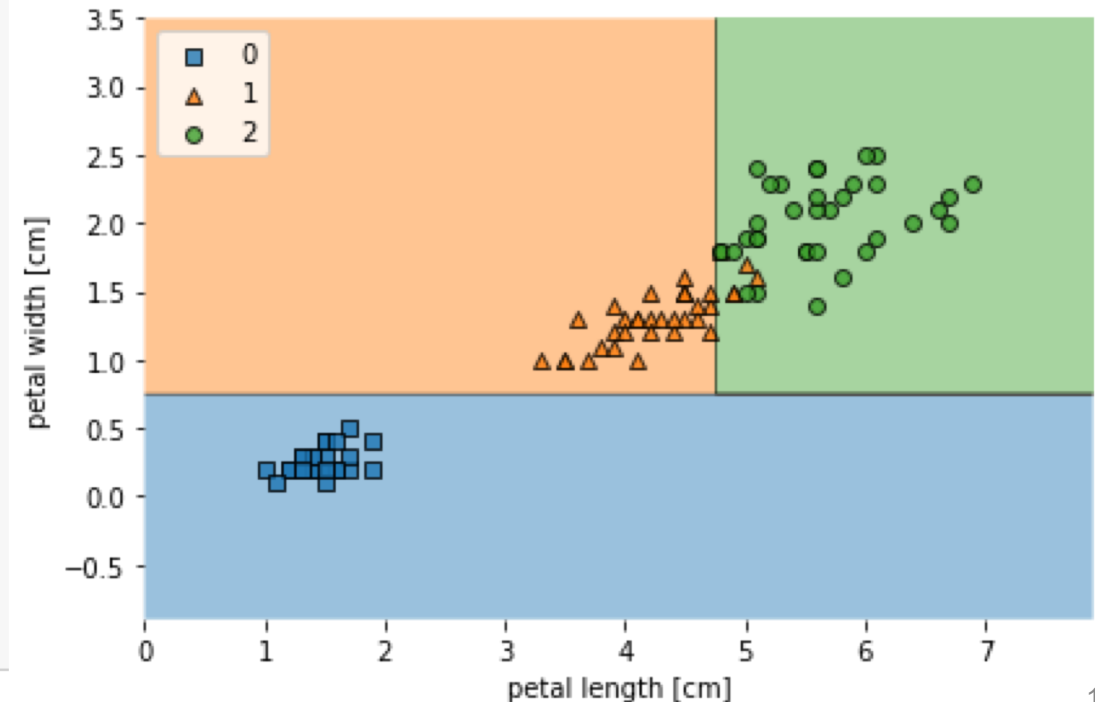
plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('images/sklearn_tree.png', dpi=300)
plt.savefig('images/sklearn_tree.pdf')
plt.show()
```

```
from sklearn import datasets
import numpy as np

iris = datasets.load_iris()
X = iris.data[:, [2, 3]]
y = iris.target
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=1, stratify=y)
```



To learn more

Ensemble Demos

- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html>
- https://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_hastie_10_2.html#sphx-glr-auto-examples-ensemble-plot-adaboost-hastie-10-2-py
- https://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html#sphx-glr-auto-examples-ensemble-plot-bias-variance-py
- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>
- http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/
- http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/
- http://rasbt.github.io/mlxtend/user_guide/classifier/StackingCVClassifier/

To learn more

LightGBM, Light Gradient Boosting Machine

<https://github.com/Microsoft/LightGBM>

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

`sklearn.ensemble`

- **Major Feature** Add two new implementations of gradient boosting trees: `ensemble.HistGradientBoostingClassifier` and `ensemble.HistGradientBoostingRegressor`. The implementation of these estimators is inspired by LightGBM and can be orders of magnitude faster than `ensemble.GradientBoostingRegressor` and `ensemble.GradientBoostingClassifier` when the number of samples is larger than tens of thousands of samples. The API of these new estimators is slightly different, and some of the features from `ensemble.GradientBoostingClassifier` and `ensemble.GradientBoostingRegressor` are not yet supported.

https://scikit-learn.org/stable/whats_new.html#version-0-21-0

Decision Tree and Ensemble learning

Hao, Qi

School of Astronomy and Space Science

THANKS

