

CS 66: Machine Learning

Prof. Sara Mathieson

Spring 2019



Admin

- Office hours: **TODAY** 12:30-2pm (Sci Center 249)
- Make sure your **registration** is correct!
- Lab 1 due **TOMORROW** night
- Make sure **BOTH** partners fill out form if you want to be paired together
- **NO CLASS**: Wednesday or Friday (I will move topics to during lab the next few weeks)

Lab 2

- Lab 2 is a two-week lab
- I will not be in lab this week, but you should still begin during your lab section with your partner
- During lab next week (Feb 6), I will “check-off” each group, as long as you have designed your **data structures** and completed a **top-down design (TDD)**

Outline for January 28

- Decision Trees and ID3 algorithm
- Overfitting
- How to find the best feature? => Entropy
- What to do about continuous features?
- Implementation suggestions

Outline for January 28

- Decision Trees and ID3 algorithm
- Overfitting
- How to find the best feature? => Entropy
- What to do about continuous features?
- Implementation suggestions

ID3 Decision Tree algorithm

- Select feature that “best” informs label prediction (i.e. y)
- **Divide**: partition data into branches based on their value at this feature
- **Conquer**: recurse on each partition

Posted as optional reading

Machine Learning 1: 81–106, 1986
© 1986 Kluwer Academic Publishers, Boston — Manufactured in The Netherlands

Induction of Decision Trees

J.R. QUINLAN (munni!nswitgould.oz!quinlan@seismo.css.gov)
Centre for Advanced Computing Sciences, New South Wales Institute of Technology, Sydney 2007, Australia

(Received August 1, 1985)

Key words: classification, induction, decision trees, information theory, knowledge acquisition, expert systems

Top-Down decision tree algorithm


MakeSubtree(D, F)

- if stopping criteria met

 - make a leaf node N

 - determine class label/probabilities for N


Top-Down decision tree algorithm

 Dataset (X,y)

MakeSubtree(D , F)

- if stopping criteria met
- make a leaf node N
- determine class label/probabilities for N

Top-Down decision tree algorithm

 **MakeSubtree**(D , F)
if stopping criteria met
 make a leaf node N
 determine class label/probabilities for N

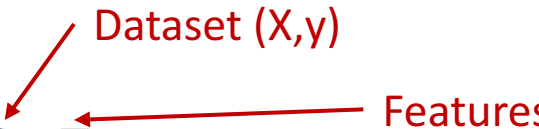
Top-Down decision tree algorithm

MakeSubtree(D , F)


- Dataset (X, y)
- Features
- if stopping criteria met
 - make a leaf node N
 - determine class label/probabilities for N

For us: use majority label
(break ties arbitrarily)

Top-Down decision tree algorithm



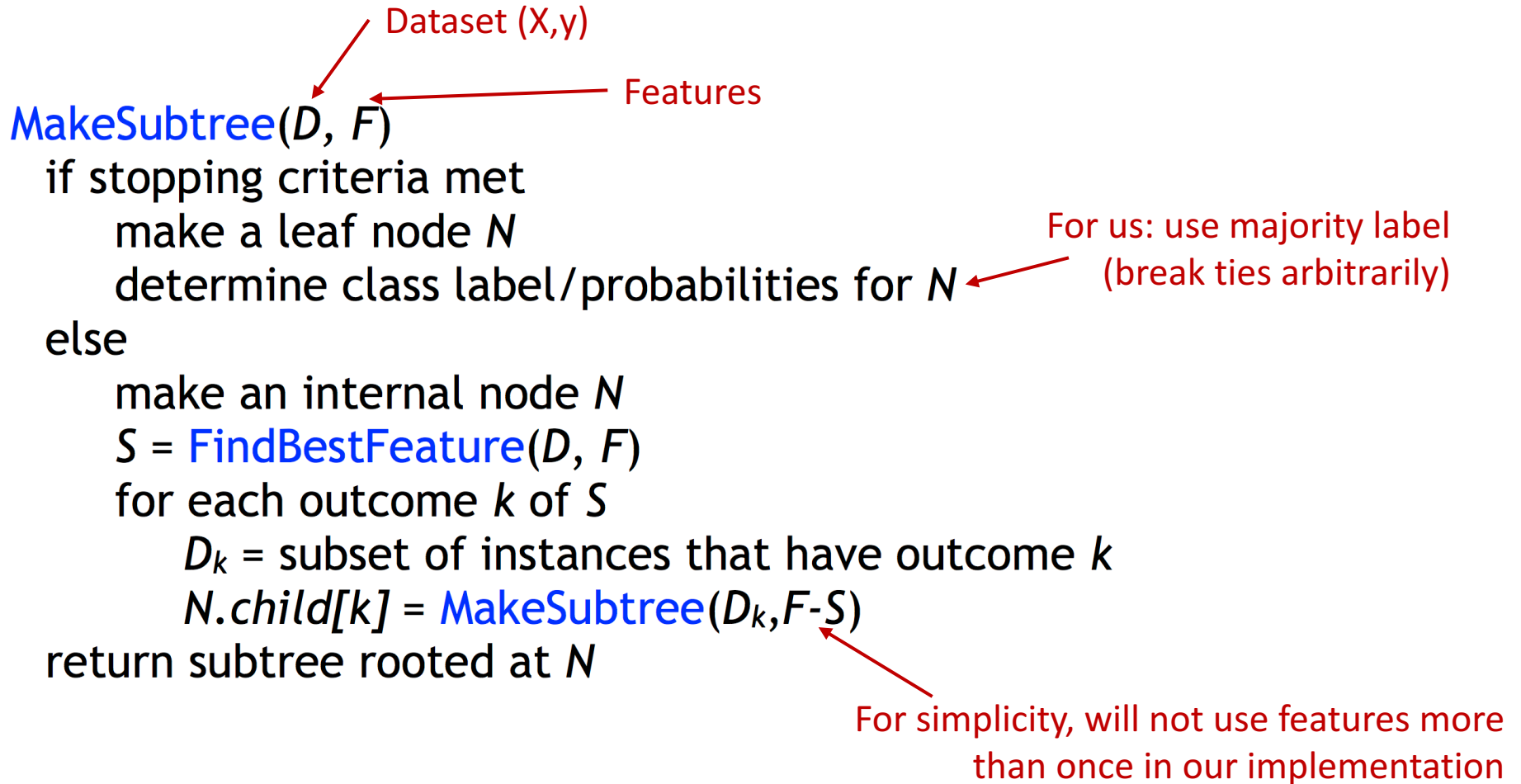
MakeSubtree(D, F)

- if stopping criteria met
 - make a leaf node N
 - determine class label/probabilities for N 
- else
 - make an internal node N
 - $S = \text{FindBestFeature}(D, F)$
 - for each outcome k of S
 - D_k = subset of instances that have outcome k
 - $N.\text{child}[k] = \text{MakeSubtree}(D_k, F-S)$

return subtree rooted at N

For us: use majority label
(break ties arbitrarily)

Top-Down decision tree algorithm

The diagram illustrates the Top-Down decision tree algorithm. It starts with a function call `MakeSubtree(D, F)`. A red arrow points from the text "Dataset (X,y)" to the parameter `D`. Another red arrow points from the text "Features" to the parameter `F`. The algorithm then follows a series of steps: a conditional check, creating a leaf node, determining its label, and an else branch for internal nodes. In the internal node branch, it calls `FindBestFeature(D, F)` and then recursively calls `MakeSubtree(Dk, F-S)`. A red arrow points from the text "For us: use majority label (break ties arbitrarily)" to the step "determine class label/probabilities for N". Another red arrow points from the text "For simplicity, will not use features more than once in our implementation" to the recursive call `MakeSubtree(Dk, F-S)`.

```
MakeSubtree( $D, F$ )  
  if stopping criteria met  
    make a leaf node  $N$   
    determine class label/probabilities for  $N$   
  else  
    make an internal node  $N$   
     $S = \text{FindBestFeature}(D, F)$   
    for each outcome  $k$  of  $S$   
       $D_k =$  subset of instances that have outcome  $k$   
       $N.\text{child}[k] = \text{MakeSubtree}(D_k, F-S)$   
  return subtree rooted at  $N$ 
```

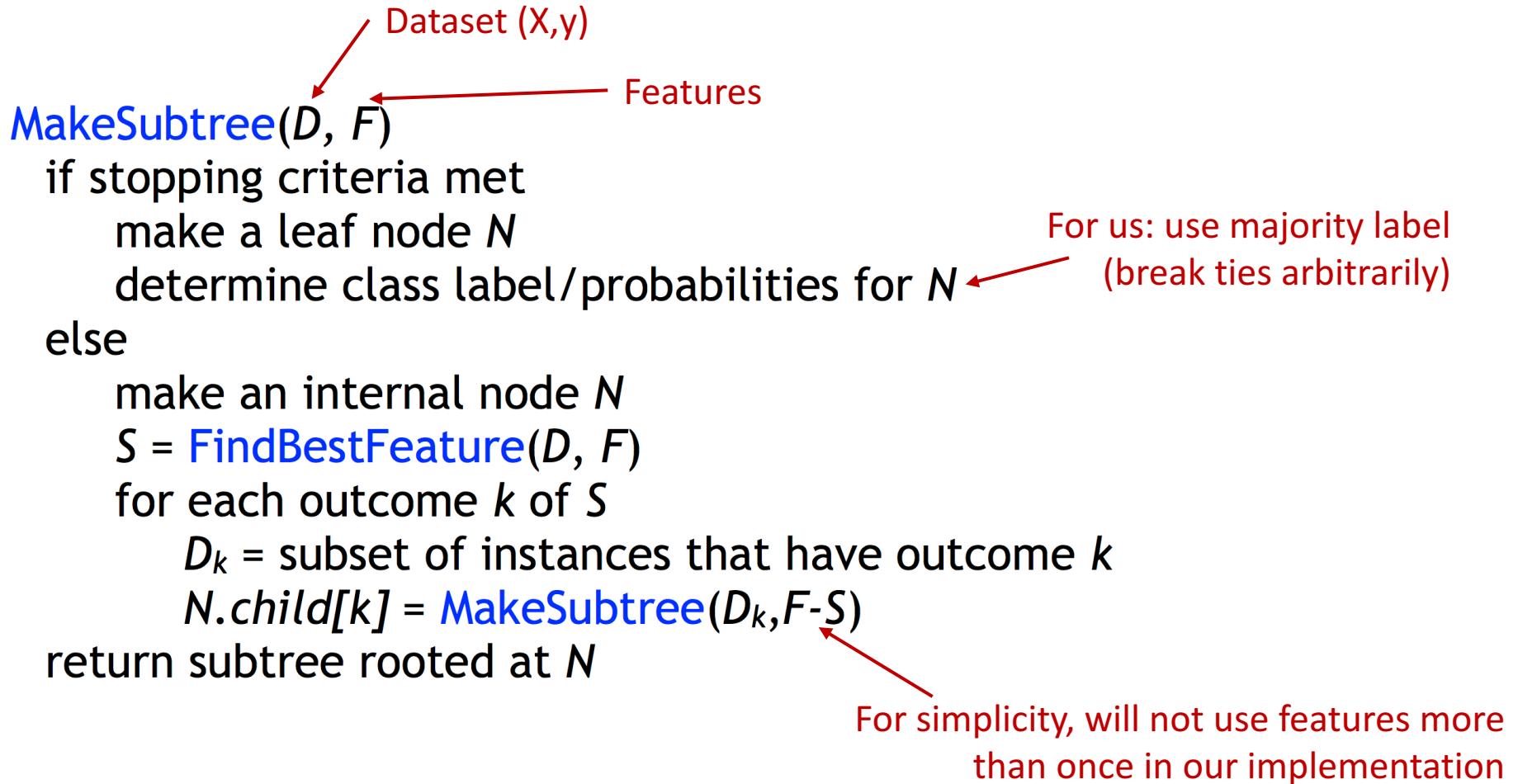
Dataset (X,y)

Features

For us: use majority label
(break ties arbitrarily)

For simplicity, will not use features more
than once in our implementation

Top-Down decision tree algorithm

The diagram illustrates the Top-Down decision tree algorithm. It starts with a function call `MakeSubtree(D, F)`. A red arrow points from the text "Dataset (X,y)" to the parameter `D`. Another red arrow points from the text "Features" to the parameter `F`. The algorithm then follows a series of steps: a conditional check "if stopping criteria met", creating a leaf node `N`, and determining class labels/probabilities for `N`. A red arrow points from the text "For us: use majority label (break ties arbitrarily)" to the step "determine class label/probabilities for N". If the stopping criteria are not met, it creates an internal node `N`, finds the best feature `S` using `FindBestFeature(D, F)`, and iterates over each outcome `k` of `S`. For each `k`, it defines `Dk` as the subset of instances with outcome `k`, and recursively calls `MakeSubtree(Dk, F-S)`. A red arrow points from the text "For simplicity, will not use features more than once in our implementation" to the recursive call. Finally, it returns the subtree rooted at `N`.

`MakeSubtree(D, F)`

- if stopping criteria met
 - make a leaf node N
 - determine class label/probabilities for N For us: use majority label (break ties arbitrarily)
- else
 - make an internal node N
 - $S = \text{FindBestFeature}(D, F)$
 - for each outcome k of S
 - $D_k =$ subset of instances that have outcome k
 - $N.\text{child}[k] = \text{MakeSubtree}(D_k, F-S)$ For simplicity, will not use features more than once in our implementation

return subtree rooted at N

Q: what design choices do we need to make?

Design choice: stopping criteria

1. All the data points in our partition have the same label

Design choice: stopping criteria

1. All the data points in our partition have the same label
2. No more features remain to split on

Design choice: stopping criteria

1. All the data points in our partition have the same label
2. No more features remain to split on
3. No features produce information gain

Design choice: stopping criteria

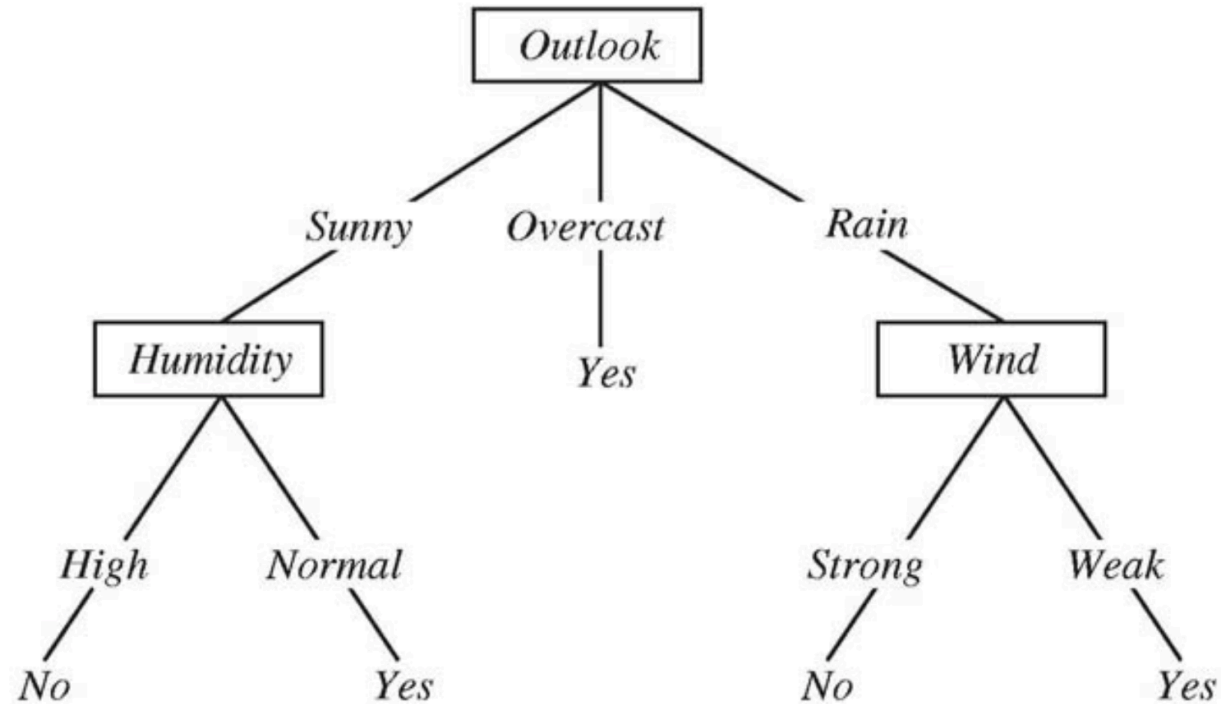
1. All the data points in our partition have the same label
2. No more features remain to split on
3. No features produce information gain
4. Reached (user specified) max depth in the tree

Outline for January 28

- Decision Trees and ID3 algorithm
- **Overfitting**
- How to find the best feature? => Entropy
- What to do about continuous features?
- Implementation suggestions

Overfitting

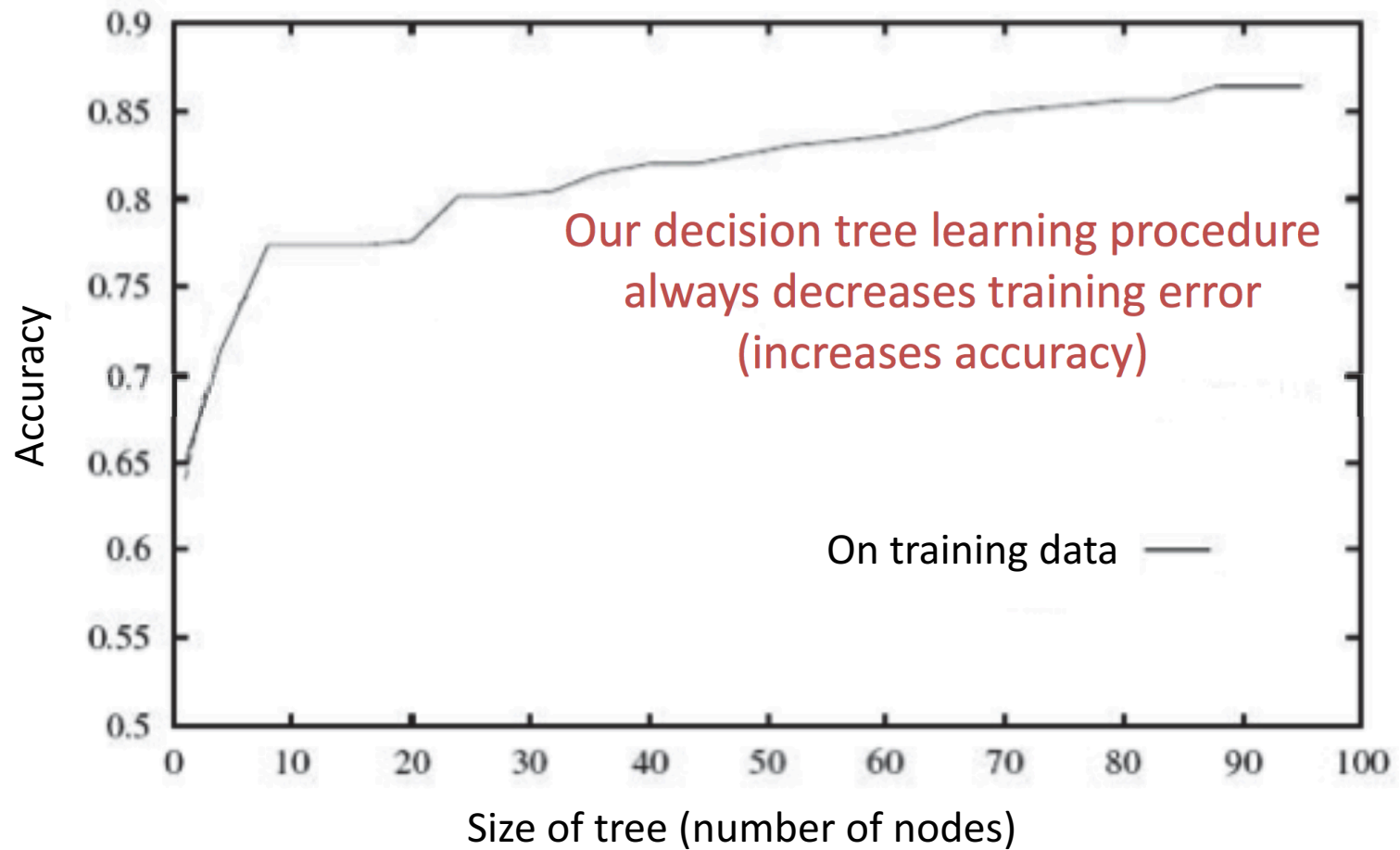
Consider adding a noisy training example to the following tree:



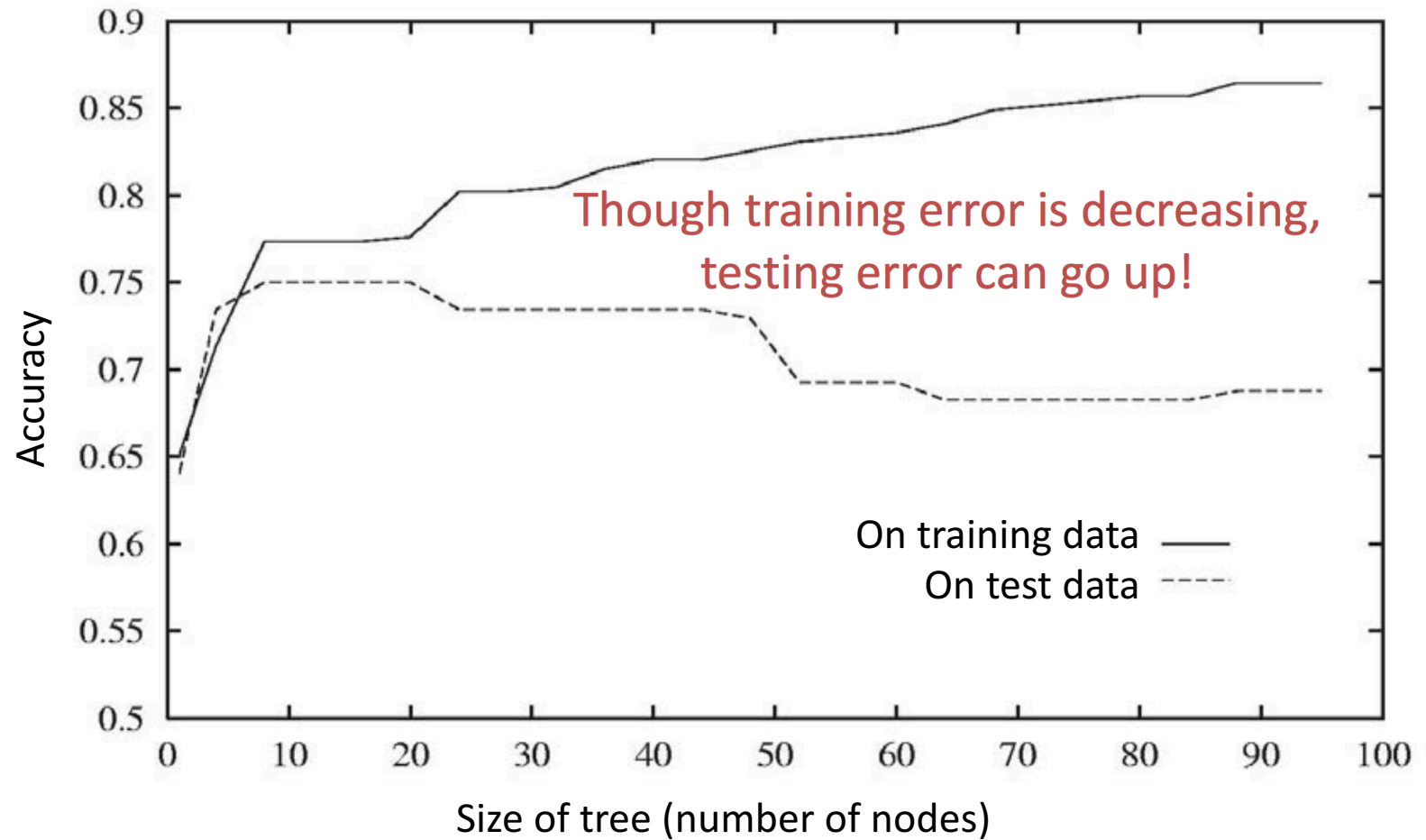
What would be the effect of adding:

(outlook=sunny, temperature=hot, humidity=normal, wind=strong, playTennis=No) ?

Overfitting



Overfitting



Overfitting definition

- Consider a hypothesis (tree): h
 - Training error: $error_{train}(h)$
 - Error over all possible data: $error_D(h)$

Overfitting definition

- Consider a hypothesis (tree): h
 - Training error: $error_{train}(h)$
 - Error over all possible data: $error_D(h)$
- A hypothesis h **overfits** training data if there exists another hypothesis h' s.t.
 - $error_{train}(h) < error_{train}(h')$ AND
 - $error_D(h) > error_D(h')$

Avoiding overfitting for us

- Stop when leaf label reaches a certain fraction (i.e. 95% “yes”, 5% “no”)

For our Lab 2 implementation

- Set a maximum depth for the tree
- Set a minimum number of examples in leaf (i.e. if we have a 2-1 split, stop)

Outline for January 28

- Decision Trees and ID3 algorithm
- Overfitting
- How to find the best feature? => Entropy
- What to do about continuous features?
- Implementation suggestions

Year	Prob (p)	Shannon	idea	Cumulative	binary	$\lceil -\log_2(p) \rceil$
senior	0.5	0	0	0	0. <u>000</u> ...	1
junior	0.25	10	1	0.5	0. <u>100</u> ...	2
soph	0.125	110	01	0.75	0. <u>110</u>	3
first	0.125	111	10	0.875	0. <u>111</u>	3
						<u># bits</u>

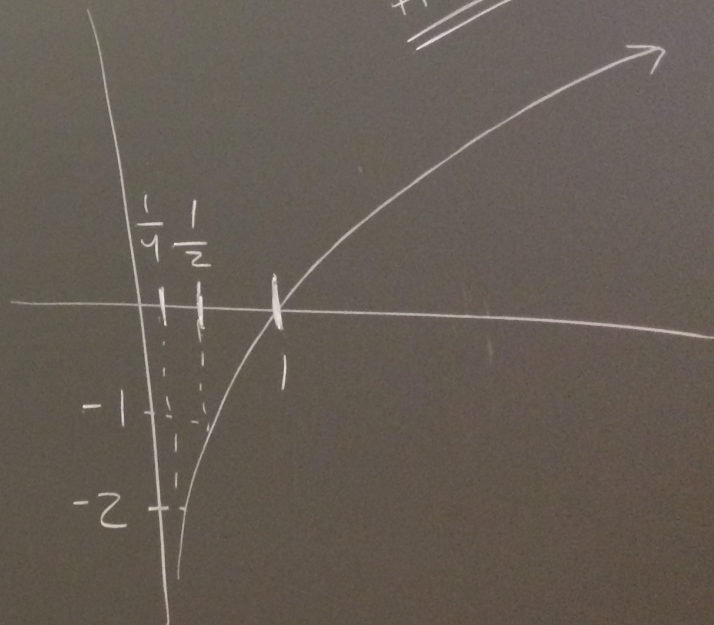
11011010001110

~~✗~~ prefix
code

$$2^{-1} = \frac{1}{2}$$

$$2^{-2} = \frac{1}{4}$$

$$2^{-3} = \frac{1}{8}$$



$$\lceil -\log(p) \rceil$$

1

2

3

3

bits

binary

$$\square \cdot 2^2 + \square \cdot 2^1 + \square \cdot 2^0 + \square \cdot 2^{-1} + \square \cdot 2^{-2}$$

decimal point

Entropy (avg # bits)

$$H(\text{year}) = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \left(\frac{1}{8} \cdot 3\right) 2 = \boxed{1.75}$$

prob of label

$$H(Y) = -$$

Condition

$$H(Y)$$

H

$$H(Y) = - \sum_{c \in \text{vals}(Y)} \underbrace{P(Y=c)}_{\text{red underline}} \log_2 \underbrace{P(Y=c)}_{\text{red underline}}$$

conditional entropy

$$\underbrace{H(Y|X=v)}_{\text{red circle}} = - \sum_{c \in \text{vals}(Y)} \underbrace{P(Y=c|X=v)}_{\text{green underline}} \log_2 P(Y=c|X=v)$$

✱

$$H(Y|X) = \sum_{\underline{v} \in \text{vals}(X)} P(X=\underline{v}) \underbrace{H(Y|X=\underline{v})}_{\text{red circle}}$$

1.75

movie	type	Length	Director	Famous	Liked?
m_1	Comedy	S	A	N	Y
m_2	Animated	S	L	N	N
m_3	Drama	M	A	N	Y
m_4	Animated	L	L	Y	N
m_5	Comedy	L	L	Y	N
m_6	Drama	M	S	Y	Y
m_7	Animated	S	S	N	Y
m_8	Comedy	L	A	Y	Y
m_9	Drama	M	L	N	Y

$$P(L_i = \text{yes}) = \frac{2}{3}$$

$$H(L_i) = - \underbrace{\frac{2}{3} \cdot \log \frac{2}{3}}_{c = \text{yes}}$$

$$\approx 0.92$$

$$P(Y=c | X=v) = \frac{4}{5}$$

$X = \text{Famous}$

$v = N$

$c = Y$

$$-\log \frac{1}{3}$$

= no

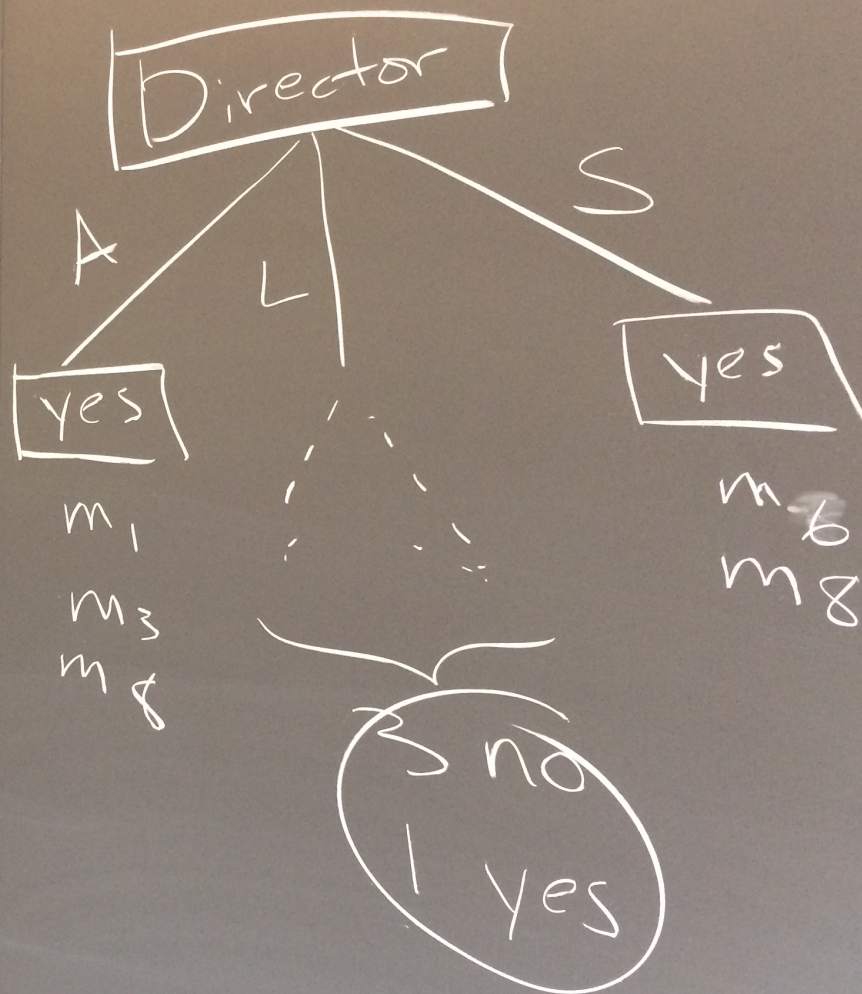
$$n(L, T) = 0.92 - 0.61 = 0.31$$

$$= 0.31$$

$$= 0.56 \quad \text{max}$$

$$= 0.07$$

gain



Outline for January 28

- Decision Trees and ID3 algorithm
- Overfitting
- How to find the best feature? => Entropy
- What to do about continuous features?
- Implementation suggestions

Continuous features

diameter	likes
10	Y
7	Y
8	N
3	Y
7	N
12	Y
2	Y

2 3 7 7 8
 Y Y Y N N
 2 3 } 7 } 8 }
 Y Y } None } N }
 $d \leq 5$ $d \leq 7.5$

gah

res

kes

Y

Y

Y

Y

Y

2 3 7 7 8 10 12 } sort

Y Y Y N N Y Y

2 3 } 7 } 8 } 10 12

Y Y } None } N } Y Y

$d \leq 5$ $d \leq 7.5$ $\nwarrow d \leq 9$

$d \leq 5$	$d \leq 7.5$	$d \leq 9$
F		
F		
F		
T		
⋮		

Outline for January 28

- Decision Trees and ID3 algorithm
- Overfitting
- How to find the best feature? => Entropy
- What to do about continuous features?
- Implementation suggestions

Implementation Suggestions

- Think back to **trees in CS35** (data structures)
- Distinguish between **data** (X,y) and **options for data** (values for each feature, classes for y)
- Start slow with entropy! Build up function by function