Machine Learning Department
School of Computer Science
Carnegie Mellon University

# Decision Trees

Matt Gormley
Lecture 3
Jan. 25, 2023

# Q&A

**Q:** In our medical diagnosis example, suppose two of our doctors (i.e. experts) disagree about whether (+) or not (-) the patient is sick. How would the decision tree represent this situation?

**A:** Today we will define decision trees that predict a single class by a majority vote at the leaf. More generally, the leaf could provide a probability distribution over output classes $p(y|\mathbf{x})$

# Q&A

## **Q:** How do these In-Class Polls work?

**A:**

- Sign into **Google Form** (**c**lick [Poll] link on Schedule page http://mlcourse.org/schedule.html) using **Andrew Email**
- Answer **during lecture for full credit**, or within 24 hours for half credit
- Avoid the **toxic option** which gives negative points!
- 8 "free poll points" but can't use more than 3 free polls consecutively. All the questions for one lecture are worth 1 point total.

Latest Poll link: http://poll.mlcourse.org

# First In-Class Poll

Question 1

**Question:**

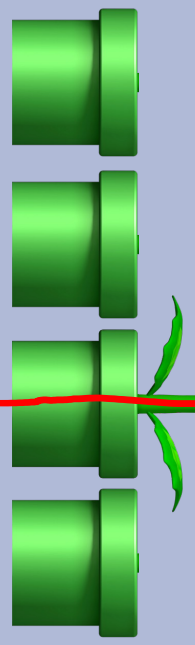
*Which of the following did you bring to class today?*

A. Smartphone

B. Flip phone

C. ~~Pay phone~~

D. No phone



negative points

**Answer:**

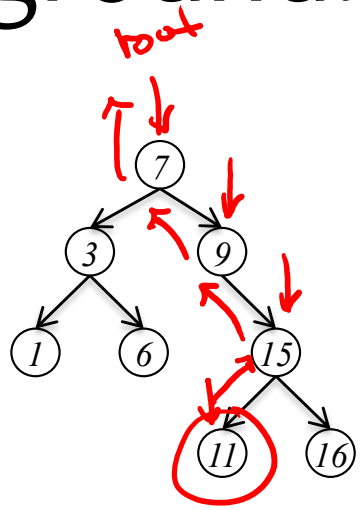
# Reminders

- **Homework 1: Background**
  - **Out: Fri, Jan 20**
  - **Due: Wed, Jan 25 at 11:59pm**
  - unique policy for this assignment: we will grant (essentially) any and all extension requests
- **Homework 2: Decision Trees**
  - **Out: Wed, Jan. 25**
  - **Due: Fri, Feb. 3 at 11:59pm**

# MAKING PREDICTIONS WITH A DECISION TREES

# Background: Recursion

- *Def*: a **binary search tree** (BST) consists of nodes, where each node:
  - has a value, v
  - up to 2 children
  - all its left descendants have values less than v, and its right descendants have values greater than v
- We like BSTs because they permit search in O(log(n)) time, assuming n nodes in the tree




contains(root, 11)

**Recursive Search**

```
def contains(node, key):
    if key < node.value & node.left != null:
        return contains(node.left, key)
    else if node.value < key & node.right != null:
        return contains(node.right, key)
    else:
        return key == node.value
```

**Node Data Structure**

```
class Node:
    int value
    Node left
    Node right
```

**Iterative Search**

```
def contains(node, key):
    cur = node
    while true:
        if key < cur.value & cur.left != null:
            cur = cur.left
        else if cur.value < key & cur.right != null:
            cur = cur.right
        else:
            break
    return key == cur.value
```

# Decision Tree: Prediction

*In-Class Exercise*: Let's **evaluate** our (already learned) decision tree's error rate on a real **test** dataset.

**Features:**

- $x_1$ : which is better? {green, orange}
- $x_2$ : which is better? {consistency, challenge}
- $x_3$ : which is better? {sandals , sneakers}
- $x_4$ : which is better? {winter, summer}

**Label:**

- y : are you a beach-person or a mountains-person?

# Decision Tree: Prediction

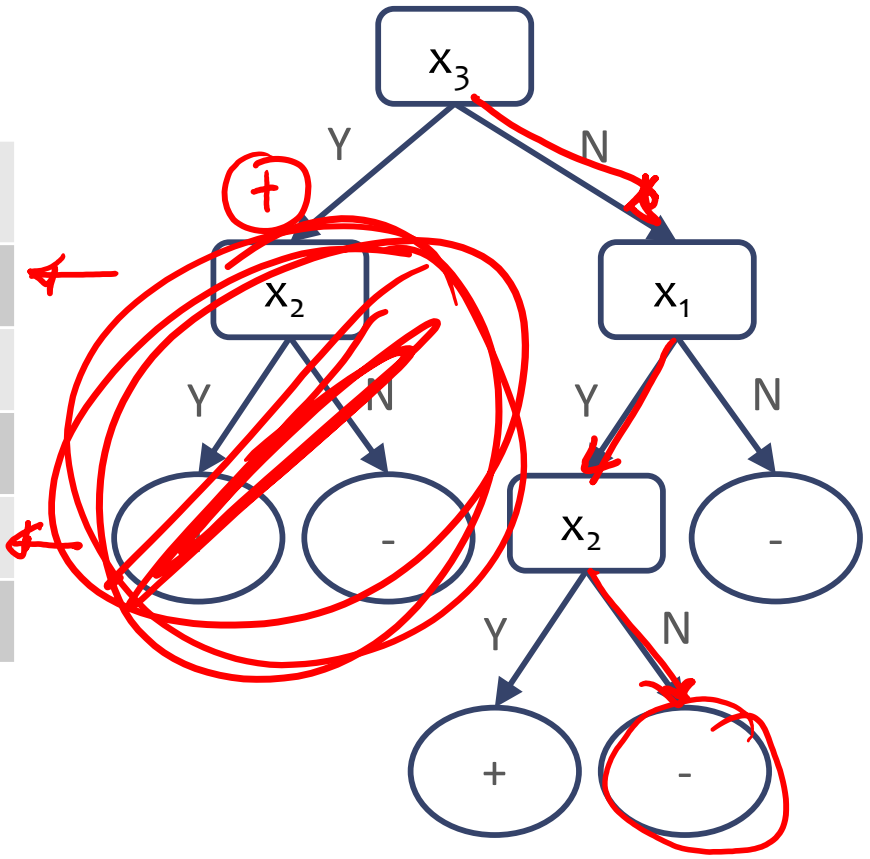
def h(**x'**):

- Let *current node* = root
- while(true):
  - if *current node* is internal (non-leaf):
    - Let m = attribute associated with current node
    - Go down branch labeled with value $x'_m$
  - if *current node* is a leaf:
    - return label y stored at that leaf

# Decision Tree: Prediction

Algorithm 3 **decision tree:** recursively walk from root to a leaf, following the attribute values labeled on the branches, and return the label at the leaf

| | y | $x_1$ | $x_2$ | $x_3$ | $x_4$ |
|---|---|---|---|---|---|
| predic-tions | allergic? | hives? | sneezing? | red eye? | has cat? |
| - | - | Y | N | N | N |
| - | - | N | Y | N | N |
| + | + | Y | Y | N | N |
| - | - | Y | N | Y | Y |
| + | + | N | Y | Y | N |




Zero training error!

# Decision Tree: Prediction

def h(**x'**):

- Let *current node* = root
- while(true):
  - if *current node* is internal (non-leaf):
    - Let m = attribute associated with current node
    - Go down branch labeled with value $x'_m$
  - if *current node* is a leaf:
    - return label y stored at that leaf

**Question:** The above pseudocode is implementing prediction with a while-loop.

Can you convert it to a recursive implementation?

# Decision Trees

*Whiteboard*

- Example Decision Tree as a hypothesis
- Defining h(x) for a decision tree

# Tree to Predict C-Section Risk

Learned from medical records of 1000 women (Sims et al., 2000)

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Figure from Tom Mitchell

# LEARNING A DECISION TREE

# Decision Trees

*Whiteboard*

– Decision Tree Learning

# Recursive Training for Decision Trees

def train(dataset D'):

- Let p = new Node()
- ***Base Case:*** If (1) all labels $y^{(i)}$ in D' are identical (2) D' is empty (3) for each attribute, all values are identical
  - p.type = Leaf // The node p is a leaf node
  - p.label = majority_vote(D') // Store the label
  - return p
- ***Recursive Step:*** Otherwise
  - Make an internal node
    - p.type = Internal // The node p is an internal node
  - Pick the *best* attribute $X_m$ according to splitting criterion
    - p.attr = $\text{argmax}_m$ splitting_criterion(D', $X_m$) // Store the attribute on which to split
  - For each value v of attribute $X_m$:
    - $D_{X_m = v}$ = {(**x**,y) in D' : $x_m$ = v} // Select a partition of the data
    - $\text{child}_v$ = train( $D_{X_m = v}$ ) // Recursively build the child
    - p.branches[v] = $\text{child}_v$ // Create a branch with label v
    - return p

# Decision Tree Learning Example

## Dataset:

Output Y, Attributes A, B, C

| Y | A | B | C |
|---|---|---|---|
| - | 1 | 0 | 0 |
| - | 1 | 0 | 1 |
| - | 1 | 0 | 0 |
| + | 0 | 0 | 1 |
| + | 1 | 1 | 0 |
| + | 1 | 1 | 1 |
| + | 1 | 1 | 0 |
| + | 1 | 1 | 1 |

## In-Class Exercise

Using **error rate** as the splitting criterion, what decision tree would be learned?

# Decision Trees

*Whiteboard*

- Example of Decision Tree Learning with Error Rate as splitting criterion

# SPLITTING CRITERION: ERROR RATE

# Decision Tree Learning

- *Definition*: a **splitting criterion** is a function that measures the effectiveness of splitting on a particular attribute
- Our decision tree learner **selects the "best" attribute** as the one that maximizes the splitting criterion
- Lots of options for a splitting criterion:
  - error rate (or *accuracy* if we want to pick the tree that *maximizes* the criterion)
  - Gini gain
  - Mutual information
  - random
  - …

# Decision Tree Learning Example

## Dataset:

Output Y, Attributes A and B

| Y | A | B |
|---|---|---|
| - | 1 | 0 |
| - | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |

Question 2

## In-Class Exercise

Which attribute would ~~error rate~~ accuracy select for the next split?

1. A
2. B
3. A or B (tie)
4. ~~Neither~~ toxic

# Decision Tree Learning Example

**Dataset:**

Output Y, Attributes A and B

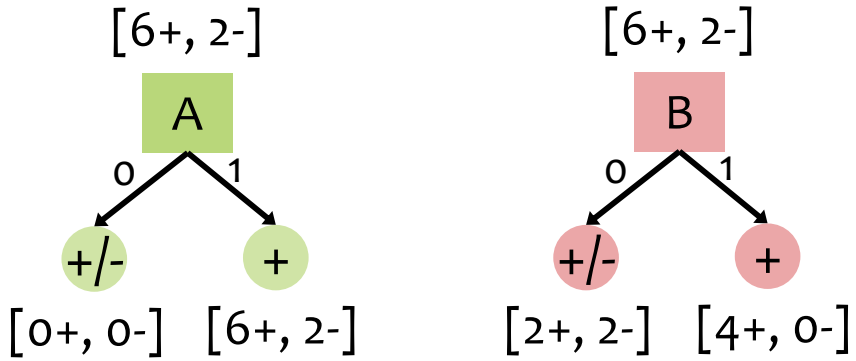
| Y | A | B |
|---|---|---|
| - | 1 | 0 |
| - | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |

# Decision Tree Learning Example

**Dataset:**

Output Y, Attributes A and B

| Y | A | B |
|---|---|---|
| - | 1 | 0 |
| - | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |




**Error Rate**

$error(h_A, D) = 2/8$

$error(h_B, D) = 2/8$

error rate treats attributes A and B as equally good

# SPLITTING CRITERION: MUTUAL INFORMATION

# Information Theory & DTs

*Whiteboard*

- Information Theory primer
  - Entropy
  - (Specific) Conditional Entropy
  - Conditional Entropy
  - Information Gain / Mutual Information
- Information Gain as DT splitting criterion

# Mutual Information

Let $X$ be a random variable with $X \in \mathcal{X}$.
Let $Y$ be a random variable with $Y \in \mathcal{Y}$.

Entropy: $H(Y) = -\sum_{y \in \mathcal{Y}} P(Y = y) \log_2 P(Y = y)$

Specific Conditional Entropy: $H(Y \mid X = x) = -\sum_{y \in \mathcal{Y}} P(Y = y \mid X = x) \log_2 P(Y = y \mid X = x)$

Conditional Entropy: $H(Y \mid X) = \sum_{x \in \mathcal{X}} P(X = x) H(Y \mid X = x)$

Mutual Information: $I(Y; X) = H(Y) - H(Y|X) = H(X) - H(X|Y)$

- For a decision tree, we can use **mutual information** of the output class *Y* and some attribute *X* on which to split **as a splitting criterion**
- Given a dataset *D* of training examples, we can estimate the required probabilities as…

$$P(Y = y) = N_{Y=y}/N$$
$$P(X = x) = N_{X=x}/N$$
$$P(Y = y|X = x) = N_{Y=y,X=x}/N_{X=x}$$

where $N_{Y=y}$ is the number of examples for which $Y = y$ and so on.

# Mutual Information

Let $X$ be a random variable with $X \in \mathcal{X}$.
Let $Y$ be a random variable with $Y \in \mathcal{Y}$.

Entropy: $H(Y) = -\sum_{y \in \mathcal{Y}} P(Y = y) \log_2 P(Y = y)$

Specific Conditional Entropy: $H(Y \mid X = x) = -\sum_{y \in \mathcal{Y}} P(Y = y \mid X = x) \log_2 P(Y = y \mid X = x)$

Conditional Entropy: $H(Y \mid X) = \sum_{x \in \mathcal{X}} P(X = x) H(Y \mid X = x)$

Mutual Information: $I(Y; X) = H(Y) - H(Y|X) = H(X) - H(X|Y)$

- **Entropy** measures the **expected # of bits** to code one random draw from X**.**
- For a decision tree, we want to **reduce the entropy of the random variable we are trying to predict**!

**Conditional entropy** is the expected value of specific conditional entropy
$E_{P(X=x)}[H(Y \mid X = x)]$

**Informally,** we say that **mutual information** is a measure of the following:
*If we know X, how much does this reduce our uncertainty about Y?*

# Decision Tree Learning Example

Q3 : skip

## Dataset:

Output Y, Attributes A and B

| Y | A | B |
|---|---|---|
| - | 1 | 0 |
| - | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |

## In-Class Exercise

Which attribute would **mutual information** select for the next split?

1. A
2. B ✓
3. A or B (tie)
4. Neither

# Decision Tree Learning Example

**Dataset:**

Output Y, Attributes A and B

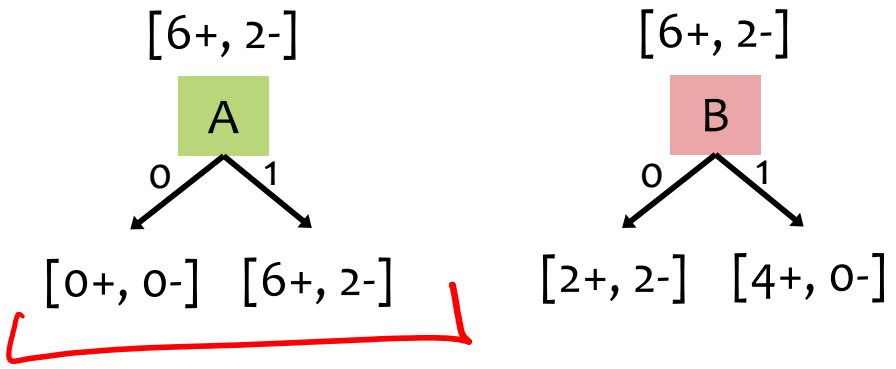
| Y | A | B |
|---|---|---|
| - | 1 | 0 |
| - | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |

# Decision Tree Learning Example

## Dataset:

Output Y, Attributes A and B

| Y | A | B |
|---|---|---|
| - | 1 | 0 |
| - | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 0 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |
| + | 1 | 1 |




### Mutual Information

$H(Y) = -2/8 \log(2/8) - 6/8 \log(6/8)$

$H(Y|A=0)$ = "undefined"

$H(Y|A=1) = -2/8 \log(2/8) - 6/8 \log(6/8)$
$= H(Y)$

$H(Y|A) = P(A=0)H(Y|A=0) + P(A=1)H(Y|A=1)$
$= 0 + H(Y|A=1) = H(Y)$

$I(Y; A) = H(Y) - H(Y|A=1) = 0$

$H(Y|B=0) = -2/4 \log(2/4) - 2/4 \log(2/4)$

$H(Y|B=1) = -0 \log(0) - 1 \log(1) = 0$

$H(Y|B) = 4/8(0) + 4/8(H(Y|B=0))$

$I(Y; B) = H(Y) - 4/8\ H(Y|B=0) > 0$

# Tennis Example

Test your understanding

## Dataset:

| Day | Outlook | Temperature | 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 |

Figure from Tom Mitchell

# Tennis Example

Test your understanding

Which attribute yields the best classifier?

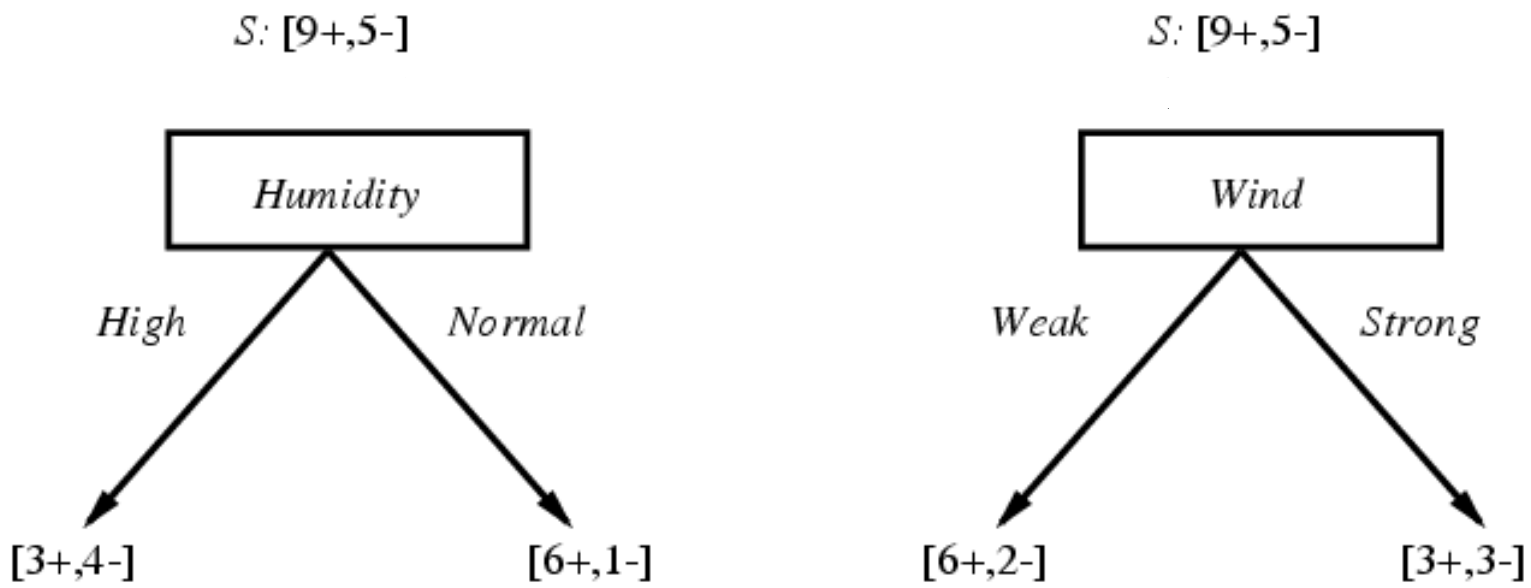



Figure from Tom Mitchell

# Tennis Example

Test your understanding

Which attribute yields the best classifier?

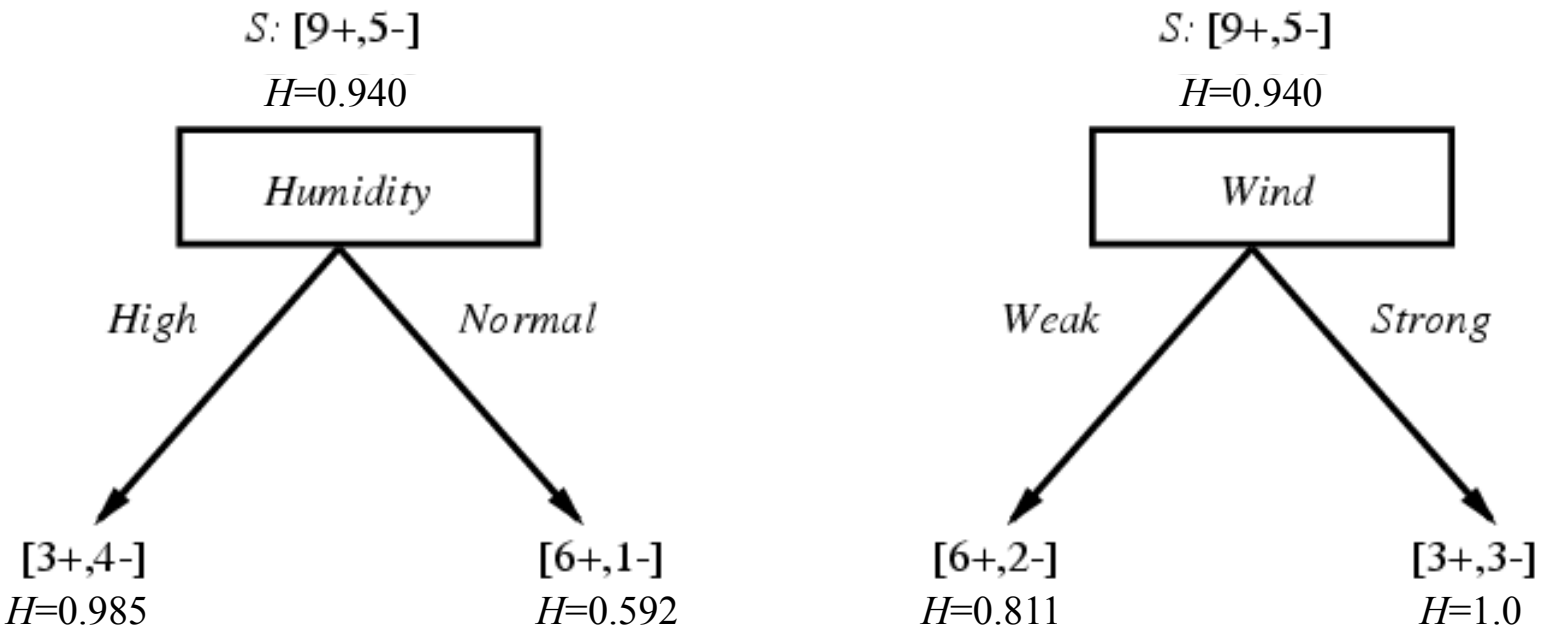
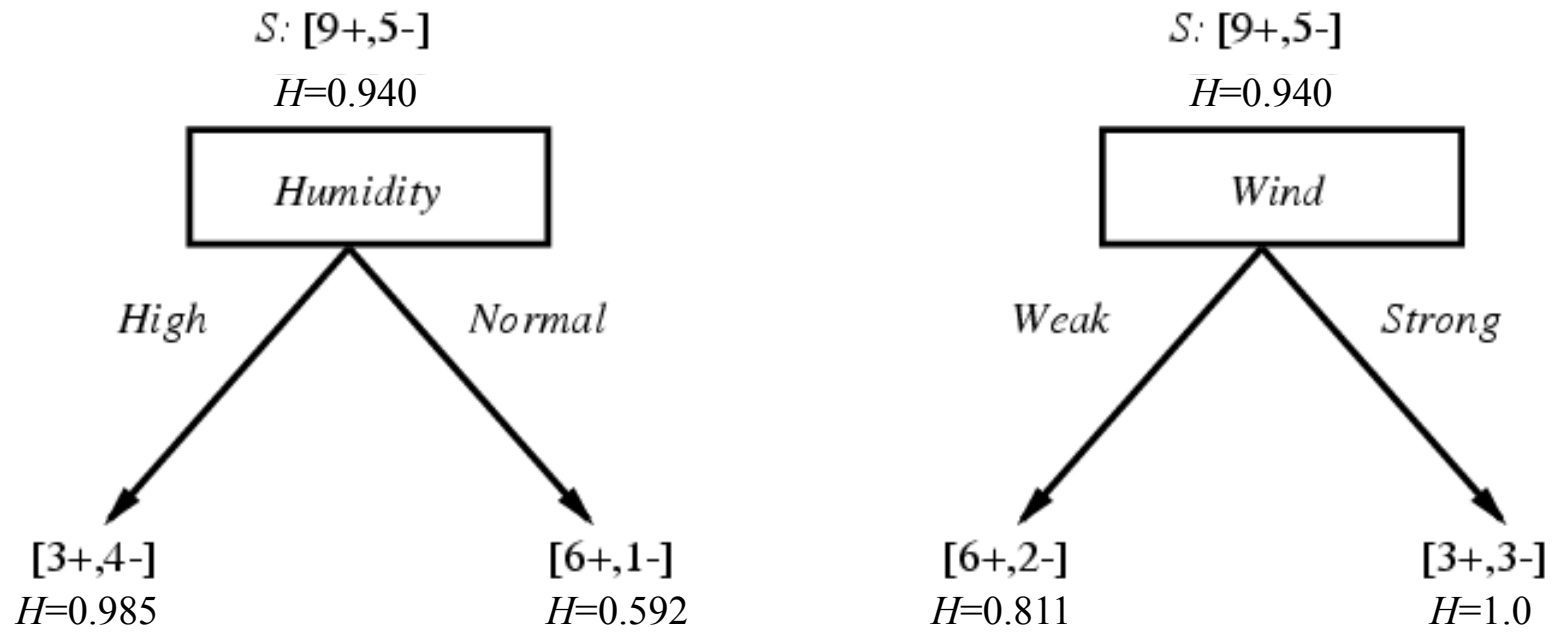



Figure from Tom Mitchell

# Tennis Example

Test your understanding

## Which attribute yields the best classifier?




*Gain (S, Humidity )*

= .940 - (7/14).985 - (7/14).592
= .151

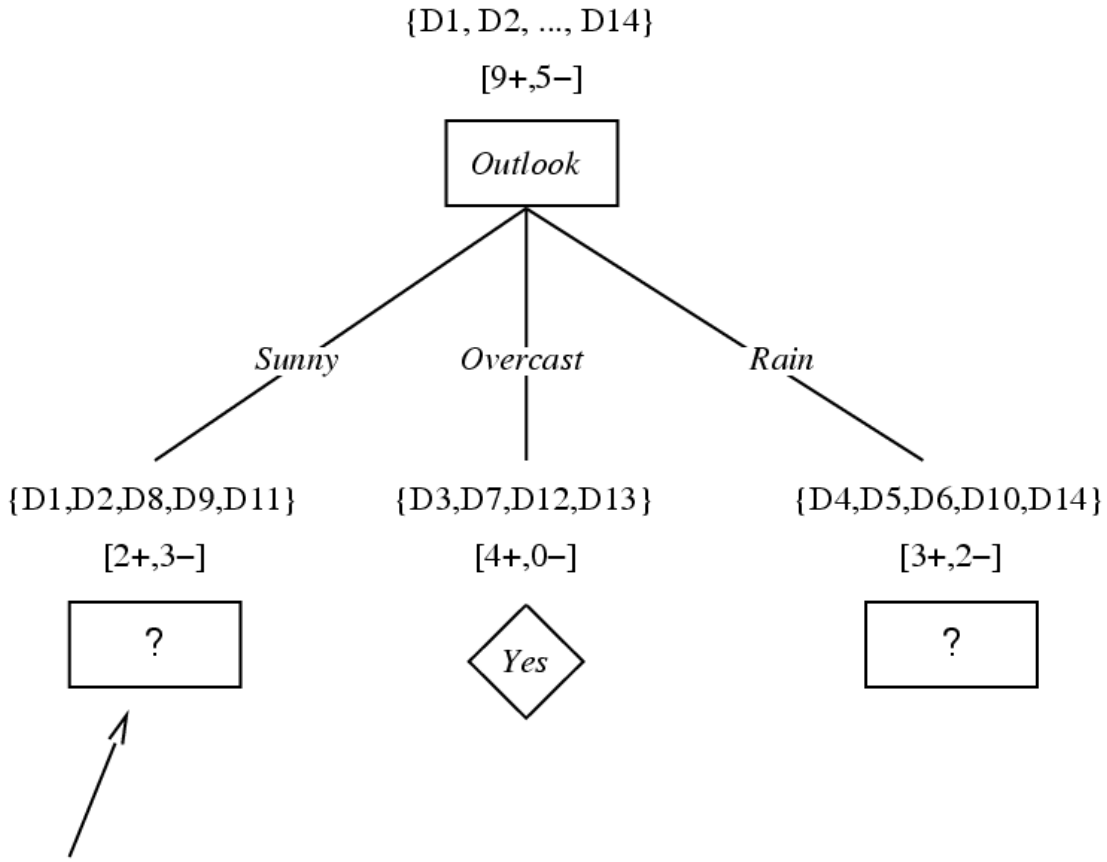
*Gain (S, Wind)*

= .940 - (8/14).811 - (6/14)1.0
= .048

Figure from Tom Mitchell

# Tennis Example

Test your understanding




*Which attribute should be tested here?*

$S_{sunny}$ = {D1,D2,D8,D9,D11}

*Gain* ($S_{sunny}$, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

*Gain* ($S_{sunny}$, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

*Gain* ($S_{sunny}$, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Figure from Tom Mitchell