Lecture 6: Learning Decision Trees

COMP 343, Spring 2022

Victoria Manfredi

WESLEYAN UNIVERSITY

Acknowledgements: These slides are based primarily on content from the book "Machine Learning" by Tom Mitchell, and on slides created by Vivek Srikumar (Utah), Dan Roth (Penn), and Jessica Wu (Harvey Mudd College)

Today's Topics

Homework 3 out

- Due Thursday, February 23 by 5p

Learning decision trees (ID3 algorithm)

- Greedy heuristic (based on information gain)
  Originally developed for discrete features
- Some extensions to the basic algorithm

Decision Trees

RECAP

Decision trees overview

Decision trees can represent any Boolean function

A way to represent a lot of data

A natural representation (think playing 20 questions)

Predicting with a decision tree classifier is easy

Clearly, given a dataset, there are many decision trees that can represent it

Learning a good representation from data is the challenge

Problem setting for decision tree learning

Set of possible instances, X

- Each instance $\mathbf{x} \in X$ is a feature vector
- E.g., <shape=square, color=red>

Set of possible labels, Y

- Y is discrete-valued

Unknown target function, $f : X \to Y$

Set of function hypotheses, $H = \{h \mid h : X \to Y\}$

- Each hypothesis h is a decision tree
- Trees sort $\mathbf{x}$ to leaf which assigns $y \in Y$

Decision Trees

LEARNING

History of decision tree research

Full search decision tree methods to model human concept learning: Hunt et al 60s, psychology

Quinlan developed the ID3 (*Iterative Dichotomiser 3*) algorithm with the information gain heuristic to learn expert systems from examples (late 70s)

Breiman, Freidman and colleagues in statistics developed CART (*Classification And Regression Trees*)

A variety of improvements in the 80s: coping with noise, continuous attributes, missing data, non-axis parallel, etc.

Quinlan's updated algorithms, C4.5 (1993) and C5 are more commonly used

Boosting (or Bagging) over decision trees is a very good general purpose algorithm

Will I play tennis today?

Features

- Outlook: {Sun, Overcast, Rain}
- Temperature: {Hot, Mild, Cool}
- Humidity: {High, Normal, Low}
- Wind: {Strong, Weak}

Will I play tennis today?

Features

- Outlook: {Sun, Overcast, Rain}
- Temperature: {Hot, Mild, Cool}
- Humidity: {High, Normal, Low}
- Wind: {Strong, Weak}

Labels

- Binary classification task: Y = {+, -}

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: **S**unny, **O**vercast, **R**ainy

Temperature: **H**ot, **M**edium, **C**ool

Humidity: **H**igh, **N**ormal, **L**ow

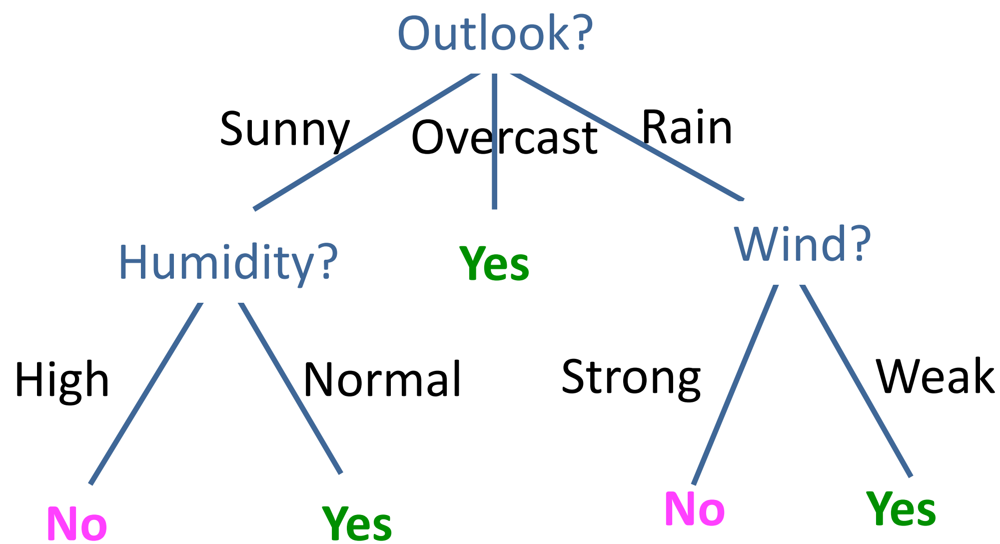
Wind: **S**trong, **W**eak

Basic decision tree learning algorithm

Data is processed in batch (i.e., all of the data available)

Recursively build a decision tree top down

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

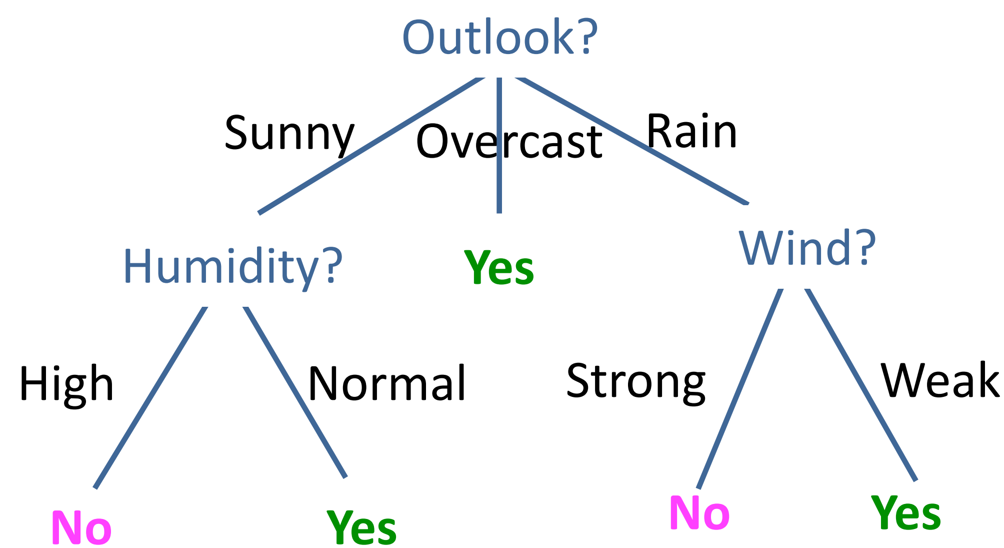


Basic decision tree learning algorithm

Data is processed in batch (i.e., all of the data available)

Recursively build a decision tree top down

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-



May not need all features, different subtrees may be different

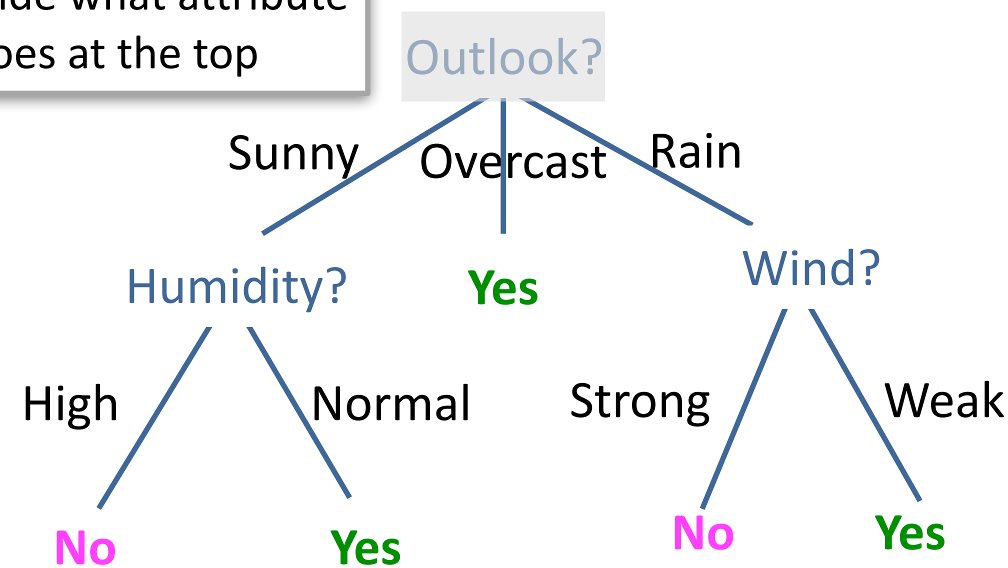
Basic decision tree learning algorithm

Data is processed in batch (i.e., all of the data available)

Recursively build a decision tree top down

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

1. Decide what attribute goes at the top

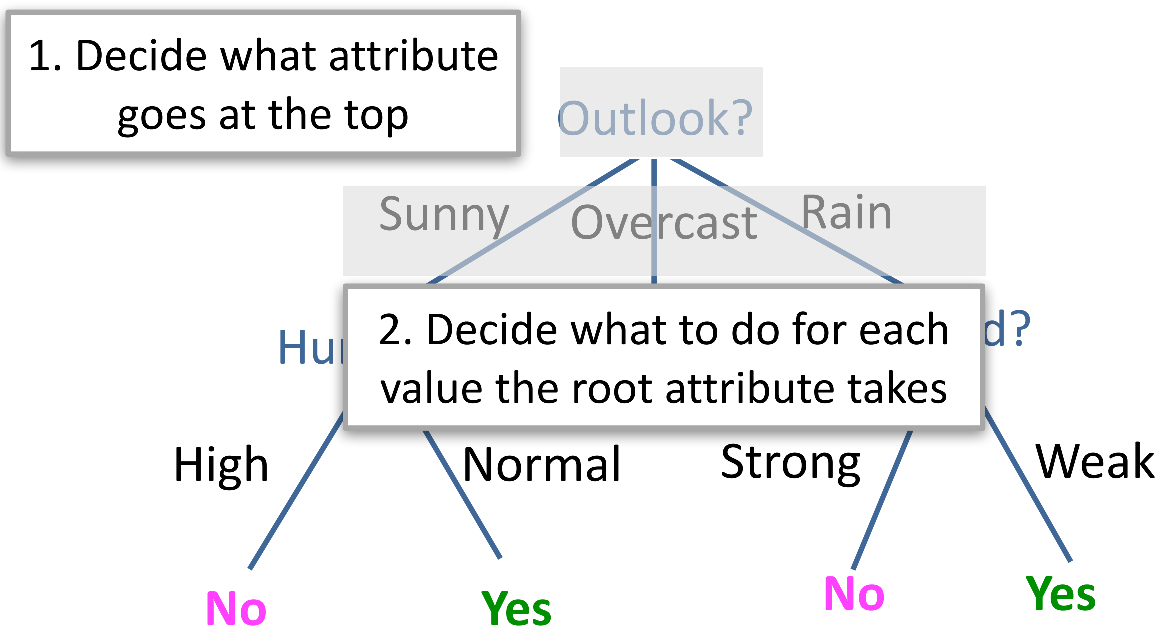


Basic decision tree learning algorithm

Data is processed in batch (i.e., all of the data available)

Recursively build a decision tree top down

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-



Basic decision tree algorithm: ID3

ID3(S, A):

Input:
S is the set of examples
A is the set of measured attributes

Algorithm takes as input the examples, attributes and producers agree

Basic decision tree algorithm: ID3

ID3(S, A):

Input:
S is the set of examples
A is the set of measured attributes

If all examples have the same label do we need to build a tree?

O	T	H	W	Play?
S	C	N	W	+
R	M	N	W	+
S	M	N	S	+
O	M	H	S	+
O	H	N	W	+

Basic decision tree algorithm: ID3

ID3(S, A):

Input:
S is the set of examples
A is the set of measured attributes

If all examples have the same label do we need to build a tree?

No! Just create a leaf with that label

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree

Decide what attribute goes at the top

?

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that ***best*** classifies S

What does this mean?

A_b is an attribute in the set A that we have determined best classifies the examples in S. So the best attribute for the new node is A_b

Outlook

Basic decision tree algorithm: ID3

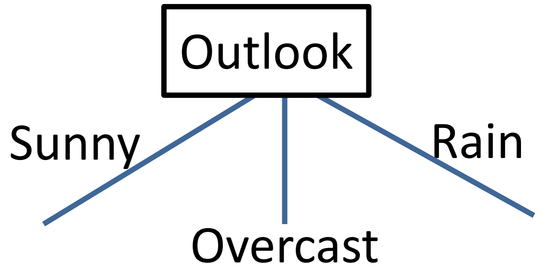
Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that *best* classifies S
   3. For each possible value v that A_b can take on
      - Add a new tree branch for attribute A_b taking value v



Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

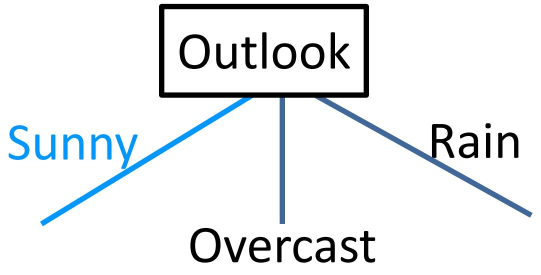
ID3(S, A):

1. **If** all examples have same label
   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that *best* classifies S
   3. For each possible value v that A_b can take on
      - Add a new tree branch for attribute A_b taking value v
      - Let $S_v \subseteq S$ be the subset of examples with $A_b = v$

What does this mean?

We focus on the subset of original data for which Outlook is Sunny, when building the subtree at a branch



Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label
   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that ***best*** classifies S
   3. For each possible value v that A_b can take on
      - Add a new tree branch for attribute A_b taking value v
      - Let $S_v \subseteq S$ be the subset of examples with $A_b = v$

What does this mean?

We focus on the subset of original data for which Outlook is Sunny, when building the subtree at a branch

O	T	H	W	Play?
S	H	H	W	-
S	H	H	S	-
S	M	H	W	-
S	C	N	W	+
S	M	N	S	+

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that ***best*** classifies S
   3. For each possible value v that A_b can take on
      - Add a new tree branch for attribute A_b taking value v
      - Let $S_v \subseteq S$ be the subset of examples with $A_b = v$
      - If $S_v = \emptyset$: add leaf node with the common value of label in S

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that ***best*** classifies S
   3. For each possible value v that A_b can take on
      - Add a new tree branch for attribute A_b taking value v
      - Let $S_v \subseteq S$ be the subset of examples with $A_b = v$
      - If $S_v = \emptyset$: add leaf node with the common value of label in S

What does this mean?

Special case: what if set of examples is empty? Like red triangle case. Picking most common value of label helps with generalization at test time

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that ***best*** classifies S
   3. For each possible value v that A_b can take on
      - Add a new tree branch for attribute A_b taking value v
      - Let $S_v \subseteq S$ be the subset of examples with $A_b = v$
      - If $S_v = \emptyset$: add leaf node with the common value of label in S

        Else: below this branch add the subtree ID3($S_v, A - \{A_b\}$)
   4. Return root node R

Set is empty

Set is not empty

Recursive call to the ID3 algorithm with all the remaining attributes and subset of data

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**
   1. Create a root node, R, for tree
   2. $A_b \in A$ is the attribute that *best* classifies S
   3. For each possible value v that A_b can take on
      - Add a new tree branch for attribute A_b taking value v
      - Let $S_v \subseteq S$ be the subset of examples with $A_b = v$
      - If $S_v = \varnothing$: add leaf node with the common value of label in S

        Else: below this branch add the subtree ID3($S_v, A - \{A_b\}$)
   4. Return root node R

Set is empty

Set is not empty

Different branches may pick attributes in different orders!

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label
   Return a single node tree with the label

2. **Otherwise**
   1.
   2.
   3.

Can you implement this?

   4. Return root node R

Basic decision tree algorithm: ID3

Input:
S is the set of examples
A is the set of measured attributes

ID3(S, A):

1. **If** all examples have same label

   Return a single node tree with the label

2. **Otherwise**

   1.
   2.
   3.

   Can you implement this?

   Everything is well defined, except for "best"

   4. Return root node R

Picking the root/best attribute

Picking the root/best attribute

Goal: have the resulting decision tree be **as small as possible**

Occam's razor:

Simpler explanations are better. Here, simpler explanations correspond to smaller trees

Picking the root/best attribute

Goal: have the resulting decision tree be **as small as possible**

Problem: finding the minimal decision tree consistent with data is NP-hard

Picking the root/best attribute

Goal: have the resulting decision tree be **as small as possible**

Problem: finding the minimal decision tree consistent with data is NP-hard (means for all practical purposes can't do)

Solution: greedy heuristic search

- recursive algorithm for a simple tree
- cannot guarantee optimality
- main decision is to select next attribute to split on

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 0 examples

< (A=1,B=1), + >: 100 examples

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 0 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 0 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

We have only 2 cases, so let's try them both out!

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

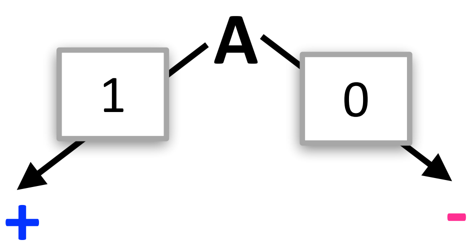
< (A=1,B=0), - >: 0 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes

Why do we get purely labeled nodes?



Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

< (A=0,B=0), - >: 50 examples

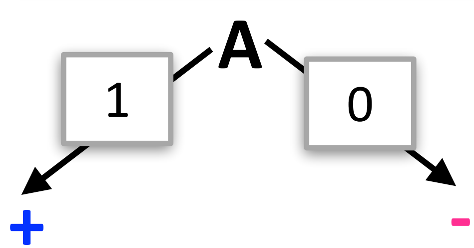
< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 0 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes



Why do we get purely labeled nodes?

Because (A=1,B=0) has no examples

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

< (A=0,B=0), - >: 50 examples

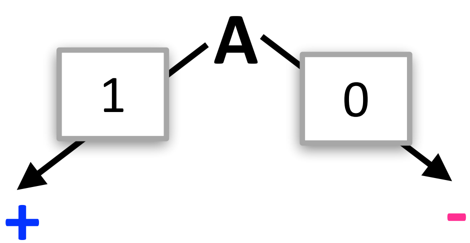
< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 0 examples

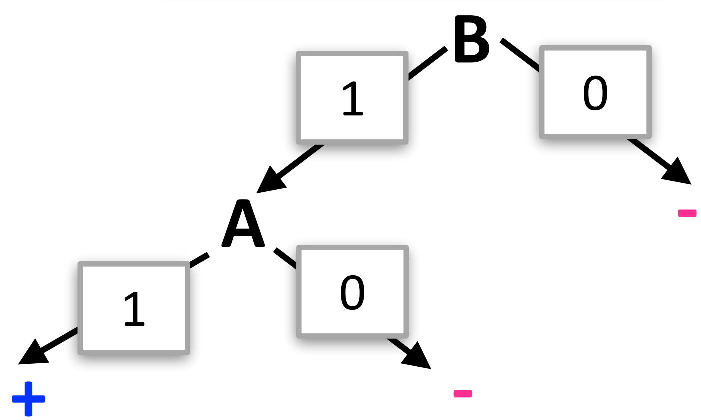
< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes



Splitting on B: we **don't** get purely labeled nodes



Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

< (A=0,B=0), - >: 50 examples

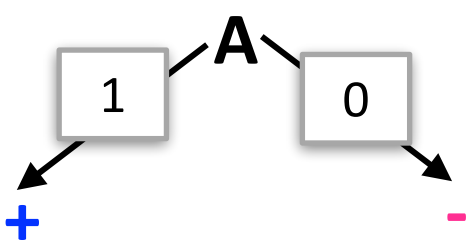
< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 0 examples

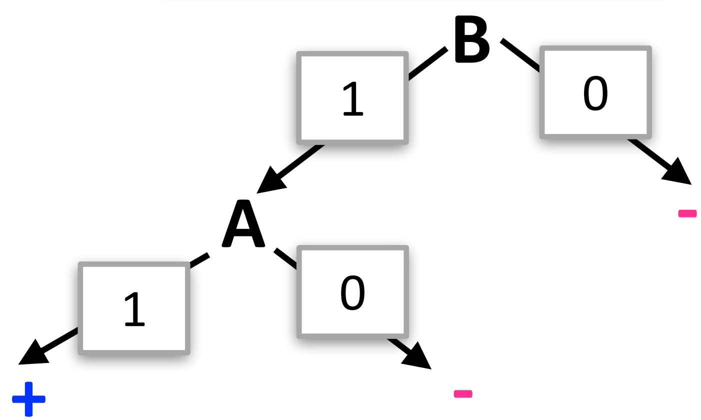
< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes



Splitting on B: we **don't** get purely labeled nodes



Which of the 2 trees is better?

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

< (A=0,B=0), - >: 50 examples

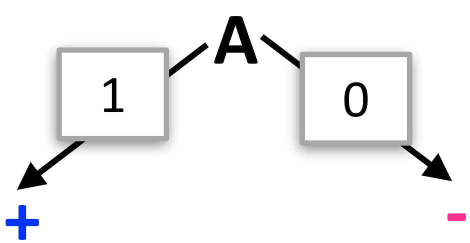
< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 0 examples

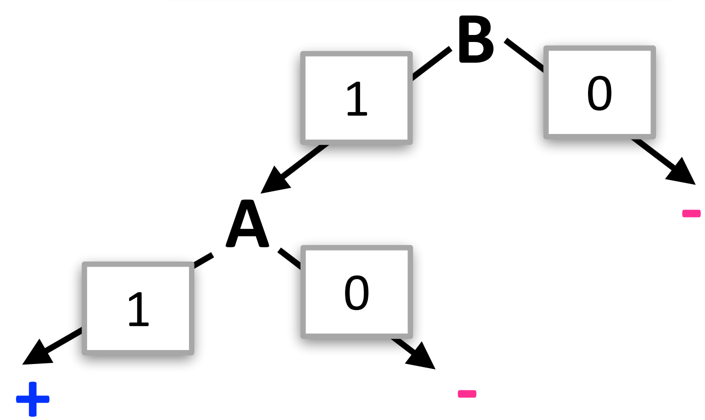
< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes



Splitting on B: we **don't** get purely labeled nodes



What if we have: <(A=1,B=0), - >: 3 examples?

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

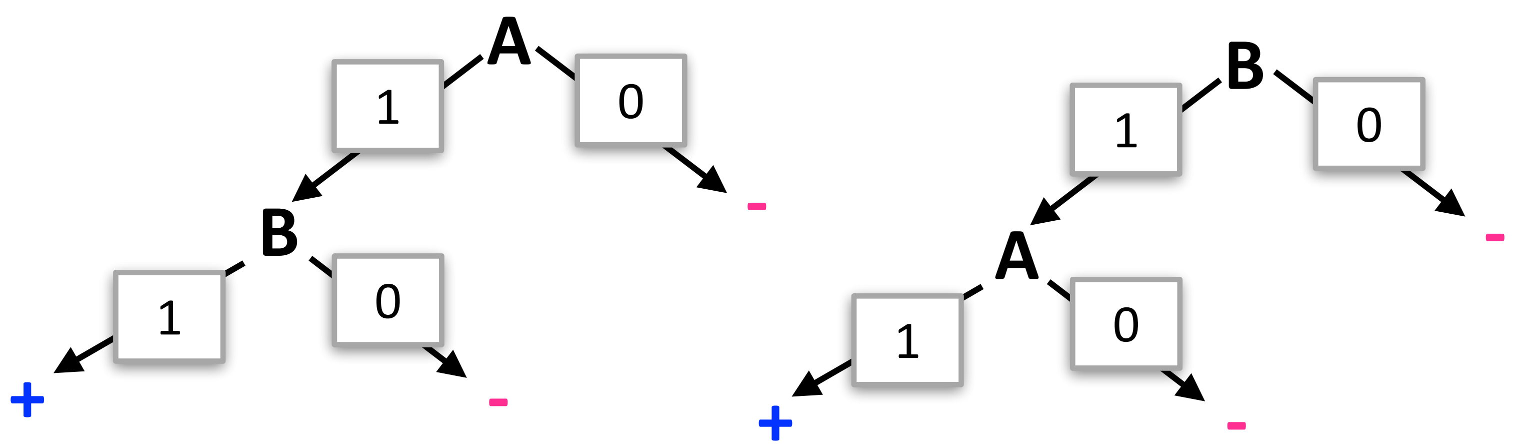
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

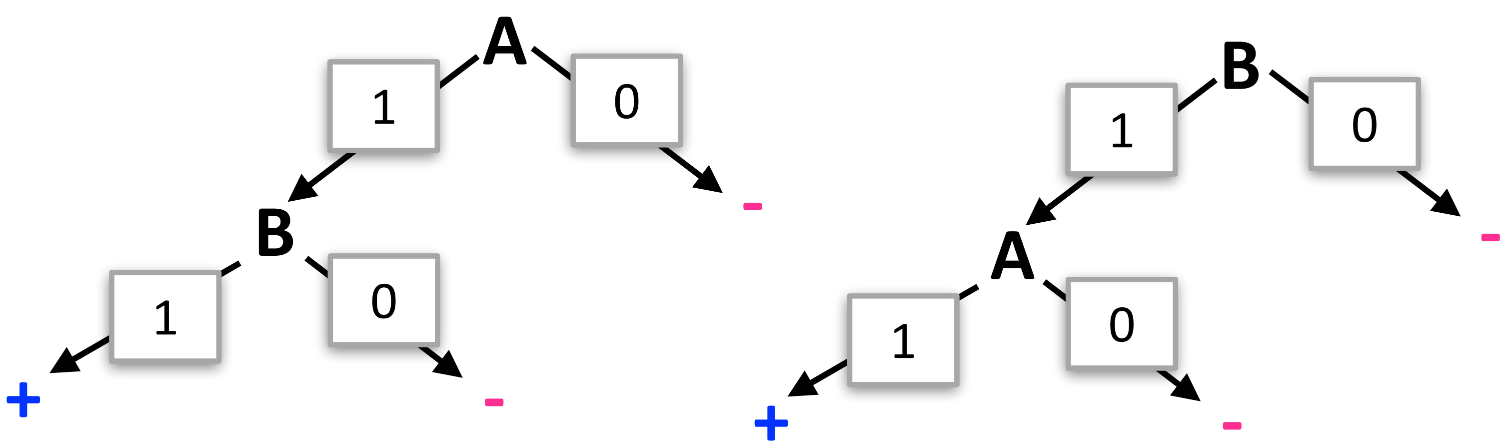
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Trees look structurally similar!

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

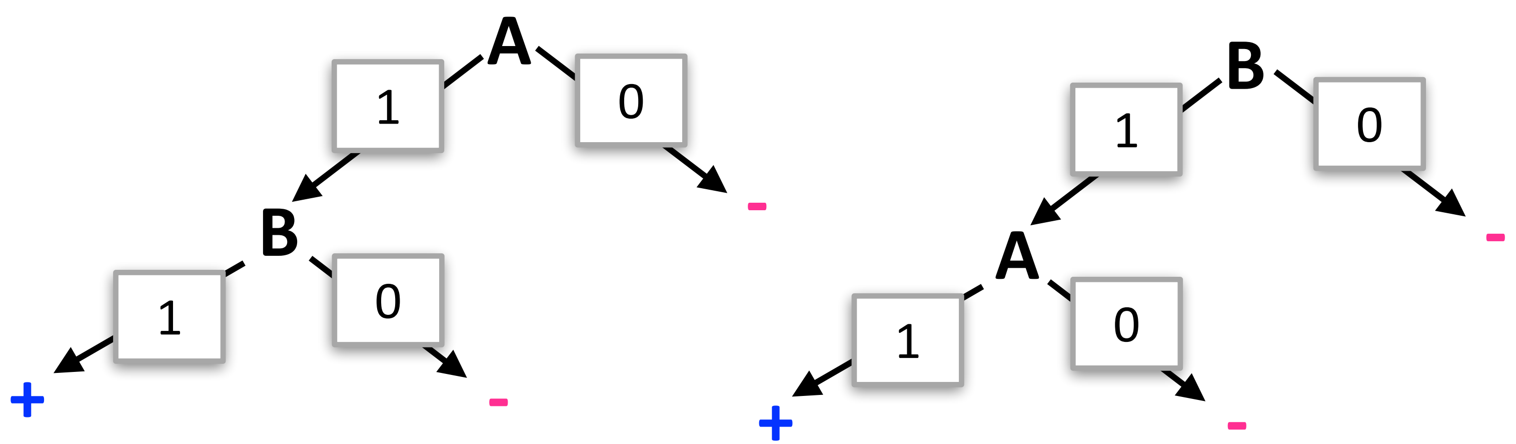
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Let's keep track of the examples used on each branch

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

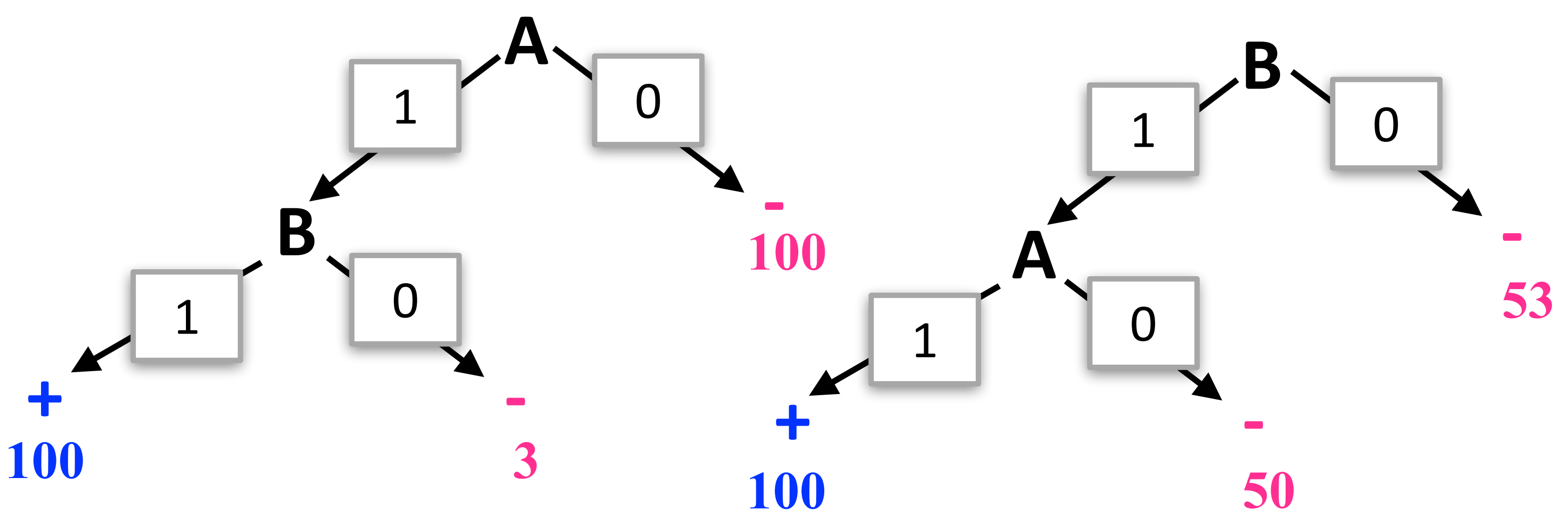
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



3 min: Which tree is better now? What do you think?

Picking the root/best attribute

Consider data with two Boolean attributes (A,B)

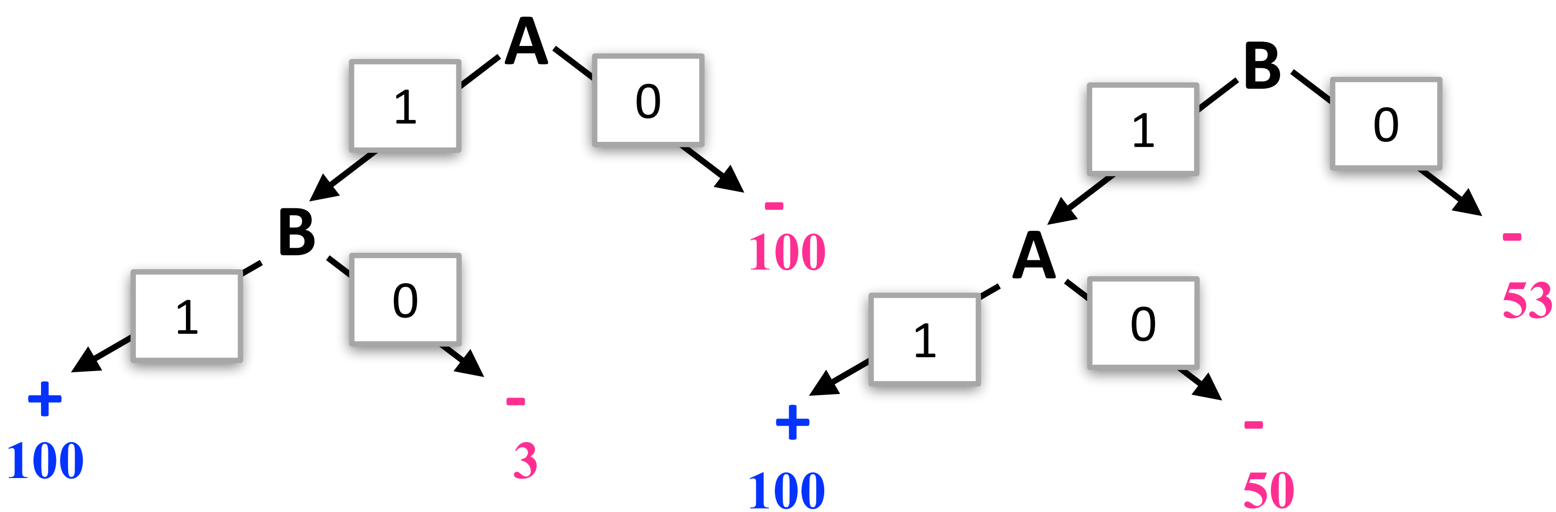
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



A as root has an advantage … let's quantify

Picking the root/best attribute

Goal: have the resulting decision tree be **as small as possible**

Picking the root/best attribute

Goal: have the resulting decision tree be **as small as possible**

Main decision in algorithm: select next attribute to split on

Picking the root/best attribute

Goal: have the resulting decision tree be **as small as possible**

Main decision in algorithm: select next attribute to split on

We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node

Picking the root/best attribute

Goal: have the resulting decision tree be **as small as possible**

Main decision in algorithm: select next attribute to split on

We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node

The most popular heuristic is information gain, originated with the ID3 system of Quinlan

Entropy

Entropy (information, impurity, disorder, randomness) of a set of examples, S, with respect to binary classification is

p_+: proportion of positive examples in S

p_-: proportion of negative examples in S

S is a set of examples. Every example in set has one of 2 labels: + or -

Entropy

Entropy (information, impurity, disorder, randomness) of a set of examples, S, with respect to binary classification is

p_+: proportion of positive examples in S

p_-: proportion of negative examples in S

What do p_+ and p_- sum to?

Entropy

Entropy (information, impurity, disorder, randomness) of a set of examples, S, with respect to binary classification is

p_+: proportion of positive examples in S

p_-: proportion of negative examples in S

$$Entropy(S) = H(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

Entropy

Entropy (information, impurity, disorder, randomness) of a set of examples, S, with respect to binary classification is

p_+: proportion of positive examples in S

p_-: proportion of negative examples in S

$$Entropy(S) = H(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

In general, for a discrete random variable with K possible values, with probabilities $\{p_1, p_2, \ldots, p_K\}$, the entropy is given by

$$H(\{p_1, p_2, \ldots, p_K\}) = -\sum_i^K p_i \log_2 p_i$$

Entropy is for a random variable: how much uncertainty is there in the values random variable takes on?

Minimum entropy

Entropy is measure of information, impurity, disorder, randomness

$$H(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

Minimum entropy is 0 (no randomness)

Occurs when $p_+ = 1$ (and $p_- = 0$) ***OR*** $p_+ = 0$ (and $p_- = 1$)

$$H(S) = -1 \log_2(1) - 0 \log_2(0) = 0$$

$$H(S) = -\underbrace{0 \log_2(0)}_{\text{By convention, 0}} - 1 \log_2(1) = 0$$

Maximum entropy

Entropy is measure of information, impurity, disorder, randomness

$$H(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

Maximum entropy is 1 for binary random variable (maximum randomness)

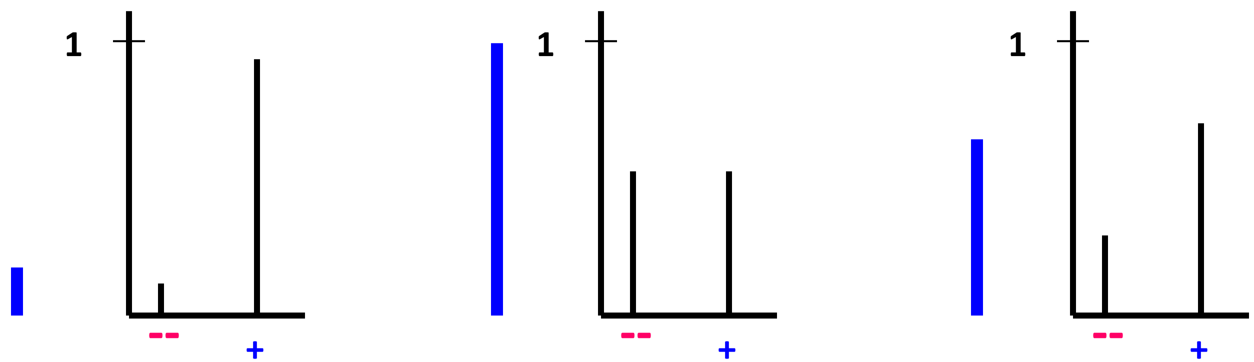
Occurs when $p_+ = 0.5$ (and $p_- = 0.5$)

$$\begin{aligned} H(S) &= -0.5\log_2(0.5) - 0.5\log_2(0.5) \\ &= -0.5(-1) - 0.5(-1) = 1 \end{aligned}$$

Entropy

High entropy: high level of uncertainty

Low entropy: low level of uncertainty

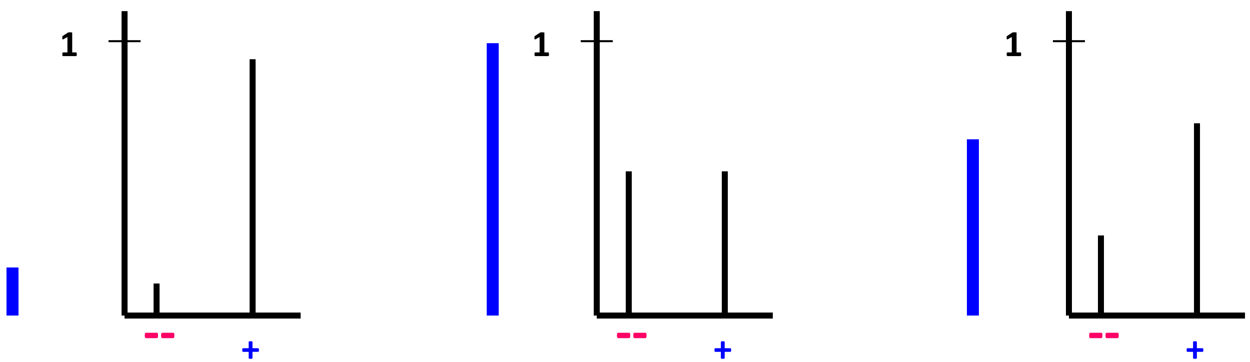


low entropy high entropy medium entropy

Entropy

High entropy: high level of uncertainty

Low entropy: low level of uncertainty

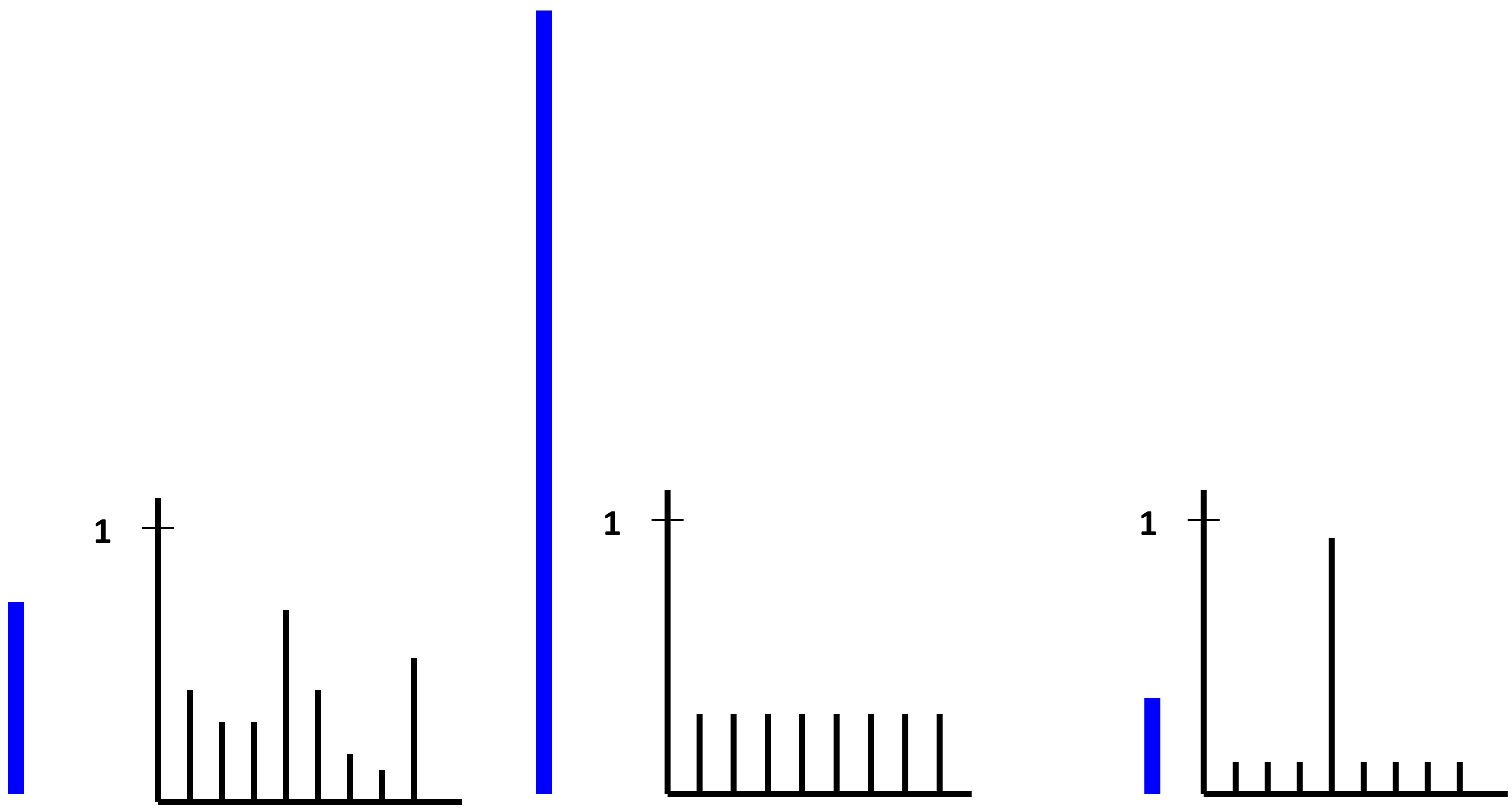


low entropy high entropy medium entropy

Uniform distribution has highest entropy

High entropy: high level of uncertainty

Low entropy: low level of uncertainty



Picking the root attribute

Goal: have the resulting decision tree be **as small as possible**

Main decision in algorithm: select next attribute to split on

We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node

The most popular heuristics is information gain, originated with the ID3 system of Quinlan

Picking the root attribute

Goal: have the resulting decision tree be **as small as possible**

Main decision in algorithm: select next attribute to split on

We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node

The most popular heuristics is information gain, originated with the ID3 system of Quinlan

Intuition: choose attribute that reduces the label entropy the most

Information gain

Information gain of an attribute A is the expected reduction in entropy caused by partitioning on this attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Tells us how important a given attribute of the feature vector is

Information gain

Information gain of an attribute A is the expected reduction in entropy caused by partitioning on this attribute

S_v is the subset of examples in S for which attribute A is set to value v

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Information gain

Information gain of an attribute A is the expected reduction in entropy caused by partitioning on this attribute

Over distribution of labels in full set S

Over distribution of labels just in subset S_v

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Always ≥ 0

Information gain

Information gain of an attribute A is the expected reduction in entropy caused by partitioning on this attribute

$$Gain(S, A) = Entropy(S) - \underbrace{\sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)}$$

Entropy of partitioning the data is calculated by **weighing the entropy of each partition** by its size relative to the original set. Partitions of low entropy (imbalanced splits) lead to high gain

Picking the Root Attribute

Consider data with two Boolean attributes (A,B)

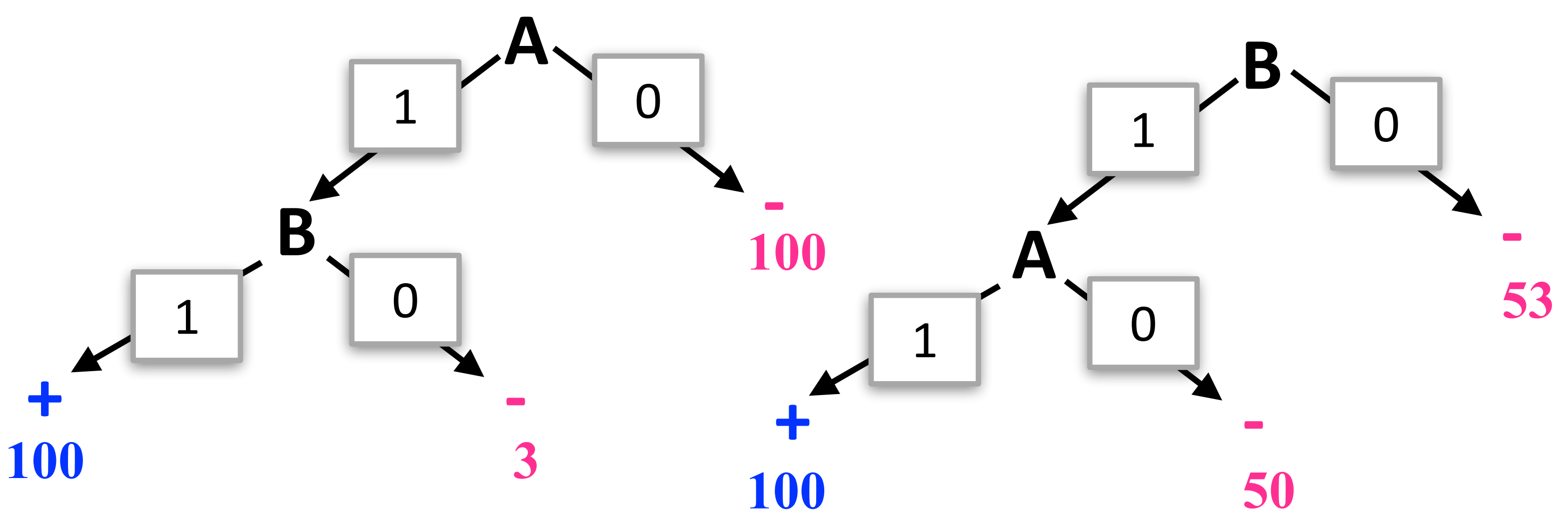
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

< (A=1,B=0), - >: 3 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?



Can see now why A is a better split! Gives more imbalanced split than splitting on B first

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: **S**unny, **O**vercast, **R**ainy

Temperature: **H**ot, **M**edium, **C**ool

Humidity: **H**igh, **N**ormal, **L**ow

Wind: **S**trong, **W**eak

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Current entropy:

positive = 9/14

negative = 5/14

$$H(\text{Play?}) =$$
$$-(9/14)\log_2(9/14)$$
$$-(5/14)\log_2(5/14)$$
$$H(\text{Play?}) = 0.94$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Should we split on outlook?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = Sunny: 5 of 14 examples

p = 2/5 n = 3/5 $H_S = .971$

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Should we split on outlook?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = Sunny: 5 of 14 examples

p = 2/5 n = 3/5 $H_S = .971$

Outlook = Overcast: 4 of 14 examples

p = 4/4 n = 0/4 $H_O = 0$

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Should we split on outlook?

	O	**T**	**H**	**W**	**Play?**
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = Sunny: 5 of 14 examples

p = 2/5 n = 3/5 $H_S = .971$

Outlook = Overcast: 4 of 14 examples

p = 4/4 n = 0/4 $H_O = 0$

Outlook = Rainy: 5 of 14 examples

p = 3/5 n = 2/5 $H_S = .971$

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Should we split on outlook?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Outlook = Sunny: 5 of 14 examples

p = 2/5 n = 3/5 $H_S = .971$

Outlook = Overcast: 4 of 14 examples

p = 4/4 n = 0/4 $H_O = 0$

Outlook = Rainy: 5 of 14 examples

p = 3/5 n = 2/5 $H_S = .971$

Expected entropy:

$$(5/14) \times .971 + (4/14) \times 0 + (5/14) \times .971$$

$$= .694$$

Information gain:

$$.940 - .694 = .246$$

Should we split on humidity?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Humidity = High: 7 of 14 examples

p = 3/7 n = 4/7 $H_H = .985$

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Should we split on humidity?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Humidity = High: 7 of 14 examples

p = 3/7 n = 4/7 $H_H = .985$

Humidity = Normal: 7 of 14 examples

p = 6/7 n = 1/7 $H_N = .592$

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Should we split on humidity?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Humidity = High: 7 of 14 examples

p = 3/7 n = 4/7 $H_H = .985$

Humidity = Normal: 7 of 14 examples

p = 6/7 n = 1/7 $H_N = .592$

Expected entropy:

$$(7/14) \times .985 + (7/14) \times 0.592 = .7885$$

Information gain:

$$.940 - .7885 = .1515$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Which feature to split on?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

$$Gain(S, A) = Entropy(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Information gain:

Outlook: 0.246

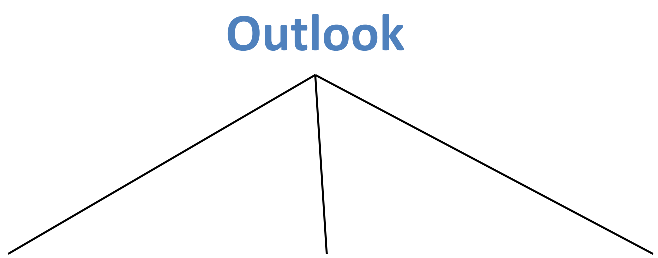
Humidity: 0.151

Wind: 0.048

Temperature 0.029

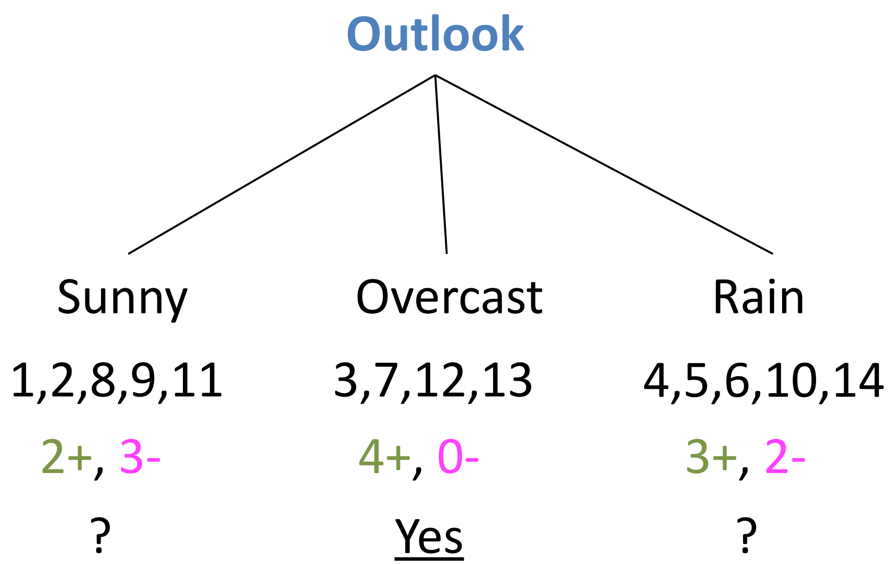
Split on outlook!

An illustrative example



Gain(S,Humidity)=0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

An illustrative example



Continue until either:
Every attribute is included in path
OR
All examples in the leaf have same label

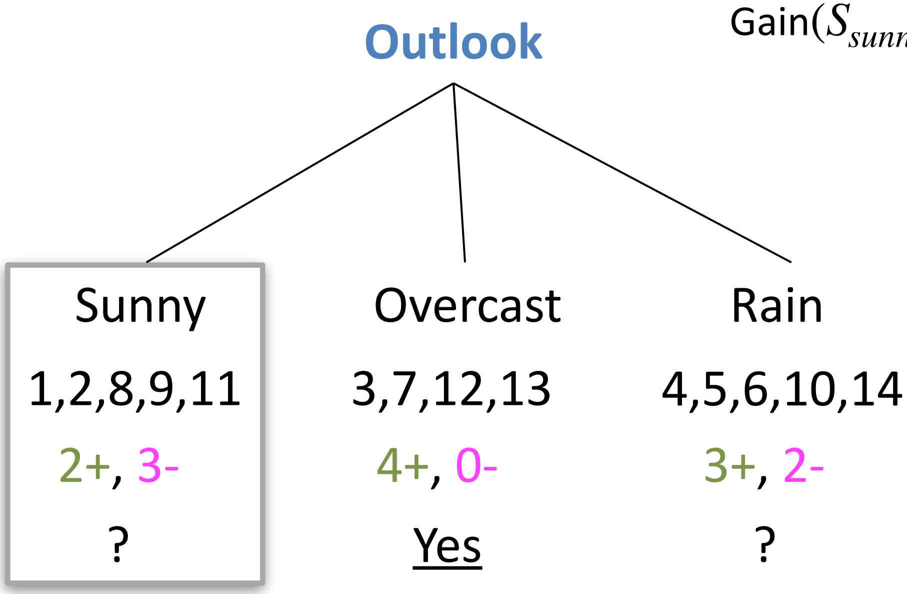
	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An illustrative example

$\text{Gain}(S_{sunny}, \text{Humidity}) = .97 - (3/5)0 - (2/5)0 = .97$

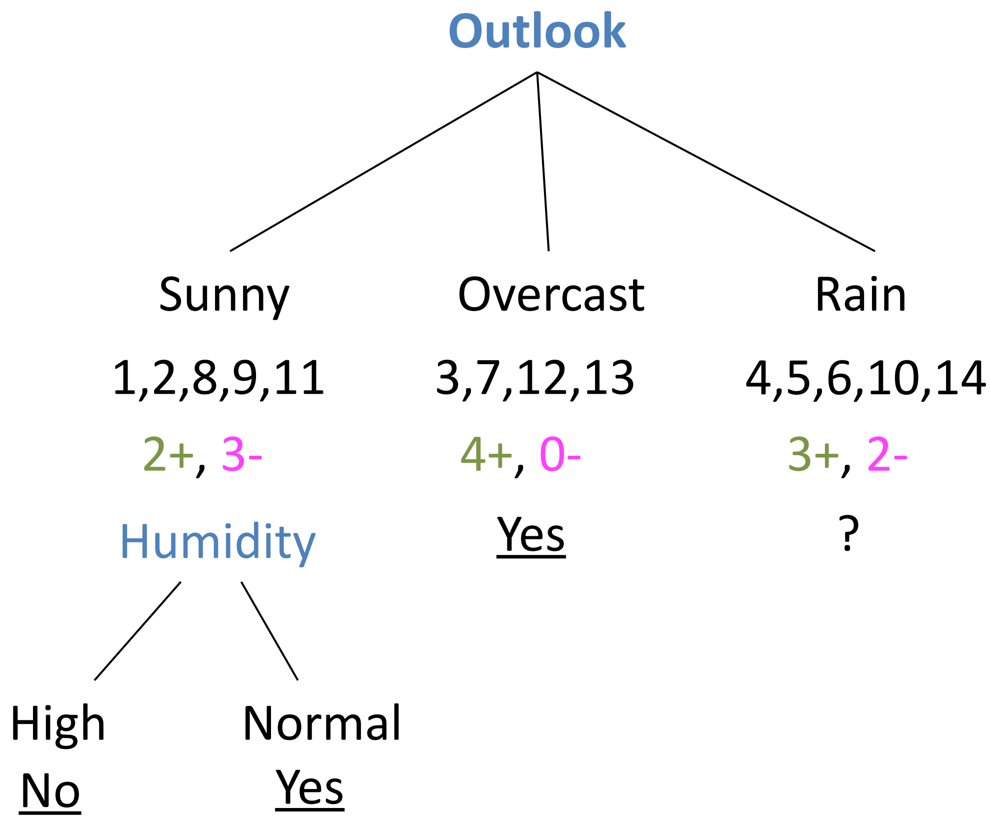
$\text{Gain}(S_{sunny}, \text{Temp}) = .97 - 0 - (2/5)1 = .57$

$\text{Gain}(S_{sunny}, \text{Wind}) = .97 - (2/5)1 - (3/5).92 = .02$

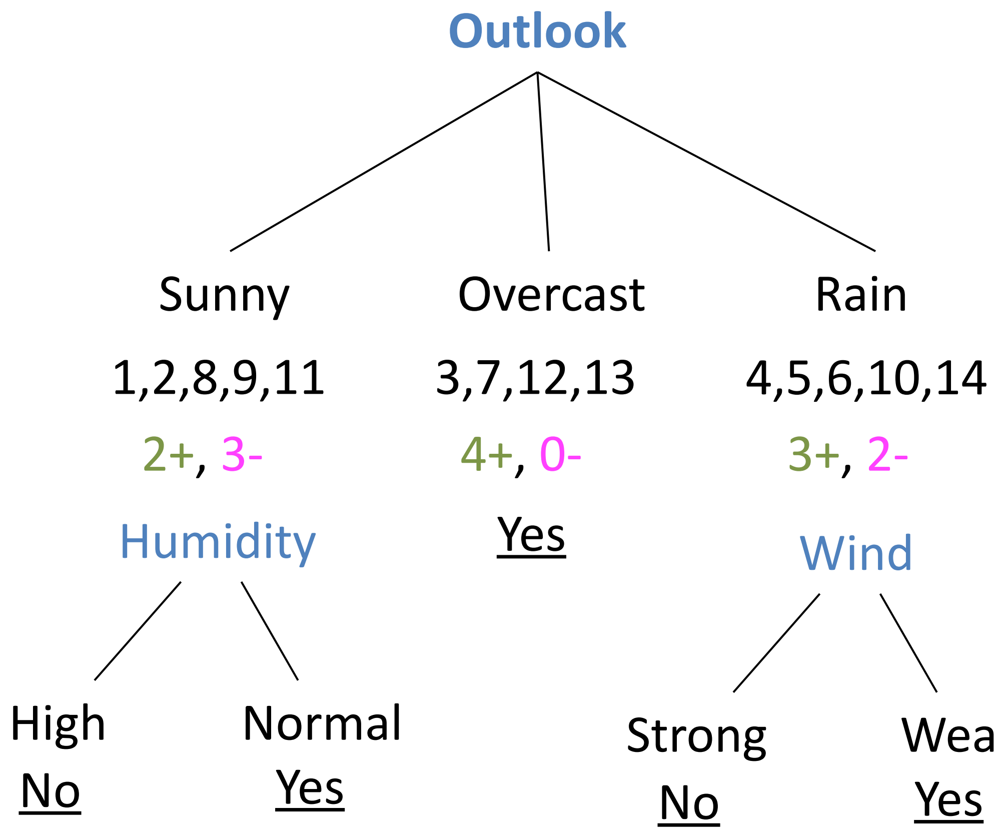


Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

An illustrative example



An illustrative example



Hypothesis space in decision tree induction

Search over decision trees, which can represent all possible discrete functions (has pros and cons)

Goal: to find the best decision tree

- Best could be “smallest depth”
- Best could be “minimizing the expected number of tests"

Finding a minimal decision tree consistent with a set of data is NP-hard

ID3 performs a greedy heuristic search (hill climbing without backtracking)

Makes statistically based decisions using all data