Lecture 6: Learning Decision Trees

COMP 411, Fall 2021

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) and Dan Roth (Penn)

Today's Topics

Homework 3 out

- Due Thursday, September 30 by 11:59p

Learning decision trees (ID3 algorithm)

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

Avoiding overfitting

- Some experimental issues

vumanfredi@wesleyan.edu

Decision Trees

LEARNING

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

   2. $A_b \in A$ is the attribute that ***best*** classifies S
   3. For each possible value v that A_b can take on

      Decide what to do for each value root attribute takes

      - 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

        Why?

        For generalization at test time

      Else: below this branch add the subtree ID3($S_v, A - \{A_b\}$)

      Recursive call to the Id3 algorithm with all the remaining attributes

   4. Return root node R

Picking the Root Attribute

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

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

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 Attribute

Consider data with two Boolean attributes (A,B)

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

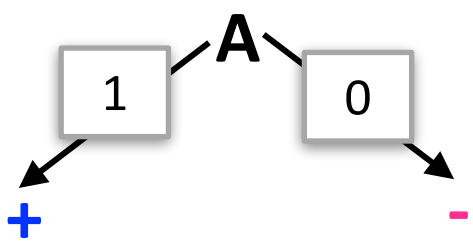
< (A=0,B=1), - >: 50 examples

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

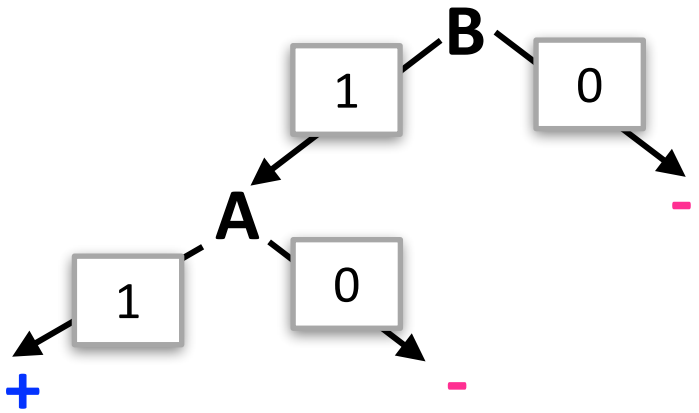
< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes



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



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

Picking the Root Attribute

Consider data with two Boolean attributes (A,B)

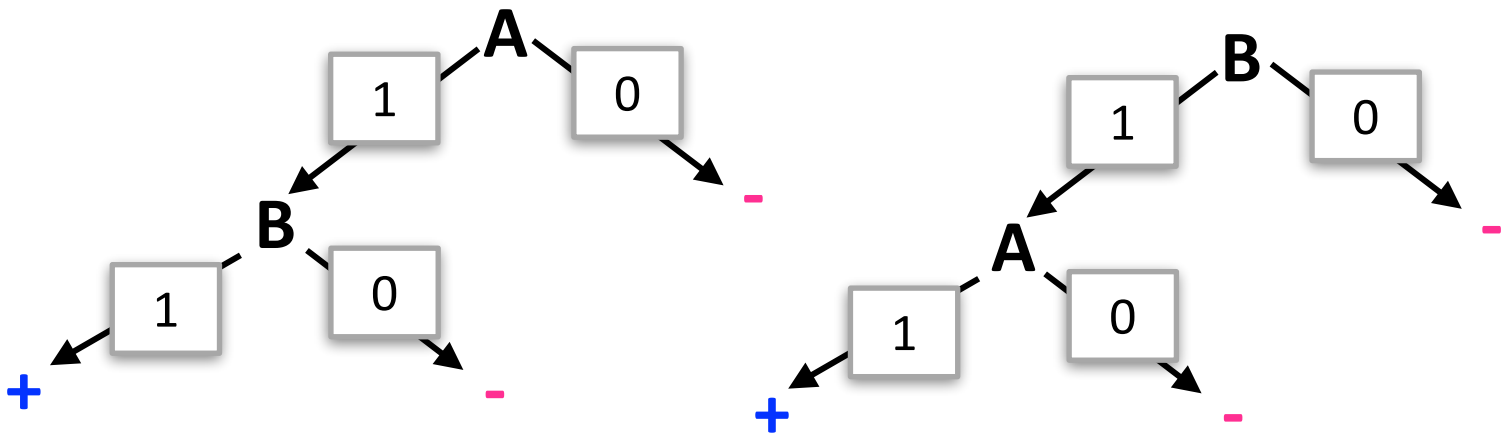
< (A=0,B=0), - >: 50 examples

< (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?



Trees look structurally similar!

Picking the Root Attribute

Consider data with two Boolean attributes (A,B)

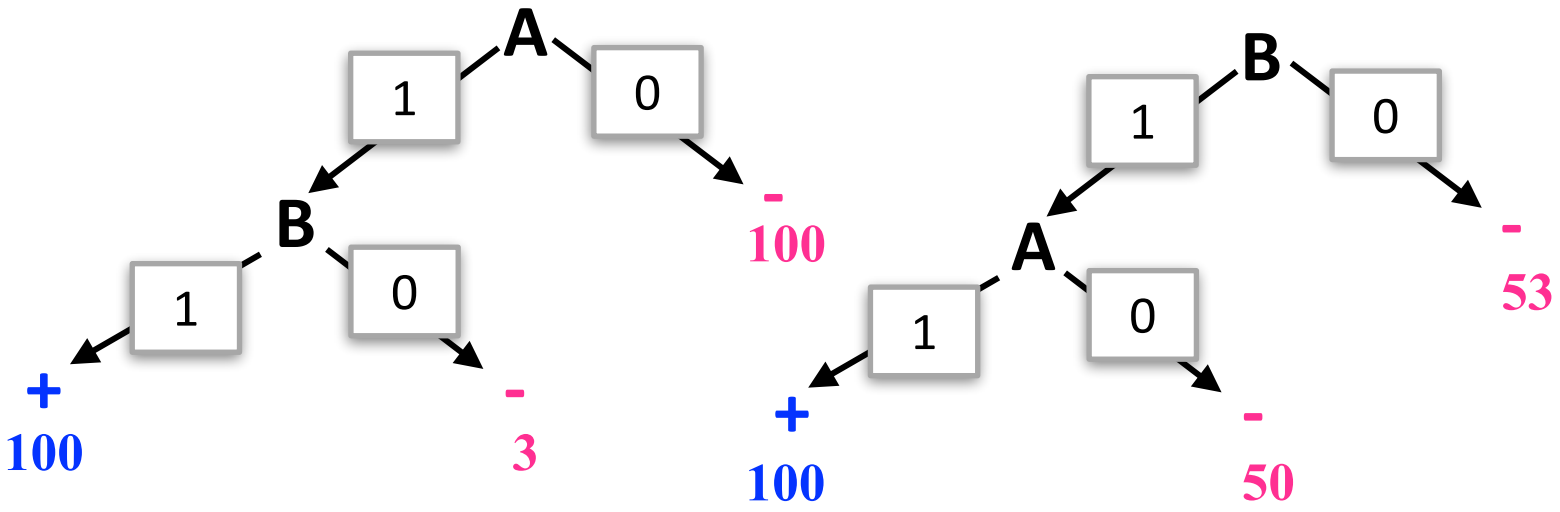
< (A=0,B=0), - >: 50 examples

< (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?



A as root has an advantage … let's quantify

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

Entropy

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

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

p_+: proportion of positive examples in S

p_-: proportion of negative examples in S

In general, for a discrete probability distribution 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

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

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

p_+: proportion of positive examples in S

p_-: proportion of negative examples in S

Minimum entropy is 0 (no randomness)

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

Maximum entropy (for binary r.v.) is 1

Occurs when $p_+ = p_- = \frac{1}{2}$

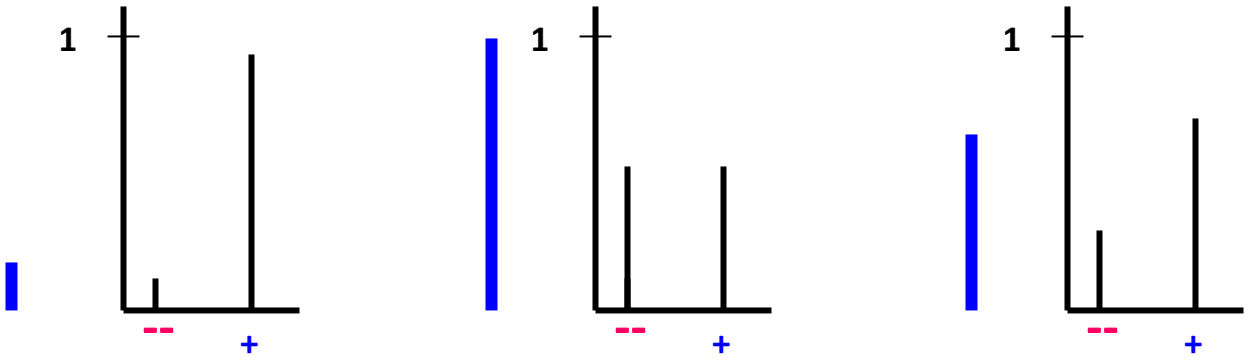
Entropy

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

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

p_+: proportion of positive examples in S

p_-: proportion of negative examples in S

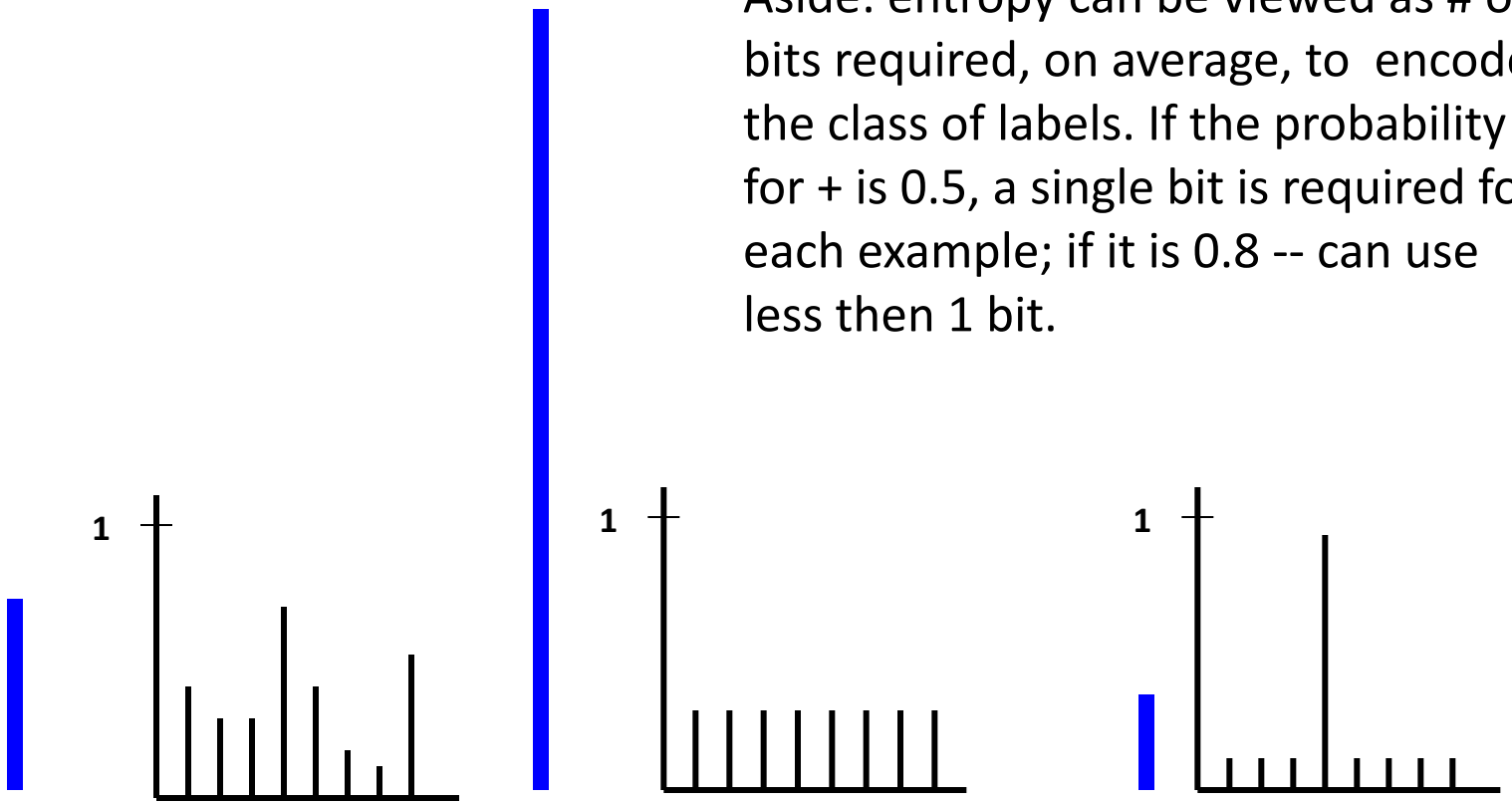


Uniform distribution has highest entropy

High entropy: high level of uncertainty

Low entropy: low level of uncertainty

Aside: entropy can be viewed as # of bits required, on average, to encode the class of labels. If the probability for + is 0.5, a single bit is required for each example; if it is 0.8 -- can use less then 1 bit.

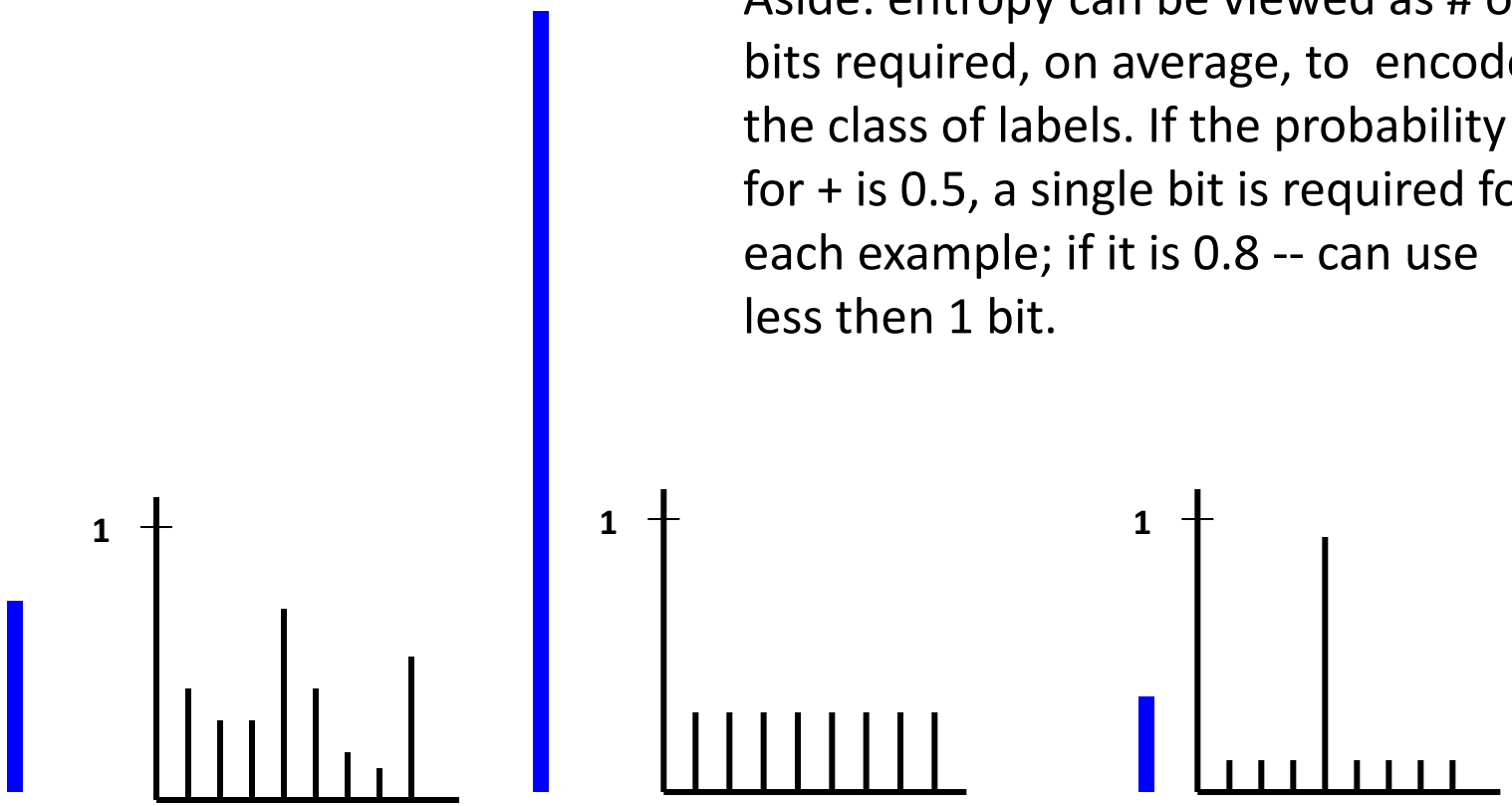


Picking the root attribute

High entropy: high level of uncertainty

Low entropy: low level of uncertainty

Aside: entropy can be viewed as # of bits required, on average, to encode the class of labels. If the probability for + is 0.5, a single bit is required for each example; if it is 0.8 -- can use less then 1 bit.



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

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

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

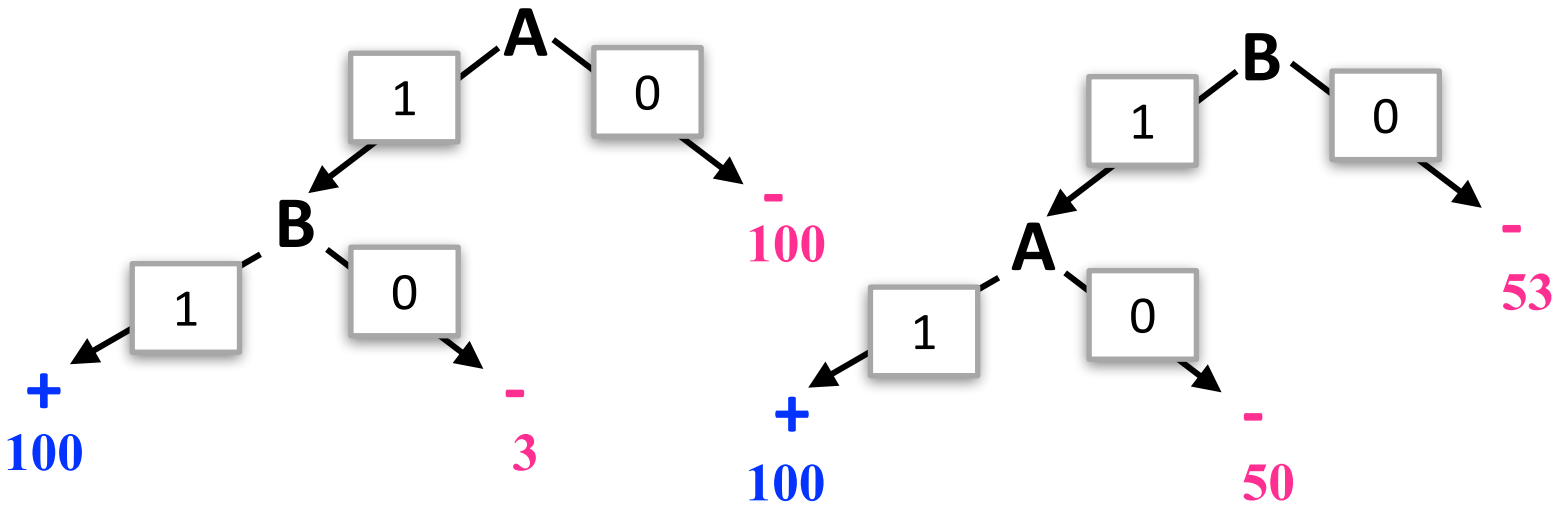
< (A=0,B=0), - >: 50 examples

< (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!

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

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

$$H(\text{Play?}) = 0.94$$

Information gain: 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 $H_S = .971$

Information gain: 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 $H_O = 0$

Information gain: 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 $H_S = .971$

Information gain: 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 $H_S = .971$

Expected entropy:

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

$$= .694$$

Information gain:

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

Information gain: 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 $H_H = .985$

Information gain: 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 $H_N = .592$

Information gain: 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:

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

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	-

Information gain:

Outlook: 0.246

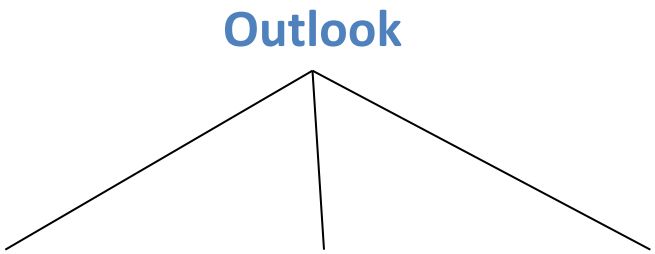
Humidity: 0.151

Wind: 0.048

Temperature 0.029

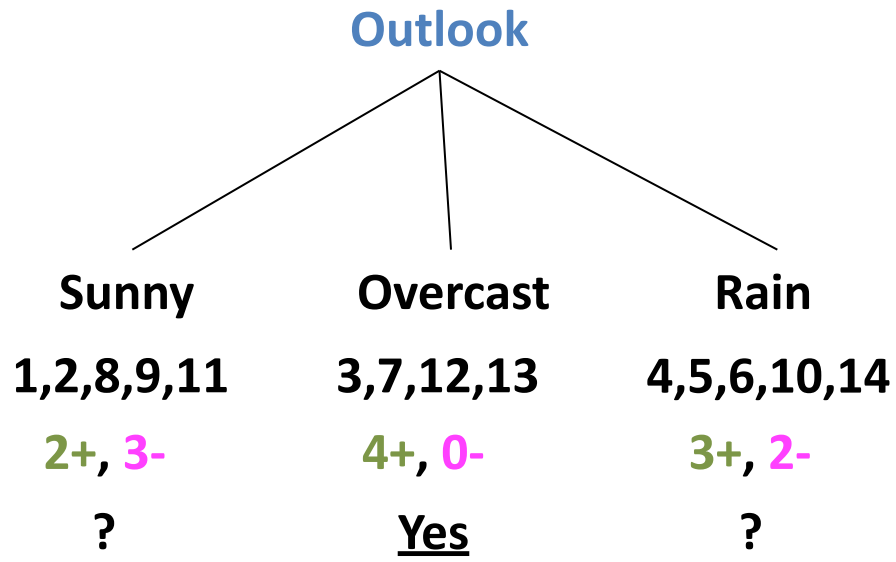
Split on outlook!

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



	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	-

Continue until either:

Every attribute is included in path

OR

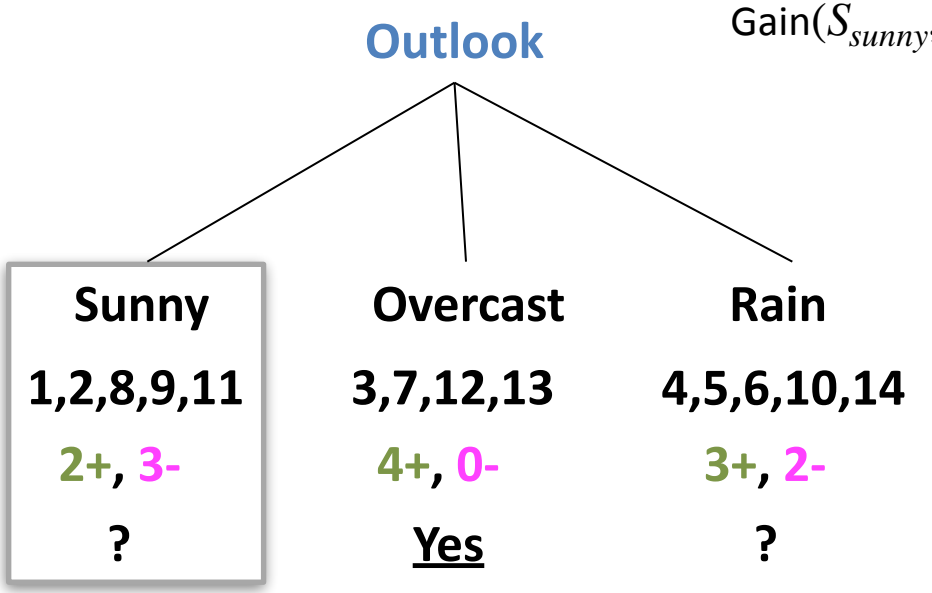
All examples in the leaf have same label

An illustrative example

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

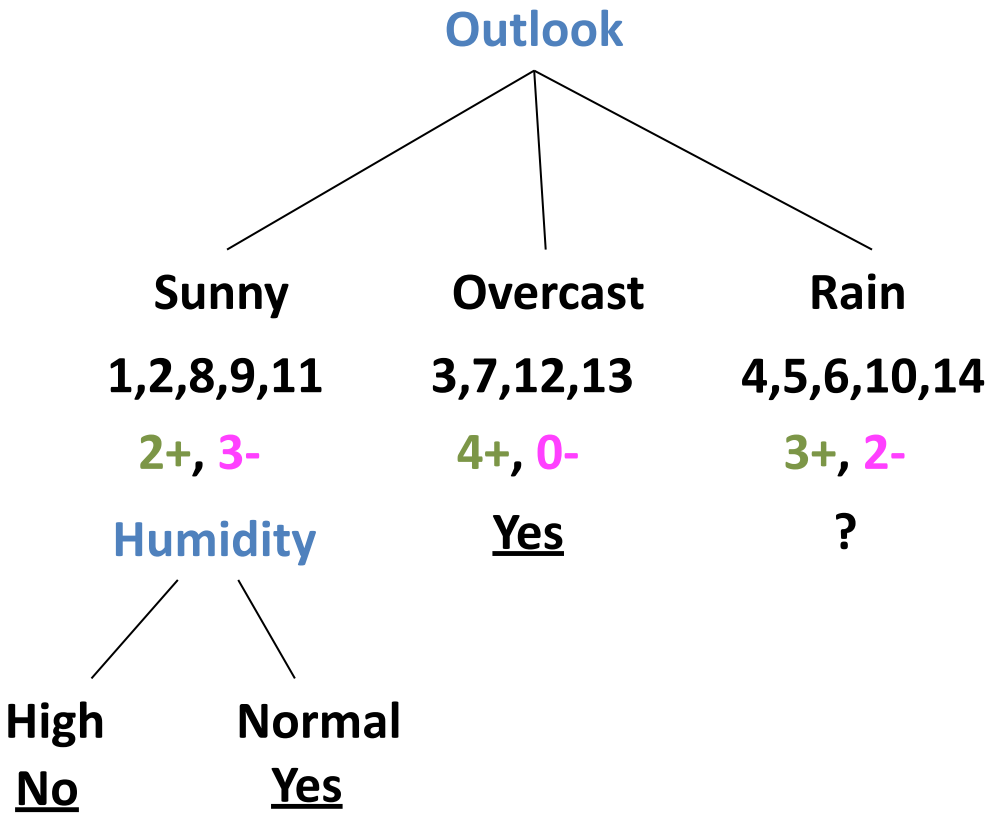
$\text{Gain}(S_{sunny}, \text{Temp}) = .97 - 0 - (2/5)1 = .57$

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

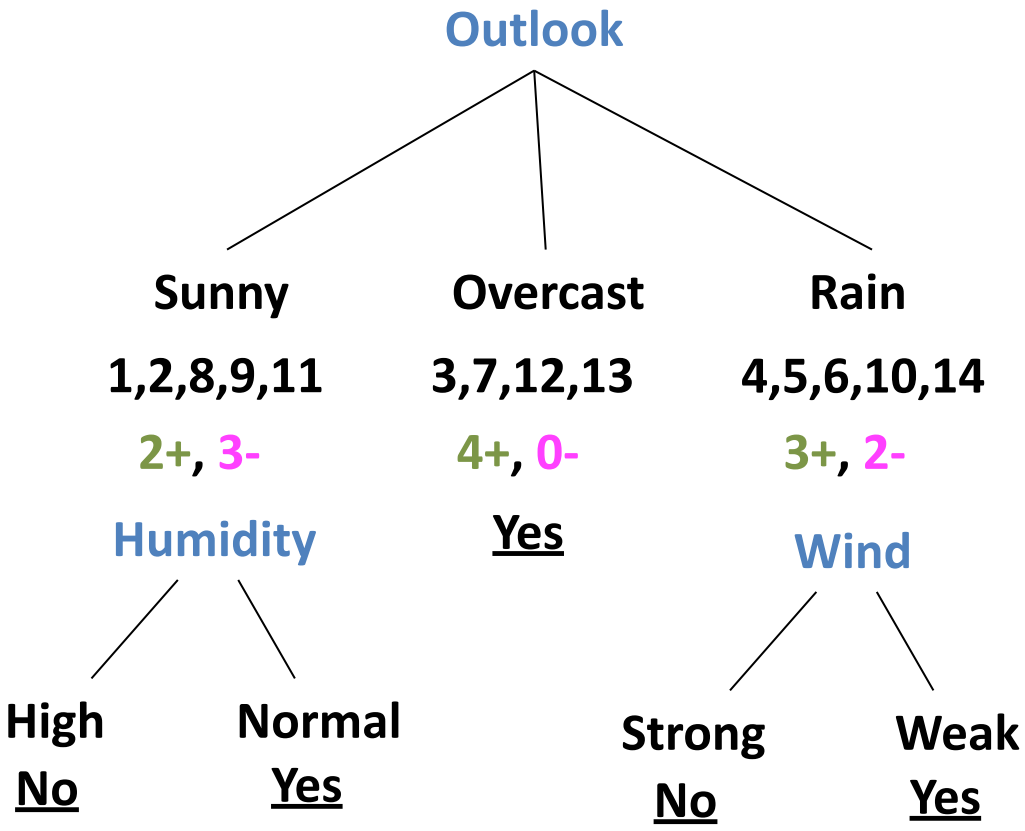


Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

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)

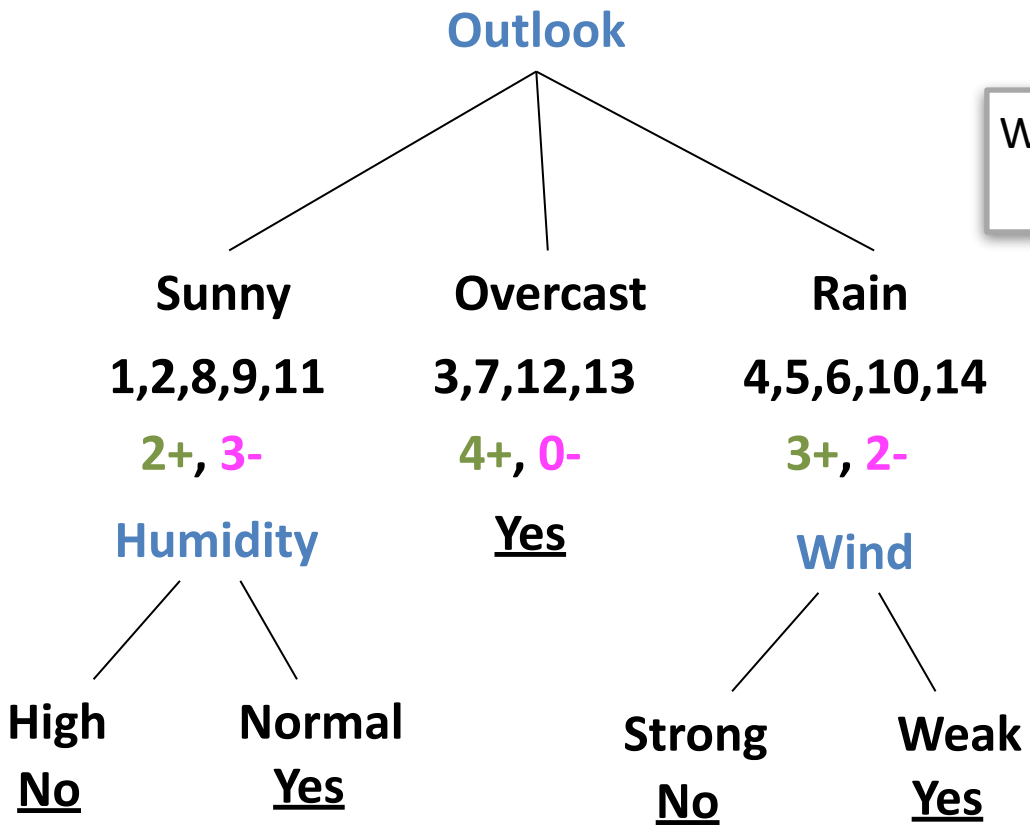
Makes statistically based decisions using all data

Decision Trees

AVOIDING OVERFITTING

Example

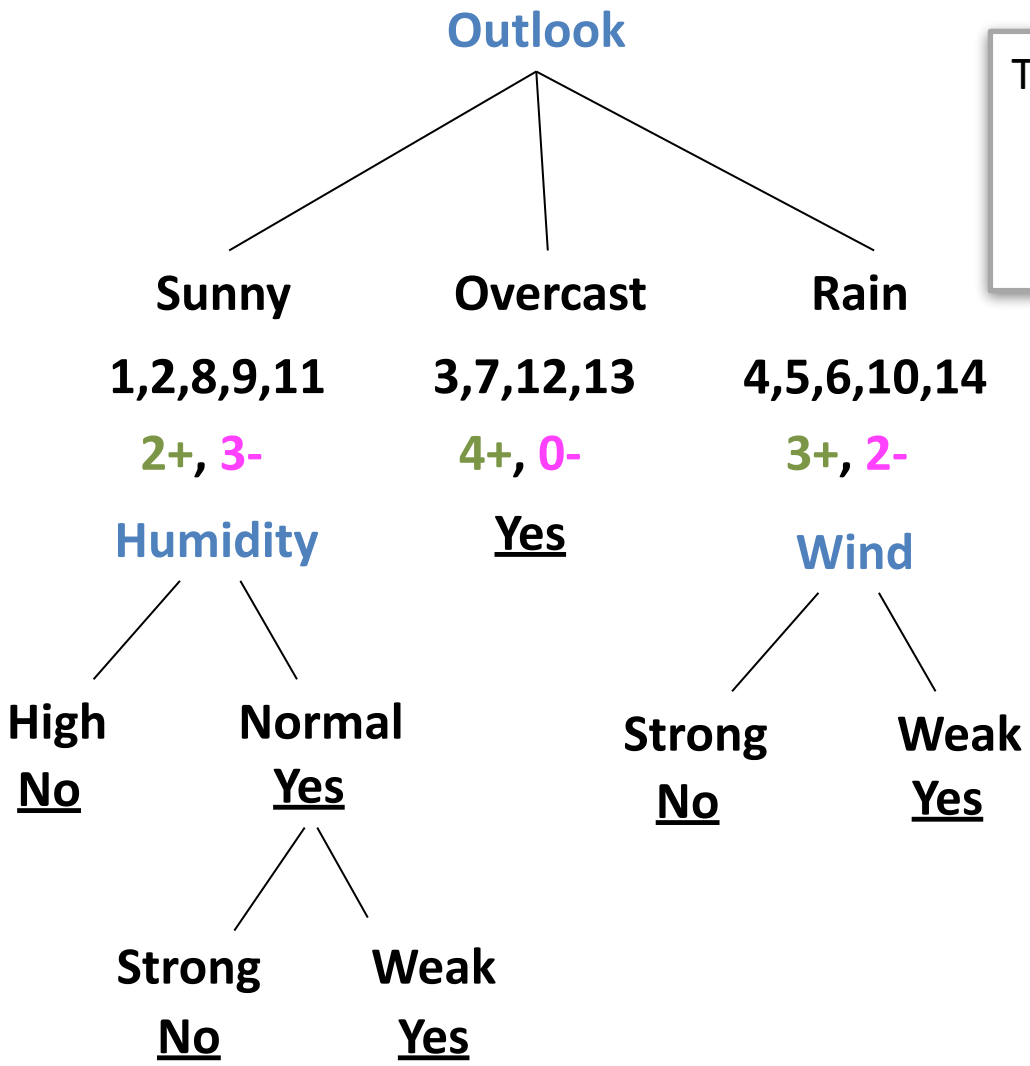
Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, **No**



What is the decision tree for this function?

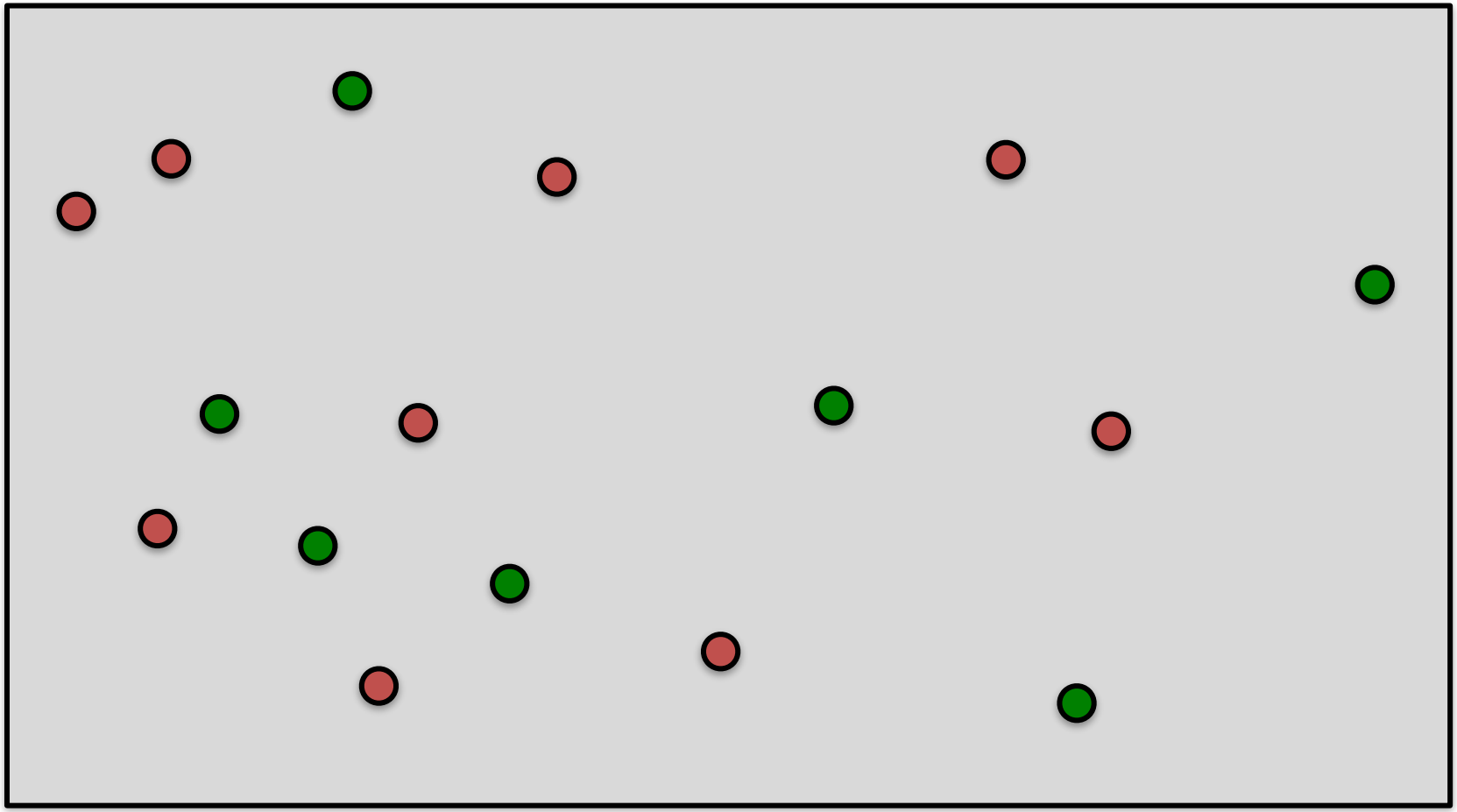
Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, **No**

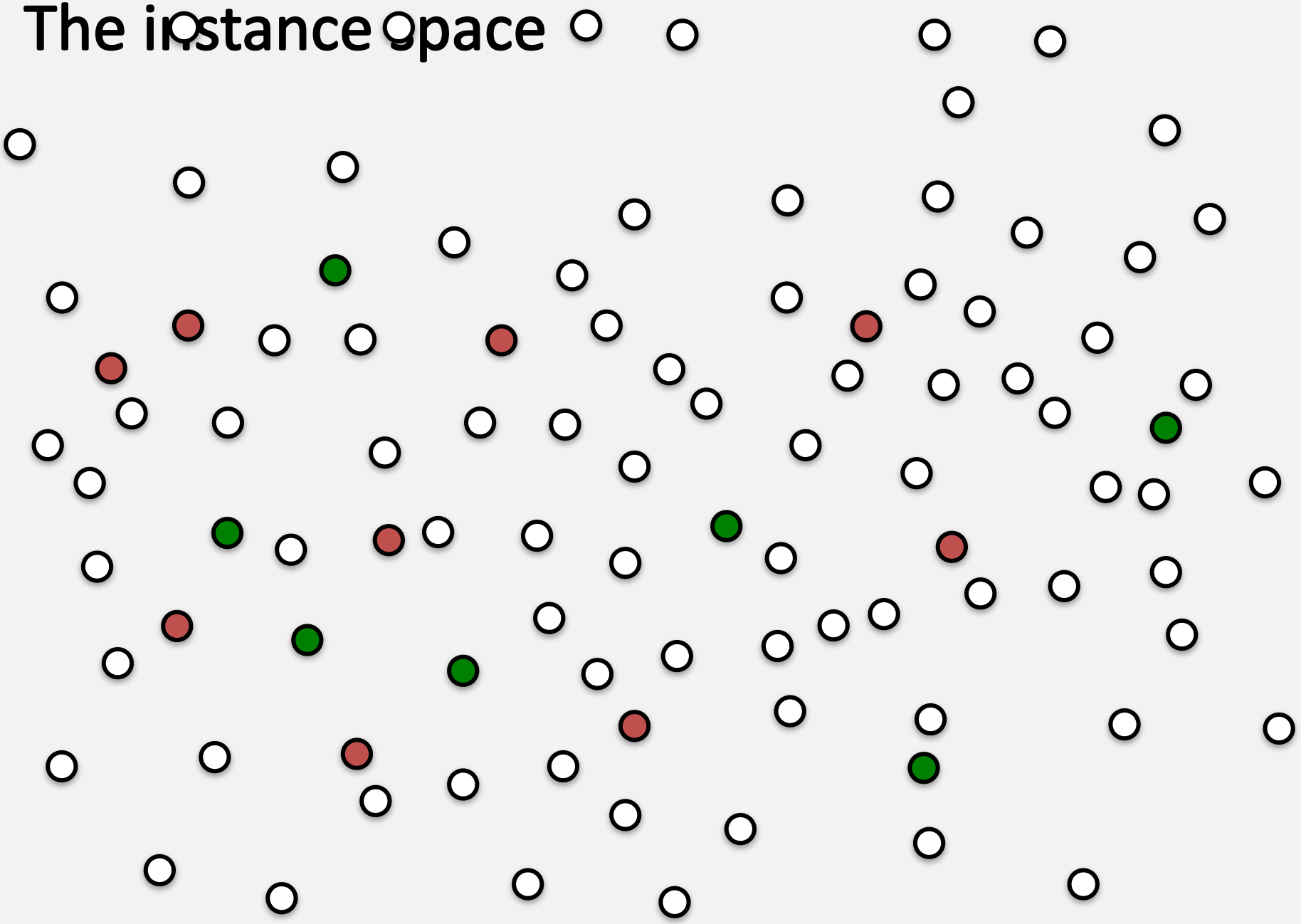


This can always be done but may fit noise or other coincidental regularities

Our training data



The instance space

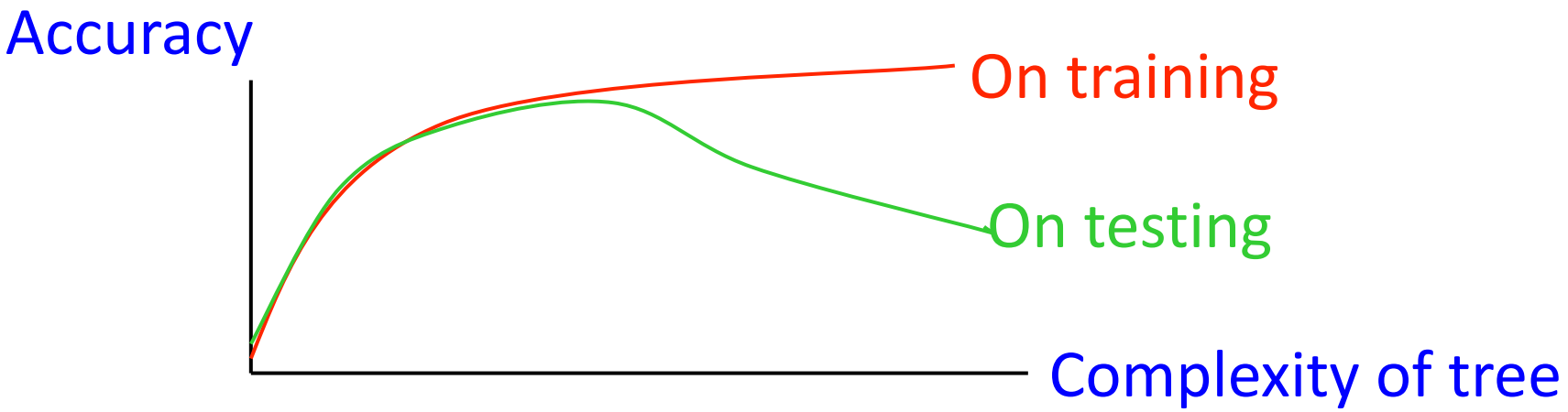


Overfitting the data

Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.

- There may be noise in the training data the tree is fitting
- The algorithm might be making decisions based on very little data

A hypothesis h is said to overfit the training data if there is another hypothesis h', such that h has a smaller error than h' on the **training data** but h has larger error on the **test data** than h'



Reasons for overfitting

Too much variance in the training data

- Training data is not a representative sample of the instance space
- We split on features that are actually irrelevant

Too much noise in the training data

- Noise = some feature values or class labels are incorrect
- We learn to predict the noise

In both cases, it is a result of our will to minimize the empirical error when we learn, and the ability to do it (with DTs)

Pruning a decision tree

Prune

- Remove leaves and assign majority label of the parent to all items

Prune the children of S if:

- all children are leaves,

and

- the accuracy on the validation set does not decrease if we assign the most frequent class label to all items at S.

Avoiding overfitting

Two basic approaches

- ***Pre-pruning***: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
- ***Post-pruning***: Grow the full tree and then remove nodes that seem not to have sufficient evidence.

Methods for evaluating subtrees to prune

- ***Cross-validation***: Reserve hold-out set to evaluate utility
- ***Statistical testing***: Test if the observed regularity can be dismissed as likely to occur by chance
- ***Minimum Description Length***: Is the additional complexity of the hypothesis smaller than remembering the exceptions?

Next: a brief detour into explaining generalization and overfitting

Avoiding overfitting with decision trees

Occam's Razor

- Favor simpler (in this case, shorter) hypotheses
- Why? Fewer shorter trees, less likely to fit better by coincidence

Approach 1: Fix the depth of the tree

- ***Decision stump*** = a decision tree with only one level
- Typically will not be very good by itself
- But, we will revisit decision stumps later (short decision trees can make very good features for a second layer of learning)

Avoiding overfitting with decision trees

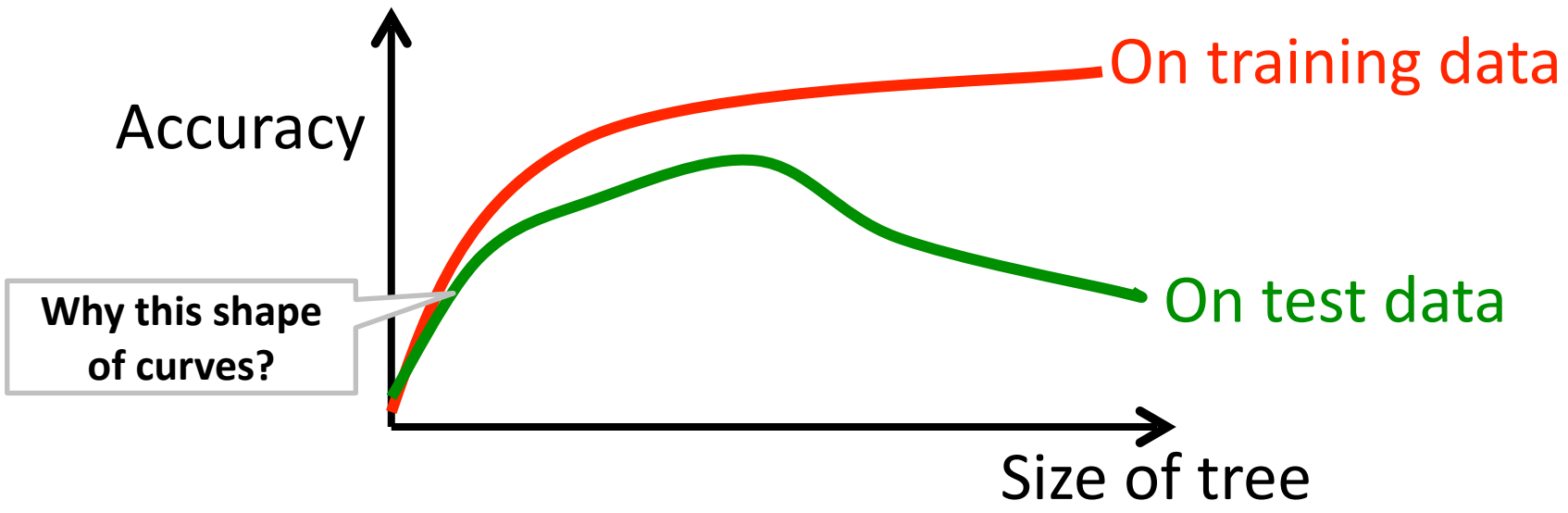
Occam's Razor

- Favor simpler (in this case, shorter) hypotheses
- Why? Fewer shorter trees, less likely to fit better by coincidence

Approach 2: Optimize on a ***held-out set*** (also called ***development set*** or ***validation set***) while growing the trees

- Split your data into two parts: training set and held-out set
- Grow your free on training split and check the performance on held-out set after every new node is added
- If growing the tree hurts validation set performance, stop growing

Overfitting



A decision tree overfits the training data when its accuracy on the training data goes up but its accuracy on unseen data goes down

Summary: Decision Trees

Popular machine learning tool

- Prediction is easy
- If we have Boolean features and binary classification, decision trees can represent any Boolean function

Greedy heuristics for learning

- ID3 algorithm (using information gain)

Decision trees are prone to overfitting unless you take care to avoid it