Decision Trees

Lecture 3

Machine Learning
Fall 2015

Some slides from Tom Mitchell, Dan Roth and others

Announcements

TA Office hours

Who	When	Where
Abhinay Duppelly	Monndays 11 AM - 12 AM	MEB 3115
Nic Bertagnolli	~~Wednesdays 10 AM - 11 AM~~	MEB 3115
Xingyuan Pan	Fridays 4:30 PM - 5:30 PM	MEB 3115

Tomorrow 4:30 - 5:30

Announcements

- Quiz #0 due on 3rd September (available on Canvas)
- Please fill out survey on Canvas
  - Optional
- Contact me after class if there are any registration issues
- Added more resources on website
  - A new online book

So far…

On using supervised learning

✓ What is our instance space?

What are the inputs to the problem? What are the features?

✓ What is our label space?

What is the learning task?

✓ What is our hypothesis space?

What functions should the learning algorithm search over?

4. What is our learning algorithm?

How do we learn from the labeled data?

5. What is our loss function or evaluation metric?

What is success?

Restricting the search space for learning

- A *hypothesis space* is the set of possible functions we consider
  - Choose a hypothesis space that is smaller than the space of all functions
    - Only simple conjunctions
    - m-of-n rules: Pick a set of n variables. At least m of them must be true
    - Linear functions
- How do we pick a hypothesis space?
  - Using some prior knowledge (or by guessing)
- What if the hypothesis space is so small that nothing in it agrees with the data?
  - We need a hypothesis space that is flexible enough

We could be wrong

- Our guess of the hypothesis space may be incorrect
- General strategy
  - Pick an expressive hypothesis space expressing concepts
    - Concept = the target classifier that is hidden from us. Sometimes we may even call it the oracle.
    - Example hypothesis spaces: m-of-n functions, decision trees, linear functions, grammars, multi-layer deep networks, etc
  - Develop algorithms that find an element the hypothesis space that fits data well (or well enough)
  - Hope that it generalizes

Key issues in machine learning

- Modeling

  How to formulate your problem as a machine learning problem? How to represent data? Which algorithms to use? What learning protocols?

- Representation

  Good hypothesis spaces and good features

- Algorithms
  - What is a good learning algorithm?
  - What is success?
  - Generalization vs overfitting
  - The computational question: How long will learning take?

Coming up… (the rest of the semester)

Different hypothesis spaces and learning algorithms

- Decision trees and the ID3 algorithm
- Nearest Neighbors
- Linear classifiers
  - Perceptron
  - Winnow
  - SVM
  - Logistic regression
- Combining multiple classifiers
  - Boosting, bagging

Coming up… (the rest of the semester)

Different hypothesis spaces and learning algorithms

- Decision trees and the ID3 algorithm
- Near
- Linea
  - P
  - W
  - S
  - L
- Com
  - B

Important issues to consider

1. What do these hypotheses represent?
2. Implicit assumptions and tradeoffs
3. Generalization?
4. How do we learn?

Today's lecture: Learning Decision Trees

1. **Representation**: What are decision trees?

2. **Algorithm**: Learning decision trees
    1. The ID3 algorithm: A greedy heuristic

    2. Some extensions

Representing data

Data can be represented as a big table, with columns denoting different attributes

Name	Label
Claire Cardie	−
Peter Bartlett	+
Eric Baum	−
Haym Hirsh	+
Shai Ben-David	−
Michael I. Jordan	+

Representing data

Data can be represented as a big table, with columns denoting different attributes

Name	Special character in name?	Second character of first name	Last character of last name	Gender	Label
Claire Cardie	No	l	e	Female	–
Peter Bartlett	No	e	t	Male	+
Eric Baum	No	r	m	Male	–
Haym Hirsh	No	a	h	Male	+
Shai Ben-David	Yes	h	d	Male	–
Michael I. Jordan	Yes	i	n	Male	+

Representing data

Data can be represented as a big table, with columns denoting different attributes

Name	Special character in name?	Second character of first name	Last character of last name	Gender	Label
Claire					
Peter					
Eric E					
Haym Hirsh	No	a	h	Male	+
Shai Ben-David	Yes	h	d	Male	−
Michael I. Jordan	Yes	i	n	Male	+

With these four attributes, how many unique rows are possible?

$2 \cdot 26 \cdot 26 \cdot 2 = 2074$

If there are 100 attributes, all binary, how many unique rows are possible?

2^{100}

Representing data

Data can be represented as a big table, with columns denoting different attributes

Name	Special character in name?	Second character of first name	Last character of last name	Gender	Label
Claire					
Peter					
Eric E					
Haym Hirsh	No	a	h	Male	+
Shai [illegible] David					−
Michael I. Jordan	Yes	i	n	Male	+

With these four attributes, how many unique rows are possible?

$2 \cdot 26 \cdot 26 \cdot 2 = 2704$

If there are 100 attributes, all binary, how many unique rows are possible?

2^{100}

We need to figure out how to represent in a better, more efficient way

What are decision trees?

A hierarchical data structure that represents data using a divide-and-conquer strategy

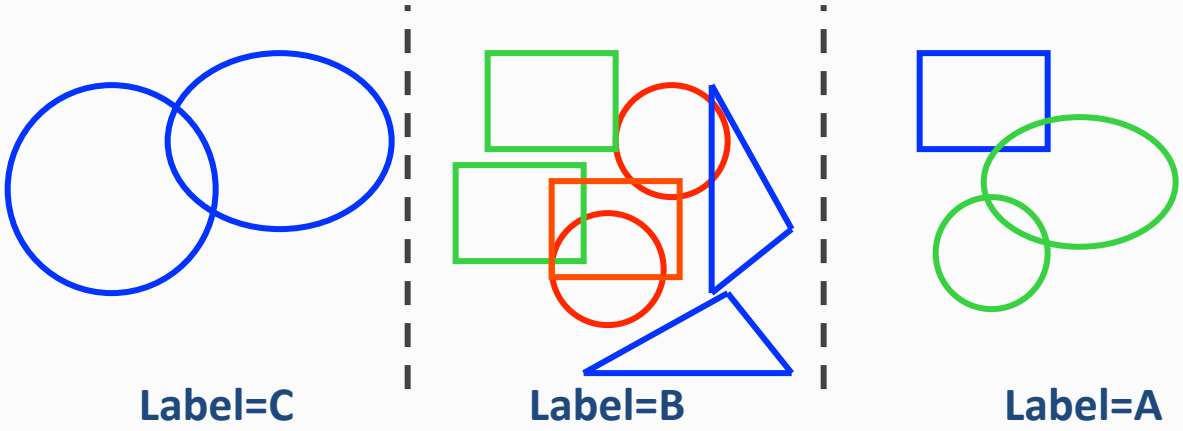
Can be used as hypothesis class for non-parametric classification or regression

General idea: Given a collection of examples, learn a decision tree that represents it

What are decision trees?

- Decision trees are a family of classifiers for instances that are represented by feature vectors (i.e vectors of attributes)
- Nodes are tests for feature values
- There is one branch for every value that the feature can take
- Leaves of the tree specify the class labels

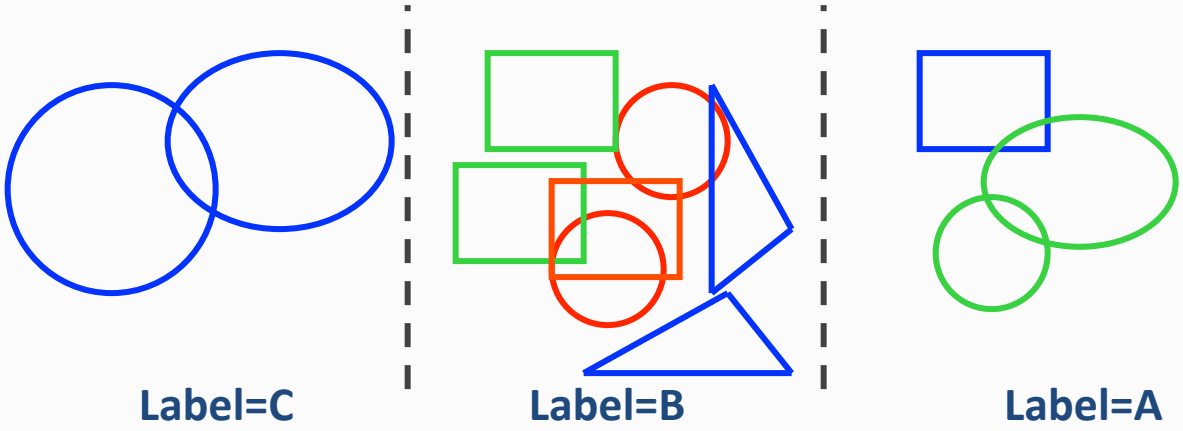
Let’s classify shapes



Before building a decision tree:

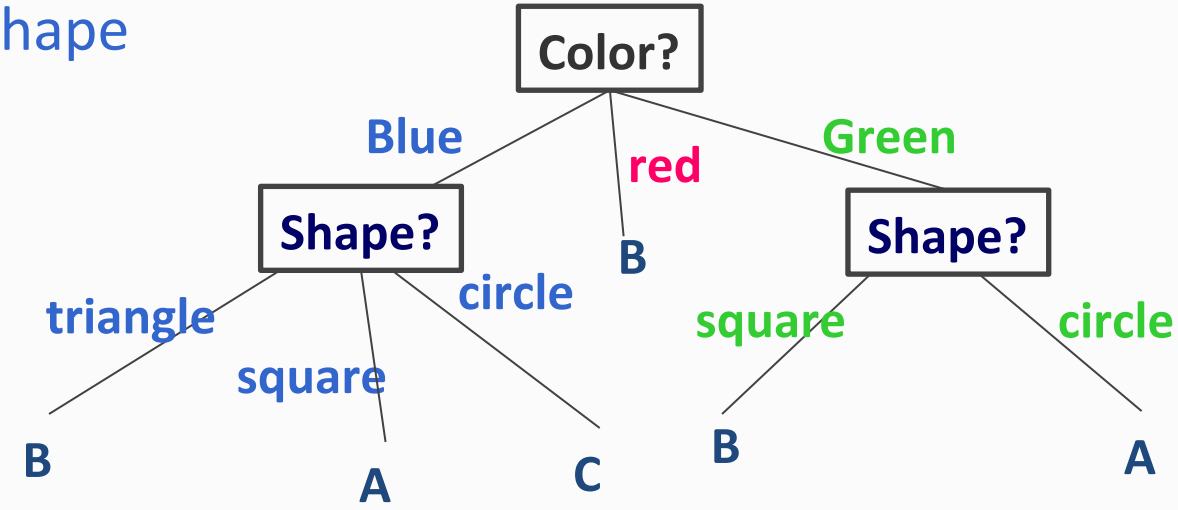
What is the label for a red triangle? And why?

Let's build a decision tree for classifying shapes



What are some attributes of the examples?

Color, Shape

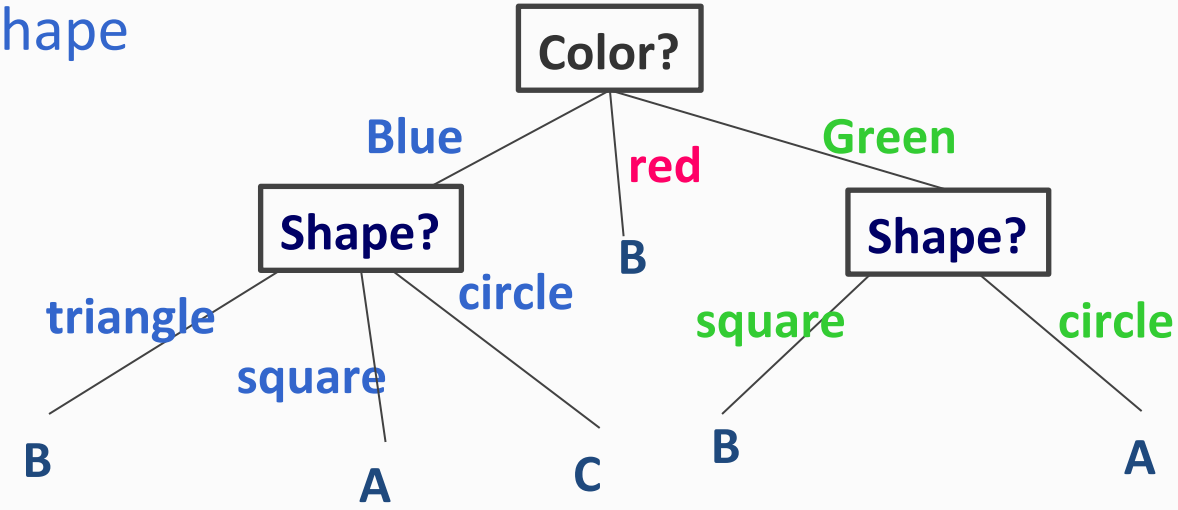


Let's build a decision tree for classifying shapes

1. How to use a decision tree for *prediction*?
   - What is the label for a red triangle?
     - Just follow a path from the root to a leaf
   - What about a green triangle?
2. How do we *learn* a decision tree?
   Coming up soon…

- What ar[illegible]

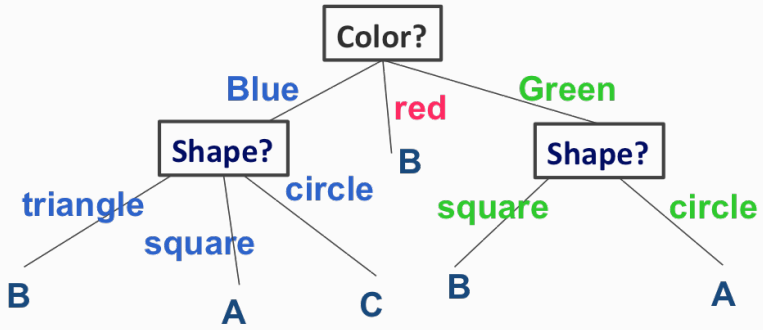
Color, Shape



Expressivity of Decision trees

What Boolean functions can decision trees represent?

- *Any* Boolean function



Every path from the tree to a root is a rule

The full tree is equivalent to the conjunction of all the rules

(Color=blue AND Shape=triangle $\Rightarrow$ Label=B) AND

(Color=blue AND Shape=square $\Rightarrow$ Label=A) AND

(Color=blue AND Shape=circle $\Rightarrow$ Label=C) AND….

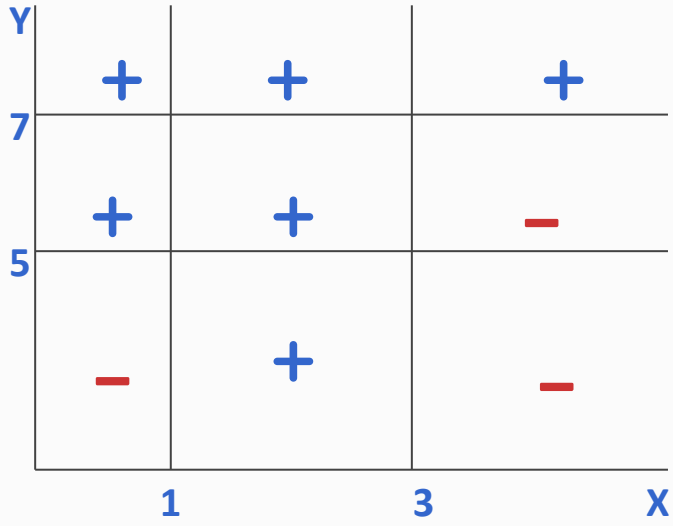
Any Boolean function can be represented as a decision tree.

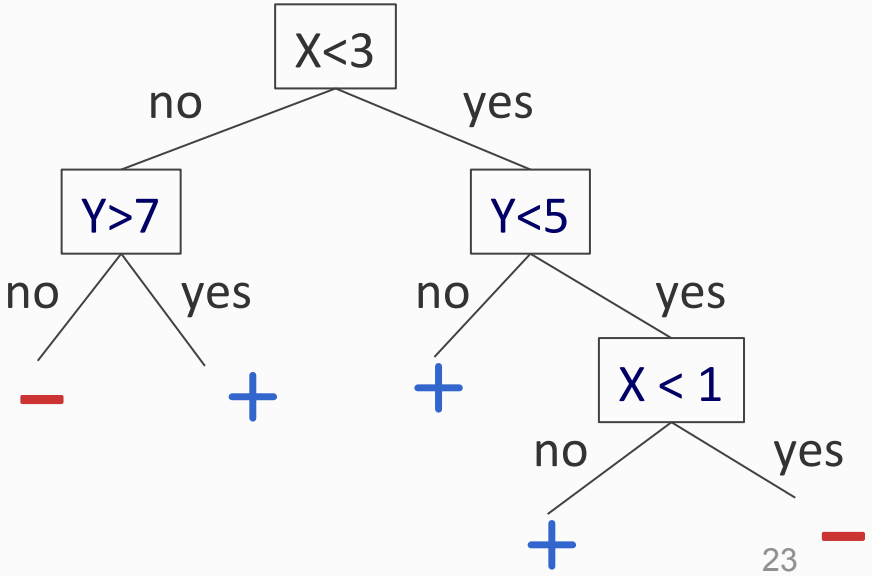
Decision Trees

- Outputs are discrete categories
- But real valued outputs are also possible (regression trees)
- Methods for handling noisy data (noise in the label or in the features) and for handling missing attributes
  - Pruning trees helps with noise
  - More on this later…

Numeric attributes and decision boundaries

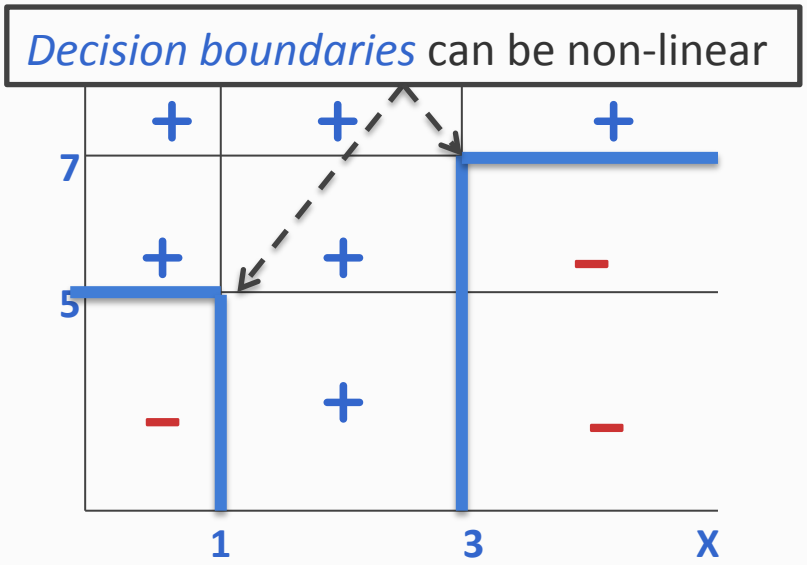
- We have seen instances represented as attribute-value pairs (color=blue, second letter=e, etc.)
  - Values have been categorical
- How do we deal with numeric feature values? (eg length = ?)
  - Discretize them or use thresholds on the numeric values
  - This example divides the feature space into axis parallel rectangles

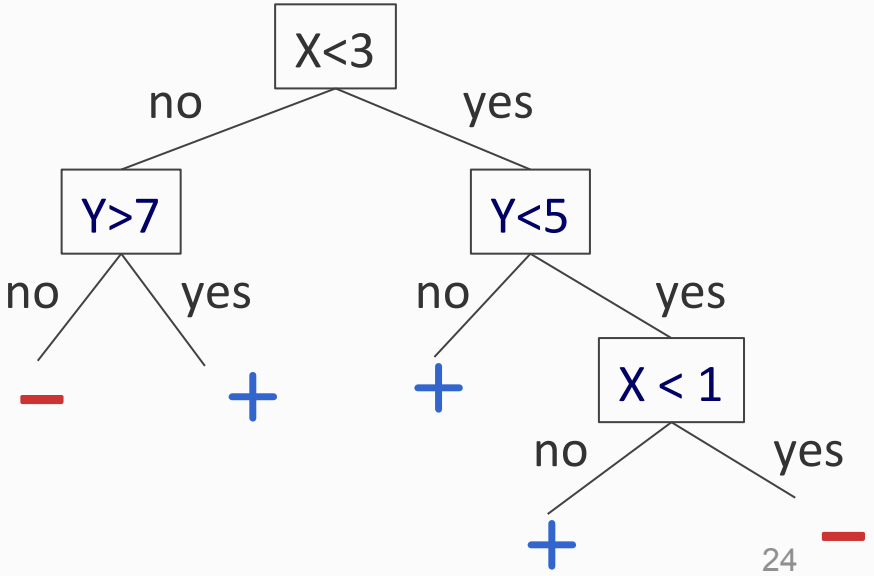




Numeric attributes and decision boundaries

- We have seen instances represented as attribute-value pairs (color=blue, second letter=e, etc.)
  - Values have been categorical
- How do we deal with numeric feature values? (eg length = ?)
  - Discretize them or use thresholds on the numeric values
  - This example divides the feature space into axis parallel rectangles





Summary: Decision trees

- Decision trees can represent any Boolean function
- A way to represent lot of data
- A natural representation (think 20 questions)
- Predicting with a decision tree is easy

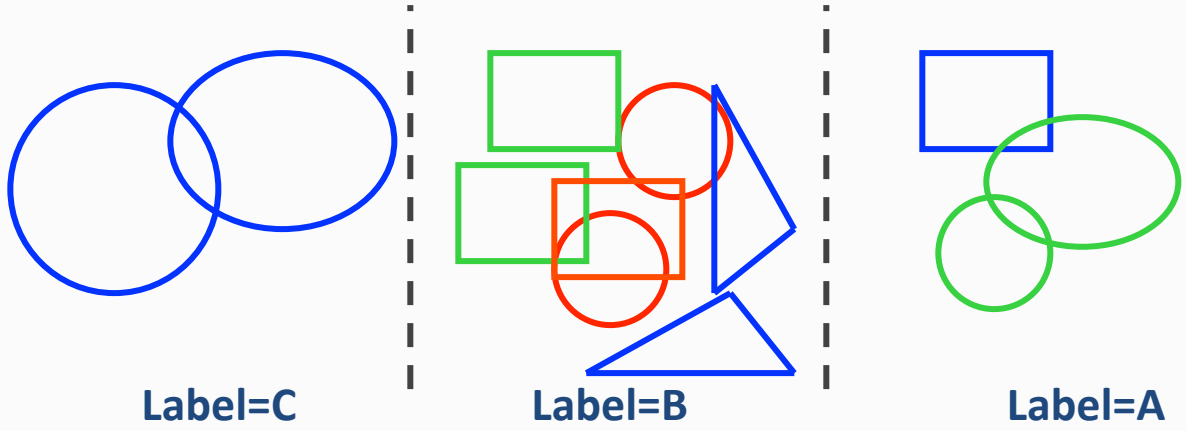
- Clearly, given a dataset, there are many decision trees that can represent it. Why?

- Learning a good representation from data is the next question

Exercise

Write down the decision tree for the shapes data if the root node was *Shape* instead of *Color*

Will the two trees make the same predictions for unseen shapes/color combinations?



Learning Decision Trees

History of Decision Tree Research

- Hunt and colleagues in Psychology: full search decision tree methods to model human concept learning in the 60s
- Quinlan developed ID3, with the information gain heuristics in the late 70s to learn expert systems from examples
- 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 algorithm, C4.5 (1993) is commonly used (New: C5)
- Boosting (or Bagging) over DTs 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}

- **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)

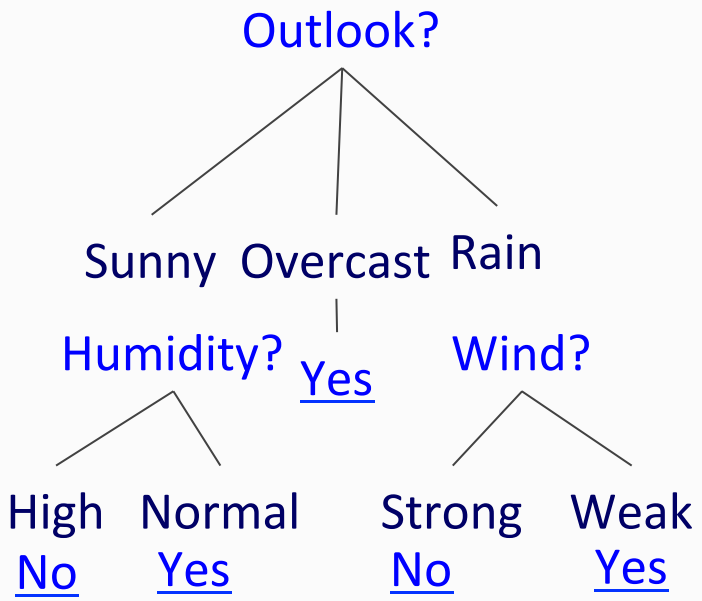
Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

Basic Decision Tree Learning Algorithm

- Data is processed in Batch (i.e. all 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

Input:
S the set of Examples
Label is the target attribute (the prediction)
Attributes is the set of measured attributes

ID3(S, Attributes, Label):

1. If all examples are have same label:
   Return a single node tree with the label
2. Otherwise
   1. Create a `Root node` for tree
   2. A = attribute in Attributes that ***best*** classifies S
   3. for each possible value v of that A can take:
      1. Add a new tree branch corresponding to A=v
      2. Let *Sv* be the subset of examples in S with A=v
      3. if *Sv* is empty:
         add leaf node with the common value of Label in S
         Else:
         below this branch add the subtree ID3(*Sv*, Attributes - {A}, Label)
   4. Return `Root node`

why?

For generalization at test time

Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (*Occam's Razor*)
  - But, finding the minimal decision tree consistent with the data is NP-hard

- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality

- The main decision in the algorithm is the selection of the next attribute to split on

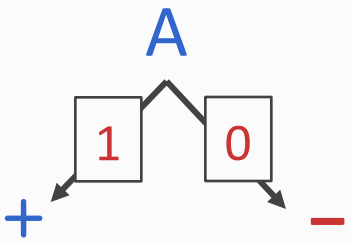
Picking the Root Attribute

Consider data with two Boolean attributes (A,B).

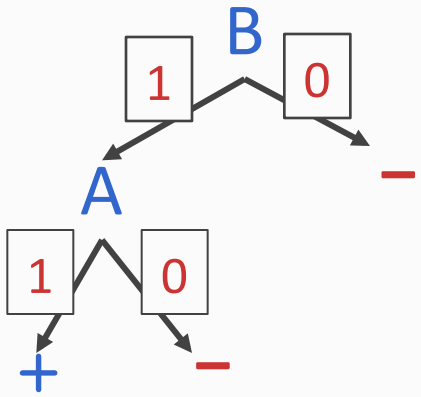
< (A=0,B=0), – >: 50 examples
< (A=0,B=1), – >: 50 examples
< (A=1,B=0), – >: 0 examples
< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes.



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



Picking the Root Attribute

Consider data with two Boolean attributes (A,B).

< (A=0,B=0), – >: 50 examples

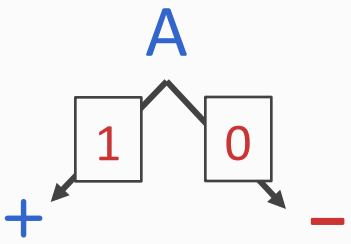
< (A=0,B=1), – >: 50 examples

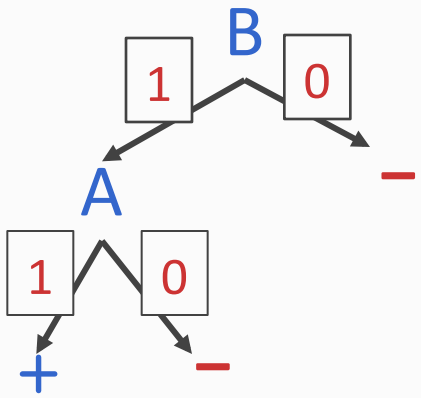
< (A=1,B=0), – >: 0 examples

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

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes.





Splitting on B: 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).

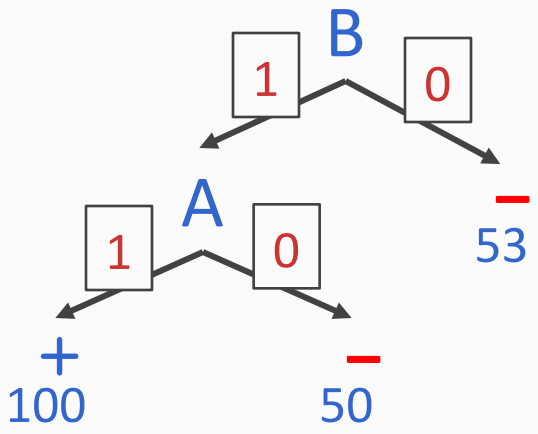
< (A=0,B=0), – >: 50 examples

< (A=0,B=1), – >: 50 examples

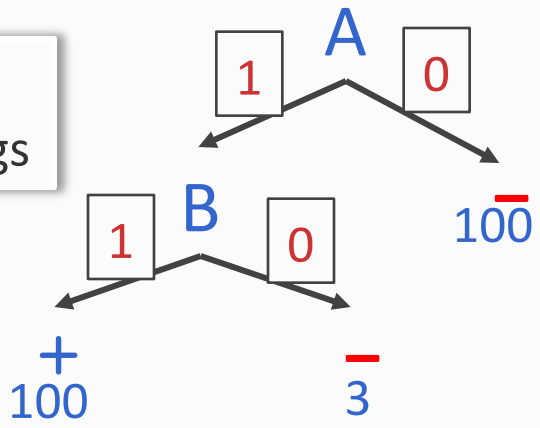
< (A=1,B=0), – >: ~~0 examples~~ 3 examples

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

Which attribute should we choose? Trees looks structurally similar!



Advantage A. But…
Need a way to quantify things



Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
- The main decision in the algorithm is the selection of the next attribute for splitting the data
- 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.

Detour: Entropy

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

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

- The proportion of positive examples is p_+
- The proportion of negative examples is p_-

In general, for a discrete probability distribution with K possible values, with probabilities $\{p_1, p_2, \cdots, p_K\}$ the entropy is given by

$$H(\{p_1, p_2, \cdots, p_K\}) = -\sum_{i=1}^{K} p_i \log(p_i)$$

Detour: Entropy

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

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

- The proportion of positive examples is p_+
- The proportion of negative examples is p_-

- If all examples belong to the same category, entropy = 0
- If $p_+ = p_- = 0.5$, entropy = 1

Entropy can be viewed as the number 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.

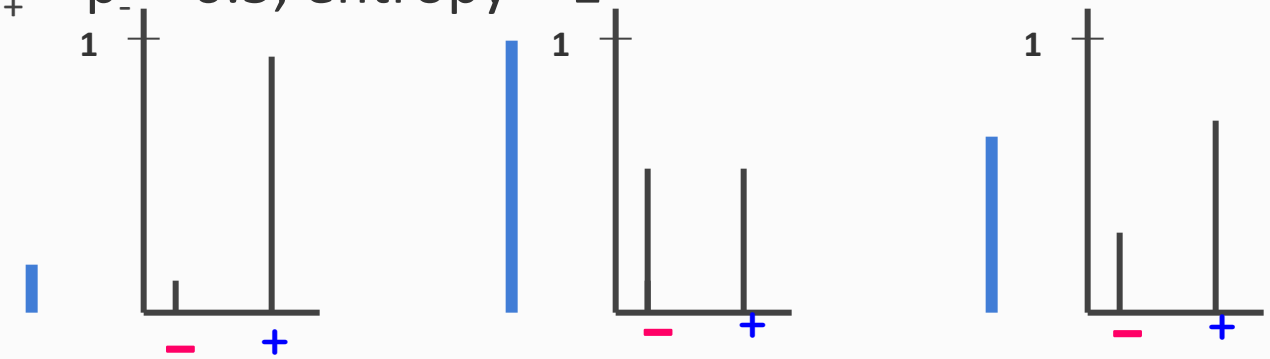
Detour: Entropy

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

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

- The proportion of positive examples is p_+
- The proportion of negative examples is p_-

- If all examples belong to the same category, entropy = 0
- If $p_+ = p_- = 0.5$, entropy = 1



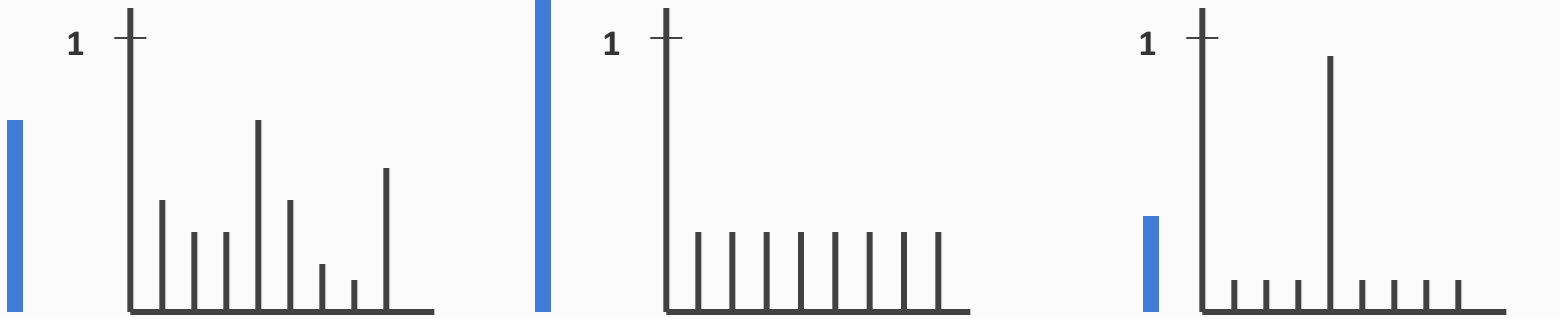
Detour: Entropy

High Entropy – High level of Uncertainty

Low Entropy – No Uncertainty.

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

The uniform distribution has the highest entropy



Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
- The main decision in the algorithm is the selection of the next attribute for splitting the data
- 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.

Information Gain

High Entropy – High level of Uncertainty

Low Entropy – No Uncertainty.

The *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: the subset of examples where the value of 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

Go back to check which of the A, B splits is better

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:

$p = 9/14$

$n = 5/14$

$H(Y) = -(9/14)\log_2(9/14) -(5/14)\log_2(5/14)$

≈ 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* $\mathbf{H_S = 0.971}$

Outlook = overcast: 4 of 14 examples

p = 4/4 *n = 0* $\mathbf{H_o = 0}$

Outlook = rainy: 5 of 14 examples

p = 3/5 *n = 2/5* $\mathbf{H_R = 0.971}$

Expected entropy:

(5/14)×0.971 + (4/14)×0 + (5/14)×0.971

= **0.694**

Information gain:

0.940 – 0.694 **= 0.246**

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:

p = 3/7 *n = 4/7* $\mathbf{H_h = 0.985}$

Humidity = Normal:

p = 6/7 *n = 1/7* $\mathbf{H_o = 0.592}$

Expected entropy:

(7/14)×0.985 + (7/14)×0.592= **0.7885**

Information gain:

0.940 – 0.7885 **= 0.1515**

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

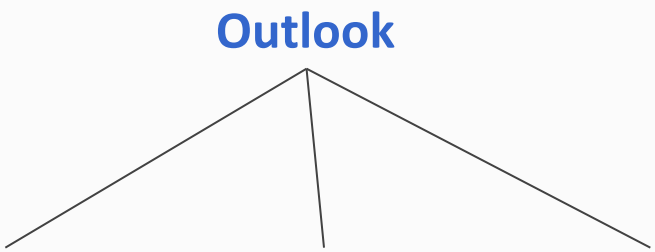
Humidity: 0.151

Wind: 0.048

Temperature: 0.029

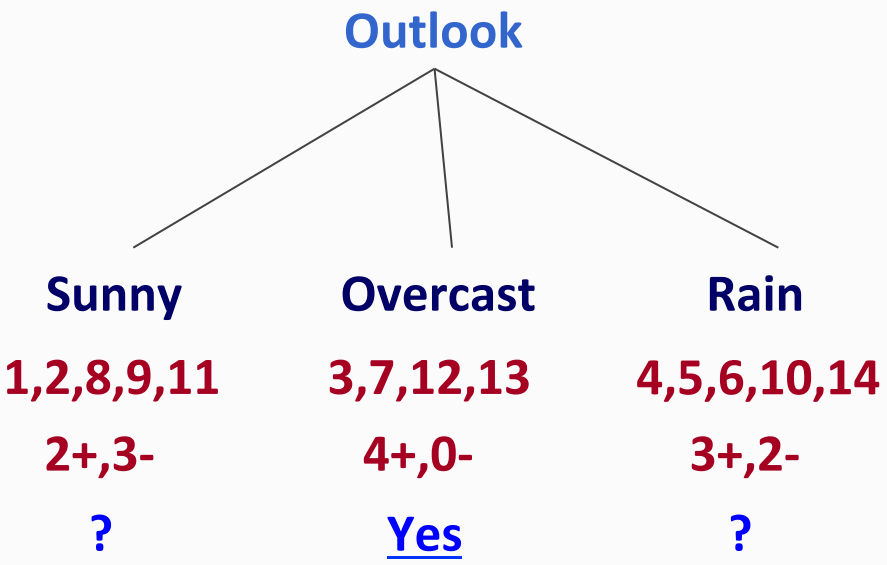
→ Split on Outlook

An Illustrative Example



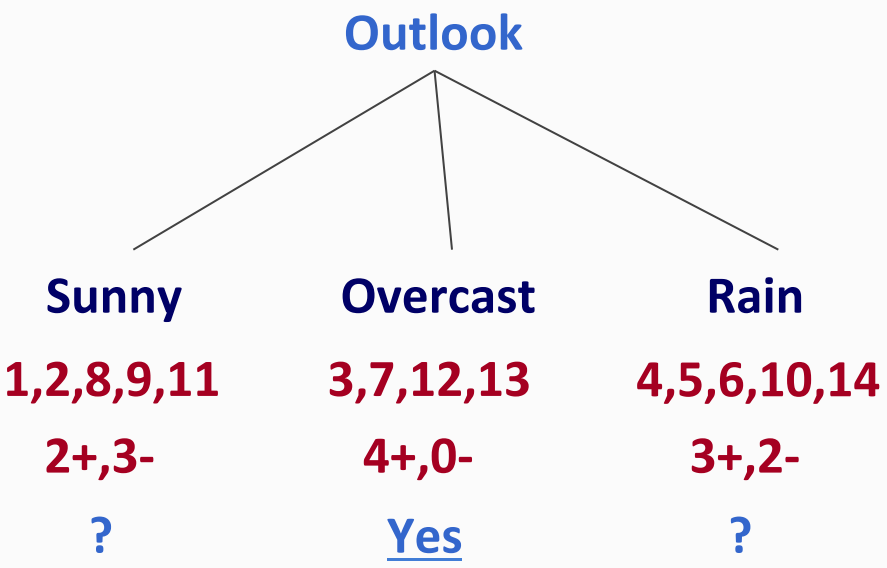
Gain(S,Humidity)=0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

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	-

An Illustrative Example

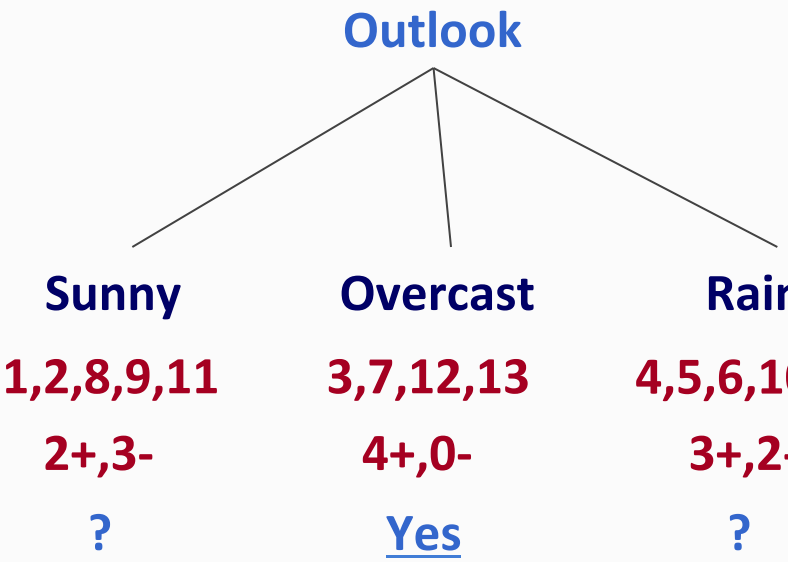


Continue until:

- 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



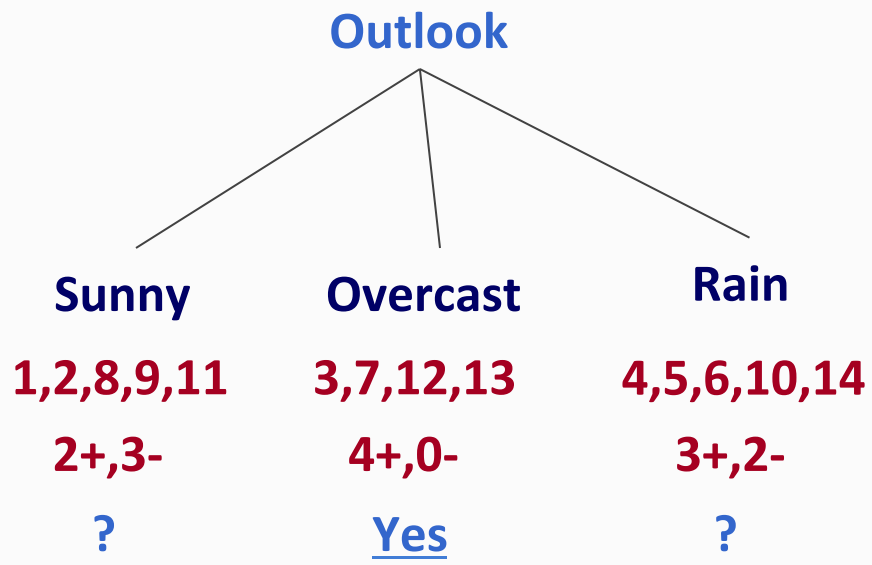
Gain(S_{sunny}, Humidity) = .97-(3/5) 0-(2/5) 0 = .97

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

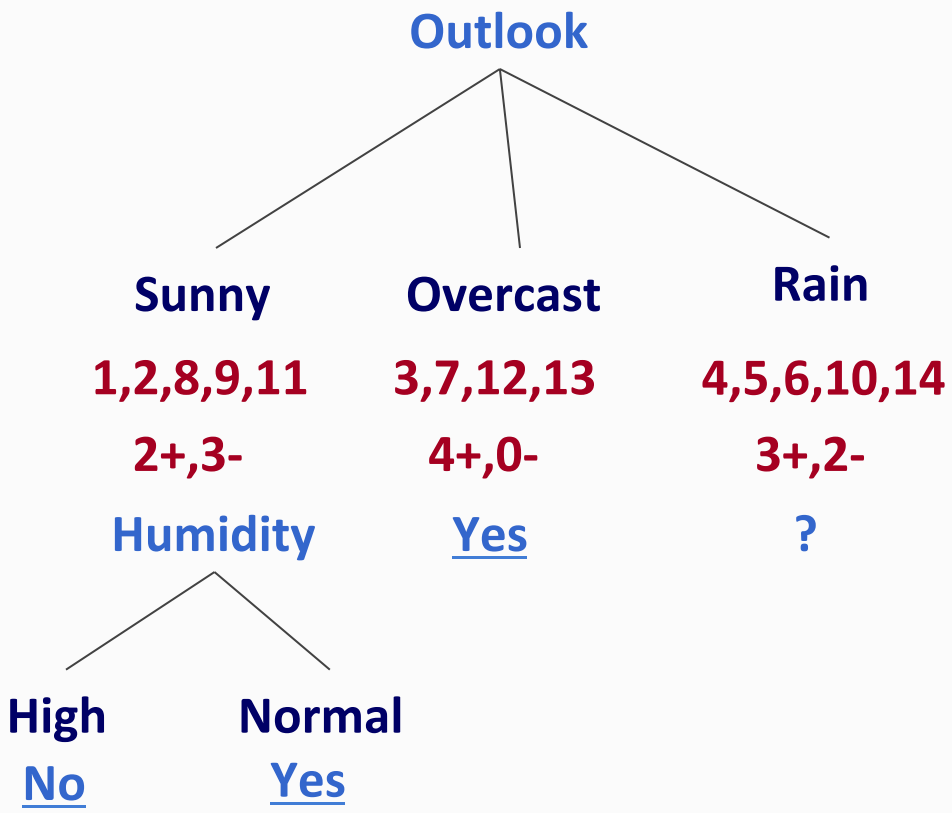
Gain(S_{sunny}, wind) = .97-(2/5) 1 - (3/5) .92= .02

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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



induceDecisionTree(S)

1. Does S uniquely define a class?

   if all $s \in S$ have the same label y: **return** S;

2. Find the feature with the most information gain:

   $i = \text{argmax}_i \; \mathit{Gain}(S, X_i)$

3. Add children to S:

   for *k* **in** Values(X_i):

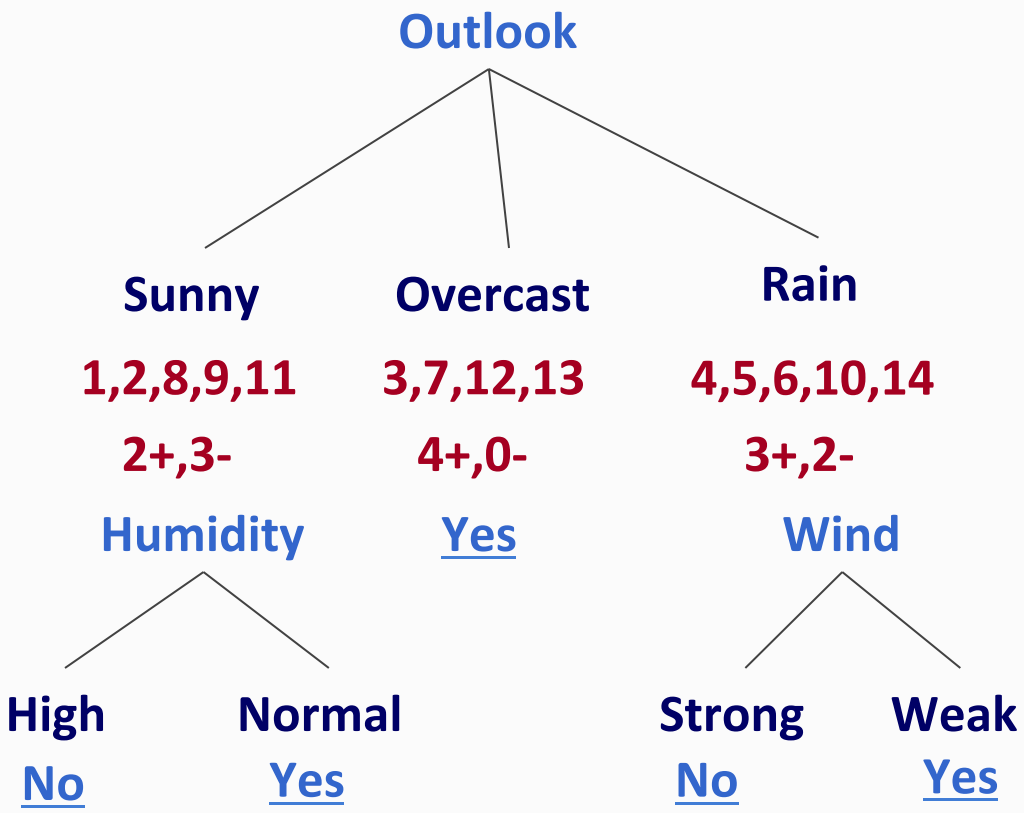
   $S_k = \{s \in S \mid x_i = k\}$

   addChild(S, S_k)

   induceDecisionTree(S_k)

   return S;

An Illustrative Example



Hypothesis Space in Decision Tree Induction

- Conduct a search of the space of decision trees which can represent all possible discrete functions. (pros and cons)
- Goal: to find the best decision tree
- 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

Summary: Learning Decision Trees

1. **Representation**: What are decision trees?
   - A hierarchical data structure that represents data
2. **Algorithm**: Learning decision trees

   The ID3 algorithm: A greedy heuristic
   - If all the examples have the same label, create a leaf with that label
   - Otherwise, find the "most informative" attribute and split the data for different values of that attributes
   - Recurse on the splits

Last lecture: Learning Decision Trees

1. **Representation**: What are decision trees?
   - A hierarchical data structure that represents data
2. **Algorithm**: Learning decision trees

   The ID3 algorithm: A greedy heuristic
   - If all the examples have the same label, create a leaf with that label
   - Otherwise, find the "most informative" attribute and split the data for different values of that attributes
   - Recurse on the splits

Announcements

- Quiz 0 due today
- Homework 1 will be released tonight
- Heads up
  - Project team information due by Sep 15
  - Email me or the TAs

Next: Tips and Tricks

1. Decision tree variants
2. Handling examples with missing feature values
3. Non-Boolean features
4. Avoiding ***overfitting***

1. Variants of information gain

Information gain is defined using entropy to measure the disorder/ impurity of the labels.

Other ways to measure disorder

Example: The *MajorityError*, which computes:

> "*Suppose the tree was not grown below this node and the majority label were chosen, what would be the error?*"

> Suppose at some node, there are 15 + and 5 − examples. What is the *MajorityError*?
>
> Answer: ¼

Works like entropy

2. Missing feature values

Suppose an example is missing the value of an attribute. What can we do at training time?

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

2. Missing feature values

Suppose an example is missing the value of an attribute. What can we do at training time?

"Complete the example" by

- Using the most common value of the attribute in the data
- Using the most common value of the attribute among all examples with the same output
- Using fractional counts of the attribute values
  - Eg: Outlook={5/14 Sunny, 4/14 Overcast, 5/15 Rain}
  - **Exercise**: Will this change probability computations?

2. Missing feature values

Suppose an example is missing the value of an attribute. What can we do at training time?

"Complete the example" by

- Using the most common value of the attribute in the data
- Using the most common value of the attribute among all examples with the same output
- Using fractional counts of the attribute values
  - Eg: Outlook={5/14 Sunny, 4/14 Overcast, 5/15 Rain}
  - **Exercise**: Will this change probability computations?

At test time?

2. Missing feature values

Suppose an example is missing the value of an attribute. What can we do at training time?

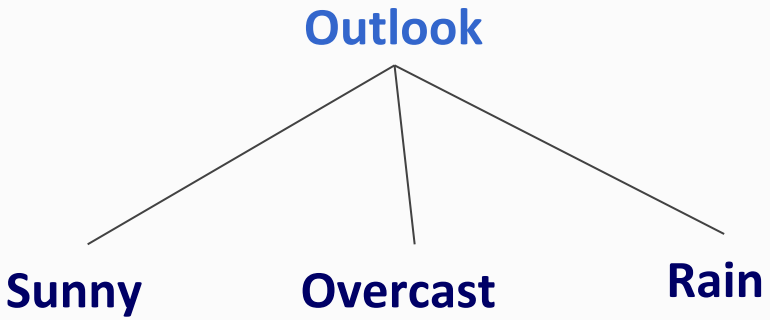
"Complete the example" by

- Using the most common value of the attribute in the data
- Using the most common value of the attribute among all examples with the same output
- Using fractional counts of the attribute values
  - Eg: Outlook={5/14 Sunny, 4/14 Overcast, 5/15 Rain}
  - **Exercise**: Will this change probability computations?

At test time? Use the same method

3. Non-Boolean features

- If the features can take multiple values
  - We have seen one edge per value (i.e a multi-way split)



3. Non-Boolean features

- If the features can take multiple values
  - We have seen one edge per value (i.e a multi-way split)
  - Another option: Make the attributes Boolean by testing for each value

    Convert Outlook=Sunny $\rightarrow$ {Outlook:Sunny=True,
    Outlook:Overcast=False,
    Outlook:Rain=False}
  - Or, perhaps group values into disjoint sets

3. Non-Boolean features

- If the features can take multiple values
  - We have seen one edge per value (i.e a multi-way split)
  - Another option: Make the attributes Boolean by testing for each value

    Convert Outlook=Sunny $\rightarrow$ [Outlook:Sunny=True,
    Outlook:Overcast=False,
    Outlook:Rain=False]
  - Or, perhaps group values into disjoint sets
- For numeric features, use thresholds or ranges to get Boolean/discrete alternatives

4. *Overfitting*

The “First Bit” function

- A Boolean function with n inputs
- Simply returns the value of the first input, all others irrelevant

$Y = X_0$

X_1 is irrelvant

X_0	X_1	Y
F	F	F
F	T	F
T	F	T
T	T	T

What is the decision tree for this function?

The "First Bit" function

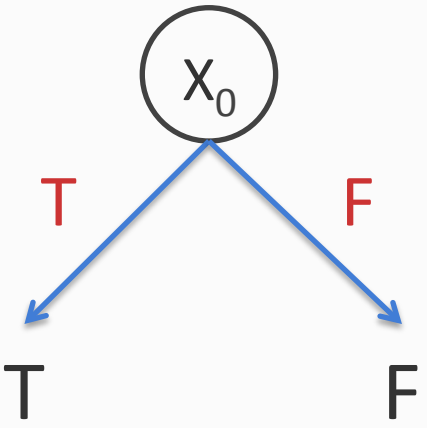
- A Boolean function with n inputs
- Simply returns the value of the first input, all others irrelevant

$Y = X_0$

X_1 is irrelvant

X_0	X_1	Y
F	F	F
F	T	F
T	F	T
T	T	T

What is the decision tree for this function?



Exercise: Convince yourself that ID3 will generate this tree

The best case scenario: Perfect data

Suppose we have all 2^n examples for training. What will the error be on any future examples?

Zero! Because we have seen every possible input!

And the decision tree can represent the function and ID3 will build a consistent tree

Noisy data

What if the data is noisy? And we have all 2^n examples.

X_0	X_1	X_2	Y
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	F
T	F	F	T
T	F	T	T
T	T	F	T
T	T	T	T

Suppose, the outputs of both training and test sets are randomly corrupted

Train and test sets are no longer identical.

Both have noise, possibly different

Noisy data

What if the data is noisy? And we have all 2^n examples.

X_0	X_1	X_2	Y
F	F	F	F
F	F	T	~~F~~ T
F	T	F	F
F	T	T	F
T	F	F	T
T	F	T	~~T~~ F
T	T	F	T
T	T	T	T

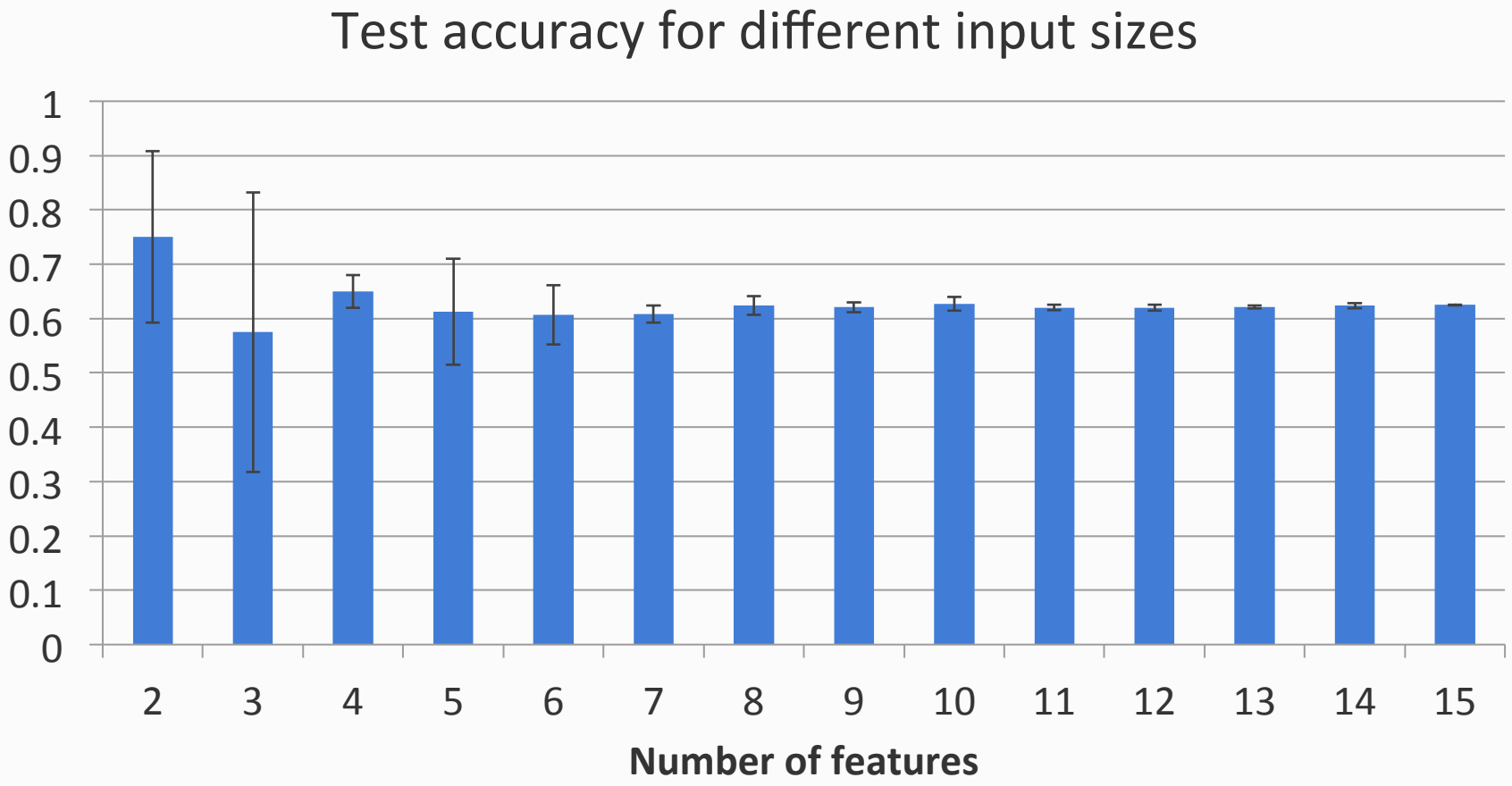
Suppose, the outputs of both training and test sets are randomly corrupted

Train and test sets are no longer identical.

Both have noise, possibly different

E.g: Output corrupted with probability 0.25

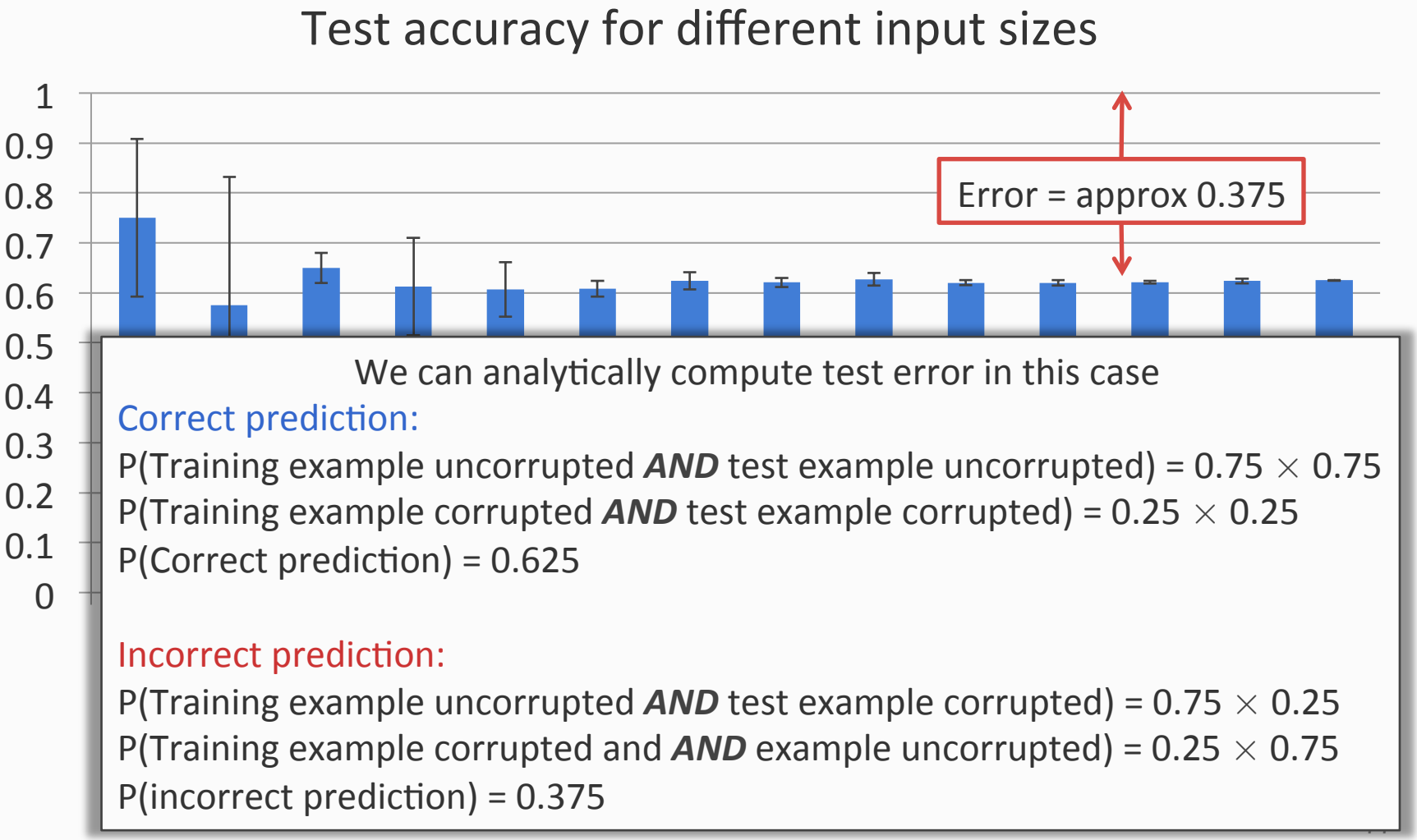
The data is noisy. And we have all 2^n examples.



The error bars are generated by running the same experiment multiple times for the same setting

E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.



We can analytically compute test error in this case

Correct prediction:

P(Training example uncorrupted ***AND*** test example uncorrupted) = 0.75 × 0.75

P(Training example corrupted ***AND*** test example corrupted) = 0.25 × 0.25

P(Correct prediction) = 0.625

Incorrect prediction:

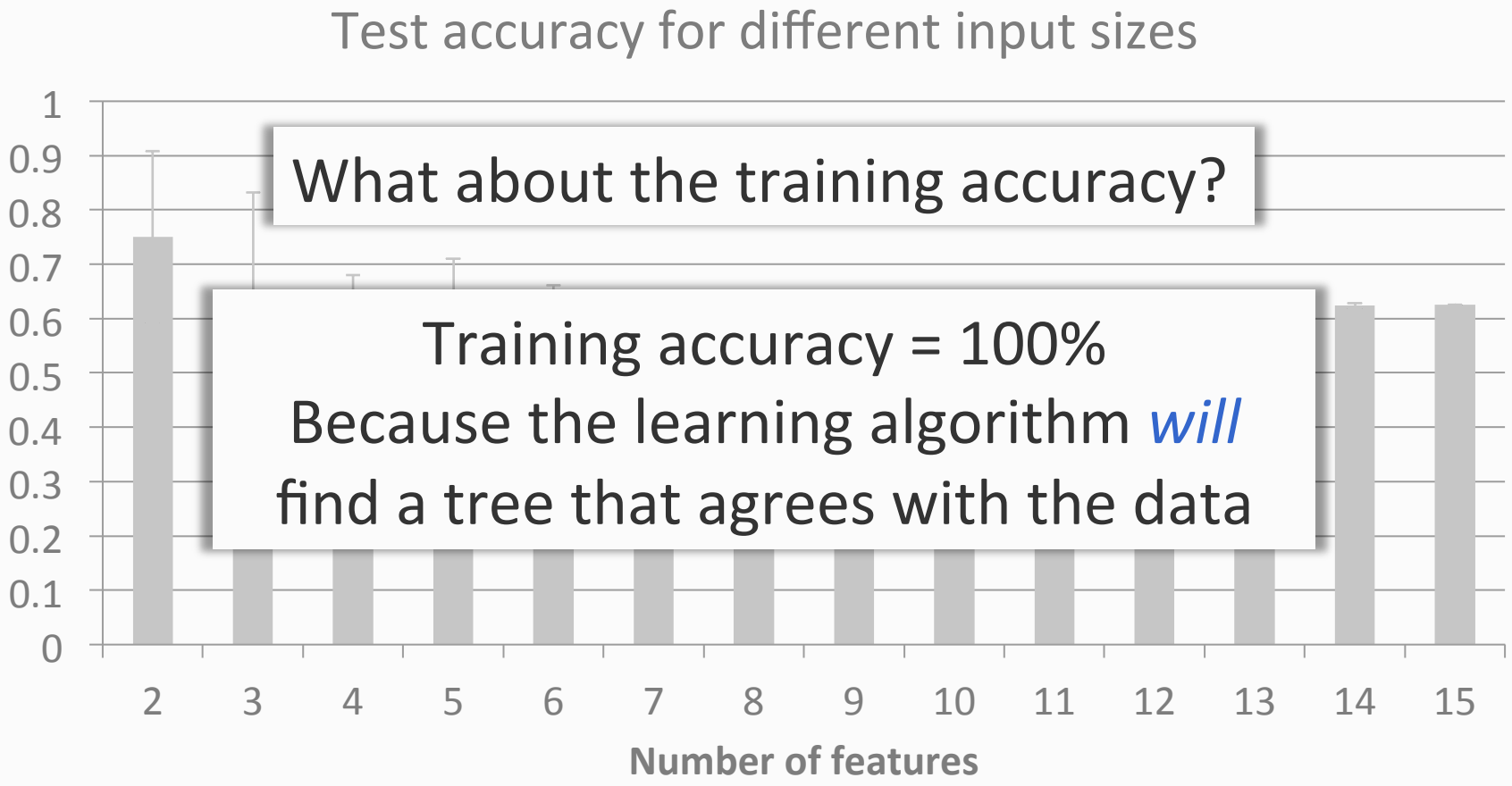
P(Training example uncorrupted ***AND*** test example corrupted) = 0.75 × 0.25

P(Training example corrupted and ***AND*** example uncorrupted) = 0.25 × 0.75

P(incorrect prediction) = 0.375

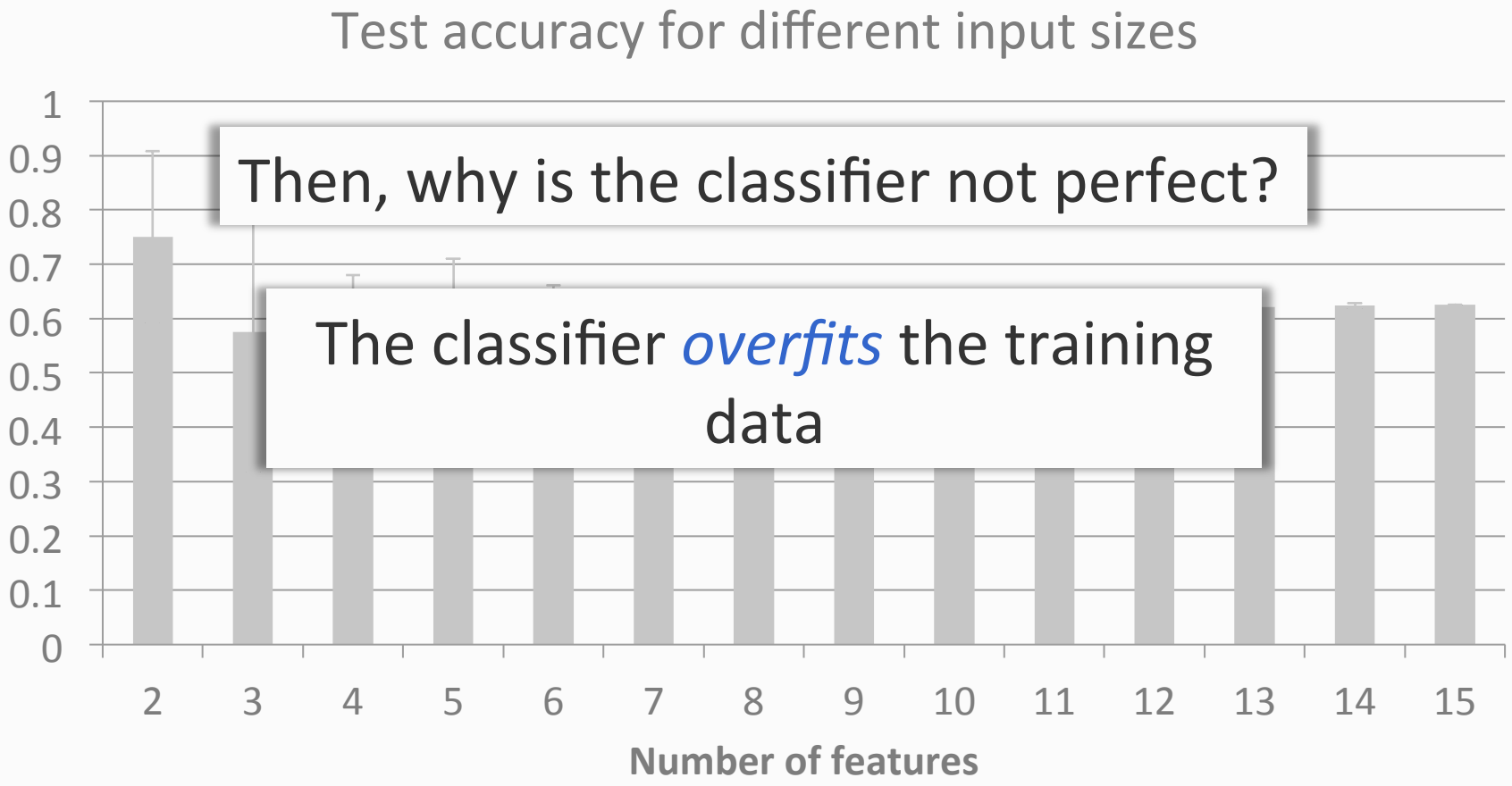
E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.



E.g: Output corrupted with probability 0.25

The data is noisy. And we have all 2^n examples.



Overfitting

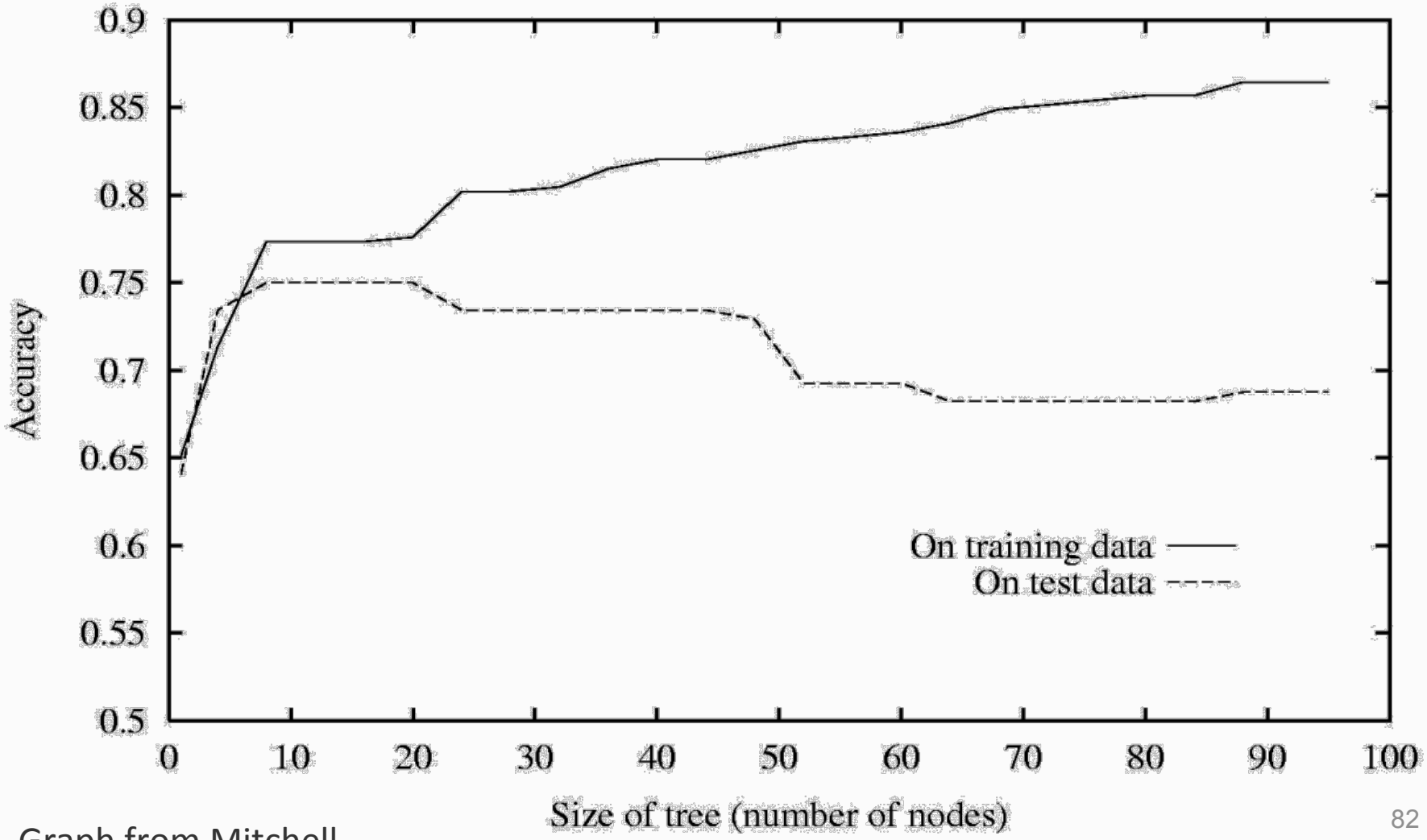
- The learning algorithm fits the noise in the data
  - Irrelevant attributes or noisy examples influence the choice of the hypothesis

- May lead to poor performance on future examples

Overfitting: One definition

- Data comes from a probability distribution D(X, Y)
- We are using a hypothesis space H
- Errors:
  - Training error for hypothesis $h \in H$: $error_{train}(h)$
  - True error for $h \in H$: $error_D(h)$
- A hypothesis h ***overfits*** the training data if there is another hypothesis h' such that
  1. $error_{train}(h) < error_{train}(h')$
  2. $error_D(h) > error_D(h')$

Decision trees *will* overfit



Graph from Mitchell

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

- Some approaches:
  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

- Some approaches:
  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 tree on the training split and check the performance on the held-out set after every new node is added
     - If growing the tree hurts validation set performance, stop growing

Avoiding overfitting with decision trees

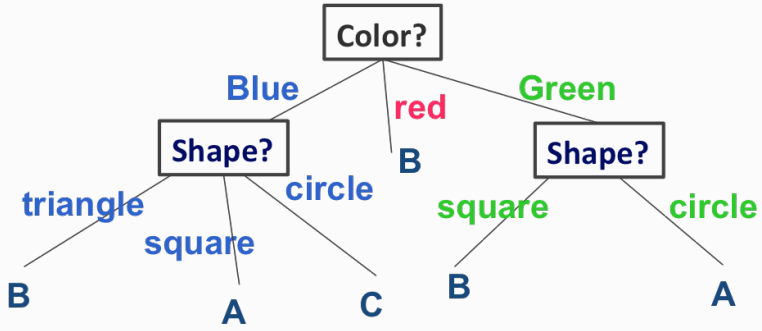
Occam's Razor

Favor simpler (in this case, shorter) hypotheses

Why? Fewer shorter trees, less likely to fit better by coincidence

- Some approaches:
  3. Grow full tree and then prune as a post-processing step in one of several ways:
     1. Use a validation set for pruning from bottom up greedily
     2. Convert the tree into a set of rules (one rule per path from root to leaf) and prune each rule independently

Reminder: Decision trees are rules



- If Color=Blue and Shape=triangle, then output = B
- If Color=Red, then output = B

  …

Summary: Decision trees

- A popular machine learning tool
  - Prediction is easy
  - Represents any Boolean functions
- Greedy heuristics for learning
  - ID3 algorithm (using information gain)
  - Robust implementations of some variants (eg. C4.5 algorithm) exist
- (Can be used for regression too)
- Decision trees are prone to overfitting unless you take care to avoid it