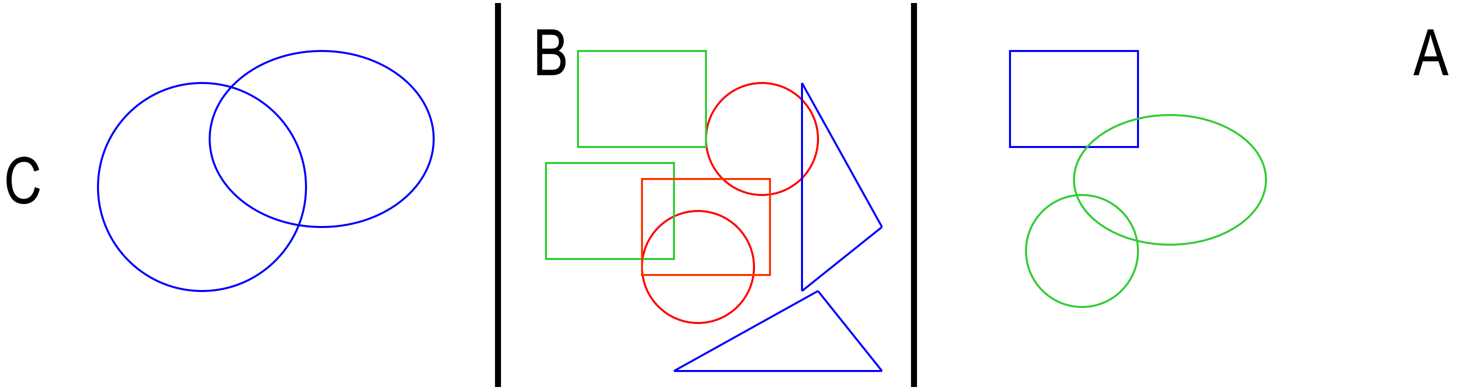
Machine Learning

Lecture 10

Decision Tree Learning

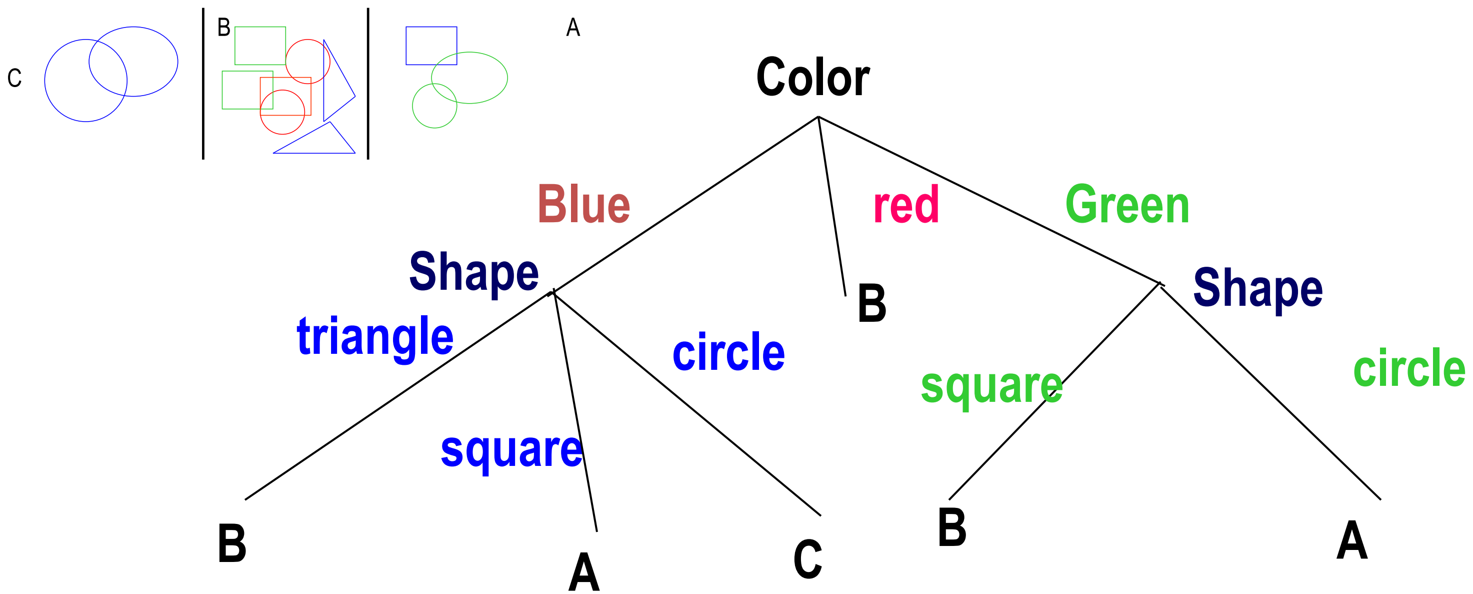
Decision Trees

- A hierarchical data structure that represents data by implementing a divide and conquer strategy
- Can be used as a non-parametric classification and regression method.
- Given a collection of examples, learn a decision tree that represents it.
- Use this representation to classify new examples



Decision Trees: The Representation

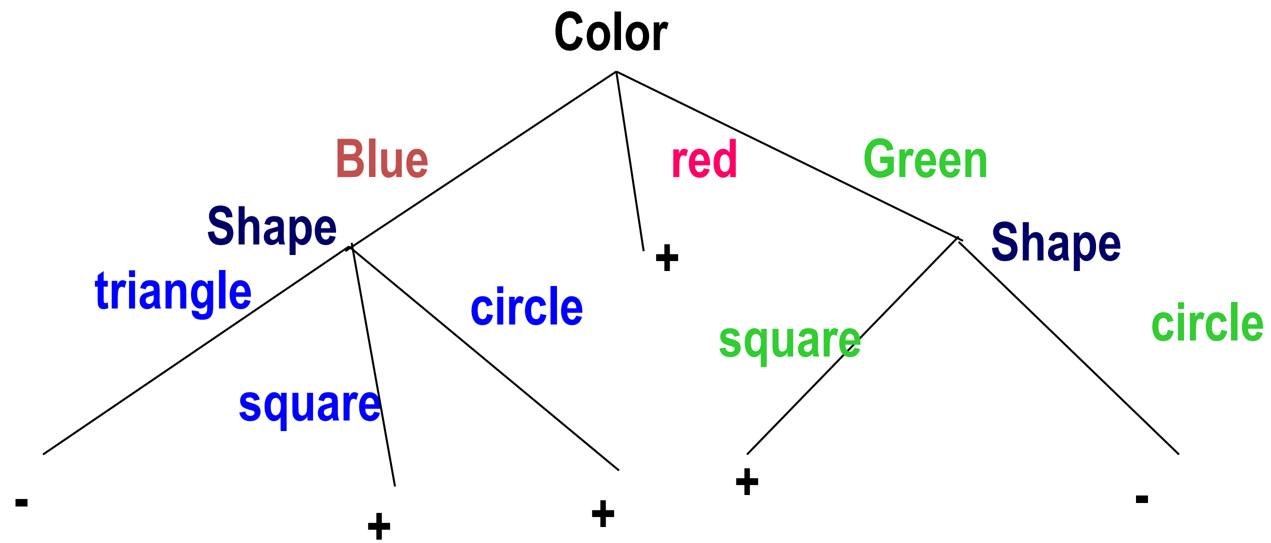
- Decision Trees are classifiers for instances represented as features vectors. (color= ;shape= ;label=)
- Nodes are tests for feature values;
- There is one branch for each value of the feature
- Leaves specify the categories (labels)
- Can categorize instances into multiple disjoint categories – multi-class



Boolean Decision Trees

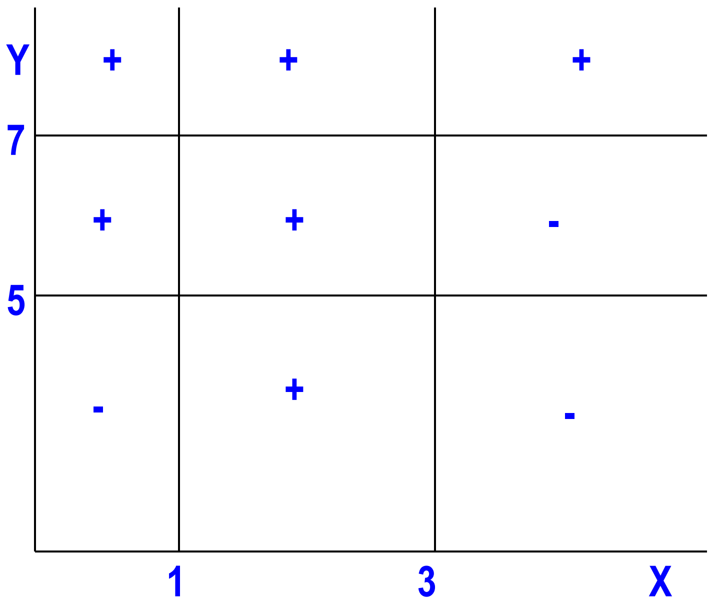
They can represent any Boolean function.

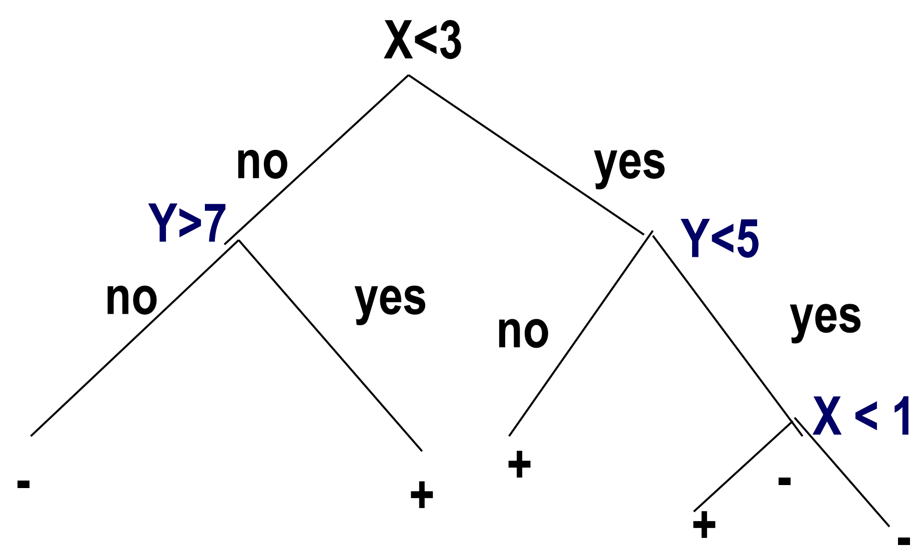
- Can be rewritten as rules in Disjunctive Normal Form (DNF)
- green $\wedge$ square $\rightarrow$ positive
- blue $\wedge$ circle $\rightarrow$ positive
- blue $\wedge$ square $\rightarrow$ positive
- The disjunction of these rules is equivalent to the Decision Tree



Decision Trees: Decision Boundaries

- Usually, instances are represented as attribute-value pairs
  (color=blue, shape=square, +)
- Numerical values can be used either by discretizing or by using thresholds for splitting nodes.
- In this case, the tree divides the feature space into axis-parallel rectangles, each labeled with one of the labels.





Decision Trees

- Can represent any Boolean Function
- Can be viewed as a way to compactly represent a lot of data.
- Advantage: non-metric data
- Natural representation: (20 questions) http://www.20q.net/

- The evaluation of the Decision Tree Classifier is easy
- Clearly, given data, there are many ways to Represent it as a decision tree.
- Learning a good representation from data is the challenge.

Basic Decision Trees Learning Algorithm

- <u>DT(Examples, Attributes)</u>

  If all Examples have same label: return a leaf node with Label

  Else

    If Attributes is empty: return a leaf with majority Label

  Else

    Pick an attribute A as root

    For each value v of A

    Let Examples(v) be all the examples for which A=v

    Add a branch out of the root for the test A=v

      If Examples(v) is empty

        create a leaf node labeled with the majority label in Examples

      Else recursively create subtree by calling

        DT(Examples(v), Attribute-{A})

Picking the Root Attribute

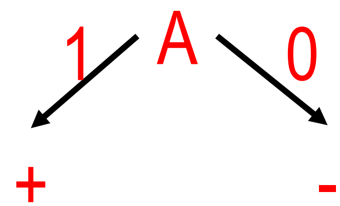
- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
- Finding the minimal decision tree consistent with the data is NP-hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on.

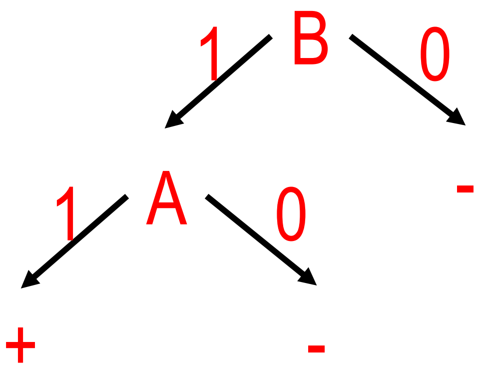
Picking the Root Attribute

- 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.





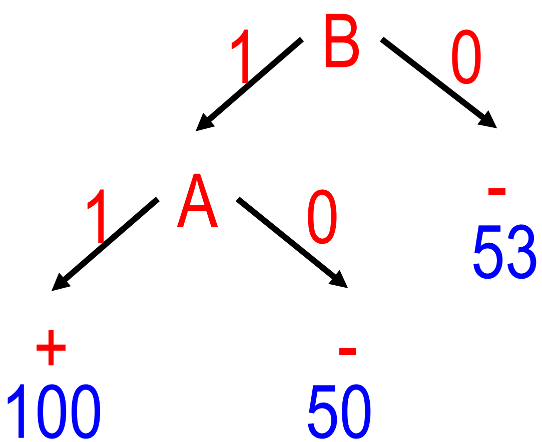
- Splitting on B: we don't get purely labeled nodes.
- What if we have: {(A=1,B=0), - }: 3 examples

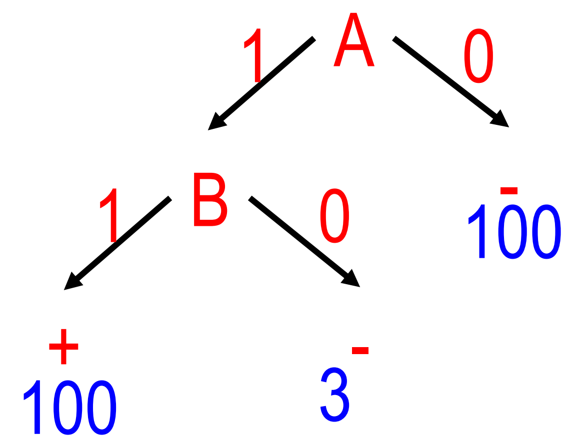
Picking the Root Attribute

- 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

- Trees looks structurally similar; which attribute should we choose?





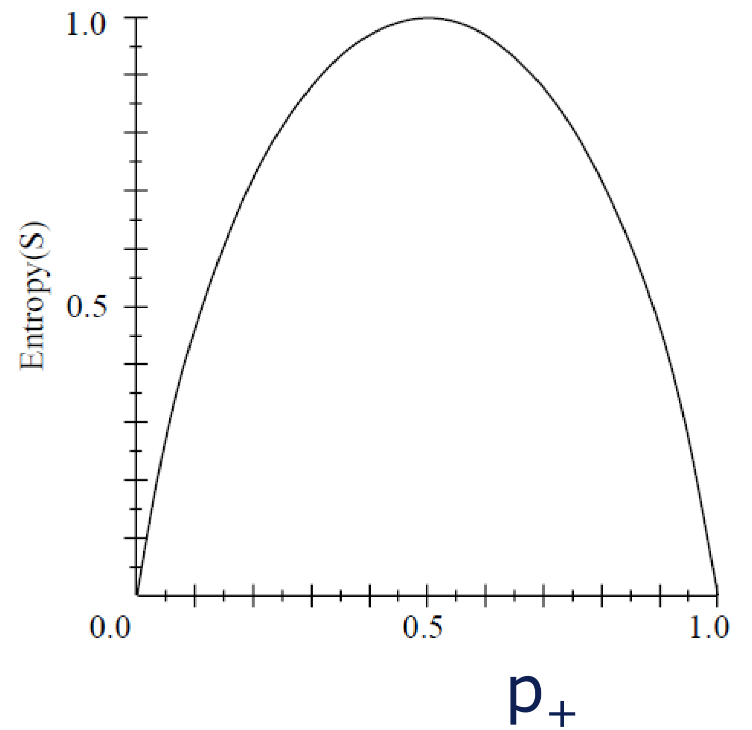
Picking the Root Attribute

- 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 to condition 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 based on information gain, originated with the ID3 system of Quinlan.

Entropy

- S is a sample of training examples
- p_+ is the proportion of positive examples in S
- P_- is the proportion of negative examples in S
- Entropy measures the impurity of S



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

Entropy

- Entropy (s) = expected number of bits needed to encode class (+ or -) of randomly drawn member of S (under the optimal, shortest length code)

Why?

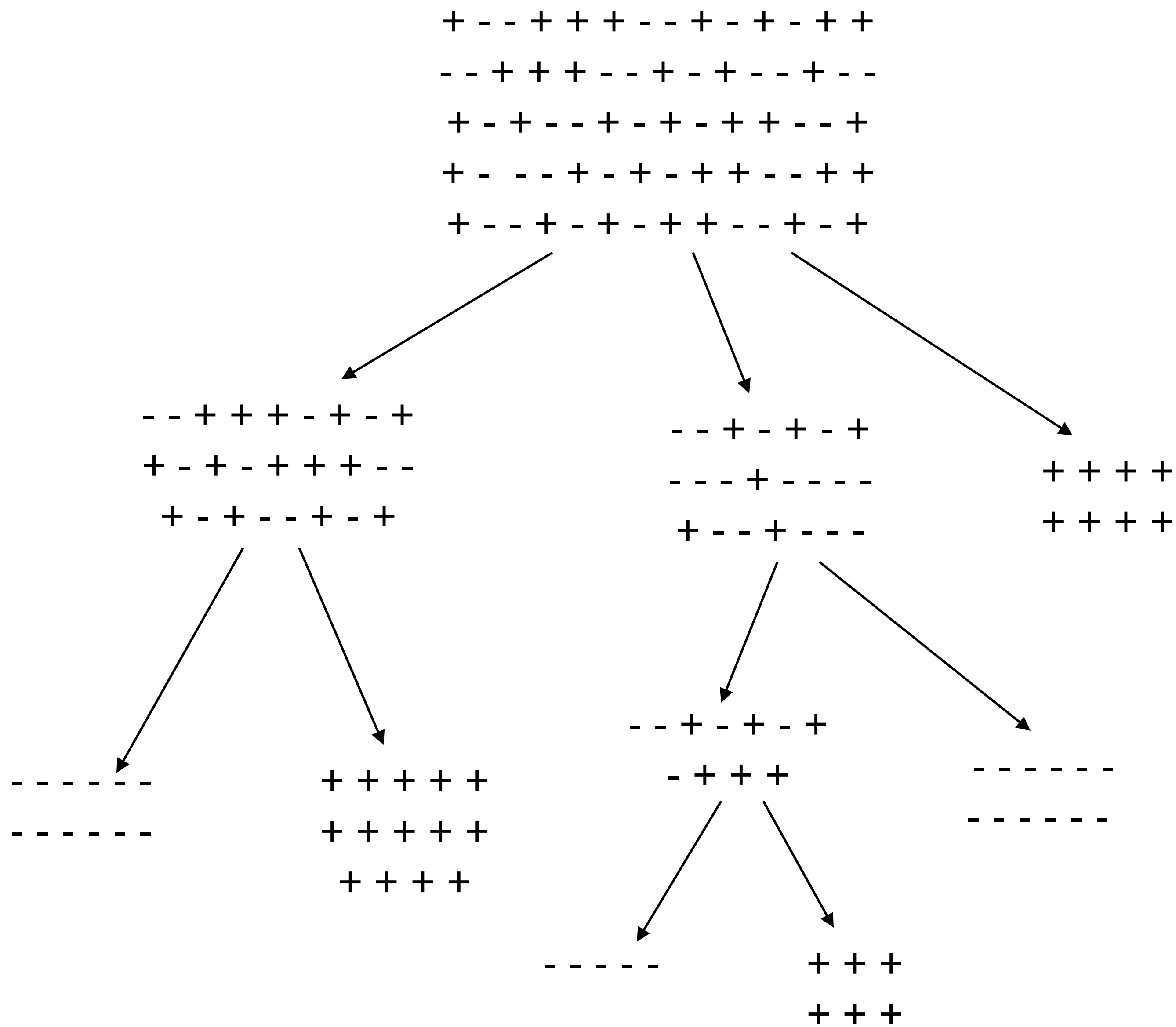
- Information theory: optimal length code assigns $-\log_2 p$ bits to message having probability p
- So, expected number of bits to encode + or - of random member of S:

$$p_+(-\log(p_+)) - p_-(-\log(p_-))$$

$$Entropy(S) \equiv -p_+ \log(p_+) - p_- \log(p_-)$$

Highly Disorganized

High Entropy



Highly Organized

Low Entropy

Information Gain

- Gain (S, A) = expected reduction in entropy due to sorting on A

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

- Values (A) is the set of all possible values for attribute A, Sv is the subset of S which attribute A has value v
- Gain(S,A) is the expected reduction in entropy caused by knowing the values of attribute A.

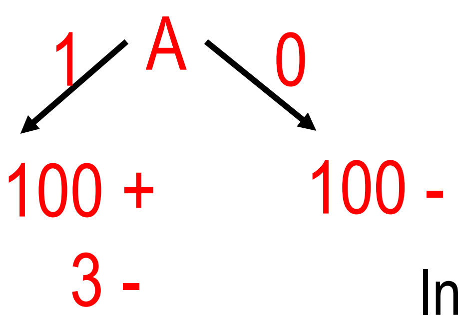
Picking the Root Attribute

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

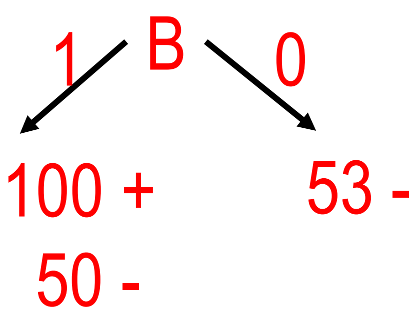
{ (A=0,B=0), - }: 50 examples
{ (A=0,B=1), - }: 50 examples
{ (A=1,B=0), - }: 3 examples
{ (A=1,B=1), + }: 100 examples

- What should be the first attribute we select?

- Splitting on A:



- Splitting on B:



Information gain of A is higher

An Illustrative Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

An Illustrative Example (2)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

$$\text{Entropy}(S) = -\tfrac{9}{14}\log(\tfrac{9}{14}) - \tfrac{5}{14}\log(\tfrac{5}{14}) = 0.94$$

Entropy

9+,5-

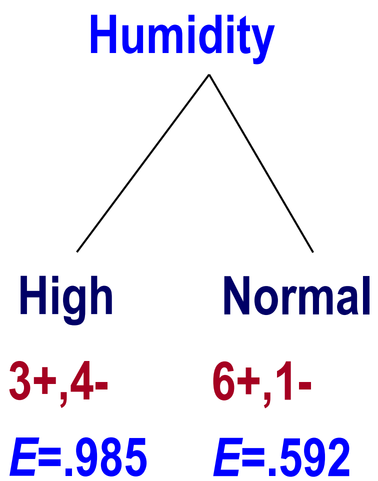
An Illustrative Example (2)

Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-

E=.94

An Illustrative Example (2)



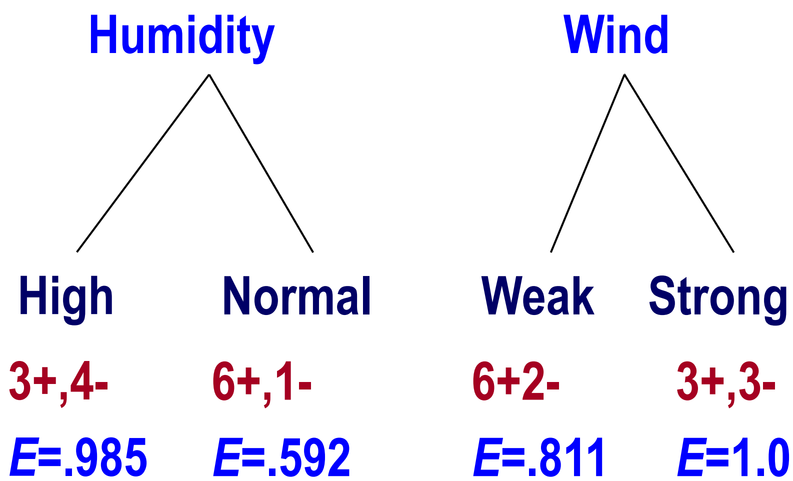
Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-

E=.94

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

An Illustrative Example (2)



Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-

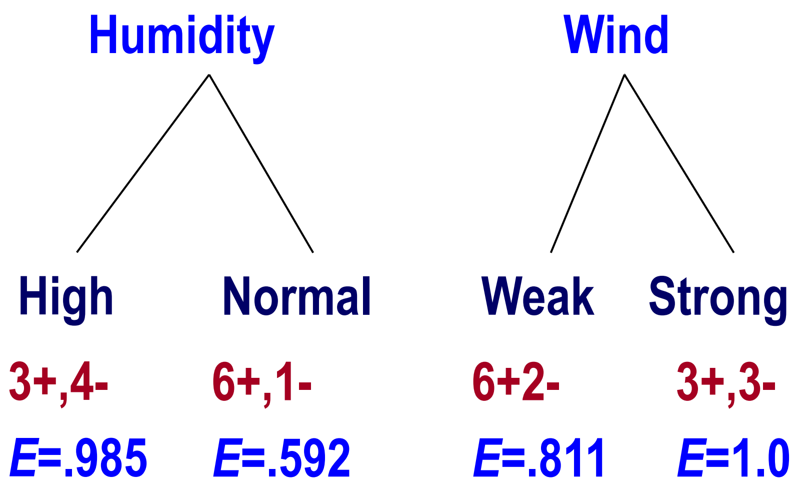
E=.94

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

An Illustrative Example (2)

Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-
E=.94



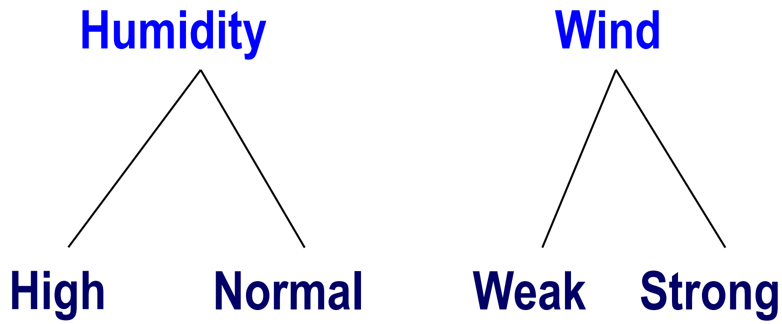
*Gain(S,*Humidity*)*=
.94 - 7/14 0.985
- 7/14 0.592=
0.151

$$\mathrm{Gain}(S,a) = \mathrm{Entropy}(S) - \sum_{v \in \mathrm{values}(a)} \frac{|S_v|}{|S|} \mathrm{Entropy}(S_v)$$

An Illustrative Example (2)

Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-
E=.94



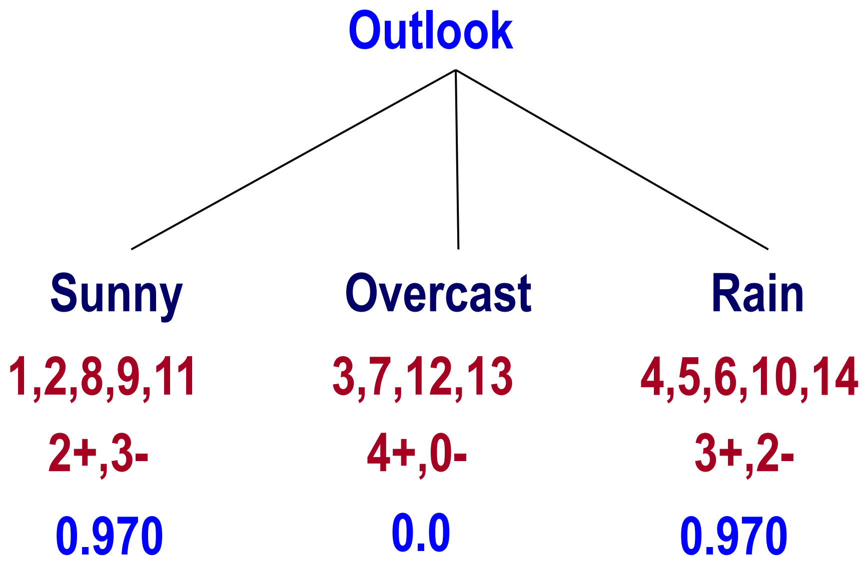
3+,4- *E*=.985 | 6+,1- *E*=.592 | 6+2- *E*=.811 | 3+,3- *E*=1.0

Gain(S,Humidity)=
.94 - 7/14 0.985
- 7/14 0.592=
0.151

Gain(S,Wind)=
.94 - 8/14 0.811
- 6/14 1.0 =
0.048

$$\mathrm{Gain}(S,a) = \mathrm{Entropy}(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} \mathrm{Entropy}(S_v)$$

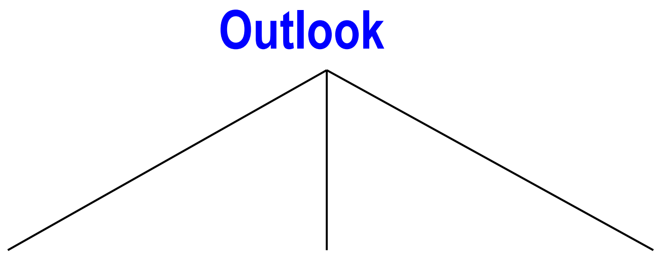
An Illustrative Example (3)



*Gain(S,*Outlook*)*=
0.246

Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

An Illustrative Example (3)



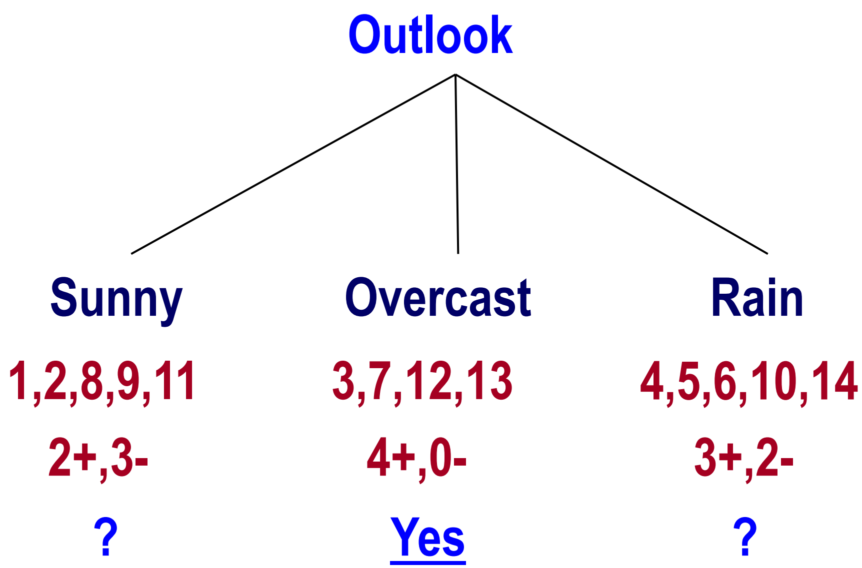
$Gain(S, \text{Humidity}) = 0.151$

$Gain(S, \text{Wind}) = 0.048$

$Gain(S, \text{Temperature}) = 0.029$

$Gain(S, \text{Outlook}) = 0.246$

An Illustrative Example (3)



Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

Continue until:

- Every attribute is included in path, or,
- All examples in the leaf have same label

An Illustrative Example (4)

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

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

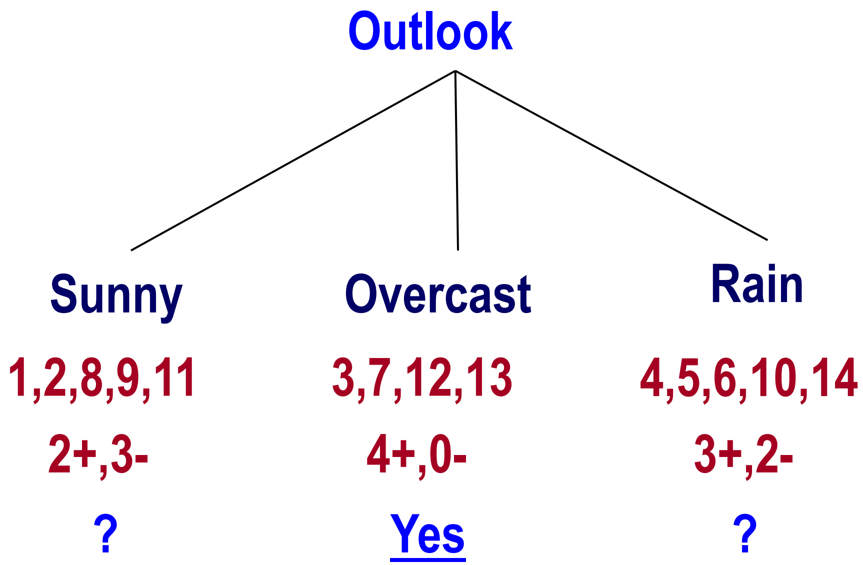
$Gain(S_{sunny}, Wind) = .97 - (2/5)\,1 - (3/5)\,.92 = .02$

Outlook

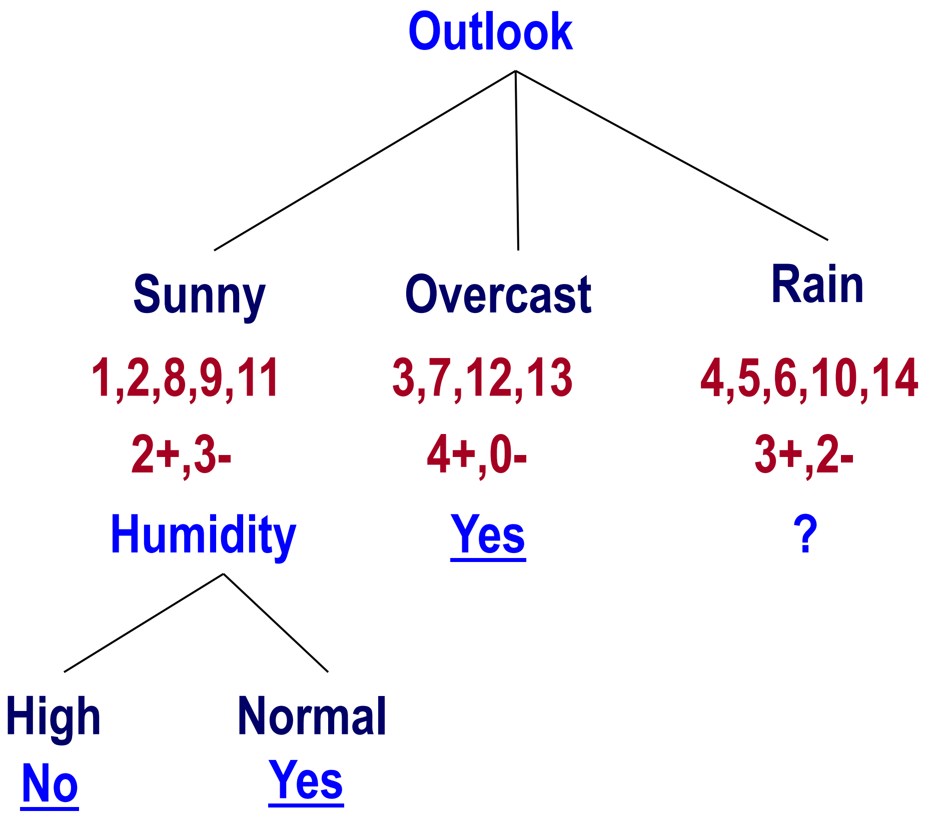
Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

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

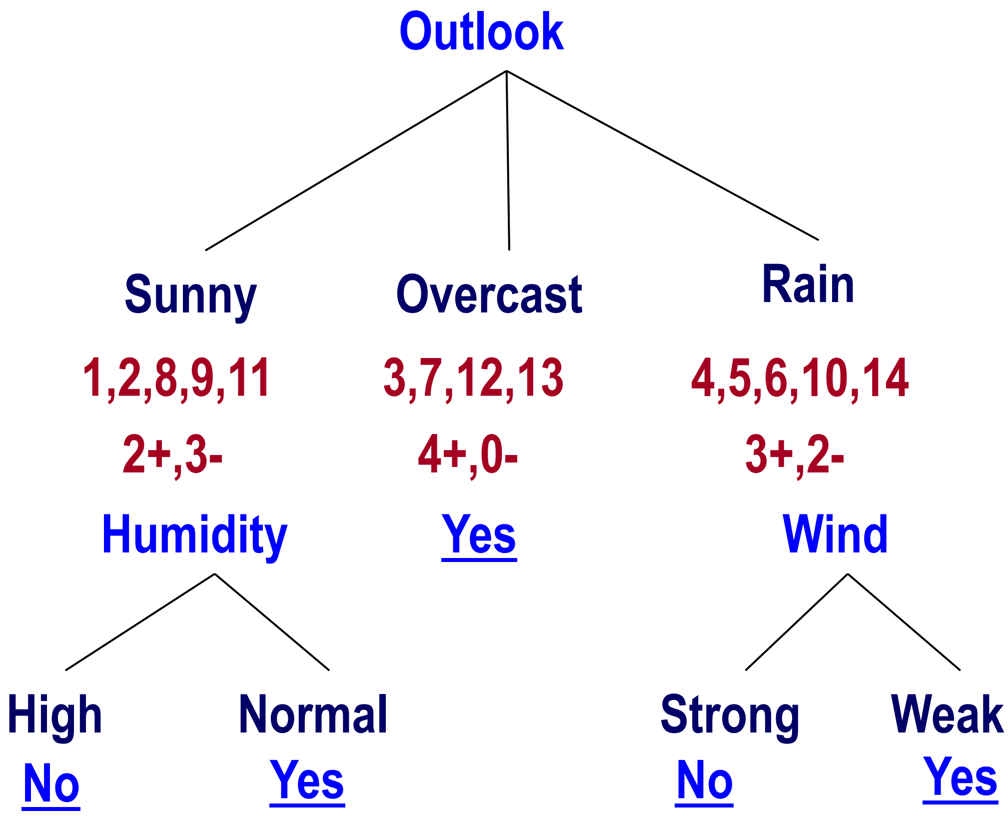
An Illustrative Example (5)



An Illustrative Example (5)



An Illustrative Example (6)



Summary: ID3 (Examples, Attributes, Label)

- Let S be the set of Examples
  Label is the target attribute (the prediction)
  Attributes is the set of measured attributes
- Create a Root node for tree
- If all examples are labeled the same return a single node tree with Label
- Otherwise Begin
- A = attribute in Attributes that *best* classifies S
- for each possible value v of A
- Add a new tree branch corresponding to A=v
- Let *Sv* be the subset of examples in S with A=v
- if *Sv* is empty: add leaf node with the most common value of Label in S
- Else: below this branch add the subtree
- ID3(*Sv*, Attributes - {a}, Label)

  End

Return Root

Hypothesis Space in Decision Tree Induction

- Conduct a search of the space of decision trees which can represent all possible discrete functions.
- Goal: to find the best decision tree
- Finding a minimal decision tree consistent with a set of data is NP-hard.
- Performs a greedy heuristic search: hill climbing without backtracking
- Makes statistically based decisions using all available data

Bias in Decision Tree Induction

- Bias is for trees of minimal depth; however, greedy search introduces complications; it positions features with high information gain high in the tree and may not find the minimal tree.

- Implements a *preference bias* (search bias) as opposed to *restriction bias* (a language bias)

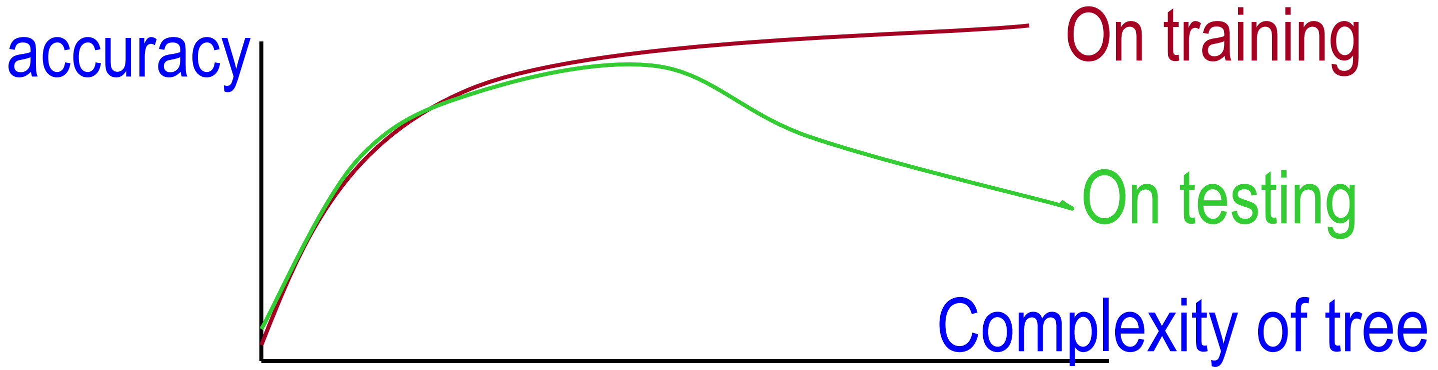
- Occam's razor can be defended on the basis that there are relatively few simple hypotheses compared to complex ones. Therefore, a simple hypothesis is that consistent with the data is less likely to be a statistical coincidence

History of Decision Tree Research

- Hunt and colleagues in Psychology used full search decision trees methods to model human concept learning in the 60's
- Quinlan developed ID3, with the information gain heuristics in the late 70's to learn expert systems from examples
- Breiman, Friedmans and colleagues in statistics developed CART (classification and regression trees) simultaneously
- A variety of improvements in the 80's: coping with noise, continuous attributes, missing data, non-axis parallel etc.
- Quinlan's updated algorithm, C4.5 (1993) is commonly used (New:C5)
- Boosting and Bagging over DTs are often good general purpose algorithms

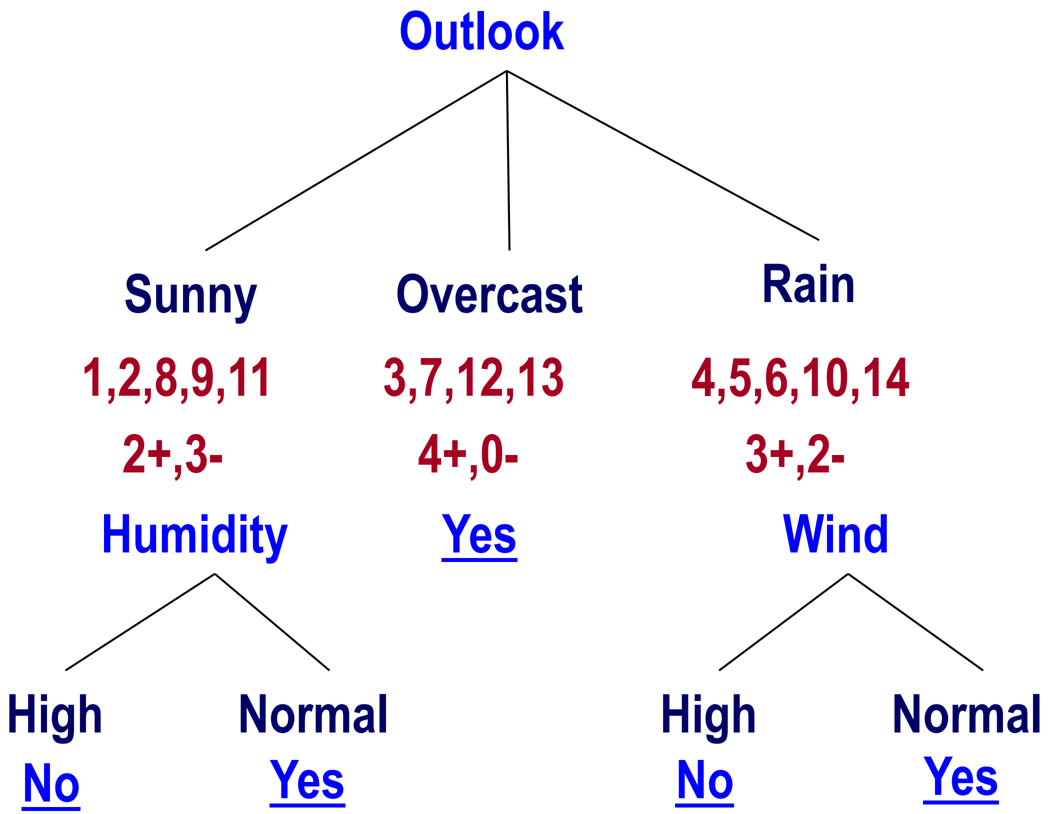
Overfitting the Data

- 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 smaller error than *h'* on the training data but *h* has larger error on the test data than *h'*.



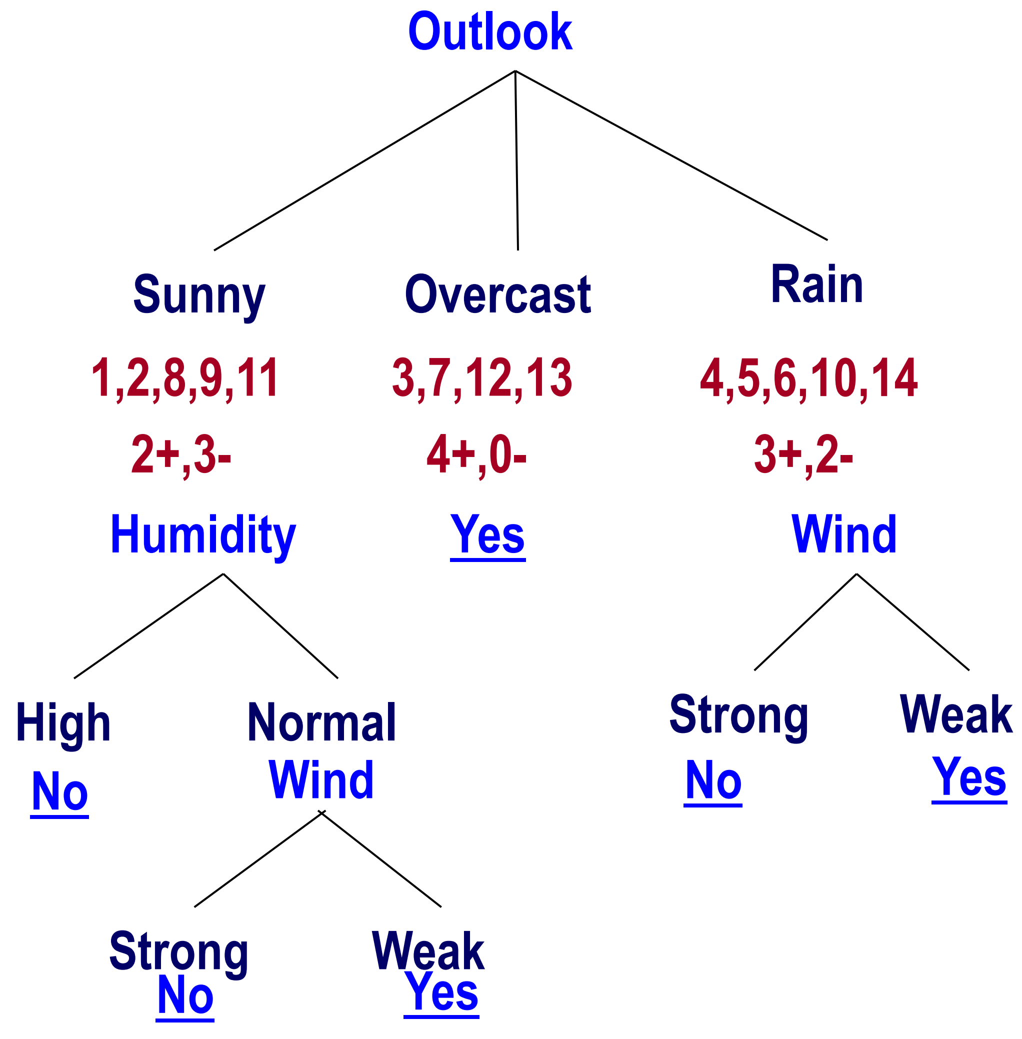
Overfitting - Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO



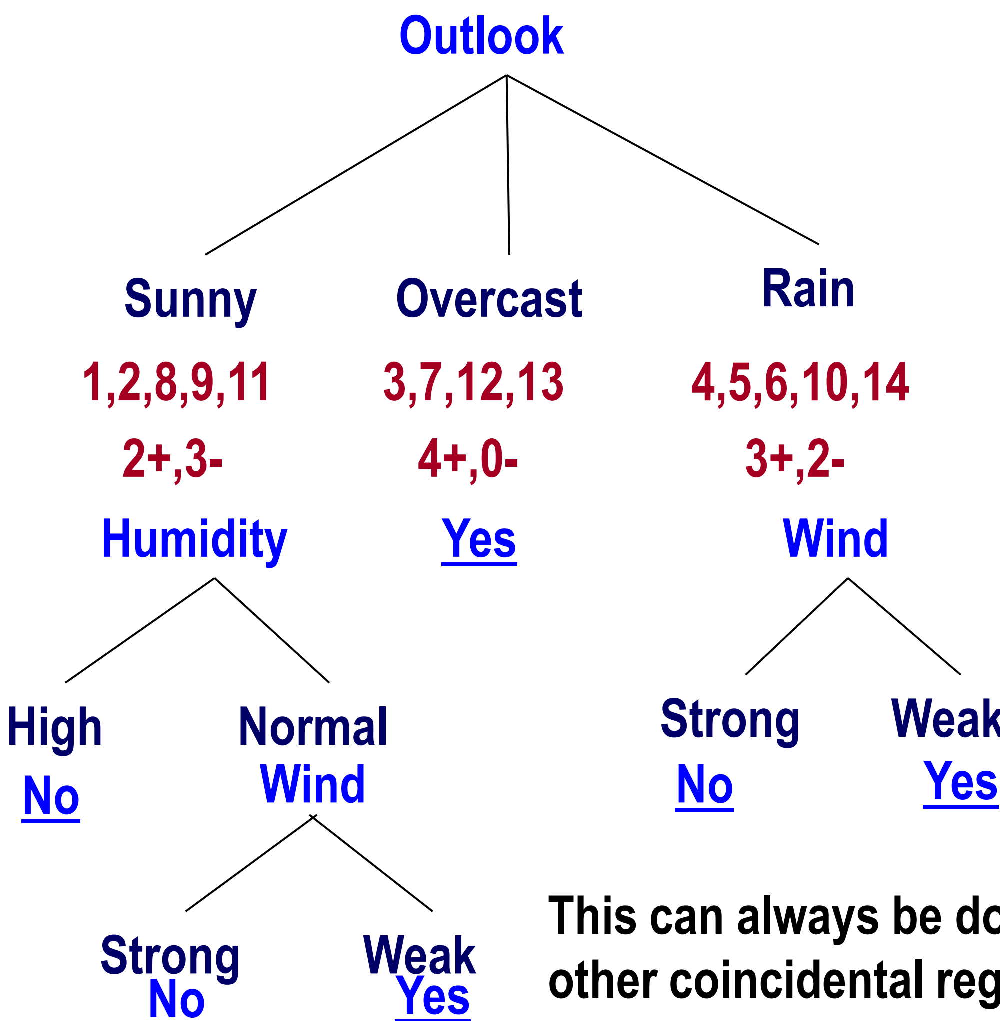
Overfitting - Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO



Overfitting - Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO



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

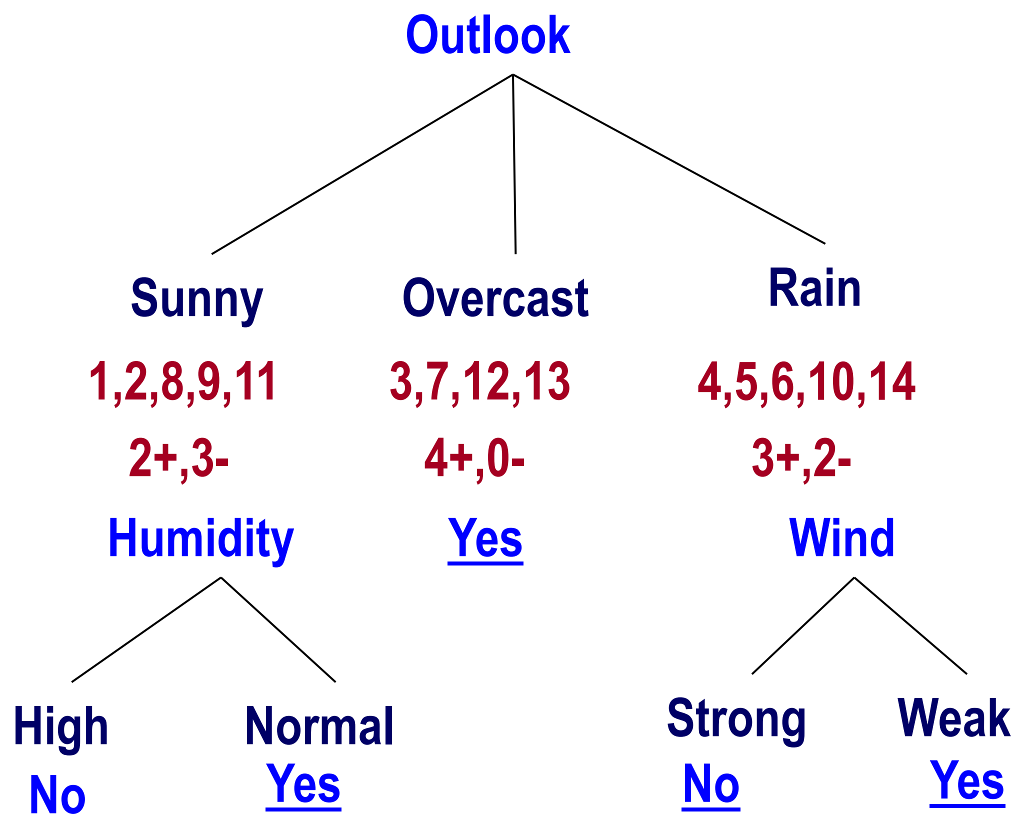
Avoiding Overfitting

- Two basic approaches
  - Prepruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Postpruning: 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 be occur by chance
  - Minimum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions ?

Trees and Rules

- Decision Trees can be represented as Rules
  - (outlook=sunny) and (humidity=high) then YES
  - (outlook=rain) and (wind=strong) then No

  ……….



Reduced-Error Pruning

- A post-pruning, cross validation approach
  - Partition training data into “grow” set and “validation” set.
  - Build a complete tree for the “grow” data
  - Until accuracy on validation set decreases, do:

    For each non-leaf node in the tree

    Temporarily prune the tree below; replace it by majority vote.

    Test the accuracy of the hypothesis on the validation set

    Permanently prune the node with the greatest increase
    in accuracy on the validation test.
- Problem: Uses less data to construct the tree

Continuous Attributes

- Real-valued attributes can, in advance, be discretized into ranges, such as *big*, *medium*, *small*
- Alternatively, one can develop splitting nodes based on thresholds of the form *A* < c that partition the data in to examples that satisfy *A* < c and *A* >= c. The information gain for these splits is calculated in the same way and compared to the information can of discrete splits.

How to find the split with the highest gain ?

- For each continuous feature *A:*

  Sort examples according to the value of *A*

  For each ordered pair (x, y) with different labels

  Check the mid-point as a possible threshold. i.e,

$$S_{a <= x}, S_{a >= y}$$

Continuous Attributes

- Example:

Length (L): 10 15 21 28 32 40 50

Class: - + + - + + -

- Check thresholds: L < 12.5; L < 24.5; L < 45

  Subset of Examples= {…}, Split= k+,j-

- How to find the split with the highest gain ?
- For each continuous feature *a:*

  Sort examples according to the value of *a*

  For each ordered pair (x,y) with different labels

  Check the mid-point as a possible threshold. i.e,

$$S_{a <= x}, S_{a>=y}$$

Missing Values with Decision Trees

- diagnosis = < fever, blood_pressure,…, blood_test=?,…>

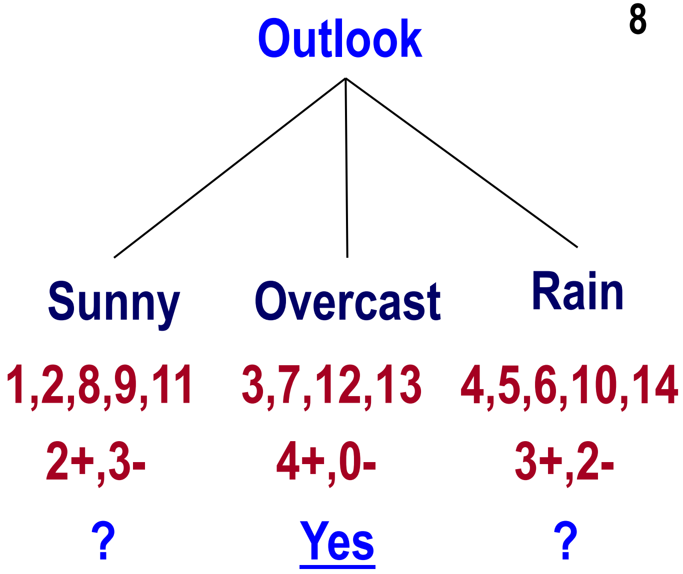
- Many times values are not available for all attributes during training or testing (e.g., medical diagnosis)

- Training: evaluate *Gain(S,a)* where in some of the examples a value for *a* is not given

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	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Missing Values

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
8	Sunny	Mild	???	Weak	No



Use:

A) the most common Humidity at Sunny

B) as (A) but with PlayTennis = No

$$Gain(S_{sunny}, Temp) =$$

$$Gain(S_{sunny}, Humidity) =$$

Missing Values

- diagnosis = < fever, blood_pressure,…, blood_test=?,…>

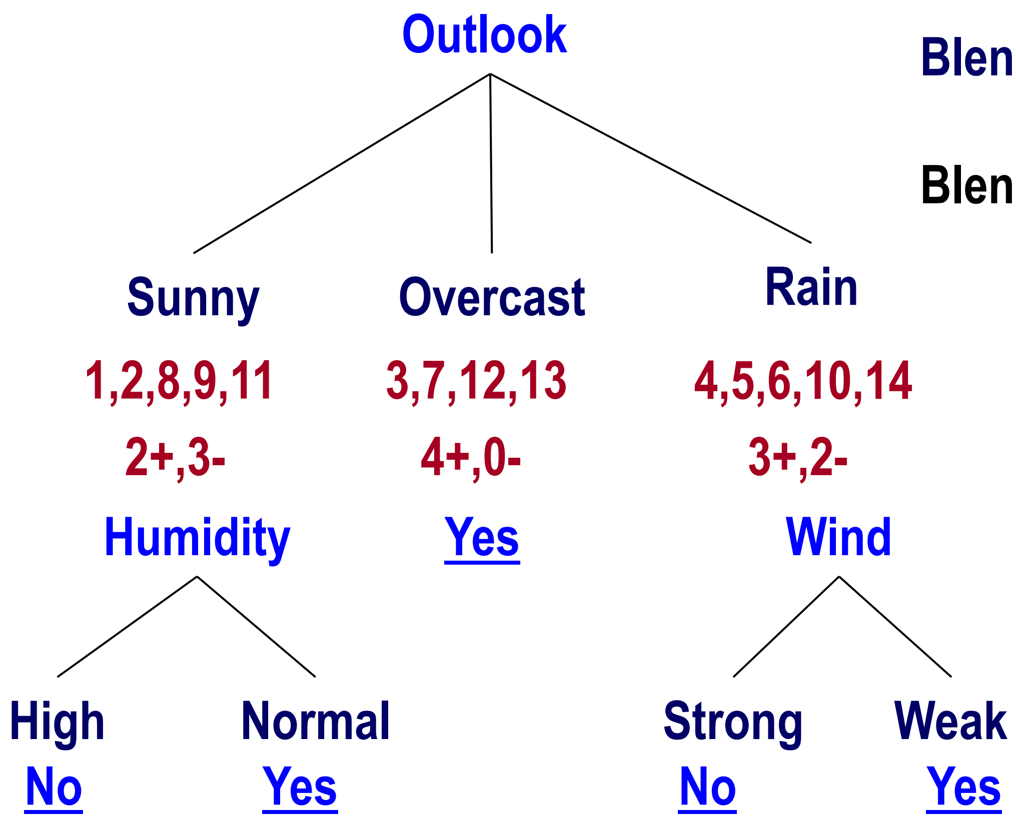
- Many times values are not available for all attributes during training or testing (e.g., medical diagnosis)

- Training: evaluate *Gain(S,a)* where in some of the examples a value for *a* is not given

- Testing: classify an example without knowing the value of *a*

Missing Values

Outlook = ???, Temp = Hot, Humidity = Normal, Wind = Strong, label = ??



Blend by labels:

1/3 Yes + 1/3 Yes +1/3 No = Yes

Blend by probability

(est. by counts)

Other Issues

- Attributes with different costs

  Change information gain so that low cost attribute are preferred

- Alternative measures for selecting attributes

  When different attributes have different number of values information gain tends to prefer those with many values

- Oblique Decision Trees

  Decisions are not axis-parallel

- Incremental Decision Trees induction

  Update an existing decision tree to account for new examples incrementally (Maintain consistency ?)

Decision Trees - Summary

- Hypothesis Space:

  Contains all functions (!)

  Variable size

  Deterministic; Discrete and Continuous attributes

- Search Algorithm

  ID3 - Eager, batch, constructive search

  Extensions: missing values

- Issues:

  What is the goal?

  When to stop? How to guarantee good generalization?