Machine Learning

Lecture 12: Decision Tree

Sibei Yang

SIST

Email: yangsb@shanghaitech.edu.cn

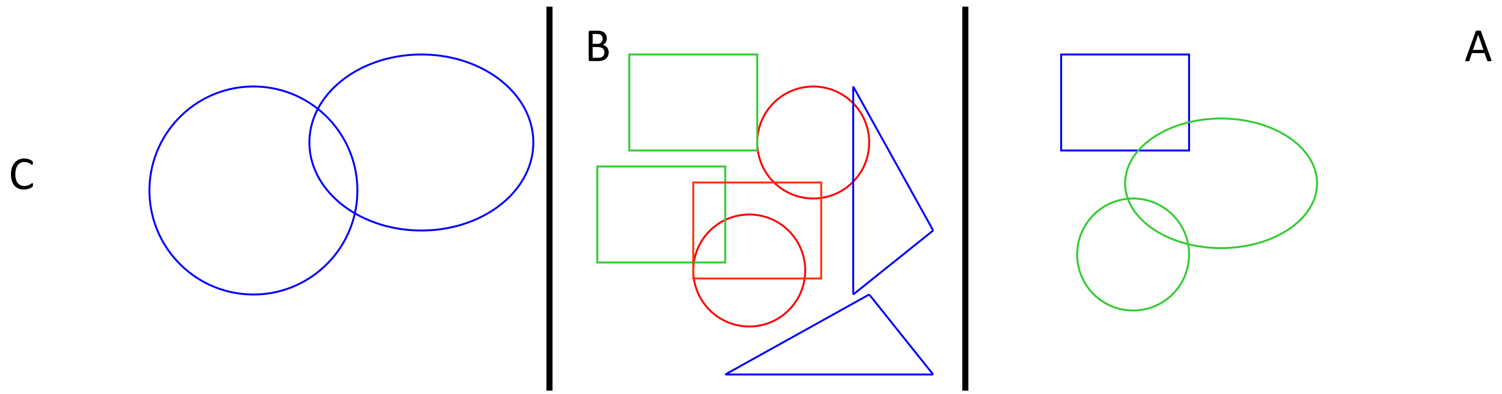
Decision Trees

- 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



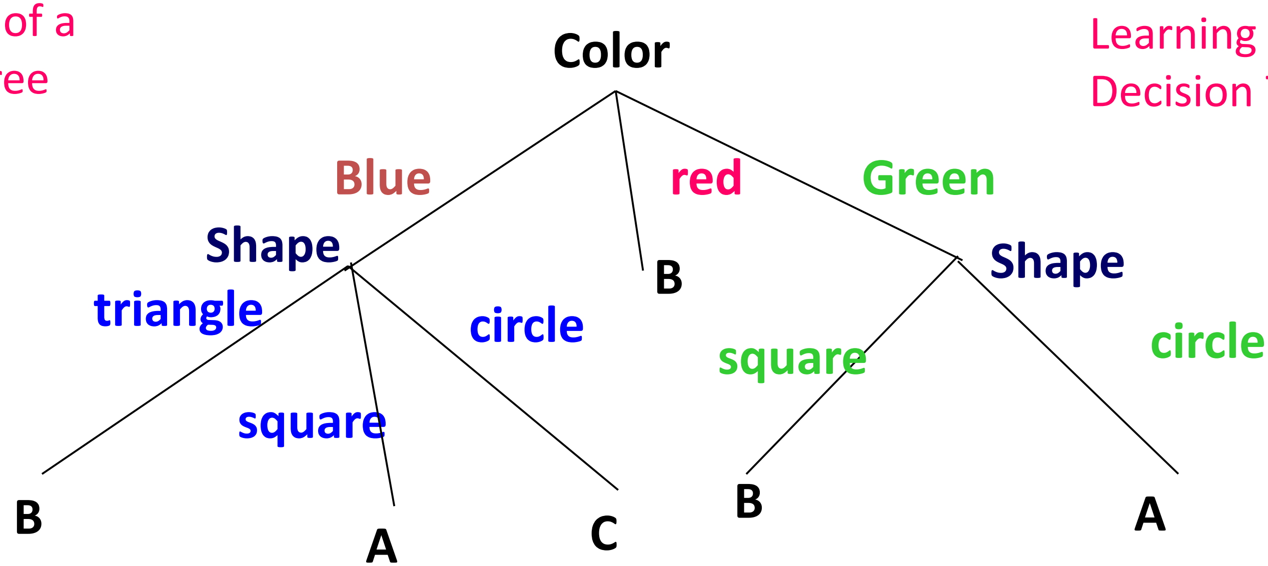
Decision Trees: The Representation

- 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

(color= RED; shape=triangle)

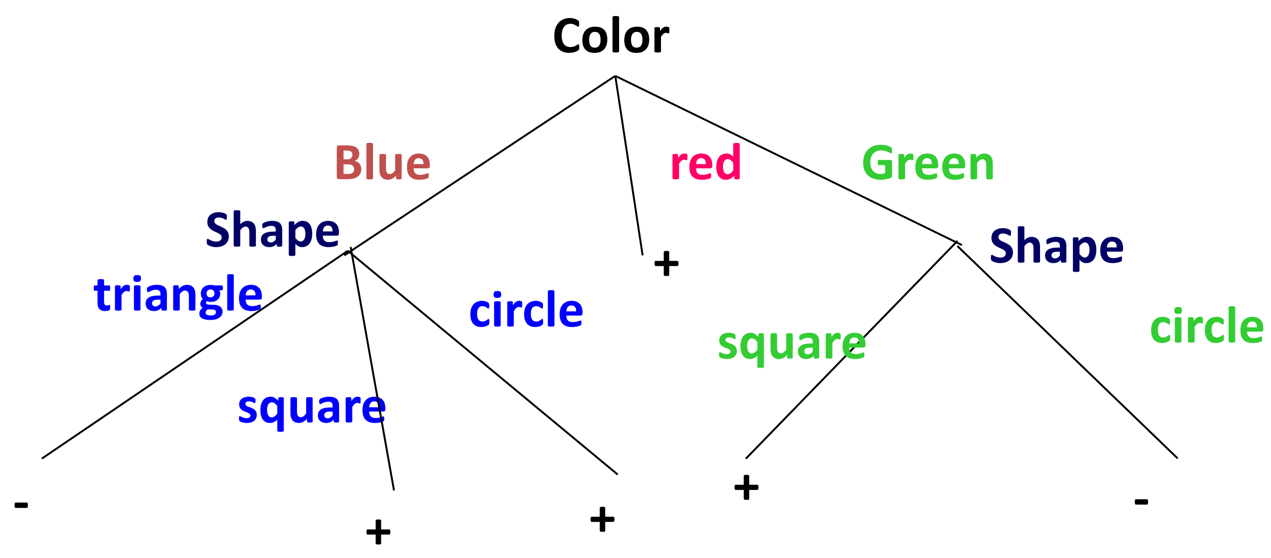
Evaluation of a Decision Tree

Learning a Decision Tree



Decision Trees

- Output is a discrete category.
  Real valued outputs are possible (regression trees)
- There are efficient algorithms for processing large amounts of data. (But not too many features)
- There are methods for handling noisy data (classification noise and attribute noise) and for handling missing attribute values.

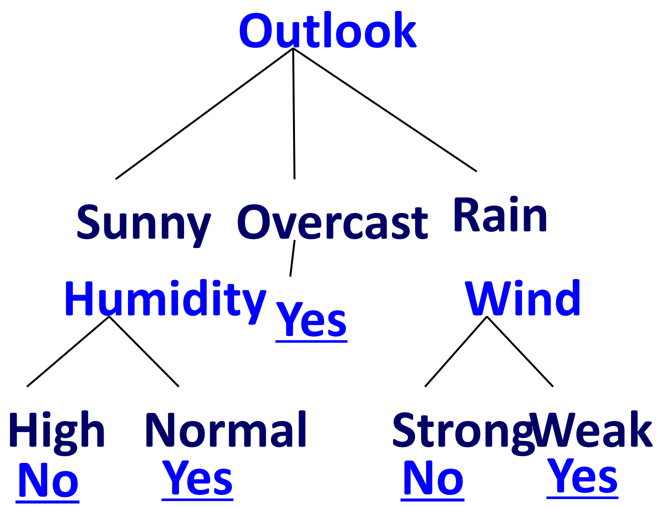


Decision Trees

- Can be viewed as a way to compactly represent a lot of data.

- The evaluation of the Decision Tree Classifier is easy

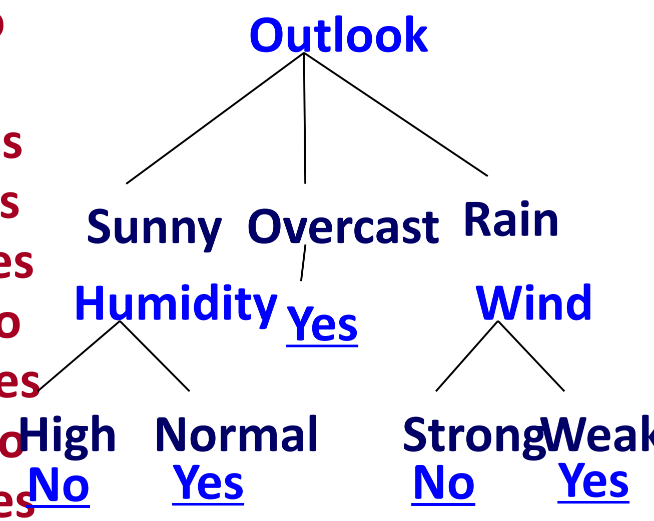
- Clearly, given data, there are many ways to Represent it as a decision tree.
- Learning a good representation from data is the challenge.



Basic Decision Trees Learning Algorithm

- Data is processed in Batch (I.e., all the data is available).
- Recursively build a decision tree top-down.

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



Basic Decision Trees Learning Algorithm

- DT(Examples, Attributes)

  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

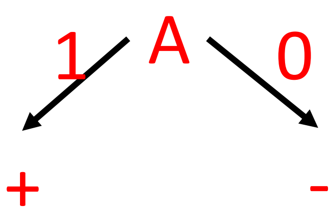
- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
- Finding the minimal decision tree consistent with the data is NP-hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm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

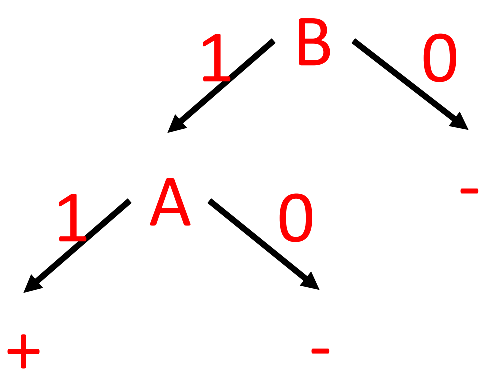
Picking the Root Attribute

- 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).

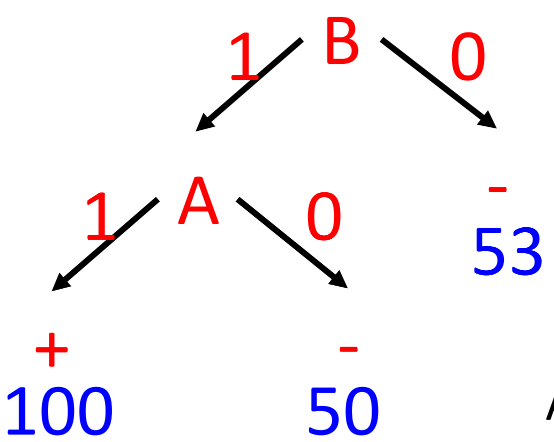
  < (A=0,B=0), - >: 50 examples

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

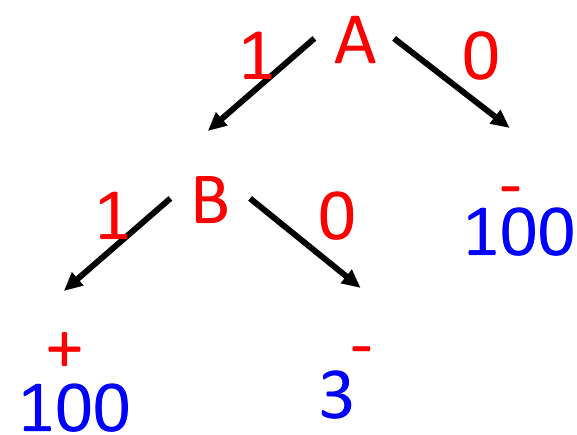
  < (A=1,B=0), - >: ~~0 examples~~ 3 examples

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

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



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 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

- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

$$\text{Entropy}(S) = -p_{+}\log(p_{+}) - p_{-}\log(p_{-})$$

where p_{+} is the proportion of positive examples in S and

p_{-} is the proportion of negatives.

- If all the examples belong to the same category: *Entropy* = 0
- If the examples are equally mixed (0.5,0.5) *Entropy* = 1

In general, when p_i is the fraction of examples labeled i:

$$\text{Entropy}(\{p_1, p_2, \ldots p_k\}) = -\sum_{i=1}^{k} p_i \log(p_i)$$

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.

Entropy

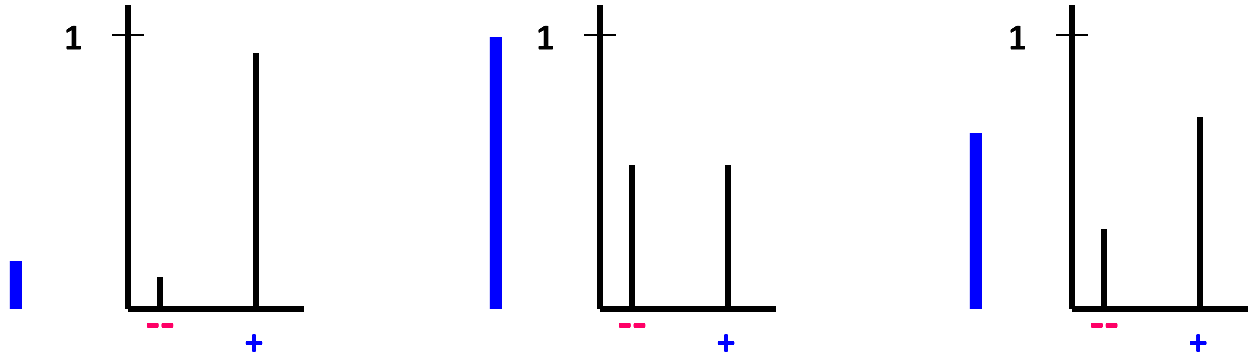
- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

$$\text{Entropy}(S) = -p_+ \log(p_+) - p_- \log(p_-)$$

where p_+ is the proportion of positive examples in S and

p_- is the proportion of negatives.

- If all the examples belong to the same category: *Entropy* = 0
- If the examples are equally mixed (0.5,0.5) *Entropy* = 1



Entropy

- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

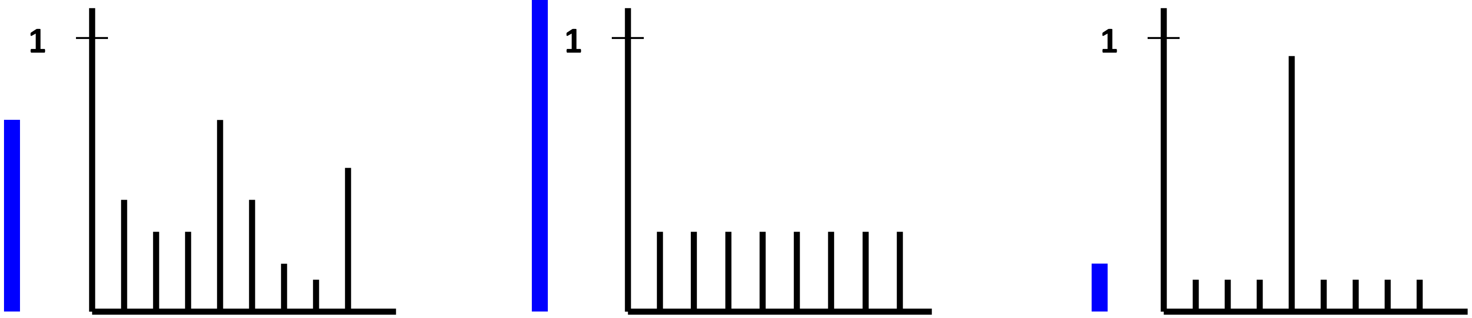
$$Entropy(S) = -p_{+}\log(p_{+}) - p_{-}\log(p_{-})$$

where p_{+} is the proportion of positive examples in S and

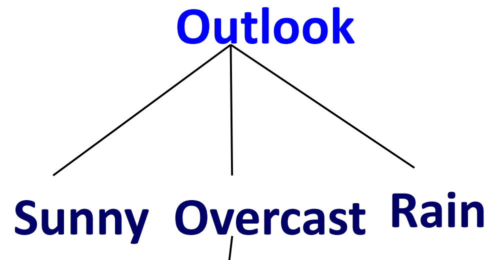
p_{-} is the proportion of negatives.

- If all the examples belong to the same category: *Entropy* = 0

- If the examples are equally mixed (0.5,0.5) *Entropy* = 1



Information Gain



- The information gain of an attribute *a* is the expected reduction in entropy caused by partitioning on this attribute.

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

where S_v is the subset of S for which attribute *a* has value *v*

and the 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 lead to high gain

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

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

9+,5-

$$\text{Entropy}(S) = -{}^{9}\!/_{14}\log({}^{9}\!/_{14}) - {}^{5}\!/_{14}\log({}^{5}\!/_{14}) = 0.94$$

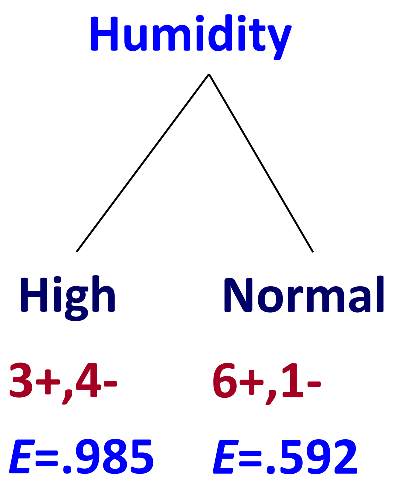
An Illustrative Example (2)

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

An Illustrative Example (2)



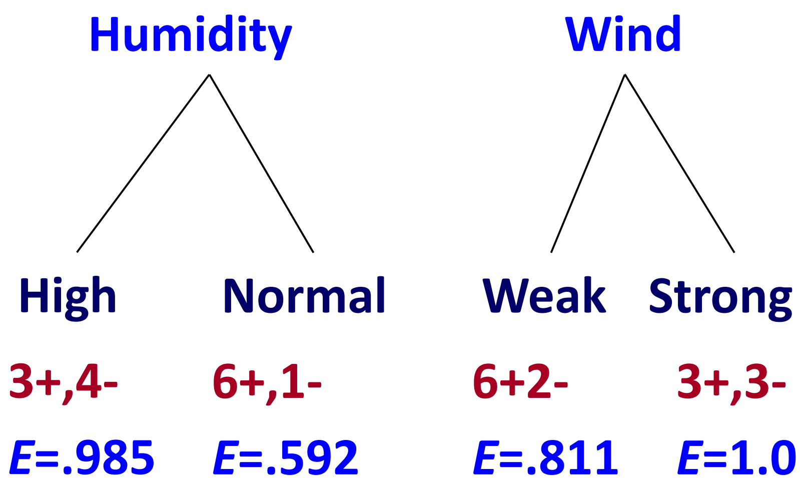
Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

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

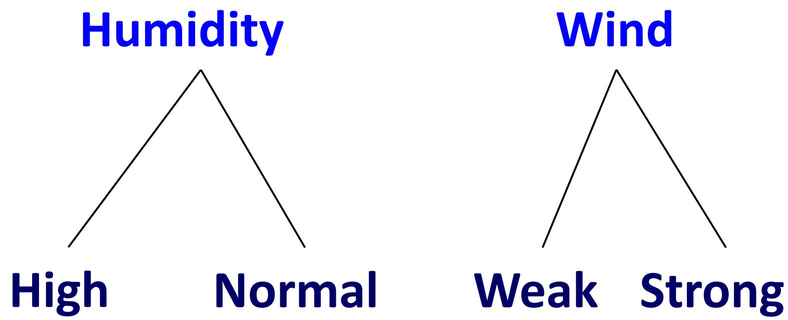
$$\text{Gain}(S,a) = \text{Entropy}(S) - \sum_{v \in \text{values}(a)} \frac{|S_v|}{|S|}\text{Entropy}(S_v)$$

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



High	Normal	Weak	Strong
3+,4-	6+,1-	6+2-	3+,3-
E=.985	*E*=.592	*E*=.811	*E*=1.0

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

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

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

Humidity

High: 3+,4- *E*=.985

Normal: 6+,1- *E*=.592

Wind

Weak: 6+2- *E*=.811

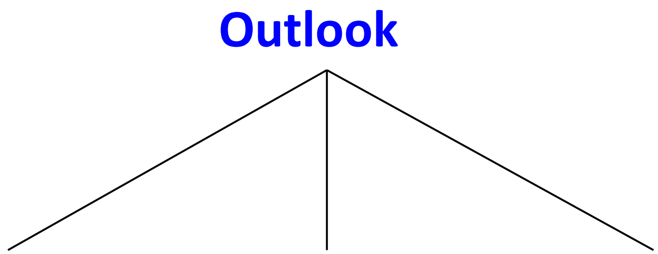
Strong: 3+,3- *E*=1.0

*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

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

An Illustrative Example (3)



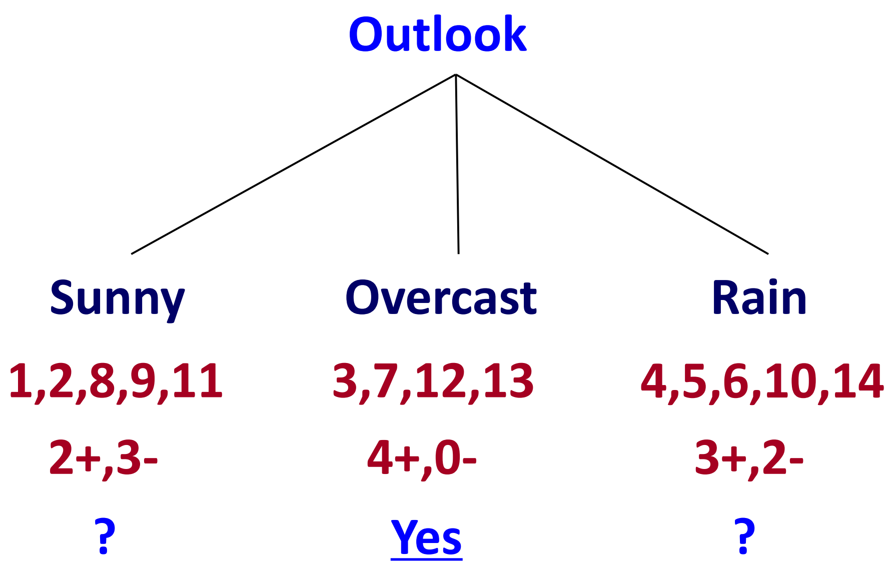
*Gain(S,*Humidity*)*=0.151

*Gain(S,*Wind*)*=0.048

*Gain(S,*Temperature*)*=0.029

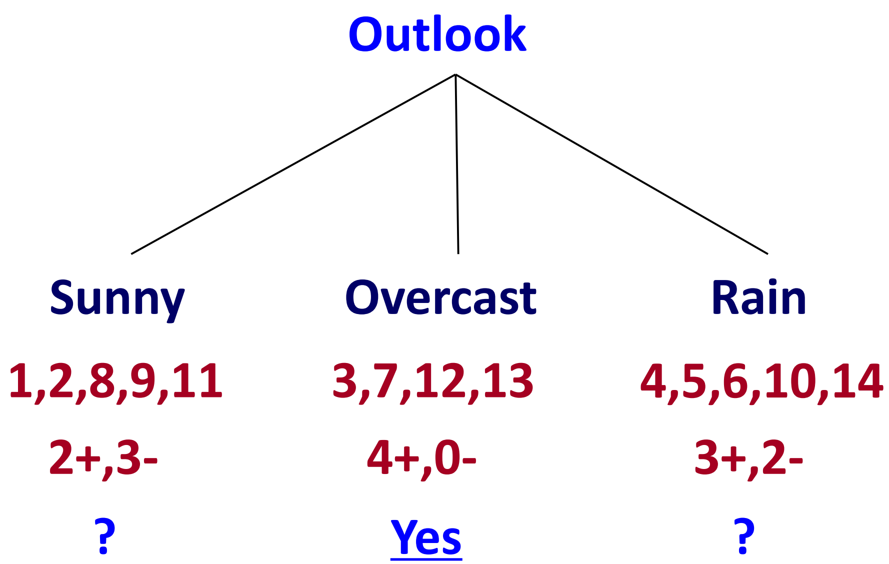
*Gain(S,*Outlook*)*=0.246

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

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

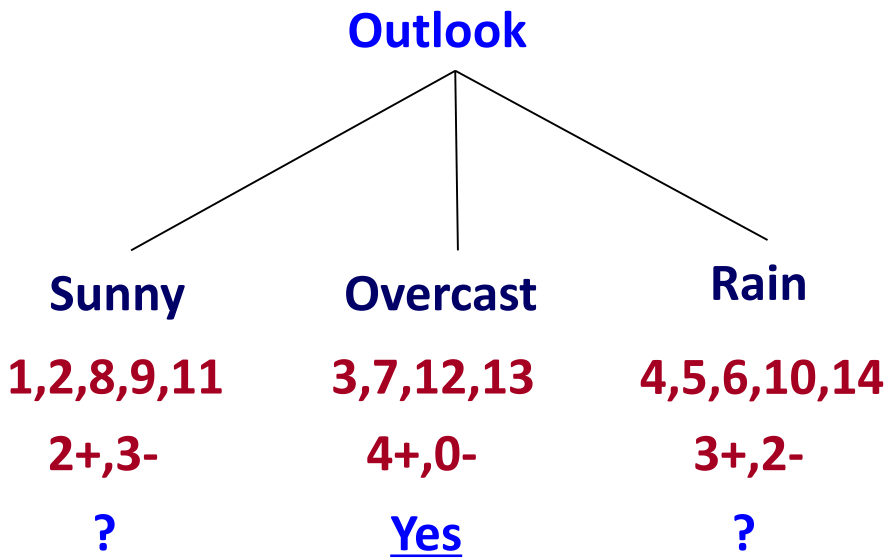
$Gain(S_{sunny}, Wind) =$ .97-(2/5) 1 - (3/5) .92= .02

Outlook

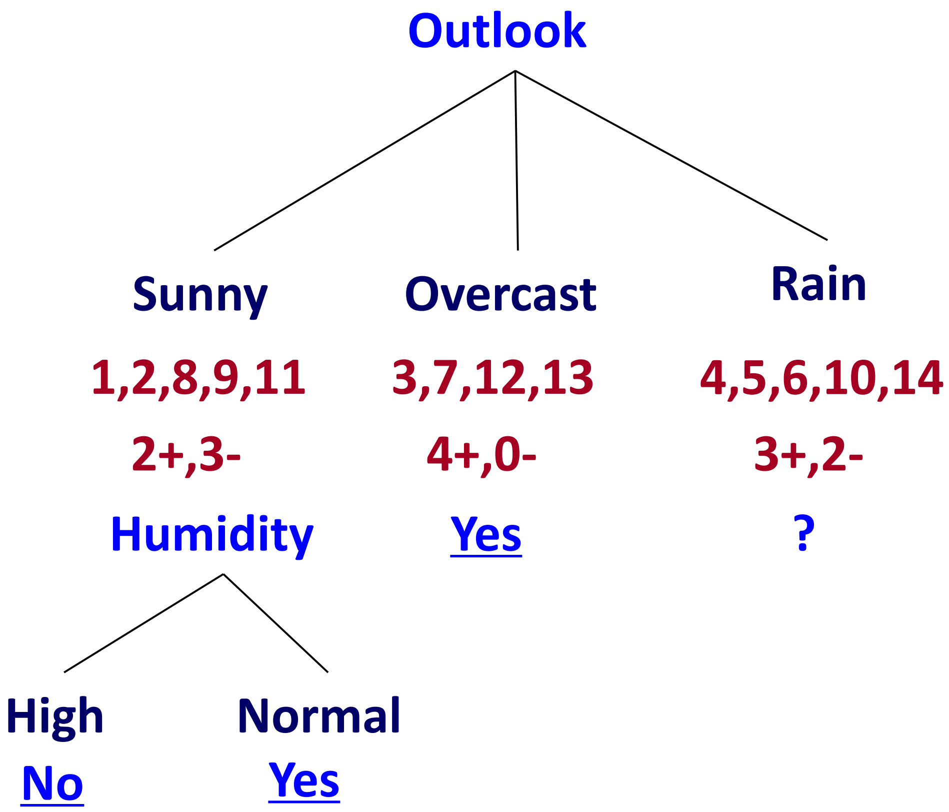
Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

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

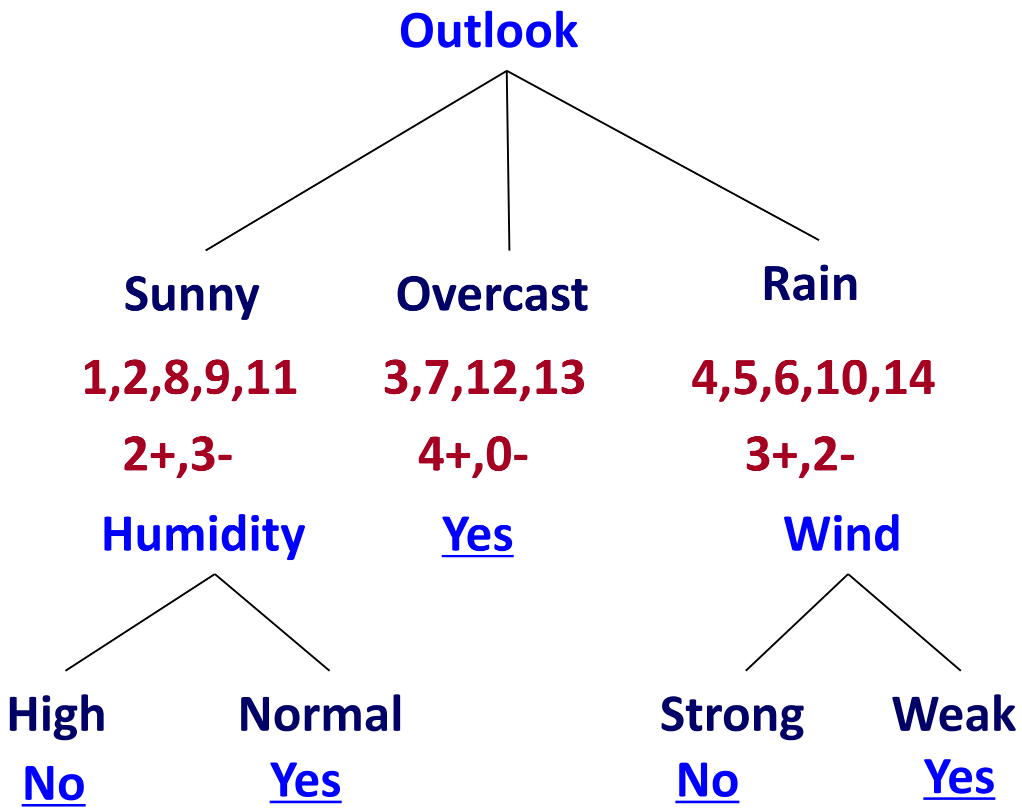
An Illustrative Example (5)



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 common value of Label in S
-    Else: below this branch add the subtree
-     ID3(*Sv*, Attributes - {a}, Label)

  End

Return Root

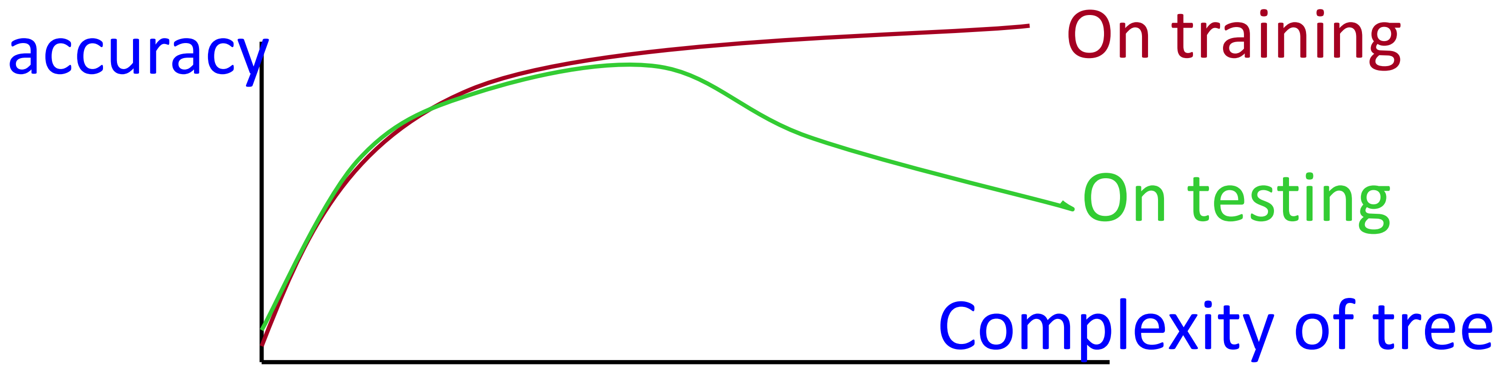
History of Decision Tree Research

- 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 (or Bagging) over DTs is a very good general purpose algorithm

How can this be avoided with linear classifiers?

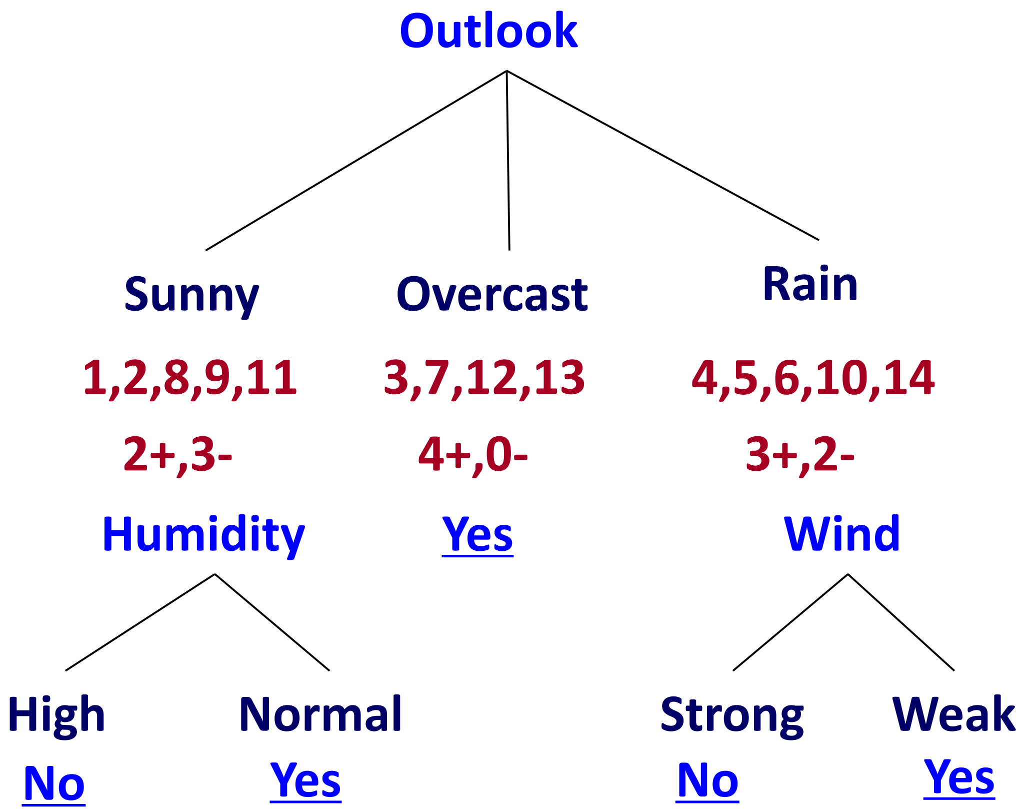
Overfitting the Data

- 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'*.



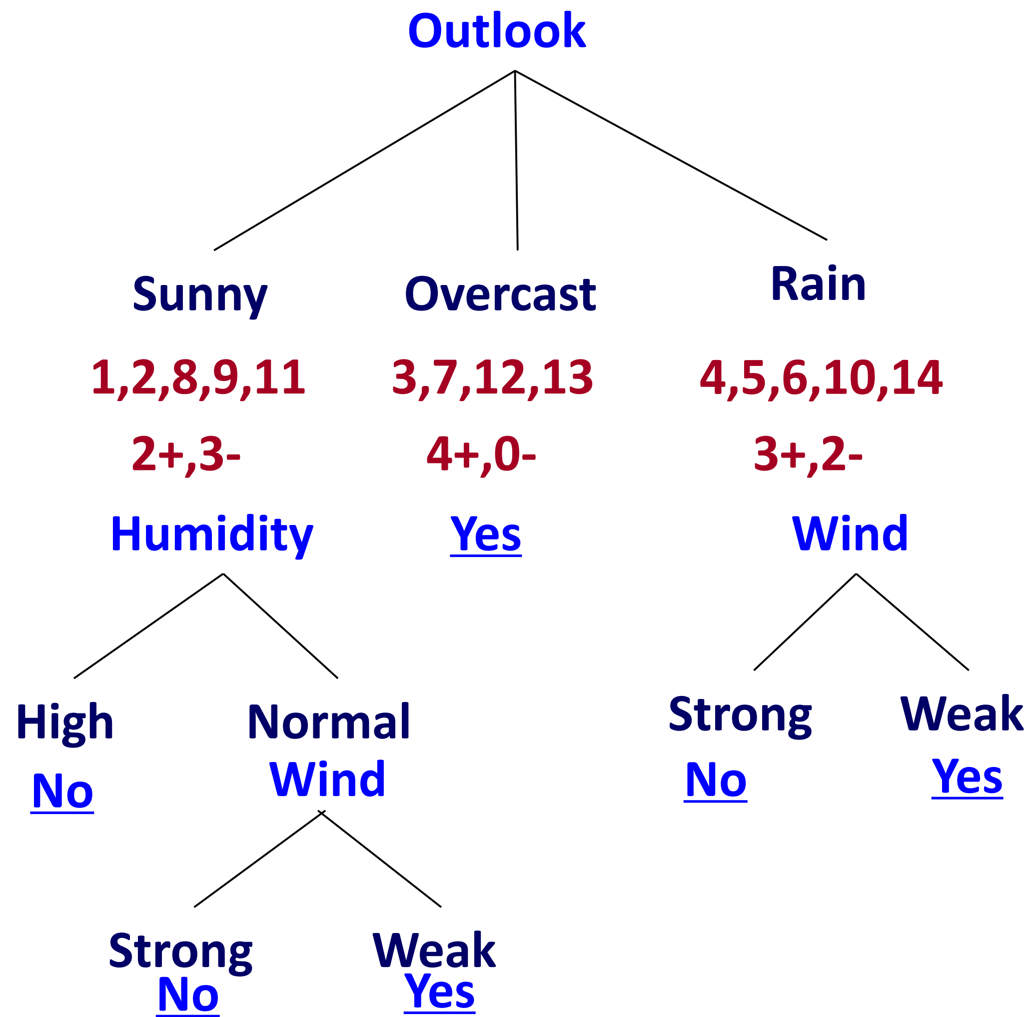
Overfitting - Example

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



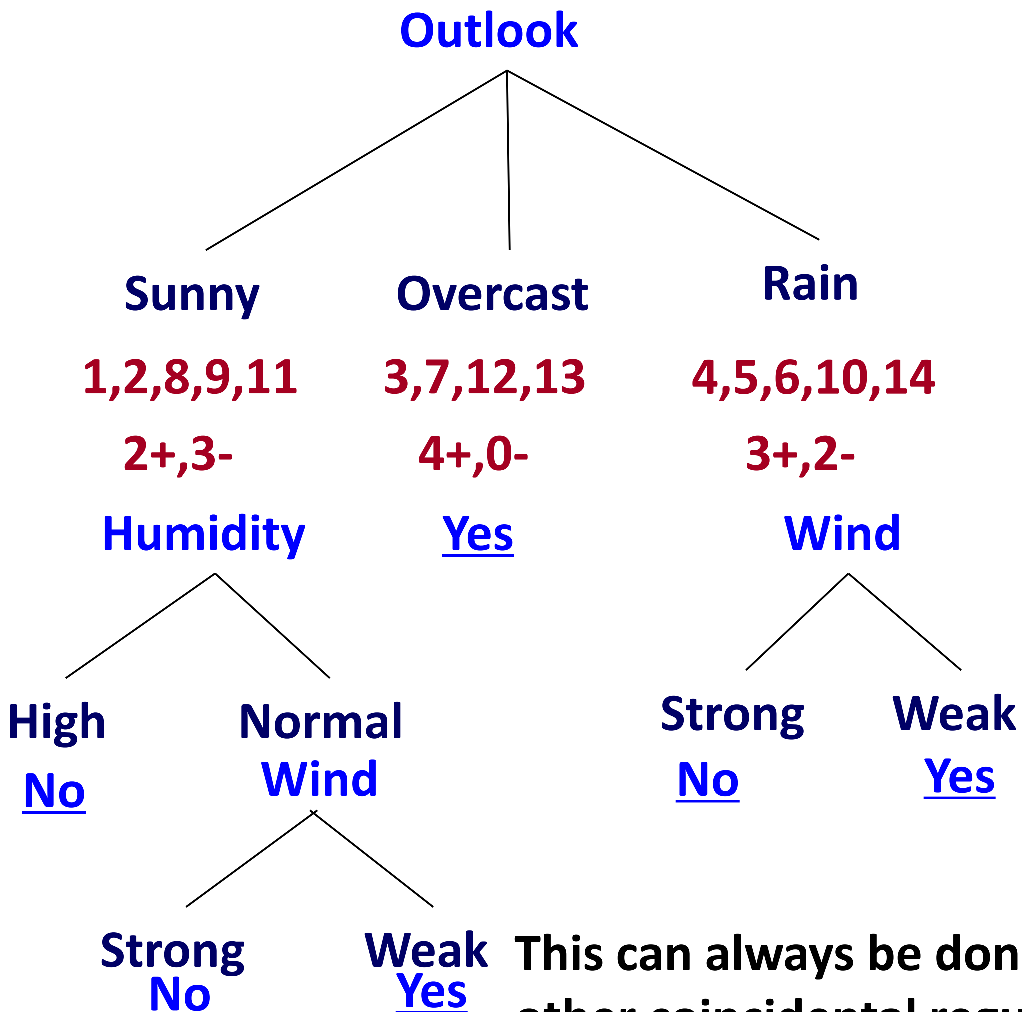
Overfitting - Example

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
  - Maximum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions ?

This is related to the notion of regularization that we have learned

Reduced-Error Pruning

- A post-pruning, cross validation approach
  - Partition training data into "training" set and "validation" set.
  - Build a complete tree for the "training" set
  - 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

General Strategy: Overfit and Simplify