Decision Trees

1

Outline

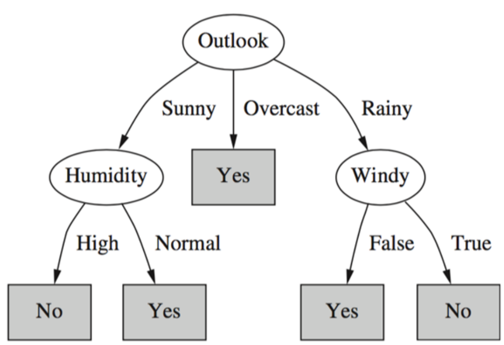
- Decision tree representation
- Decision tree learning (ID3)
  - Information gain
- C4.5
  - Deal with overfitting: pruning
- Trees for numeric prediction
  - Regression tree and Model tree

2

The Weather Data

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

A decision tree for this problem



Decision trees

- Decision tree representation
  - each internal node tests on an attribute
  - each branch corresponds to an attribute value
  - each leaf node corresponds to a class label
- When to consider decision trees
  - Produce comprehensible results
  - Decision trees are especially well suited for representing simple rules for classifying instances that are described by discrete attribute values
  - Decision tree learning algorithms are relatively efficient – linear in the size of the decision tree and the size of the data set
  - Are often among the first to be tried on a new data set

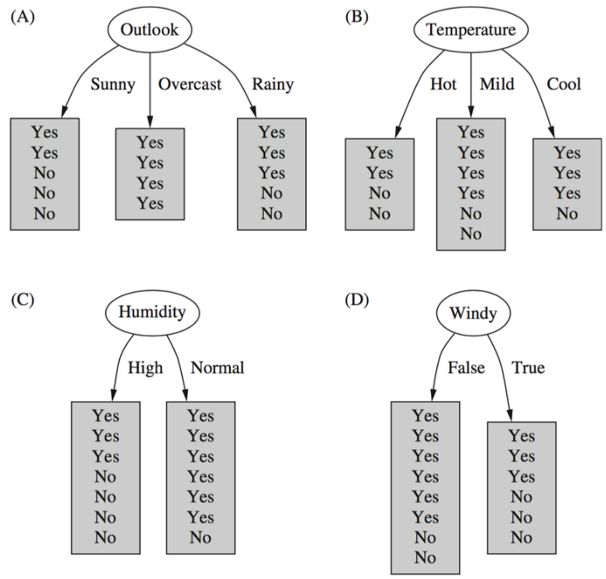
5

Constructing decision trees

- First consider discrete valued attributes
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

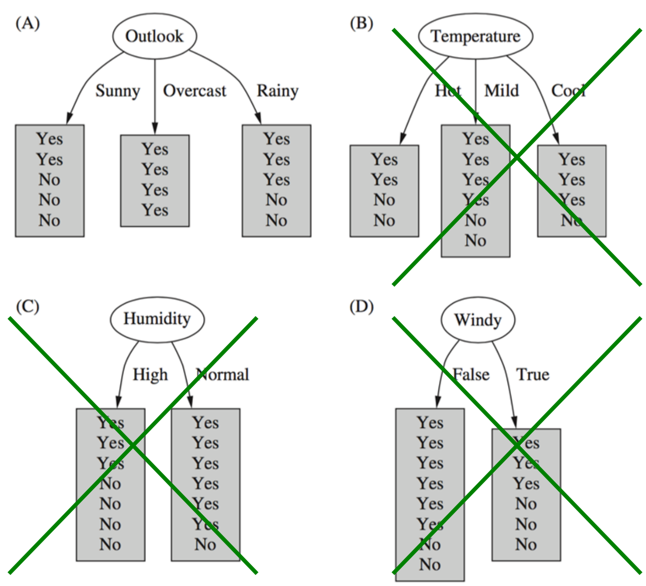
6

Which attribute to select?



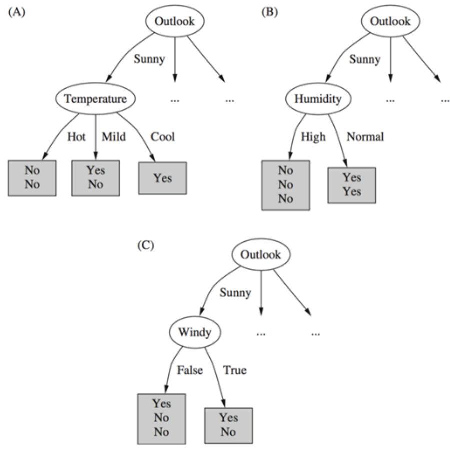
7

Which attribute to select?



8

Continuing to split



9

Constructing decision trees

- Strategy: top down learning using recursive *divide-and-conquer* process
  - First: select attribute for root node
    Create branch for each possible attribute value
  - Then: split instances into subsets
    One for each branch extending from the node
  - Finally: repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class

10

Decision tree learning

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
      best ← CHOOSE-ATTRIBUTE(attributes, examples)
      tree ← a new decision tree with root test best
      for each value v_i of best do
          examples_i ← {elements of examples with best = v_i}
          subtree ← DTL(examples_i, attributes − best, MODE(examples))
          add a branch to tree with label v_i and subtree subtree
      return tree
```

- Base cases:
  - uniform example classification
  - empty examples: majority classification at the node's parent
  - empty attributes: use a majority vote

11

Criterion for attribute selection

- Which is the best attribute?
  - Want to get the smallest tree
  - Heuristic: choose the attribute that produces the "purest" nodes
- Popular selection criterion: *information gain*
  - from information theory
- Strategy: amongst attributes available for splitting, choose attribute that gives greatest information to the class variable

12

Entropy

- The expected information required to determine an outcome (i.e., class value) of a random variable, is its *entropy*
- Formula for computing the entropy of a discrete random variable with distribution $(p_1, \ldots, p_n)$:

$$\text{Entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

- A measure of the uncertainty associated with a random variable
- Using base-2 logarithms, entropy gives the information required in expected *bits*
- Entropy is maximal when all classes are equally likely and minimal when one of the classes has probability 1

13

Example

- The entropy of the class variable (before splitting)

Info([9,5]) = 0.940 bits

14

Example: attribute *Outlook*

The entropy of the class variable conditioned on *Outlook*

- *Outlook* = *Sunny* :

$$\text{Info}([2,3]) = 0.971 \text{ bits}$$

- *Outlook* = *Overcast* :

$$\text{Info}([4,0]) = 0.0 \text{ bits}$$

- *Outlook* = *Rainy* :

$$\text{Info}([3,2]) = 0.971 \text{ bits}$$

- The *conditional entropy*, the remaining information needed or the average uncertainty about the class after observing the value of *Outlook*:

$$\begin{aligned}\text{Info}([2,3],[4,0],[3,2]) &= (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971\\ &= 0.693 \text{ bits}\end{aligned}$$

15

Computing information gain

- Information gain (mutual information): information before splitting – information after splitting

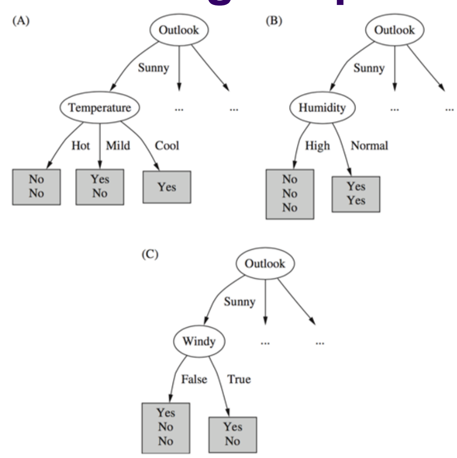
Gain(*Outlook*) = Info([9,5]) – info([2,3],[4,0],[3,2])
= 0.940 – 0.693
= 0.247 bits

- Information gain for attributes from weather data:

Gain(*Outlook*)	= 0.247 bits
Gain(*Temperature*)	= 0.029 bits
Gain(*Humidity*)	= 0.152 bits
Gain(*Windy*)	= 0.048 bits

16

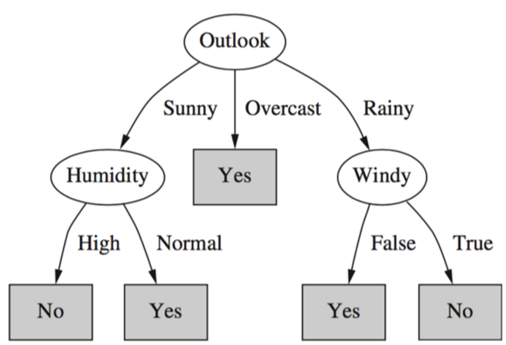
Continuing to split



Gain(*Temperature*) = 0.571 bits
Gain(*Humidity*) = 0.971 bits
Gain(*Windy*) = 0.020 bits

17

Final decision tree



18

Discussion

- Top-down induction of decision trees: ID3, algorithm developed by Ross Quinlan
- Similar approach: CART tree learner
  - Classification and Regression Trees
  - Uses Gini index rather than entropy to measure impurity
- There are many other attribute selection criteria!
  (But little difference in accuracy of result)

Industrial-strength algorithms

- For an algorithm to be useful in a wide range of real-world applications it must:
  - Permit numeric attributes
  - Allow missing values
  - Be robust in the presence of noise
- Basic scheme needs to be extended to fulfill these requirements

From ID3 to C4.5

- Extending ID3:
  - to permit numeric attributes: *Discretize data*
  - to deal sensibly with missing values: *trickier*
  - stability for noisy data: *requires pruning mechanism*
- End result: C4.5 (Quinlan)
  - Best-known and (probably) most widely-used learning algorithm
  - Commercial successor: C5.0

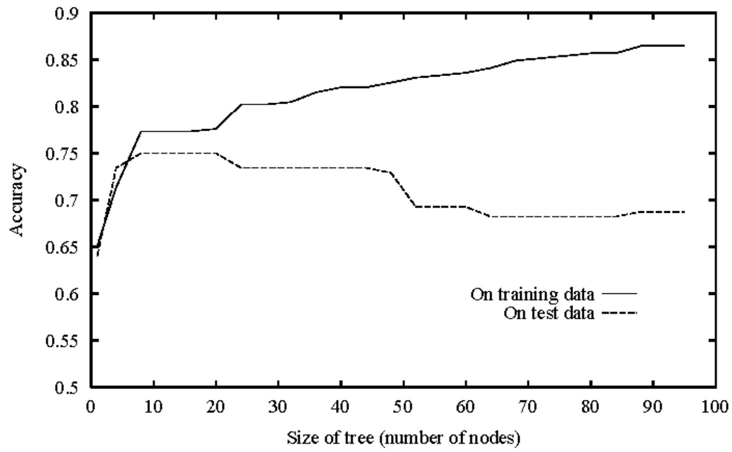
21

Overfitting in Decision Trees

- The algorithm grows each branch of the tree to perfectly classify the training examples
- When there is noise in the data -- adding an incorrect example leads to a more complex tree with irrelevant attributes
- When the number of training examples in a branch is too small -- poor estimates of entropy, irrelevant attributes may partition the examples well by accident
- **Overfitting** the data

22

Overfitting in Decision Trees



23

Pruning

- Prevent overfitting the training data: "prune" the decision tree
- Two strategies:
  - *Prepruning*
    stop growing a branch when information becomes unreliable, based on statistical significance test
  - *Postpruning*
    take a fully-grown decision tree and discard unreliable parts
- Postpruning is preferred in practice—prepruning can "stop early"

24

Discussion

TDIDT: Top-Down Induction of Decision Trees

- Extensively studied method of machine learning used in data mining
- Different criteria for attribute selection rarely make a large difference
- Different pruning methods change the size of the resulting tree
- C4.5 (Quinlan 1993): best-known and (probably) most widely-used learning algorithm
  - Commercial successor C5.0 much faster and a bit more accurate
- CART (Classification and Regression Trees) (Breiman et al. 1984)
  - CART's pruning method can often produce smaller trees than C4.5's method

25

Trees for numeric prediction

- *Regression*: the process of computing an expression that predicts a numeric quantity
- *Regression tree*: "decision tree" where each leaf predicts a numeric quantity
  - Predicted value is average value of training instances that reach the leaf
- *Model tree:* "regression tree" with linear regression models at the leaf nodes
  - Linear patches approximate continuous function

26

Predicting CPU performance

- Example: 209 different computer configurations

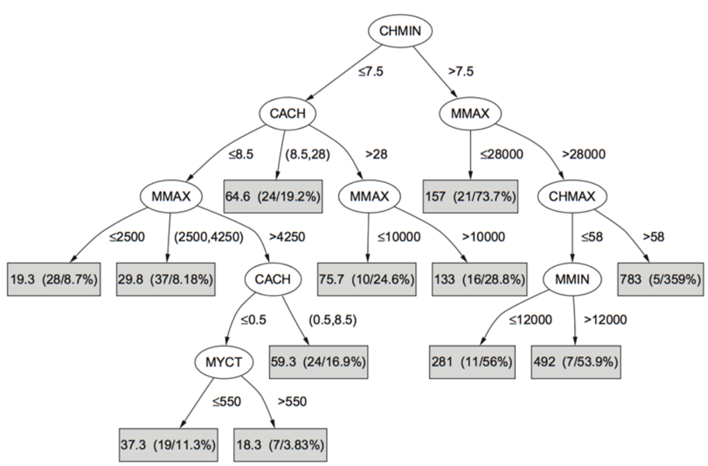
	Cycle time (ns)	Main memory (Kb)		Cache (Kb)	Channels		Performance
	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	PRP
1	125	256	6000	256	16	128	198
2	29	8000	32000	32	8	32	269
…							
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

- Linear regression function

```
PRP = -55.9 + 0.0489 MYCT + 0.0153 MMIN + 0.0056 MMAX
      + 0.6410 CACH - 0.2700 CHMIN + 1.480 CHMAX
```

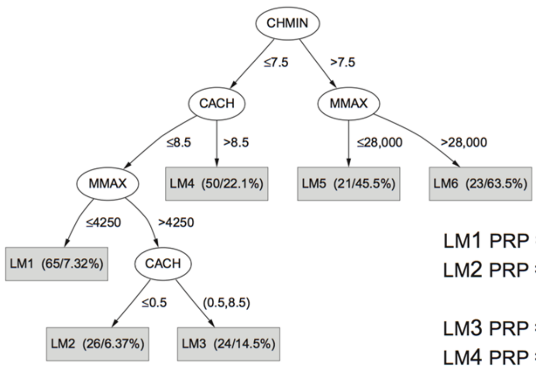
27

Regression tree for the CPU data



28

Model tree for the CPU data



LM1 PRP = 8.29 + 0.004 MMAX + 2.77 CHMIN
LM2 PRP = 20.3 + 0.004 MMIN – 3.99 CHMIN
+ 0.946 CHMAX
LM3 PRP = 38.1 + 0.012 MMIN
LM4 PRP = 19.5 + 0.002 MMAX + 0.698 CACH
+ 0.969 CHMAX
LM5 PRP = 285 1.46 MYCT + 1.02 CACH
– 9.39 CHMIN
LM6 PRP = – 65.8 + 0.03 MMIN – 2.94 CHMIN
+ 4.98 CHMAX

29