Machine Learning

Prof. Dr. Stefan Kramer
Johannes Gutenberg-Universität Mainz

Outline

- Decision tree learning
- Choosing attributes: entropy and information gain
- Overfitting avoidance and decision tree pruning
- Other issues in decision tree learning
- Ensembles: bagging, random forests and bias-variance decomposition

Decision Tree Learning

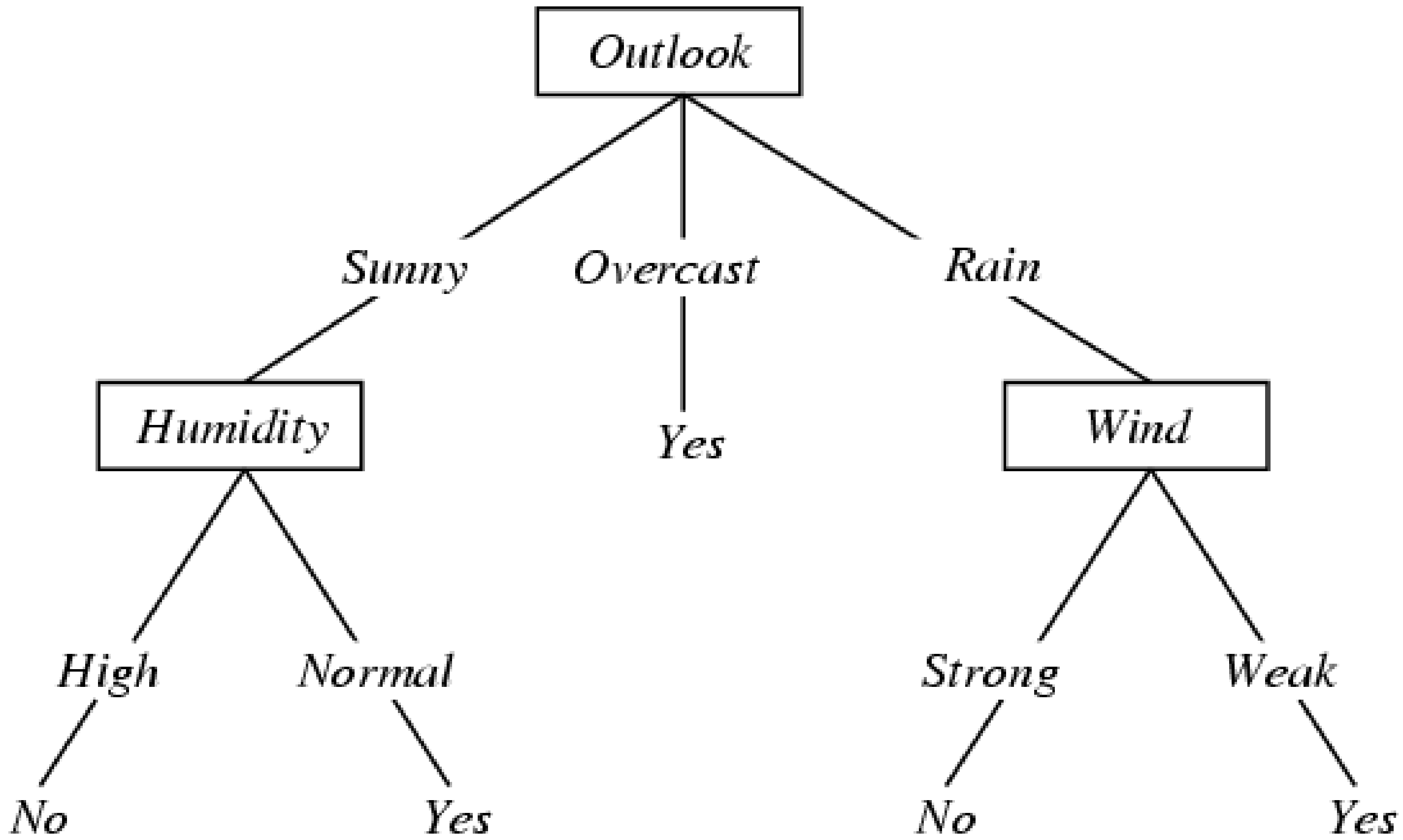
Example Dataset

Day	Outlook	Temp.	Hum.	Wind	PlayT.
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Training Set

Test Set

Example Tree



Decision Tree Representation

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent?

- and, or, XOR
- (A and B) or (C and not D and E)
- M of N

Top-Down Induction of Decision Trees

Main loop:

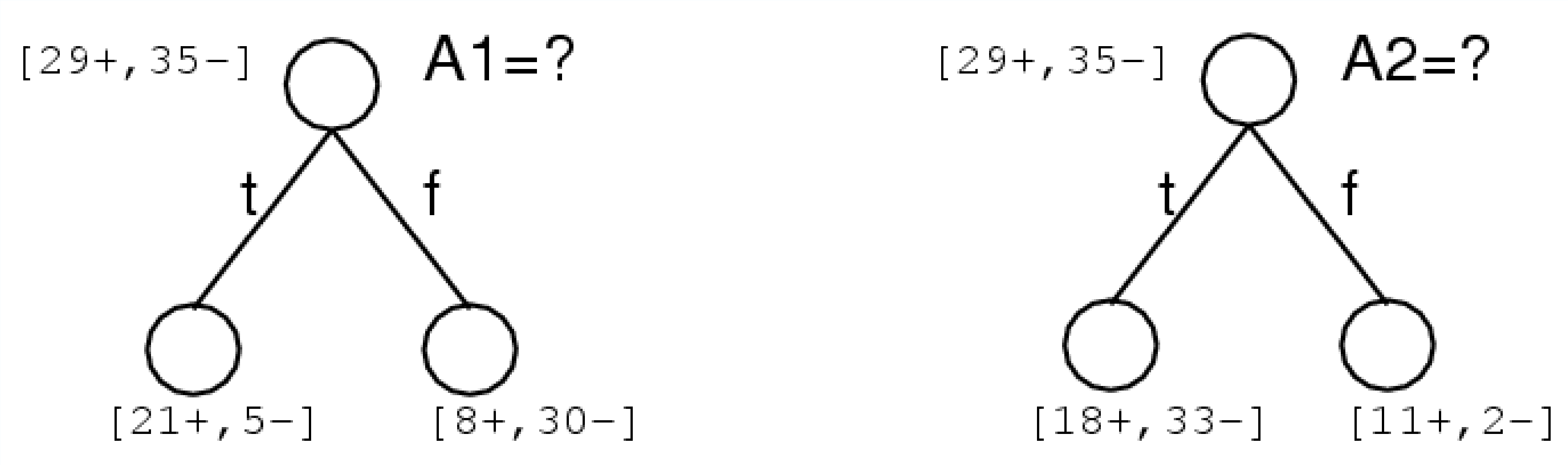
1. A $\leftarrow$ the "best" decision attribute for next node
2. Assign A as decision attribute for node
3. For each value of A, create new descendant of node
4. "Sort" training examples to leaf nodes
5. If training examples perfectly classified, then stop, else iterate over new leaf nodes and apply procedure recursively

Why Greedy Search?

- Early NP completeness results for decision tree construction
- However, there are (older and more recent) dynamic programming approaches to construct all decision under user-defined constraints (cf. constraint-based data mining)

Choosing Attributes: Entropy and Information Gain

Which Attribute is Best?

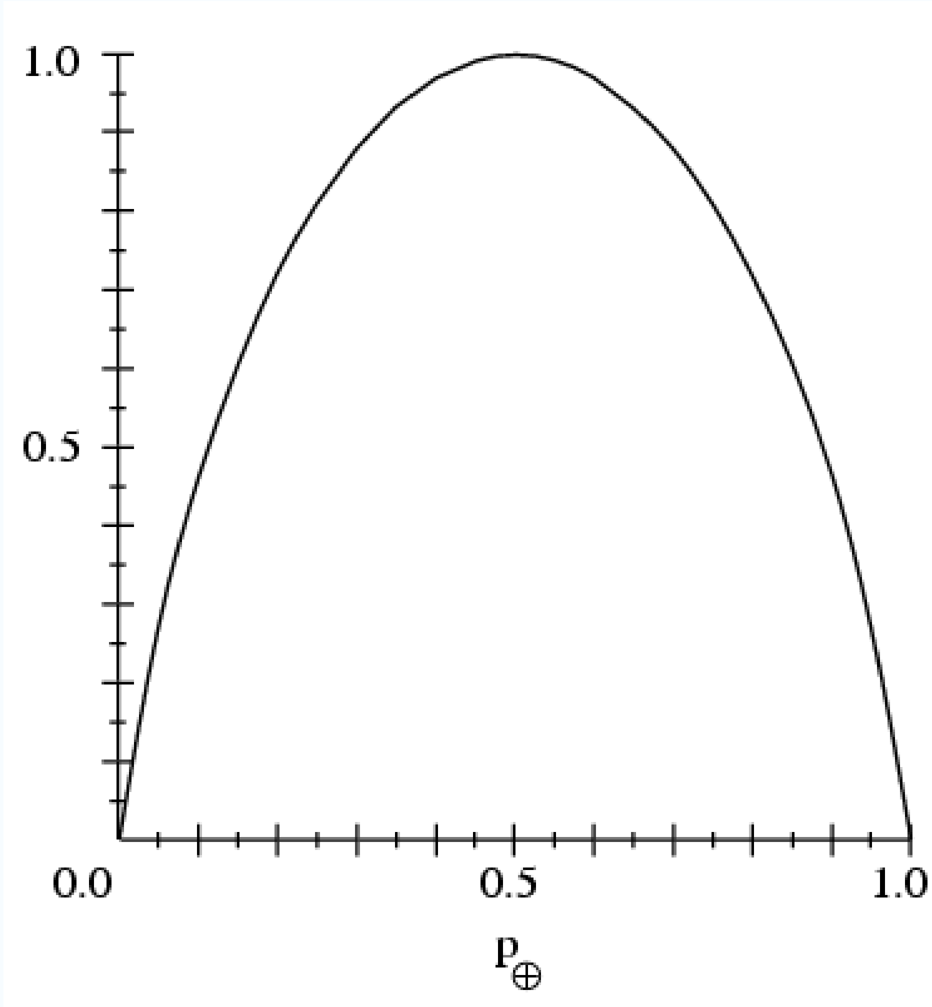


Evaluation of Splits by Information Gain

- Evaluation by so-called *information gain*
- Optimal length code of message of probability p: $-\log_2(p)$
- Expected number of bits needed to encode class (positive or negative) of random member of a set S:
  $\text{Entropy}(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$

Entropy

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



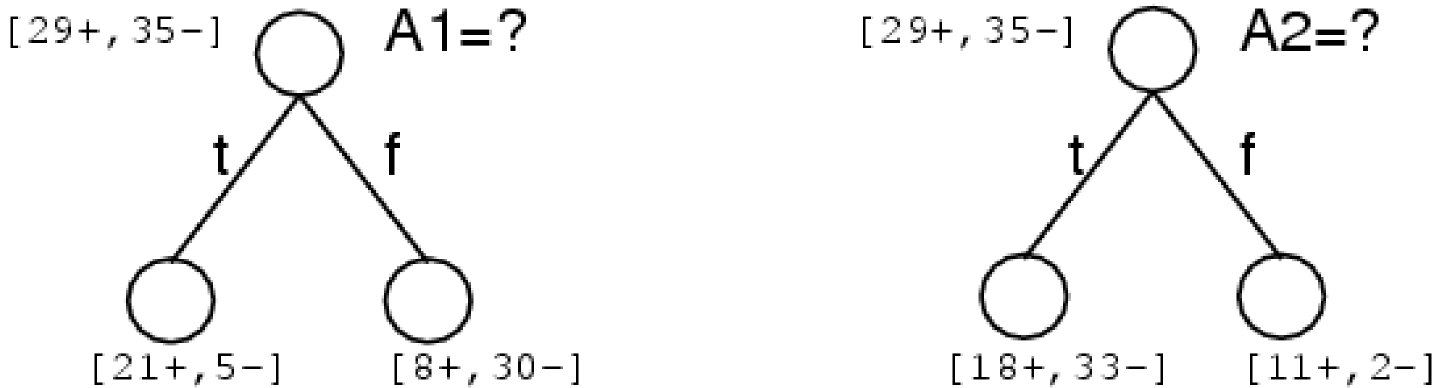
Evaluation of Splits by Information Gain

- Maximum of 1 for $p_+ = p_- = 0.5$
- Minimum of 0 for $p_+ = 1$, $p_- = 0$ or vice versa
- Expected reduction in entropy by splitting up S into S_t and S_f according to literal L
- Gain(S, L) =
  Entropy(S) –
  Entropy(S_t)$|S_t| / |S|$–
  Entropy(S_f)$|S_f| / |S|$

Evaluation of Splits by Information Gain

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

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

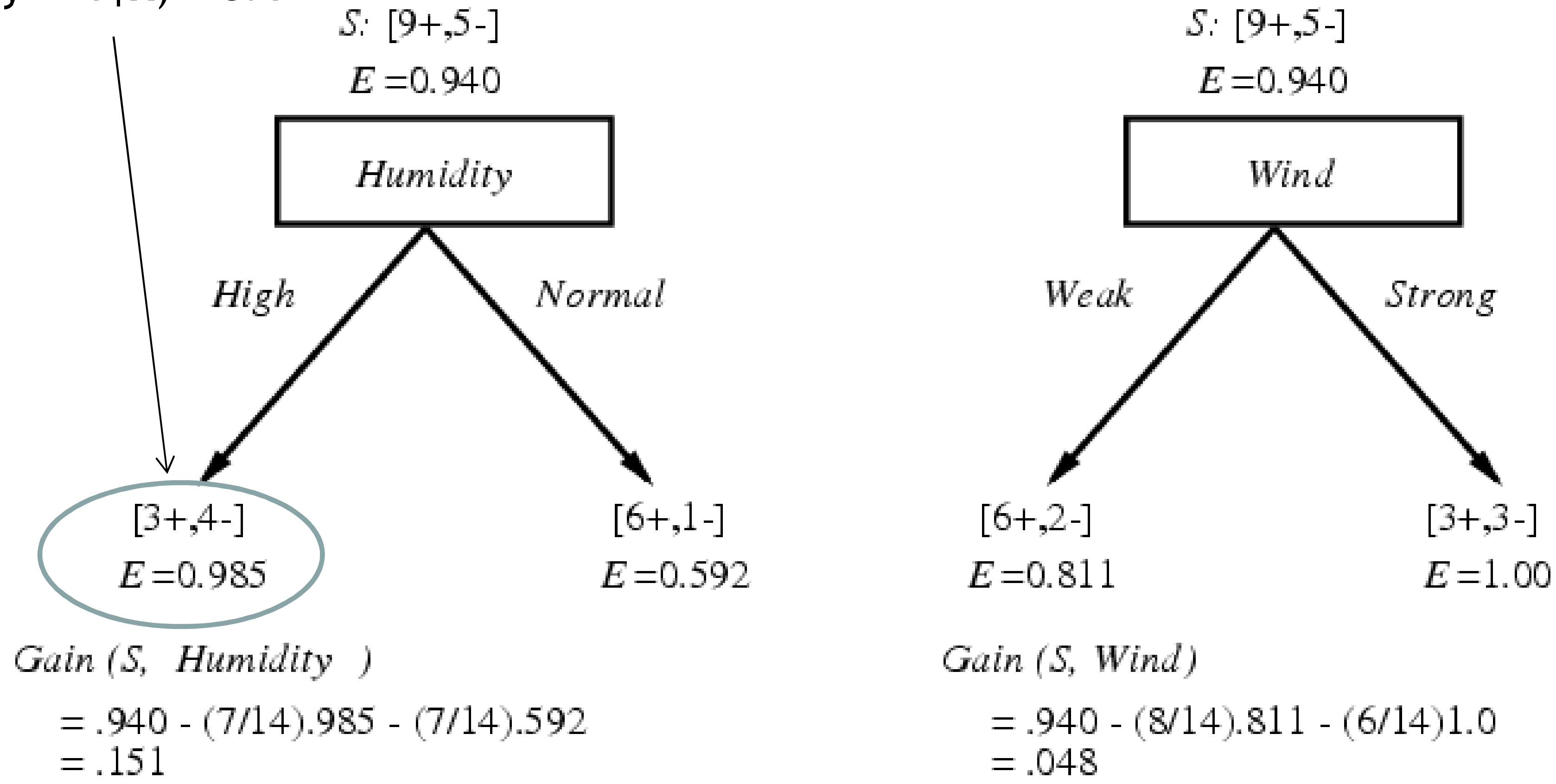


Example Dataset

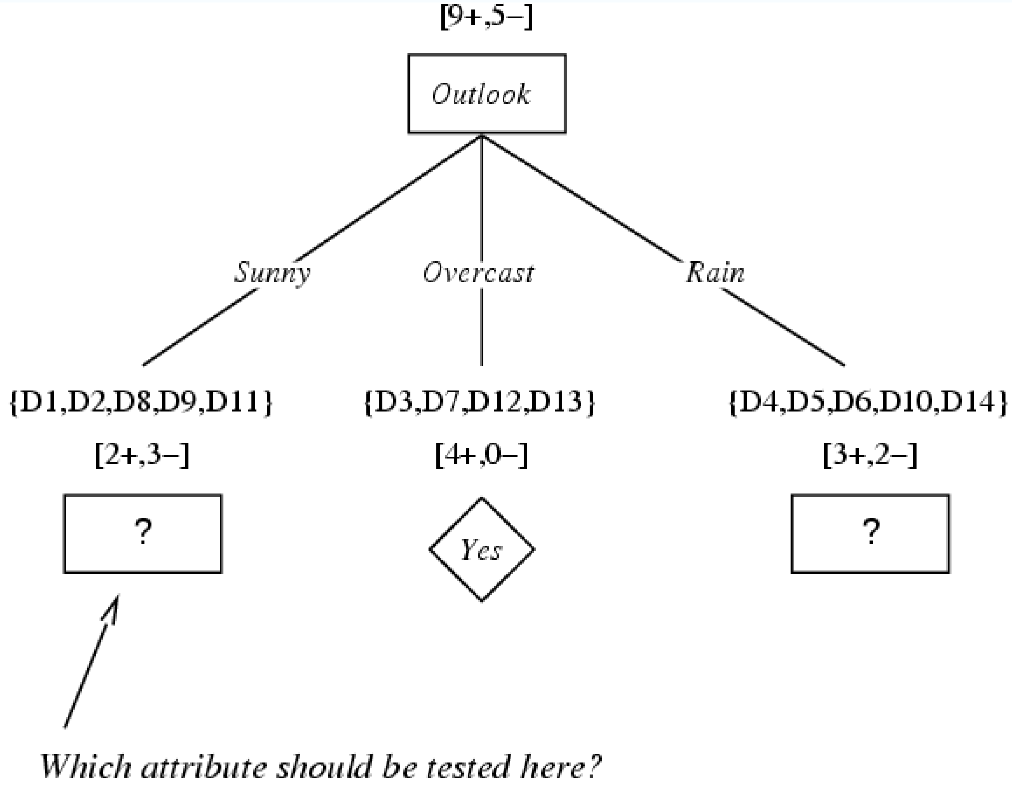
Day	Outlook	Temp.	Hum.	Wind	PlayT.
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example Calculation

Class probability estimates (!):
P(y =+1|**x**) = 3/7



Evaluating the Next Attribute



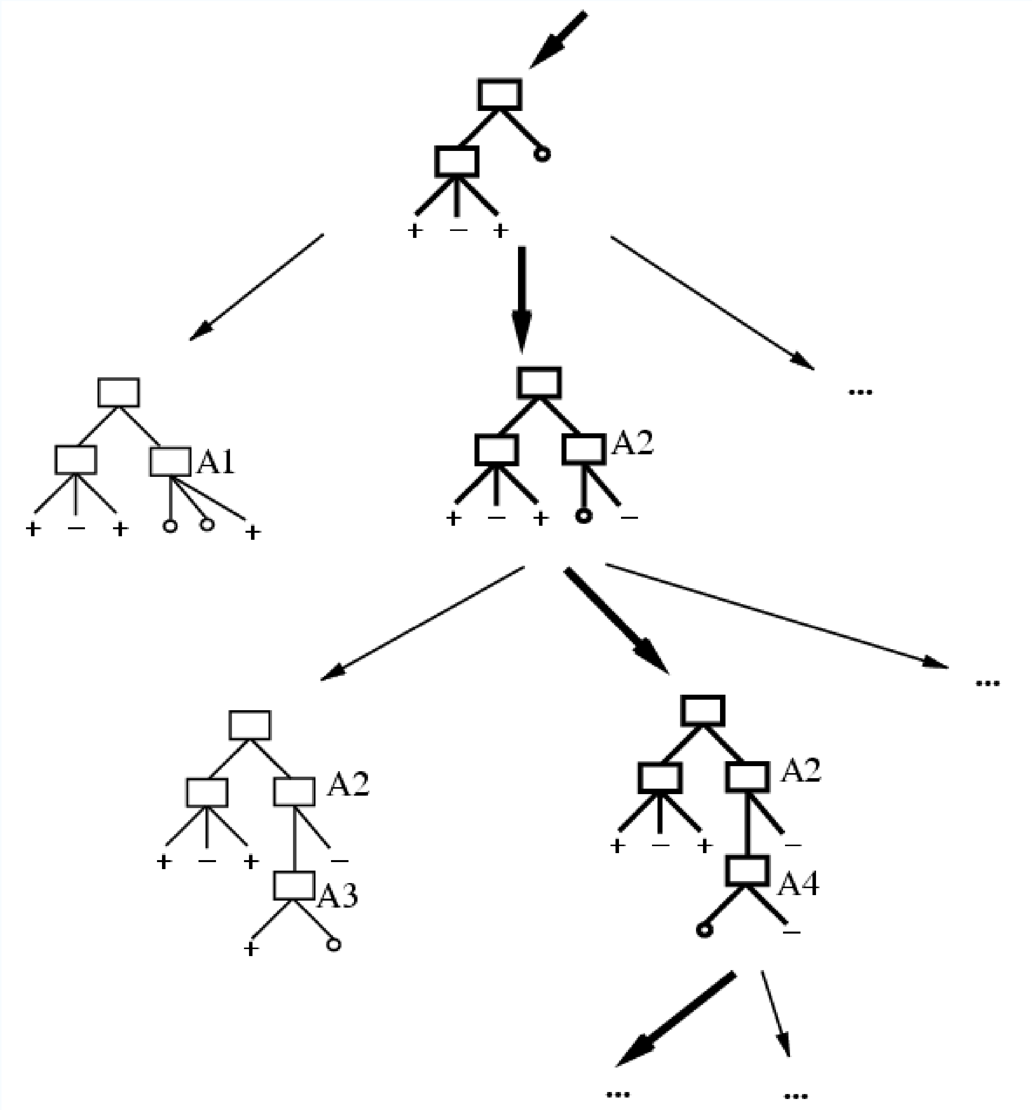
$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Gain\ (S_{sunny}, Humidity) = .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = .970$

$Gain\ (S_{sunny}, Temperature) = .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0$

$Gain\ (S_{sunny}, Wind) = .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

- Hypothesis space is *complete*
  - target function surely contained in it
- Outputs a *single* hypothesis (*which one?*)
- No backtracking (*why?*)
  - local minima
- Statistically-based search choices
  - robust to noisy data
- Inductive bias: *approximately* *"prefer shortest tree"*...

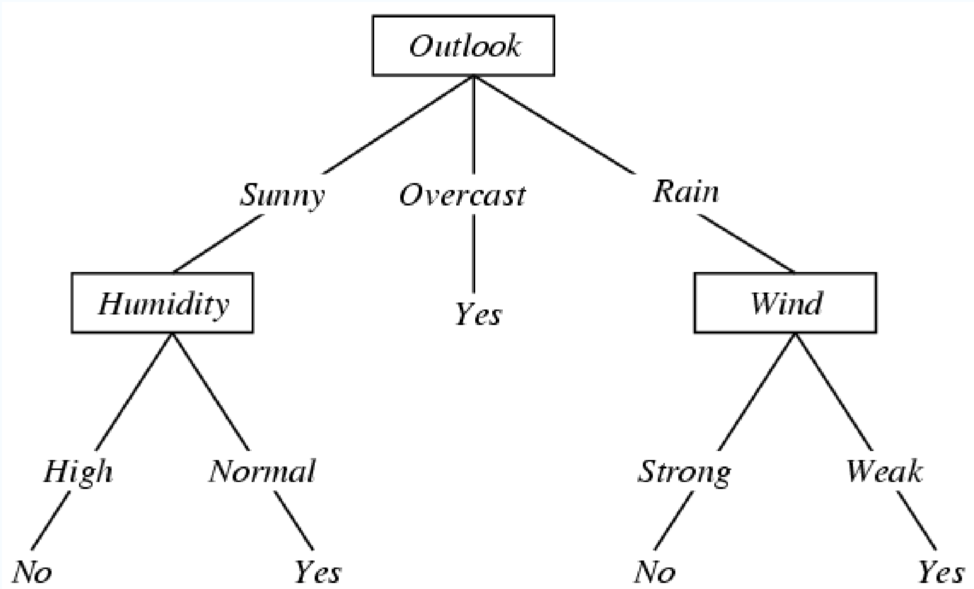
Occam‘s Razor

- Why prefer short hypotheses?
- Arguments in favor:
  - fewer short hypotheses than long hypotheses
  - a *short hypothesis that fits* data unlikely to be *coincidence*
  - a long hypothesis that fits data might be coincidence
- Arguments opposed:
  - there are many ways to define small sets of hypotheses
  - e.g., all trees with a prime number of nodes that use attributes beginning with "Z“
  - *what’s so special about small sets based on size of hypothesis?*

Overfitting Avoidance and Decision Tree Pruning

Overfitting in Decision Trees

- Consider adding noisy (erroneous) training example #15:
  (Sunny, Hot, Normal, Strong, PlayTennis=No)
- What effect on earlier tree?



Overfitting

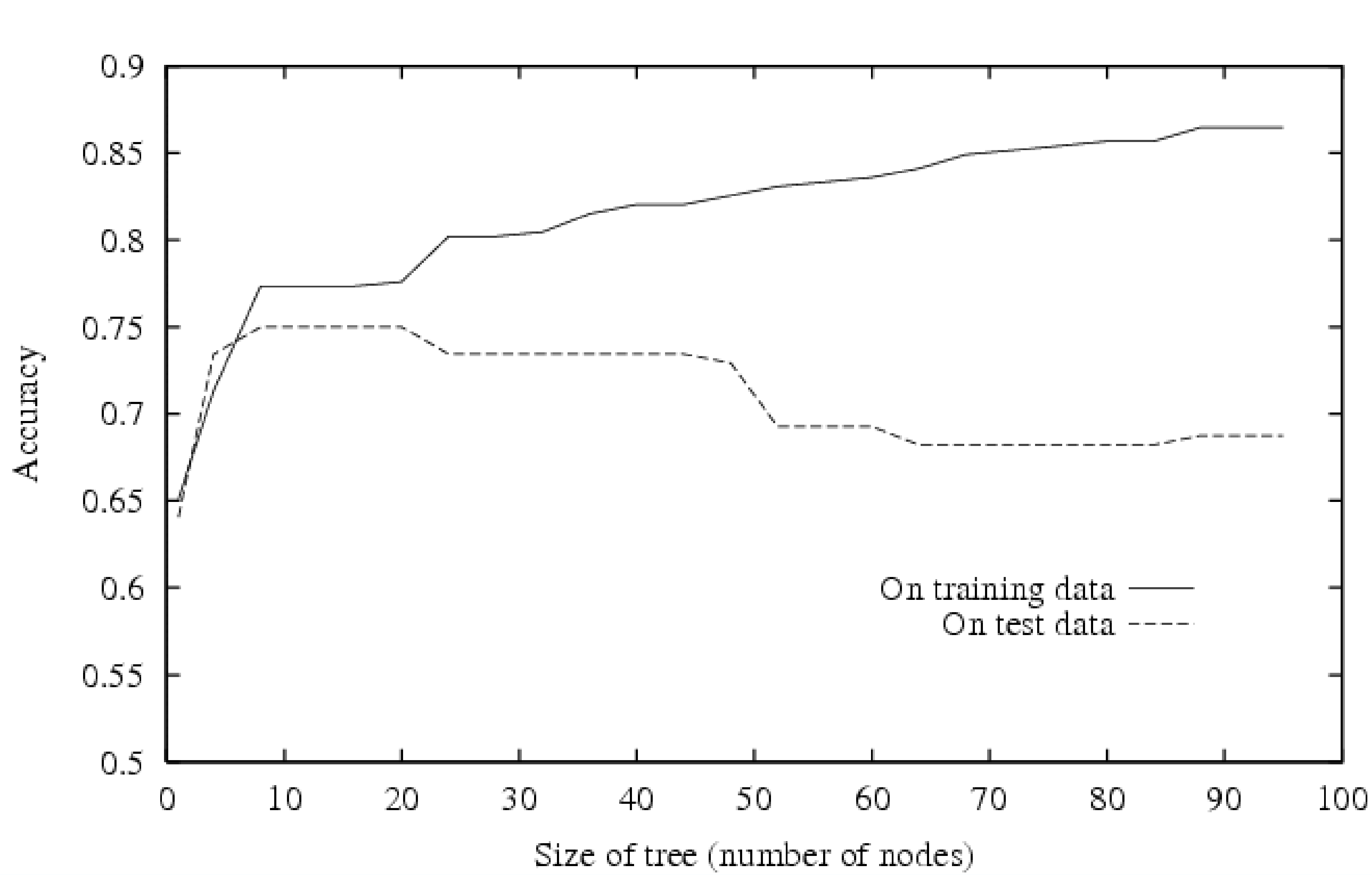
- Consider error of hypothesis h over training data $error_{train}(h)$ and error over entire distribution D of data $error_D(h)$
- Hypothesis h in H *overfits training data* if there exists an alternative hypothesis h´ in H such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_D(h) > error_D(h')$$

Overfitting: Predictive Accuracy on Training and Test Data



Avoiding Overfitting

- How can we avoid overfitting in decision tree induction?
- *Pre-pruning*: *stop growing* when data split not statistically significant
- *Post-pruning*: first grow full tree, then cut it back (prune)

Pruning for Overfitting Avoidance

How to select the best tree: *model selection*

- validation set (or cross-validation)
- Minimum Description Length (MDL) principle
  (information theory: compression of data)
  coding length(tree) + coding length(exceptions)
- …

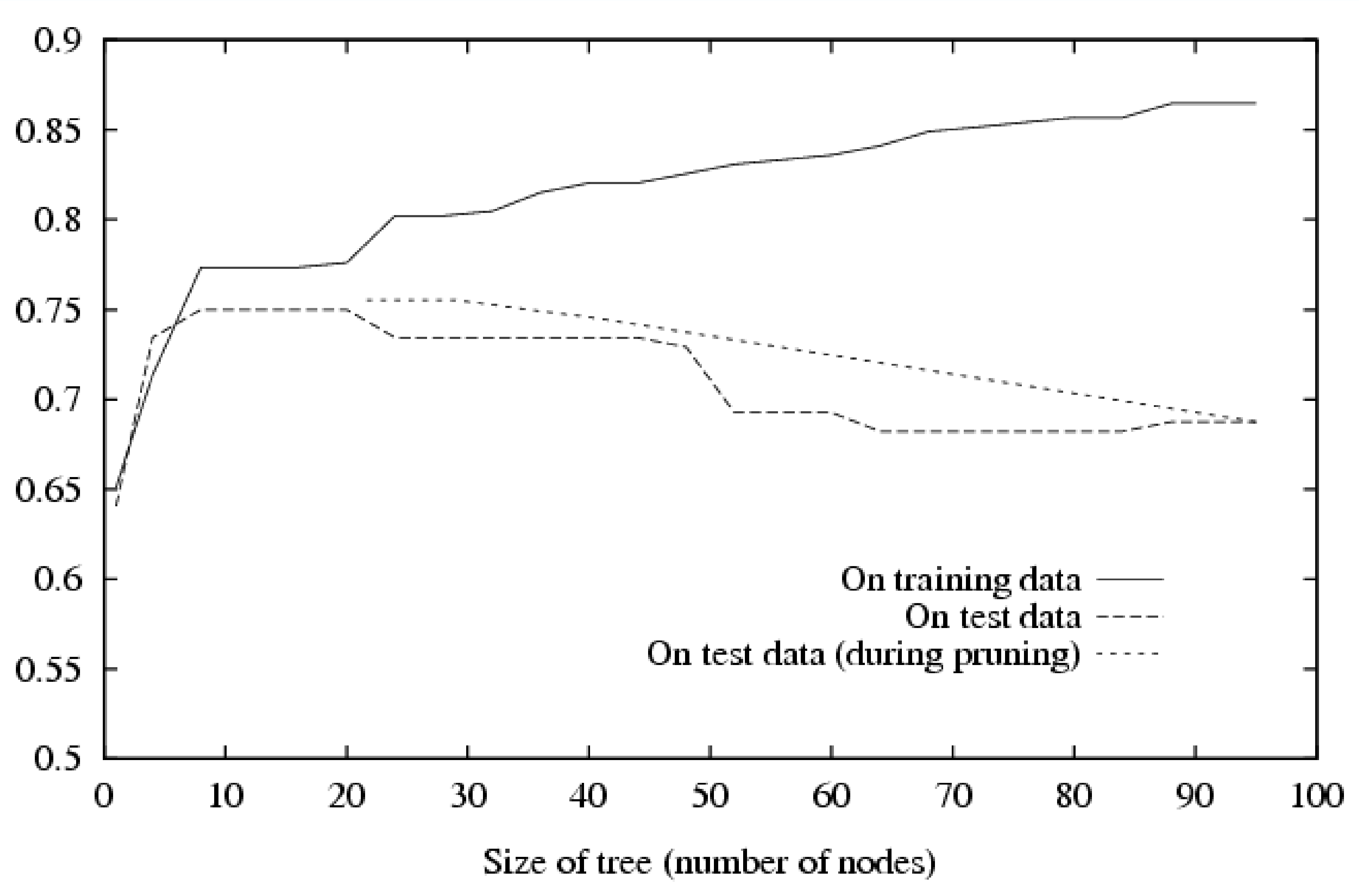
- Popular scheme for post-pruning:
  Reduced error pruning
  greedily remove the one node that reduces the error on the validation set the most

Reduced Error Pruning

- Split data into training and validation set
- Do until further pruning is harmful
  - evaluate impact on validation set of pruning each possible node (plus those below it)
  - greedily remove the one that most improves validation set accuracy
- Produces smallest version of most accurate subtree

What if data is limited?

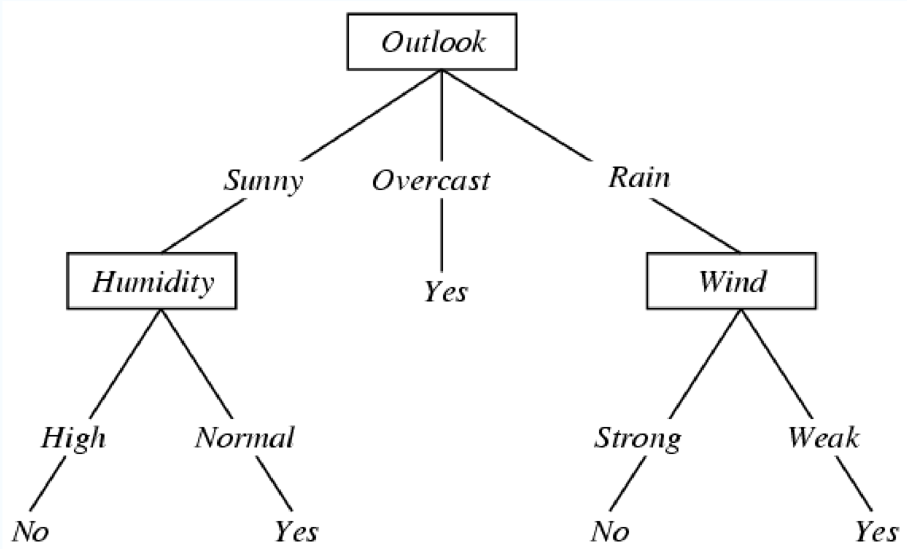
Effect of Reduced-Error Pruning



Rule Post Pruning

- Convert tree to equivalent set of rules
- Prune each rule independently of others
- Sort final rules into desired sequence for use
- One of the most frequently used methods (e.g., C4.5 or See5)

Converting A Tree to Rules



- IF (Outlook=Sunny) $\wedge$ (Humidity=High) THEN PlayTennis=No
- IF (Outlook=Sunny) $\wedge$ (Humidity=Normal) THEN PlayTennis=Yes

Other Issues in Decision Tree Learning

Continuous Valued Attributes

- Create a discrete attribute to test continuous
- Temperature = 82.5
- (Temperature>72.3) = true, false
- Temperature:
  | 40 | 48 | 60 | 72 | 80 | 90
  | No | No | Yes | Yes | Yes | No
  (Play Tennis)
- *Discretization* on-the-fly

Attributes With Many Values

- Problem: if attribute has many values, information gain will select it
- Imagine using attribute “date”
- One approach: use *GainRatio* instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Attributes With Costs

- Consider
  - medical diagnosis, BloodTest has cost 150
  - robotics, Width_from_1ft has cost 23 secs
- How to learn a tree with low expected cost?
- One approach: replace gain by different measures
- (Tan & Schlimmer, 1990): $Gain^2(S, A)/Cost(A)$
- (Nunez, 1988): $(2^{Gain(S,A)} - 1)/(Cost(A)+1)^w$
  where w in $[0,1]$ determines importance of cost

Unknown Attribute Values

- What if some examples having missing values of A?
- Use training example anyway, sort through tree
  - if node n tests A, assign most common value of A among other examples sorted to node n
  - assign most common value of A among other examples with same target value
  - **assign probability p_i to each possible value v_i of A (assign fraction p_i of example to each descendant in tree)**
- Classify new examples in same fashion

Regression Trees and Model Trees

- *Regression trees*: trees for numeric prediction
- Prediction in leaf: mean of instances instead of majority class
- Error measure: mean squared error or the like

```
log_fluence <= -6.01 :
|   log_hr321 <= -0.112 : y := -0.879
|   log_hr321 >  -0.112 : y :=  0.001
log_fluence >  -6.01 :
|   log_hr321 <= 0.0846 : y := 1.83
|   log_hr321 >  0.0846 : y := 1.73
```

Regression Trees and Model Trees

- *Model trees*: trees with (mostly) linear models in leaves
- Prediction in leaf: prediction of (linear) model
- Issue: find good splits w.r.t. the models

```
log_fluence <= -6.01 :
|   log_hr321 <= -0.112 : LM1
|   log_hr321 >  -0.112 : LM2
log_fluence >  -6.01 :
|   log_hr321 <= 0.0846 : LM3
|   log_hr321 >  0.0846 : LM4

LM1:  log_t90 = -0.879 + 0.0353log_hr321 - 0.373log_fluence
                    + 0.0394mfbmfr_class=inter,long + 0.0327mfbmfr_class=long
LM2:  log_t90 = 0.00965 - 0.138log_hr321 - 0.203log_fluence
                    + 0.0394mfbmfr_class=inter,long + 0.0327mfbmfr_class=long
```

Bagging and Random Forests

Bagging

- **B**ootstrap **agg**regat**ing**
- Simplest way of combining predictions: voting/averaging, where each model receives equal weight
- How can we vote if we have only one dataset?

Bootstrap

- Resampling with replacement
- Repeatedly: Given dataset of size n, sample new dataset of size n by drawing with replacement (also called 0.632 bootstrap)
- A particular instance has a probability of 1-1/n of not being picked
- Thus its probability of ending up in the test data is:
$$\left(1-\frac{1}{n}\right)^n \approx e^{-1} = 0.368$$
- Thus, a bootstrap sample will contain approx. 63.2% of the instances

More on Bagging

- Bagging reduces the variance component of the expected error by voting/averaging
- Improves performance almost always if the base classifier is *unstable* and the data are *noisy*

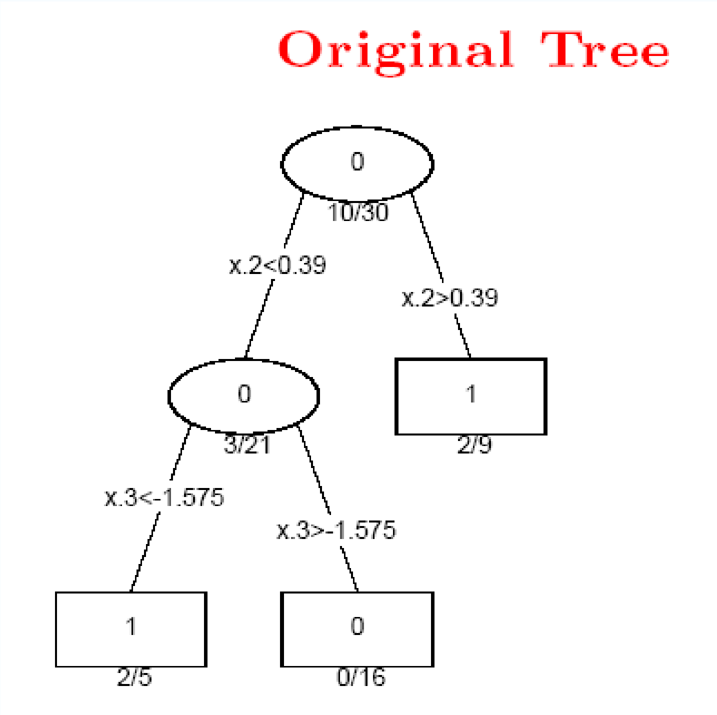
Bagging Classifiers

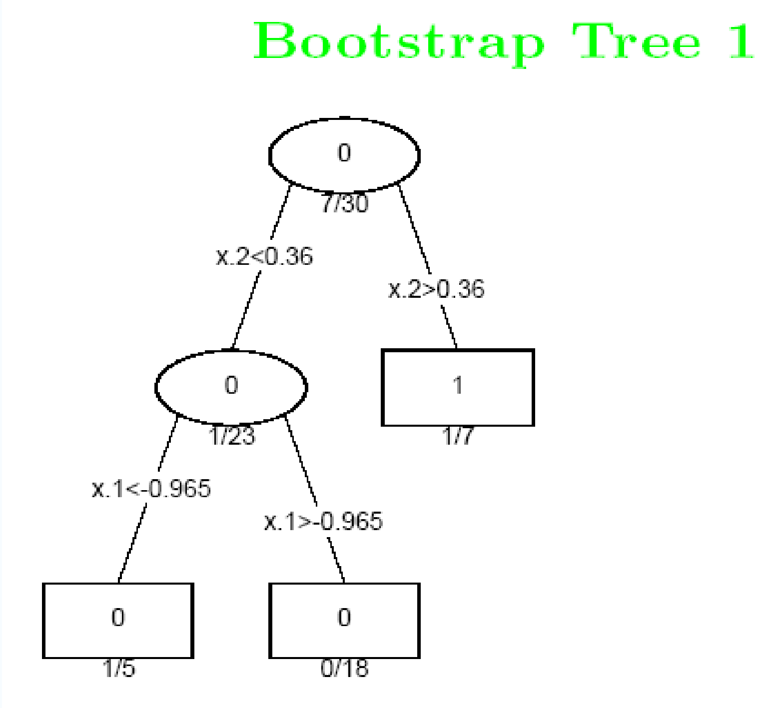
```
model generation
Let n be the number of instances in the training data.
For each of t iterations:
  Sample n instances with replacement from training set.
  Apply the learning algorithm to the sample.
  Store the resulting model.

classification
For each of the t models:
  Predict class of instance using model.
Return class that has been predicted most often.
```

Example

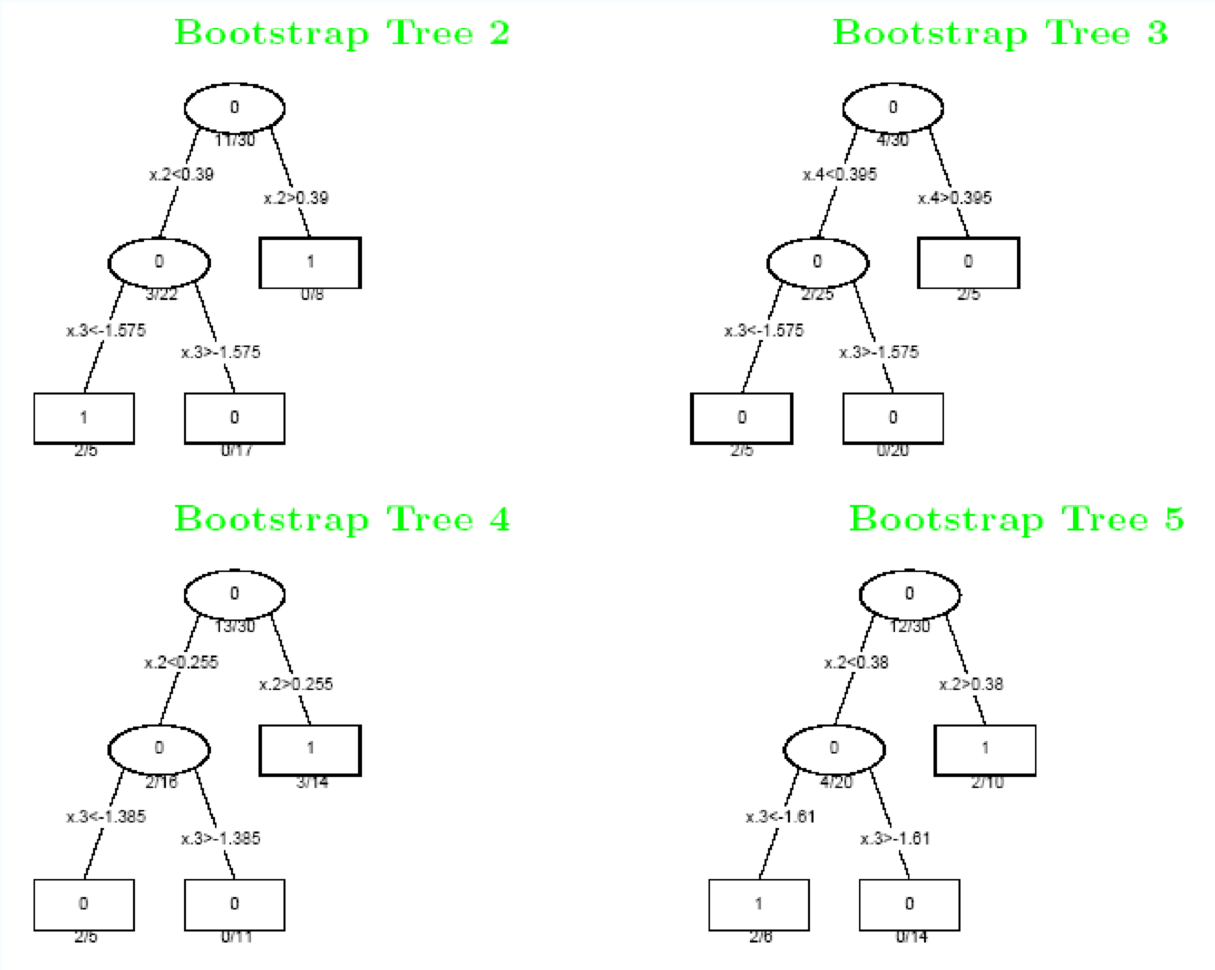
Tree with simulated data





Elements of Statistical Learning (c) Hastie, Tibshirani & Friedman 2001

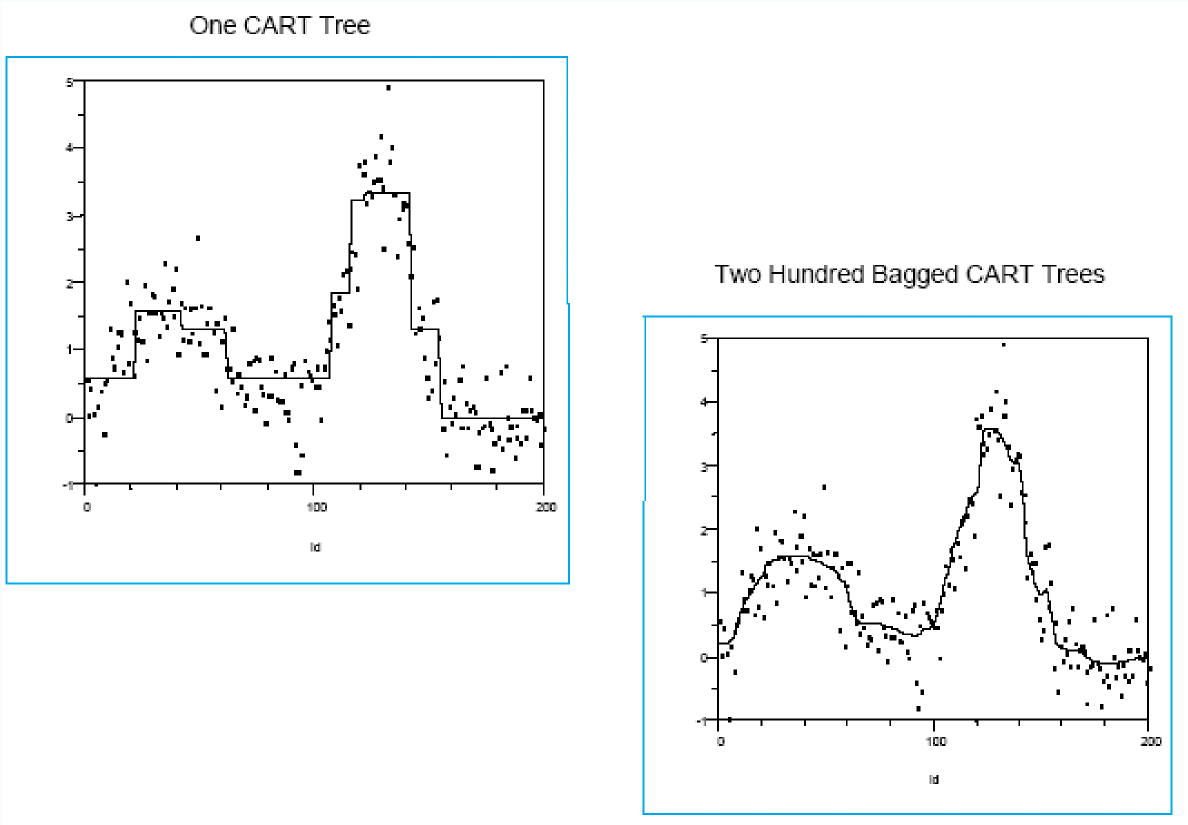
Example



Bootstrap Tree 2
0
11/30
x.2<0.39
x.2>0.39
0
3/22
1
0/8
x.3<-1.575
x.3>-1.575
1
2/5
0
0/17
Bootstrap Tree 3
0
4/30
x.4<0.395
x.4>0.395
0
2/25
0
2/5
x.3<-1.575
x.3>-1.575
0
2/5
0
0/20
Bootstrap Tree 4
0
13/30
x.2<0.255
x.2>0.255
0
2/16
1
3/14
x.3<-1.385
x.3>-1.385
0
2/5
0
0/11
Bootstrap Tree 5
0
12/30
x.2<0.38
x.2>0.38
0
4/20
1
2/10
x.3<-1.61
x.3>-1.61
1
2/6
0
0/14

Example

CART (Classification And Regression Trees)



Rick Higgs & Dave Cummins, Statistical & Information Sciences, Lilly Research Laboratories, 2003