Decision Tree

Jayanta Mukhopadhyay
Dept. of Computer Science and Engg.

Courtesy: Prof. Pabitra Mitra, CSE, IIT Kharagpur

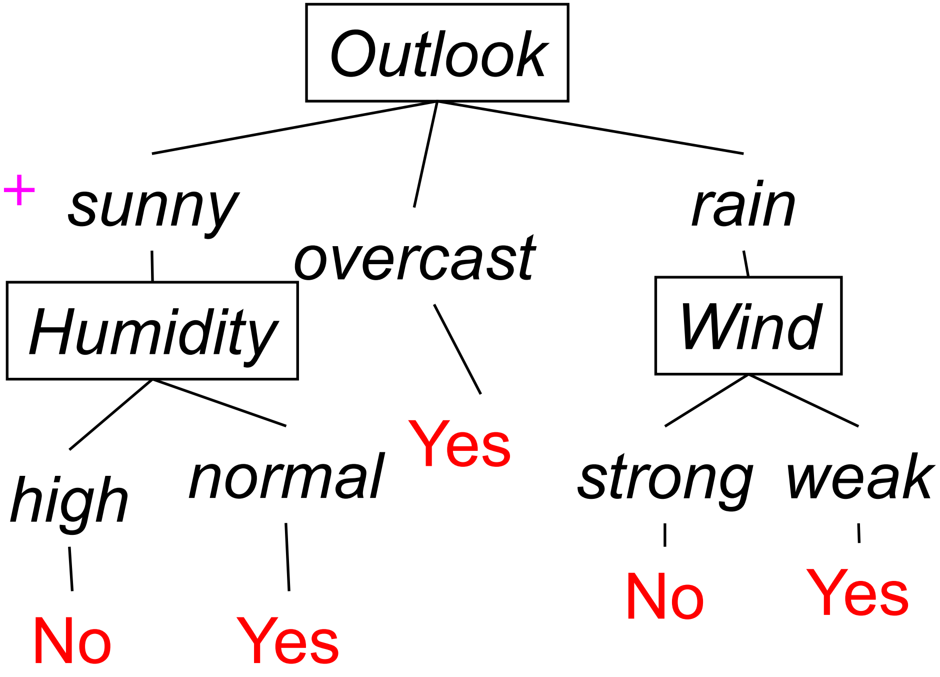
Books

- Chapter 3 of “Machine learning” by Tom M. Mitchel

Representation of Concepts

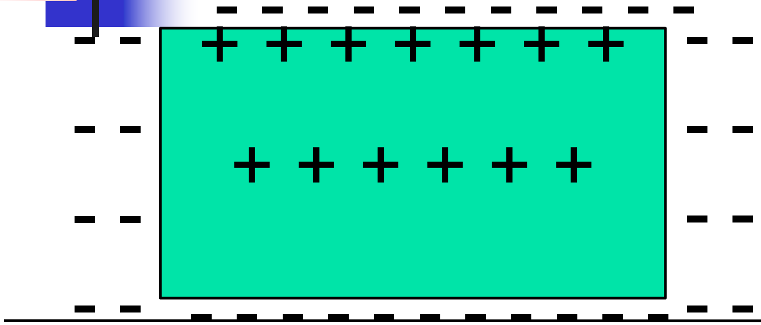
- Concept learning: conjunction of attributes
  - (Sunny & Hot & Humid & Windy) +

- Decision trees: disjunction of conjunction of attributes
  - (Sunny & Normal) | (Overcast) | (Rain & Weak) +
  - More powerful representation
  - Larger hypothesis space H
  - Can be represented as a tree
  - Common form of decision making in humans

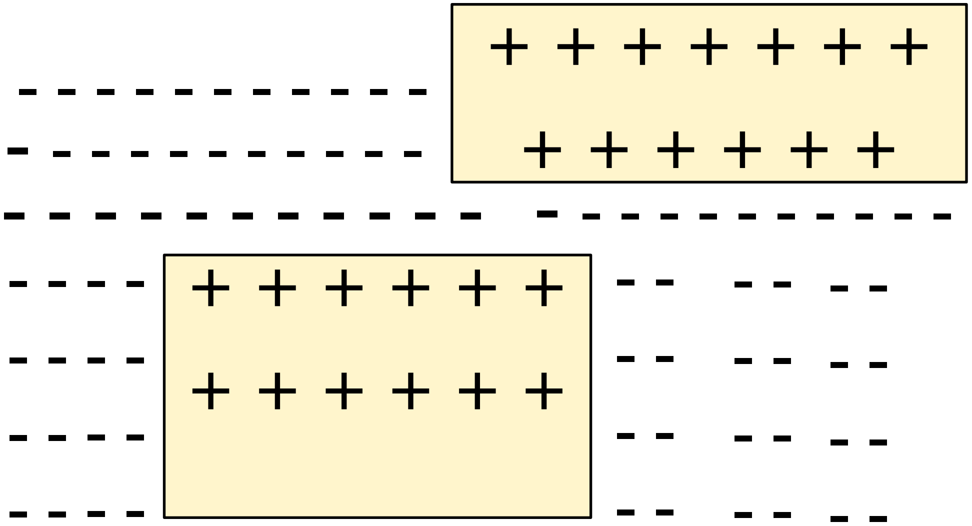


OutdoorSport

Rectangle learning….



Conjunctions
(single rectangle)



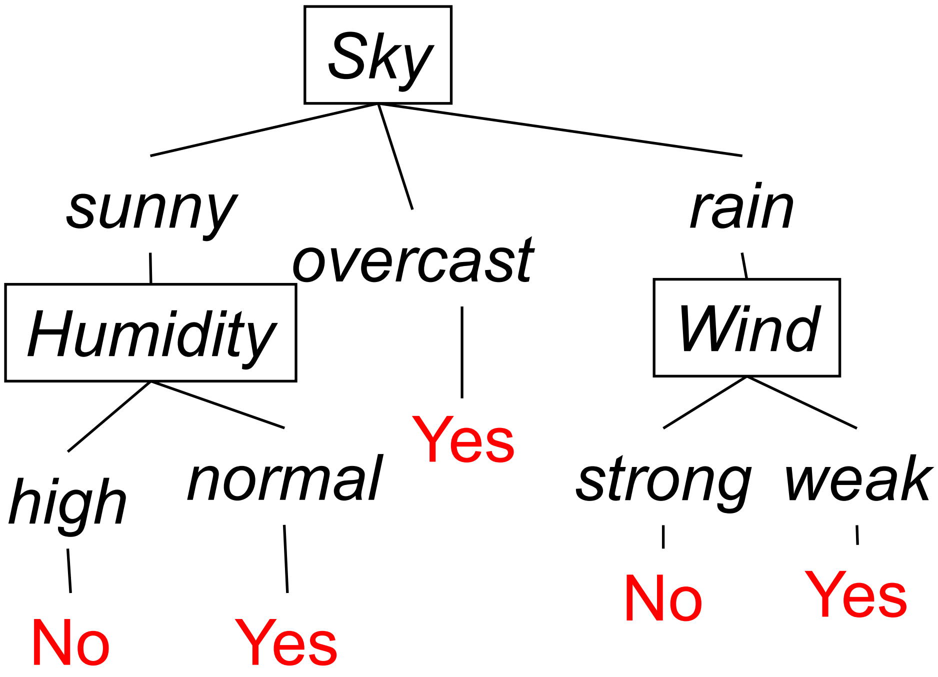
Disjunctions of Conjunctions
(union of rectangles)

Training Examples

Day	Outlook	Temp	Humidity	Wind	Tennis?
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*

Decision Trees

- Decision tree to represent learned target functions.
  - Each internal node tests an attribute
  - Each branch corresponds to attribute value
  - Each leaf node assigns a classification



- Can be represented by logical formulas.

<sunny,?, normal,?> | <overcast, ?,?,?> | <rain,?,?,weak>

Representation of rules in decision trees

- Example of representing rule in DT’ s:

  if outlook = sunny AND humidity = normal

  OR

  if outlook = overcast

  OR

  if outlook = rain AND wind = weak

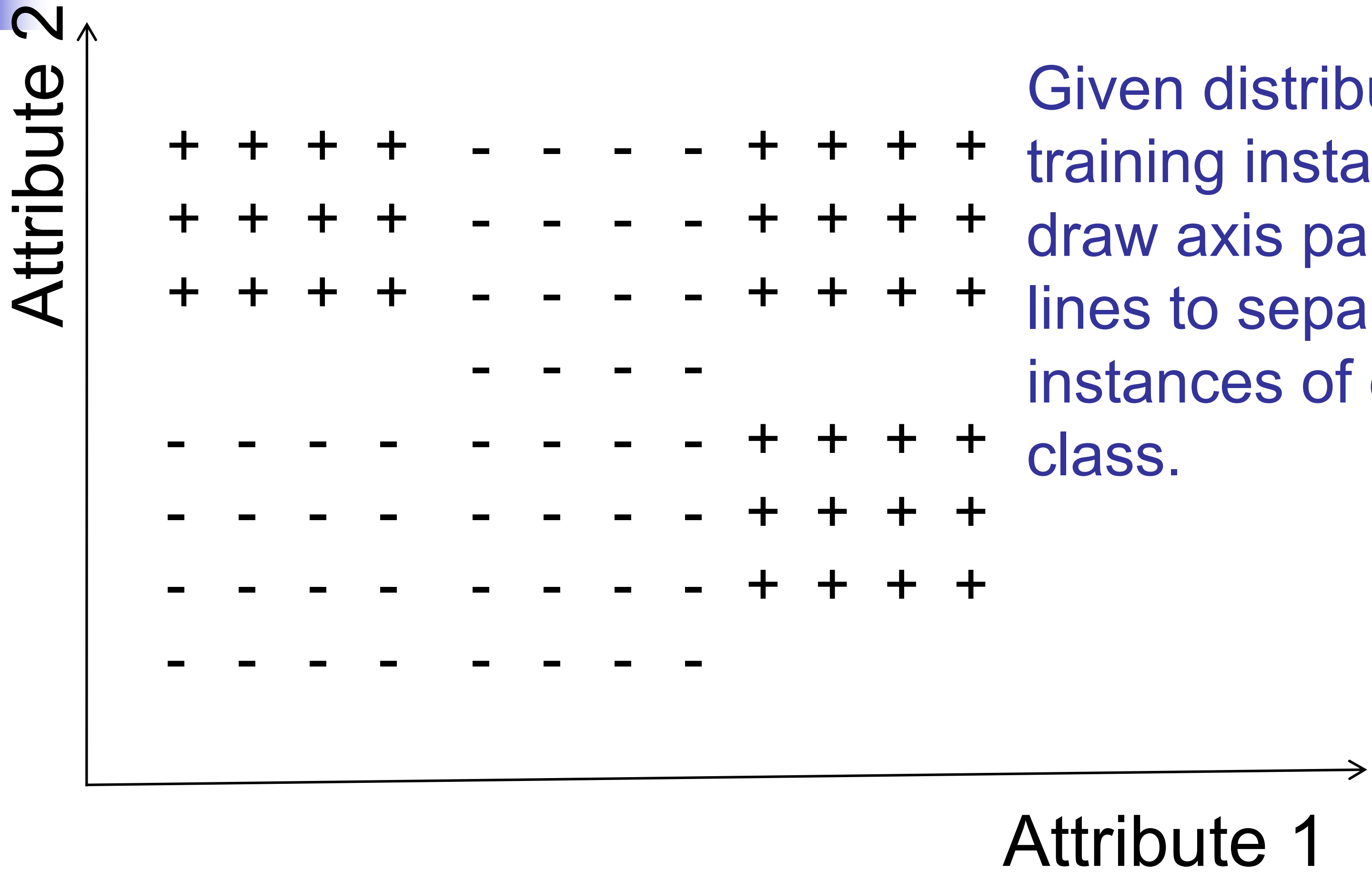
  then playtennis

Applications of Decision Trees

- Instances describable by a fixed set of attributes and their values
- Target function is discrete valued
  - 2-valued
  - N-valued
  - But can approximate continuous functions
- Disjunctive hypothesis space
- Possibly noisy training data
  - Errors, missing values, …

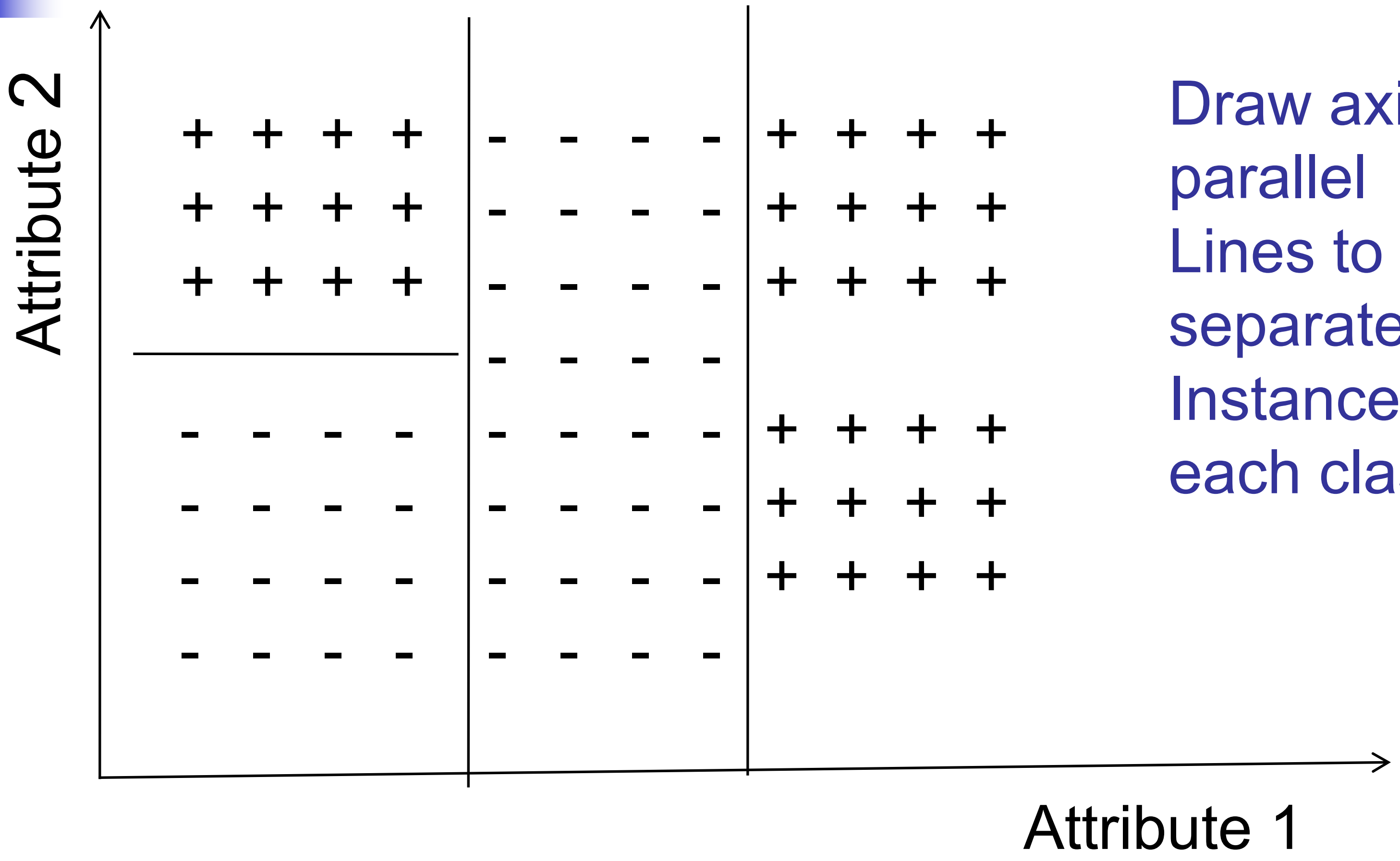
- Examples:
  - Equipment or medical diagnosis
  - Credit risk analysis
  - Calendar scheduling preferences

Decision Trees



Given distribution of training instances draw axis parallel lines to separate the instances of each class.

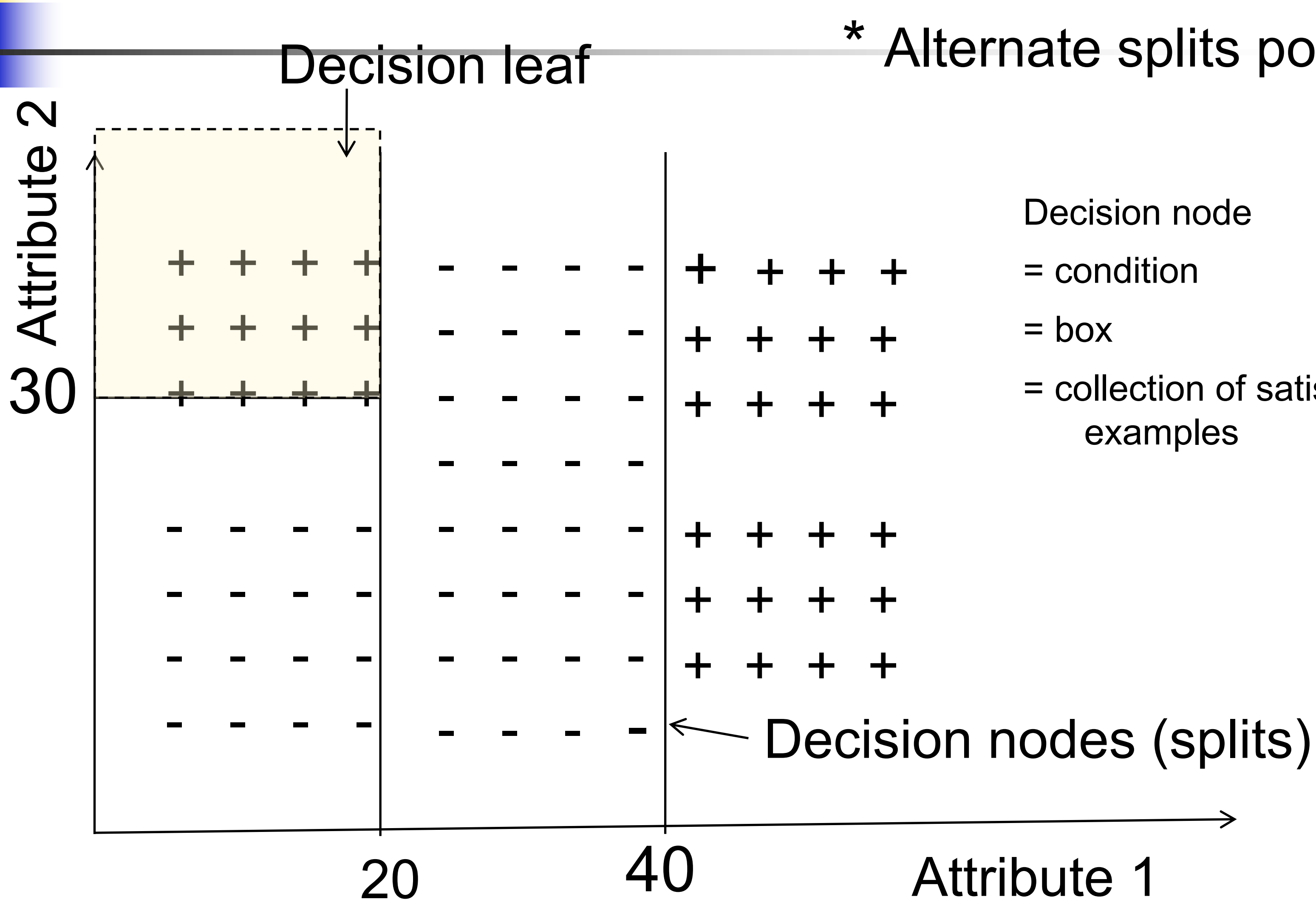
Decision Tree Structure



Draw axis parallel Lines to separate the Instances of each class

Decision Tree Structure

* Alternate splits possible



Decision node

= condition

= box

= collection of satisfying examples

Decision Tree Construction

- Find the best structure
- Given a training data set

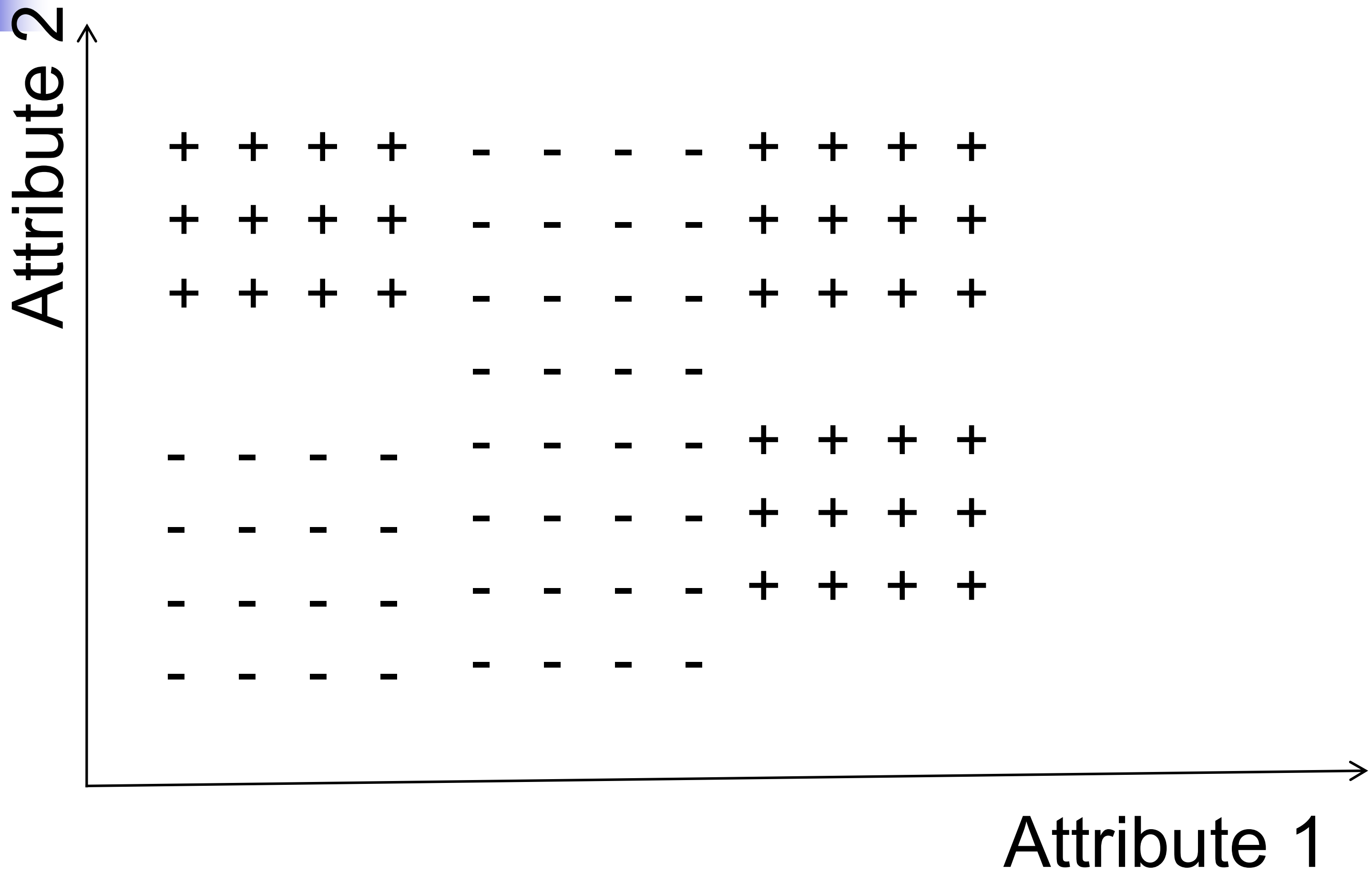
Top-Down Construction

- Start with an empty tree
- Main loop:

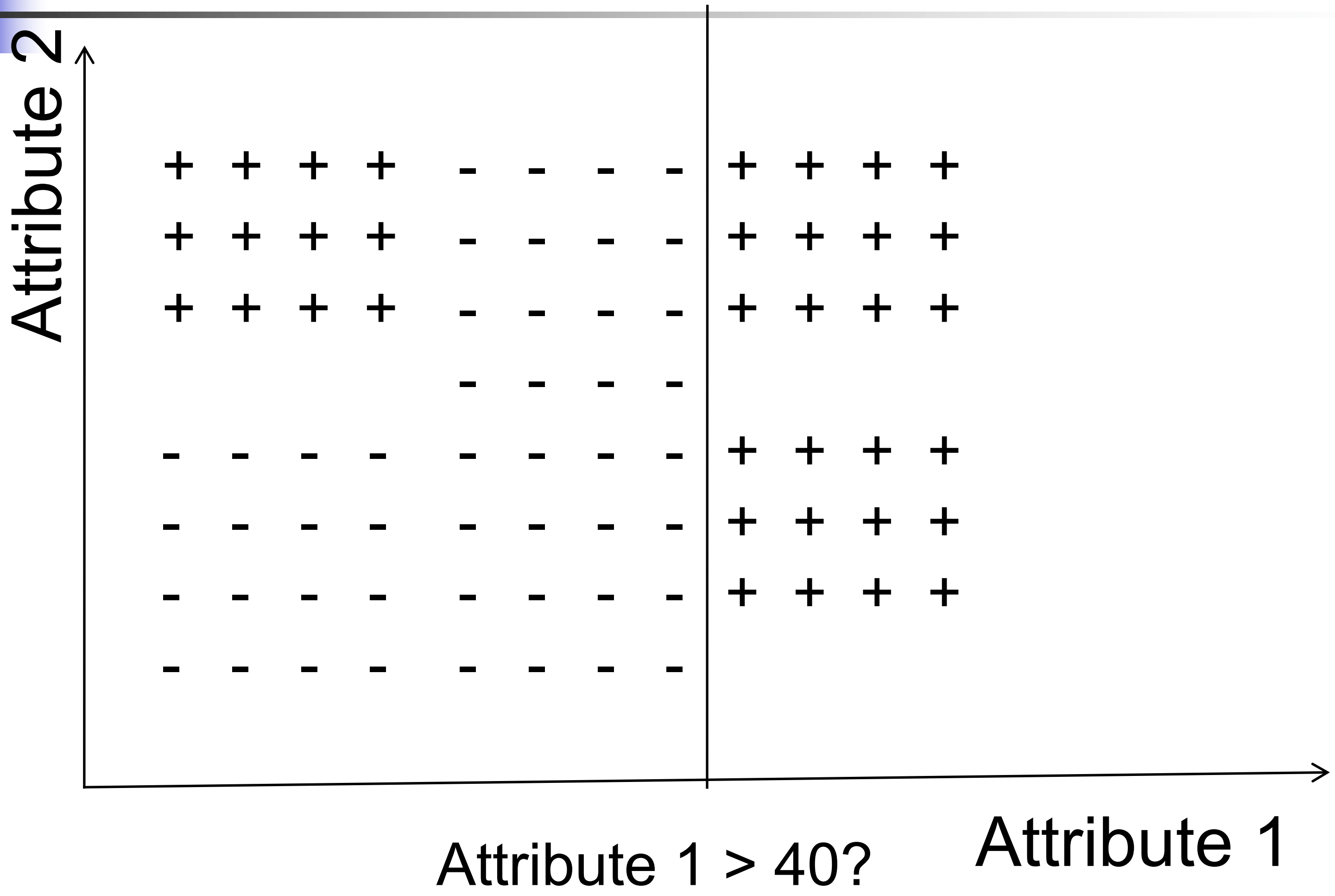
  1. Split the "best" decision attribute (*A*) for next node
  2. Assign *A* as decision attribute for node
  3. For each value of *A*, create new descendant of node
  4. Sort training examples to leaf nodes
  5. If training examples perfectly classified, STOP,
     Else iterate over new leaf nodes

- Grow tree just deep enough for perfect classification
  - If possible (or can approximate at chosen depth)
- Which attribute is the best?

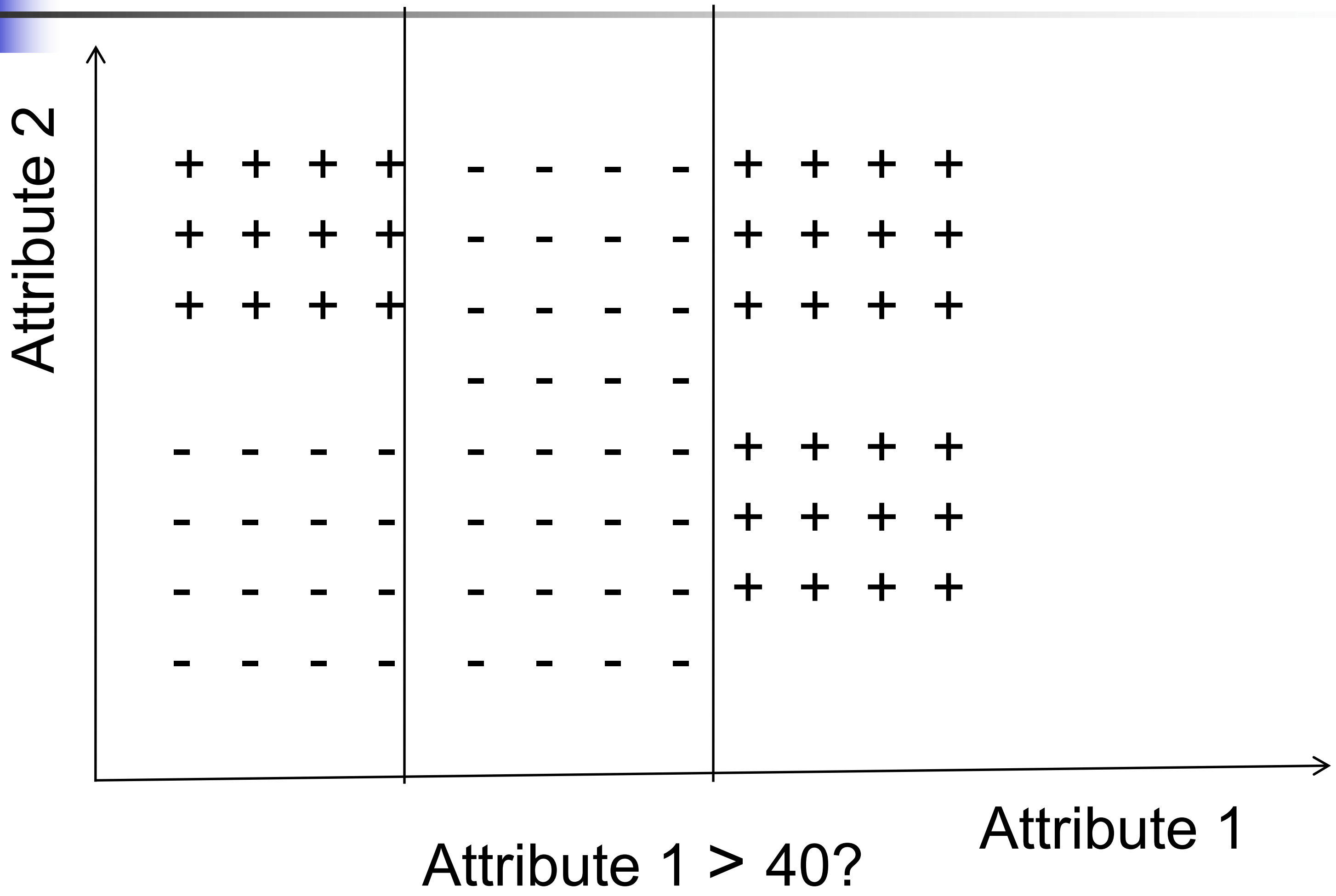
Best attribute to split?



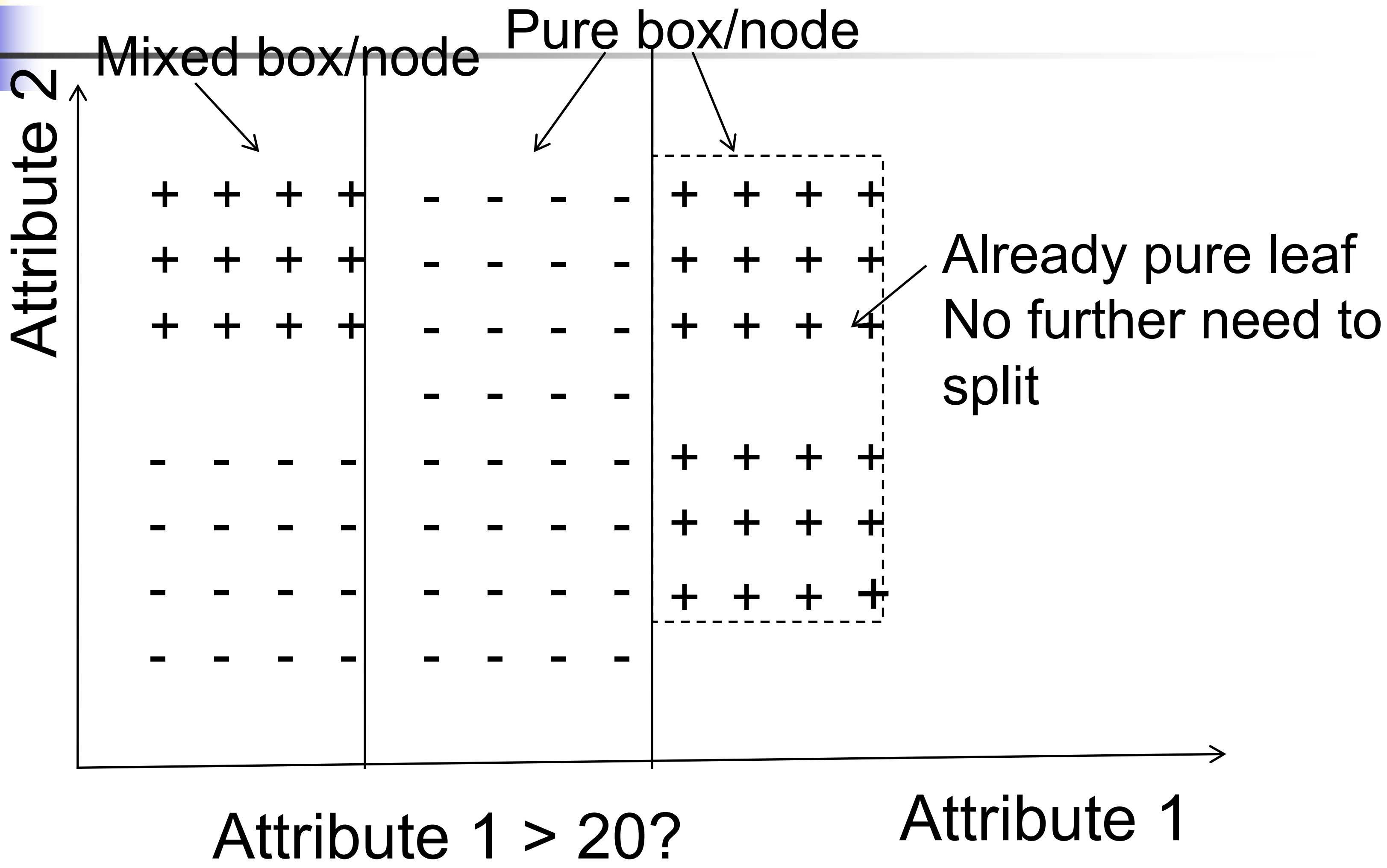
Best attribute to split?



Best attribute to split?



Which split to make next?

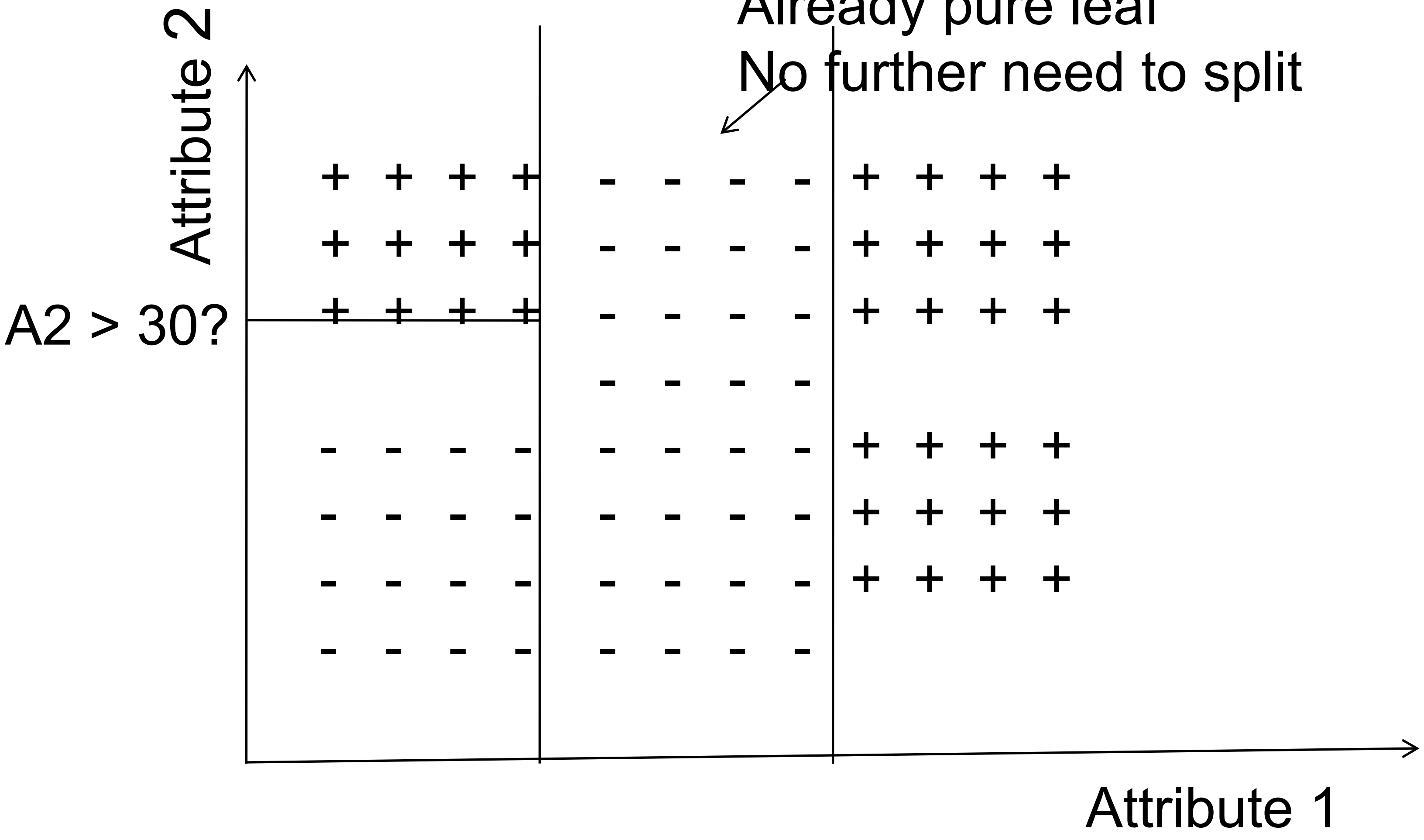


Mixed box/node

Pure box/node

Already pure leaf
No further need to split

Which split to make next?



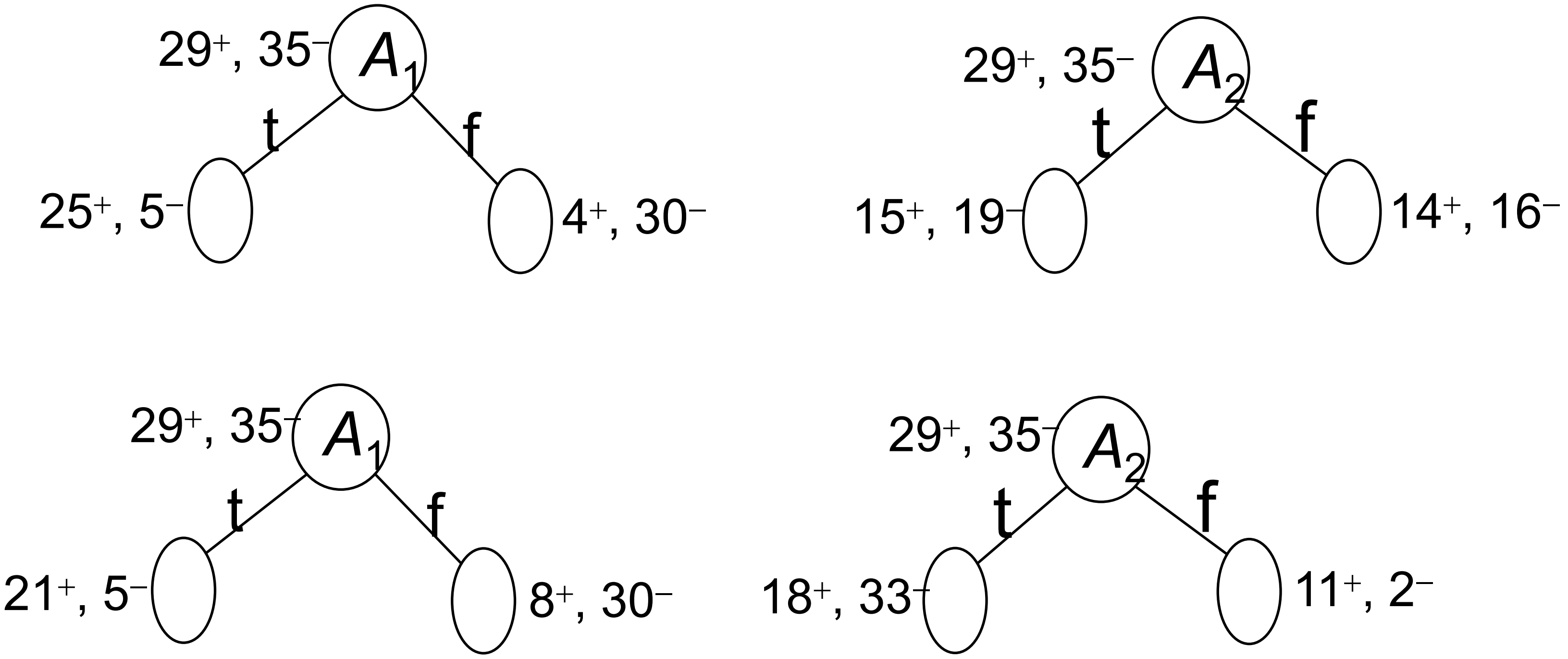
Already pure leaf
No further need to split

Principle of Decision Tree Construction

- Try to form a tree with pure leaves
  - Correct classification
- A greedy approach
  1. Initially treat the entire data set as a single box.
  2. For each box choose the spilt with an attribute that reduces its impurity (in terms of class labels) by the maximum amount.
  3. Split the box having highest reduction in impurity
  4. Continue to Step 2.
  5. Stop when all boxes are pure.

Choosing the Best Attribute?

- Consider 64 examples, 29^+ and 35^-
- Which one is better?



Entropy

- A measure for uncertainty, purity and information content.
  - optimal length code assigns $(-\log_2 p)$ bits to message having probability p
- Let S be a sample of training examples
  - p_+ : the proportion of positive examples in S
  - p_- : the proportion of negative examples in S
- Entropy of S:
  - average optimal number of bits to encode information about certainty/uncertainty about S

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

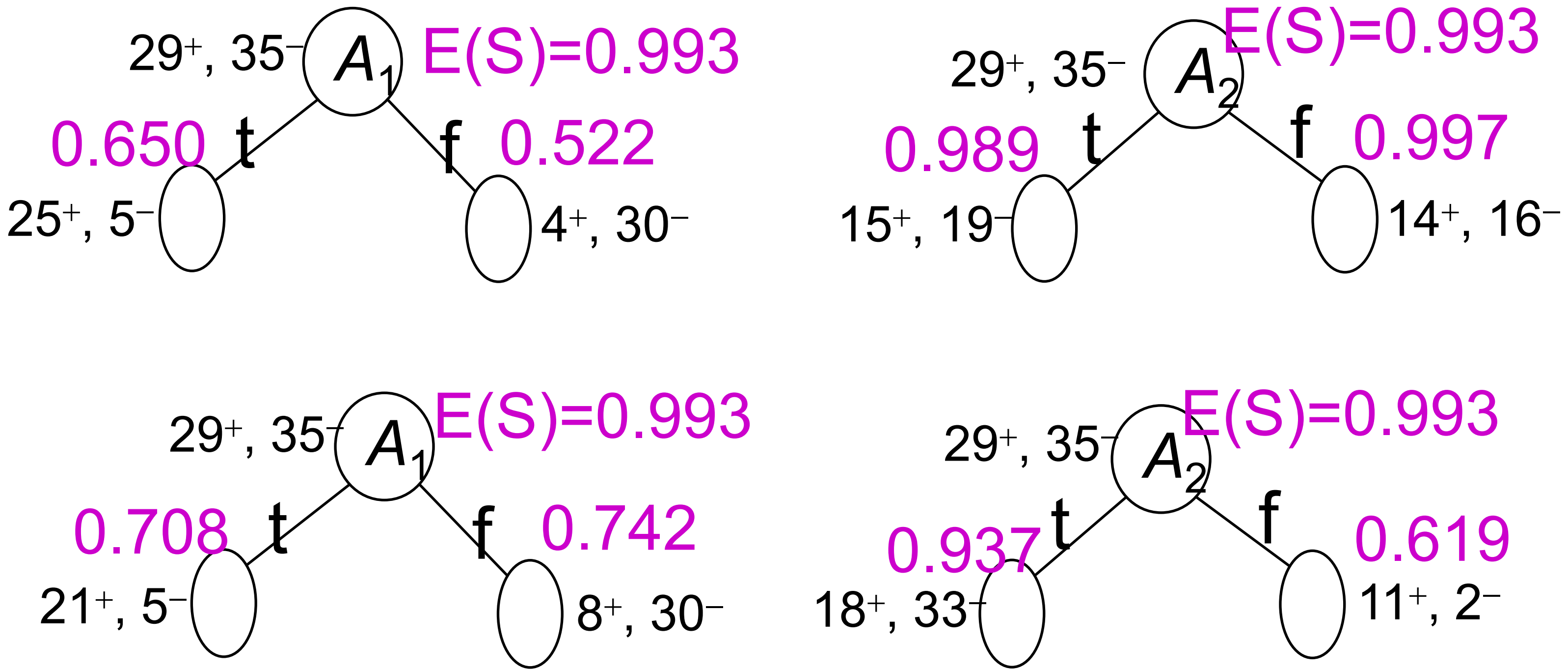
- Can be generalized to more than two values

Entropy

- Entropy can also be viewed as measuring
  - purity of S,
  - uncertainty in S,
  - information in S, …
- E.g.: values of entropy for p+=1, p+=0, p+=.5
- Easy generalization to more than binary values
  - Sum over p_i *(-$\log_2 p_i$) , i=1,n
    - ❖i is + or – for binary
    - ❖i varies from 1 to n in the general case

Choosing Best Attribute?

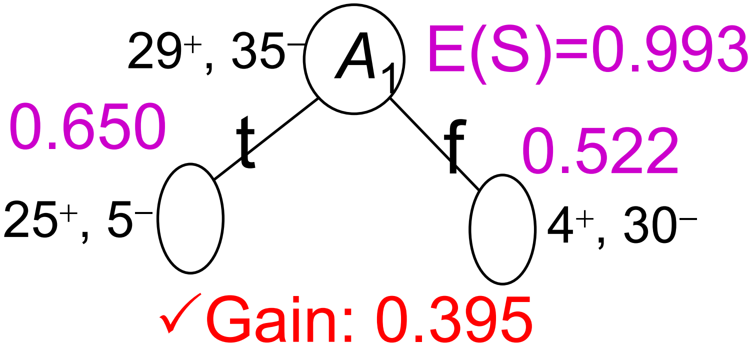
- Consider 64 examples ($29^+,35^-$) and compute entropies:
- Which one is better?

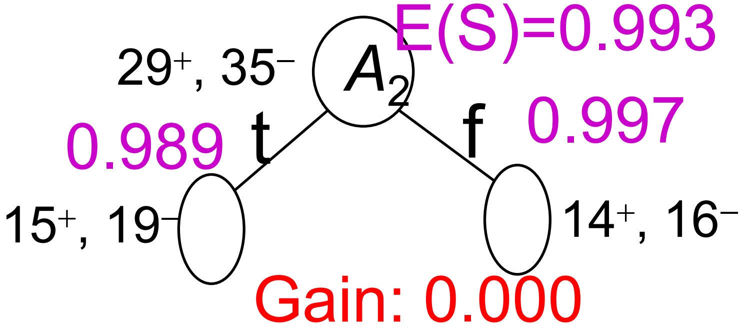


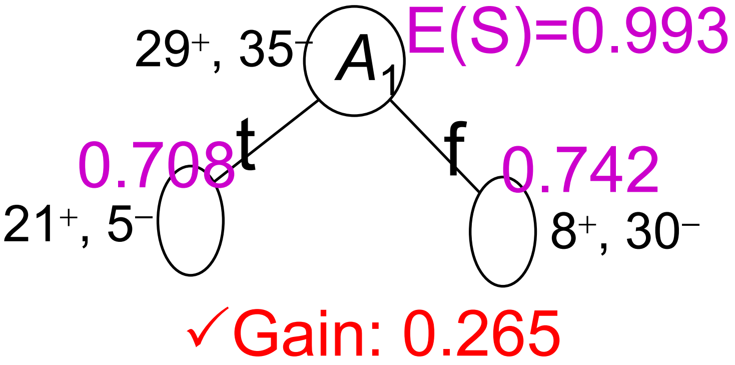
Information Gain

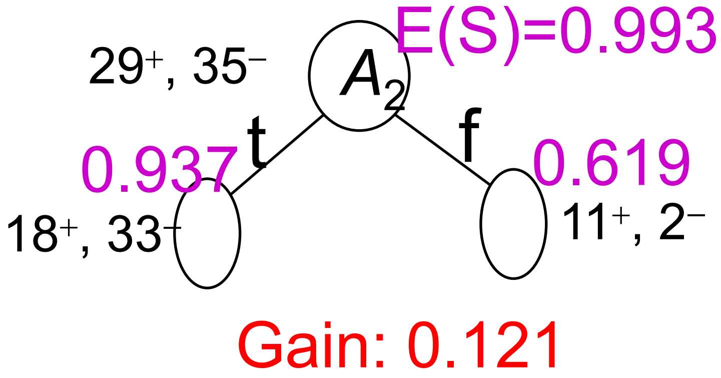
- *Gain(S,A)*: reduction in entropy after choosing attr. *A*.

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









Gain function

- A measure of reduction of uncertainty
  - Value lies between 0,1

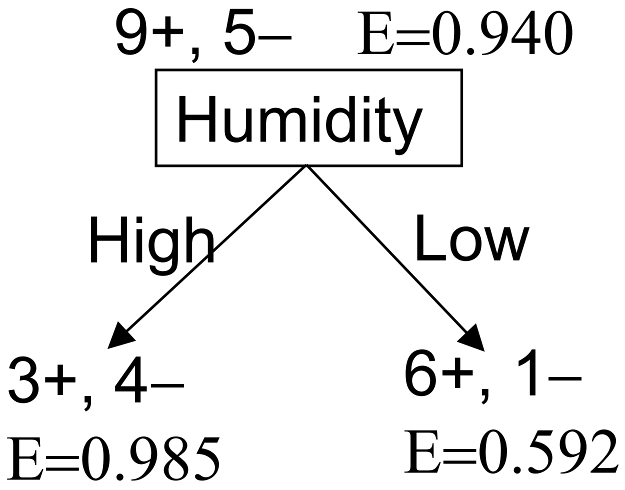
For two classes (+ve, -ve), max entropy: 1.

- Significance
  - gain of 0?
    - 50/50 split of +/- both before *and* after discriminating on attributes values
  - gain of 1?
    - from “perfect uncertainty” to perfect certainty after splitting example with predictive attribute
  - Find “patterns” in TE’s relating to attribute values
    - Move to locally minimal representation of TE’s

Training Examples

Day	Outlook	Temp	Humidity	Wind	Tennis?
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*

Determine the Root Attribute



9+, 5– E=0.940

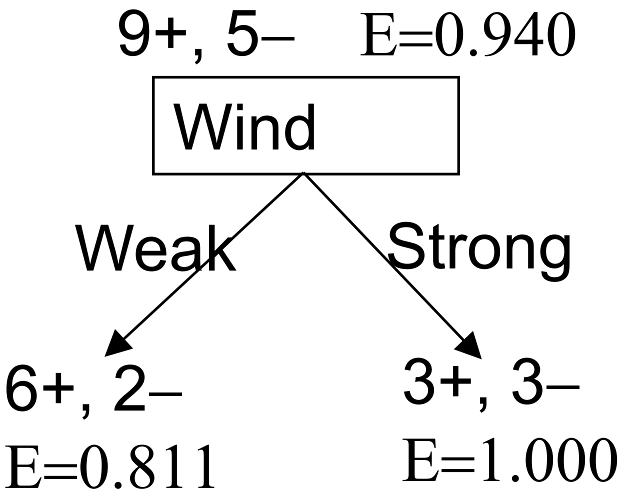
Humidity

High

Low

3+, 4–
E=0.985

6+, 1–
E=0.592



9+, 5– E=0.940

Wind

Weak

Strong

6+, 2–
E=0.811

3+, 3–
E=1.000

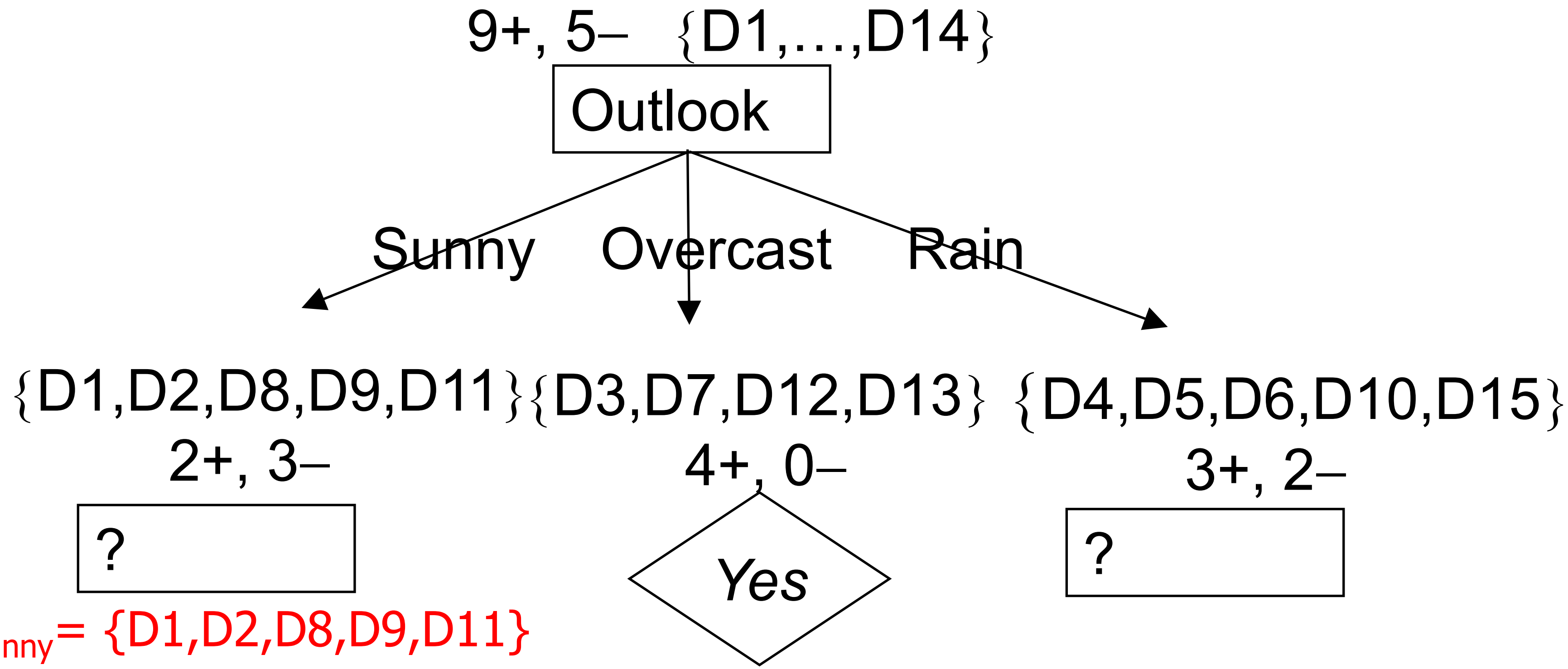
Gain (S, Humidity) = 0.151

Gain (S, Wind) = 0.048

Gain (S, Outlook) = 0.246

Gain (S, Temp) = 0.029

Sort the Training Examples



9+, 5– {D1,…,D14}

Outlook

Sunny Overcast Rain

{D1,D2,D8,D9,D11} 2+, 3–

{D3,D7,D12,D13} 4+, 0–

{D4,D5,D6,D10,D15} 3+, 2–

?

Yes

?

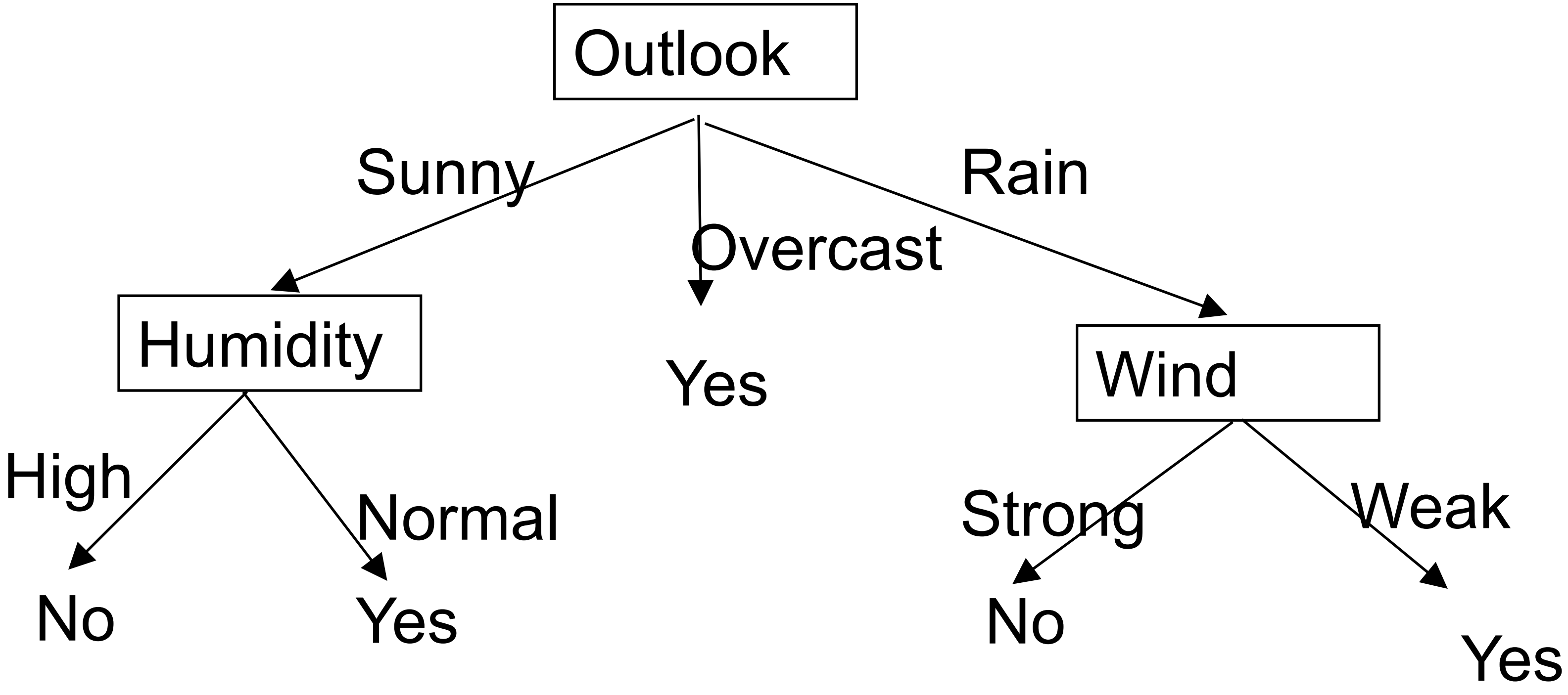
S_{sunny}= {D1,D2,D8,D9,D11}

Gain (S_{sunny}, Humidity) = .970

Gain (S_{sunny}, Temp) = .570

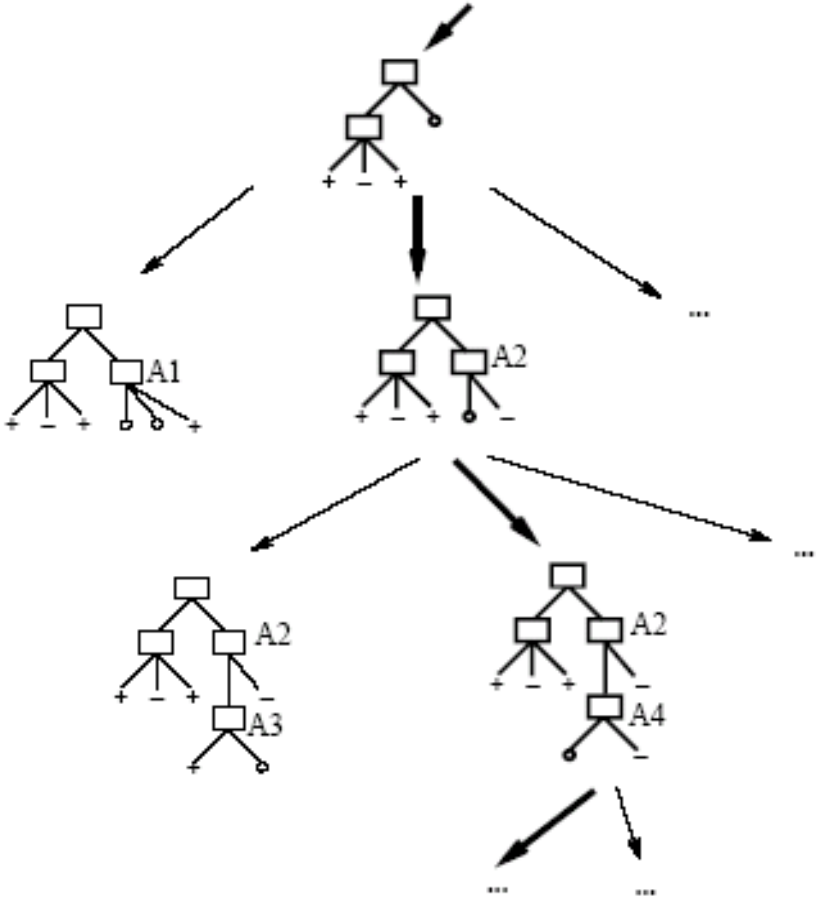
Gain (S_{sunny}, Wind) = .019

Final Decision Tree for Example



Hypothesis Space Search (ID3)

- Hypothesis space (all possible trees) is complete!
  - Target function is included in there



ID3

- Input: TEs
    - Attributes, Target_attribute (+ve / -ve)
- Create a root node
    - If all TEs +ve / -ve, return a single node with label +ve / -ve.
    - If Attr. empty, return single node with the most freq. label.
- Select the best attribute A
    - For each possible value v_i of A
        - Form the subset S of TEs whose value of A is v_i
        - If S empty
            - return by adding a leaf node below with the most freq. label.
        - Else
            - recursively call ID3 with S to form the subtree.

Hypothesis Space Search in Decision Trees

- 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.
- Perform a greedy heuristic search: hill climbing without backtracking
- Statistics-based decisions using all data

Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - The space of all finite DT’s (all discrete functions)
  - Target function included
- Simple to complex hill-climbing search of H
  - Use of gain as hill-climbing function
- Outputs a single hypothesis (which one?)
  - Cannot assess all hypotheses consistent with D (usually many)
  - Analogy to breadth first search
    - Examines all trees of given depth and chooses best…
- No backtracking
  - Locally optimal ...
- Statistics-based search choices
  - Use all TE’s at each step
  - Robust to noisy data

Restriction bias vs. Preference bias

- Restriction bias (or Language bias)
  - Incomplete hypothesis space
- Preference (or search) bias
  - Incomplete search strategy
- Candidate Elimination has restriction bias
- ID3 has preference bias
- In most cases, we have both a restriction and a preference bias.

Inductive Bias in ID3

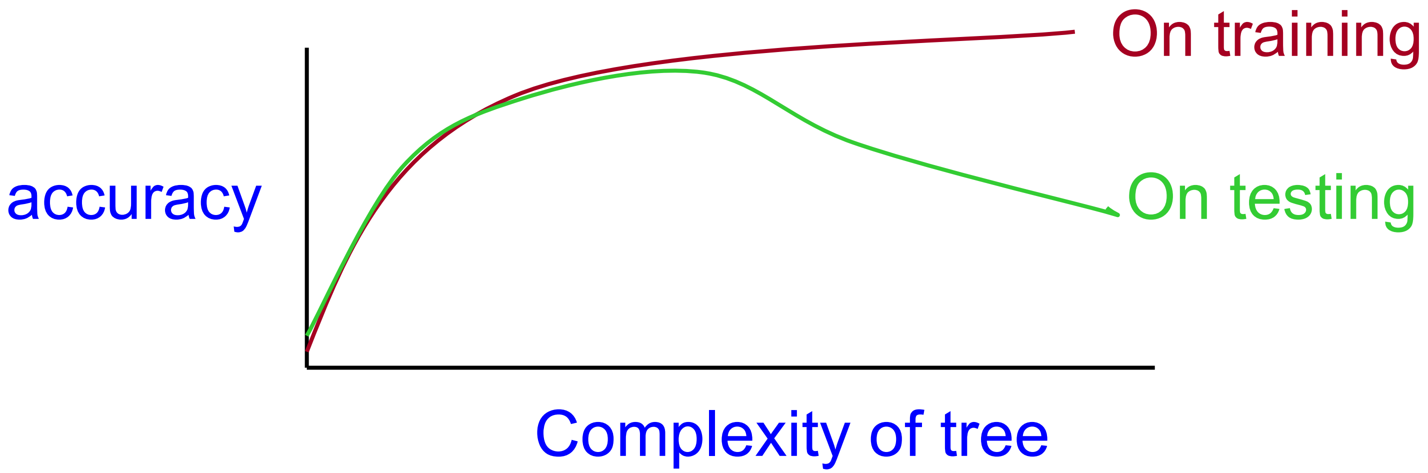
- Preference for short trees, and for those with high information gain attributes near the root
- Principle of Occam's razor
  - prefer the shortest hypothesis that fits the data
- Justification
  - Smaller likelihood of a short hypothesis fitting the data at random
- Problems
  - Other ways to reduce random fits to data
  - Size of hypothesis based on the data representation
    - Minimum description length principle

Overfitting the Data

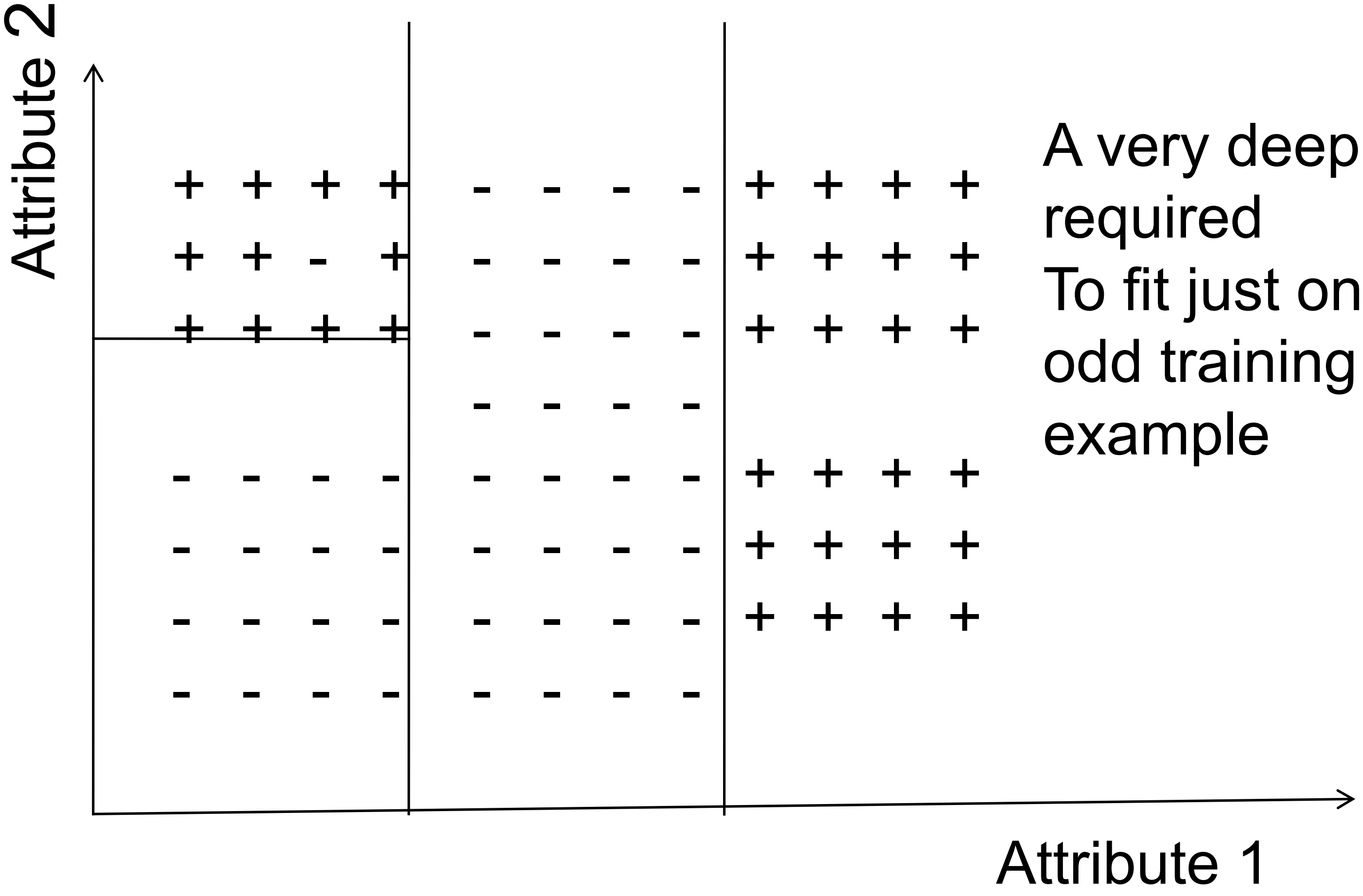
•May not have the best generalization performance.

- noise in the training data
- very little data

•*h* overfits the training data if there exists another hypothesis, h', such has greater error than *h* on the training data, but *smaller* error on the test data than *h*.

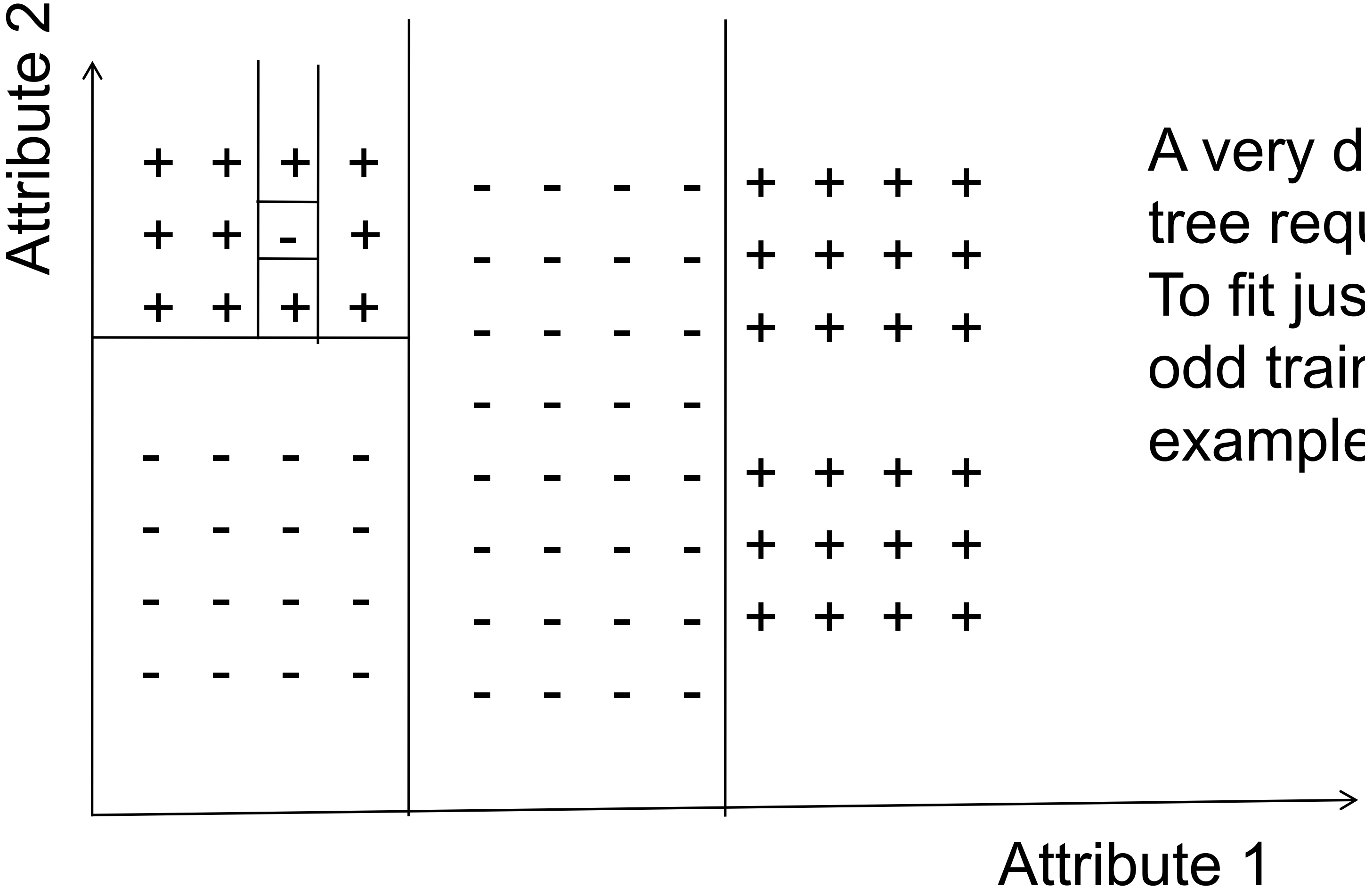


Overfitting



A very deep tree required
To fit just one odd training example

When to stop splitting further?



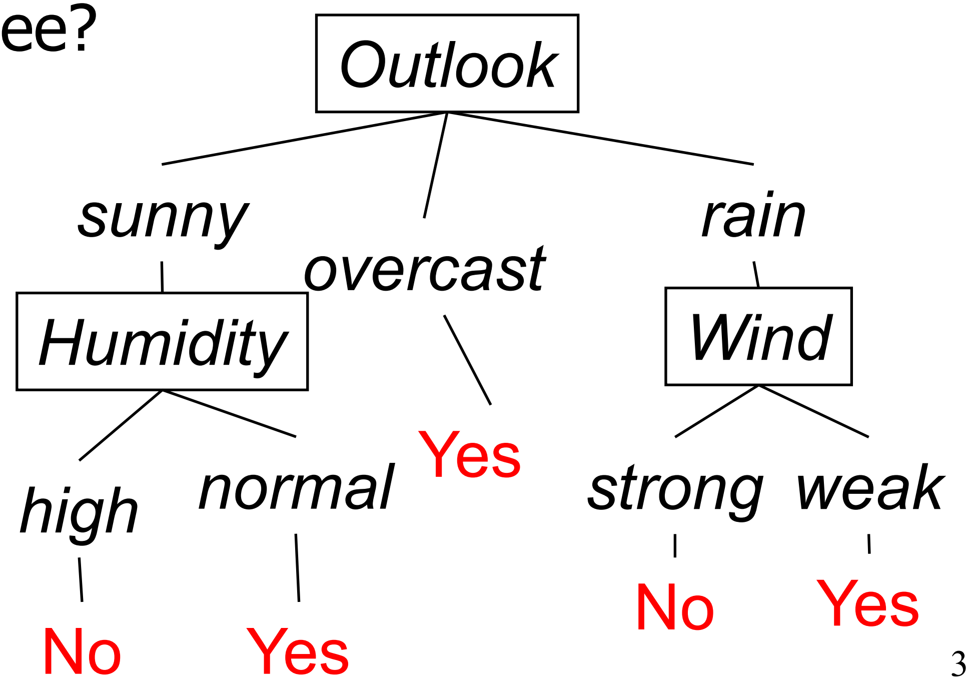
A very deep tree required To fit just one odd training example

Overfitting in Decision Trees

- Consider adding *noisy* training example (should be +):

Day	Outlook	Temp	Humidity	Wind	Tennis?
D15	Sunny	Hot	Normal	Strong	*No*

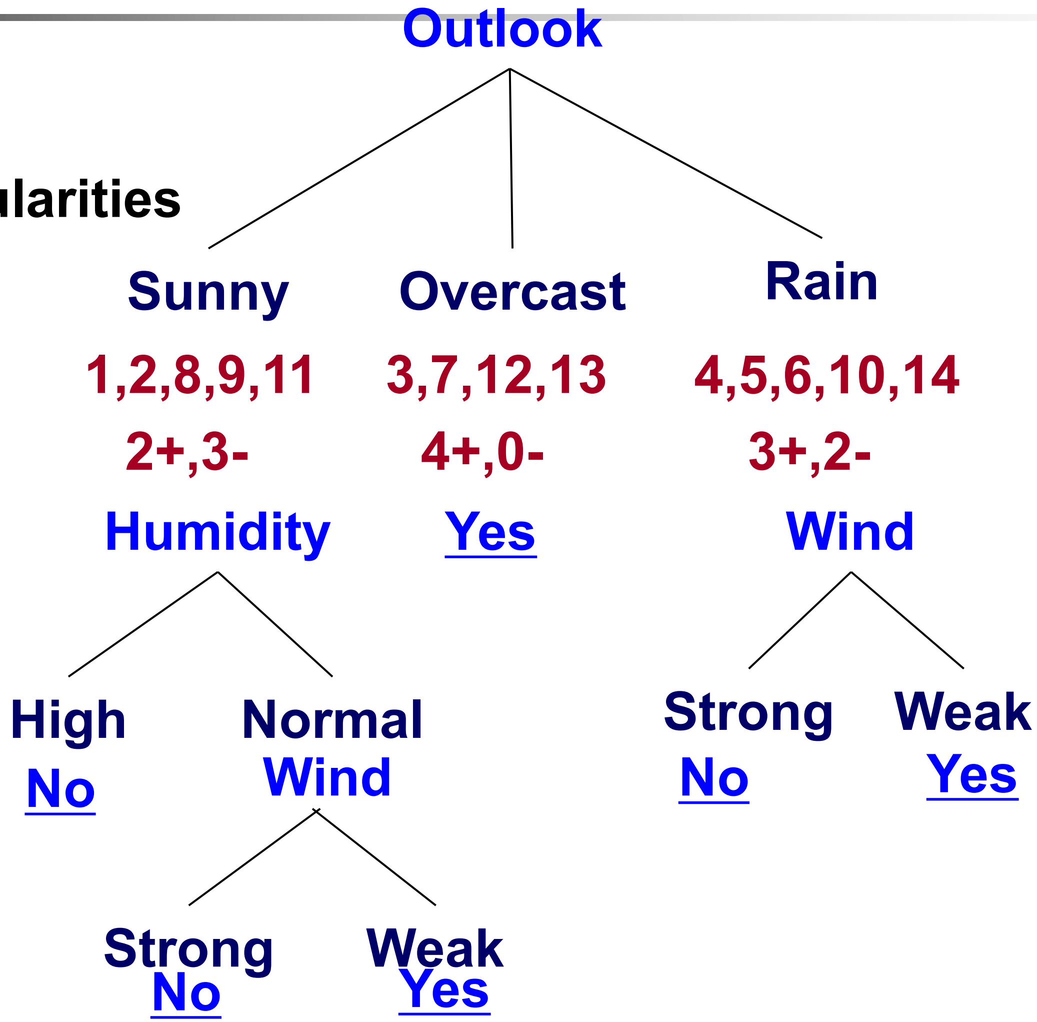
- What effect on earlier tree?



Outlook
sunny
overcast
rain
Humidity
Yes
Wind
high
normal
strong
weak
No
Yes
No
Yes

Overfitting - Example

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. (more popular)

Evaluating subtrees to prune

- Methods of evaluation
  - Cross-validation:
    - Reserve hold-out set to evaluate utility (more popular)
  - Statistical testing:
    - Test if the observed regularity can be dismissed as likely to occur by chance
  - Minimum Description Length:
    - Is the additional complexity of the hypothesis smaller than remembering the exceptions ? (regularization in other contexts)

General Strategy: Overfit and Simplify

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:

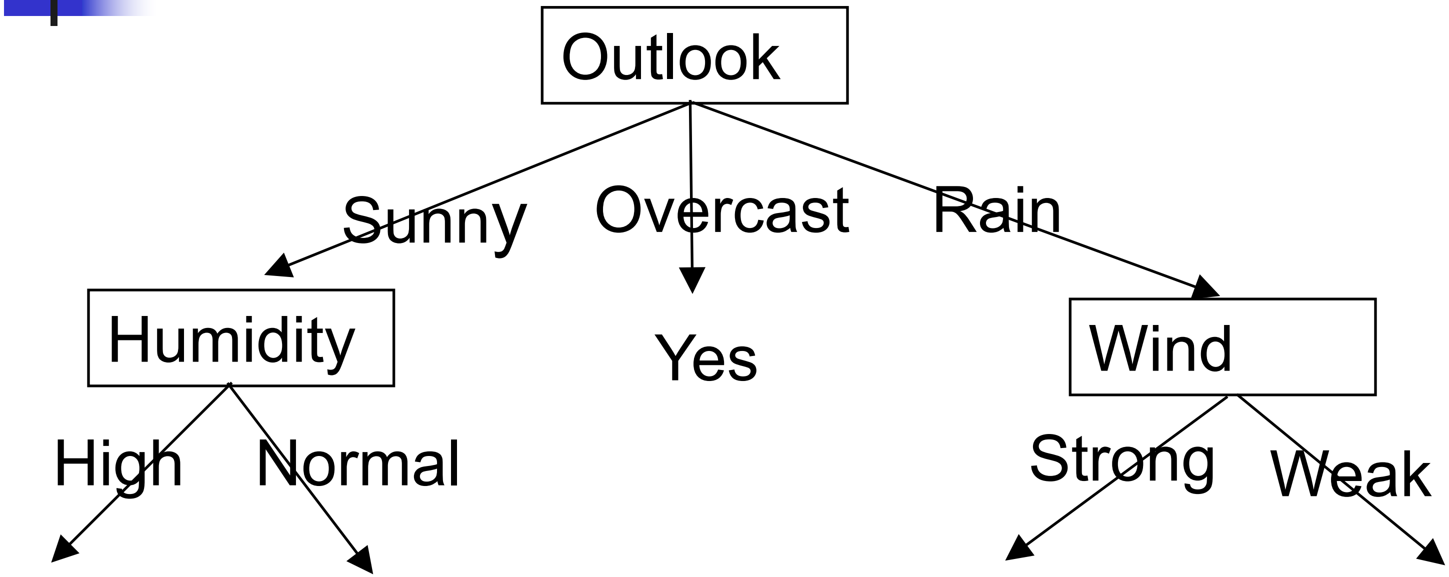
```
For each non-leaf node in the tree
    Temporarily prune the tree below;
                replace it by majority vote.
    Test the accuracy on the validation set
Permanently prune the node with the greatest increase
in accuracy on the validation test.
```

- Problem: Uses less data to construct the tree
- Sometimes done at the rules level.

Rule post-pruning

- Allow tree to grow until best fit (allow overfitting)
- Convert tree to equivalent set of rules
  - One rule per leaf node
- Prune each rule independently of others
    - Remove various preconditions to improve performance
- Sort final rules into desired sequence for use

Example of rules



- IF (Outlook = Sunny) ^ (Humidity = High)
  - THEN PlayTennis = No
- IF (Outlook = Sunny) ^ (Humidity = Normal)
  - THEN PlayTennis = Yes

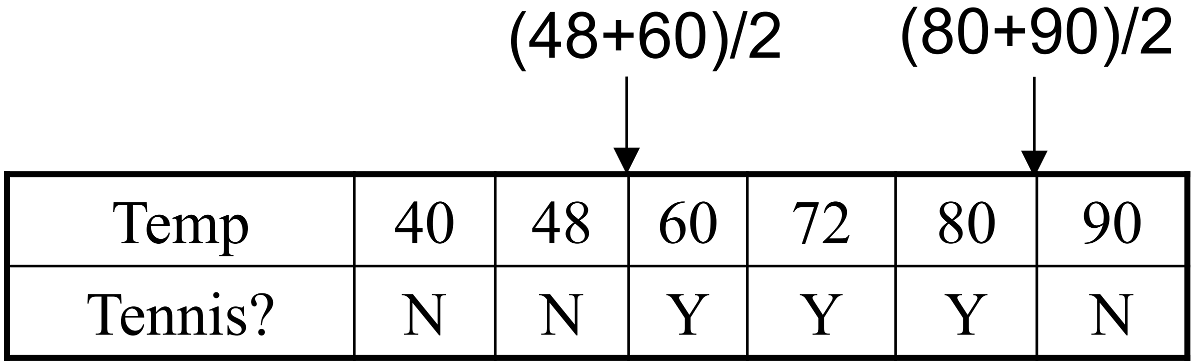
Prune preconditions and evaluate.

Extensions of basic algorithm

- Continuous valued attributes
- Attributes with many values
- TE’s with missing data
- Attributes with associated costs
- Other impurity measures
- Regression tree

Continuous Valued Attributes

- Create a discrete attribute from continuous variables
  - E.g., define critical Temperature = 82.5

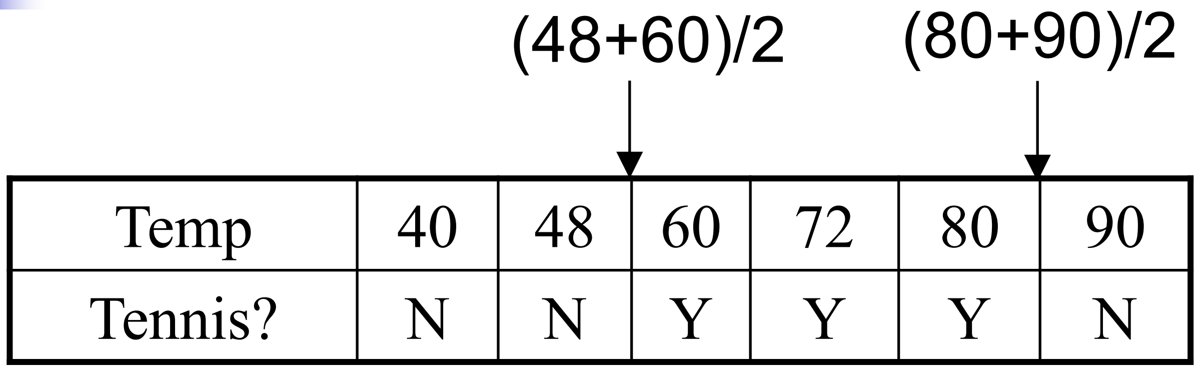


(48+60)/2 (80+90)/2

Temp	40	48	60	72	80	90
Tennis?	N	N	Y	Y	Y	N

- Candidate thresholds
  - chosen by information gain function
    - Check gain for the attribute with possible thresholds and choose the maximum
  - can have more than one threshold

Candidate Thresholds



(48+60)/2

(80+90)/2

Temp	40	48	60	72	80	90
Tennis?	N	N	Y	Y	Y	N

- Chosen by information gain function
  - Check gain for the attribute with possible thresholds and choose the maximum
- Can have more than one threshold
- Typically where target values change quickly

Attributes with Many Values

- Problem:
  - If attribute has many values, *Gain* will select it
  - e.g. of birthdates attribute
    - 365 possible values
    - Likely to discriminate well on small sample
      - But poor predictor on unseen instances
    - Many small partitions and likely to have low entropy in many of them.
      - Information gain likely to be high

Attributes with many values

- Problem: *Gain* will select attribute with many values
- One approach: use *GainRatio* instead

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

Measures entropy from distribution in attribute space

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

Entropy of the partitioning penalizes higher number of partitions

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

(example of S_i / S = 1/N: SplitInformation = log N)

Unknown Attribute Values

- How to handle missing values of attribute *A*?
- Use training examples 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 test instances with missing values in same fashion
- Used in C4.5

Attributes with Costs

- Consider
  - medical diagnosis: BloodTest has cost \$150, Pulse has a cost of \$5.
  - robotics, Width-From-1ft has cost 23 sec., from 2 ft 10s.
- How to learn a consistent tree with low expected cost?
- Replace gain by $\frac{Gain^2(S,A)}{Cost(A)}$
  - Tan and Schlimmer (1990)

a constant in [0,1]

$$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^{\omega}}$$

Gini Index

- Another sensible measure of impurity
  (i and j are classes)

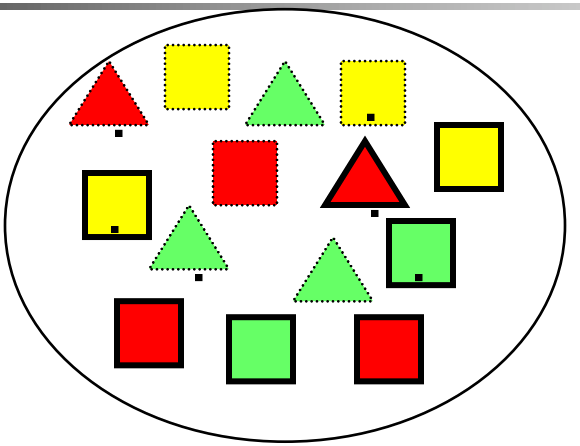
$$Gini = \sum_i p(i)(1 - p(i)) \rightarrow Gini = 1 - \sum_i p(i)^2$$

- After applying attribute A, the resulting Gini index is

$$Gini(A) = \sum_{v \epsilon A} p(v) \sum_i p(i|v)(1 - p(i|v))$$

- Gini can be interpreted as expected error rate

Gini Index



$$p(\square) = \frac{9}{14}$$

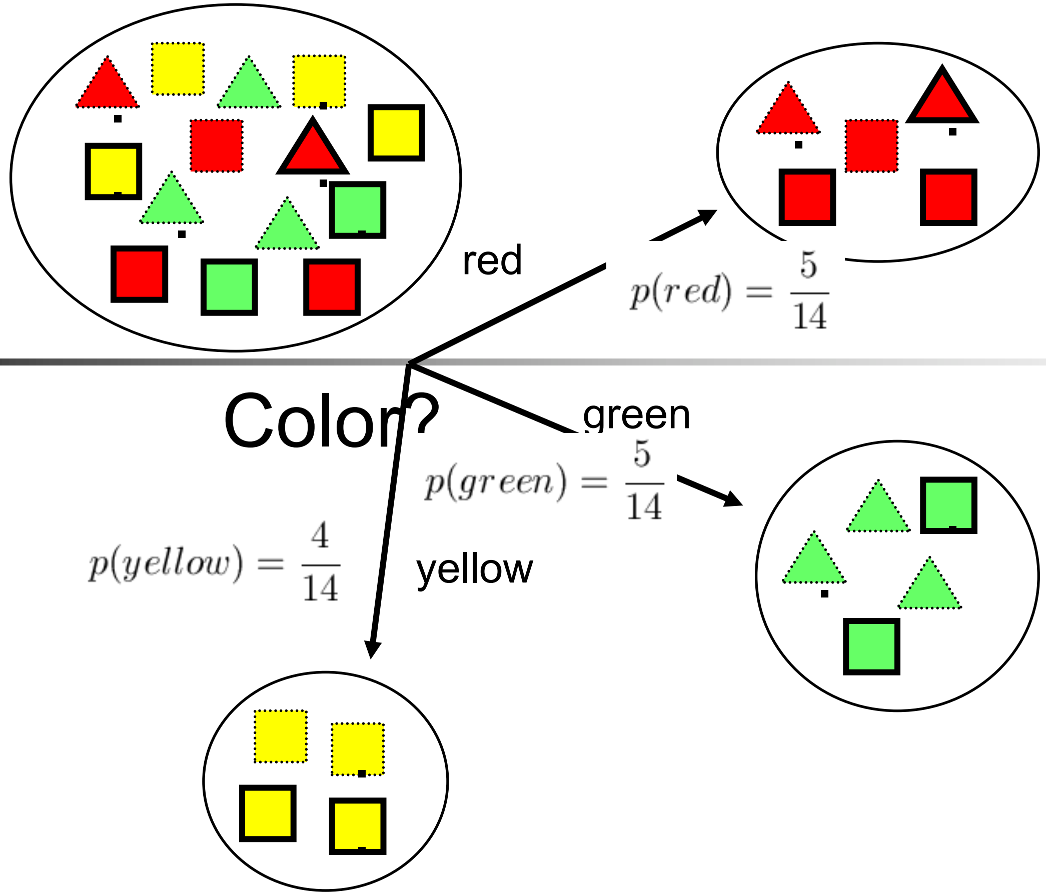
$$p(\Delta) = \frac{5}{14}$$

Attributes: color, border, dot
Classification: triangle, square

$$Gini = \sum_i p(i)(1 - p(i))$$

$$Gini = 2 \times \frac{9}{14} \times \frac{5}{14} = 0.46$$

Gini Index for Color



$$Gini(A) = \sum_{v \epsilon A} p(v) \sum_{i} p(i|v)(1 - p(i|v))$$

$$Gini(color) = \frac{5}{14} \times \left(2 \times \frac{3}{5} \times \frac{2}{5}\right) + \frac{5}{14} \times \left(2 \times \frac{2}{5} \times \frac{3}{5}\right) + \frac{4}{14} \times \left(2 \times \frac{4}{4} \times \frac{0}{4}\right) = 0.342$$

Gain of Gini Index

$$Gini = 2\times\frac{9}{14}\times\frac{5}{14} = 0.46$$

$$Gini(color) = \frac{5}{14}\times\left(2\times\frac{3}{5}\times\frac{2}{5}\right) + \frac{5}{14}\times\left(2\times\frac{2}{5}\times\frac{3}{5}\right)+\frac{4}{14}\times\left(2\times\frac{4}{4}\times\frac{0}{4}\right) = 0.342$$

GiniGain (Color) = 0.46-0.342=.116

Three Impurity Measures

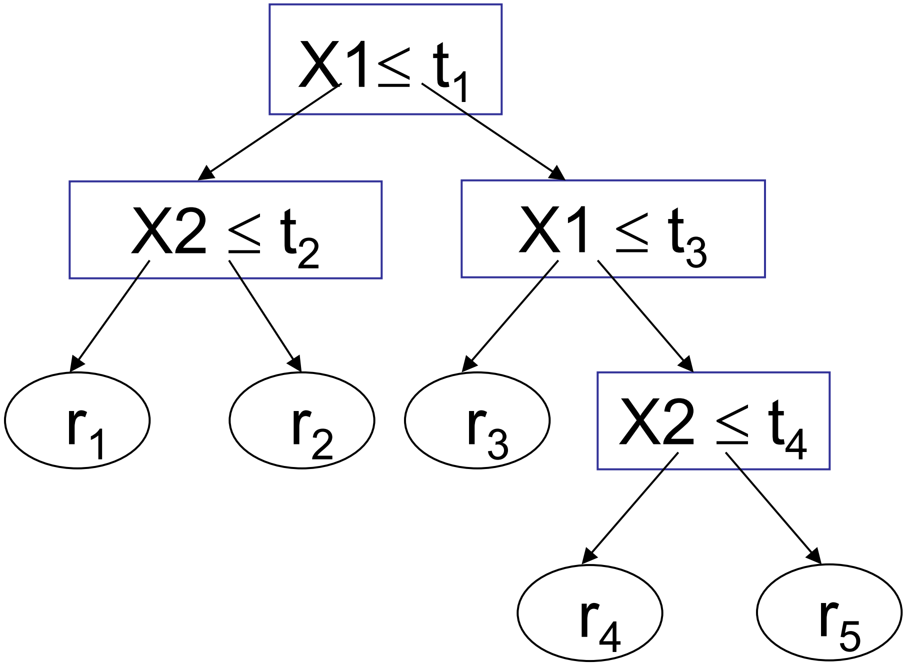
A	Gain(A)	GainRatio(A)	GiniGain(A)
Color	0.247	0.156	0.116
Outline	0.152	0.152	0.092
Dot	0.048	0.049	0.03

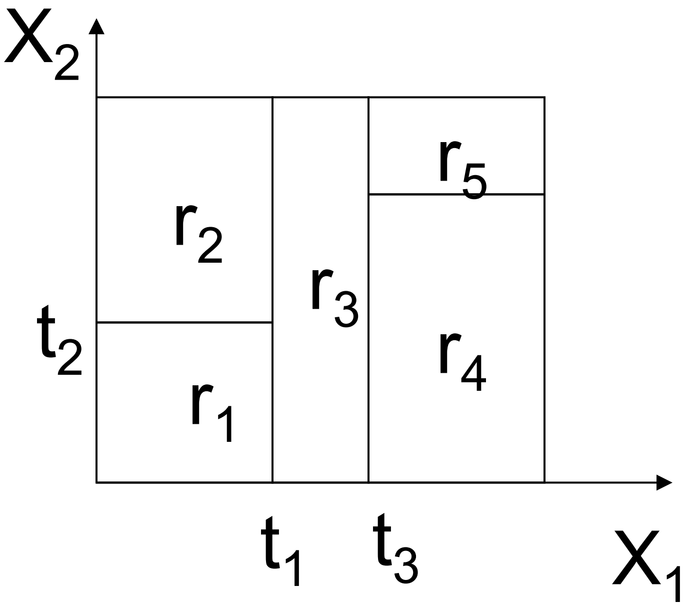
Regression Tree

- Similar to classification
- Use a set of attributes to predict the value (instead of a class label)
- Instead of computing information gain, compute the sum of squared errors
- Partition the attribute space into a set of rectangular subspaces, each with its own predictor
  - The simplest predictor is a constant value

Rectilinear Division

- A regression tree is a piecewise constant function of the input attributes





Growing Regression Trees

- To minimize the square error on the learning sample,
  - the prediction at a leaf is the average output of the learning cases reaching that leaf
- Impurity of a sample defined by the variance of the output in that learning sample (LS):

$$I(LS)=\mathrm{var}_{y|LS}\{y\}=\mathrm{E}_{y|LS}\{(y-\mathrm{E}_{y|LS}\{y\})^2\}$$

- The best split is the one that reduces the most variance:

$$\Delta I(LS,A)=\mathrm{var}_{y|LS}\{y\}-\sum_a \frac{|LS_a|}{|LS|}\mathrm{var}_{y|LS_a}\{y\}$$

Regression Tree Pruning

- Exactly the same algorithms
  - apply pre-pruning and post-pruning.
- In post-pruning,
  - the tree that minimizes the squared error on *VS* is selected.
- In practice, pruning more important in regression because full trees are much more complex
  - often all objects have a different output values and hence the full tree has as many leaves as there are objects in the learning sample

When Are Decision Trees Useful ?

- Advantages
  - Very fast: can handle very large datasets with many attributes
  - Flexible: several attribute types, classification and regression problems, missing values…
  - Interpretability: provide rules and attribute importance
- Disadvantages
  - Instability of the trees (high variance)
  - Not always competitive with other algorithms in terms of accuracy

Learning rules directly

- From a decision tree a set of rules can be derived and can be applied for inference.
- Direct learning of rule from TEs?
  - IF <antecedent> THEN <consequent>
  - Assume antecedent in the form of CNF
    - (Outlook=SUNNY) ^ (Temperature=HIGH)^(Wind=WEAK)
  - Consequent as assignment of target attribute value
    - PlayTennis=YES

Sequential covering algorithms

- Learn rules one by one
  - Learn a rule from training examples and remove those covered by the rule.
  - Iterate the process till all examples are covered.
- Learn-One-Rule
  - General to specific beam search
    - Most general rule
      - If <EMPTY > THEN <consequent>
    - Greedy depth first search on forming CNF of attribute conditions in the form of (Attr=value)

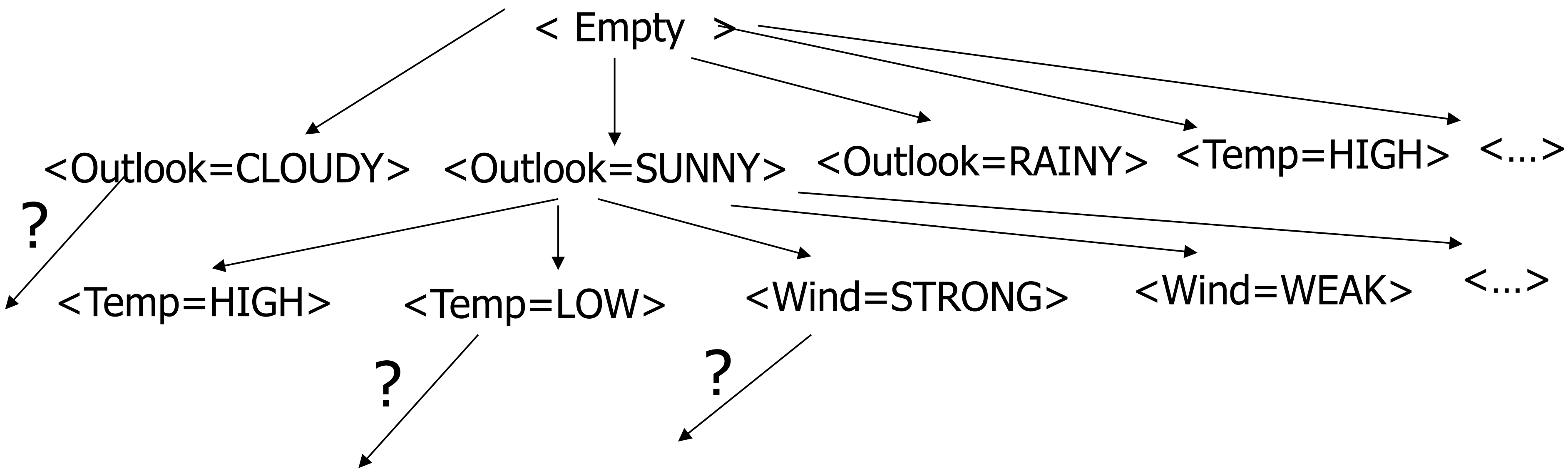
Chapter 10 of “Machine learning” by Tom M. Mitchel

General to specific beam search

- Initialize h with the most general rule
    - Maintain a candidate list of k top performing rules.
- For each rule in the candidate list
    - Specialize with a constraint (A=v) by adding in CNF
    - Consider all possible attr-value pairs in TEs
        - Exclude attribute sets already present in the antecedent.
    - Update h with the best performance measure.
        - Entropy of the class distribution (lower better)
- Update the candidate list with k top performing rules
- Iterate above till it covers all attributes in the list.
- Choose the best and form the consequent
    - by assigning most frequent target value in the covered set.

General to specific beam search

- Form the antecedent of the best performing hypothesis.



- Form consequent assigning maximally occurring target value in covered examples

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 good general purpose algorithm

Beyond decision trees …

Oblique Decision Tree

- Oblique Decision Tree (Heath, Kasif and Salzberg, IJCAI'93)
  - A combination of multiple attributes checked in a node.
    - Not on a single attribute as in an ordinary DT.
  - A hyperplane dividing into two halves of a space.
    - For an ordinary DT all planes are axis parallel.
  - Usually smaller DT
    - Highly dependent on choice of hyperplanes.
    - Central axis projection
      - Get two clusters and search a hyperplane separating them normal to the axis formed by their centers.
    - Perceptron Training
      - Use perceptron classifier for two sets to get the hyperplane

Random Decision Forest

- Random decision forest (Ho, ICDAR'95)
  - Multiple trees from randomly constructed subspaces.
    - For m attributes, how many subspaces?
      - 2^m
      - How many to use?
      - How to select?
        - Randomly!
    - Each providing consistent decision tree.
      - No training error
  - An aggregation policy on the class labels obtained from each tree.
      - e.g. Voting, Avg. Posterior Probability of a class, etc.

Summary

- DTs allowing concept learning with noisy data.
  - Learn a set of rules
- Basic information measure and gain function for best first search of space of DTs.
- ID3 procedure
  - search space is complete
  - Preference for shorter trees
- Overfitting an important issue with various solutions
- Many variations and extensions possible
  - ODT, Random Forest

Software

- In R:
  - Packages tree and rpart
- C4.5:
  - http://www.cse.unwe.edu.au/~quinlan
- Weka
  - http://www.cs.waikato.ac.nz/ml/weka

Thank you!