Chapter 8 Non-metric Methods

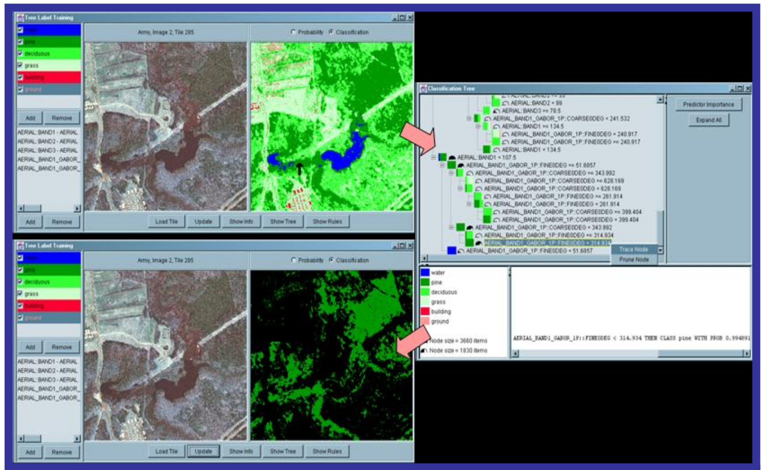
DECISION TREES

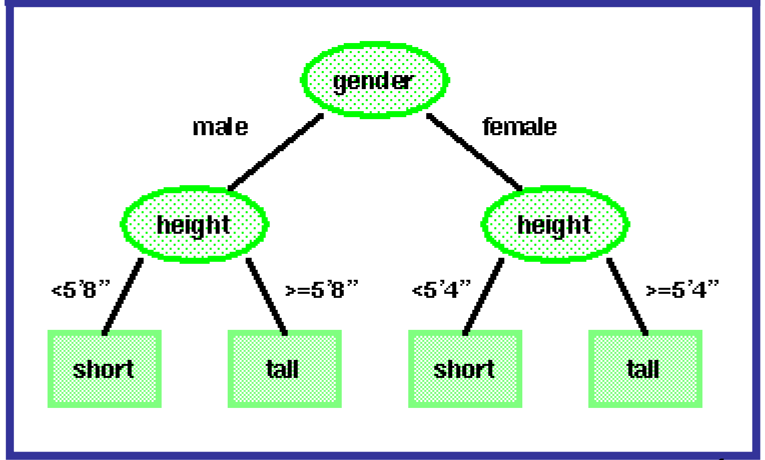
- **Objectives:**

  Tree-Growing Via CART
  Splitting, Stopping and Pruning
  Attributes
  Node Impurity
  Priors and Costs

- **Resources:**

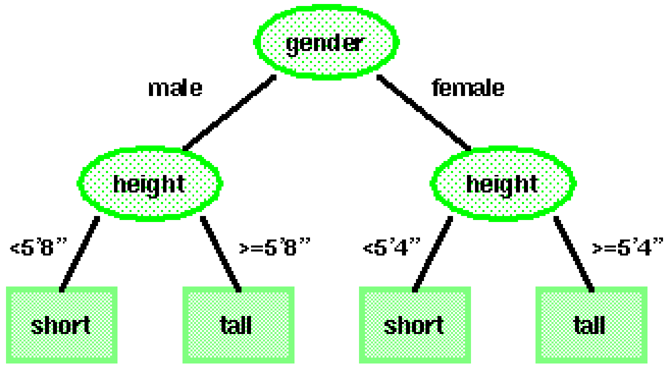
  DHS: Chapter 8
  WIKI: Definitions
  AAAI: Decision Trees
  AM: Data Mining
  DTDM: Resources
  JB: Examples
  ISIP: Software



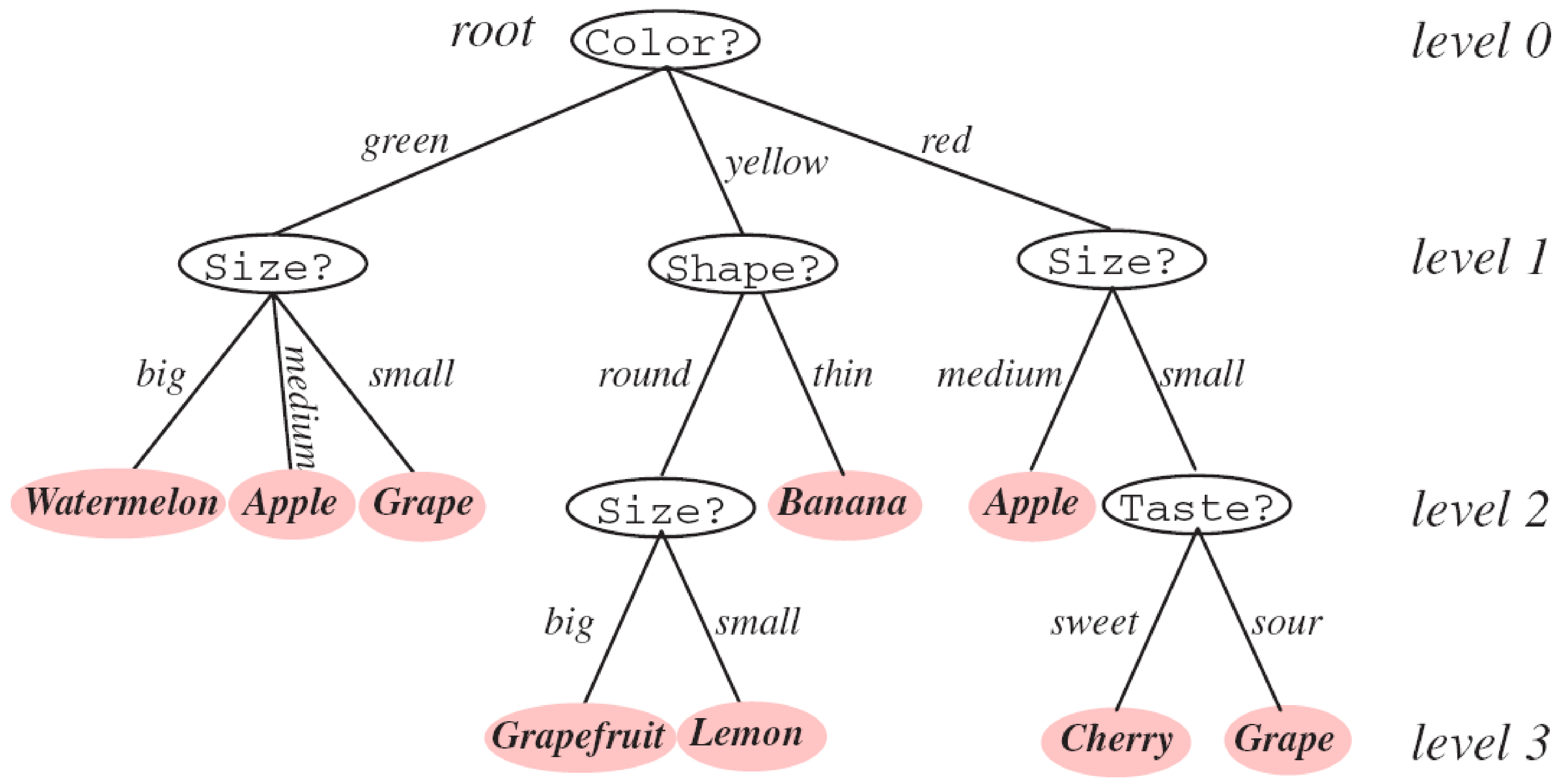


Overview

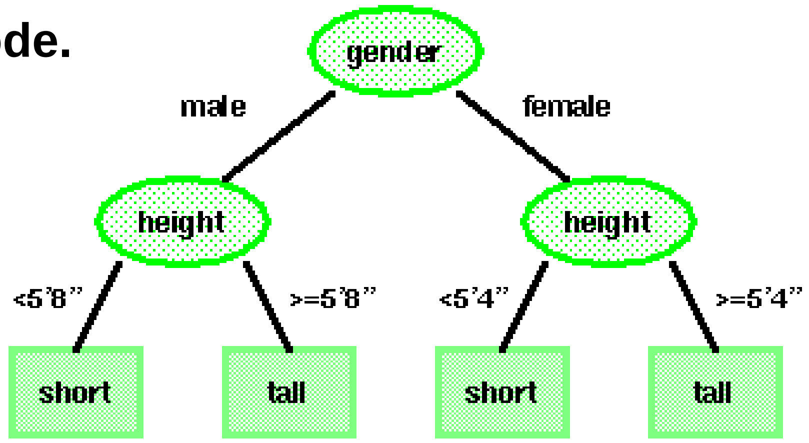
- Previous techniques have consisted of real-valued feature vectors (or discrete-valued) and natural measures of distance (e.g., Euclidean).
- Consider a classification problem that involves nominal data – data described by a list of attributes (e.g., categorizing people as short or tall using gender, height, age, and ethnicity).
- How can we use such nominal data for classification? How can we learn the categories of such data? *Nonmetric* methods such as decision trees provide a way to deal with such data.
- Decision trees attempt to classify a pattern through a sequence of questions. For example, attributes such as gender and height can be used to classify people as short or tall. But the best threshold for height is gender dependent.



- A decision tree consists of nodes and leaves, with each leaf denoting a class.
- Classes (tall or short) are the outputs of the tree.
- Attributes (gender and height) are a set of features that describe the data.
- The input data consists of values of the different attributes. Using these attribute values, the decision tree generates a class as the output for each input data.


root
Color?
level 0
green
yellow
red
Size?
Shape?
Size?
level 1
big
medium
small
round
thin
medium
small
Watermelon
Apple
Grape
Size?
Banana
Apple
Taste?
level 2
big
small
sweet
sour
Grapefruit
Lemon
Cherry
Grape
level 3

Basic Principles

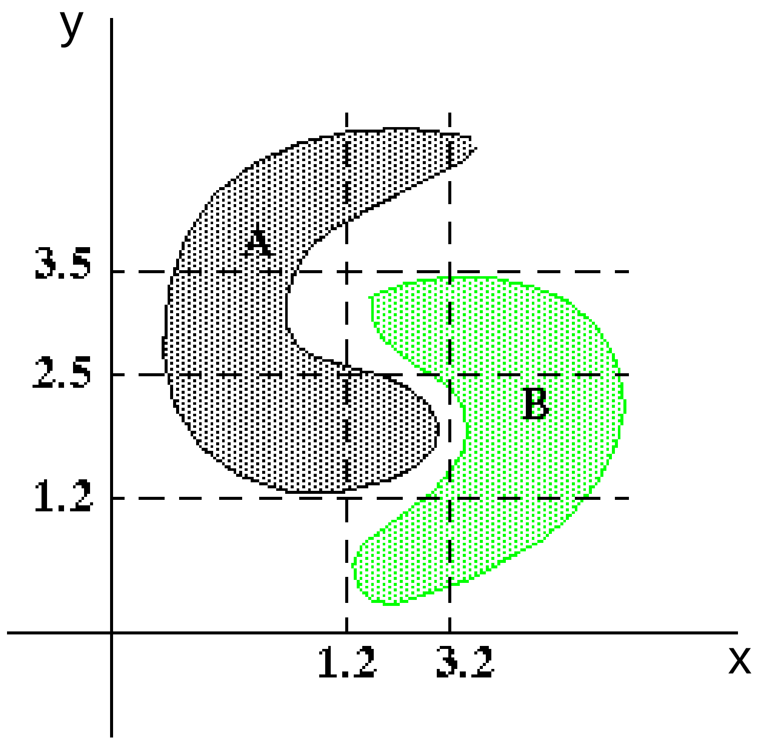


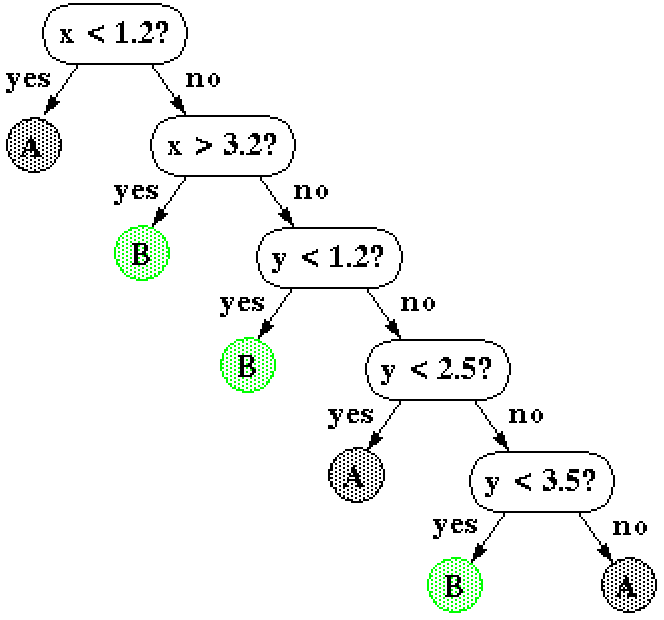
- The top, or first node, is called the root node.
- The last level of nodes are the leaf or terminal nodes and contain the final classification.
- The intermediate nodes (links or branches) are the descendant or "hidden" layers.
- Binary trees, like the one shown to the right, are the most popular type of tree. However, M-ary trees (M branches at each node) are possible.
- Nodes can contain one more questions. In a binary tree, by convention if the answer to a question is "yes", the left branch is selected. Note that the same question can appear in multiple places in the network.
- Decision trees have several benefits over neural network-type approaches, including interpretability and data-driven learning.
- Key questions include how to grow the tree, how to stop growing, and how to prune the tree to increase generalization.

- Benefit of decision tree:
  - Interpretability: a tree can be expressed as logical expression
  - Rapid classification: a sequence of simple queries
- Decision trees are very powerful and can give excellent performance on closed-set testing. Generalization is a challenge.

Nonlinear Decision Surfaces

- **Decision trees can produce nonlinear decision surfaces:**





- **They are an attractive alternative to other classifiers we have studied because they are data-driven and can give arbitrarily high levels of precision on the training data.**
- **But… generalization becomes a challenge.**

How to Grow A Tree?

- Given a set of features and set D of labeled training samples
- How to organize the tests/questions into a tree?
- Starting with all the training samples at the root node, A decision tree progressively splits the training set into smaller and smaller subsets

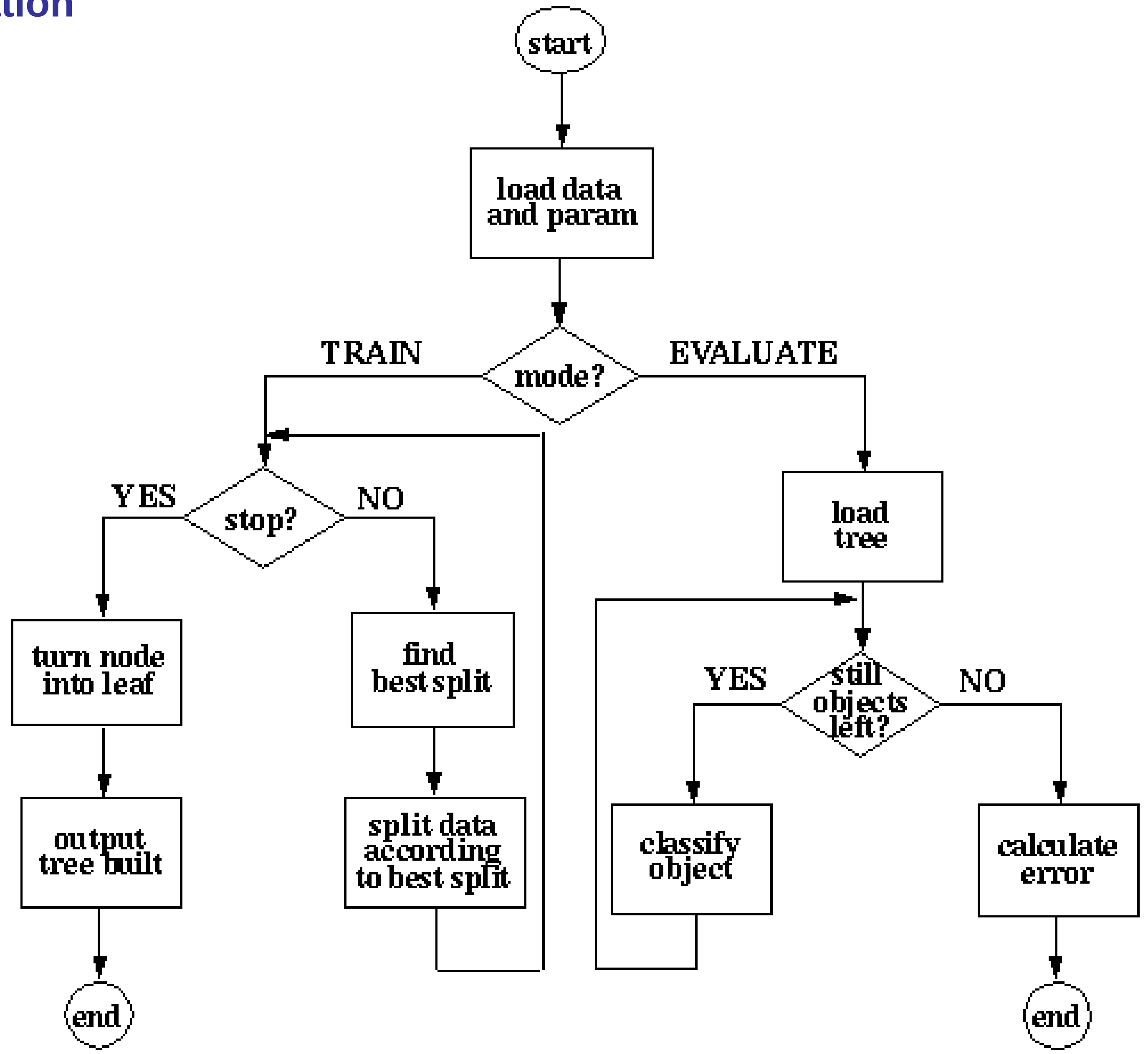
- Pure node: all the samples at a pure node have the same label; it becomes a leaf node and no need to further split that leaf node
- Recursive tree-growing process: Given data at a node, decide the node as a leaf node or find another feature to split the node
- This generic procedure is called CART (Classification And Regression Trees)

Classification And Regression Trees (CART)

- Consider a set D of labeled training data and a set of properties (or questions), T.
- How do we organize the tree to produce the lowest classification error?
- Any decision tree will successively split the data into smaller and smaller subsets. It would be ideal if all the samples associated with a leaf node were from the small class. Such a subset, or node, is considered pure in this case.
- A generic tree-growing methodology, known as CART, successively splits nodes until they are pure. Six key questions:

- Should the properties/features be restricted to binary or allowed to be multi-valued? In other words, how many splits should be made at a node?
- Which feature should be tested at a node?
- When should a node be declared a leaf?
- If the tree becomes “too large”, how can it be pruned to make it smaller and simpler?
- If a leaf node is impure, how should the category label be assigned to it?
- How should missing data be handled?

Operation


start
load data and param
mode?
TRAIN
EVALUATE
stop?
YES
NO
turn node into leaf
output tree built
end
find best split
split data according to best split
load tree
still objects left?
YES
NO
classify object
calculate error
end

Query Selection and Node Impurity

- Which property test or query should be performed at each node?
- Fundamental principle of simplicity: prefer simple, compact trees with few nodes
- Seek a query T at node N to make the descendent nodes as pure as possible
- Query of the form $x_i \leq x_{is}$ leads to hyperplanar decision boundaries that are perpendicular to the coordinate axes (monothetic tree; one feature/node)

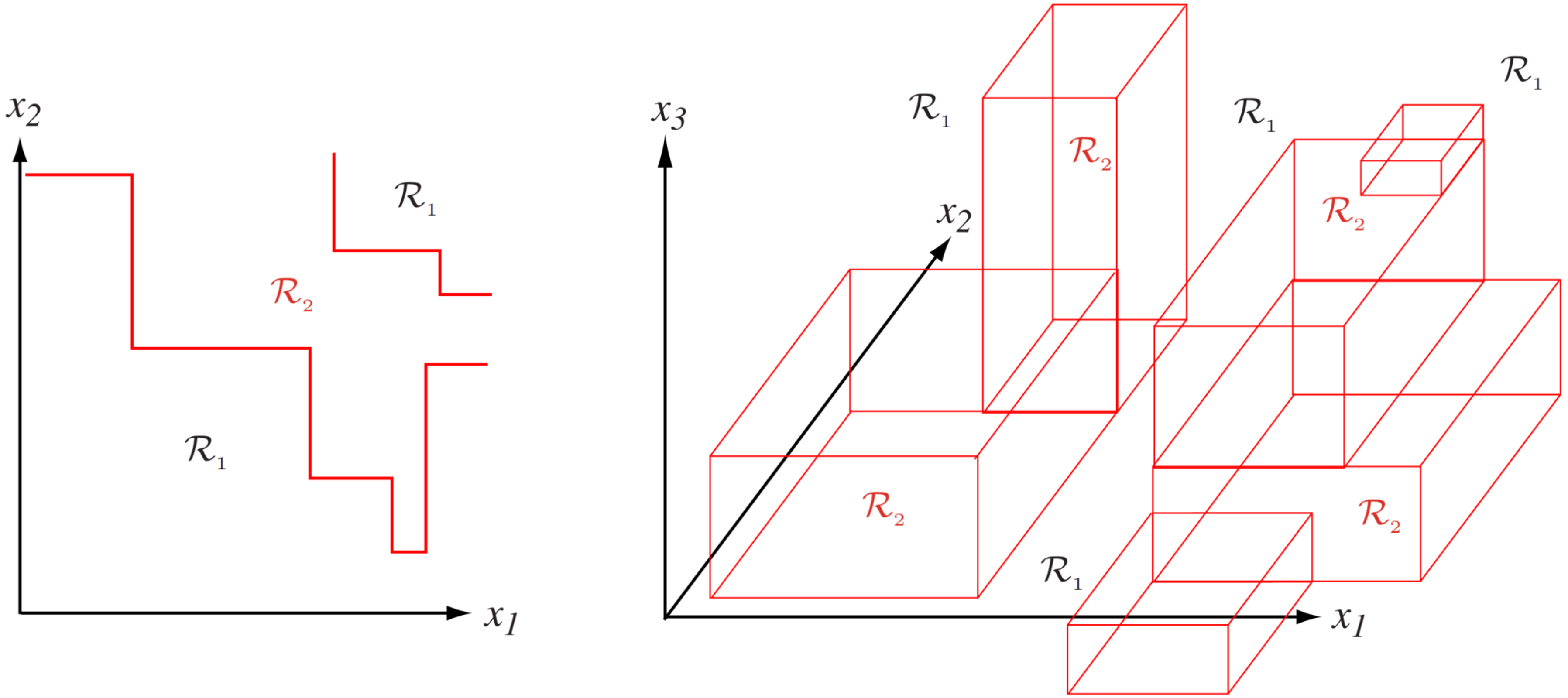


Figure 8.3: Monothetic decision trees create decision boundaries with portions perpendicular to the feature axes. The decision regions are marked R_1 and R_2 in these two-dimensional and three-dimensional two-category examples. With a sufficiently large tree, any decision boundary can be approximated arbitrarily well.

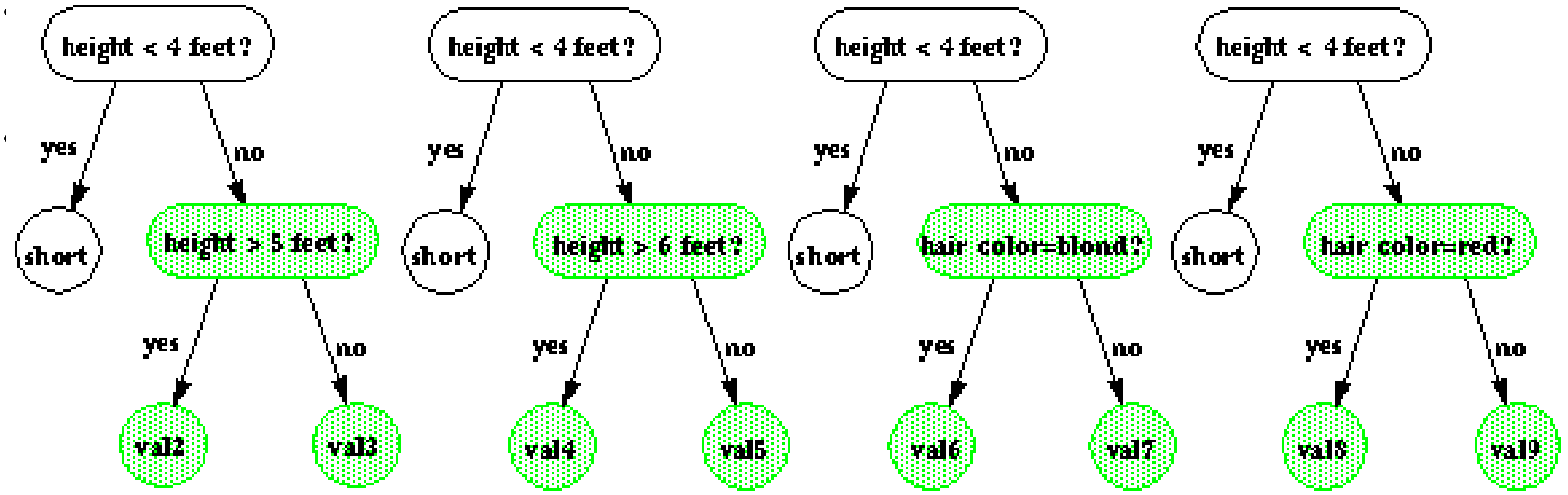
Query Selection and Node Impurity

- $P(\omega_j)$: fraction of patterns at node *N* in category ω_j.
- Node impurity is 0 when all patterns at the node are of the same category.
- Impurity becomes maximum when all the classes at node *N* are equally likely
- Entropy impurity of node N ($i(N) \geq 0$; greatest value when $P(\omega_j)$ is same for all *j*)

$$i(N) = -\sum_j P(\omega_j)\log_2(P(\omega_j))$$

Entropy-Based Splitting Criterion

- **We prefer trees that are simple and compact. Why? (Hint: Occam's Razor).**



Occam's razor is a principle that generally recommends selecting the competing hypothesis that makes the fewest new assumptions.

$$i(N) = -\sum_j P(\omega_j)\log(P(\omega_j))$$

Information Gain (*entropy*) is used to select the most useful attribute for classification.

Alternate Splitting Criteria

- **Variance impurity**:

$$i(N) = P(\omega_1)P(\omega_2)$$

because this is related to the variance of a distribution associated with the two classes.

- **Gini Impurity (generalization of the variance impurity):**

$$i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = \frac{1}{2}[1 - \sum_j P^2(\omega_j)]$$

The expected error rate at node *N* if the category label is selected randomly from the class distribution present at node *N*.

- **Misclassification impurity**:

$$i(N) = 1 - \max_j P(\omega_j)$$

measures the minimum probability that a training pattern would be misclassified at node *N*.

- **In practice, simple entropy splitting (choosing the question that splits the data into two classes of equal size) is very effective.**

Choosing a Question

- **An obvious heuristic is to choose the query that maximizes the decrease in impurity:** $$\Delta i(N) = i(N) - P_L(N_L)i(N_L) - (1-P_L)i(N_R)$$
  where N_L and N_R are the left and right descendant nodes, i(N_L) and i(N_R) are their respective impurities, and P_L is the fraction of patterns at node N that will be assigned to N_L when query T_i is chosen.

- **Then the "best" test value s is the choice for T that maximizes Δi(T).**
- **This approach is considered part of a class of algorithms known as "greedy."**
- **Note this decision is "local" and does not guarantee an overall optimal tree.**
- **A multiway split can be optimized using the gain ratio impurity:**
$$\Delta i^*(s) = \max_B \frac{\Delta i(s)}{-\sum_{k-1}^{B} P_k \log P_k}$$
  where P_k is the fraction of training patterns sent to node N_k, and B is the number of splits, and:
$$\Delta i(s) = i(N) - \sum_{k=1}^{B} P_k i(N_k)$$

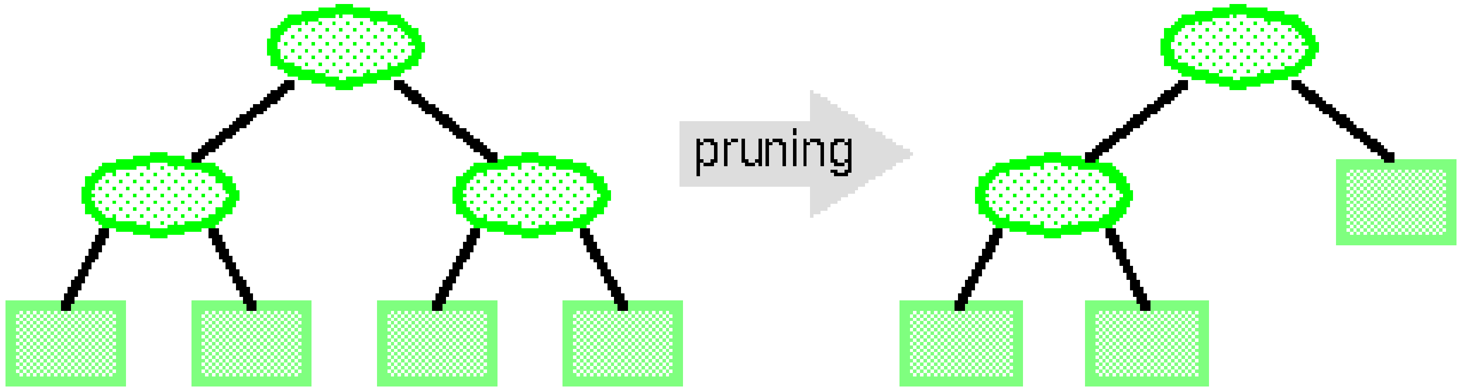
Generalization of $\Delta i(N)$.

When To Stop Splitting

- **If we continue to grow the tree until each leaf node has the lowest impurity, then the data will be overfit.**
- **Two strategies: (1) stop tree from growing or (2) grow and then prune the tree.**
- **A traditional approach to stopping splitting relies on *cross-validation*:**
  - ***Validation*: train a tree on 90% of the data and test on 10% of the data (referred to as the held-out set).**
  - ***Cross-validation*: repeat for several independently chosen partitions.**
  - ***Stopping Criterion*: Continue splitting until the error on the held-out data is minimized.**
- ***Reduction In Impurity*: stop if the candidate split leads to a marginal reduction of the impurity (drawback: leads to an unbalanced tree).**
- ***Cost-Complexity*: use a global criterion function that combines size and impurity: $\alpha \cdot size + \sum_{leaf\ nodes} i(N)$. This approach is related to *minimum description length* when the impurity is based on entropy.**
- **Other approaches based on *statistical significance* and *hypothesis testing* attempt to assess the quality of the proposed split.**

Pruning

- **The most fundamental problem with decision trees is that they "overfit" the data and hence do not provide good generalization. A solution to this problem is to prune the tree:**



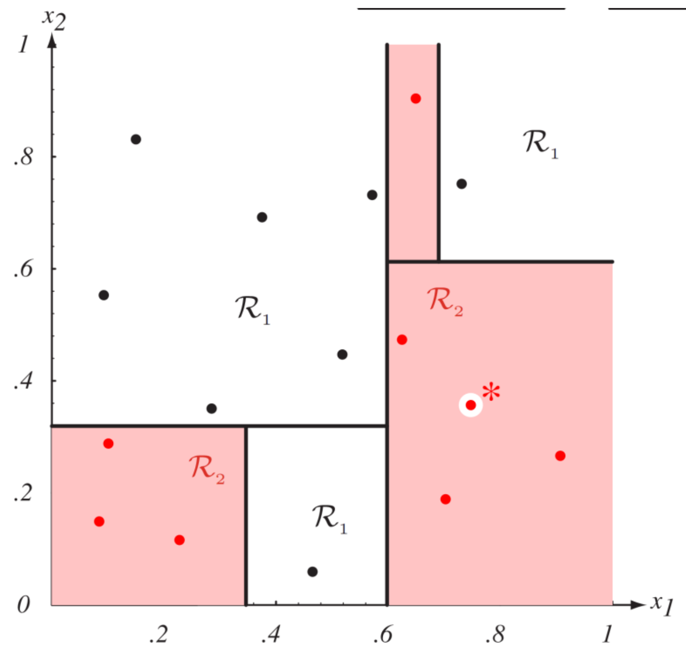
- **But pruning the tree will always increase the error rate on the training set ☹.**
- **Cost-complexity Pruning:** $\alpha \cdot size + \sum_{leaf\ nodes} i(N)$**. Each node in the tree can be classified in terms of its impact on the cost-complexity if it were pruned. Nodes are successively pruned until certain heuristics are satisfied.**
- **By pruning the nodes that are far too specific to the training set, it is hoped the tree will have better generalization. In practice, we use techniques such as cross-validation and held-out training data to better calibrate the generalization properties**.

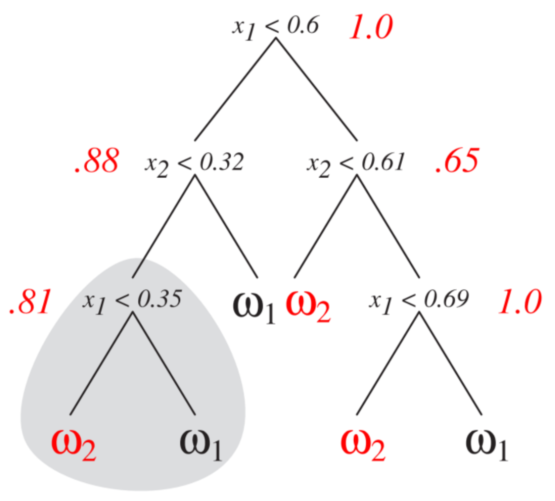
Example 1: A simple tree classifier

- Consider the following *n* = 16 points in two dimensions for training a binary CART tree (*B* = 2) using the entropy impurity (Eq. 1).

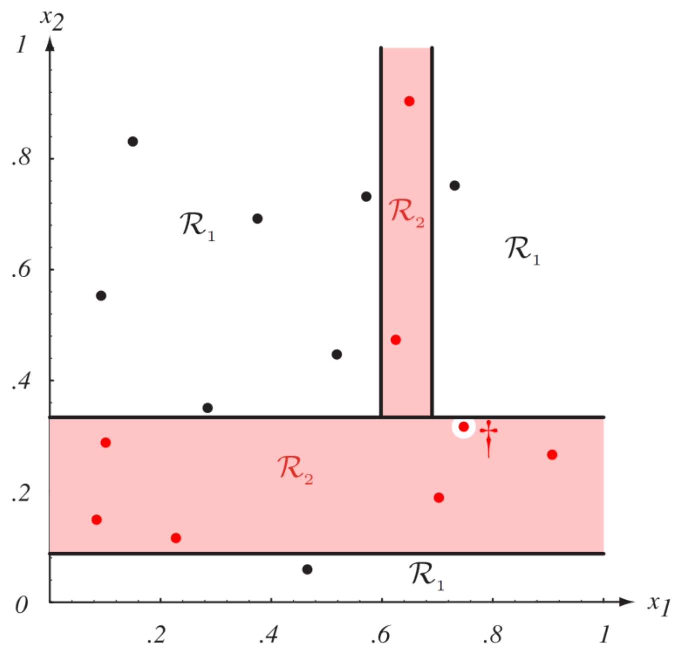
ω_1 (black)	
x_1	x_2
.15	.83
.09	.55
.29	.35
.38	.70
.52	.48
.57	.73
.73	.75
.47	.06

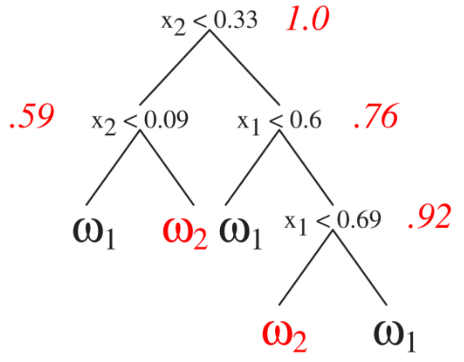
ω_2 (red)	
x_1	x_2
.10	.29
.08	.15
.23	.16
.70	.19
.62	.47
.91	.27
.65	.90
.75	.36* (.32†)





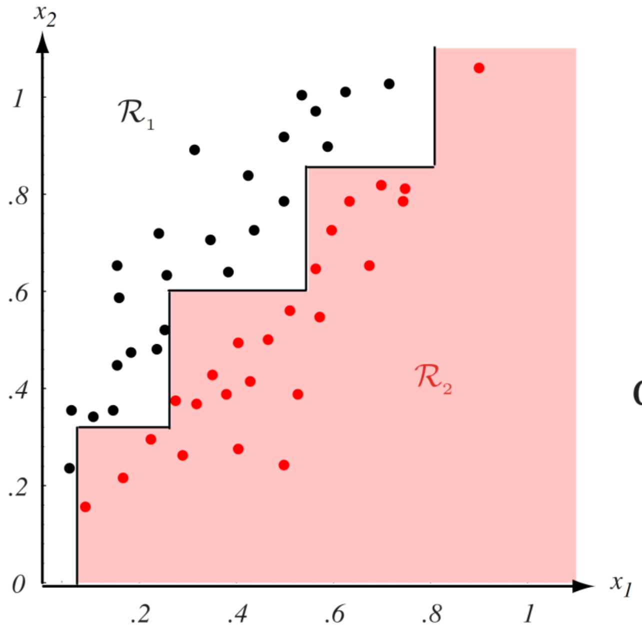
Entropy impurity at nonterminal nodes is shown in red and impurity at each leaf node is 0

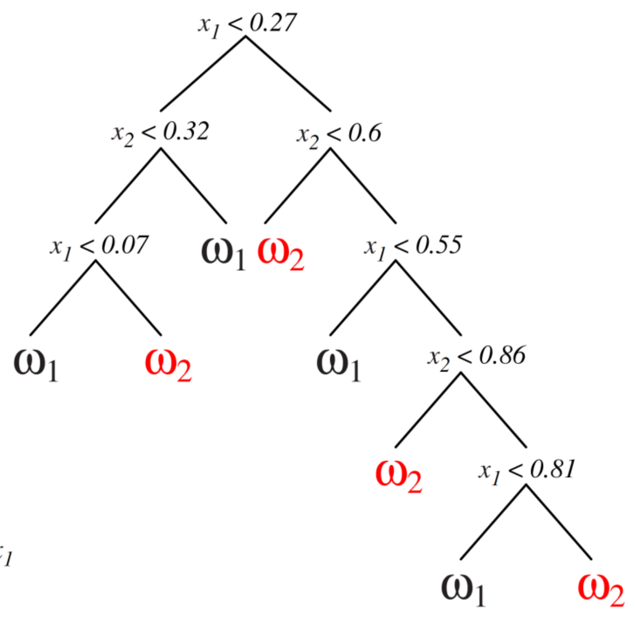


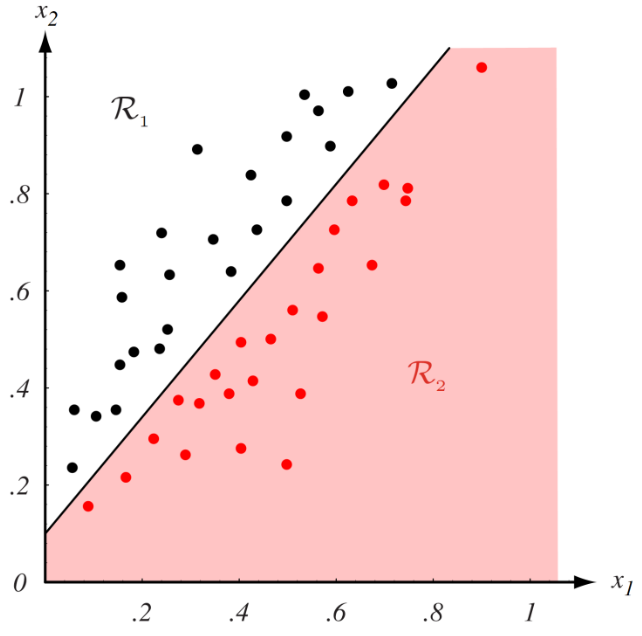


Instability or sensitivity of tree to training points; alteration of a single point leads to a very different tree; this is due to discrete & greedy nature of CART

Feature choice







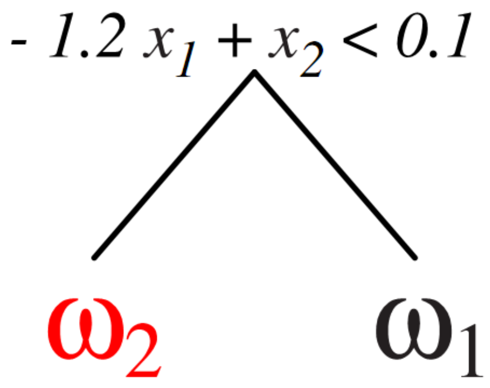


Figure 8.5: If the class of node decisions does not match the form of the training data, a very complicated decision tree will result, as shown at the top. Here decisions are parallel to the axes while in fact the data is better split by boundaries along another direction. If however "proper" decision forms are used (here, linear combinations of the features), the tree can be quite simple, as shown at the bottom.

ID3 and C4.5

- **Third Interactive Dichotomizer (ID3) uses nominal inputs and allows node-specific number of branches, B_j. Growing continues until all nodes as pure.**
- **C4.5, the successor to ID3, is one of the most popular decision tree methods:**
  - **Handles real-valued variables;**
  - **Allows multiway splits for nominal data;**
  - **Splitting based on maximization of the information gain ratio while preserving better than average information gain;**
  - **Stopping based on node purity;**
  - **Pruning based on confidence/average node error rate (pessimistic pruning).**
- **Bayesian methods and other common modeling techniques have been successfully applied to decision trees.**

ID3

- A greedy algorithm for decision tree construction developed by Ross Quinlan, 1987
- Top-down construction of the decision tree by recursively selecting the “best attribute” to use at the current node in the tree
  - Once the attribute is selected for the current node, generate children nodes, one for each possible value of the selected attribute
  - Partition the examples using the possible values of this attribute, and assign these subsets of the examples to the appropriate child node
  - Repeat for each child node until all examples associated with a node are either all positive or all negative

Choosing the best attribute

- The key problem is choosing which attribute to split a given set of examples
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected information gain–i.e., the attribute that will result in the smallest expected size of the subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

How well does it work?

Many case studies have shown that decision trees are at least as accurate as human experts.

- A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time; the decision tree classified 72% correct
- British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example

Example 1

Consider the following table

Day	Outlook	Temp.	Humidity	Wind	Play 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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

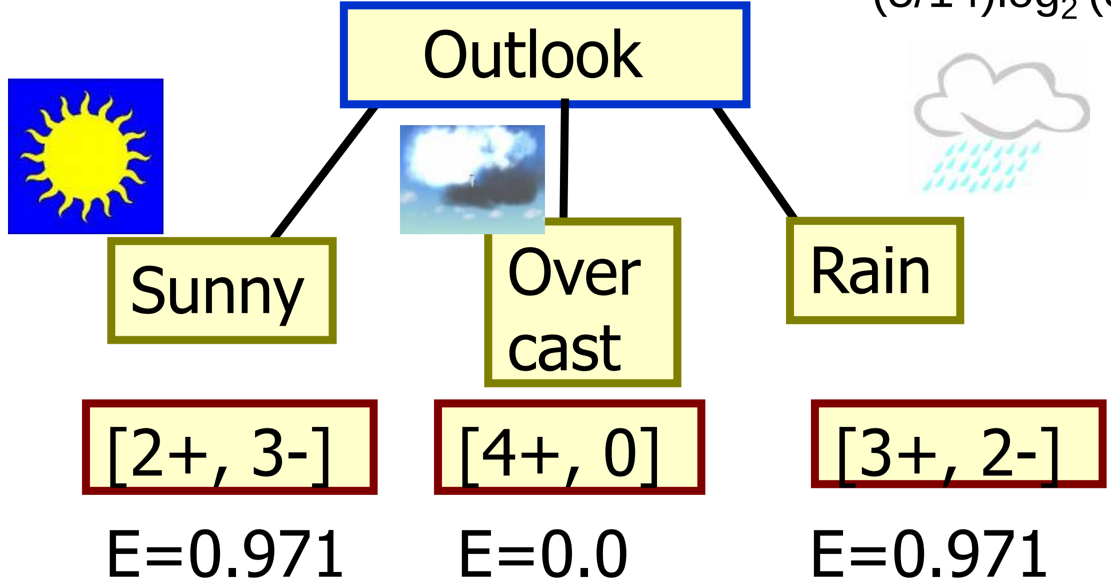
Example 2

- We want to build a decision tree for the tennis matches
- The schedule of matches depend on the weather (Outlook, Temperature, Humidity, and Wind)
- Calculating the information gains for each of the weather attributes:
  - For the Outlook
  - For the Temperature
  - For the Humidity
  - For the Wind

For the Outlook

[positive, Negative] $\nearrow$ S=[9+,5-]

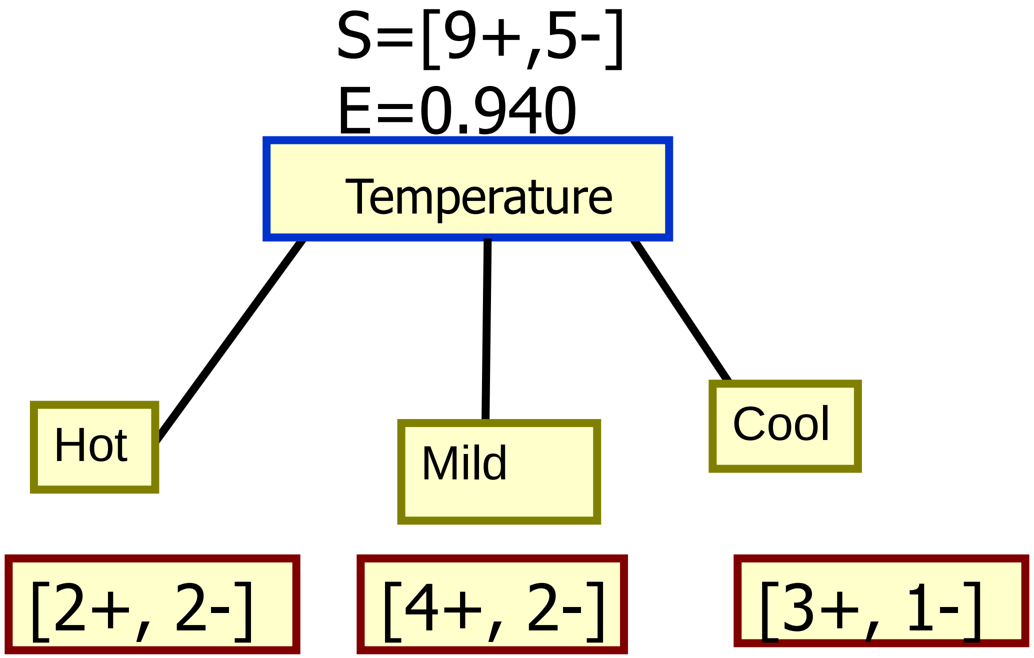
E=0.940 $\leftarrow$ Entropy(rootNode.subset) = -(9/14)$\log_2$(9/14) - (5/14)$\log_2$ (5/14)=0.940



Gain(S,Outlook)

=0.940-(5/14)*0.971 -(4/14)*0.0 – (5/14)*0.0971
=0.247

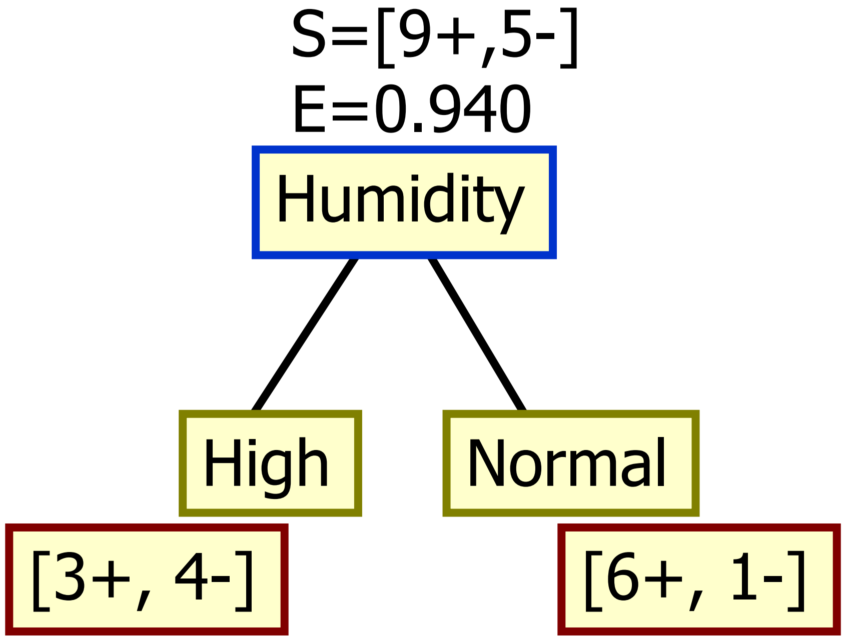
For the Temperature



Gain(S,Temperature)

=0.029

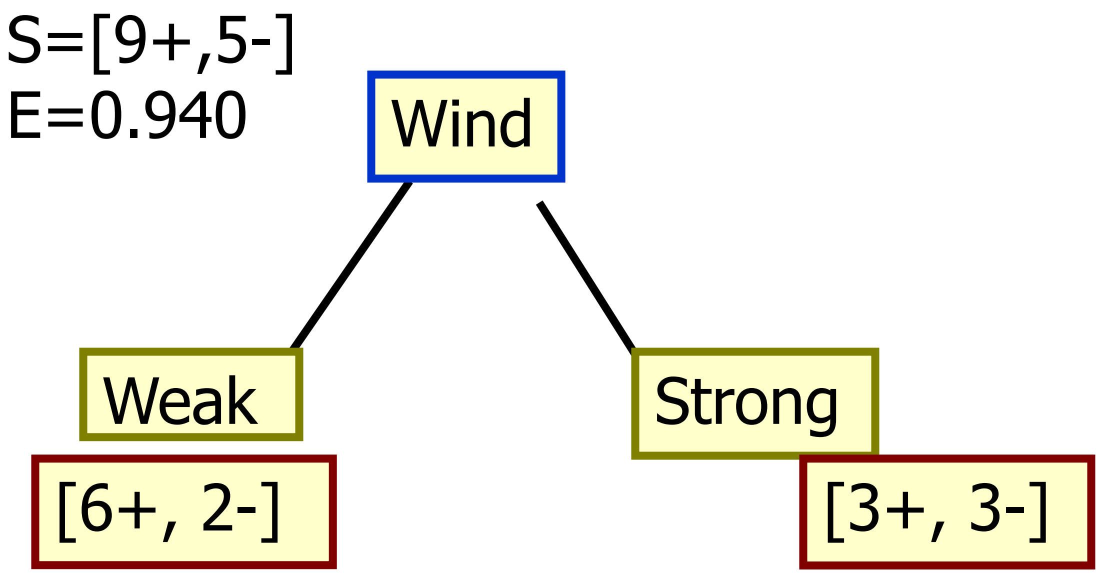
For the Humidity



Gain(S,Humidity)

=0.940-(7/14)*0.985 – (7/14)*0.592
=0.151

For the Wind



Gain(S,Wind):

=0.940 - (8/14)*0.811 - (6/14)*1.0
=0.048

Selecting the Next Attribute

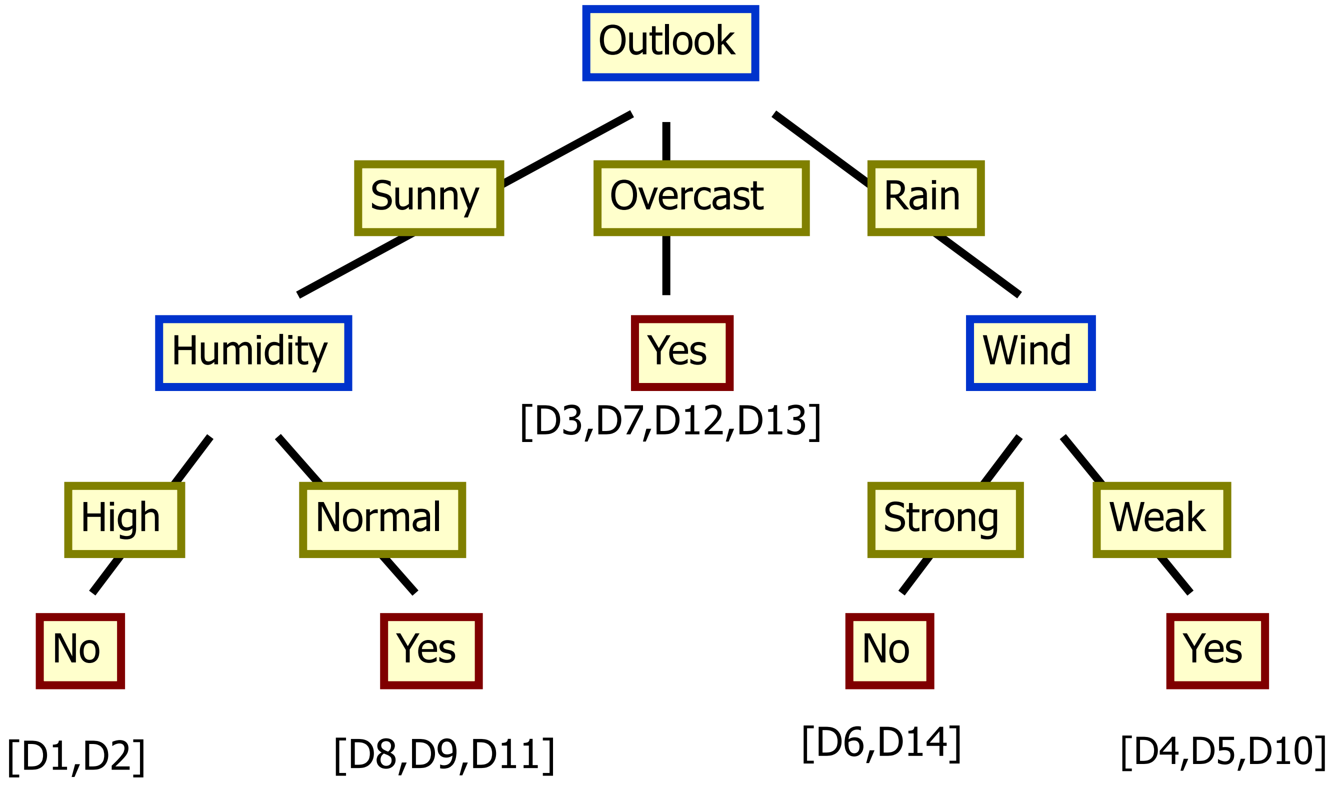
The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

where S denotes the collection of training examples

Complete tree

- Then here is the complete tree:

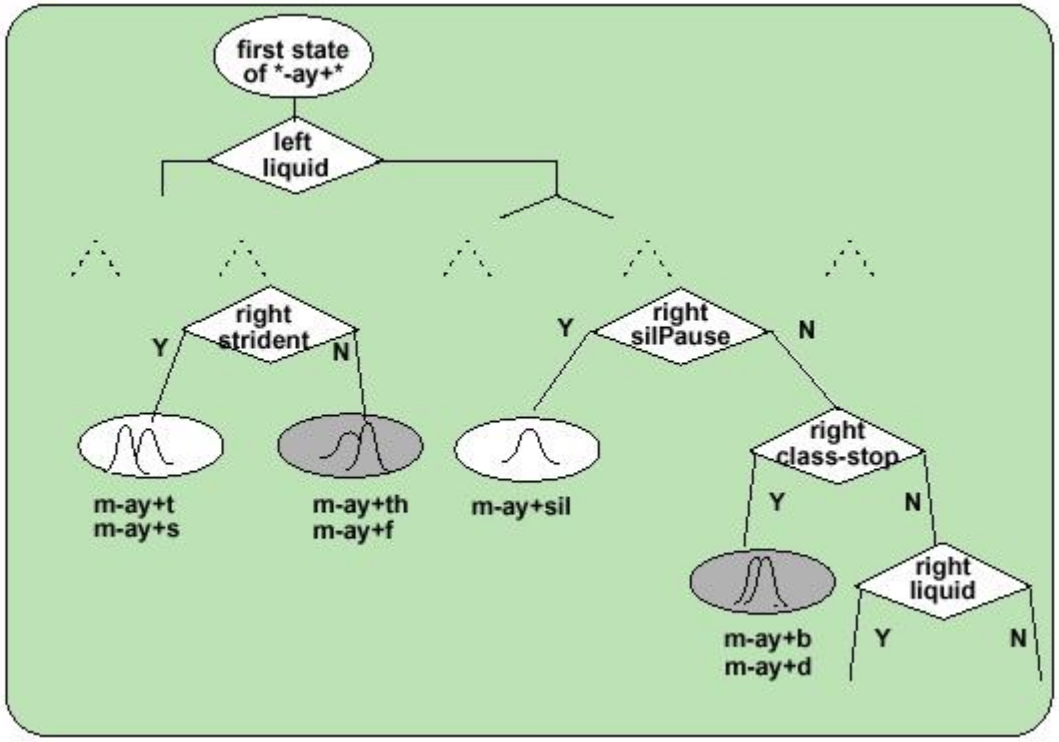


Extensions of the decision tree learning algorithm

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on.

Example Application:

- Decision trees are popular for many reasons including their ability to achieve high performance on closed-set evaluations.
- They can be closely integrated with hidden Markov models to provide a very powerful methodology for clustering and reducing complexity.
- Consider the problem in speech recognition of context-dependent phonetic modeling, which can potentially involve ten thousand acoustic models.
- On what basis should we cluster or reduce the number of acoustic models?
- The questions can be drawn from linguistics (e.g., vowel, consonant, sibilant).
- The tree growing process is intimately integrated into the Baum-Welch training process using the same likelihood calculations available during HMM parameter training.



Summary

- A classification and regression tree (CART) algorithm can be summarized as follows:
  - Create a set of questions that consists of all possible questions about the measured variables (phonetic context).
  - Select a splitting criterion (likelihood).
  - Initialization: create a tree with one node containing all the training data.
  - Splitting: find the best question for splitting each terminal node. Split the one terminal node that results in the greatest increase in the likelihood.
  - Stopping: if each leaf node contains data samples from the same class, or some pre-set threshold is not satisfied, stop. Otherwise, continue splitting.
  - Pruning: use an independent test set or cross-validation to prune the tree.
- these methods somewhat predate Bayesian methods.
- Decision trees can be used in many ways and closely integrated with other pattern recognition algorithms (e.g., hidden Markov models).
- They can be used to control complexity in a system by supporting decisions about parameter tying.
- Computational complexity is very low for both evaluation and training

- **We did not discuss:**
  - String matching
  - Grammatical methods
  - Grammatical inference
  - Rule-based methods