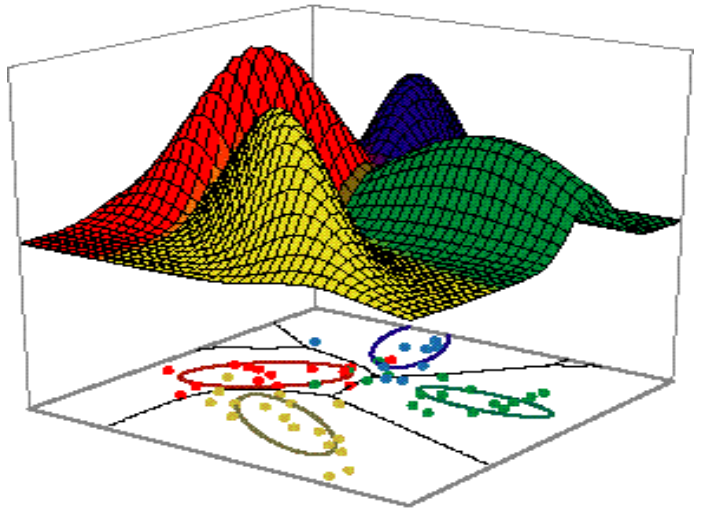
Part 10: Decision Trees



Introduction

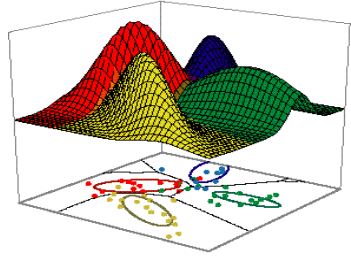
Training

Complexity, Pruning

CART vs. ID3 vs. C4.5

Some materials in these slides were taken from Pattern Classification (2nd ed) by R. O. Duda, P. E. Hart and D. G. Stork, John Wiley & Sons, 2000, **Chapter 8.1-8.4**.

Introduction

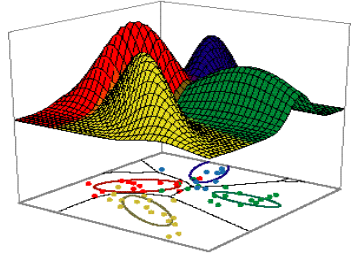


- Consider a classification problem where features are categorical or ordinal data
  - Cannot easily measure ‘distance’ between two feature vectors.
    - One-hot encoded features
  - Instead of feature vectors, consider lists of attributes
    - e.g., represent fruit using 4-tuple {red, shiny, sweet, small}
  - Turn to non-metric methods such as decision trees, rule-based systems, grammars, etc.
- Decision trees simple but effective method of pattern classification
  - Similar accuracy as artificial neural networks (ANN), or k- nearest neighbor (K-NN)
- Clear interpretation of the resulting classifier
  - As opposed to ANN’s…
  - Can express as rules
    - e.g., `if shape=thin AND colour=yellow` → `banana`
  - Analogous to “20 questions” game.

Visualizing a Decision Tree - Machine Learning Recipes #2 (7 min)

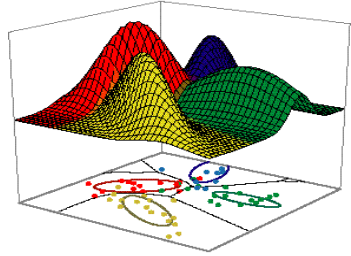
By Google Developers *https://www.youtube.com/watch?v=tNa99PG8hR8*

Training DTs



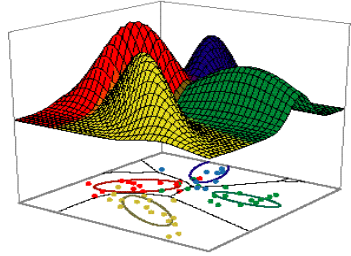
- How to train/build a DT from sample training data?
- Once we have labelled training data, we have already decided on which features will be measured.
  - How to organize in a tree?
  - Adding tests (i.e. nodes) divides our training data into subsets.
  - Ideal if each subset had the same class ID
    - i.e. the node becomes “pure”
  - However, not typically the case. Therefore must either:
    - Decide to stop splitting and accept imperfect decision
    - Or, find another property to split on.

Classification Based on DT



- Classification of a sample from a decision tree (DT) is straightforward:
  - Start at the root node.
  - Apply the prescribed test on the feature contained in the root node.
  - Proceed down to the correct child
  - If it is a leaf node, apply class label for that node. Done.
  - Else, apply the prescribed test at that node.
  - Continue down tree in this manner until leaf reached.
- One advantage: not all features must be measured to make a decision in some cases.
  - Big advantage when tests are costly (e.g. medical tests)
- For numerical data, decision boundaries are parallel to feature axes (see next slide):

Example Decision Tree



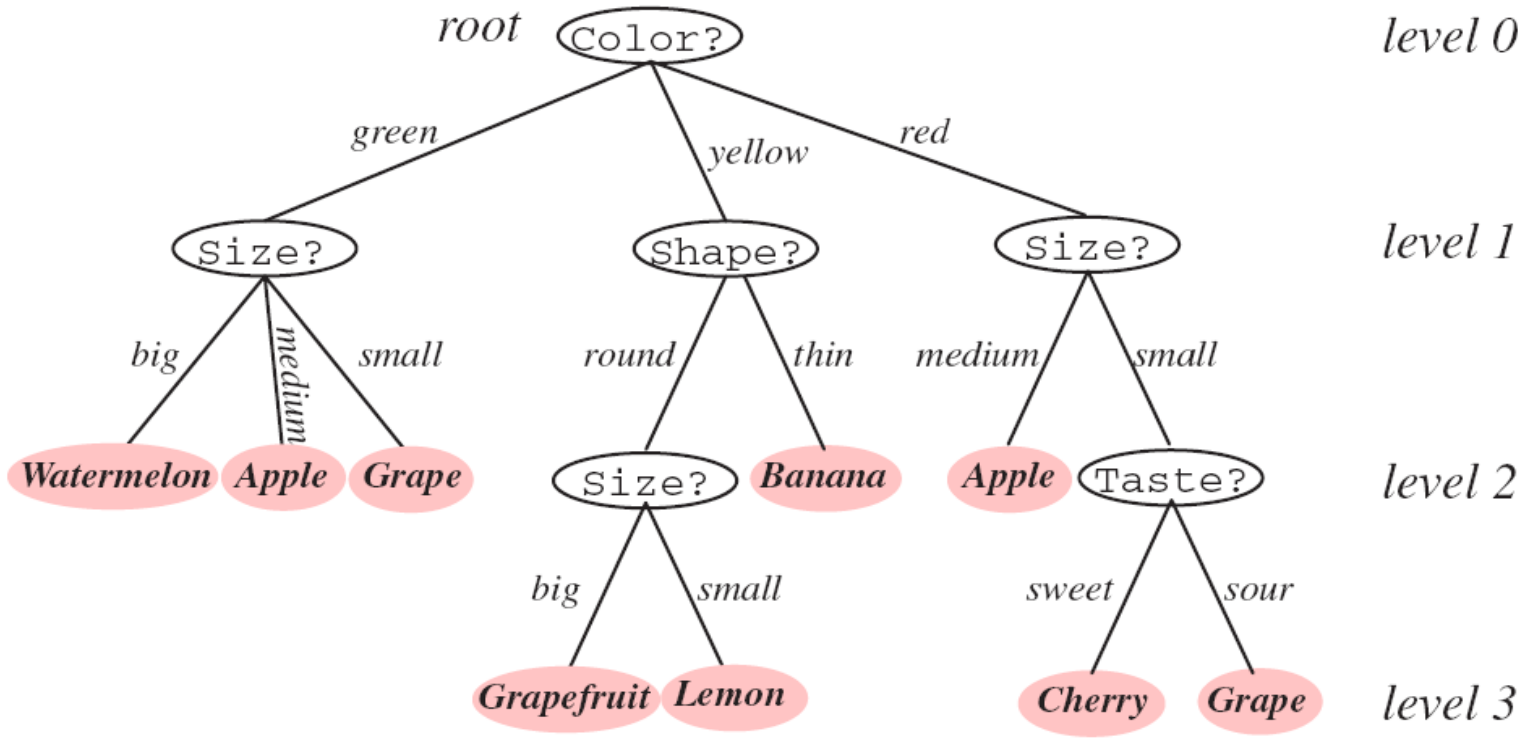


FIGURE 8.1. Classification in a basic decision tree proceeds from top to bottom. The questions asked at each node concern a particular property of the pattern, and the downward links correspond to the possible values. Successive nodes are visited until a terminal or leaf node is reached, where the category label is read. Note that the same question, `Size?`, appears in different places in the tree and that different questions can have different numbers of branches. Moreover, different leaf nodes, shown in pink, can be labeled by the same category (e.g., **Apple**). From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*.

Decision Boundaries

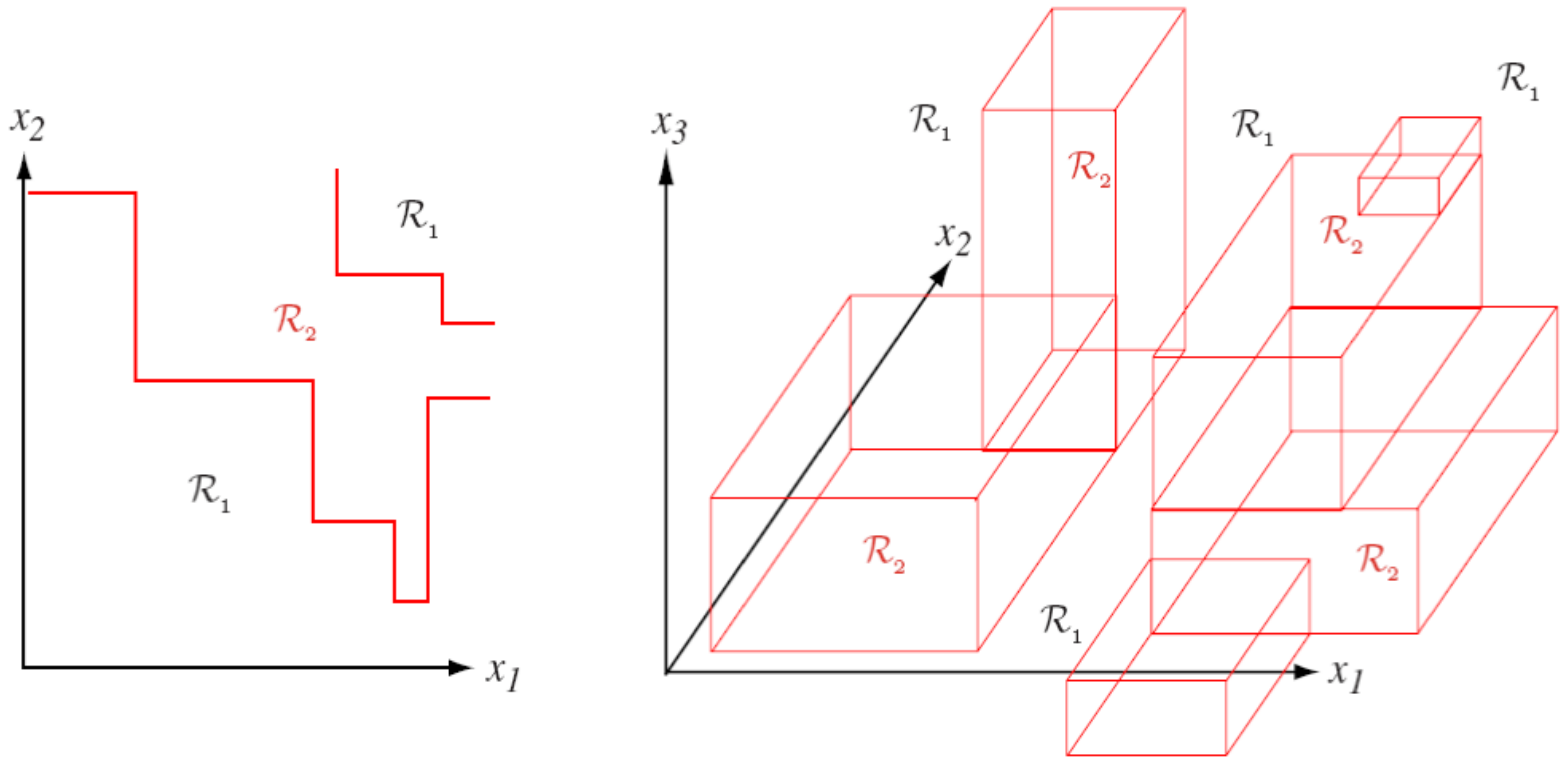
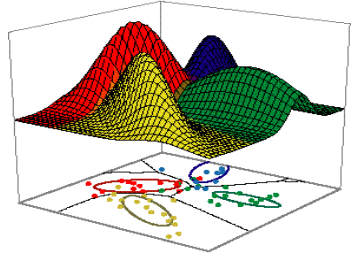


FIGURE 8.3. Monothetic decision trees create decision boundaries with portions perpendicular to the feature axes. The decision regions are marked $\mathcal{R}_1$ and $\mathcal{R}_2$ in these two-dimensional and three-dimensional two-category examples. With a sufficiently large tree, any decision boundary can be approximated arbitrarily well in this way. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Number of splits



- Decision outcome at each node is called a *split*
  - Splits data into subsets
- *Branching factor/ratio* (B) is number of children from a node.
  - Can vary through tree, or be fixed.
  - Determining number of splits at a node is closely related to deciding which feature to split on
  - Any tree can be represented as a binary tree where each node is split into two (B=2)
    - For example, see next slide

Binary tree structure

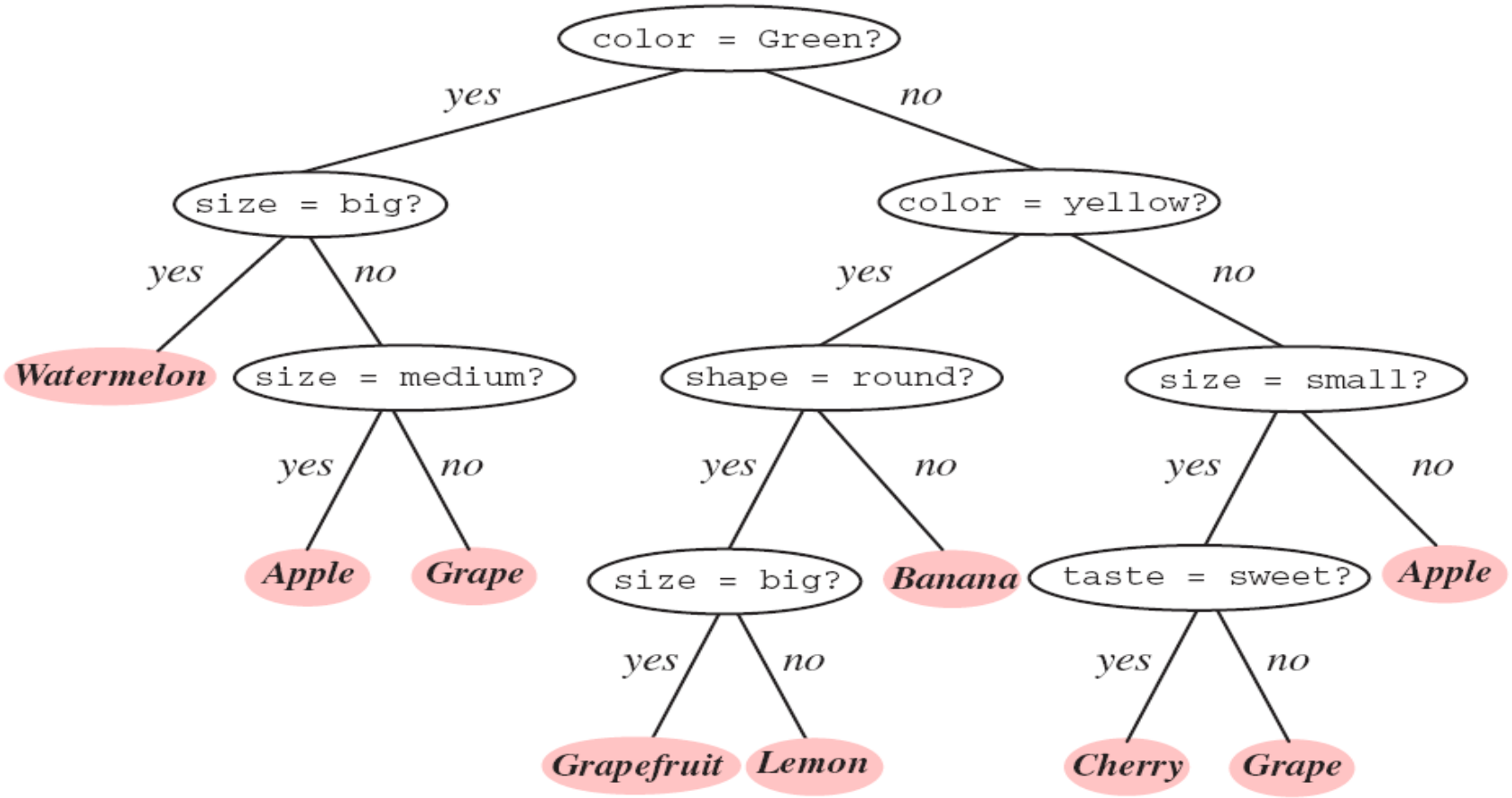
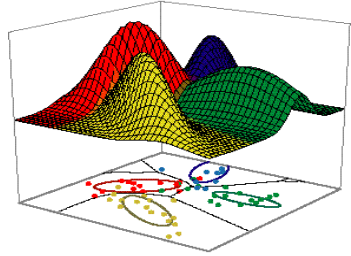
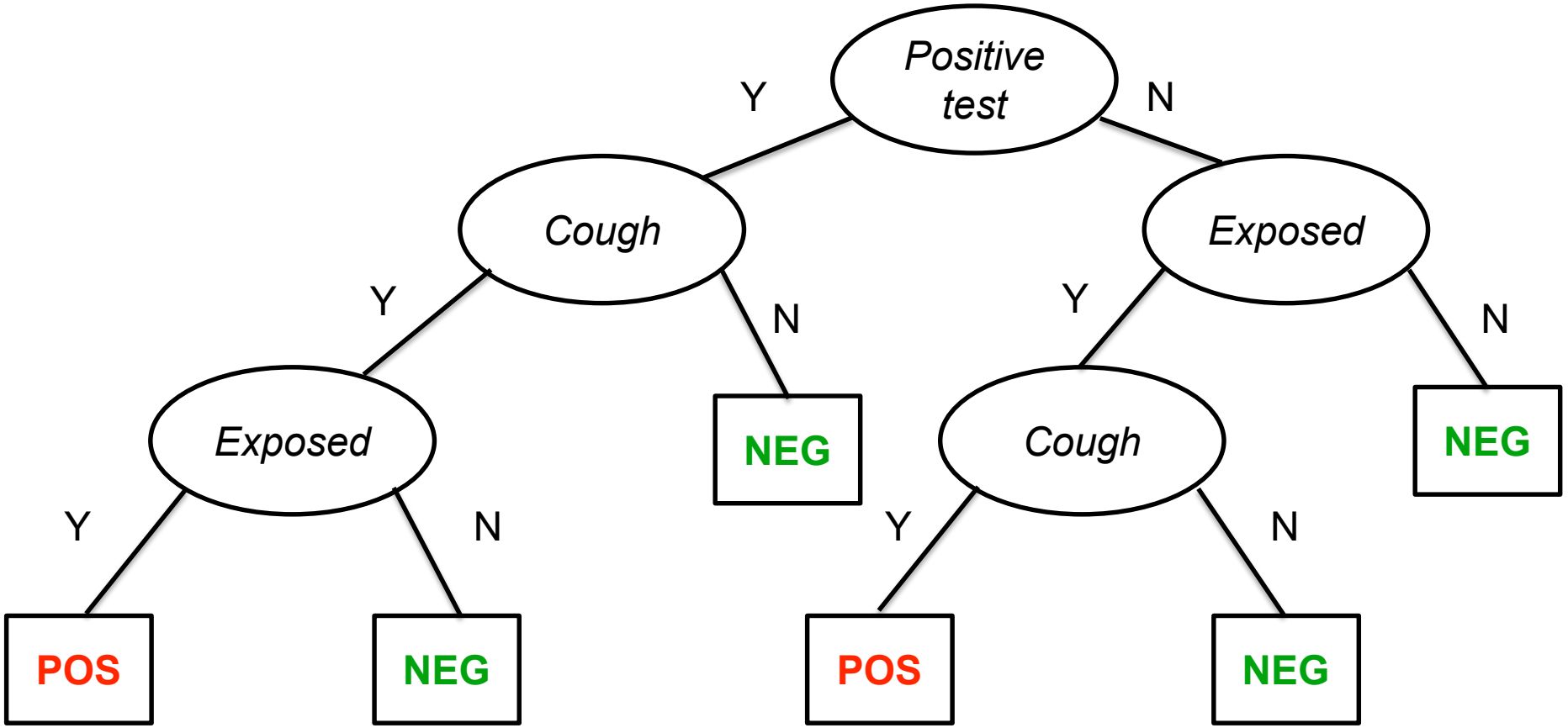


FIGURE 8.2. A tree with arbitrary branching factor at different nodes can always be represented by a functionally equivalent binary tree—that is, one having branching factor $B = 2$ throughout, as shown here. By convention the "yes" branch is on the left, the "no" branch on the right. This binary tree contains the same information and implements the same classification as that in Fig. 8.1. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Classifying with a decision tree (COVID)

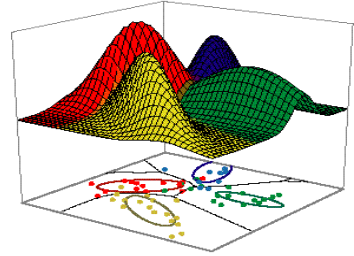


x = [exposed, negative test, cough]



x = [exposed, negative test, cough]

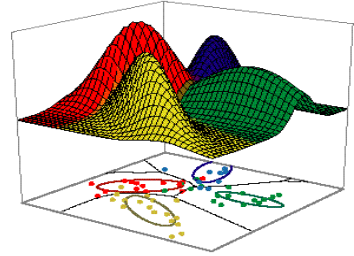
Training DTs: CART



- CART (classification and regression trees)
  - DT training framework based on 6 questions:

1) Should the properties be restricted to binary-valued or allowed to be multi-valued?
   - How many decision outcomes or splits will there be at a node? (branching factor of the node limited to 2 for binary)
2) Which property should be tested at a node?
3) When should a node be declared a leaf?
4) If the tree becomes ‘too large’, how can it be made smaller/simpler via pruning?
5) If a leaf node is impure, how should the category label be assigned?
6) How should missing data be handled?

Query Selection & Node Impurity



- Which feature to query at each node?
  - Want a simple, compact tree with few nodes
    - (Occam's razor: simplest tree)
  - Therefore each node should be chosen such that child subsets be as 'pure' as possible.
- First, define *impurity* at node N:
  - Should be maximal for equal mix; reach zero for purely one class
  - a) Entropy impurity: $i(N) = -\sum_j P(\omega_j)\log_2 P(\omega_j)$

    $P(\omega_j)$ = probability of Class j in the data @ node N
  - b) Variance impurity *(2 class case)*:

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

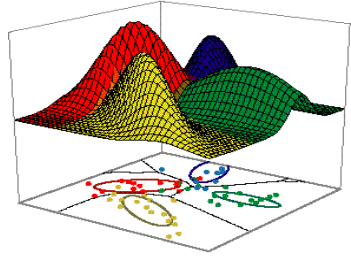
    $$i(N) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = \frac{1}{2}\left[1 - \sum P^2(\omega_j)\right]$$

    - Extend to multiple classes using *Gini impurity*

      $$i(N) = 1 - \max_j P(\omega_j)$$
  - c) Misclassification impurity:
    - Min probability that training pattern misclassified at N

      Equal to probability of choosing wrong class randomly at node N

Measures of Node Impurity



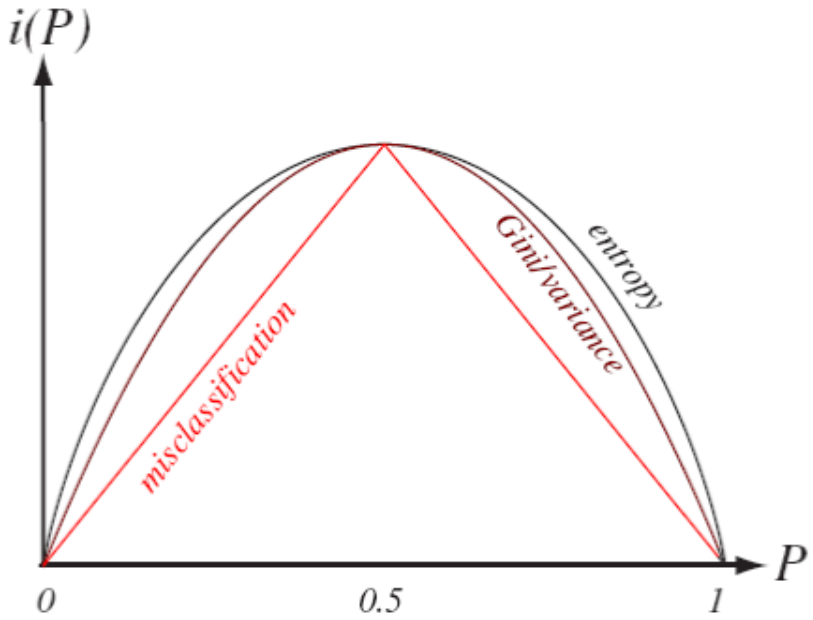
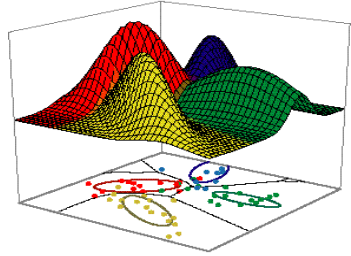


FIGURE 8.4. For the two-category case, the impurity functions peak at equal class frequencies and the variance and the Gini impurity functions are identical. The entropy, variance, Gini, and misclassification impurities (given by Eqs. 1–4, respectively) have been adjusted in scale and offset to facilitate comparison here; such scale and offset do not directly affect learning or classification. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Query Selection & Node Impurity

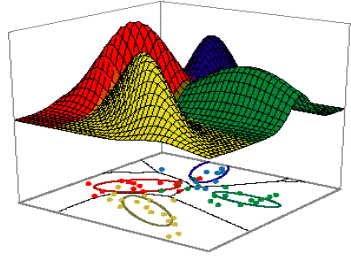


- Examine features to look for greatest drop in impurity.
  - Reduction in impurity at node N:

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

    - (P_L is fraction of patterns that go to N_L)
    - If entropy measure is used, reduction in impurity is **information gain** (limited to 1 bit for binary splits)
  - Also search for optimal split point, s, for node T
    - e.g. once we choose to split on “weight”, must also define threshold value for split point (e.g. if w<1.5kg → s=1.5)
  - Search simplified if you assume binary tree and tests are based on a single feature *(monothetic tree).*

Impurity reduction (sample calculation)



Dataset (fake cancer dataset)

Size	Growth rate	Class (ω)
Small	Slow	Neg (1)
Small	Fast	Pos (2)
Large	Slow	Neg (1)
Large	Slow	Pos (2)
Large	Slow	Neg (1)
Small	Fast	Neg (1)
Large	Slow	Pos (2)
Large	Fast	Pos (2)
Small	Fast	Pos (2)
Small	Slow	Neg (1)
Small	Slow	Neg (1)
Small	Slow	Pos (2)
Large	Slow	Neg (1)
Small	Fast	Neg (1)

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$

Root node

1. Compute $i(N)$ *(entropy impurity)*

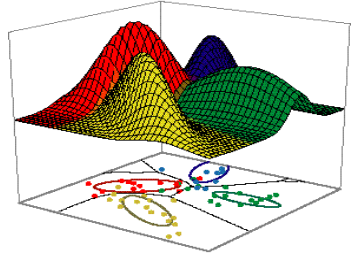
$$i(N) = -\sum_j P(\omega_j)\log_2 P(\omega_j)$$

$$i(N) = -(P_1 \log_2 P_1 + P_2 \log_2 P_2)$$

$$i(N) = -\left(\left(\frac{8}{14}\right)\log_2\left(\frac{8}{14}\right) + \left(\frac{6}{14}\right)\log 2\left(\frac{6}{14}\right)\right)$$

$$i(N) = 0.985$$

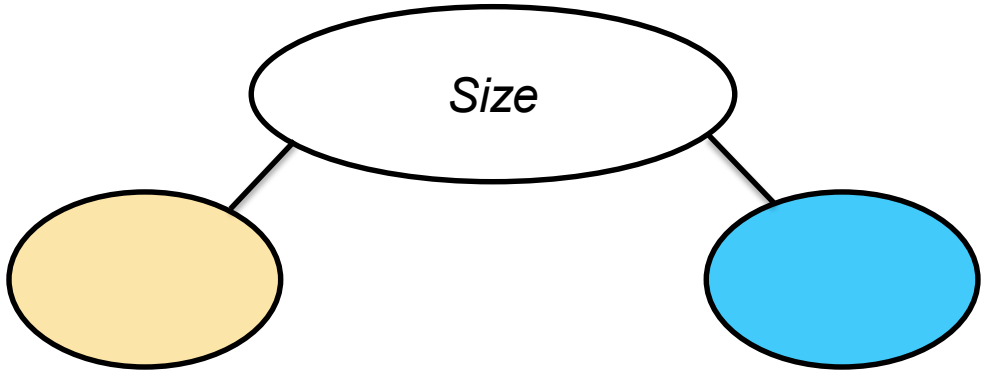
Impurity reduction (sample calculation)



Dataset (fake cancer dataset)

Size	Growth rate	Class (ω)
Small	Slow	Neg (1)
Small	Fast	Pos (2)
Large	Slow	Neg (1)
Large	Slow	Pos (2)
Large	Slow	Neg (1)
Small	Fast	Neg (1)
Large	Slow	Pos (2)
Large	Fast	Pos (2)
Small	Fast	Pos (2)
Small	Slow	Neg (1)
Small	Slow	Neg (1)
Small	Slow	Pos (2)
Large	Slow	Neg (1)
Small	Fast	Neg (1)

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$



2. Compute $P_L i(N_L)$ and $P_R i(N_R)$ if we split by size

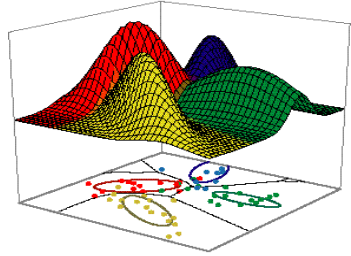
$$i(N_L) = -\sum_j P(\omega_j)\log_2 P(\omega_j)$$

$$i(N_L) = -(P_1 \log_2 P_1 + P_2 \log_2 P_2)$$

$$i(N_L) = -\left(\left(\frac{5}{8}\right)\log_2\left(\frac{5}{8}\right) + \left(\frac{3}{8}\right)\log 2\left(\frac{3}{8}\right)\right)$$

$$i(N_L) = 0.954$$

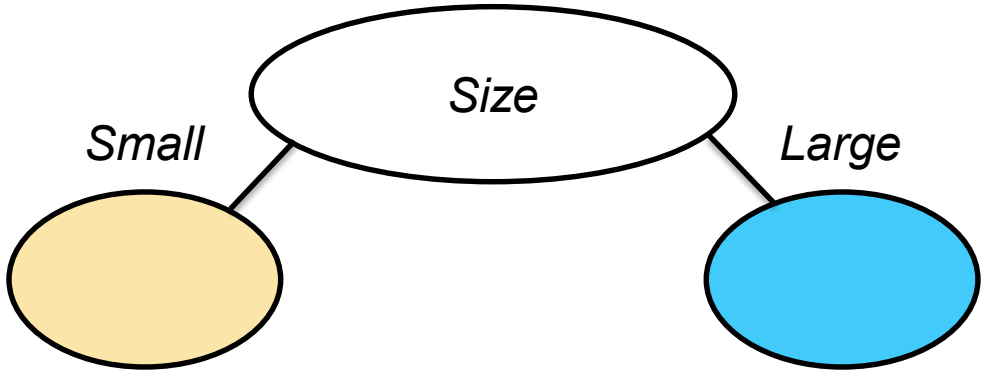
Impurity reduction (sample calculation)



Dataset (fake cancer dataset)

Size	Growth rate	Class (ω)
Small	Slow	Neg (1)
Small	Fast	Pos (2)
Large	Slow	Neg (1)
Large	Slow	Pos (2)
Large	Slow	Neg (1)
Small	Fast	Neg (1)
Large	Slow	Pos (2)
Large	Fast	Pos (2)
Small	Fast	Pos (2)
Small	Slow	Neg (1)
Small	Slow	Neg (1)
Small	Slow	Pos (2)
Large	Slow	Neg (1)
Small	Fast	Neg (1)

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$



2. Compute $P_L i(N_L)$ and $P_R i(N_R)$ if we split by size

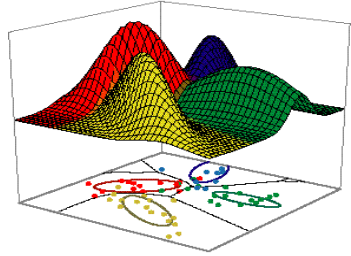
$$i(N_R) = -\sum_j P(\omega_j)\log_2 P(\omega_j)$$

$$i(N_R) = -(P_1 \log_2 P_1 + P_2 \log_2 P_2)$$

$$i(N_R) = -\left(\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) + \left(\frac{3}{6}\right)\log 2\left(\frac{3}{6}\right)\right)$$

$$i(N_R) = 1$$

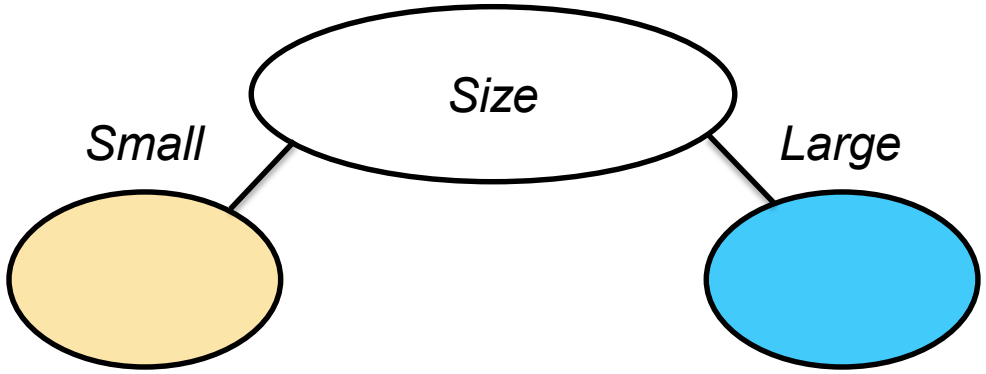
Impurity reduction (sample calculation)



Dataset (fake cancer dataset)

Size	Growth rate	Class (ω)
Small	Slow	Neg (1)
Small	Fast	Pos (2)
Large	Slow	Neg (1)
Large	Slow	Pos (2)
Large	Slow	Neg (1)
Small	Fast	Neg (1)
Large	Slow	Pos (2)
Large	Fast	Pos (2)
Small	Fast	Pos (2)
Small	Slow	Neg (1)
Small	Slow	Neg (1)
Small	Slow	Pos (2)
Large	Slow	Neg (1)
Small	Fast	Neg (1)

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R)$$



3. Compute $\Delta i(N)$

$$\Delta i(N) = i(N) - P_L i(N_L) - P_R i(N_R)$$

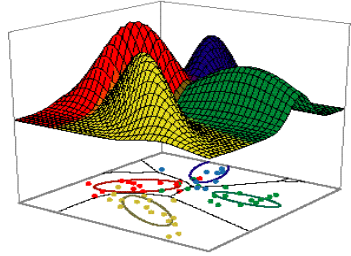
$$\Delta i(N) = (0.985) - (8/14)(0.954) - (6/14)(1)$$

$$\Delta i(N) = 0.011$$

I repeat the calculation if I split by growth rate and get:

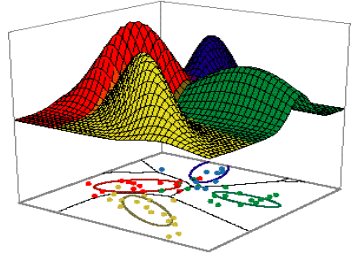
$$\Delta i(N) = 0.048$$

Query Selection & Node Impurity



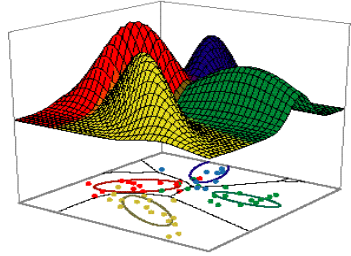
- May find a range of optimal split points
  - Typically choose median/mean of range.
- Greedy search – finds local optimum
  - Testing one node at a time, in isolation.
- Gini vs. misclassification impurity
  - Gini often preferred since it 'anticipates future splits'.
  - e.g. have 90 ω_1 and 10 ω_2 at node N. Assume no split point leads to ω_2 majority in either child → misclassification impurity unchanged at 0.1. Gini would prefer a split that leads to L={70 ω_1, 0 ω_2} and R={20 ω_1, 10 ω_2}
- In practice, pruning and stopping criteria have a bigger impact than impurity measure on final tree.

When to Stop Splitting



- If we grow until each leaf contains a single sample, will have perfect purity → almost certainly overfit.
- Several strategies:
  - 1) Make use of validation (hold-out) or n-fold cross-validation
    - Stop when minimum error reached on validation data
  - 2) Stop when Δi gets too small
    - Apply at each node (not global stop) → leads to different depths in different branches.
      - Leads to an unbalanced tree
  - 3) Stop when node represents very few training samples
    - e.g. fewer than 5% of total training data
    - Benefits analogous to k-NN (density of data defines decision partition size)

When to Stop Splitting



- Several strategies:
  - 4) Minimize tree complexity and total node impurity:

$$J = \alpha \cdot size + \sum_{\substack{leaf \\ nodes}} i(N)$$

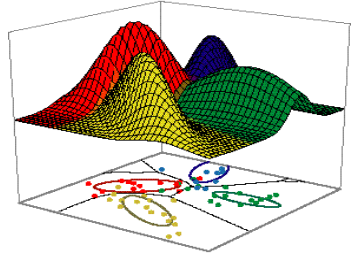
    - Related to MDL if entropy used for i(N): Total i(N) of all leaves measure uncertainty of data, given model represented by tree; size of tree is measure of model complexity.
  - 5) Test of significance
    - a) Form population of Δi(N) from tree. Only accept a new split if new Δi is *significantly* different from zero (χ^2 test)
    - b) Form null hypothesis that split is equivalent to random. Test if observed distribution of class labels is significantly different.

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}}$$

degrees of freedom = 1

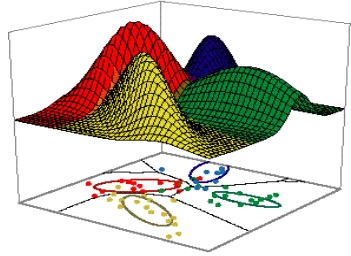
$n_{ie}=Pn_i$, n_{iL}= # ω_i sent left by proposed split

Pruning



- Can stop prematurely from lack of sufficient ‘look ahead’ – *horizon effect*
  - When determining whether to stop splitting, we don’t consider quality of splits at child nodes
  - Biases learning algorithm to trees with greatest impurity reduction is near the root node
- Alternative is pruning
  - Grow tree completely, then eliminate/prune/merge pairs of leaf nodes when gain in impurity < T
  - For large datasets, computational complexity of pruning may be too high. Otherwise, use it!
  - Can prune non-leaf nodes, replacing subtree with a leaf, or removing decision node and replacing with a child node.

Example 1: Stability of DT Training



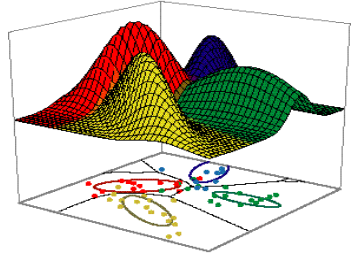
- Small changes in the training data can lead to large differences in final classification boundaries
- Example: Consider 16 training points with 2 features. Build a binary CART tree using entropy:

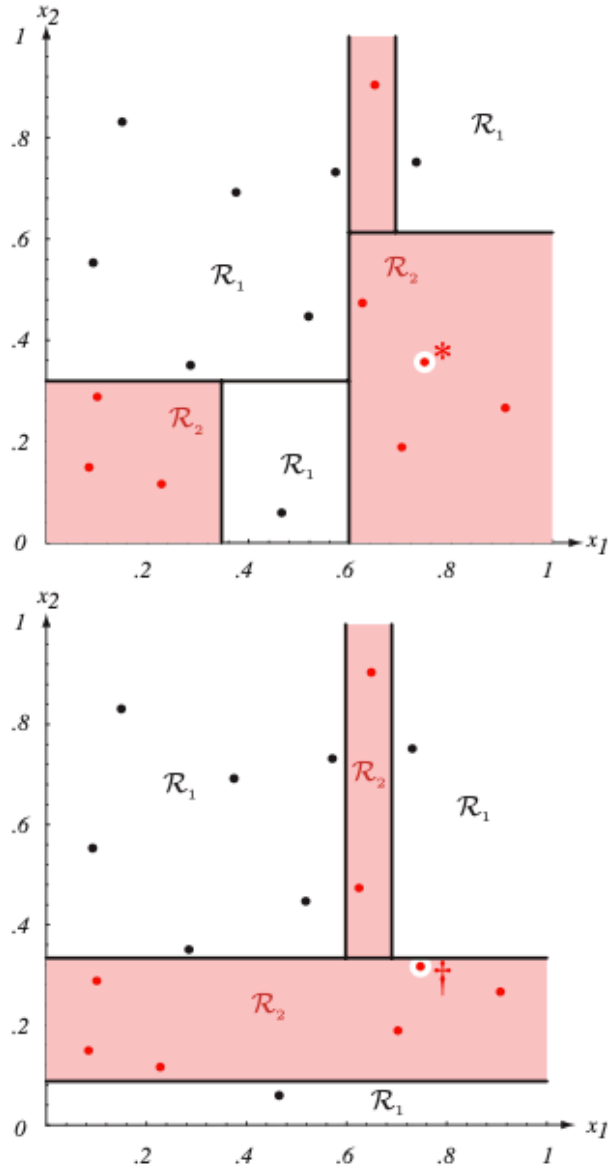
ω_1 (black)	
x_1	x_2
.15	.83
.09	.55
.29	.35
.38	.70
.52	.44
.57	.73
.73	.75
.47	.06

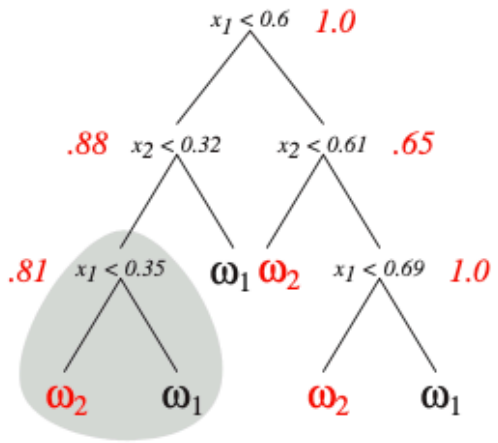
ω_2 (red)	
x_1	x_2
.10	.29
.08	.15
.23	.16
.70	.19
.62	.47
.91	.27
.65	.90
.75	.36* (.32†)

← *x_2 value could be .36 or .32*

Example 1 – Unpruned Trees

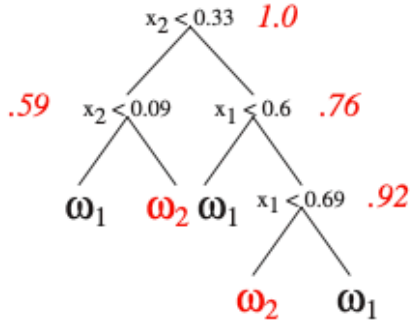




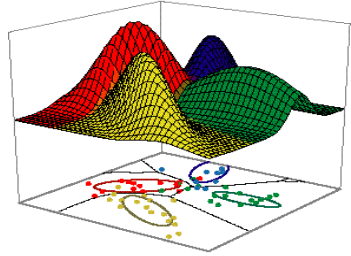


Tree 1: Assume x_2 was .36

Tree 2: Assume x_2 was .32



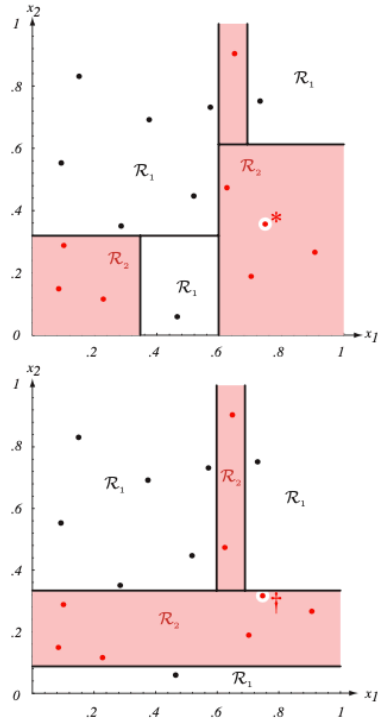
Example 1: Sample Calculations

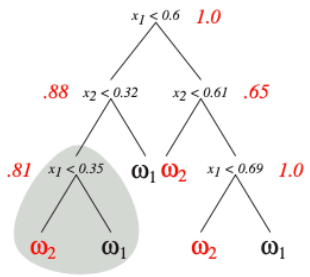


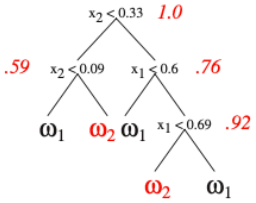
- Small change in one measured feature results in vastly different classification boundaries.
- Sample calculations:
  - Impurity at root node is:

$$i(N_{root}) = -\sum_{i=1}^{2} P(\omega_i)\log_2 P(\omega_i) = -[0.5\log_2(0.5) + 0.5\log_2(0.5)] = 1.0$$

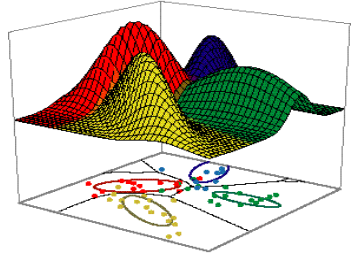
  - Split point of test @ root:
    - Try all n-1=15 possible split points in each dimension
    - Greatest reduction in impurity occurs near x_{1s}=0.6
  - If pruning were applied, shaded subtree (pair of leaf nodes) would be first deleted/merged/pruned
    - Would lead to smallest gain in impurity







Feature Selection, Multivariate DTs, & Unbalanced training sets



- Tree learning will not work well if individual features do not discriminate data well
  - See next slide
- Can preprocess data
  - E.g. run PCA first, then build tree on principal components.
- Otherwise can permit more complicated decisions at nodes, involving multiple features
  - See next, next slide.
- Unbalanced training set
  - Can use loss function to weight errors in under-represented class more heavily
  - Weighted Gini impurity: $i(N) = \sum_{i \neq j} \lambda_{ij} P(\omega_i) P(\omega_j)$

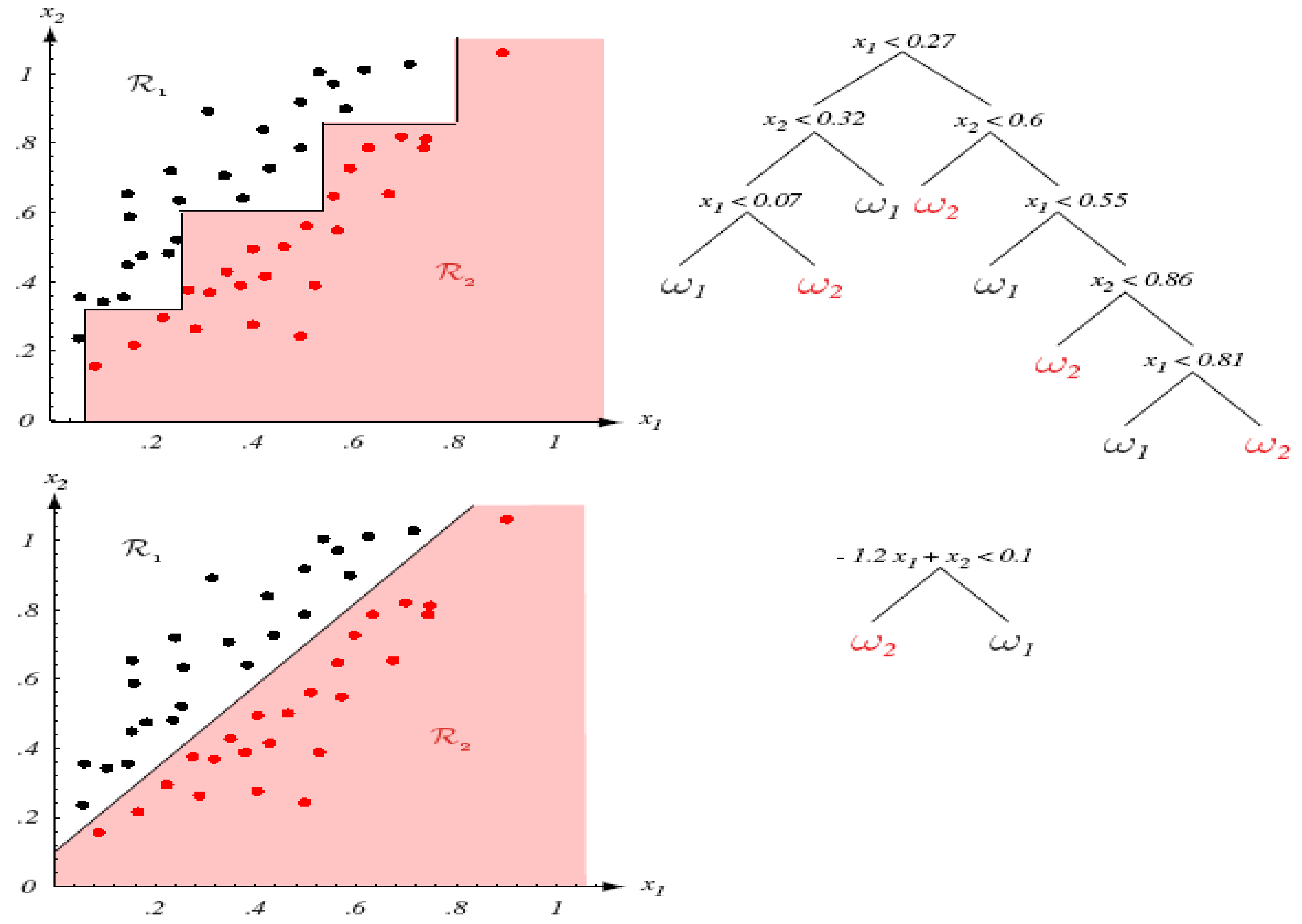
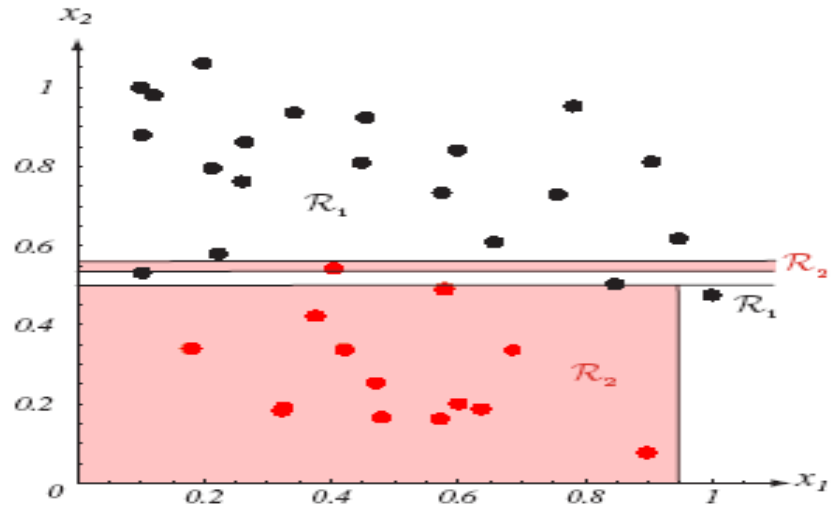
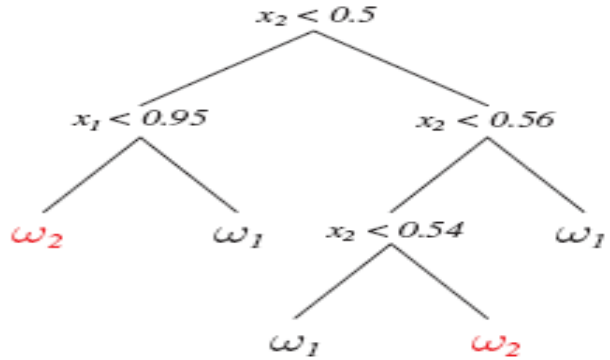
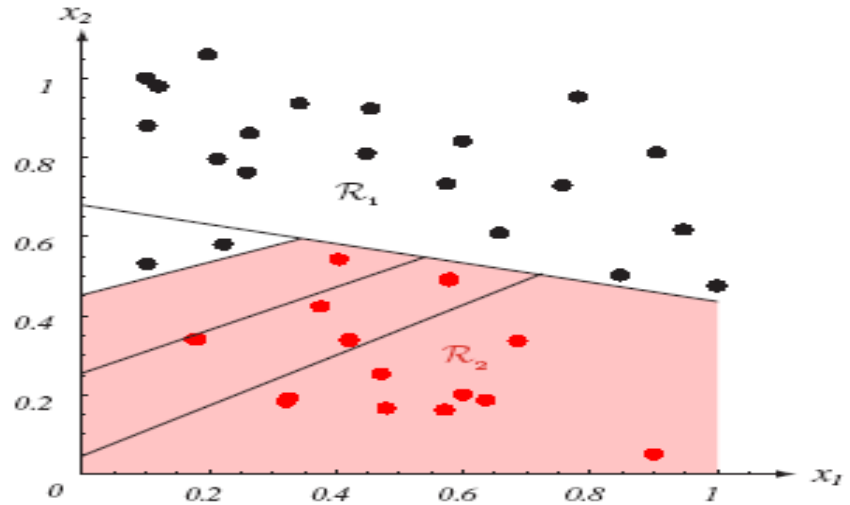
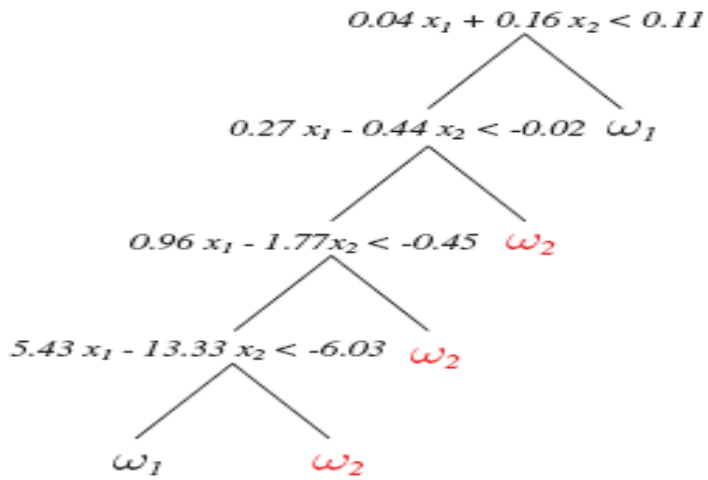


FIGURE 8.5. If the class of node decisions does not match the form of the training data, a very complicated decision tree will result, as shown at the top. Here decisions are parallel to the axes while in fact the data is better split by boundaries along another direction. If, however, "proper" decision forms are used (here, linear combinations of the features), the tree can be quite simple, as shown at the bottom. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.









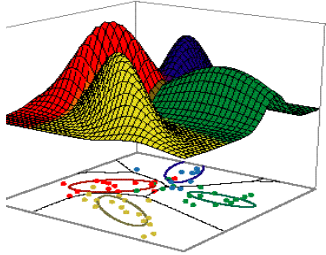
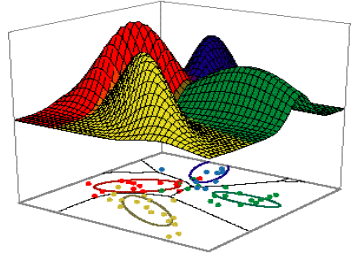


FIGURE 8.6. One form of multivariate tree employs general linear decisions at each node, giving splits along arbitrary directions in the feature space. In virtually all interesting cases the training data are not linearly separable, and thus the LMS algorithm is more useful than methods that require the data to be linearly separable, even though the LMS need not yield a minimum in classification error (Chapter 5). The tree at the bottom can be simplified by methods outlined in Section 8.4.2. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

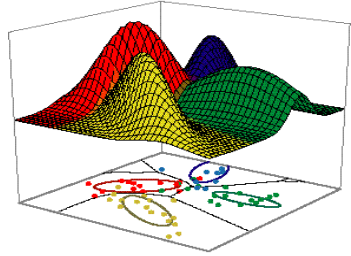
Missing Attributes



- May have missing attributes:
  - i) During training
    - Instead of throwing out *deficient patterns*, instead calculate impurity at each node using only attribute information that is present.
    - Calculate best split point using data available
  - ii) During classification
    - Use primary decision at a node whenever possible, use *alternative tests* when not available.
    - During training, in addition to identifying optimal split at each node, also provide *surrogate splits* (label & rule)
    - Maximize 'predictive association' with primary split
      - e.g. look for similar splits between left/right children
      - Analogous to replacing missing value by nonmissing attribute most correlated with it.

The fact that an attribute is missing, may be informative and may become a separate test.

Example 2: Surrogate Splits

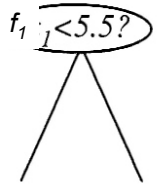


$$\omega_1: \begin{pmatrix} 0 \\ 7 \\ 8 \end{pmatrix}_{\mathbf{x}_1}, \begin{pmatrix} 1 \\ 8 \\ 9 \end{pmatrix}_{\mathbf{x}_2}, \begin{pmatrix} 2 \\ 9 \\ 0 \end{pmatrix}_{\mathbf{x}_3}, \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix}_{\mathbf{x}_4}, \begin{pmatrix} 5 \\ 2 \\ 2 \end{pmatrix}_{\mathbf{x}_5} \begin{matrix} f_1 \\ f_2 \\ f_3 \end{matrix}$$

$$\omega_2: \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}_{\mathbf{y}_1}, \begin{pmatrix} 6 \\ 0 \\ 4 \end{pmatrix}_{\mathbf{y}_2}, \begin{pmatrix} 7 \\ 4 \\ 5 \end{pmatrix}_{\mathbf{y}_3}, \begin{pmatrix} 8 \\ 5 \\ 6 \end{pmatrix}_{\mathbf{y}_4}, \begin{pmatrix} 9 \\ 6 \\ 7 \end{pmatrix}_{\mathbf{y}_5}.$$

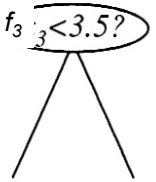
3 features for each sample
f_1, f_2, f_3

primary split



$x_1, x_2, x_3, x_4, x_5, y_1$ y_2, y_3, y_4, y_5

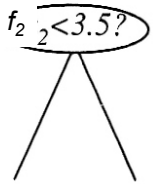
first surrogate split



x_3, x_4, x_5, y_1 y_2, y_3, y_4, y_5, x_1, x_2

predictive association with primary split = 8

second surrogate split



x_4, x_5, y_1, y_2 y_3, y_4, y_5, x_1, x_2, x_3

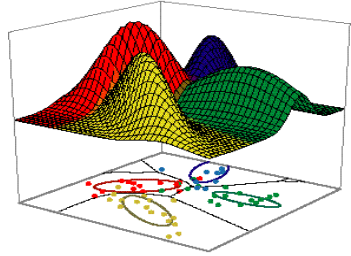
predictive association with primary split = 6

Minimizes entropy

Mimics primary split using different attribute

Likewise, but not as well…

Other DT Packages



- ID3
  - Uses only nominal (unordered data)
  - Real-valued data are binned first (discards ordering information).
  - Branching factor is always equal to number of nominal values/bins for that variable
    - Use ‘ratio impurity’ which penalizes for number of splits
    - Tree depth is always equal to number of features
  - Algorithm continues until all nodes are pure
- C4.5
  - Successor to ID3
  - Real-valued data treated like CART
  - Branching factor same as ID3 for nominal data
  - Pruning using statistical significance of splits.
  - In case of missing data during classification, follow all possible branches, then use weighted voting on final leaf nodes.