Decision Trees

Nominal Data

- So far we consider patterns to be represented by feature vectors of real or integer values
- Easy to come up with a distance (similarity) measure by using a variety of mathematical norms
- What happens if features are not numbers?
  - May not have a numerical representation
  - Distance measures might not make sense

Examples

- Colors of fruit
  - Green fruits are no more similar to red fruits then black fruits to red ones
- Smell
- Usefulness
- Interesting
- Etc.

Examples (cont.)

- Instead of a feature *vector* of numbers, we have a feature *list* of anything
- Fruit {color, texture, taste, size}
- DNA as a string of four letters AGCTTCGA, etc.

Classification

- Visualizing using n-dimensional space might be difficult
  - How to map, say, smell, onto an axis?
  - There might only be few discrete values (an article is highly interesting, somewhat interesting, not interesting, etc.)
- Even though that helps, do remember you cannot take distance measure in that space
  - (e.g., Euclidean distance in r-g-b color space does not correspond to human perception of color)

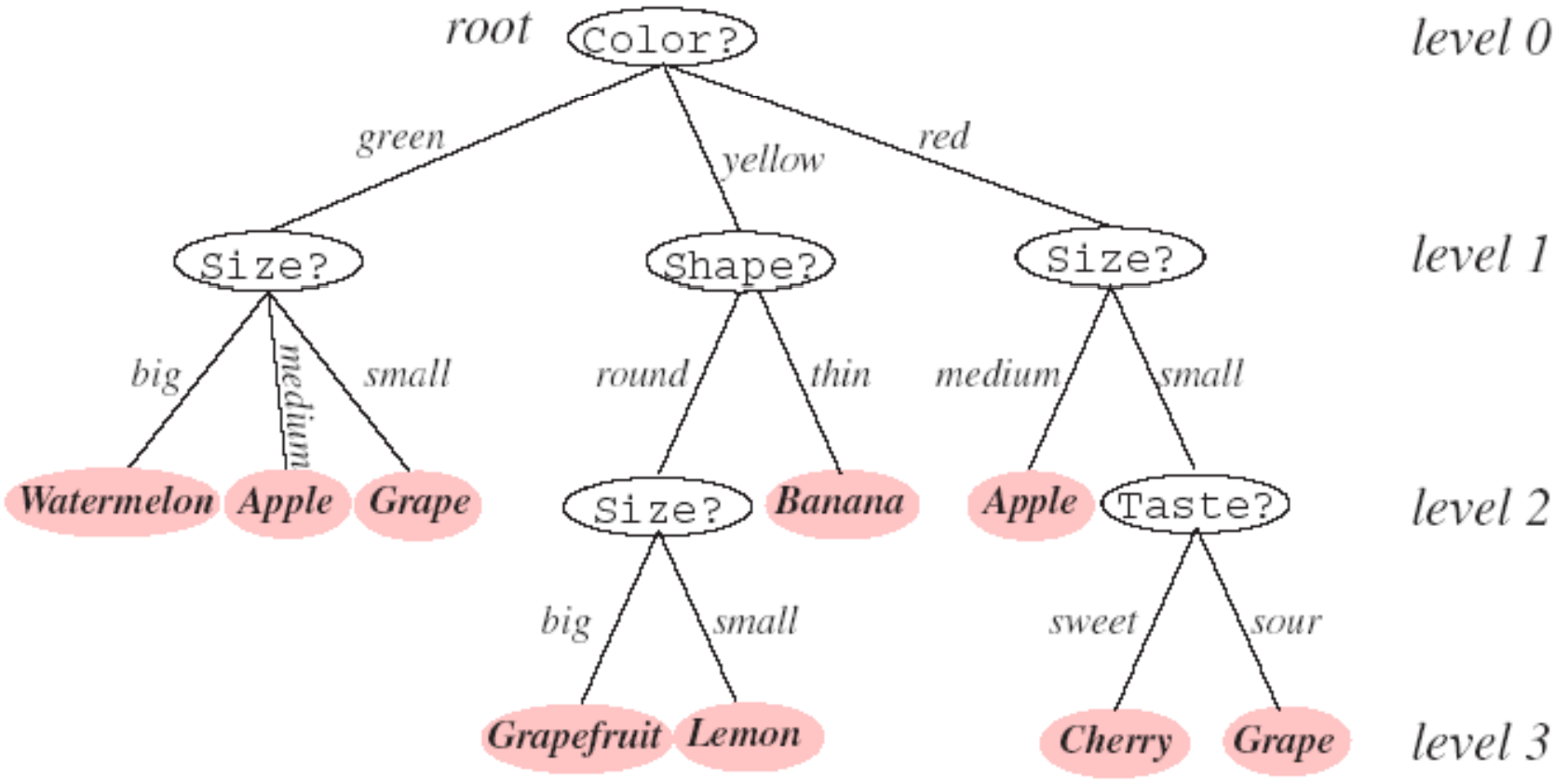
Decision Trees

- A classification based on a sequence of questions on
  - A particular feature (E.g., is the fruit sweet or not?) or
  - A particular set of features (E.g., is this article relevant and interesting?)
- Answer can be either
  - Yes/no
  - Choice (relevant & interesting, interesting but not relevant, relevant but not interesting, etc.)
  - Usual a finite number of discrete values

Use of a Decision Tree

- Relatively simple
  - Ask question represented at the node
  - Traverse down the right branch until a leaf node is reached
  - Use the label at that leaf for the sample
- Work if
  - Links are mutually distinct and exhaustive (may include branches for default, N/A, D/N, etc.)

Use of a Decision Tree (cont.)

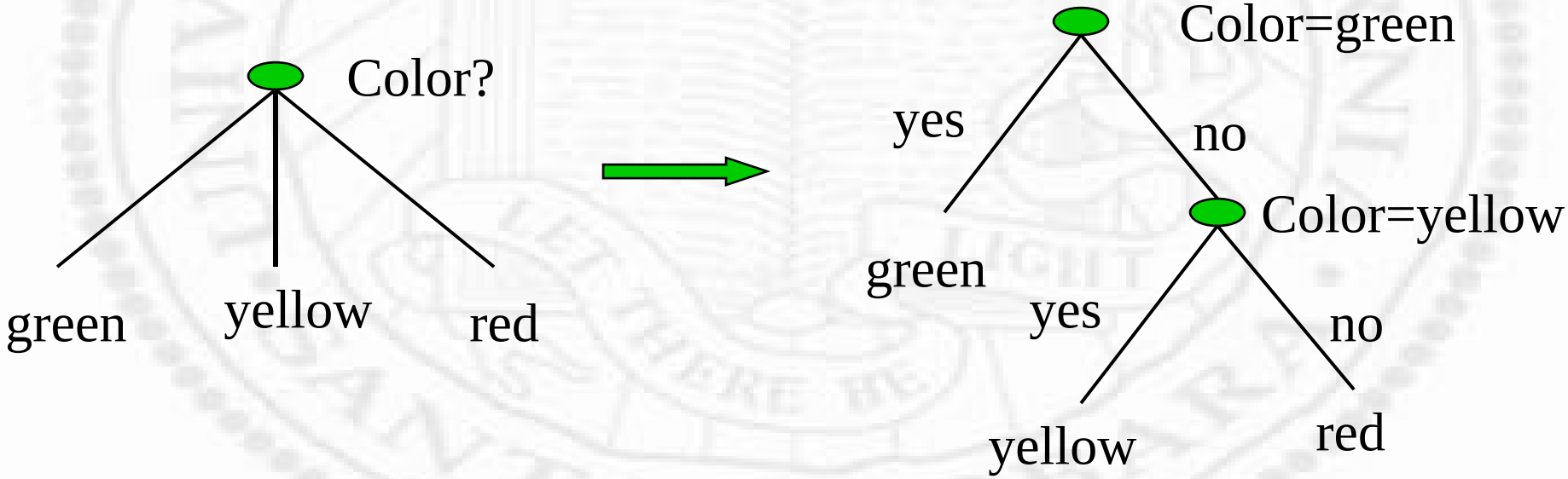


Creation of a Decision Tree

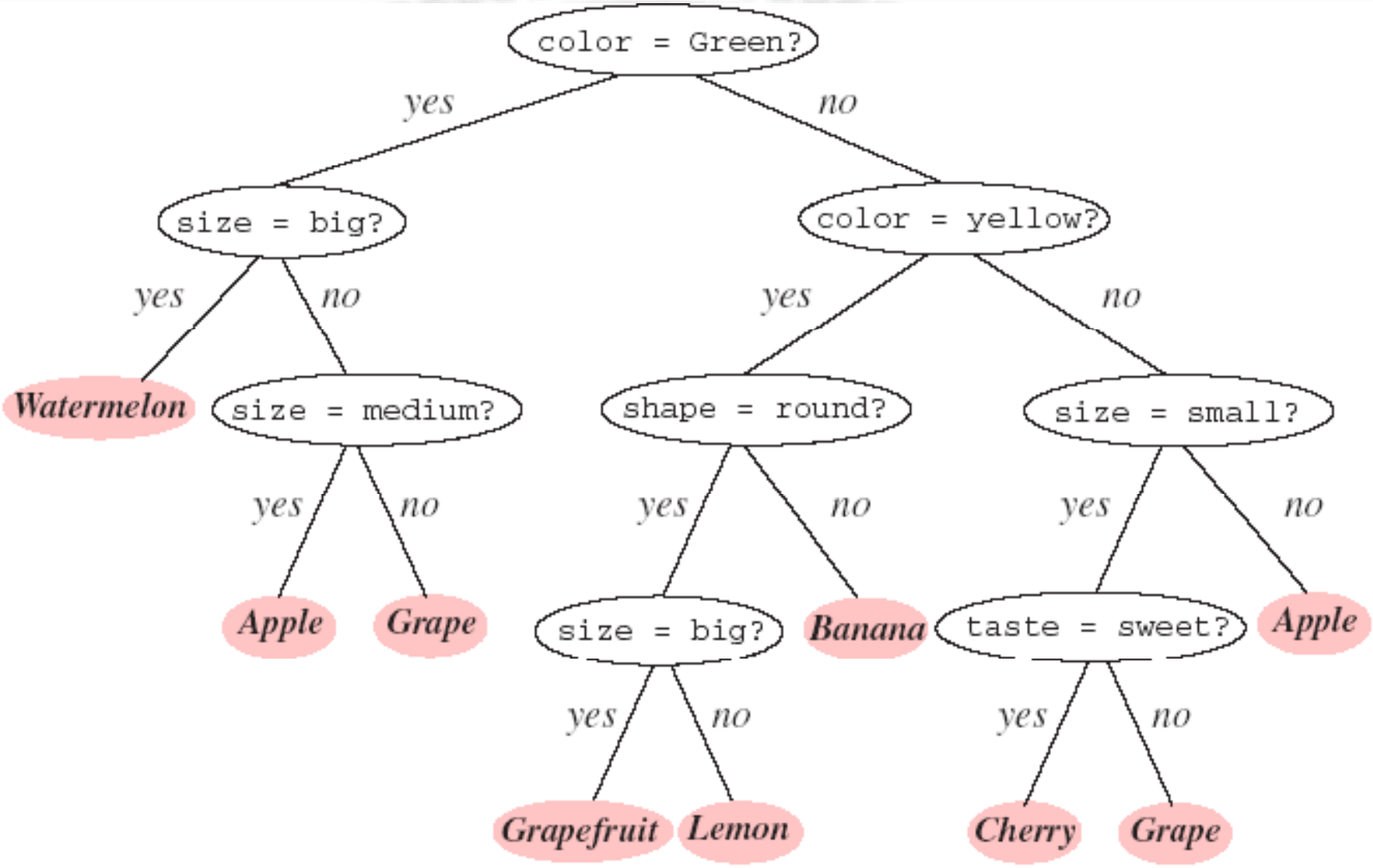
- Use supervised learning
  - Samples with tagged label (just like before)
- Process
  - Number of splits
  - Query selection
  - Rule for stopping splitting and pruning
  - Rule for labeling the leaves
  - Variable combination and missing data
- *Decision tree CAN be used with metric data (even though we motivate it using nonmetric data)*

Number of Splits

- Binary vs. Multi-way
  - Can always make a multi-way split into binary splits



Number of Splits (cont.)



Test Rules

- If a feature is an ordered variable, we might ask is x>c, for some c
- If a feature is a category, we might ask is x in a particular category
- *Yes* sends samples to left and *no* sends samples to right
- Simple rectangular partitions of the feature space
- More complicated ones: is x>0.5 & y<0.3

Test Rules (cont.)

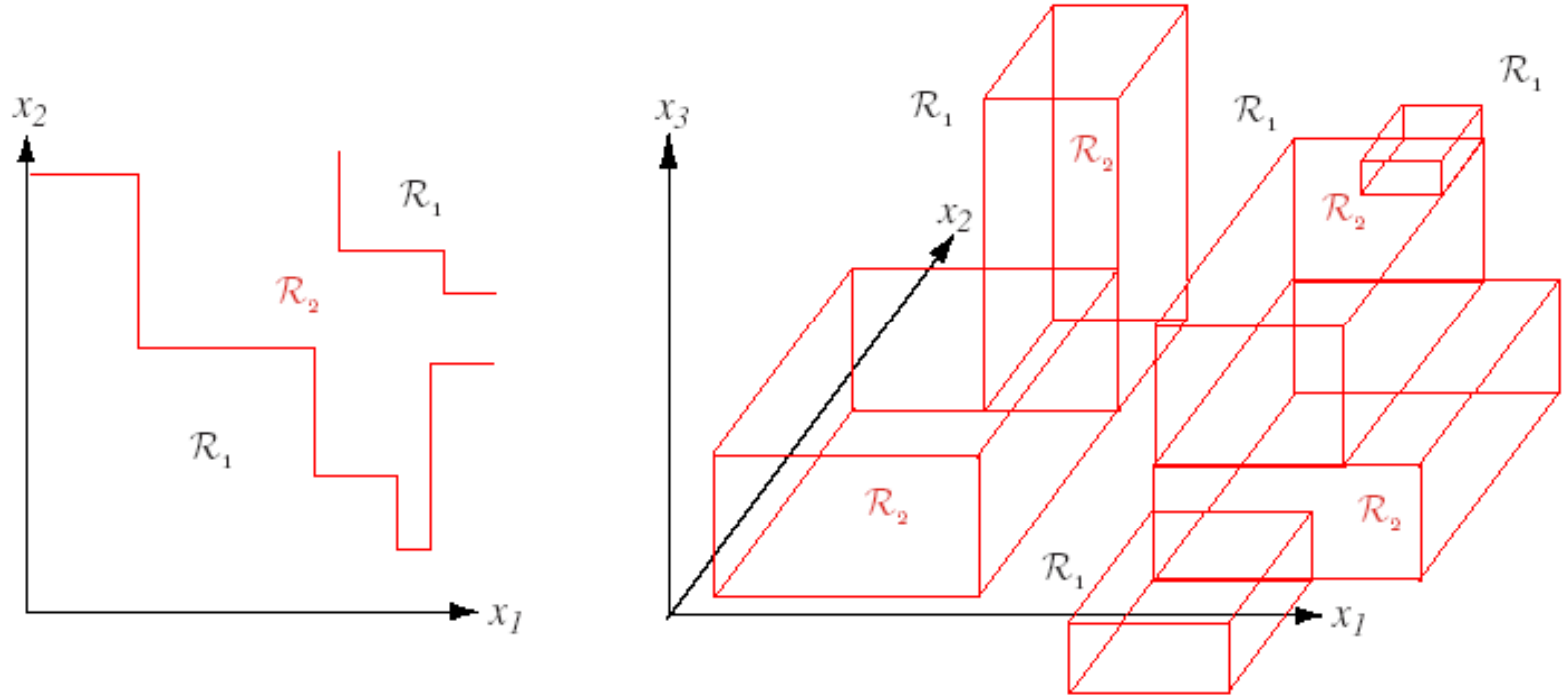


FIGURE 8.3. Monothetic decision trees create decision boundaries with portions perpendicular to the feature axes. The decision regions are marked $\mathcal{R}_1$ and $\mathcal{R}_2$ in these two-dimensional and three-dimensional two-category examples. With a sufficiently large tree, any decision boundary 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.

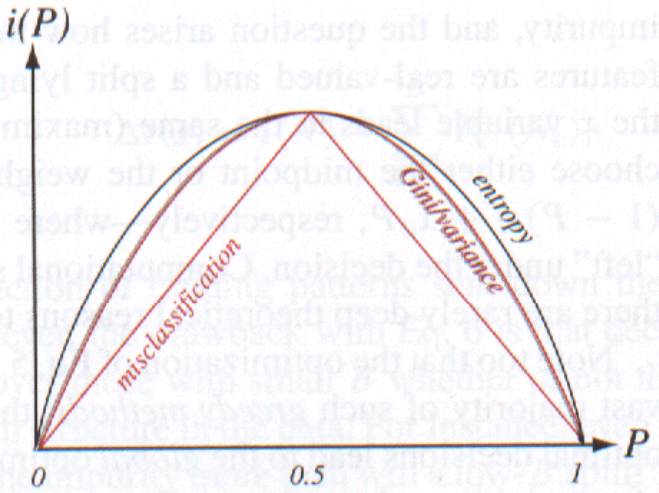
Criteria for Splitting

- Intuitively,to make the populations of the samples in the two children nodes *purer* than the parent node
- What do you mean by pure?
- General formulation
  - At node n, with k classes
  - Impurity depends on probabilities of samples at that node being in a certain class

$$P(w_i \mid n) \quad i = 1, \cdots, k$$
$$i(n) = f(P(w_1 \mid n), P(w_2 \mid n), \cdots, P(w_k \mid n))$$

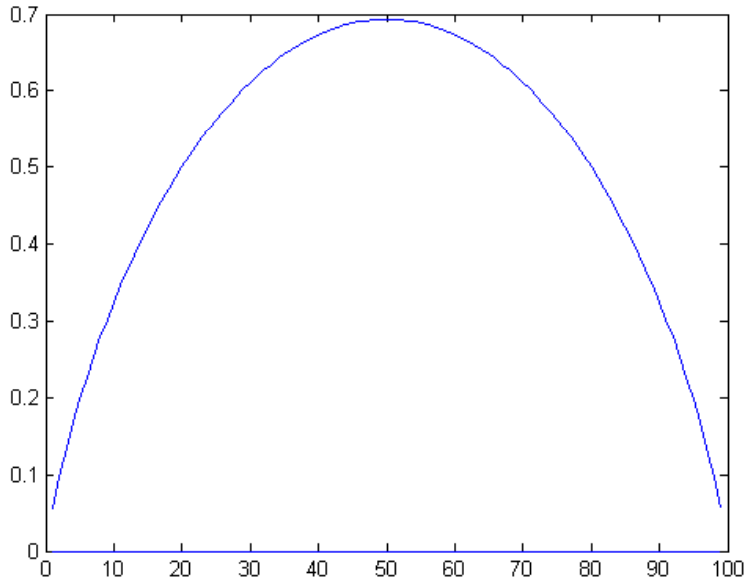
Possibilities

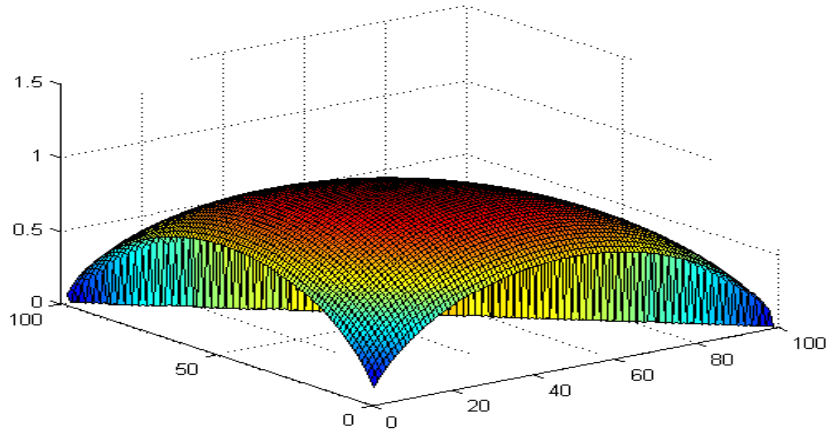
- Entropy impurity $$i(n) = -\sum_j P(w_j)\log_2 P(w_j)$$
- Variance (Gini) impurity $$i(n) = \sum_{i \neq j} P(w_i)P(w_j) = 1 - \sum_j P(w_j)^2$$
- Misclassification impurity $$i(n) = 1 - \max P(w_j)$$



Entropy

- A measure of "randomness" or "unpredictability"
- In information theory, the number of bits that are needed to code the transmission

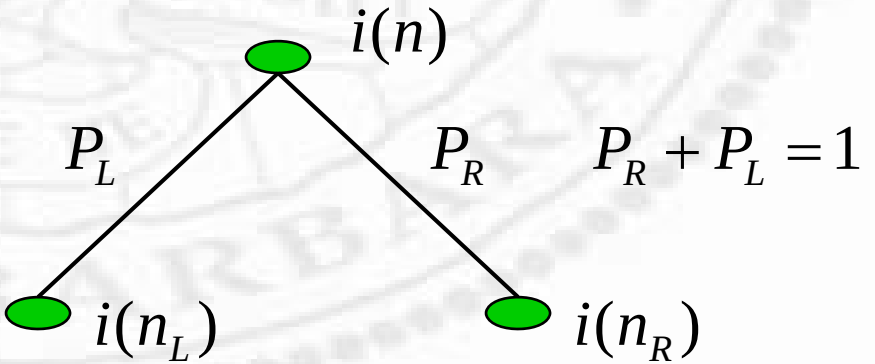




Split Decision

- Before split – fixed impurity
- After split – impurity depends on decisions
- The goal is maximize the drop in impurity
- Difference between
  - Impurity at root
  - Impurity at children (weighted by population)

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



Split Decision

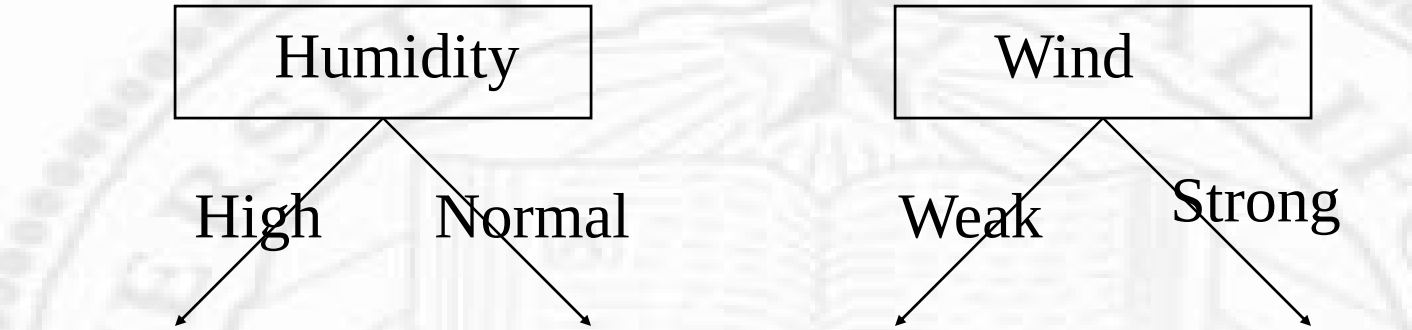
- We might also just minimize $P_L i(N_L) + P_R i(N_R)$
- The reason to use delta is because if entropy impurity is used, the delta is information gain
- Usually, a single feature is selected from among all remaining ones (combination of features is possible, but very expensive)

Example

Day	Outlook	Temperature	Humidity	Wind	Play tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Host	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cold	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

$$root: (9+,5-): E = -(9/14)\log(9/14) - (5/14)\log(5/14) = 0.94$$



$$(6+,1-): E = -(6/7)\log(6/7) - (1/7)\log(1/7) = 0.592$$

Gain:

$.940 - (7/14).985 - (7/14).592$

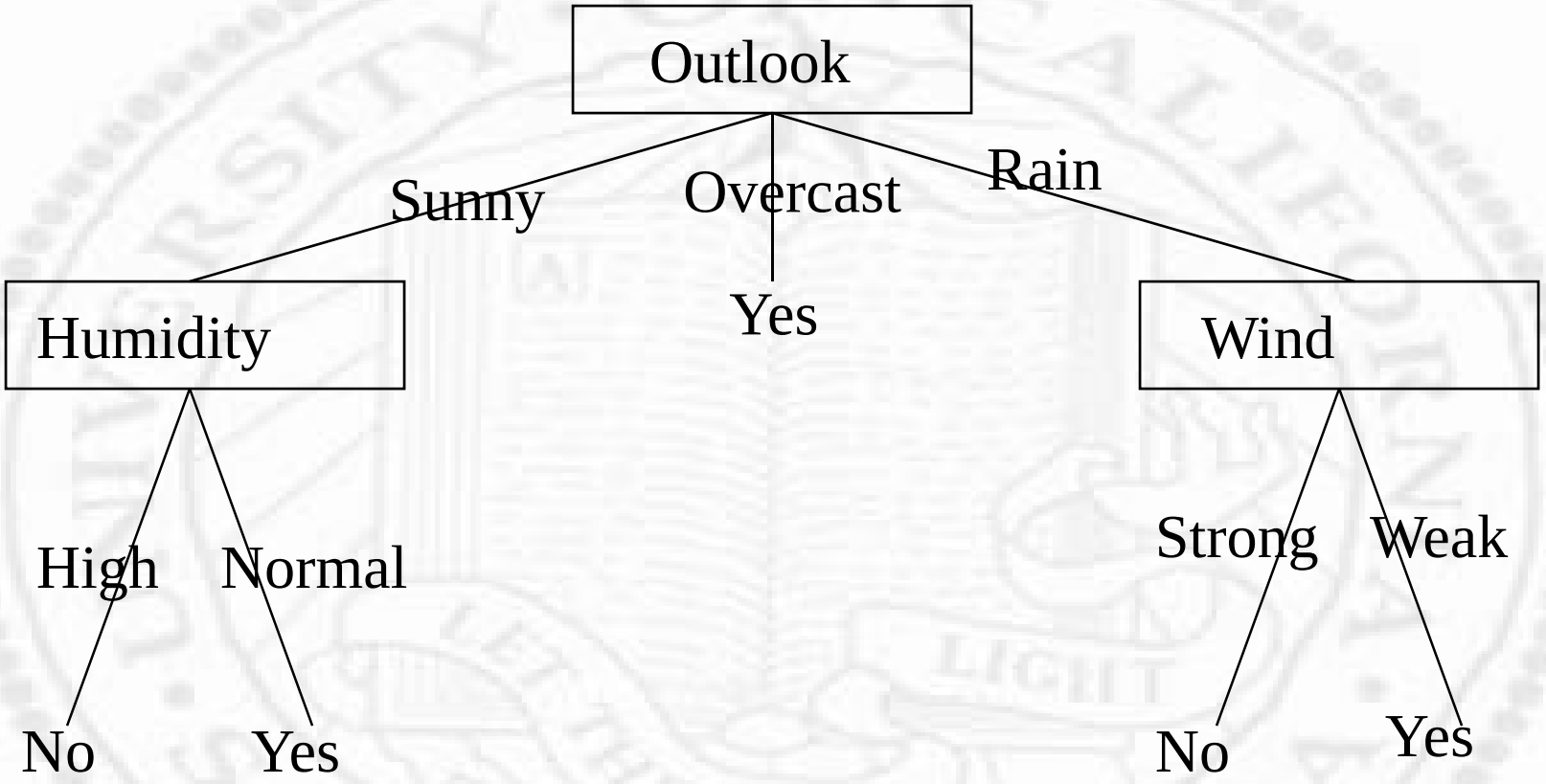
$= .151$

Gain:

$.940 - (8/14).811 - (6/14)1.0$

$= .048$

Example


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Multiway Splits

- In general, more splits allow impurity to drop
  - Splits reduce the samples in each branch
  - With few samples, it is likely that one sample might dominate (1 sample, impurity=0, 2 samples, 50% chance impurity=0)
- Proper scaling of change of impurity
  - Large split is penalized

$$\Delta i(n) = i(n) - \sum_{k=1}^{B} P_k i(N_k) \longrightarrow \Delta i_B(n) = \frac{\Delta i(n)}{-\sum_{k=1}^{B} P_k \log_2 P_k}$$

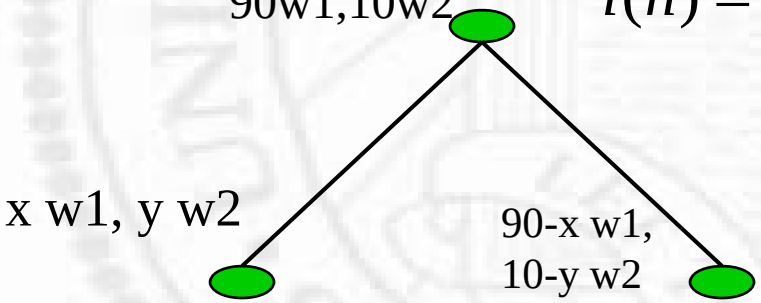
Large entropy -> bad split

Caveats

- Decisions are made locally, one steps at a time (greedy)
  - Search in a “tree space” with all possible trees
  - There is no guarantee that the tree is going to be optimal (shortest height)
  - Other techniques (e.g., DP) can guarantee the optimal solution, but are more expensive

Caveats (cont.)

- Some impurity may not be as good as others
- E.g., 90 of w_1 and 10 of w_2 at a node, x of w_1 and y of w_2 go left and x > y and w_1 still dominate at both children
- Using misclassification impurity



$$i(n) = 1 - \max(\frac{90}{100}, \frac{10}{100}) = 0.1$$

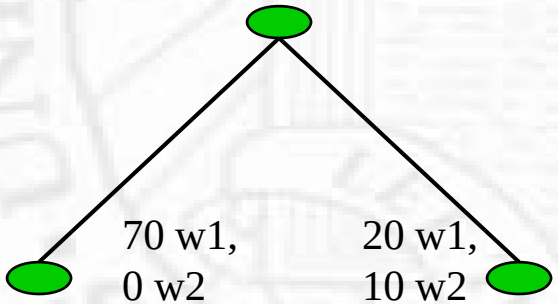
$$P_L i(n_L) + P_R i(n_R) = \frac{x+y}{100}(1 - \frac{x}{x+y}) + \frac{100-x-y}{100}(1 - \frac{90-x}{100-x-y}) = 0.1$$

$$i(n_L) = 1 - \frac{x}{x+y} \qquad i(n_R) = 1 - \frac{90-x}{100-x-y}$$

Caveats (cont.)

- E.g., 90 of w_1 and 10 of w_2 at a node, 70 of w_1 and 0 of w_2 go left
- Using misclassification impurity

$$i(n) = 1 - \max\left(\frac{90}{100}, \frac{10}{100}\right) = 0.1$$



$$P_L i(n_L) + P_R i(n_R) = 0 + 0.3 * 0.33 = 0.099$$

$$i(n_L) = 0$$

$$i(n_R) = 1 - \frac{20}{20+10} = 0.33$$

Caveats (cont.)

- It may also be good to split based on classes (in a multi-class binary tree) instead of based on individual samples (twoing criterion)
- C = {c1,c2,…, cn} breaks into
  - C1 = {ci1,ci2,…cik}
  - C2 = C –C1

When to Stop Split?

- Keep on splitting you might end up with a single sample per leaf (overfitting)
- Not doing enough you might not be able to get good labels (it might be an apple, or an orange, or an …)

When to Stop Split? (cont.)

- Thresholding: if split results in small impurity reduction, don't do it
  - What should the threshold be?
- Size limit: if the node contains too few samples, don't split anymore
  - If region is sparse, don't split
- Combination of above (too small, stop)

$$\alpha \cdot size + \sum_{leaves} i(n)$$

Pruning (bottom-up)

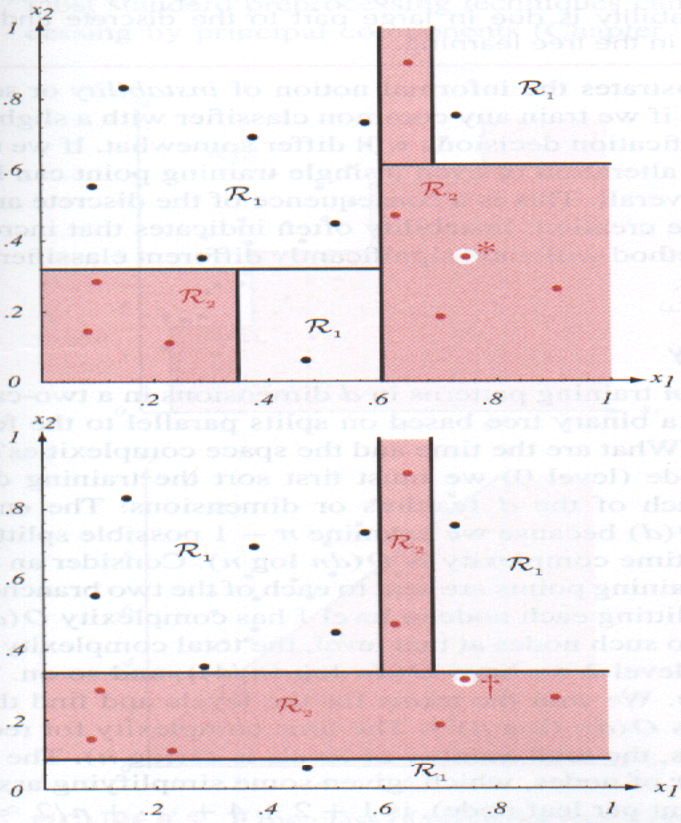
- Splitting is a top-down process
- We can also do bottom up by splitting all the ways down, followed by a merge (bottom-up process)
- Adjacent nodes whose merge induces a small increase in impurity can be joined
- Avoid *horizontal effect*
  - Hard to decide when to stop splitting, the arbitrary threshold set an artificial “floor”
- Can be expensive (explore more branches that might eventually be thrown away and merged)

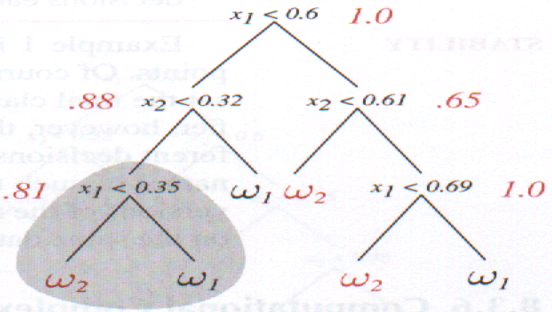
Assigning Labels

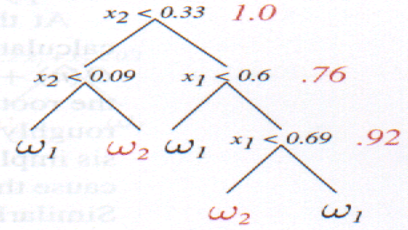
- If a leaf is of zero impurity – no brainer
- Otherwise, take the label of the dominant samples (similar to k-nearest neighbor classifier) – again, no brainer

Example

ω_1 (black)		ω_2 (red)	
x_1	x_2	x_1	x_2
.15	.83	.10	.29
.09	.55	.08	.15
.29	.35	.23	.16
.38	.70	.70	.19
.52	.48	.62	.47
.57	.73	.91	.27
.73	.75	.65	.90
.47	.06	.75	.36* (.32†)







Training data and associated (unpruned) tree are shown at the top. The entropy impurity at nonterminal nodes is shown in red and the impurity at each leaf is 0. If the single training point marked * were instead slightly lower (marked †), the resulting tree and decision regions would differ significantly, as shown at the bottom.

Missing Attributes

- A sample can miss some attributes
- In training
  - Don’t use that sample at all – no brainer
  - Use that sample for all available attributes, but don’t use it for the missing attributes – again no brainer

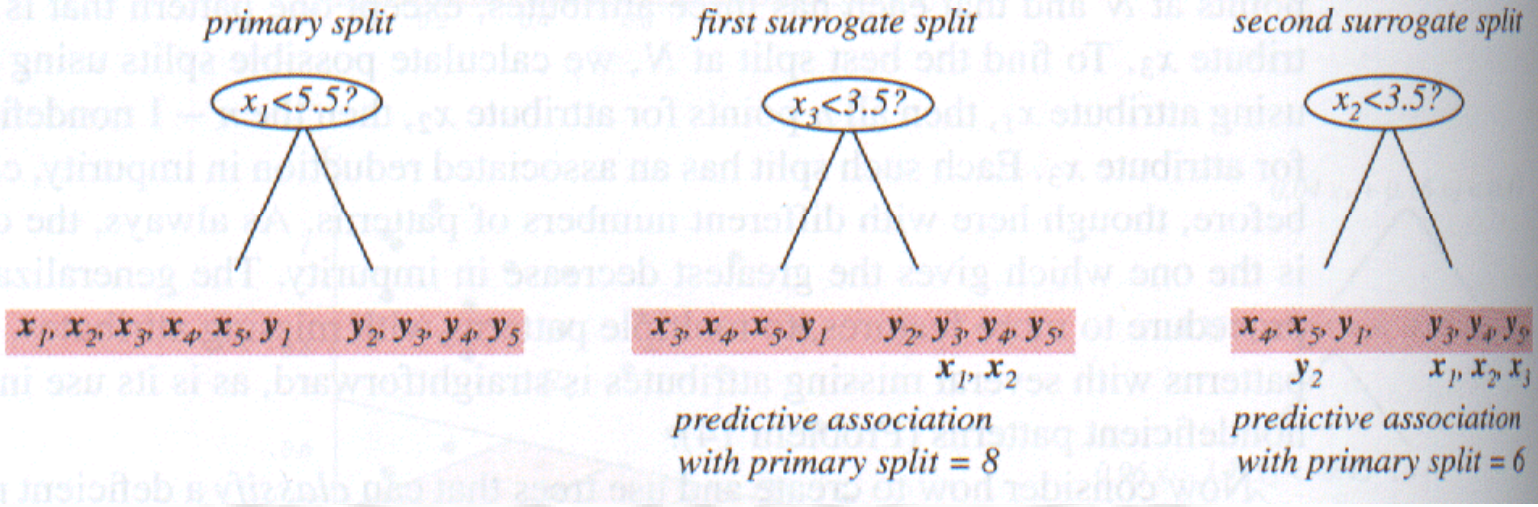
Missing Attributes (cont.)

- In classification
  - What happens if the sample to be classified is missing some attributes?
  - Use surrogate splits
    - Define more than one split (primary + surrogates) at nonterminal nodes
    - The surrogates should maximize predictive association (e.g., surrogates send the same number of patterns to the left and right as the primary)
    - Expensive
  - Use virtual values
    - The most likely value for the missing attribute (e.g., the average value for the missing attribute of all training data that end up at that node)

Missing Attributes (cont.)

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

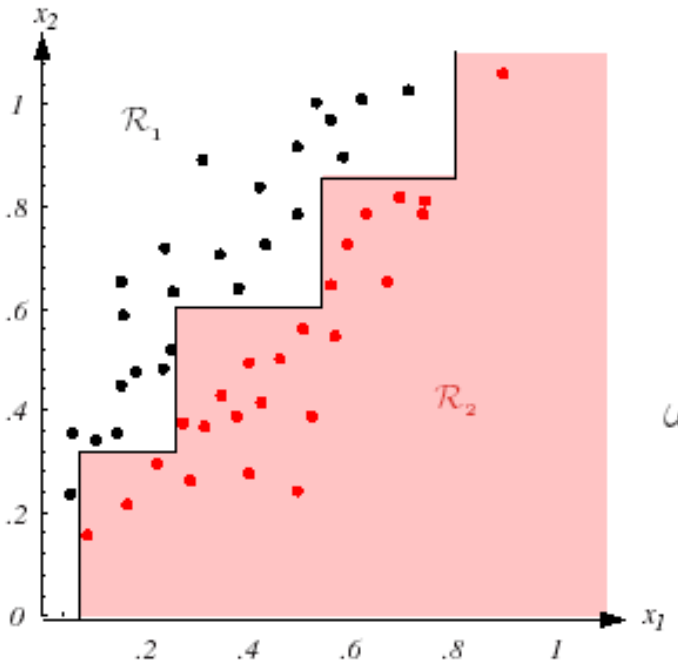
$$\omega_2: \quad \mathbf{y}_1 \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}, \quad \mathbf{y}_2 \begin{pmatrix} 6 \\ 0 \\ 4 \end{pmatrix}, \quad \mathbf{y}_3 \begin{pmatrix} 7 \\ 4 \\ 5 \end{pmatrix}, \quad \mathbf{y}_4 \begin{pmatrix} 8 \\ 5 \\ 6 \end{pmatrix}, \quad \mathbf{y}_5 \begin{pmatrix} 9 \\ 6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

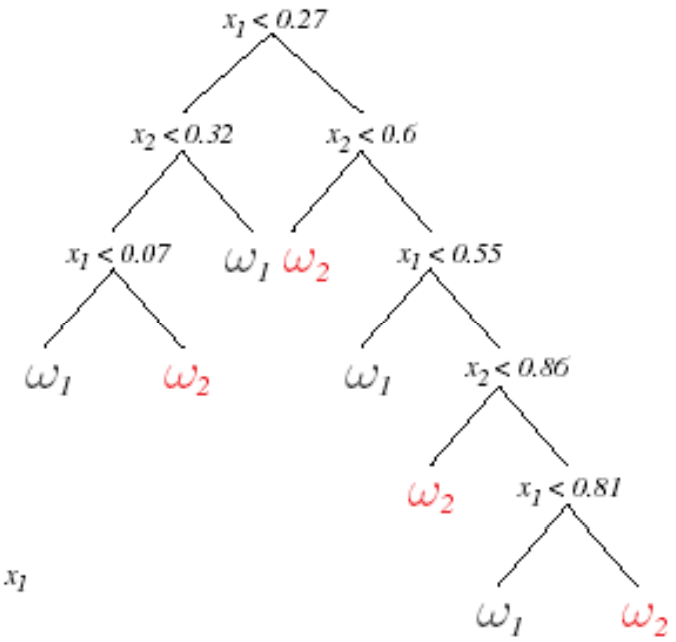


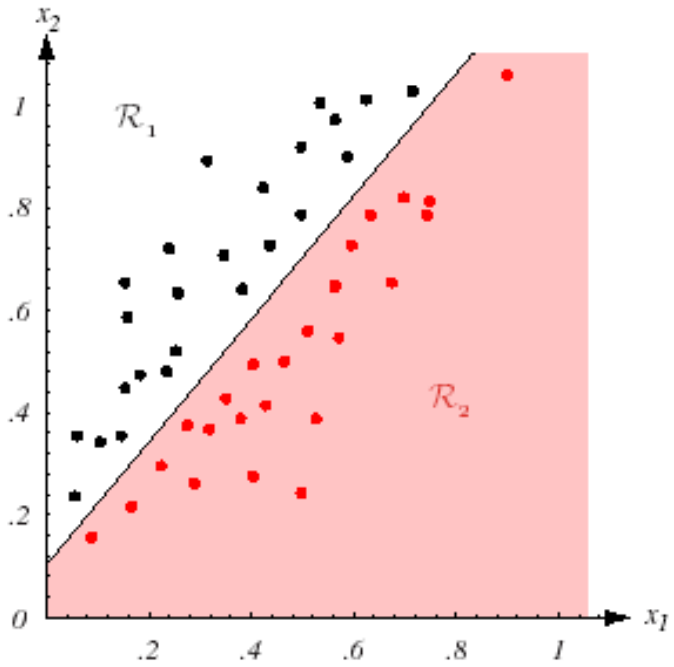
Feature Choice

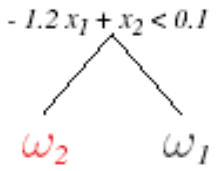
- No magic here – if feature distribution does not line up with the axes, the decision boundary is going to zigzag, which implies trees of a greater depth
- Principal component analysis might be useful to find dominant axis directions for cleaner partitions

Feature Choice (cont.)

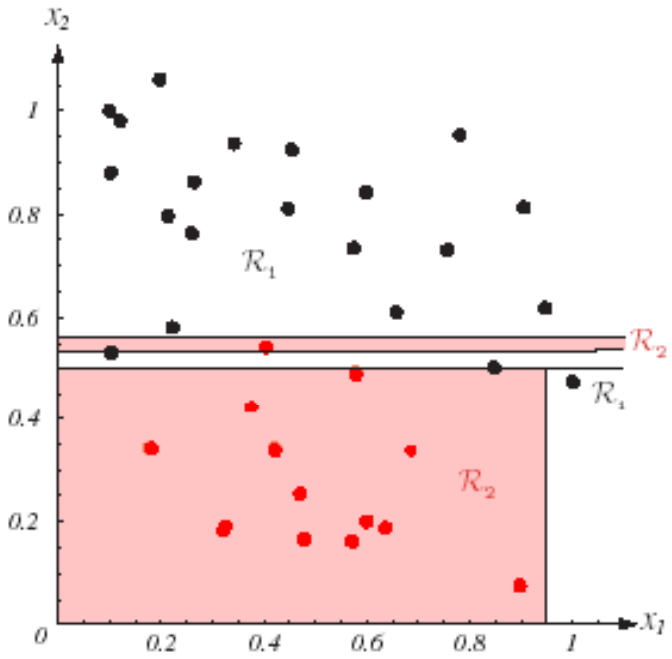


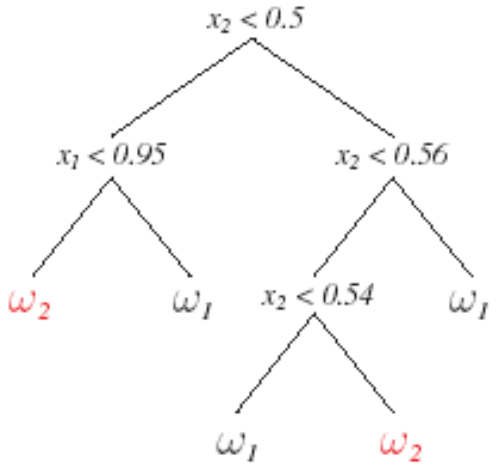


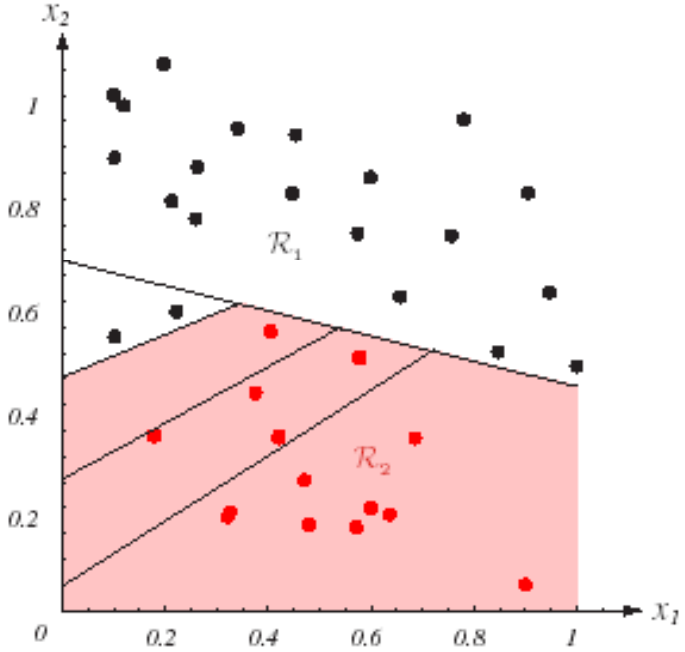


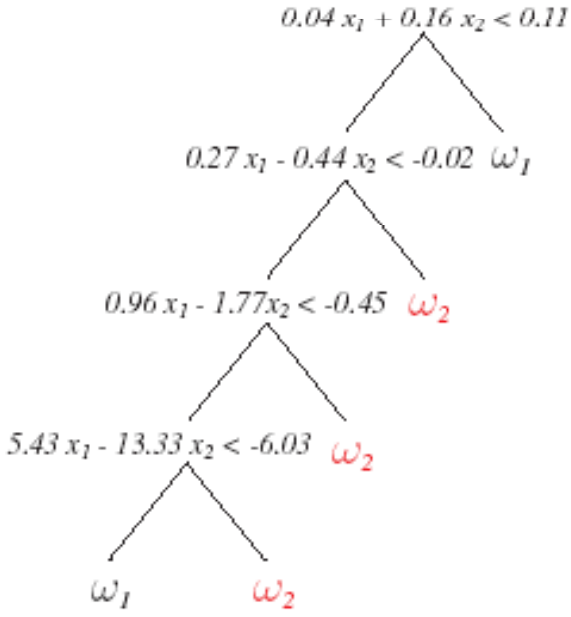


Feature Choice (cont)









Other Alternatives

- What we described is the CART (classification and regression tree)
- ID3 (interactive dichotomizer)
  - Nominal data
  - Multi-way split at a node
  - # level = # variables
- C4.5
  - No surrogate split (save space)

A Pseudo Code Algorithm

- ID3(Examples, Attributes)
  - Create a *Root* node
  - If all Examples are positive, return *Root* with label = +
  - If all Examples are negative, return *Root* with label = -
  - If no attributes are available for splitting, return *Root* with label = most common values (+ or -) in Examples

- Otherwise
  - Choose a best attribute (*A*) to split (based on infogain)
  - The decision attribute for *Root* = *A*
  - For each possible value *vi* of *A*
    - Add a new branch below Root for *A*=*vi*
    - Example_{vi} = all Examples with *A*=*vi*
    - If Example_{vi} is empty
      - Add a leaf node under this branch with label=most common values in Example
    - Else
      - Add a leaf node under this branch with label = ID3(Example_{vi},Attributes-A)