EE 649 Pattern Recognition

Decision Trees

Ulisses Braga-Neto

ECE Department

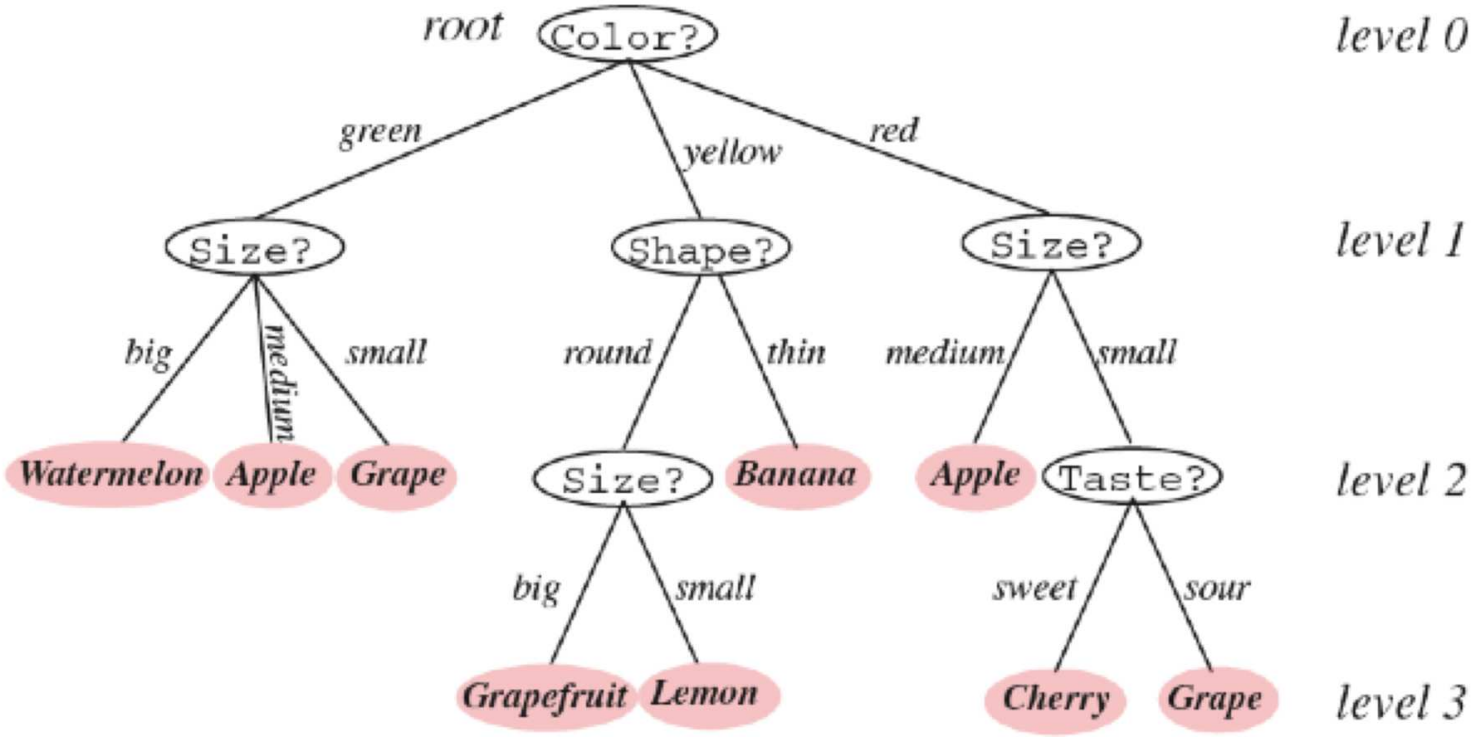
Texas A&M University

Main Ideas

- Adjustable partition-based discriminants that cut up the space recursively (“20-questions” approach).
- Non-metric data: tree classifiers can handle data where some or all the features are *nominal*, i.e., not numeric.
- High degree of *interpretability*: Inferred classifier can be specified in terms of logical rules that can lead to scientific hypotheses about the mechanism that produced the data.
- Very popular in *Data Mining*.

Tree Classifiers

A tree classifier consists of nodes where data splitting occurs. There is a *root* node, and terminal *leaf* nodes where label assignment occurs.



Logical Rules

- Each leaf node in a tree classifier correponds to a rule, obtained by *traversing* the tree from the root node to the leaf. For example:

  Banana := `(color = yellow) AND (shape = thin)`

  `= (yellow AND thin)`

- We can combine rules for identical leafs by using the operator OR:

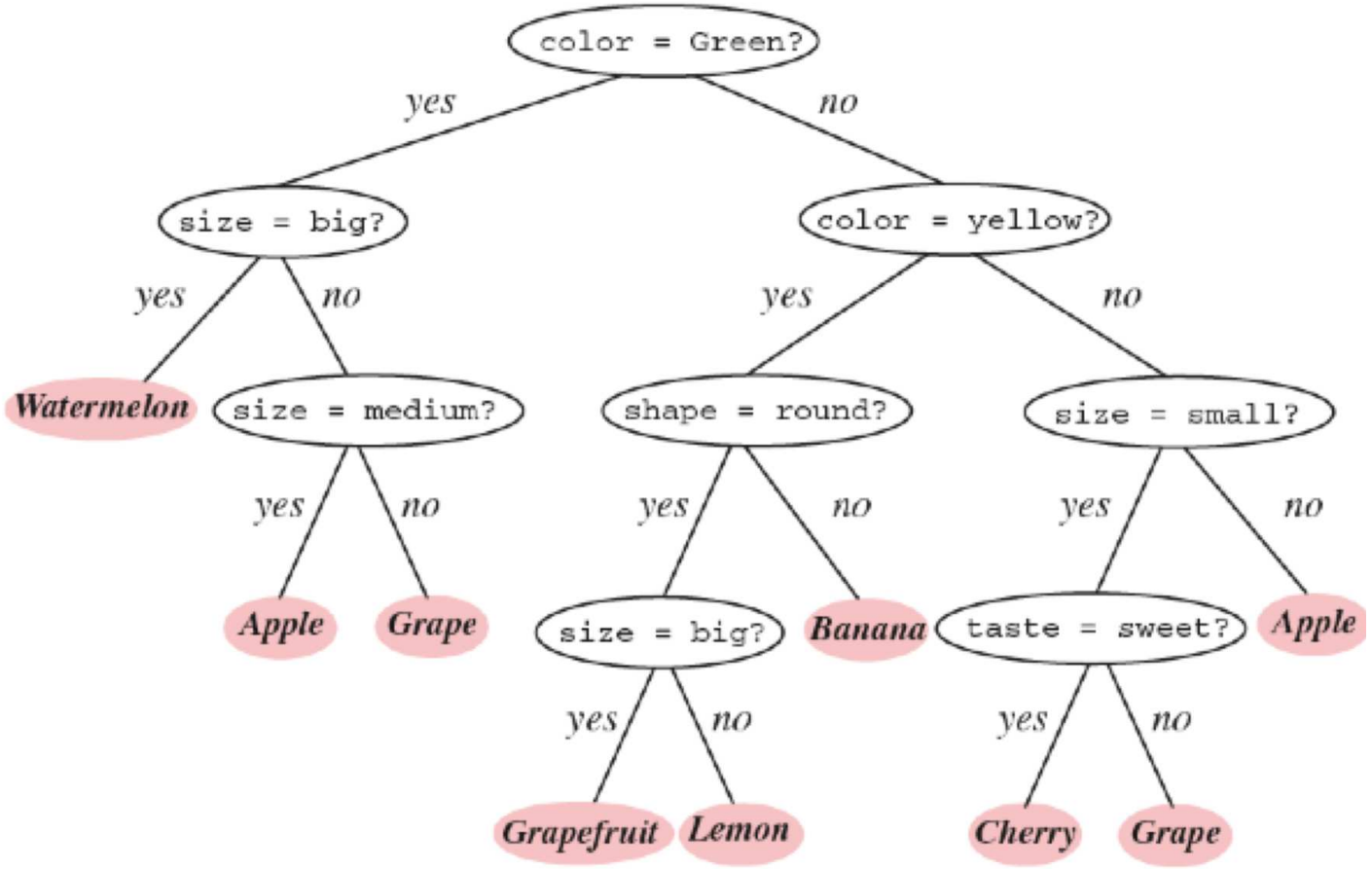
  Apple := `(green AND medium) OR (red AND medium)`

- Rules can also be simplified logically to aid interpretability

  Apple := `(medium AND NOT yellow)`

Binary Tree Classifiers

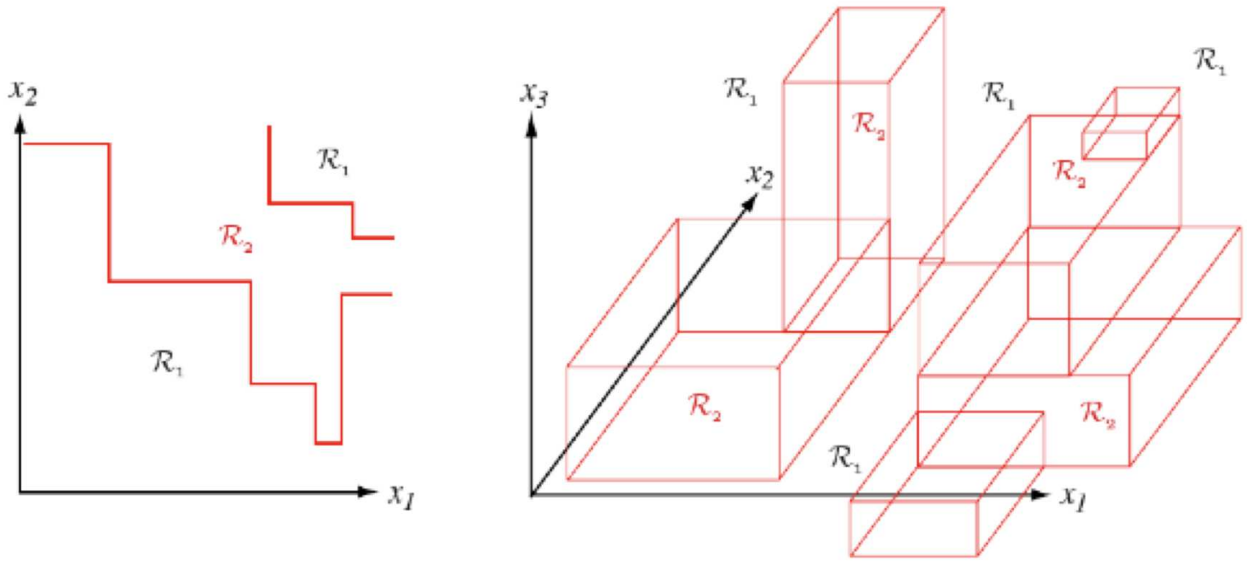
Any decision tree can be written as a *binary* tree, i.e, trees for which the splitting criteria are all binary.



CART

CART (Classification and Regression Trees) is a very popular tree rule for numeric data.

CART produce tree classifiers where the decision at each node is of the form $x^j \leq \alpha?$ where x^j is one of the coordinates of the pattern x. Such classifiers produce linear boundaries that are parallel to the axes.



Impurity Criterion

For each node, the coordinate to split on j and the split threshold α are determined in training by using the concept of an *impurity* criterion.

Given a node R (always a rectangle in feature space), let
$N_1(R)$ = number of 1-labeled points in R
$N_0(R)$ = number of 0-labeled points in R
$N(R) = N_0(R) + N_1(R)$ = total number of points in R.

The *impurity* of R is defined by

$$\kappa(R) = \xi(p, 1-p)$$

where $p = N_1(R)/N(R)$ and $1 - p = N_0(R)/N(R)$.

Impurity Criterion - II

The *impurity function* $\xi(p, 1-p)$ is nonnegative and satisfies the following conditions:

(1) $\xi(0.5, 0.5) \geq \xi(p, 1-p)$ for any $p \in [0, 1]$

(2) $\xi(0, 1) = \xi(1, 0) = 0$

(3) as a function of p, $\xi(p, 1-p)$ increases for $p \in [0, 0.5]$ and decreases for $p \in [0.5, 1]$.

These conditions correspond to the following observations:

(1) $\kappa(R)$ is maximum when $p = 1-p = 0.5$, i.e. $N_1(R) = N_0(R)$ (corresponding to maximum impurity).

(2) $\kappa(R) = 0$ if either $N_1(R) = 0$ or $N_0(R) = 0$; i.e. R is *pure*;

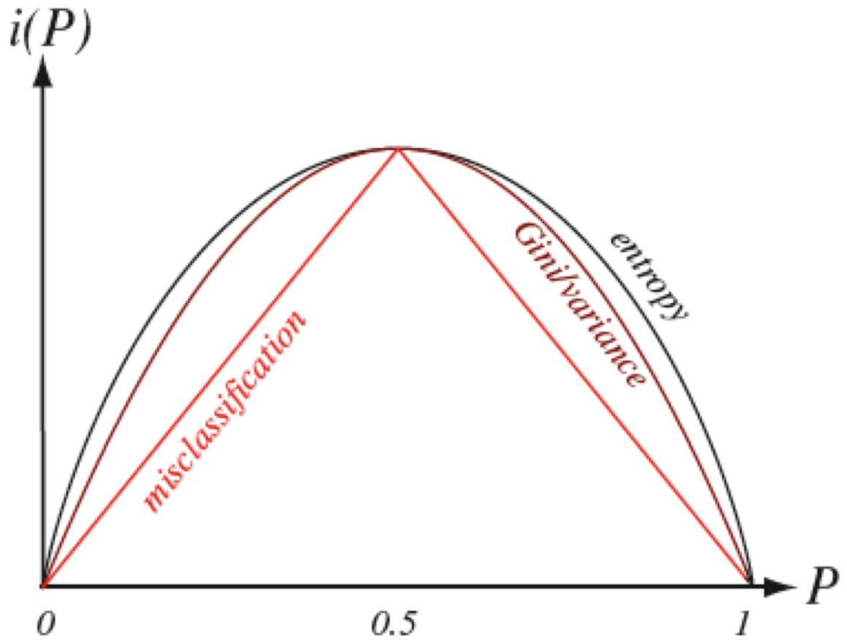
(3) $\kappa(R)$ increases as the ratio $N_1(R)/N_0(R)$ approaches 1.

Impurity Criterion - III

We mention three possible choices for ξ:

(1) $\xi_e(p, 1-p) = -p \log p - (1-p) \log(1-p)$ (*entropy impurity*)
(2) $\xi_g(p, 1-p) = 2p(1-p)$ (*Gini or variance impurity*)
(3) $\xi_m(p, 1-p) = \min(p, 1-p)$ (*misclassification impurity*)

Note that these are identical to F criteria used in F-errors.



Best Split Search

Given a node R, the coordinate j to split on and the split threshold α are determined as follows. Let

$R^j_{\alpha,-}$ = sub-rectangle resulting from $x^j \leq \alpha$

$R^j_{\alpha,+}$ = sub-rectangle resulting from $x^j > \alpha$

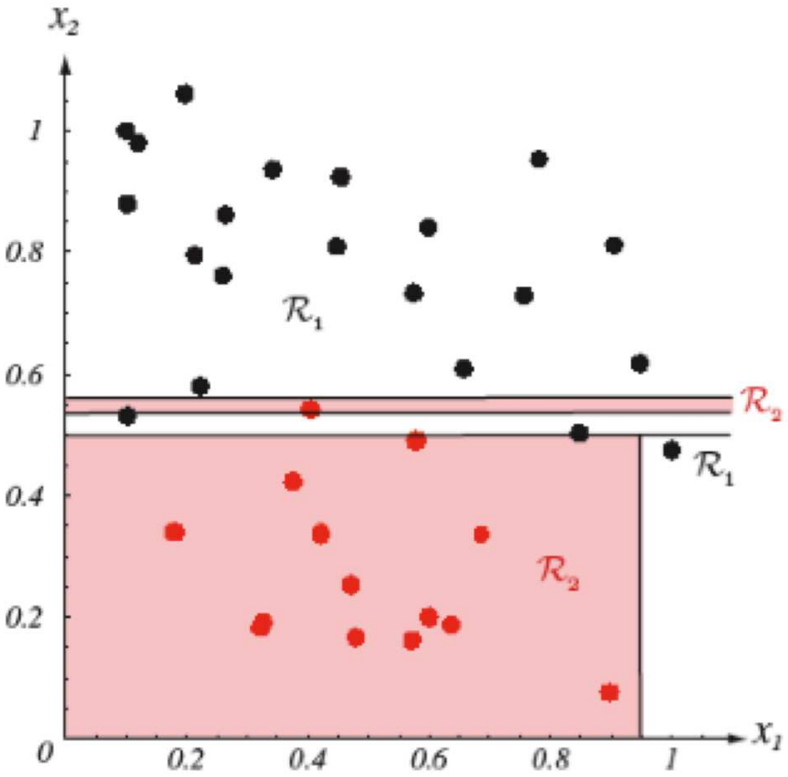
Define the *impurity drop* by

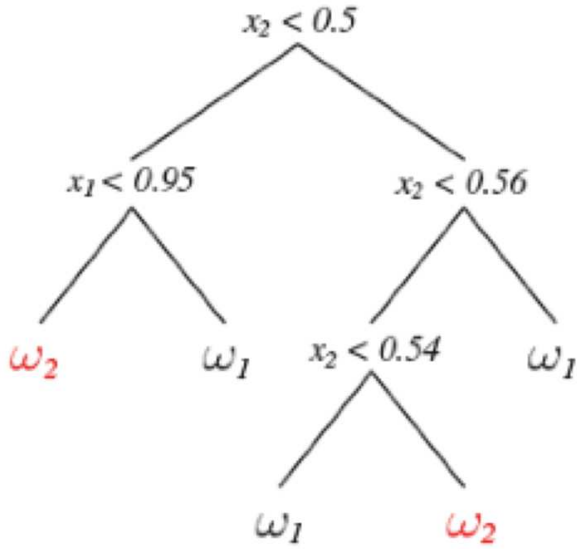
$$\Delta_R(j,\alpha) \;=\; \kappa(R) \;-\; \frac{N(R^j_{\alpha,-})}{N(R)}\kappa(R^j_{\alpha,-}) \;-\; \frac{N(R^j_{\alpha,+})}{N(R)}\kappa(R^j_{\alpha,+})$$

The strategy is to search for the j and α that maximize the impurity drop. There is only a finite number of candidate splits (j,α) (why?) so the search can be exhaustive.

Splitting stops when all nodes are pure (fully-grown tree).

Example of Fully-Grown CART Tree





Regularization

A fully-grown CART tree classifier almost surely overfits the training data, and is not a good classifier in terms of true classification error.

Two basic regularization techniques are available:

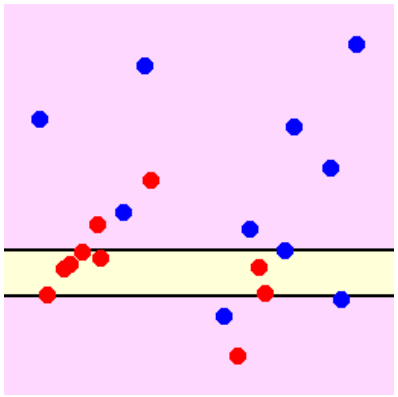
- Stopped Splitting: Call a node a leaf and assign the majority label if
  (1) there are fewer than k samples in it.
  (2) the best impurity drop is below a threshold θ.
- Pruning: Fully grow tree then consider pairs of neighboring leafs that can be merged without increasing impurity too much.

In addition, one can design several fully-grown CARTs via perturbation of the training data and combine the decisions by majority voting — this is the *random forest* approach.

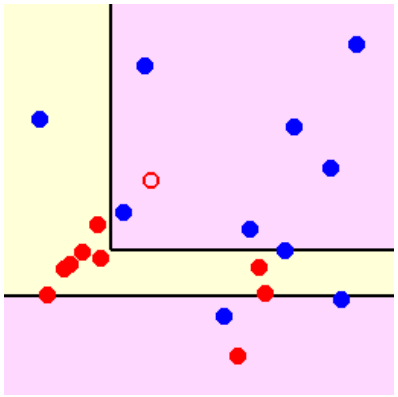
Small-Sample Instability of CART

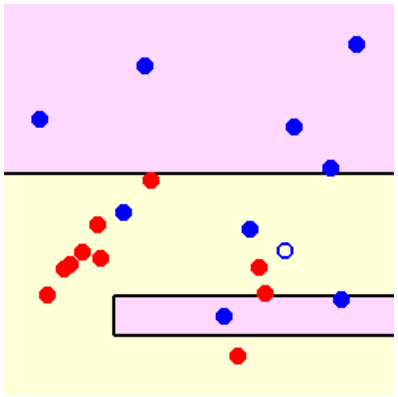
Example of CART with perturbed data (with stopping rule!)

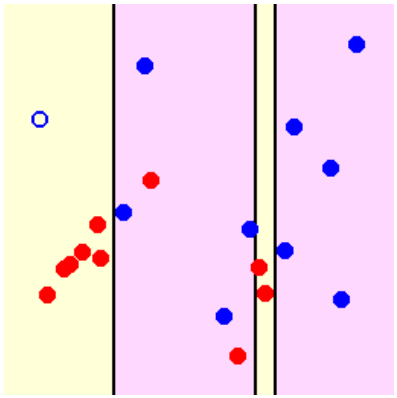
Original CART classifier



CART classifiers with one point deleted from training data

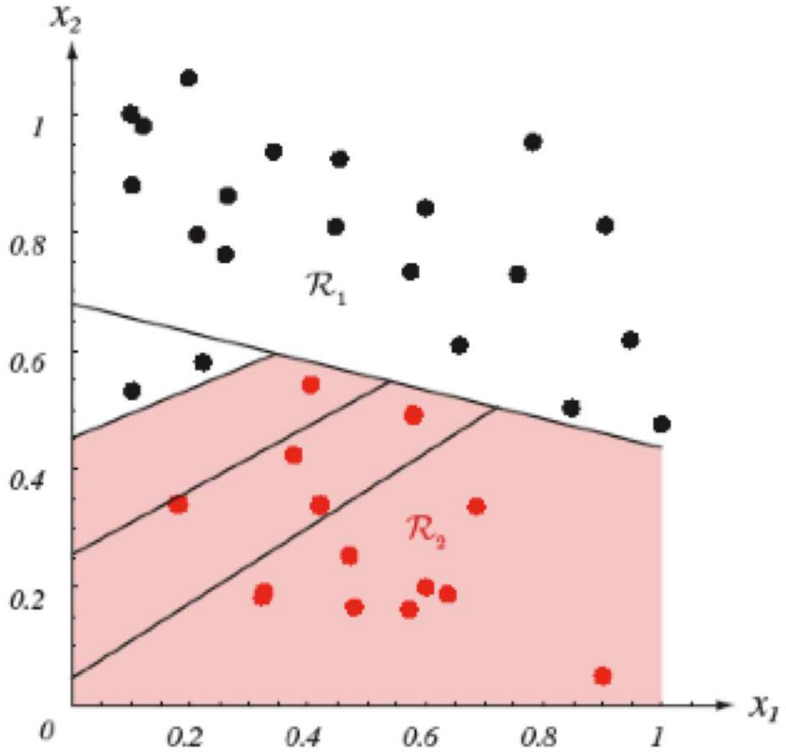


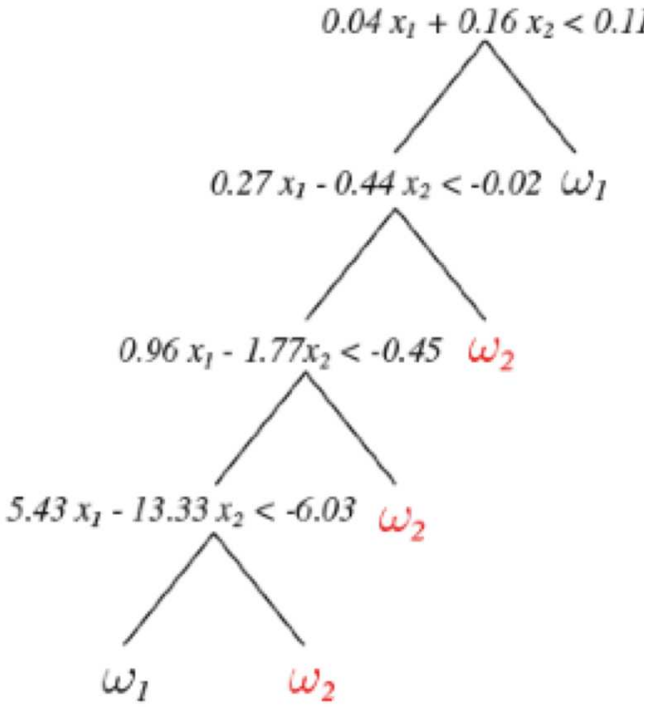




BSP Trees

Binary Space Partition (BSP) trees are similar to CART, but they employ hyperplane splits along arbitrary directions.



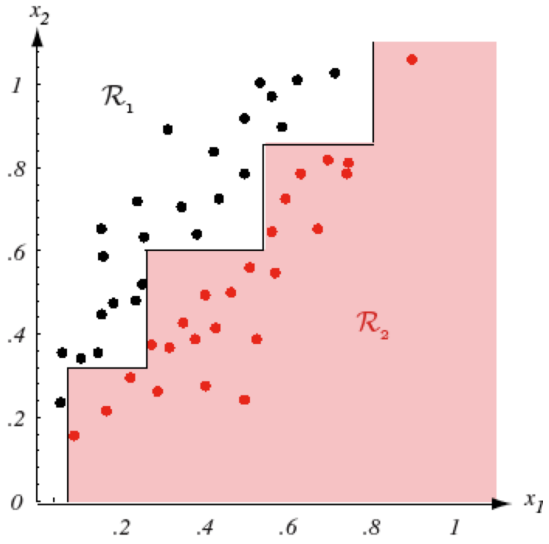


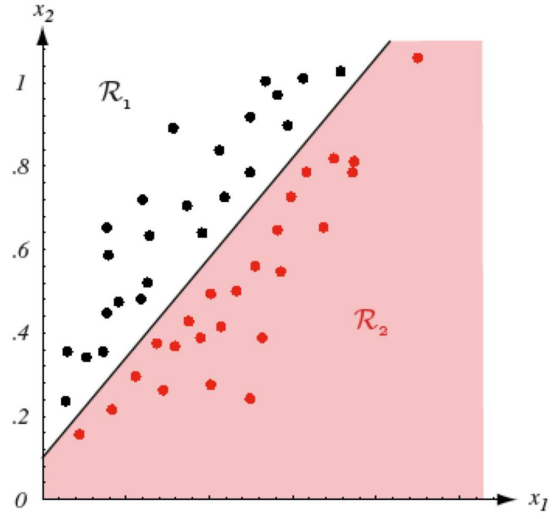
Invariance

CART rules are invariant with respect to scaling of the axes and translation, meaning that if T denotes such a transformation, we have:

$$\Psi_n(T(x); T(S_n) = \{(T(x_i), y_i)\}) \; = \; \Psi_n(x; \; S_n = \{(x_i, y_i)\})$$

CART rules are *NOT* invariant to more complicated transformations, such as rotation.





Consistency

- CART rules with impurity-based splitting are *not* universally consistent. For every impurity criterion, there is a distribution of the data for which things go wrong.
- There are however other universally consistent decision tree rules. For example, if splitting does not depend on the labels, and these are used only for majority voting at the leaves, we can apply Theorem 6.1 in DGL: the tree rule is consistent if its partition $A(x)$ is such that:
  - $\mathrm{diam}(A(X)) \to 0$ in probability
  - $N(X) \to \infty$ in probability
- It is possible to have universally consistent tree rules that use the labels in splitting; see DGL Theorem 21.2.