
Decision Trees

Prof. Richard Zanibbi

Decision Tree Classifier

Design

Hierarchical representation for a decision process. Decision tree used by answering a series of questions regarding properties of a situation or thing.

- Widely used: taxonomy, medical diagnosis, technical manuals, etc.
- Example: 20 Questions (or, nearly equivalently, 'Animal, Vegetable, or Mineral')

Tree Structure

'Questions' at internal nodes, decisions at leaf nodes. For classification, questions are regarding attributes (feature values). Examples of questions:

- 'What is the patient's age?' (non-negative value (*quantitative*))
- 'Does it have a backbone?' (binary (yes/no) response)
- 'Which of the following is your favorite food?' (*qualitative: nominal*)

Note

Classification and Regression Trees (CART)

(Breimann et al., 1984)

These will be the type of tree that we will be studying primarily. As the names suggests, the method may be used both for discrete output domains (classification), or continuous output domains (regression).

The Decision Tree represents a **function** from features to classes or values.

Example

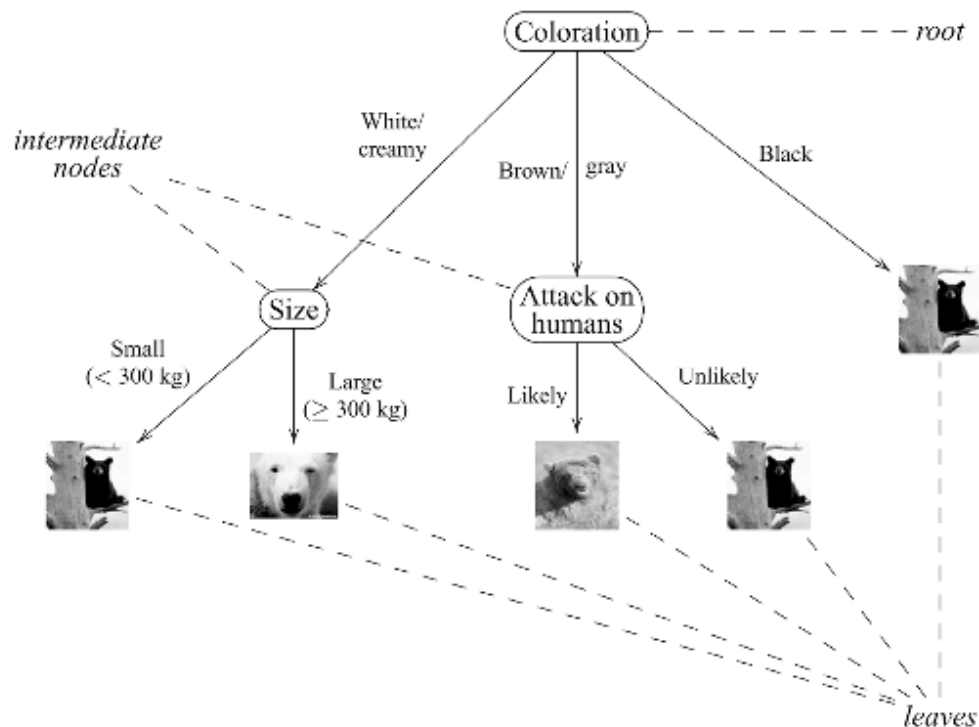
Ursus maritimus
(Polar Bear)



Ursus americanus
(American Black Bear)



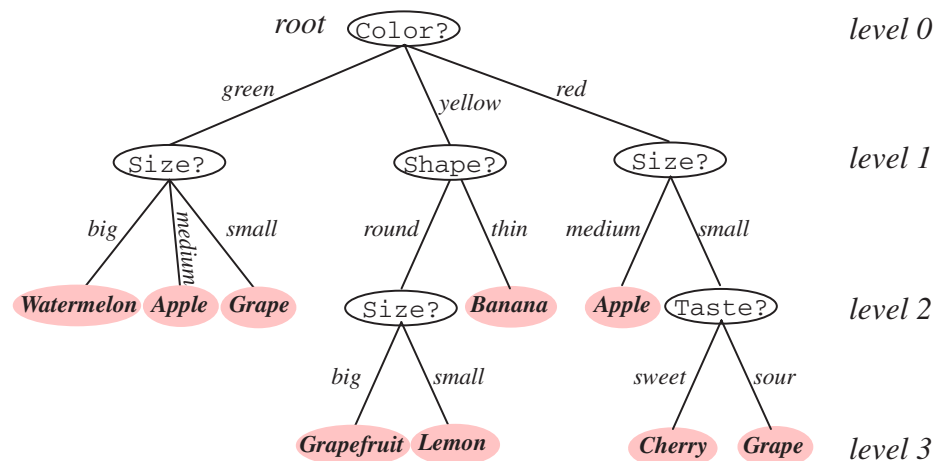
Ursus arctos
(Grizzly Bear)



From training data, more than one type of bear belongs to each leaf node (decision)

i.e. no perfect decisions in this tree; resubstitution error $> 0\%$.

Normally one feature is partitioned at each node, for ease of understanding.



N-ary decision trees
may be converted to
binary trees without
any loss of information.

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*. Copyright © 2001 by John Wiley & Sons, Inc.

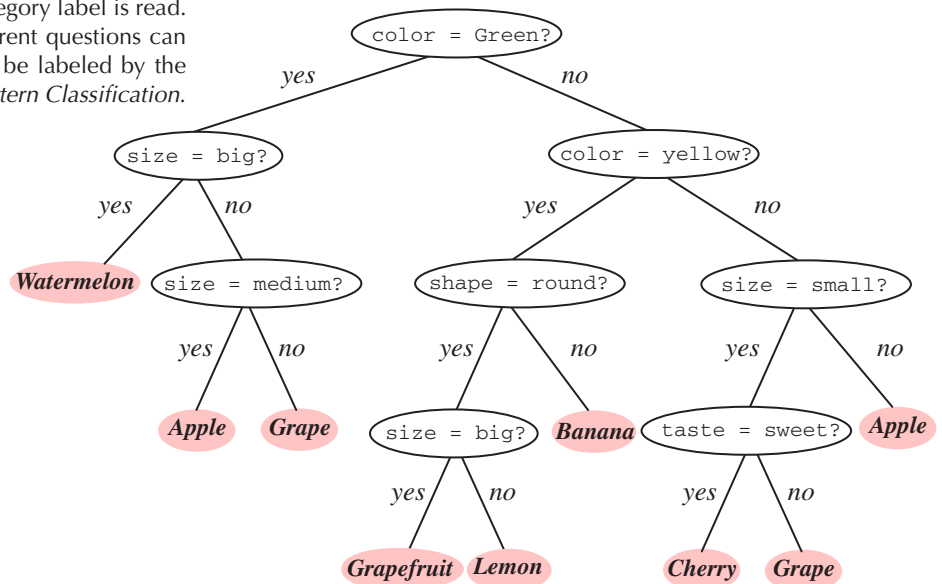
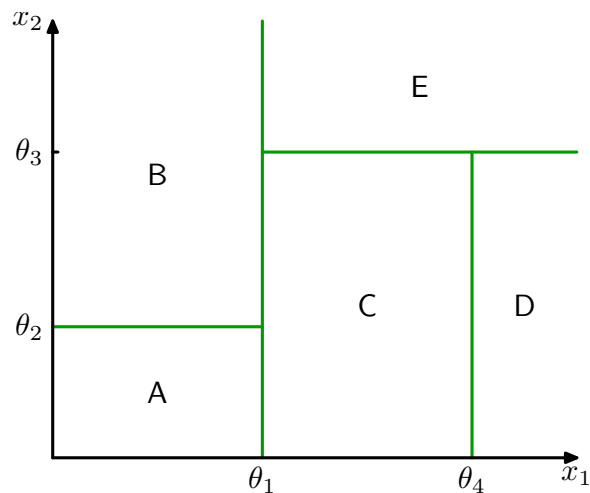
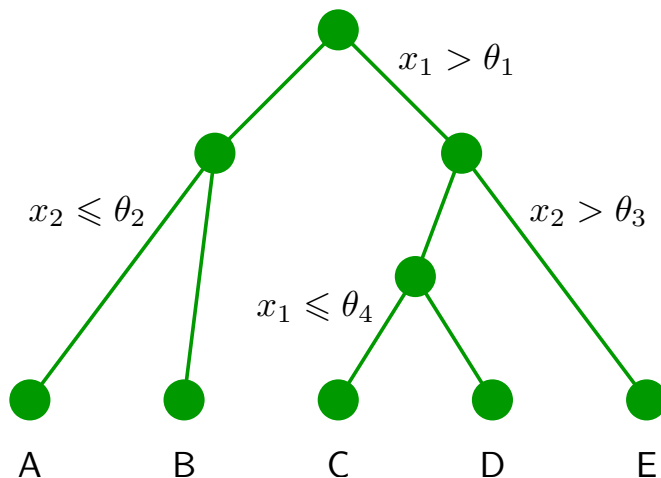


FIGURE 8.2. A tree with arbitrary branching factor at different nodes can always be represented by a functionally equivalent binary tree—that is, one having branching factor $B = 2$ throughout, as shown here. By convention the “yes” branch is on the left, the “no” branch on the right. This binary tree contains the same information and implements the same classification as that in Fig. 8.1. From: Richard O. Duda, Peter E. Hart, and David

Feature Space View



A Combination of Models

Exactly one model makes a decision at each location in feature space.

These models may be understood as logical *rules*, e.g.

- $\forall x_1, x_2 (x_1 > \theta_1) \wedge (x_2 > \theta_3) \rightarrow \text{class}(x_1, x_2, E)$

Decision Tree Properties

0% Resubstitution Error (on Training Data)

...if no feature vector has conflicting class labels; can 'tile' classes in feature space.

Additional Key Properties

- **Easily Understood.** The decision process is a simple sequence of questions.
- **Instable:** can 'memorize' the training data, but training set changes may restructure the decision tree.
- **'Non-metric':** feature(s) partitioned at each node to maximize label similarity in child nodes: no distance measure for features is used, vs. *parametric (LDC, QDC) and non-parametric (Histogram, Parzen, k-NN) classifiers.*
 - Permits *qualitative features* (e.g. names of schools or persons)
 - Quantitative (numerical) features: **define split point** in feature dimension

Feature space structure vs. tree size

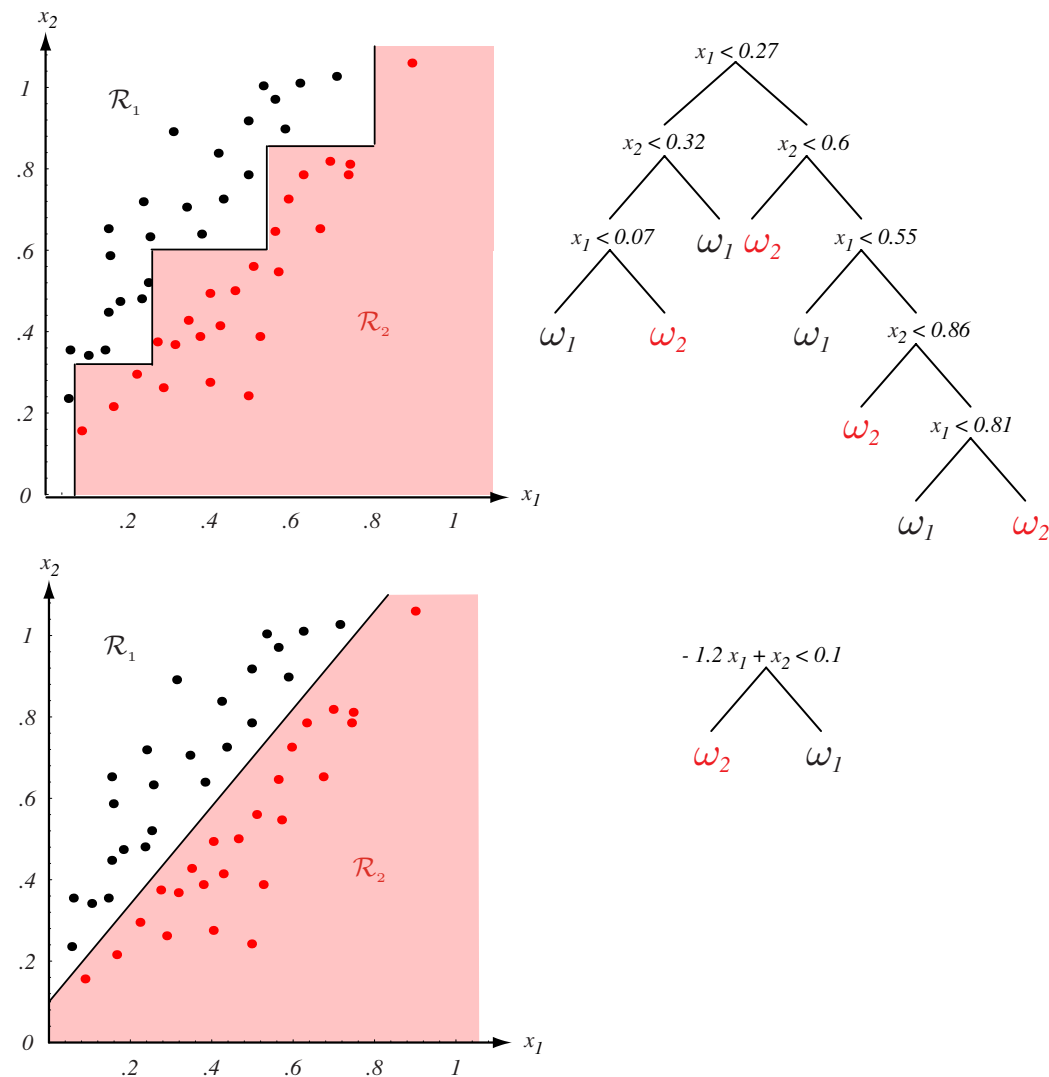


FIGURE 8.5. If the class of node decisions does not match the form of the training data, a very complicated decision tree will result, as shown at the top. Here decisions are parallel to the axes while in fact the data is better split by boundaries along another direction. If, however, “proper” decision 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.

Axis-aligned vs. arbitrary linear cuts

Note: *spherical* cuts also possible: inside/outside as defined by hyperspheres (see Devroye et al.)

While more expressive, non-axis-aligned cuts harder to compute, and harder to interpret.

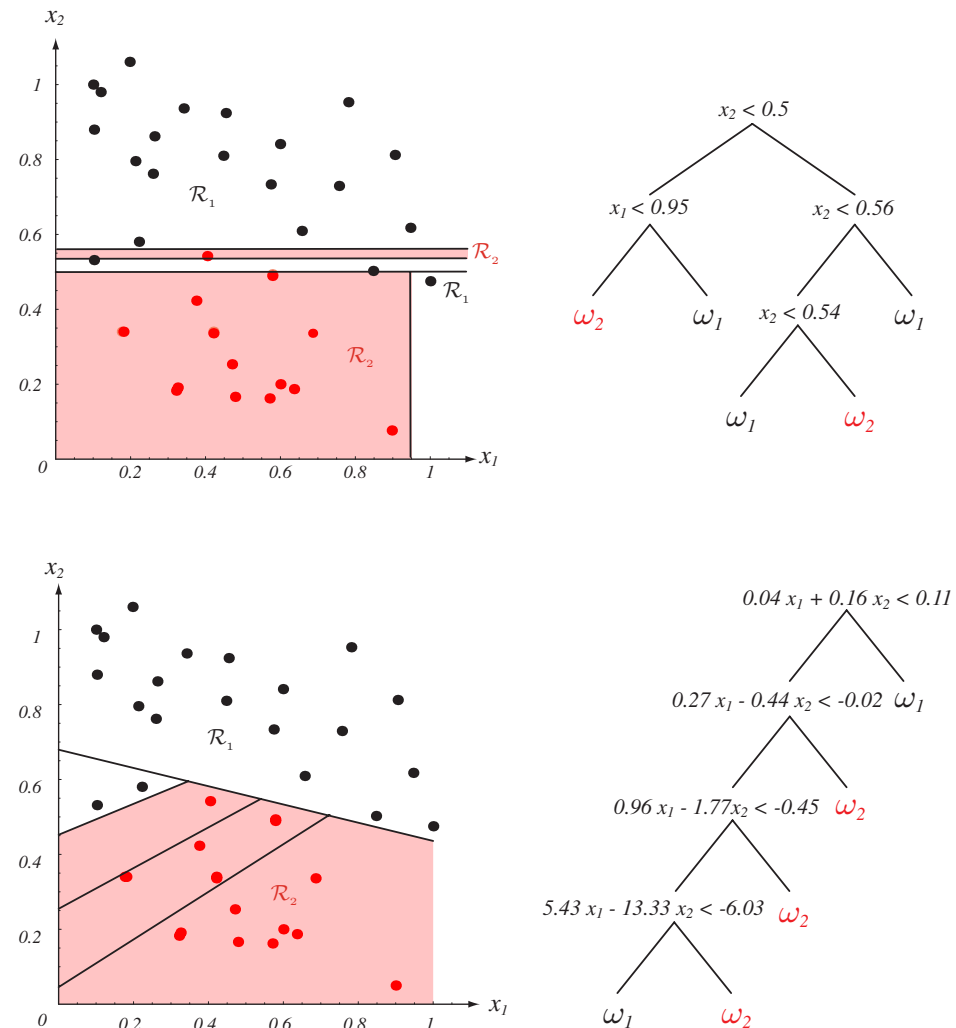


FIGURE 8.6. One form of multivariate tree employs general linear decisions at each node, giving splits along arbitrary directions in the feature space. In virtually all interesting cases the training data are not linearly separable, and thus the LMS algorithm is more useful than methods that require 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.

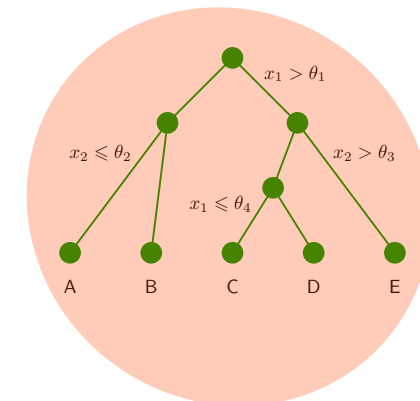
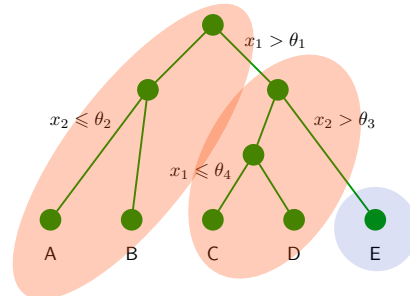
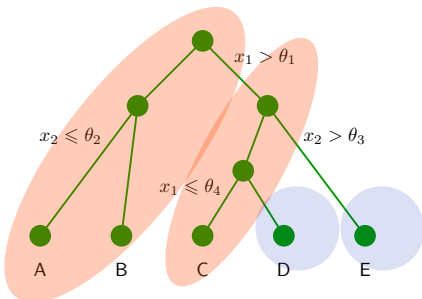
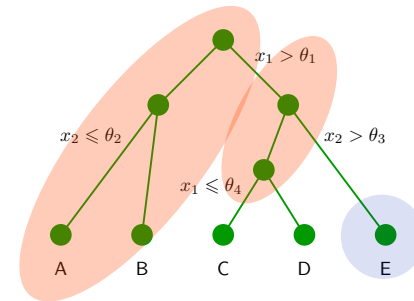
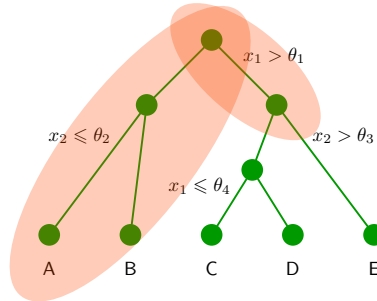
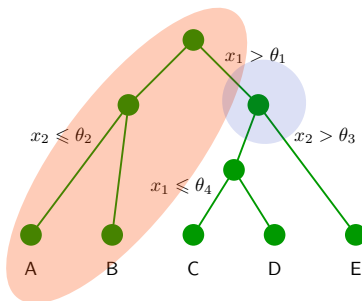
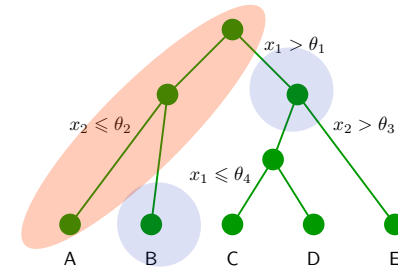
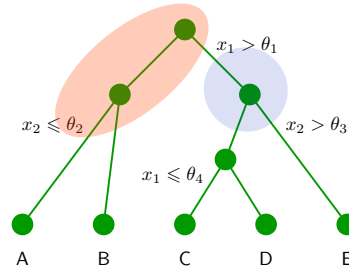
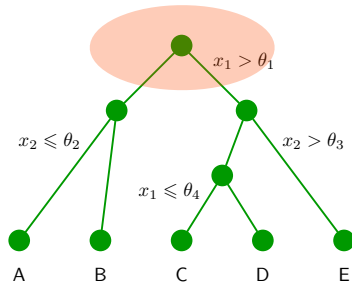
Decision Tree Construction

Initialization: all training samples associated with root node.

1. **Base case:** all samples have same class at current node, or stopping criterion holds, **stop and return**.
2. Find feature partition that minimizes *impurity* of class labels
3. Create child nodes for subsets created by selected partition.
4. Apply steps 1-4 to new child nodes.

Note that this is a greedy algorithm, as the best partition is chosen locally for each node. The is constructed depth-first.

Example: 2D Continuous Features



“Impurity” of Sample Labels at a Node

Definition

Measure of consistency for training data class labels associated with a decision tree node

- “Pure”: all samples have the same class label. Based on training data, correct decision is clear.
- Maximum “impurity”: **all classes have the same number of samples**. Making a decision at such a node amounts to a random guess (training data).

Impurity Measures

Entropy

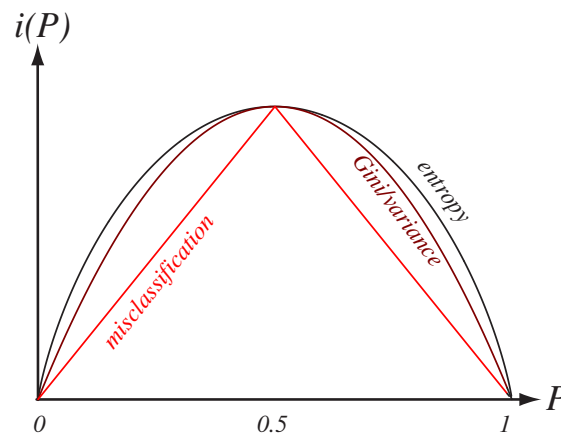
Gini Index

Misclassification

$$i(t) = - \sum_{j=1}^c P_j \log P_j$$

$$i(t) = 1 - \sum_{j=1}^c P_j^2$$

$$i(t) = 1 - \max_{j=1}^c \{P_j\}$$



*Misclassification has discontinuous derivative (not smooth): problems when searching for optimal threshold over continuous-valued features.

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.

Determining Node Splits

Goal

Given the training samples at a node, use the attribute partition that minimizes impurity in the child nodes relative to the parent (the node in question).

- All possible splits are considered (**brute-force/exhaustive search**).


Note

This is a *greedy* decision, where the decision is optimized locally (e.g. based only on the data at the node), rather than globally. This is because determining the globally optimal tree is prohibitively expensive.

- $2^N - 1$ non-empty subsets for N samples (just to start!)
- All possible partitions of N elements into non-empty subsets: the Bell number, where $B_0, B_1 = 1$:

- $2^{10} = 1024, B_{10} = 115,975$

$$B_n = \sum_{k=0}^{n-1} B_k \binom{n-1}{k}$$

- This only accounts for *leaf* nodes, not all possible arrangements of internal nodes (i.e. all possible splits at nodes, which require examining feature values)! 

Determining the Split

Binary Attributes (e.g. gender)

Split on attribute value (e.g. male/female)

Ordinal Attributes (e.g. belt in Karate)

Sort, then consider all possible thresholds ($M - 1$ alternatives for M values)

Nominal Attributes (e.g. names)

Consider impurity of all possible splits for given values ($2^M - 1$ alternatives, number of non-empty subset pairs for M values)

Quantitative Attributes (e.g. real-valued)

Sort training samples by attribute value and consider possible partition values. May be computed at equally-spaced points (e.g. Kuncheva example, D estimates), as the **midpoint** between the best pair of adjacent training samples to split ($N - 1$ alternatives, where N is # samples at the node), etc.

Gain from a Split (Impurity Reduction)

$$\begin{aligned}\Delta i(t) &= i(t) - P(X = 0) \times i(t_0) - P(X = 1) \times i(t_1) \\ &= i(t) - P(X = 0) \times i(t_0) - (1 - P(X = 0)) \times i(t_1)\end{aligned}$$

t: current node, t_0 : left child, t_1 : right child

X: binary or partitioned attribute: $X = 0$ for one value/value set, $X = 1$ for the other.

The ‘gain’ is the difference in impurity between the current node and the weighted average of the impurities at the left/right (true/false) nodes.

Estimate $P(X = 0)$ in the usual fashion, divide number of ‘left’ children by number points at current node (N_L/N)

2 Classes
(‘8’ and ‘stroke’)

Example:
Choosing
split point
for
continuous
features

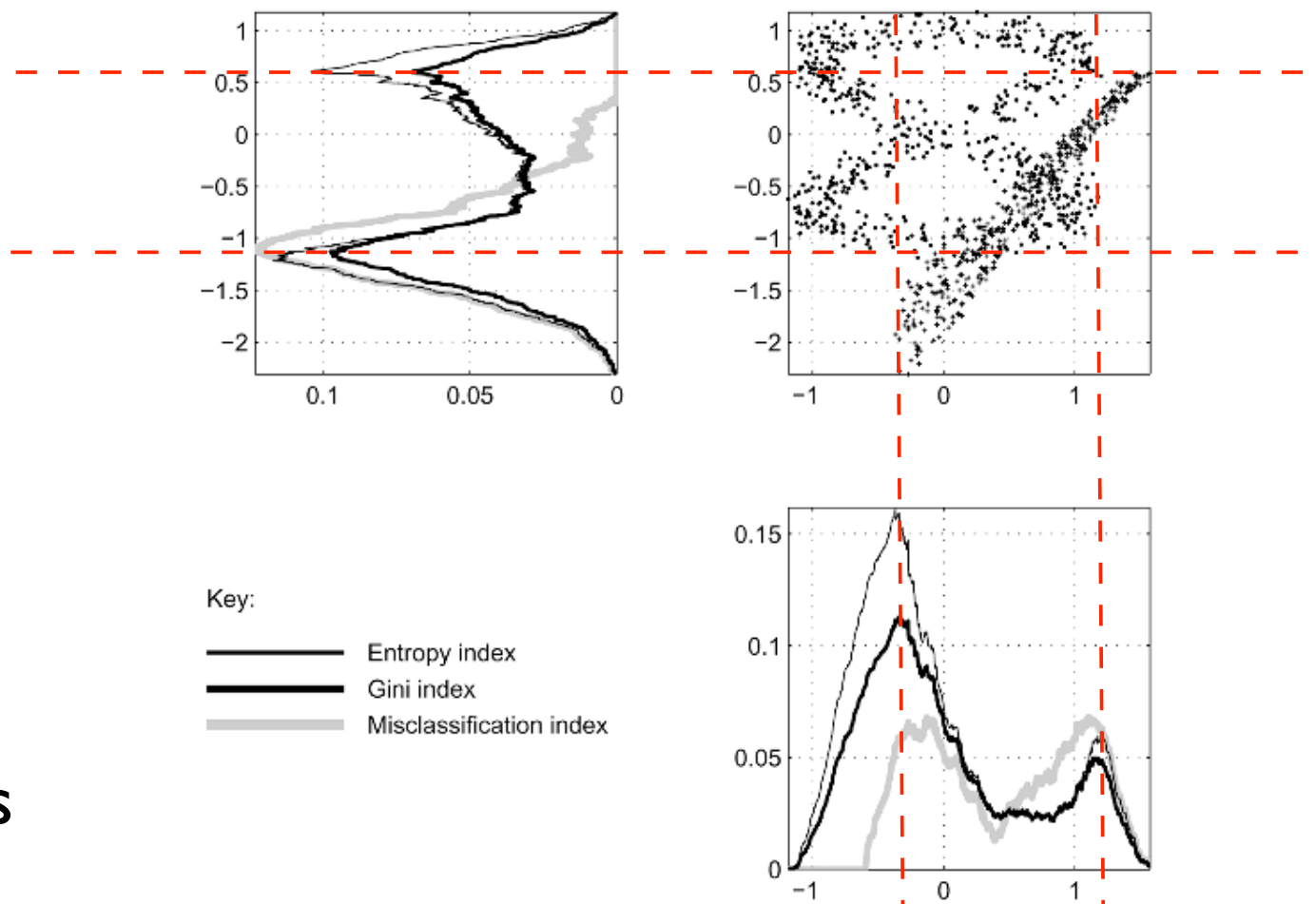


Fig. 2.9 Three indices of impurity used in growing decision trees: Entropy-based, Gini, and Misclassification indices, calculated for 1000 points of split for each of X and Y, for a generated data set (see Figure 1.9).

TABLE 2.3 Maximal Values of Δi and the Respective Split Points for the Three Impurity Indices.

	Entropy		Gini		Misclassification	
	X	Y	X	Y	X	Y
Δi	0.1616	0.1168	0.1127	0.0974	0.0680	0.1230
x_s, y_s	-0.3706	-1.2105	-0.3458	-1.1269	-0.1278	-1.1269

Stopping Criteria

Avoiding Overfitting

To avoid overfitting, we may use a stopping criterion (function) to stop cutting within the decision tree construction algorithm (in base case)

Risk

“Horizon effect”: rule may cause cutting to stop too early (e.g. a very good cut may not appear until after a number of ‘weak’ cuts)

Stopping Criteria

- **Validation set**: stop when error on validation set increases
- **Impurity reduction threshold β** : stop cutting when impurity reduction $i(t) \leq \beta$
- **Training sample threshold k** : stop cutting when number of samples at a node is $\leq k$ (alt. as percentage of training sample size)
- **Penalize tree complexity, using parameter α** : split if the following term is reduced by the split (size = # nodes)

$$\alpha \times \text{size} + \sum_{\text{leaves } t} i(t)$$

Hypothesis Testing (χ^2)

Statistical Hypothesis Testing

Determine whether two sampled distributions differ *significantly* (i.e. low probability of arising from chance, typically .01 or .05)

‘Rejecting the Null Hypothesis’

We wish to determine whether the sample class distributions in the child nodes differ significantly from the class distribution in the parent (if so, we reject the *null hypothesis*, that the two distributions do not differ)

*A brief introduction to hypothesis testing is provided in DHS Appendix A.6; Table A.2 provides critical values for the chi-squared test.

χ^2 Criterion

Two Class Problem (one degree of freedom):

Left Node

$$\chi_L^2 = \frac{(n \times n_{L1} - n_L \times n_1)^2}{n \times n_L \times n_1} + \frac{(n \times n_{L2} - n_L \times n_2)^2}{n \times n_L \times n_2}$$

***Average over Left/Right**

$$\chi^2 = \frac{1}{2}(\chi_L^2 + \chi_R^2) = \frac{n}{2n_R} \chi_L^2 = \frac{n}{2n_L} \chi_R^2$$

Right Node

$$\chi_R^2 = \frac{(n \times n_{R1} - n_R \times n_1)^2}{n \times n_R \times n_1} + \frac{(n \times n_{R2} - n_R \times n_2)^2}{n \times n_R \times n_2}$$

Critical value:
3.84 at .05 level

Multi-class Problem (c-1 degrees of freedom)

Left/right node expressions change: critical value increases as degrees of freedom (# classes) increases (see DHS page 630)

$$\chi_L^2 = \sum_{i=1}^c \frac{(n \times n_{Li} - n_L \times n_i)^2}{n \times n_L \times n_i}$$

$$\chi_R^2 = \sum_{i=1}^c \frac{(n \times n_{Ri} - n_R \times n_i)^2}{n \times n_R \times n_i}$$

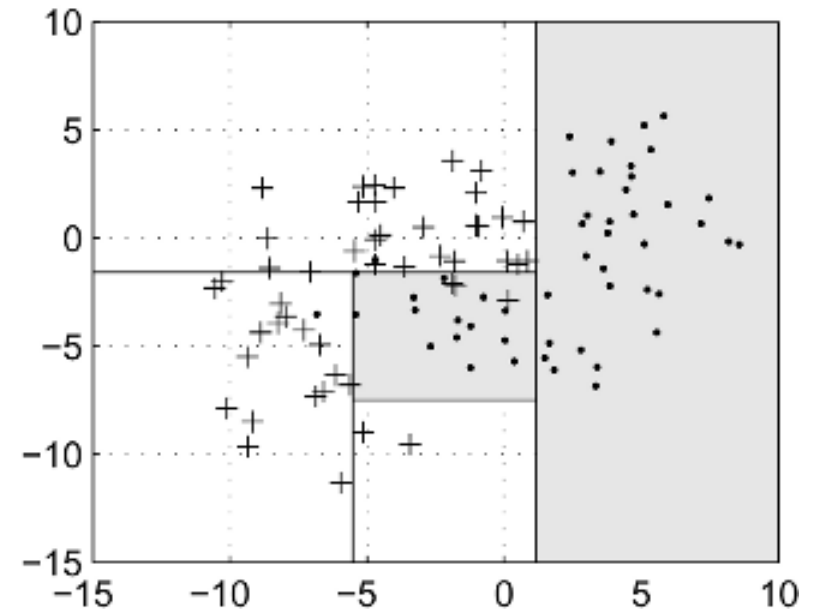
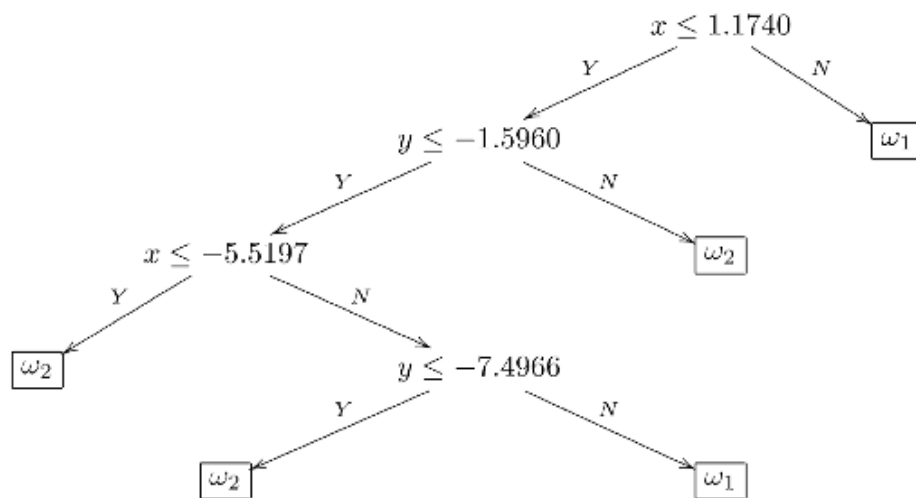


Fig. 2.10 Graphical representation of the tree classifier for the banana data. A threshold $t = 3$ was used to preprune the tree. Training error = 5%, testing error = 11%.

TABLE 2.4 A Tabular Description of the Tree Classifier for the Banana Data with Threshold $t = 3$.

Index	Feature (or Class Label)	Split Point	Left Child Node	Right Child Node
1	1	1.1740	2	9
2	2	-1.5960	3	8
3	1	-5.5197	4	5
4	2	0	0	0 (leaf)
5	2	-7.4966	6	7
6	2	0	0	0 (leaf)
7	1	0	0	0 (leaf)
8	2	0	0	0 (leaf)
9	1	0	0	0 (leaf)

Tree created, **stopping** if single sample at node, or chi-squared value is ≤ 3 .

Training error is 5%, testing 11%.

Pruning

Stopping...

Can be shortsighted, and prevent beneficial splits ('horizon effect')

Preferred method: Grow, then prune

Intuition: fit the data perfectly (build entire tree), then prune decisions to prevent over-fitting, while avoiding the horizon effect.

Pruning Criteria

- **Reduced Error Pruning**: keep aside a separate data set (similar to validation set). Bottom-up from leaf parent nodes to the root, prune tree at each node, assigning the class with the most instances at the node. Stop pruning along a path when error on pruning set is higher for pruned vs. original tree.
- **Pessimistic error pruning**: a *top-down* pruning method which takes the size of subtrees into account when considering pruning locations.
- **Critical Value Pruning (CVP)**: bottom-up from leaf parent nodes to the root, prune all nodes where the increase in error on the training set is below a threshold ϵ . Can produce a family of trees by decreasing the threshold (family of nested trees), choose best tree for an independent training set.

Pruning Criteria, Cont'd

- **Cost-complexity pruning** (CART): Use increase in apparent error rate per leaf on training data (α) as pruning criterion. Iterate to obtain a family of trees, selecting the 'best' one using a pruning set or cross-validation.
 - First prune using $\alpha = 0$ (no increase in error), identify smallest α_1 for the resulting tree, and prune using again using α_1 to obtain T_1 .
 - Repeat until T_k , where only the root remains.
 - Identify best tree in the generated set. Below, t is the current node, L_T the number of leaves rooted at node T :

$$\alpha = \frac{e(t) - \sum_{l \in \mathcal{L}_t} e(l)}{n(|\mathcal{L}_t| - 1)}$$

Pruning, Cont'd

- **Error-Based Pruning:** nodes may be replaced by the *subtree* rooted at a child node ('grafting,' restructuring the tree *internally*). Makes use of confidence interval for error at a node; pruning decision considers expected error, errors of children, error of child node with the most samples.

Utility of Pruning (Esposito et al.)

In a set of experiments in different domains, found pruning helps more often than not. Surprisingly, often better results obtained by using all data for training and pruning, rather than using a separate pruning set.

Full tree: 25 nodes

Pruning results shown:
Using CVP with X^2 , pruning
if $X^2 < \tau$.

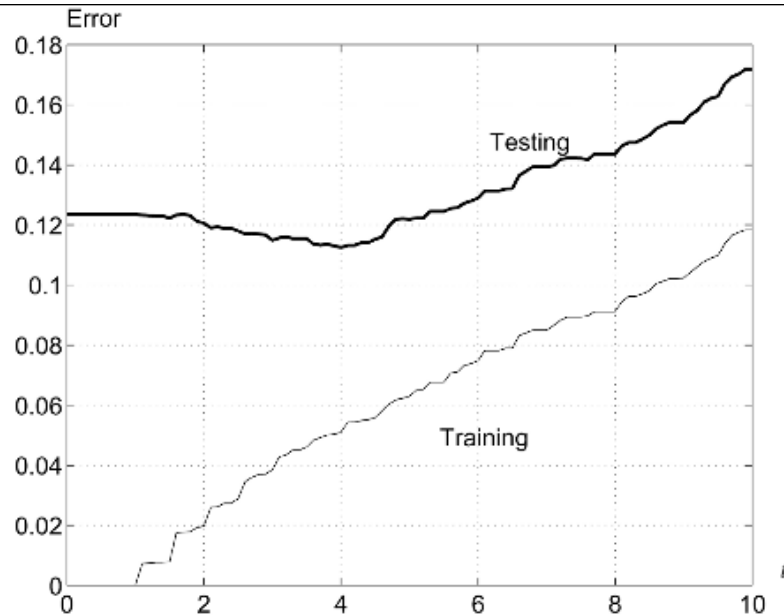


Fig. 2.12 Training and testing errors of a tree classifier as functions of the pruning threshold (averaged across 100 experiments).

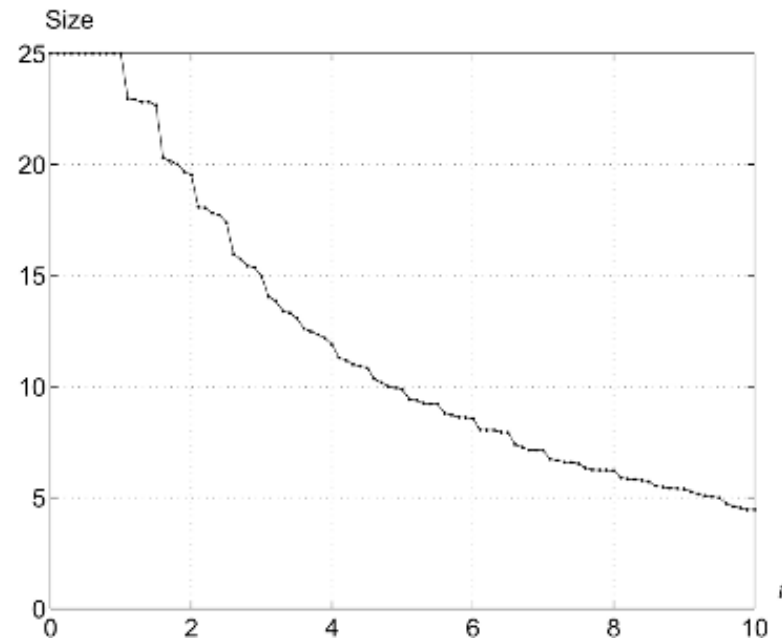


Fig. 2.13 Size of the tree classifier as a function of the pruning threshold (averaged across 100 experiments).

Missing Attributes

Missing features ('attributes')

Especially common in data collected in real life, e.g. data from experiments involving human participants, sensor/hardware failures.

- Metric-based methods (e.g. k-NN) require *all* features.

Accommodation using Decision Trees:

- Interpolate the missing value (e.g. using similar samples)
- Define alternative cuts at every node, using the remaining features. One method is to rank remaining features by the number of samples that can be identically/similarly partitioned in the manner of the primary/original feature selected at a node.

Missing Attributes, Cont'd

Classification Using Missing Attributes

Classify as before, using interpolated value or 'surrogate' feature partition at nodes where attributes are missing.

Extensions: ID3, C4.5

ID3 (Quinlan): 'Interactive Dichotomizer'

For nominal data: all continuous variables must first be discretized.

- Each node has all possible attribute values as children ('uses up' the feature), maximum depth of n , where n is the number of features, optionally followed by pruning.
- Impurity measured using *gain ratio impurity*, to prevent preference for multiple vs. binary splits:

$$\Delta i_M = \frac{\Delta i}{-\sum_{i=1}^M P_i \log P_i}$$

ID3/C4.5 Cont'd

C4.5 (Quinlan again)

Combines CART trees with ID3 (and very popular):

- Continuous data: managed per CART
- Nominal data: managed per ID3.