Decision Trees
(Sections 8.1-8.4)

- Non-metric methods
- CART (Classification & regression Trees)
- Number of splits
- Query selection & node impurity
- Multiway splits
- When to stop splitting?
- Pruning
- Assignment of leaf node labels
- Feature choice
- Multivariate decision trees
- Missing attributes

Data Type and Scale

- Data type refers to the degree of quantization in the data
  - binary feature: have exactly two values (Yes-No response)
  - discrete feature: finite, usually small, number of possible values (gray values in a digital image)
  - continuous feature: any real value in a fixed range of values
- Data scale indicates the relative significance of numbers
  - qualitative scales
    - Nominal (categorical): not really a scale since numerical values are simply used as names; e.g., (yes, no) response can be coded as (0,1) or (1,0) or (50,100); numerical values are meaningless in a quantitative sense
    - Ordinal: numbers have meaning only in relation to one another (e.g., one value is larger than the other); e.g., scales (1, 2, 3), (10, 20, 30), and (1, 20, 300) are all equivalent from an ordinal viewpoint (military rank)
  - quantitative scales
    - Interval scale: separation between numbers has meaning; a unit of measurement exists, and the interpretation of the numbers depends on this unit (Fahrenheit scale for temperature. Equal differences on this scale represent equal differences in temperature, but a temperature of 30 degrees is not twice as warm as one of 15 degrees.
    - Ratio scale: numbers have an absolute meaning; an absolute zero exists along with a unit of measurement, so the ratio between two numbers has meaning (height)

Decision Trees

- Also known as
  - Hierarchical classifiers
  - Tree classifiers
  - Multistage classification
  - Divide & conquer strategy
- A single-stage classifier assigns a test pattern X to one of C classes in a single step: compute the posteriori probability for each class & choose the class with the max posteriori
- Limitations of single-stage classifier
  - A common set of features is used for distinguishing all the classes; for large no. of classes, this common feature set may not be the best for specific pairs of classes
  - Requires a large no. of features when the no. of classes is large
  - For each test pattern, C posteriori probs. need to be computed
  - Does not perform well when classes have multimodal distributions
  - Not easy to handle nominal data (discrete features without any natural notion of similarity or even ordering)

Decision Trees

- Most pattern recognition methods address problems where feature vectors are real valued and there exists some notion of metric
- Suppose the classification involves nominal data– attributes that are discrete & without any natural notion of similarity or even ordering
- Describe patterns by using a list of attributes rather than by vectors of real numbers
- Describe a fruit by {color, texture, taste, size}
  - {red, shiny, sweet, small}
- How to learn categories using such non-metric data?

Decision Trees

- Classify a pattern through a sequence of questions (20-question game); next question asked depends on the answer to the current question
- This approach is particularly useful for non-metric data; questions can be asked in a “yes-no” or “true-false” style that do not require any notion of metric
- Sequence of questions is displayed in a directed decision tree
- Root node, links or branches, leaf or terminal nodes
- Classification of a pattern begins at the root node until we reach the leaf node; pattern is assigned the category of the leaf node
- Benefit of decision tee:
  - Interpretability: a tree can be expressed as a logical expression
  - Rapid classification: a sequence of simple queries
  - Higher accuracy & speed:

Decision Tree

Seven-class, 4-feature classification problem

Apple = (green AND medium) OR (red AND medium) = (Medium AND NOT yellow)

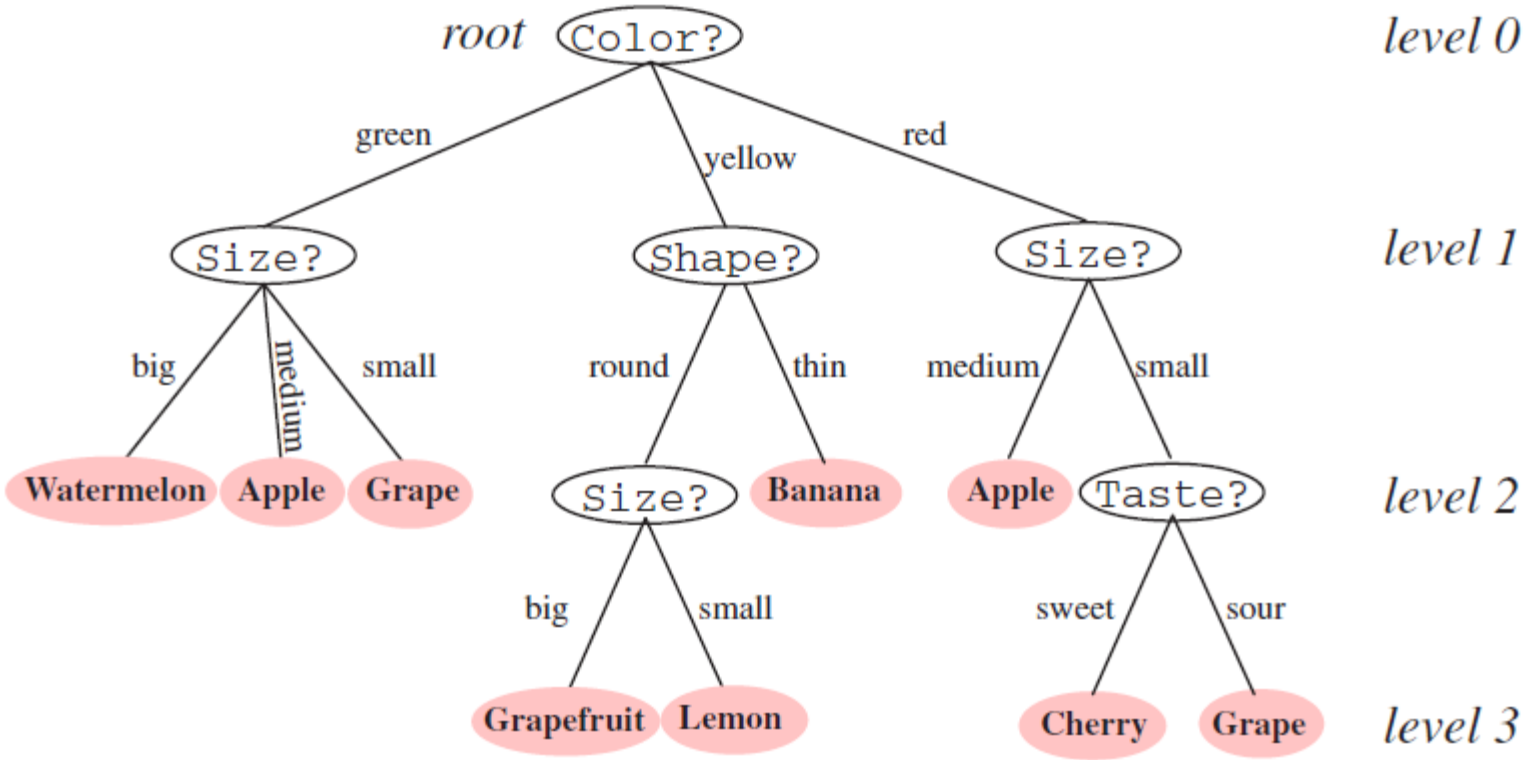


Figure 8.1: Classification in a basic decision tree proceeds from top to bottom. The questions asked at each node concern a particular property of the pattern, and the downward links correspond to the possible values. Successive nodes are visited until a terminal or leaf node is reached, where the category 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**).

How to Grow A Tree?

- Given a set *D* of labeled training samples and a set of features
- How to organize the tests into a tree? Each test or question involves a single feature or subset of features
- A decision tree progressively splits the training set into smaller and smaller subsets
- Pure node: all the samples at that node have the same class label; no need to further split a pure 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 & Regression Trees)

Classification & Regression Tree (CART)

- Six types of questions
  - Should the attributes (answers to questions) be binary or multivalued? In other words, how many splits should be made at a node?
  - Which feature or feature combination should be tested at a node?
  - When should a node be declared a leaf node?
  - If the tree becomes “too large”, can it be pruned to make it smaller and simpler?
  - If a leaf node is impure, how should category label be assigned to it?
  - How should missing data be handled?

Number of Splits

Binary tree: every decision can be represented using just binary outcome; tree of Fig 8.1 can be equivalently written as

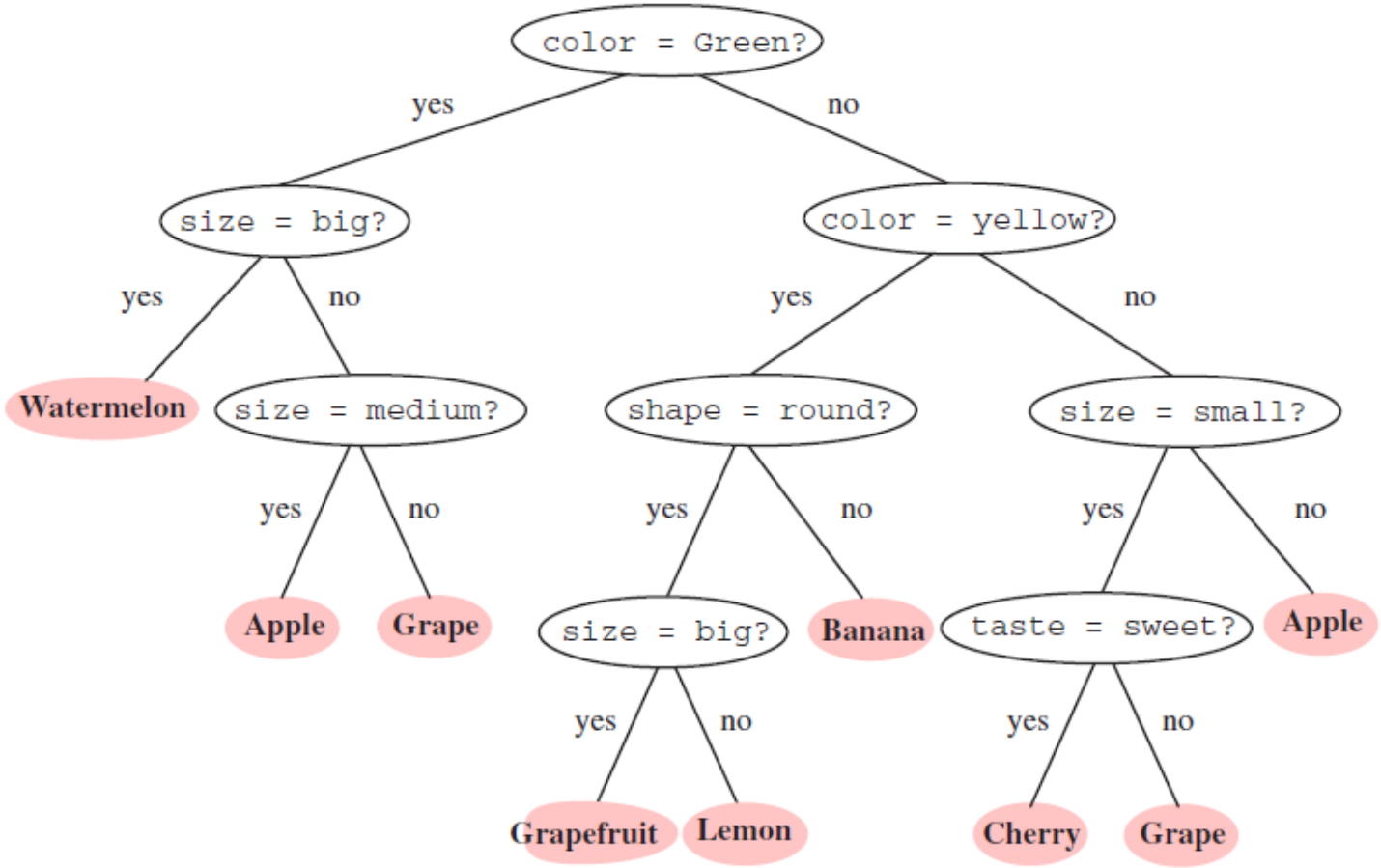


Figure 8.2: A tree with arbitrary branching factor at different nodes can always be represented by a functionally equivalent binary tree, i.e., one having branching factor $B = 2$ throughout. By convention the "yes" branch is on the left, the "no" branch on the right. This binary tree contains the same information and implements the same classification as that in Fig. 8.1.

Query Selection & Node Impurity

- Which attribute test or query should be performed at each node?
- Fundamental principle of simplicity: obtain 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 $x_i <= x_{is}$ leads to hyperplanar decision boundaries that are perpendicular to the coordinate axes (monothetic tree; one feature/node))

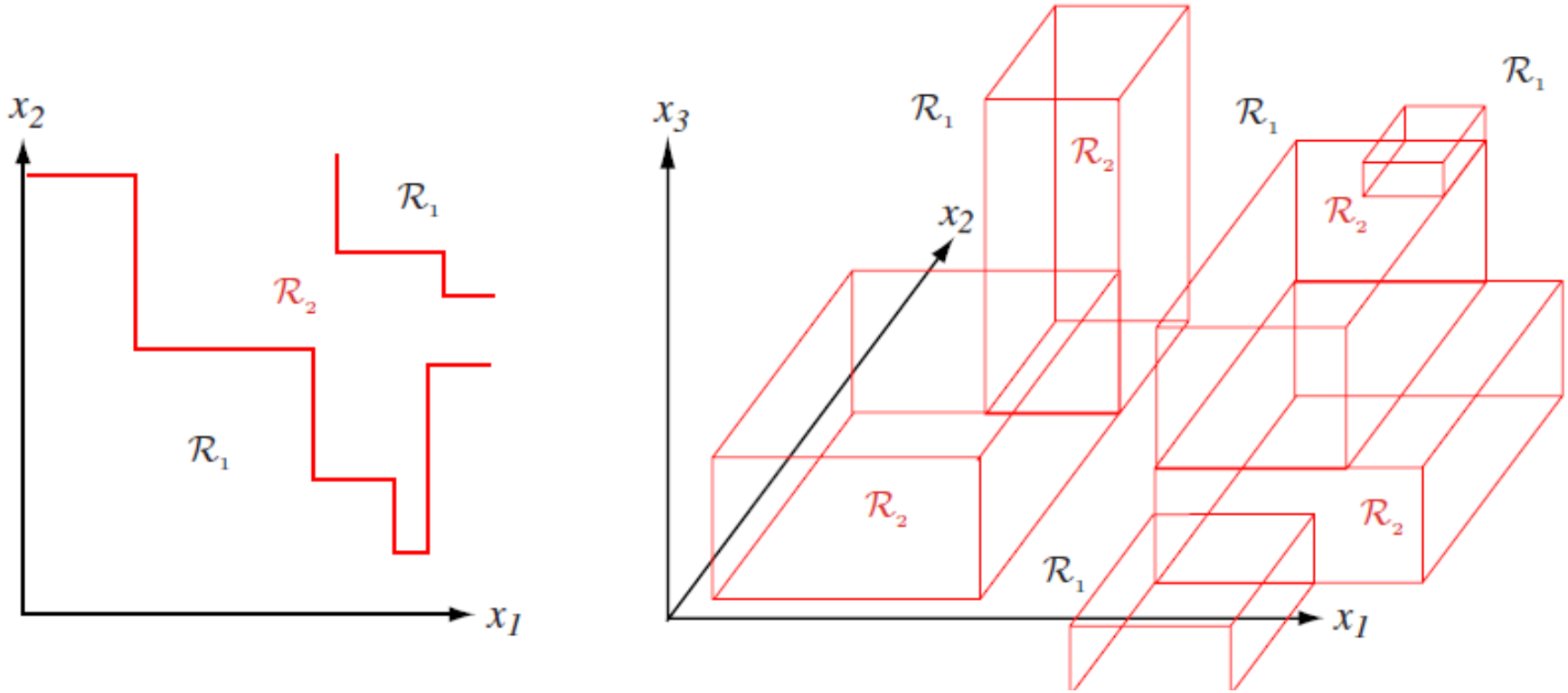


Figure 8.3: Monothetic decision trees create decision boundaries with portions perpendicular to the feature axes. The decision regions are marked $\mathcal{R}_1$ and $\mathcal{R}_2$ in these two-dimensional and three-dimensional two-category examples. With a sufficiently large tree, any decision boundary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

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

$$i(N) = -\sum_j P(\omega_j)\ \log_2\ P(\omega_j), \qquad (1)$$

- Gini impurity
  - Expected error rate at node N if the category label is selected randomly from the class distribution present at N

$$i(N) = \sum_{i \neq j} P(\omega_i) P(\omega_j) = \frac{1}{2}\left[1 - \sum_j P^2(\omega_j)\right] \qquad (3)$$

- Misclassification impurity
  - Minimum probability that a training pattern will be misclassified at N

$$i(N) = 1 - \max_j P(\omega_j), \qquad (4)$$

Query Selection and Node Impurity

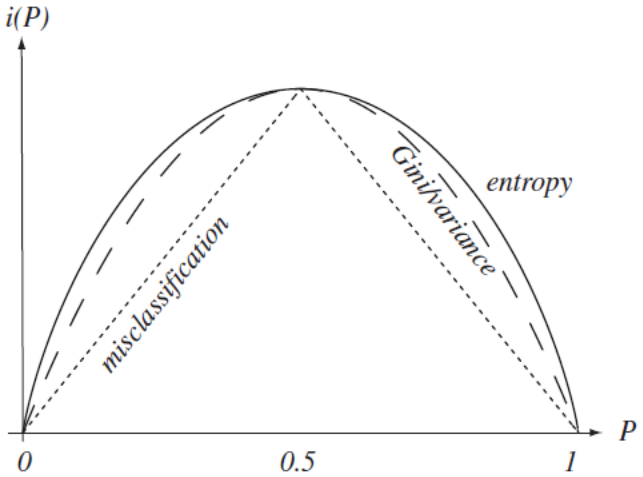


Figure 8.4: For the two-category case, the impurity functions peak at equal class frequencies and the variance and the Gini impurity functions are identical. To facilitate comparisons, the entropy, variance, Gini and misclassification impurities (given by Eqs. 1 – 4, respectively) have been adjusted in scale and offset to facilitate comparison; such scale and offset does not directly affect learning or classification.

- Given a partial tree down to node N, what value *s* to choose for property test T?
- Choose the query at node N to decrease the impurity as much as possible
- Drop in impurity is defined as

$$\Delta i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R), \qquad (5)$$

Best query value s is the choice for test T that maximizes the drop in impurity

Optimization in Eq. (5) is local—done at a single node. This greedy method does not assure that global optimum of impurity will be reached

When to Stop Splitting?

- If we continue to grow the tree fully until each leaf node corresponds to the lowest impurity, then the data have typically been overfit; in the limit, each leaf node has only one pattern!
- If splitting is stopped too early, error on training data is not sufficiently low and performance will suffer
- Validation and cross-validation
  - Continue splitting until error on validation set is minimum
  - Cross-validation relies on several independently chosen subsets
- Stop splitting when the best candidate split at a node reduces the impurity by less than the preset amount (threshold)
- How to set the threshold? Stop when a node has small no. of points or some fixed percentage of total training set (say 5%)
- Trade off between tree complexity or size vs. test set accuracy

Pruning

- Occasionally, stopping tree splitting suffers from the lack of sufficient look ahead
- A stopping condition may be met too early for overall optimal recognition accuracy
- Pruning is the inverse of splitting
- Grow the tree fully—until leaf nodes have minimum impurity. Then all pairs of leaf nodes (with a common antecedent node) are considered for elimination
- Any pair whose elimination yields a satisfactory (small) increase in impurity is eliminated, and the common antecedent node is declared as leaf node

Example 1: A Simple Tree

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

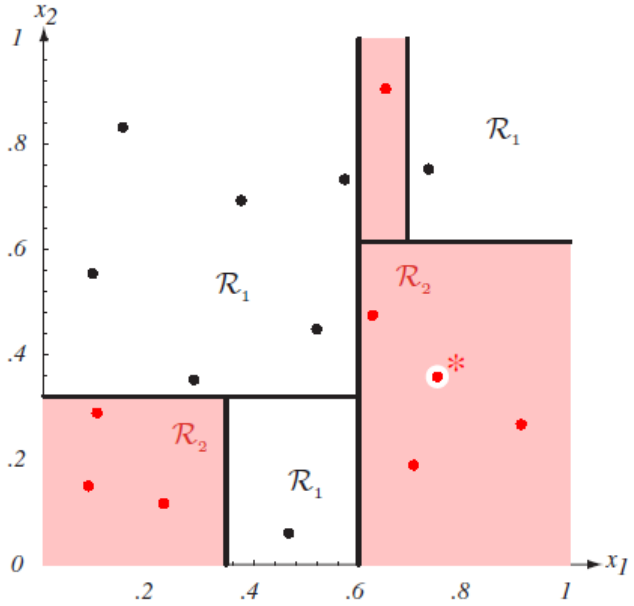
$$i(N) = -\sum_j P(\omega_j) \log_2 P(\omega_j), \tag{1}$$

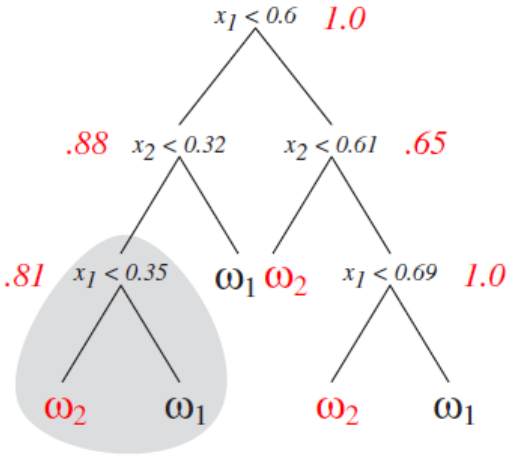
ω_1 (black)	
x_1	x_2
.15	.83
.09	.55
.29	.35
.38	.70
.52	.48
.57	.73
.73	.75
.47	.06

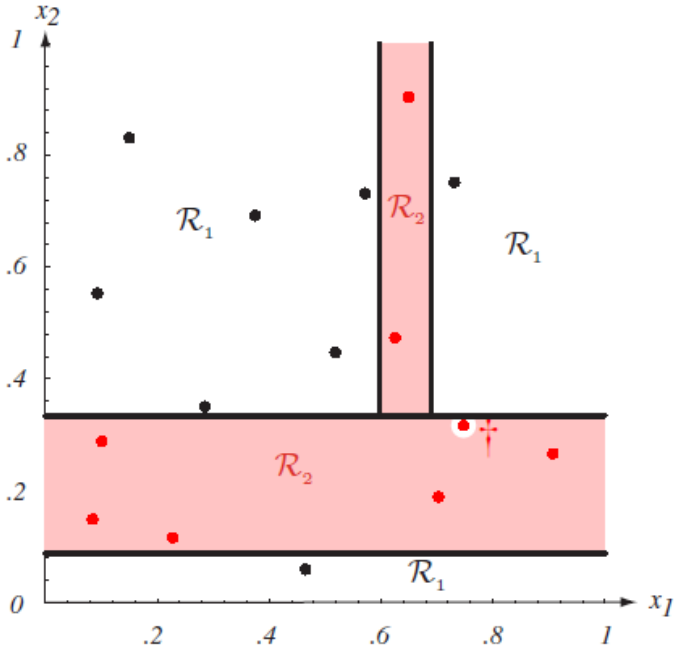
ω_2 (red)	
x_1	x_2
.10	.29
.08	.15
.23	.16
.70	.19
.62	.47
.91	.27
.65	.90
.75	.36* (.32†)

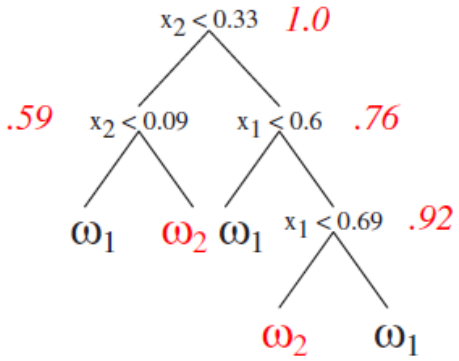
Example 1. Simple Tree

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; due to discrete & greedy nature of CART

Decision Tree

$$i(N_{root}) = -\sum_{i=1}^{2} P(\omega_i)\log_2 P(\omega_i) = -[.5\log_2.5 + .5\log_2.5] = 1.0.$$

For simplicity we consider candidate splits parallel to the feature axes, i.e., of the form "is $x_i < x_{is}$?". By exhaustive search of the $n-1$ positions for the x_1 feature and $n-1$ positions for the x_2 feature we find by Eq. 5 that the greatest reduction in the impurity occurs near $x_{1s} = 0.6$, and hence this becomes the decision criterion at the root node. We continue for each sub-tree until each final node represents a single category (and thus has the lowest impurity, 0), as shown in the figure. If pruning were invoked, the pair of leaf nodes at the left would be the first to be deleted (gray shading) since there the impurity is increased the least. In this example, stopped splitting with the proper threshold would also give the same final network. In general, however, with large trees and many pruning steps, pruning and stopped splitting need not lead to the same final tree.

This particular training set shows how trees can be sensitive to details of the training points. If the ω_2 point marked * in the top figure is moved slightly (marked †), the tree and decision regions differ significantly, as shown at the bottom. Such instability is due in large part to the discrete nature of decisions early in the tree learning.

Feature Choice

As with most pattern recognition methods, tree-based methods work best if proper features are used; preprocessing by PCA can be effective because it finds "important" axes

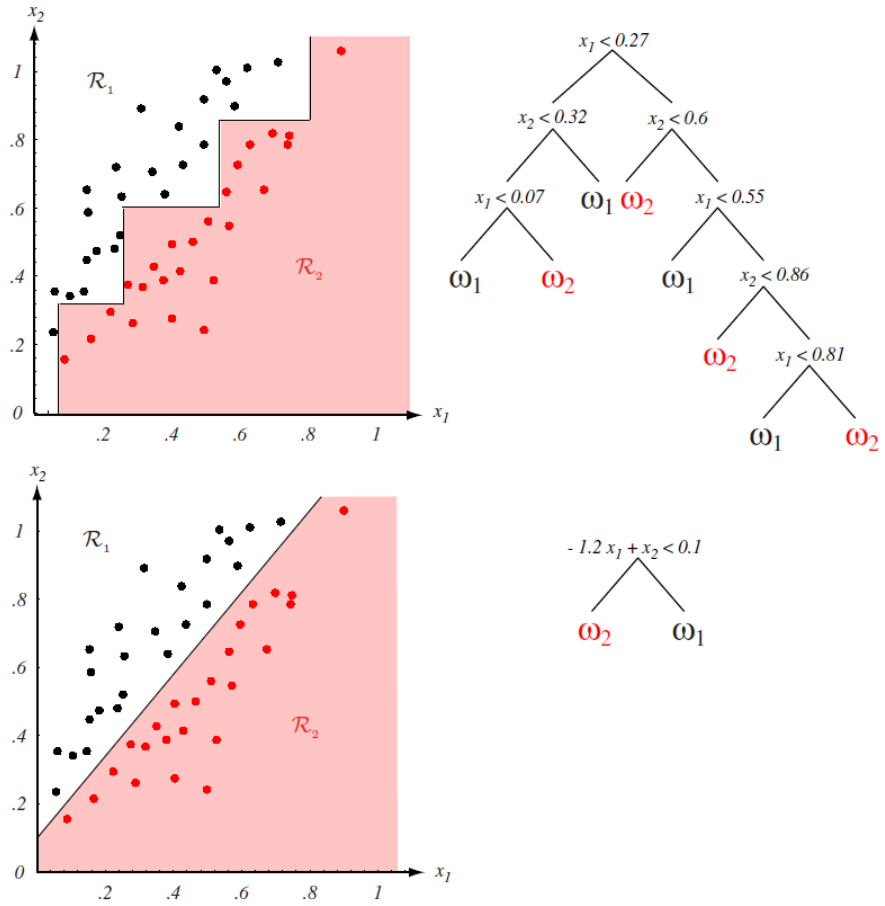


Figure 8.5: If the class of node decisions does not match the form of the training data, a very complicated decision tree will result, as shown at the top. Here decisions are parallel to the axes while in fact the data is better split by boundaries along another direction. If however "proper" decision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

Multivariate Decision Trees

Allow splits that are not parallel to feature axes

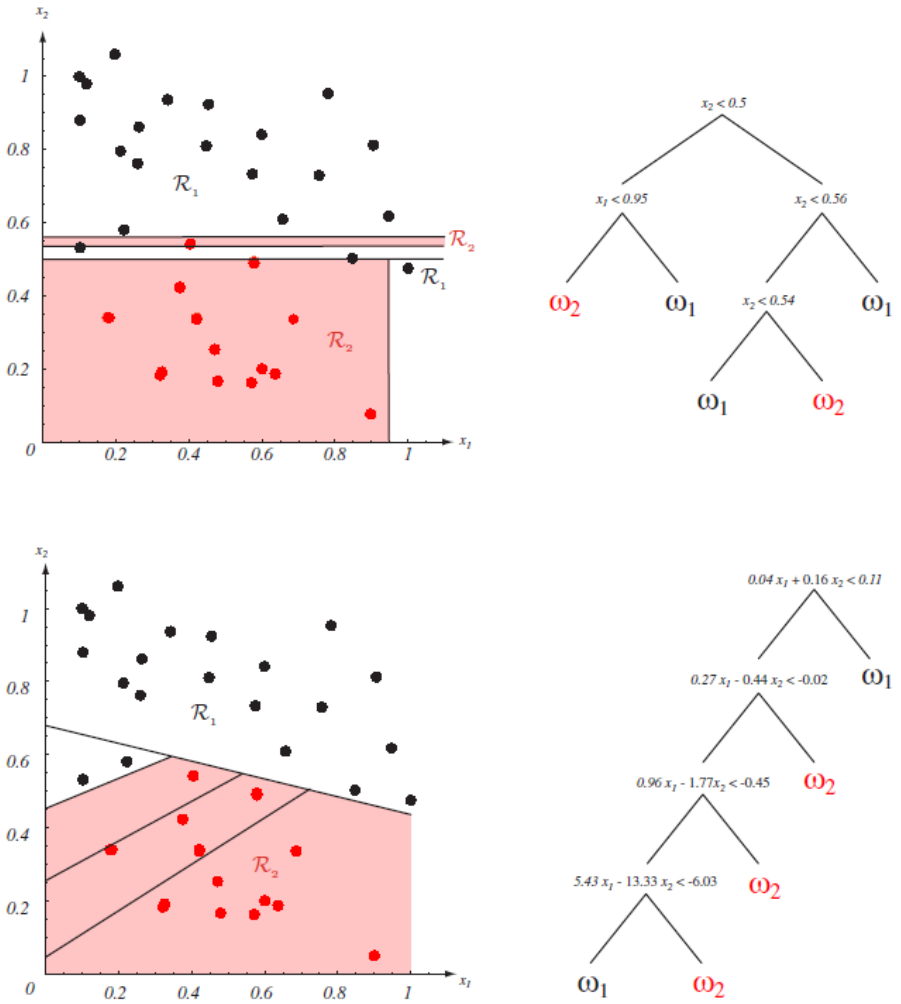


Figure 8.6: One form of multivariate tree employs general linear decisions at each node, giving splits along arbitrary directions in the feature space. In virtually all interesting cases the training data is not linearly separable, and thus the LMS algorithm is more useful than methods that require the data to be linearly separable, even though the LMS need not yield a minimum in classification error (Chap. 5). The tree at the bottom can be simplified by methods outlined in Sect. 8.4.2.

Popular Tree Algorithms

- ID3: Third in a series of "Interactive dichotomizer" procedures. Intended to be used for nominal (unordered) inputs only. If the problem involves real-valued variables, they are first binned into intervals, each interval being treated as an unordered nominal attribute
- C4.5: the successor and refinement of ID3 is the most popular classification tree method. Real valued variables are treated the same as in CART. Multiway (B>2) splits are used with nominal data

Missing Attributes

- Missing attributes during training, during classification, or both
- Naïve approach; delete any such deficient patterns
- Calculate impurities at anode N using only the attribute information present

Decision Tree – IRIS data

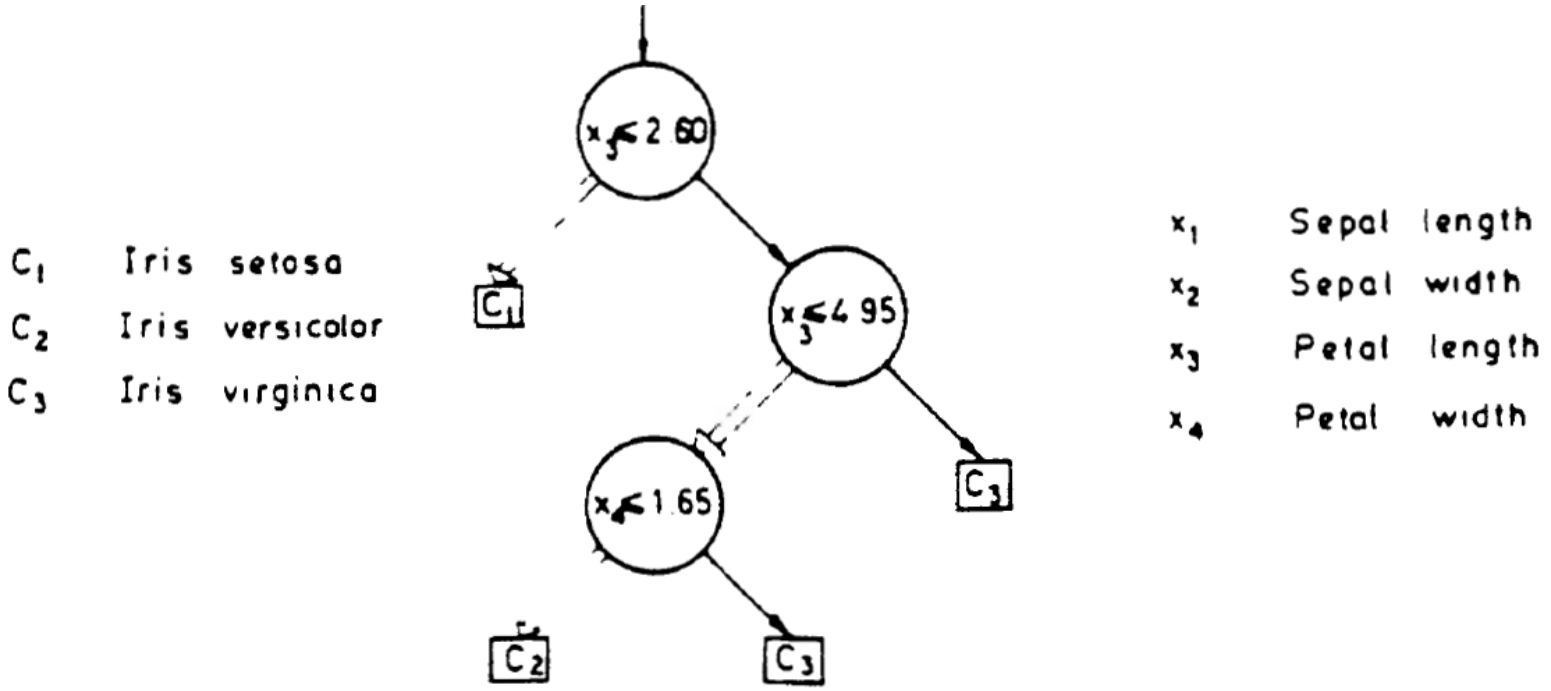


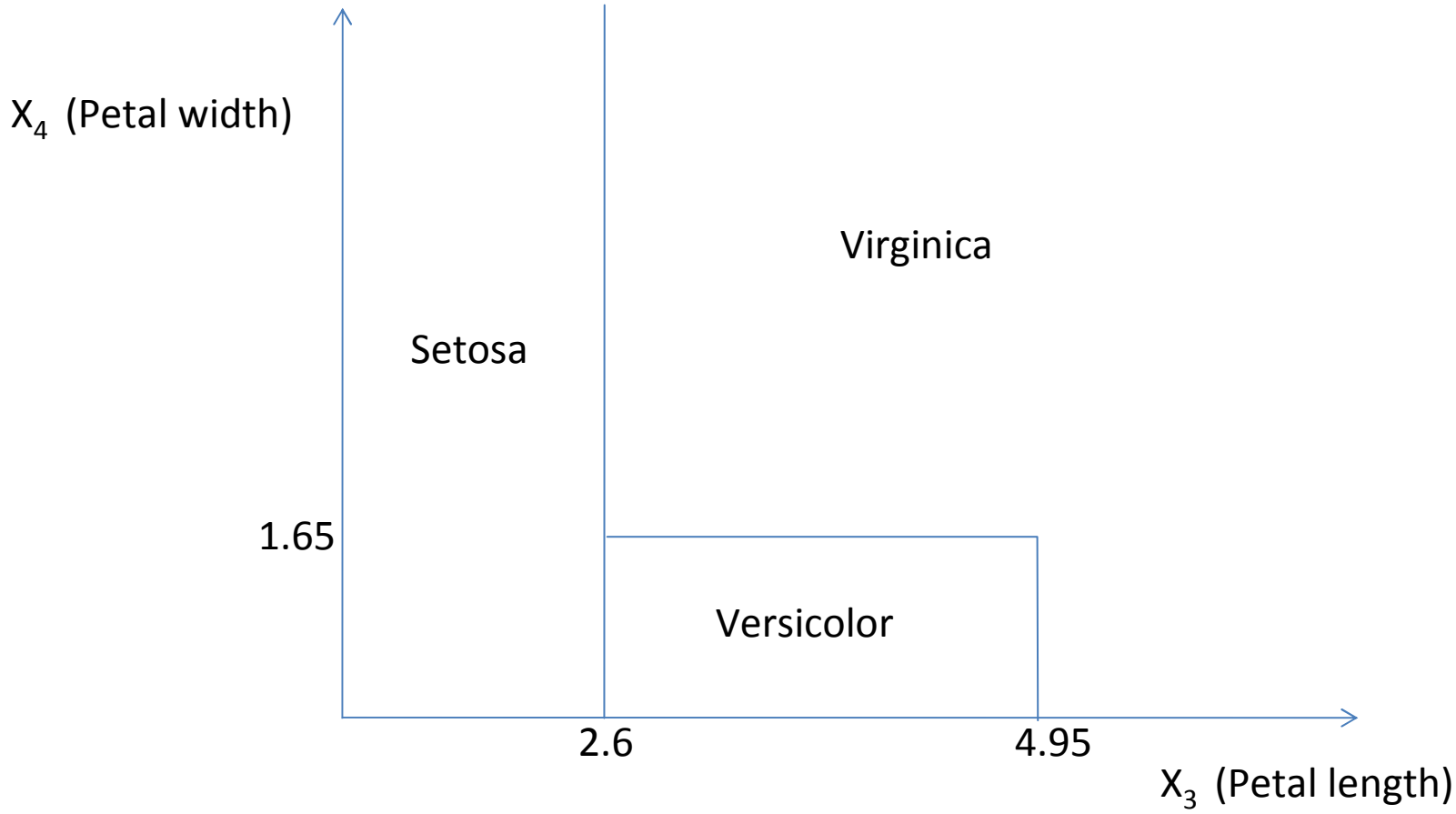
Fig. 4. Partitioning tree for the Iris data of Example 2.

- Used first 25 samples from each category
- Two of the four features x1 and x2 do not appear in the tree → feature selection capability

Sethi and Sarvaraydu, IEEE Trans. PAMI, July 1982

Decision Tree for IRIS data

- 2-D Feature space representation of the decision boundaries



Decision Tree – Hand printed digits

160 7-dimensional patterns from 10 classes; 16 patterns/class. Independent test set of 40 samples

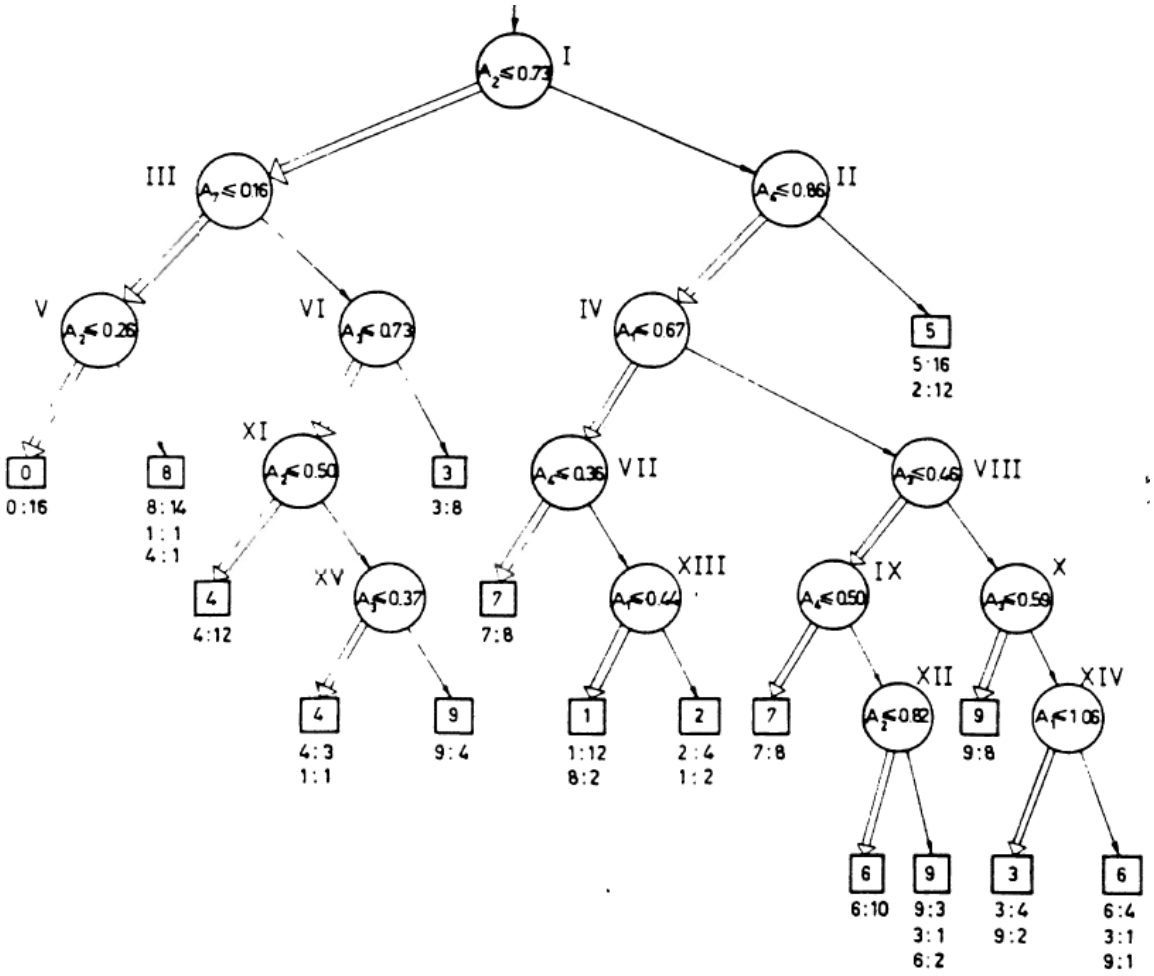


Fig. 6. Partitioning tree for handprinted numeral recognition. Numbers in the square boxes represent the terminal labeling. Numbers below the square boxes represent the label and number of design samples falling at that terminal node.

Properties of a Metric

- A metric D(.,.) is merely a function that gives a generalized scalar distance between two argument patterns
- A metric must have four properties: For all vectors **a**, **b**, and **c**, the properties are:
  - Non-negativity: D(**a**, **b**) >= 0
  - reflexivity: D(**a**, **b**) = 0 if and only if **a** = **b**
  - symmetry: D(**a**, **b**) = D(**b**, **a**)
  - triangle inequality: D(**a**, **b**) + D(**b**, **c**) >= D(**a**, **c**)
- It is easy to verify that the Euclidean formula for distance in d dimensions possesses the properties of metric

$$D(\mathbf{a},\mathbf{b}) = \left(\sum_{k=1}^{d} (a_k - b_k)^2 \right)^{1/2}$$

Scaling the Data

- Although one can always compute the Euclidean distance between two vectors, the results may or may not be meaningful
- If the space is transformed by multiplying each coordinate by an arbitrary constant, the Euclidean distance in the transformed space is different from original distance relationship; such scale change

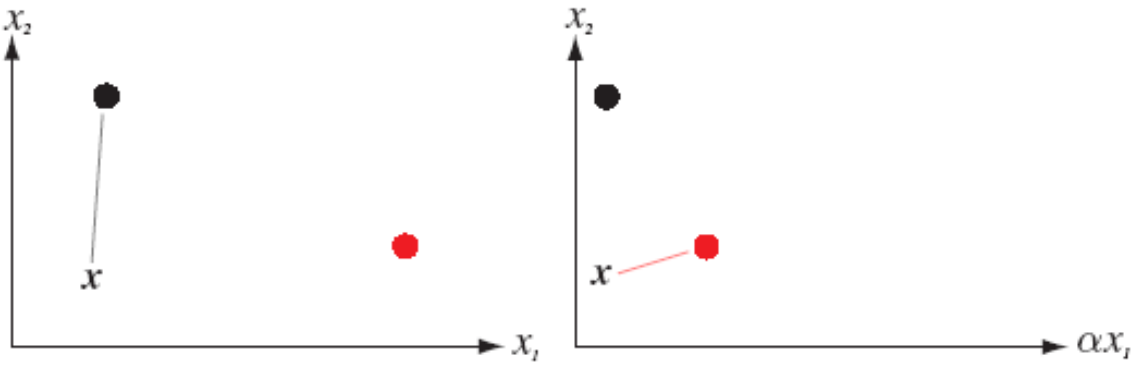


FIGURE 4.18. Scaling the coordinates of a feature space can change the distance relationships computed by the Euclidean metric. Here we see how such scaling can change the behavior of a nearest-neighbor classifer. Consider the test point **x** and its nearest neighbor. In the original space (left), the black prototype is closest. In the figure at the right, the x_1 axis has been rescaled by a factor 1/3; now the nearest prototype is the red one. If there is a large disparity in the ranges of the full data in each dimension, a common procedure is to rescale all the data to equalize such ranges, and this is equivalent to changing the metric in the original space. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

General Class of Metrics

- Minkowski metric

$$L_k(\mathbf{a},\mathbf{b}) = \left(\sum_{i=1}^{d} | a_i - b_i |^k \right)^{1/k}$$

- Manhattan distance

$$L_1(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{d} | a_i - b_i |$$

Example 2: Surrogate Splits and Missing Attributes

Consider the creation of a monothetic tree using an entropy impurity and the following ten training points. Since the tree will be used to classify test patterns with missing features, we will give each node surrogate splits.

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

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