

Applications of Artificial Intelligence Methods in Environmental Science

Edited by S. E. Haupt, C. Marzban, and A. Pasini

Chapter 4

Decision Trees

G. R. Dattatreya

Department of Computer Science

University of Texas at Dallas

Richardson Texas 75083-0688

Phone: 972-883-2189

Fax: 972-883-2349

Email: datta@utdallas.edu

Contents

4	Decision Trees	3
4.1	Introduction	7
4.2	Decision-making and pattern classification	9
4.2.1	Statistical pattern classification	9
4.2.2	Use of logical inter-relationships	10
4.3	Decision regions	12
4.4	Discriminant functions	14
4.5	Graphs and trees	15
4.6	Decision tree examples	21
4.7	Decision tree design from decision boundaries	25
4.7.1	The feature space	26
4.7.2	Problem formulation	26
4.7.3	Optimization criterion	26
4.7.4	Solution approach	27
4.7.5	Efficient implementation of binary trees from decision regions	29
4.8	Decision tree design from labeled training samples	29
4.8.1	Linear discriminant functions	30
4.9	Unsupervised decision tree design	33
4.9.1	Bottom-up method	35
4.9.2	Top-down method	36
4.9.3	Interactive approaches	37
4.10	Further reading	37
4.11	Conclusion	39

References	40
Index	66

4.1 Introduction

Statistical decision-making is widely used in experimental earth sciences. The topic plays an even more important role in Environmental Sciences due to the time varying nature of a system under observation and the possible necessity to take corrective actions. A set of possible corrective actions is usually available in a decision-making situation. Such a set is also known as the set of decisions. A number of observations of physical attributes (or variables) would also be potentially available. It is desirable for the corrective action selected in a situation to minimize the damage or cost, or maximize the benefit. Considering that a cost is a negative benefit, scientists and practitioners develop a composite single criterion that should be minimized, for a given decision-making problem. A best decision, one that minimizes the composite cost criterion, is also known as an optimal decision.

The process of obtaining or collecting the values that the physical variables take in an event is also known by other names such as extracting features (or feature variables) and making measurements of the variables. The variables are also called by other names such as features, feature variables, and measurements. Among the many possible physical variables that might influence the decision, collecting some of them may pose challenges. There may be a cost, risk, or some other penalty associated with the process of collecting some of these variables. In some other cases, the time delay in obtaining the measurements may also add to the cost of decision-making. This may take the form of certain losses because a corrective action could not be implemented earlier due to the time delay in the measurement process. These costs should be included in the overall cost criterion. Therefore, the process of decision-making may also involve deciding whether or not to collect some of the measurements.

A mathematical space of the entire set of variations in the variables and their costs can be imagined in such a decision-making situation. Associated with every combination of values of variables, the overall cost of assigning a decision, including any measurement costs, can be imagined. Following this, the optimal decision for each combination of feature measurements can also be imagined. Such a mathematical representation of inter-relationships between all the variables involved is known as a “model.” The variables of feature measurements, the costs, the parameters used for combining the costs to a single criterion, and every other mathematical quantity and function used in the representation of inter-relationships are relevant aspects of the model.

Unfortunately, a precise mathematical space of costs of decisions and hence the map of optimal

decisions is merely hypothetical or ideal. Usually, there are uncertainties in exactly quantifying the mathematical inter-relationships required for such a construction. Some of the relationships may be deterministic. Some others may be statistical. There may be limited *a priori* knowledge to precisely quantify the statistical relationships. Finally, even with an imagined perfect mathematical space of inter-relationships, their representation and evaluation of optimal decisions may require formidable amounts of computer memory space and computations. Artificial Intelligence approaches for modeling and decision-making are helpful in many such situations. They are useful in reducing the complexity of representations. In some cases, they dynamically develop the representations of the model through the course of decision-making, instead of attempting to build a possibly unmanageably large static representation. They are also useful for approximate representation of imprecise relationships. Finally, they are useful in reducing the complexity of the computation required to evaluate optimal decisions, through the use of heuristics to evaluate nearly optimal decisions. Decision Trees is one of the Artificial Intelligence approaches and is the subject of the present chapter.

The purpose of working with a model is to help us in decision-making. In qualifying models, clarifications between various adjectives such as exact, precise, complete, and statistical are in order. A complete model accounts for all possible inter-relationships. A precise model specifies the inter-relationships without ambiguity. For example, the statement “high ambient Ozone levels cause considerable discomfort for people with respiratory sensitivity” specifies a relationship. But it is not mathematically precise due to the subjectivity of words such as “high,” and “considerable.” A specification may be precise, but only approximate, as opposed to being exact. Some relationships may be statistical, as opposed to being deterministic. Statistical relationships with complete, precise, and correct specifications are as good as similarly specified deterministic relationships in the following sense. In the case of statistical relationships, the statistical mean or the expected overall cost of the decision is minimized to obtain an optimal decision, as opposed to minimizing the exact overall cost.

Clearly, from the above arguments, a complete and exact model cannot usually be constructed in many Environmental Sciences applications. Even if a practitioner is willing to accept approximate but completely specified models, they may not be available in a timely fashion for many applications. Models may be only partially specified or the parameters of the model may not be accurate if observations are made and decisions are required to be made with limited resources;

time is one of the resources. Meteorology is such an application. Meteorological phenomena are observable weather events. These are influenced by temperature, pressure, and water vapor, among others. These physical variables interact with one another. Their variations over the three dimensional space and time are also physical attributes and these contribute significantly to the occurrence of a meteorological event. Furthermore, the above physical attributes are very useful in predicting future meteorological events, within reasonable time frames. Although tremendous advances in research have increased the accuracy of forecasting, there is always room for improvement. Determination of ranges of various physical attributes and their combinations for accurate identifications of various important events is extremely useful. There is virtually no limit to the number of various transformations of variables and combinations of transformations that may potentially increase the accuracy of such classification. Moreover, different transformations over different ranges of attributes (and their combinations) may be required. Therefore, research on this topic is open-ended.

The present chapter studies a class of approaches for classification (decision-making) algorithms. These methods integrate models based on partial information about logical inter-relationships with statistical representations. The overall objective is to develop guided decision-making algorithms called Decision Trees. The approach is applicable in many experimental areas of earth sciences for reasons mentioned above. The final algorithm in this class is also known by other names such as *Multistage Classification* and *Hierarchical Classification*.

4.2 Decision-making and pattern classification

4.2.1 Statistical pattern classification

In its simplest form, pattern classification (Duda, *et al.*, 2001) requires that a given data vector \mathbf{x} be assigned to one of several known categories, $\omega_1, \dots, \omega_k$. The data vector variable \mathbf{x} is composed of m measurements so that

$$\mathbf{x} = [x(1), x(2), \dots, x(m)]. \quad (4.1)$$

As mentioned earlier, each measurement is also called a feature, whose value is influenced by the pattern class corresponding to the data vector \mathbf{x} . Each feature may be cardinal, ordinal, or nominal valued. A cardinal valued variable takes values over continuous segments of a real line.

An ordinal valued variable, over a countable set of ordered values, such as integers. A nominal valued variable takes values from a finite set in which the values in the set have no natural order. An example of a nominal variable is the presence or absence of a phenomenon, such as the presence or absence of a particular pollutant in a material sample.

In many completely designed classification applications, we know the *a priori* class probabilities, P_i respectively, for each class ω_i . We also know the *class conditional probability density functions*, $p(\mathbf{x}|\omega_i)$, for each class ω_i and for all vector points $\{\mathbf{x}\}$ in the observation space. We then maximize the *a posteriori* probability of the class to be assigned to the observed data. That is, assign class ω_i to the observed data vector \mathbf{x} if the *a posteriori* probability of class ω_i ,

$$P[\omega_i|\mathbf{x}] \geq P[\omega_j|\mathbf{x}], \text{ for every } j \in \{1, 2, \dots, k\}. \quad (4.2)$$

Using the Bayes theorem in probability theory, an *a posteriori* class probability can be expressed as a function of its *a priori* class probability and class conditional density functions, as follows.

$$P[\omega_j|\mathbf{x}] = \frac{p(\mathbf{x}|\omega_j)P_j}{\sum_{l=1}^k p(\mathbf{x}|\omega_l)P_l}, \text{ for every } j \in \{1, 2, \dots, k\}. \quad (4.3)$$

The denominator in the right hand side of the above equation is independent of j . Therefore, the decision rule in equation (4.2) simplifies to maximizing the numerator of the right hand side of equation (4.3) over all j . That is, assign class ω_i to the observed data vector \mathbf{x} if

$$p(\mathbf{x}|\omega_i)P_k \geq p(\mathbf{x}|\omega_j)P_j, \text{ for every } j \in \{1, 2, \dots, k\}. \quad (4.4)$$

The above approach to decision-making depends on the ability to statistically represent all the variations of the data, over the entire multivariate space of all the measurements.

4.2.2 Use of logical inter-relationships

Purely statistical approaches constitute one extreme way to model data for decision-making. At the other extreme is a set of pure logical inter-relationships. Such logical inter-relationships may be constructed through various types of data analyses other than with pure statistical models. These inter-relationships may be completely deterministic or approximated to be deterministic. In practice, a combination of logical inter-relationships and statistical data analysis is commonly employed. Logical inter-relationships can be considered to be *perfect* if its use is guaranteed to be

error-free in every instance that it is used to make decisions. The same information can be considered to be *complete* if its application is guaranteed to lead to a final decision (as opposed to a partial decision) for every combination of measurements. An availability of such complete and perfect logical inter-relationships obviates statistical approaches. Such an ideal case is seldom found in applications. In real life applications, we usually have only imperfect and incomplete information about the model. This is also usually accompanied by past cases of data and any decisions that may have been made using such data. Such data are called training pattern samples. Practical decision-making algorithms are best designed with the help of both available information about logical inter-relationships and statistical training data. The following illustrates this approach with the help of simple fictitious example.

A patient visits his family-practice physician with flu-like symptoms. The cause may be upper respiratory allergies or a viral infection. Although there is no cure for viral infection, a secondary bacterial infection may follow in either case, under some conditions. Patients with a history of such risks should be treated differently in comparison with those having no such history. A possible model of logical inter-relationships is shown in Figure 4.1.

<Insert Figure 4.1>

The physician examines if the patient has fever. For three possible levels (or grades) of fever, the courses of actions are different. For low fever, the doctor examines the patient's records to check if he has risk factors. If the patient is at risk, the doctor prescribes medicines to relieve symptoms of Cold (indicated by a decision C in the figure). If the patient's fever is high, the doctor prescribes both antibiotics and Cold medicines (indicated by A & C). Under other conditions, the physician does not prescribe any medication. Of course, if the symptoms worsen within a day or two, the patient will return to the doctor's office. This is indicated by the decision "Wait." This is an example of a model of logical inter-relationships. This example assumes that the physician has a list of risk factors and there is no ambiguity about risk factors. However, this model is still imperfect since there is no specification of how to distinguish between low and high grades of fever. The final decision-making algorithm requires a threshold of body temperature to decide between low and high fever. A good threshold can be determined with the help of numerous past cases of how patients' conditions changed starting from different observed temperatures. The threshold determination may also be influenced by how past patients with different temperatures responded to different treatments. The observations about the past patients constitute statistical

training data. In the above example, the physician arrives at a final decision through a sequence of partial decisions. At each stage, some information about the case (the patient in the above example) is examined and further actions are contemplated. At each stage, one of the possible actions is selected. Such an approach to decision-making is called the decision tree approach. The corresponding pictorial representation of the decision-making scheme is called the decision tree.

In a general decision-making scheme (including in decision trees), there is an optimal decision associated with every combination of feature measurements. Thus, the mathematical space of measurements is divided into regions of different optimal decisions. These regions are called decision regions. The boundaries between adjacent regions of different decisions are called decision boundaries.

4.3 Decision regions

Decision-making algorithms induce decision boundaries and decision regions in the data space $\{\mathbf{x}\}$, as noted in the above section. That is, the multidimensional data space is divided into many regions and a class label is assigned to each region. There may be multiple disjoint regions constituting a single class. The following is a hypothetical example. Figure 4.2 shows an example of decision regions with 4 classes and two measurements, x and y .

<Insert Figure 4.2>

In this example, the x axis segment extends from 0 to 24 units. The y axis segment extends from 0 to 16. The decision region for class 1 is inside an ellipse with its major axis parallel to the x axis. The ellipse is centered at (4, 12). Its major axis measures 6 and its minor axis, 4. The decision region for class 4 is inside a circle centered at (12, 8) with radius 4. The decision region for class 3 is above and to the right side of the inclined line segment passing through the two points (12, 16) and (24, 8). The rest of the area corresponds to class 2.

If the data vector also has ordinal and/or nominal features, the space of the feature measurements will be composed of continuous as well as discrete variables. In practice, the *a priori* class probabilities and the class conditional probability density functions of the observations are not accurately known. Design of pattern classifiers are then based on finite sample training data sets. There are several approaches to designing classifiers starting from training data sets. Typically, we may have access to several data vectors from each class. A labeled sample set of training data

consists of the set of classes is $\{\omega_1, \dots, \omega_k\}$ with k classes, and n_i data vectors from class ω_i where

$$\mathbf{x}_{ij}, \quad j = 1, \dots, n_i \text{ and } i = 1, \dots, k \quad (4.5)$$

is the j -th data vector from class ω_i . In some applications, the relative numbers of training data samples may adequately represent the *a priori* class probabilities. That is,

$$\frac{n_i}{\sum_{j=1}^k n_j} \quad (4.6)$$

may be a good estimate for P_i . In other applications, there may be attempts to supply a training data set with a maximum possible size. In such a case, the relative numbers of samples from different classes may not approximate the *a priori* class probabilities. The relative proportions in field, that is in the real application, may be known or estimated by other means. For example, during a particular season, we may know that the weather for the midday (without any additional information) has the following probabilities.

$$P[\text{sunny}] = 0.75, \quad (4.7)$$

$$P[\text{rain}] = 0.15, \quad (4.8)$$

$$P[\text{cloudy}] = 0.07, \text{ and} \quad (4.9)$$

$$P[\text{snow}] = 0.03. \quad (4.10)$$

In yet other applications, it may be reasonable to assume equal *a priori* probabilities for all classes. The data sets for individual pattern classes may be used to estimate the parameters of class conditional probability functions based on a known or assumed family of probability density function. An example of a family of probability density functions is the set of Gaussian probability density functions representing a single feature measurement. Different values for the mean and variance correspond to different members of this family of probability density functions. The mean and variance of the density function are known as the parameters of the probability density function. Other families of probability density functions may have parameters other than (or in addition to) the mean and variance. For example, a uniform probability distribution of a continuous random variable is conveniently represented by the extreme (the lowest and the highest) values that the random variable can take. In this case, these extreme values are the parameters of the probability density function. The probability density function of a random variable is the derivative of the probability distribution function. The latter is also called the cumulative distribution function. A

completely specified statistical model for pattern classification or decision-making uses a probability density (or distribution) function for the feature measurements for objects (or events) from each class of patterns. The *a priori* class probabilities of the occurrence of the classes in nature are also used. All the parameter values required for decision-making are required for a complete specification. This approach of employing a family of probability density functions and using estimated values for these parameters to design a pattern classifier is known as the parametric approach. The final classification algorithm assigns a class label for every data point in the space of feature measurement, following a maximum probability of correct classification rule of equation (4.4) developed in Section 4.2. The resulting map of the class assignment divides the feature space into decision regions.

4.4 Discriminant functions

There are alternatives to the above described parametric approach. After all, the final classifier is required to specify decision regions in the data space of the available features. Of course, once the decision regions are specified, it is desirable to have an efficient algorithm to sift through the decision regions to arrive at the class label corresponding to an observed data vector. There are approaches to use parametric forms for the decision boundaries and optimize the parameters to minimize a function of the errors, based on training sets. This approach determines the parameters of such “discriminant functions.” In simple cases, a discriminant function is associated with each class label. The classification algorithm evaluates these discriminant functions, for the given data, for all the classes and picks the class label that maximizes the discriminant function. In this sense, the *a posteriori* class probabilities in equation (4.2) and the joint probability density function in equation (4.4) are discriminant functions.

A classifier that uses linear discriminant functions (Duda, *et al.*, 2001) for classification is known as a linear classifier. As a simple example, consider the decision regions in Figure 4.2. If the classification problem is to decide between class 3 and all other classes, there would be only two decision regions, on either side of the straight line separating class 3 and all other classes. The resulting classifier is a linear classifier. Linear classifiers are very popular for several reasons. The resulting decision regions are easy to visualize geometrically, in the data space. The determination of optimal parameters of linear discriminant functions are often simpler than in the nonlinear case.

In some cases, original data vectors are subjected to nonlinear transformations and then optimal linear discriminant functions are developed. In the original case of full knowledge of *a priori* class probabilities and class conditional probability density functions, it can be shown that there is nothing gained by using any transformation of existing feature variables, in terms of performance of the classifier, for example in the probability of correct classification. However, in the case of design from a finite sample training data set, some nonlinear transformations may lead to a simple classification algorithm and with better accuracy of classification. A very simple generalization of linear classifiers uses *piecewise linear functions*. This is very easily illustrated with the help of another example below.

Figure 4.3 depicts decision regions for an application.

<Insert Figure 4.3>

The overall rectangular region of measurements is 24 units wide and 16 high. The intercept values indicated for the line segments completely define each straight line. Every closed region has an assigned class label indicated in the figure. The broken portions of some line segments in the figure do not separate class labels. They are drawn only to clearly specify the lines. In a field application of the above pattern classifier, an observation (data) vector (x, y) would be given and the task of the classification scheme is to determine the final class label corresponding to the point (x, y) in the decision regions. A particularly convenient approach to implement the classification algorithm for the decision regions of Figure 4.3 is to examine which side of the different straight line segments that a given point (x, y) falls. This requires a few steps and the corresponding scheme is appropriately known as a multi-stage decision-making algorithm. An elegant and convenient way to represent such a classification scheme is with the help of a decision tree. Trees are a subclass of mathematical structures (objects, entities, etc.) called graphs. In many cases, such decision trees are extracted from graphs. Therefore, an elementary study of definitions and properties of graphs and trees is useful. The next section introduces principles of graphs and trees.

4.5 Graphs and trees

In a graph, we have a set of vertices or nodes commonly denoted by set V . Distinct nodes are identified by numbers, letters, names, etc. Each node may represent a physical or an abstract object. An edge in a graph is an entity different from a node. An edge connects two nodes. The set

of edges is commonly denoted by E . Edges are also known by other names such as arcs, branches, and links. At most one edge may be present from a vertex to another. An edge may also connect a vertex to the same vertex. The following are formal definitions.

A graph G is a set $\{V, E\}$ where V is a set of vertices and E is a set of edges defined over V . Vertices and Edges are defined as follows. A vertex is a physical or an abstract entity. An edge is a pair of vertices. A set of vertices induces a set of all possible edges. Edges can also be identified by numbers, letters, names, etc. If the same type of identifiers are used for both vertices and edges, it is important to be able to distinguish between a node identifier and an edge identifier. In the literature, it is not too uncommon to find graphs with numbers used for identifying nodes as well as edges. In such a case, we need to avoid confusion by explicitly referring to them as “vertex k ” and “edge k .” Pictorially, a vertex is represented by a heavy dot, a small circle, an ellipse, or a square, etc. An edge is represented by a straight or curved line drawn from one vertex to another. We use the following descriptions with obvious meanings. An edge connects its two vertices. A vertex touches its edges. An edge originates at a vertex and terminates at a vertex, etc.

Figure 4.4 depicts a graph.

<Insert Figure 4.4>

Its vertex set is $V = \{A, B, C, D, E\}$ and its edge set is

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}. \quad (4.11)$$

We find that $e_1 = (A, C)$, $e_2 = (C, E)$, $e_3 = (C, D)$, $e_4 = (D, E)$, $e_5 = (B, D)$, $e_6 = (A, B)$, $e_7 = (A, D)$, and $e_8 = (B, E)$. In some graphs, a direction may be associated with every edge. If an edge is not directed (also called undirected), it is considered to connect from each of its two vertices to the other. It is convenient to have all edges of a graph as directed, or all of them as undirected. This is not restrictive, since, an undirected edge can always be represented by two directed edges in opposite directions. Therefore, each pair of vertices identifying a corresponding edge in Figure 4.4 is an unordered pair.

If all the edges of a graph are unordered pairs of vertices, the graph is said to be undirected. If all the edges of a graph are ordered pairs of vertices, the graph is said to be directed. If (R, S) is an ordered pair representing a directed edge, by convention, the edge originates at the vertex R . The number of edges originating and/or terminating at a vertex is often an important characteristic of the vertex. These are defined separately for undirected and directed graphs. In an undirected graph, the degree of a node is the number of edges that the node touches. In a directed graph,

the number of edges terminating at a node is known as the in-degree of the node. The number of edges originating from a node is known as the out-degree of the node.

Figure 4.5 shows a directed graph.

<Insert Figure 4.5>

It is constructed by making the following modifications to the undirected graph of Figure 4.4. Specific directions have been imposed for edges $e_1, e_2, e_3, e_4, e_5,$ and e_8 of the original undirected graph of Figure 4.4. Each of the other two edges e_6 and e_7 of the original undirected graph are duplicated to create corresponding edges in opposite directions. Therefore, the vertex set for this directed graph is the same as in the above undirected graph example, given by $V = \{A, B, C, D, E\}$. The edge set E has the following ordered pairs of vertices for edges. Edge $e_1 = (A, C), e_2 = (E, C), e_3 = (C, D), e_4 = (D, E), e_5 = (D, B), e_6 = (B, A), e_7 = (A, D), e_8 = (B, E), e_9 = (D, A),$ and $e_{10} = (A, B)$. The in-degree of vertex A is 2 and its out-degree is 3. The in-degree of vertex C is 2 and its out-degree is 1. Clearly, an n vertex directed graph can have a maximum of n^2 edges. An n vertex undirected graph can have a maximum of $\frac{n(n+1)}{2}$ edges. These numbers include possible edges from a node to itself. If we exclude possible edges from a node to itself, the number of possible edges in an n -vertex directed graph is $n(n-1)$; for an undirected graph, it is $\frac{n(n-1)}{2}$.

A path, in a graph, is a sequence of $k + 1$ vertices $\{v_0, v_1, \dots, v_k\}$ and a corresponding sequence of k edges

$$\{(v_0, v_1), (v_1, v_2), \dots, (v_{k-2}, v_{k-1}), (v_{k-1}, v_k)\}, \quad (4.12)$$

provided such edges exist in the graph to create the path $\{v_0, v_1, \dots, v_k\}$. Note that v_i is a variable vertex in the above definition, in the sense that v_i is some vertex of the graph under consideration. The number of edges in a path is known as the number of hops, length of the path, etc. The first vertex in the sequence of vertices forming a path is known by different names such as the initial, originating, beginning, or the starting vertex of the path. Similarly, the last vertex in a path is known by different names such as final, terminating, ending vertex. We use descriptions such as “traversing the path,” “traversing the graph through the path,” etc. Some simple examples help in understanding the definition of a path. In the undirected graph of Figure 4.4, the sequence of vertices $\{A, D, C, E\}$ is a path of length 3. The sequence of vertices $\{A, B, C\}$ is *not* a path since the graph has no edge between the vertices B and C . In Figure 4.4, again, the sequence of vertices $\{C, A, B, D, A\}$ is a path of length 4 from C to A , even though the vertex A is traversed

twice. In many applications, such paths are undesirable and paths without repeating vertices are desired. In the directed graph of Figure 4.5, the sequence of vertices $\{A, C, D, B\}$ is a path of length 3 from node A to node B . The sequence of nodes $\{A, B\}$ is also a path, of length 1, from node A to node B . However, the sequence of nodes $\{A, C, E\}$ is *not* a path since there is no directed edge (C, E) in this directed graph.

A path originating from a node and terminating at the same node is called a cycle. For example, $\{A, B, D, E, C, A\}$ in the undirected graph of Figure 4.4 is a cycle of length 5. The path $\{C, D, E, C\}$ in the same figure is also a cycle, of length 3. In the directed graph of Figure 4.5, each of the paths $\{A, B, A\}$ and $\{B, E, C, D, A, B\}$ is a cycle. An undirected graph is said to be connected if there is a path from every vertex to every other vertex. The undirected graph of Figure 4.4 is a connected graph. A connected undirected graph without any cycle is called a tree.

We usually form a tree from a connected undirected graph by removing some edges to satisfy the requirements. Since there are no cycles in a tree, there must be one and only one path from any vertex to any other vertex. The reason such a graph is called a tree is that we can identify any node and branch out to all other nodes, just as a real tree starts from the ground and branches off to its ends. The following is a fundamental result in Graph theory.

An $n > 0$ vertex tree has exactly $n - 1$ edges. The statement is proved by induction as follows. A tree with one vertex has no other vertex to connect to and hence has no edges. A tree with two vertices must have exactly one edge connecting the two nodes. Therefore, the theorem is valid for trees with $n \leq 2$ nodes. Let every tree with a particular value of $k > 2$ nodes have exactly $k - 1$ edges. Now, add a node to such a tree to make it a $k + 1$ node graph. In order to make the new graph connected, we must add an edge between the newly added $(k + 1)$ -th node and any other node, increasing the number of edges to k . When we add this new node and a corresponding new edge, we will have a path from every node to every other node, so that the graph is connected. If we now add an additional $(k + 1)$ -th edge between any two nodes, say, between nodes A and B , we create a *second* path between nodes A and B , since there was already an existing path. The two paths create a cycle and hence adding the $(k + 1)$ -th edge destroys the tree. Similarly, any additional number of edges over and above k edges will create one or more cycles. This shows that if every k node tree has $(k - 1)$ edges, then every $(k + 1)$ node tree has k edges. Since the property is known to be true for $k = 1$ and 2, it follows that every $n > 0$ vertex tree has exactly $n - 1$ edges.

In a rooted tree, any particular node of the tree is identified as the root node. If a node of a

rooted tree is not its root and if its degree is one, we call it a leaf node or a terminal node. Figure 4.6 shows a rooted tree. It is a sub-graph of the undirected graph in Figure 4.4. Node B is identified as the root node. Nodes C , D , and E are leaf nodes.

<Insert Figure 4.6>

Starting from the root of a tree, we can reach all the nodes by the following informal traversal algorithm. Start from the root node. Traverse along any path without reversing the direction of traversal. Color or mark every traversed edge. Also mark the direction in which every edge is traversed. When a leaf node is reached, start from any non-leaf node that has been reached in an earlier traversal and that touches an unmarked edge. Continue traversing until another leaf node is reached. Continue this procedure until all edges are marked. Notice that once we start traversing from a node, we never reach a node that has already been visited, since there are no cycles in the tree. The second observation is that every edge is traversed only once. As a consequence, the above procedure assigns directions to every edge in the tree. Every direction moves away from the root node and towards leaf nodes. The third observation is that given a rooted tree, the directions associated with every edge along every path towards a leaf node is unique.

A rooted tree in which obvious directions are assigned for every edge to create a (directed) path from the root node to every leaf node is a rooted and directed tree. Along every path to a leaf node, a non-root node is called the child of the node preceding the node in the path under consideration. All the child-nodes of a node are called siblings. If a node B is a child node of the node A , then A is called the parent node of node B . Clearly, the root node has no parent. Every leaf node has no child node. A consequence of the property that directions of edges in a rooted and directed tree are unique is that we may elect to not explicitly show the directions of edges in figures. In this sense, a rooted tree and the corresponding rooted and directed tree are one and the same. In the rooted tree of Figure 4.6, nodes A , D , and E are siblings and all these are the child-nodes of the node B . It is convenient to redraw a rooted tree with the root node at the top and all its child nodes below the root node. Similarly, all succeeding child nodes of all newly drawn nodes are drawn below the corresponding parent nodes. Such modifications to a graph do not change a graph, since the definition of the graph (that is, the set of vertices, the set of edges over the set of vertices, directions of edges, if any, and the root node), are not changed. Figure 4.7 shows a such a redrawn tree of the one in Figure 4.6.

<Insert Figure 4.7>

The height of a rooted tree is the length of the longest path from the root node to any leaf node in the tree. The height of the rooted tree in Figure 4.7 is two.

A directed graph that has no cycles is called a directed acyclic graph (DAG). Unlike in a tree formed from an undirected graph, it is not necessary for a DAG to have a node from which there are paths to all other nodes. Figure 4.8 is a simple example of a DAG illustrating this peculiarity.

<Insert Figure 4.8>

Multiple paths between a pair of nodes are allowed in a DAG. For example, the DAG in Figure 4.8 has two paths from node *A* to node *E*. In some decision tree design algorithms, decision trees are extracted as subgraphs of DAGs. If in a rooted tree, every non-leaf node has exactly two child nodes, the tree is called a binary tree. We are now ready to define a decision tree.

Consider a rooted (and directed) tree. Let every non-leaf node examine a data vector. The data vector is specific to the non-leaf node under consideration. The data vector examined at a node can be a specific transformation of the overall data vector in the application system. Let the set of possible outcomes of the examined data vector be partitioned into as many proper subsets as the number of child nodes of the non-leaf node under consideration. Let there be a simple test at every non-leaf node to determine which child node to traverse to, at every non-leaf node. Let every leaf-node be assigned a decision from a set of decisions. Such a tree is called a decision tree.

The motivation for such a definition is the following. If the data vector used at the root node is specified, the partial decision of which of its child nodes to traverse to can be determined. If the data vector at that child node is available, the next level child node to traverse to can be determined. This process can continue with partial decisions until a leaf node is reached. The decision tree is merely an approach or an algorithm to determine the final decision for every possible overall data vector point. We say that every non-leaf node “tests” a given data vector and decides the child node to which the decision-making algorithm traverses. Note that two or more different leaf nodes may be identified with the same decision. Decisions can be identified by numbers, names, subscripted symbols, etc. A decision tree extracted from a DAG has one node from which there are paths to all terminal nodes of the DAG. Such a DAG is identified by the following definition.

If a DAG has one node from which there is a path to every terminal node of the DAG, the DAG is said to be a rooted and directed acyclic graph (RDAG). Figure 4.9 shows an RDAG extracted from the directed graph of Figure 4.5. Since the edges in an RDAG are directed, the root node is the unique node *D*, in Figure 4.9.

<Insert Figure 4.9>

A binary decision tree is a binary tree as well as a decision tree. Each test in a decision-making node of a binary decision tree is a test with two possible outcomes, “yes” or “no.” In this chapter, we use the convention that the left branch of a decision-making node in a binary decision tree corresponds to the outcome “yes,” and the right branch, to “no.” Figure 4.10 is a binary decision tree; it is obtained by modifying the original (non-binary) decision tree of Figure 4.1.

<Insert Figure 4.10>

The above definitions of the decision tree and the binary decision tree are very general. It is clear that a decision tree simply specifies decision regions in the overall data space. That is, a decision tree is merely a convenient artifice to implement a decision-making algorithm over a completely specified set of decision regions. To be useful, the tests at individual non-leaf nodes must be simple. Therefore, the art of designing decision trees can actually influence the partitioning of the overall data space into decision regions, often in an interactive and iterative approach. We will study design of decision regions and decision trees in later sections; the following section illustrates implementation of decision trees for given decision regions, through examples.

4.6 Decision tree examples

The implementation of a pattern classifier algorithm for a given set of decision regions (for example, the decision regions in Figure 4.2 or Figure 4.3) requires a sequence of steps. Typically, for a given data point, that is, a point (x, y) , we should sequentially determine on which side of various straight or curved lines the point (x, y) falls. If we have a three dimensional data vector, the geometrical figure separating two adjacent regions is a surface. If we have a higher dimensional data vector, the mathematical equation separating two adjacent regions in the data space is referred to as a hypersurface. If we are careful, it may not always be necessary to determine the correct side of *every* line (in our two dimensional data vector cases) found in the overall region of possible measurements. In a sequence of steps, results of currently examined lines can direct us to select which line to examine next, until a final determination of the class label is made. Such algorithms are appropriately called multistage classification schemes. The distinction between a general multistage classifier and a decision tree is somewhat subjective. Generally, in decision

trees, the evaluation of the test function and determination of the next action at every stage is very simple. As an example, a decision to pick one of two options based on a threshold comparison of a single variable at every stage is a very simple multistage classifier. Such a classifier is also a binary decision tree.

Figure 4.11 shows decision regions for another pattern classifier. Notice that every decision boundary is a straight line segment parallel to one or the other of the two coordinate axes. Again, the overall rectangle is 24 units wide and 16 units high, starting from the bottom left corner as the origin. The abscissa and ordinate values of the line segments, as necessary, are given outside the overall rectangle to completely specify the decision regions. The class labels of the regions are the numbers 1, 2, 3, 4, and 5, and these are given inside the decision regions.

<Insert Figure 4.11>

We can implement a decision making algorithm for this classifier in the form of a decision tree. In particular, we consider a binary form of decision trees. Recall that in a binary decision tree, each decision-making node examines a condition, the result of which can be yes or no, until a final decision about the class label is unambiguous. Figure 4.12 shows such a binary tree in which each stage is a comparison of one of the features, x or y to a specified threshold.

<Insert Figure 4.12>

The first stage examines if $y < 7$. As stated earlier, the left branch is for yes (or true), and right branch, for no (or false), a convention used here.

Clearly, we can construct such decision trees in which a test at a particular stage can produce one of many (more than two) possible outcomes. In this case, the result is *not* a binary tree but a more general tree. Figure 4.13 shows such a tree for implementing a decision-making algorithm for the example in Figure 4.11.

<Insert Figure 4.13>

In Figure 4.13, if the test node indicates a condition with only two answers, the left branch is for the “true” answer and the right branch, for the “false” answer to the test. At some stages, multiple branches indicate the conditions under which a particular branch should be followed.

We can also construct decision trees in which a test can involve more than one feature variable. An appealing example is the following. The decision boundaries in Figure 4.3 are all straight line segments. But these straight lines are *not* parallel to either the x or the y axis. However, each straight line is expressed by a simultaneous linear equation involving both the variables x and y .

The two sides of one such equation is specified by the two possible outcomes of the corresponding inequality. Thus we can construct a binary tree for the decision regions of Figure 4.3 in which each test is the examination of an inequality involving a linear combination of both the variables x and y . Constructing such a tree for the decision regions of Figure 4.3 is a comprehensive exercise and is suggested for the reader.

The decision boundaries are not required to be straight line segments to implement the decision making algorithm as a decision tree. Of course, if tests at individual stages of a decision tree are very complicated, the use of the tree approach to decision-making is questionable. Let us implement a decision tree for classification over the decision regions of the example in Figure 4.2. The basic equation for an ellipse whose center is the origin of the coordinate plane and whose major and minor axes are respectively collinear with the coordinate axes x and y is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad (4.13)$$

where a is the intercept on the positive x axis and b , on the positive y axis. The major axis, the longest distance between two extremities of the ellipse and passing through its origin is $2a$. The minor axis, the shortest distance, is $2b$. Shifting the center of the ellipse and substituting for the values of the semi-major and semi-minor axes for our example, we have the equation for the ellipse as

$$\frac{(x - 4)^2}{9} + \frac{(y - 12)^2}{4} = 1. \quad (4.14)$$

Simplifying, we obtain the equation for our ellipse as

$$4x^2 - 9y^2 - 32x - 216y + 1324 = 0. \quad (4.15)$$

The region corresponding to the inside of the ellipse is class label 1 and it is the set of all (x, y) following the inequality

$$u = 4x^2 - 9y^2 - 32x - 216y + 1324 < 0. \quad (4.16)$$

In the above inequality (4.16), the variable u is defined to be a composite feature variable, one that is derived as a transformation from the original feature measurements of x and y .

The equation for a circle of radius r and centered at the origin is

$$x^2 + y^2 = r^2. \quad (4.17)$$

Translating to the center of the circle in our example and using the radius of 4 units, we have

$$(x - 12)^2 + (y - 8)^2 = 16 \quad (4.18)$$

which simplifies to

$$x^2 + y^2 - 24x - 16y + 192 = 0. \quad (4.19)$$

The region corresponding to class 4 is inside and it is the set of all (x, y) satisfying the inequality

$$v = x^2 + y^2 - 24x - 16y + 192 < 0. \quad (4.20)$$

As in the earlier case, the variable v is a composite feature variable, derived from the original feature measurements x and y .

If a straight line passes through two points (x_1, y_1) and (x_2, y_2) , its equation is given by

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}. \quad (4.21)$$

The straight line delineating class 3 from other classes in our example lies on the two points (12, 16) and (24, 8). Therefore, its equation is

$$\frac{y - 16}{x - 12} = \frac{8 - 16}{24 - 12}. \quad (4.22)$$

Simplifying, the equation for the straight line is

$$2x + 3y - 72 = 0. \quad (4.23)$$

The region of class 3 is the set of all (x, y) corresponding to the inequality

$$w = 2x + 3y - 72 > 0. \quad (4.24)$$

Again, w is a derived feature variable from the original measurements x and y . With this preparation, the decision tree implementation for classification in this example is simple. Figure 4.14 shows such a tree. The tests at individual stages in the tree are simple threshold comparison; however, the variables which are compared with the thresholds are nonlinear transformations for the ellipse and the circle and an affine transformation for the straight line.

<Insert Figure 4.14>

The above examples demonstrate several characteristics of decision trees, their constructions and manipulations. All these decision trees are constructed with the help of three different examples of decision regions marked and given to us. Thus, these approaches are directly useful if any of the following is satisfied.

1. We know the decision regions.
2. We know the *a priori* class probabilities and the class conditional probability distribution functions of the measurement vector. In this case, the decision regions can be mathematically developed with the help of the maximum *a posteriori* probability decision-making scheme, given by equation (4.4).

In many practical applications, pattern classifiers are required to be designed with the help of partial knowledge about the physical system, some training data with sample sets of known class labels, unclassified data, or some combination thereof. In such cases, we can attempt to design classifiers that can be implemented as efficient decision trees. That is, instead of designing the decision boundaries and then converting them to decision trees (as is the case in earlier examples), the design of decision regions and a corresponding decision tree can proceed hand in hand. But, first, the design of decision trees from a given set of decision regions is discussed in the next section.

4.7 Decision tree design from decision boundaries

The above examples in Section 4.6 illustrate the development of decision trees from given decision regions. In this section, a systematic design methodology and possible optimization approaches are studied for the same problem. The decision boundaries may be given merely as a convenient approach to partition the space of feature measurements. But the actual assignment of a class label (decision) to each region enclosed by the the decision boundaries may not be available. In such a case, assignment of a class label to the decision regions is part of the decision tree design problem. This section develops the concepts of the overall graph from which decision trees can be extracted, an appealing criterion for optimization, and descriptions of the general approach to optimal tree design. A simple but fairly general example of decision boundaries is used to illustrate the concepts and approaches. A detailed presentation appears in Dattatreya and Kanal

(1985), which also includes a discussion on its suitability for problems with a modest number of features, each of which is quantized to only a modest number of levels.

4.7.1 The feature space

The decision boundaries are assumed to be hyperplane segments parallel to the coordinate axes of the m feature variables $x(1), \dots, x(m)$. As demonstrated with examples in Section 4.6, this assumption is not restrictive since nonlinear transformations can be represented by new feature variables. Let the feature variable $x(i)$ be partitioned into n_i segments with $n_i - 1$ threshold values. Name these segments as $0, 1, \dots, n_i - 1$ for convenience. That is if $x(i)$ is in the j -th segment, we say that $x(i) = j$. The entire feature space is partitioned into $\prod_{i=1}^m n_i$ regions. Each region is bounded by hyperplanes parallel to the coordinate axes. Each such region is represented by a point in a lattice mathematically represented by the vector $\mathbf{x} = [x(1), \dots, x(m)]$. Each point can be called a cell in the lattice. A sublattice is obtained by fixing one or more of the $x(i)$ variables to some constants. Sublattices can also be formed by successively cutting and considering one part of the lattice parallel to one or more coordinate axes.

4.7.2 Problem formulation

The pattern classification problem is that a class label from the set of classes $\Omega = \{\omega_1, \dots, \omega_k\}$ be assigned for every cell, with a minimum expected cost of decision-making. In the completely specified probabilistic framework, we will also be given the *a priori* class probabilities P_1, \dots, P_k respectively for the k classes, the class conditional probability values $P[\mathbf{x}|\omega_i]$ for each cell and for each class ω_i , and the elements c_{ij} of the cost matrix of classifying a data vector belonging to class ω_i with a decision of ω_j .

4.7.3 Optimization criterion

Clearly, if the only cost involved in assigning a decision to a data vector (pattern sample) is the classification cost, we should determine the expected costs of assigning a cell to different classes and choose a class label for each cell based on the criterion of minimum expected cost. The expected cost of assigning a cell $\mathbf{x} = [x(1), \dots, x(m)]$ to the class label ω_j is obtained as follows.

Let $S(\omega_j)$ be the random variable cost of assigning class ω_j .

$$E[S(\omega_j)|\mathbf{x}] = \sum_{i=1}^k c_{ij} \frac{P[\mathbf{x}|\omega_i]P_i}{\sum_{l=1}^k P[\mathbf{x}|\omega_l]P_l}. \quad (4.25)$$

However, if obtaining feature measurements involves costs, then for some cases, it may be advantageous to make a class assignment based on less than all the feature measurements. The following applications illustrate this scenario. In medical diagnosis, obtaining some test measurements may be time consuming (risking a worsening condition), costly, invasive, or risky to the body. In meteorology, numerous measurements and numerically evaluated derived features are potentially available. Some of them require fine grain modeling of the atmosphere and are computationally inefficient to calculate all the time. However, certain combinations of other routine measurements may require a more careful consideration of these expensive features, especially for decisions on severe weather patterns. Other applications are in banking and other financial institutions to determine whether or not to increase interest rates, whether or not to change prices of commodities, process control, root cause analysis (Wilson, *et al.*, 1993) of adverse event occurrences in industry, etc. In all these, decisions may normally be made based on commonly available measurements. The overall expected cost of decision-making may be less if additional, expensive measurements are extracted for certain combinations of values taken by the commonly available measurements. Let r_i be the cost of making the measurement to extract the feature $x(i)$. Of course r_i and c_{ij} are in the same physical dimensions so that the total expected cost of assigning a class label to a data vector is the sum of the feature measurement costs and the expected cost of classification, conditioned on the observed measurements. The decision tree design problem is to develop a tree that minimizes the expected overall cost of making the decision on a pattern sample.

4.7.4 Solution approach

The general approach to designing decision trees uses the “Principle of Optimality.” If we already know the optimal decision-making algorithms for some sublattices, the principle of optimality helps us in optimizing over larger sublattices that can be formed by the union of original sublattices. Algorithms that use the principle of optimality are known as “Dynamic Programming” algorithms. The principle of optimality is stated as follows. Let a problem B be a subproblem of problem A . Consider an optimal solution to the problem A . Let the optimal solution to problem A also solve the

subproblem B . In such a case, the solution of the subproblem B used by the optimal solution for the problem A is an *optimal* solution for the subproblem B . Some general conditions are required for the principle of optimality to hold for a class of problems. In our case, $r_i \geq 0$ for $i = 1, \dots, m$ is true. Also, $c_{ij} \geq c_{ii}$, $j = 1, \dots, k$ and $i = 1, \dots, k$. That is, the feature measurements may incur a positive or no cost. But feature measurement costs cannot be negative. A correct classification does not incur more cost than an incorrect classification. These are sufficient conditions for the principle of optimality to hold for the present problem.

To illustrate the solution approach, consider a specific case with three feature measurements x , y , and z . Let x take values 0, 1, or 2 only. Let y take values 0, 1, 2, or 3 only. Let z take values 0 and 1 only. Let the number of classes be 4. Figure 4.15 shows a directed acyclic graph of all possible feature sequences and with all their possible outcomes.

<Insert Figure 4.15>

At the lowest level, every node has the measurements of all the features. Each of the lowest level nodes represents a cell. Edges from the third row of nodes to the fourth row of nodes are not drawn in Figure 4.15 to reduce clutter. Also, all the edges in the graph are directed downwards and the tips of arrows are not drawn in the figure. The minimum expected cost of each node is the sum of the costs of all the features measured so far plus the minimum expected cost of classification. At each of the lowest level of nodes, all the feature measurements (x , y , and z) have already been obtained, incurring corresponding measurement costs. Therefore, there are no other feature measurements to consider at these lowest level nodes and a final class label decision is required to be made. Intermediate nodes represent values of measurements obtained with one or at most two of the features x , y , and z . In Figure 4.15, if a feature is not measured, it is represented by a hyphen. Therefore, $(2, -, z)$ represents a node at which the feature x has been measured and found out to be 2, feature y is not measured, and a decision to measure z is being considered. In the Figure, 4.15, within each node, the measurements are written top-down, for lack of space. At the node $(2, -, z)$, in addition to the possibility of measuring the feature z , the option to assign a minimum classification cost class label based on the available information of $x = 2$ is possible. In order to make an optimal decision, we need to compare such minimum expected cost of classification with the overall expected cost of the alternative decision, that is of recommending making the measurement of the feature z . That is, we should already know the expected overall cost of the optimal rooted and directed acyclic subgraph rooted at the node $(2, -, z)$, in Figure 4.15. The decision tree can be

designed in a bottom-up fashion, starting from assigning class label decisions to the bottom-most nodes in the DAG of Figure 4.15. The principle of optimality plays an important role here. As a consequence of the principle, we know that if the node $(2, -, z)$ eventually becomes a part of the optimal rooted and directed acyclic graph (RDAG), then the sub-graph below the node $(2, -, z)$ must be an optimal subgraph for the intermediate problem of dealing with the partial information $(2, -, z)$. We can separate the leaf nodes of the optimal RDAG so that the final result would be a decision tree. A possible RDAG as an optimal solution extracted from the Figure 4.15 is drawn in Figure 4.16. In this figure, every final decision of a class label is depicted by a separate node so that the figure is a decision tree.

<Insert Figure 4.16>

4.7.5 Efficient implementation of binary trees from decision regions

In some applications requiring representation of decision tables through efficient binary trees, the class assignment for every cell is rigid and given. Such a problem of efficient implementation of decision-making through the use of binary trees can also be solved by dynamic programming procedures similar to the above approach (Payne and Meisel, 1977). Many such optimal decision tree design problems are computationally complex (Murthy, 1998). The fundamental reason for this is that the number of subcases to be examined increases exponentially as a function of some key parameter. Even in the case of designing a binary decision tree to examine which of the n bins a given scalar measurement falls into, the complexity is exponential in n . Therefore, many researchers recommend the use of heuristics to design decision trees. However, these comments apply more to large directory look-up problems. In practice, many pattern classification problems have only a modest number of measurements and a modest number of quantization levels. Computations for the design of optimal trees in these cases are indeed feasible.

4.8 Decision tree design from labeled training samples

A very common starting point for the design of decision trees in application areas is with a set of labeled training samples. Simple approaches to designing decision trees in such cases is the subject of study here. The problem of developing optimal separating hypersurfaces from the training data

set is not the topic here. So far, we have seen ample evidence to the effect that the most convenient form of decision boundaries for decision tree design are segments of hyperplanes parallel to the coordinate axes of the feature space (data vector space). Hyperplane segments inclined to the coordinate axes form a second choice set. In practice, a good combination of these two types of decision boundaries serves the best. It is always possible to separate a set of labeled training samples, even from multiple classes, with the help of hyperplane segments¹. However, the use of a decision tree is in applying the classification algorithm to new data samples. Although the new samples come from the same cause as the training samples do, the finite set training samples do not perfectly depict all the characteristics and variability in all possible samples from the original cause.

4.8.1 Linear discriminant functions

Decision regions separated by hyperplanes are obtained through the use of linear discriminant functions. A linear discriminant function separates the feature space into two regions separated by a hyperplane. Decision regions formed through the use of j multiple linear discriminant functions divide the feature space into a maximum of 2^j regions. It is not necessary for all of these 2^j regions to be assigned a different class label. Indeed, if the number of class labels is k , we merely need $j \geq \log_2(k)$ and $(2^j - k)$ regions in the feature space have class labels repeated from the other k regions. The general form of a linear discriminant function is

$$\sum_{j=1}^m w_j x(j) + w_0. \quad (4.26)$$

All the data vectors that satisfy

$$\sum_{j=1}^m w_j x(j) + w_0 > 0 \quad (4.27)$$

lie on one side of the corresponding hyperplane and all the data vectors satisfying

$$\sum_{j=1}^m w_j x(j) + w_0 < 0 \quad (4.28)$$

¹The only exception is if two or more training samples from *different* classes are identical in all feature measurements.

lie on the other side. Data vectors satisfying

$$\sum_{j=1}^m w_j x(j) + w_0 = 0 \quad (4.29)$$

lie exactly on the hyperplane separating the two regions. A linear discriminant function for a decision boundary that is perpendicular to the $x(j)$ axis and parallel to all other coordinate axes is of the form

$$x(j) + w_0. \quad (4.30)$$

The approach suggested here develops optimal hyperplanes between neighboring pairs of classes. This leads to decision regions separated by piecewise hyperplanes. Linear combinations of the data vectors are formed that lead to rectangular decision regions in a transformed feature space. The approaches of Section 4.7 are then applicable for completion of decision tree design. Figure 4.17 shows a simple example of data samples generated with random numbers.

<Insert Figure 4.17>

Two-dimensional data samples from four classes are plotted with different symbols for points from different classes. Such plots are called scatter plots. These random numbers are generated to simulate the decision boundaries in Figure 4.2, except for the data of class 2. Representative straight line segments are drawn for possible decision boundaries. The line segment parallel to the y axis completely separates samples from classes 2 and 3. One of the line segments also completely separates samples from classes 3 and 4. The other two line segments separate samples from corresponding pairs of classes, but these boundaries result in a few misclassified samples. It is clear that there is no single straight line that separates samples from classes 1 and 2. Similarly, It is also clear that there is no single straight line that separates samples from classes 1 and 3. It is possible to draw several straight line segments to completely separate the sets of samples from different categories to be in different regions. For example, the straight line parallel to the x -axis in Figure 4.17 may be replaced by a set straight line segments to correctly classify all the given labeled training samples. Each such line would be parallel to the x -axis or the y -axis. Together, these line segments would retain all the samples belonging to class 2 on the lower side of such a complicated boundary, and all the samples belonging to class 1 on the higher side of the same boundary. However, no such approach to correctly classify all the given labeled training samples can guarantee perfect classification of future pattern samples. Such an attempt to classify all or most of labeled training samples with complicated boundaries is called “over-fitting.” Such over-fitting leads to larger decision trees

in which a decision region is split into multiple sub-regions with different classes. A large decision tree can be pruned to eliminate some over-fitting.

The above characteristics about decision boundaries constructed with straight lines give us clues on the types of optimizations used to find good straight lines. These are briefly described here. In the case of labeled sample sets, the distance between the mean values of samples from different classes allow us to form pairs of neighboring classes. Therefore, procedures for determining straight lines can attempt to concentrate on neighboring pairs of classes.

1. **Lines parallel to coordinate axes**

Examining one-dimensional scatter plots in each feature dimension is very simple. The entire interval over which all the samples fall can be divided into several regions, based on whether or not these lines reasonably well separate the samples. This procedure can be applied for each individual measurement.

2. **Hyperplanes with arbitrary orientations**

There are two subcases in this category.

- (a) In the first one, there exists a hyperplane that can completely separate samples from two classes. Such pairs of classes are called linearly separable classes. The straight line separating samples from classes 3 and 4 in Figure 4.17 is such an example. Given two sets of training samples, one for each class label, whether or not they are linearly separable can be determined through linear programming. Intuitively, the best hyperplane to separate a linearly separable pair of classes is the one that has equal and the largest distance from the hyperplane to the nearest samples, from each class. There are mathematical programming techniques to find such an optimally separating hyperplane. The resulting hyperplane is called a Support Vector Machine. There is a rich class of other criteria and corresponding mathematical programming algorithms to determine optimal separating hyperplanes. Duda, *et al.* (2001), is an excellent book on this topic.
- (b) In the second case, there is no hyperplane that can perfectly separate samples from two classes. Samples from classes 1 and 3 in Figure 4.17 constitute such an example. In this case, optimization problems to minimize some meaningful criteria are formulated and solved to find optimal hyperplanes. An example of such a criterion is the minimum

of the sum of squared distance from all the samples of both the classes to the hyperlane. This problem can be solved through the use of the pseudo-inverse of a matrix formed with labeled data vector samples from both the classes (Duda, *et al.*, 2001).

These are only some general suggestions to develop linear decision boundaries. The task of decision tree design from such boundaries follows approaches developed in earlier sections.

4.9 Unsupervised decision tree design

In some applications, we have unlabeled or unclassified pattern samples for training. Each pattern sample is a point in the data vector space. There is no decision or class label assigned to any of the samples. Inferring the existence of any natural grouping of samples, designing a scheme to separate them into such groups, and designing a classification scheme to assign future appearances of pattern samples to one such group – all fall under unsupervised learning and classification approaches. The structure of the data set may indicate that the set naturally groups into a few categories. Each category may correspond to some physical significance. Specifically, observations within a group may point to some strongly likely consequences. In meteorology, for example, certain physical conditions may result in an unstable state or cause an imbalance. Such temporary conditions may cause further changes until the overall state stabilizes. In the multidimensional mathematical space of all physical variables, there may be several regions of stable states separated by regions of unstable states. This may be the underlying reason for the data to be found in subjectively distinct groups. Some combinations of unstable physical attributes may result in regenerative effects creating severe weather patterns. An example of physical conditions causing regenerative effects is in the occurrence of snow avalanches in mountains. A disturbance causing some movement of snow at high altitudes may trigger more disruption in the snow. Beyond a critical point, the regenerative effect can be very strong.

Another example of occurrence of natural clusters is in biological taxonomy. The overall set of plants and/or animals is hierarchically (repeatedly) split into finer and finer groups. Members in a group have strong similarities and members in different groups have strong dissimilarities. There may be cases (members) that do not clearly belong to one or another class, but such cases are rare. One reason for natural formation of such distinct hierarchical groups is that certain combinations of physical attributes favor regenerative survival.

Clustering of unlabeled data is a first step in designing a decision tree from unlabeled pattern samples. The emphasis in this chapter is not on general clustering approaches. Anderberg (1973) is a classic book on clustering. Kaufman and Rousseeuw (1990) is a more recent one on the same topic. Clustering approaches can be influenced for better design of decision trees. That is, decision boundaries parallel to the coordinate axes of the space of features are sought whenever feasible. There are two main approaches to clustering. The bottom-up or the data driven approach starts grouping data samples into pairs, groups of three, etc., and grow into larger clusters all the way to a single group. This approach is called agglomerative clustering. Some subjectively appealing quantitative criteria to determine a reasonable number of distinct clusters are available. Alternatively, top-down or model driven approaches are also available. In its most elementary form, the unlabeled training samples are divided from an initial single group all the way down to separate groups for individual data samples. This approach is known as divisive clustering. Again, appealing criteria are available to stop at a reasonable number of clusters.

Such clustering approaches finally lead to decision assignments for every sample in the entire set of pattern samples. These results should be fed to algorithms for determining decision regions. That is, instead of assigning decisions to data points in the data space, larger regions should be identified for decision regions. Simpler approaches construct piecewise hyperplane segments parallel to the coordinate axes in the data space. Other approaches with arbitrarily oriented hyperplanes and even nonlinear hypersurfaces are possible.

In both the top-down and bottom-up methods, attempts should be made to incorporate all available information about the application system and data. Often, much of such information may be in subjective forms. The decision tree designer would use judgments and mathematical interpretations of such information. These approaches go hand-in-hand with the above approach of determining decision regions from clustered data. A particular example of such a refinement technique is “pruning of decision trees.” After designing a decision tree, the designer can examine the worthiness of distinguishing some pairs or groups of decisions (either at intermediate or at final decision-making stages). Merging some decision regions with this approach eliminates some nodes in the tree and results in a pruned tree.

4.9.1 Bottom-up method

A pure bottom-up method of clustering successively combines (merges) sets of samples until a specified number of classes (groups) is reached. Usually, such combinations are continued until all the samples are in one single group. Thereafter, it is a little easier to decide on the number of classes and this may be influenced by a suggested number of classes. Central to determining which pair of a partition of the sample set are the *closest* for merging is the definition of a distance measure between two sets of samples. A simple distance is the Euclidean distance between the centroids of the two sample sets. If $\{\mathbf{x}_{1j}, j = 1, \dots, n_1\}$ and $\{\mathbf{x}_{2j}, j = 1, \dots, n_2\}$ are the two sets of samples, the mean data vector of the set i is

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbf{x}_{ij}. \quad (4.31)$$

The Euclidean distance between two m -dimensional vectors $\mathbf{y} = [y(1), \dots, y(m)]$ and $\mathbf{z} = [z(1), \dots, z(m)]$ is simply the geometric distance given by

$$d(\mathbf{y}, \mathbf{z}) = \sqrt{\sum_{l=1}^m [y(l) - z(l)]^2}. \quad (4.32)$$

A drawback of the Euclidean distance is that it treats relative distances along all the feature measurement axes in the same way. In reality, different features may have different physical dimensions and may require normalization. Kaufman and Rousseeuw (1990), introduce methods for combining variables of different types into a single dissimilarity measure.

Attempting to separate the samples with hyperplanes parallel to the coordinate axes is preferable for the eventual implementation of a decision tree. This will also avoid the problem of different scales along different axes. However, during a hierarchical separation of a set of points along a line (along one feature measurement), we do not know the number of classes to split the data into. Therefore, separating or clustering over individual axes in the feature space is recommended only if sample sets are “well separated.” If such an approach is viable, the resulting clusters may be further separated by the usual hierarchical approaches. A quantitative criterion of well separate-ness considers the spread of data samples within each class as well as the spread between the two clusters. Following is one appealing definition of such a measure. The mean square distance of all

samples within a class to its centroid is a measure of spread within a class.

$$\sigma_i^2 = \frac{1}{n_i} \sum_{j=1}^{n_i} \sum_{l=1}^m [x_{ij}(l) - \mu_i(l)]^2. \quad (4.33)$$

The inter-class spread is the square of the distance between the centroids of the two classes,

$$d^2(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = \sum_{l=1}^m [\mu_1(l) - \mu_2(l)]^2. \quad (4.34)$$

A normalized measure of separateness between these two populations is the ratio of the average of the intra-class spread to the inter-class spread,

$$\frac{\sigma_1^2 + \sigma_2^2}{2d^2(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)}. \quad (4.35)$$

The above quantities are based on multi-dimensional data vectors. If data sets over individual measurement variable are considered, $m = 1$ in equations (4.33) - (4.35). In the case of one dimension, the above normalized measure of separateness is invariant to scale. If the measure is less than one, the two groups of samples may be considered to be well separated. Figure 4.18 shows data samples that can be separated into two sets along the x axis by a threshold value of about 10.

<Insert Figure 4.18>

4.9.2 Top-down method

Instead of combining individual samples into pairs, and larger groups up to a desired number of clusters, the overall data set can be considered to be a single group in the beginning and repeatedly divided until we have a desired number of clusters. This approach is known as divisive clustering.

An advantage of divisive clustering over agglomerative clustering is that known information about inter-relationships can be incorporated into the divisive clustering algorithm. For example, in an application, approximate locations of the central portions of data vectors for one or more of the classes may be known. Alternatively, some examples of causes and effects may be known and these may be used to obtain a skeletal versions of trees for further refinement using many unlabeled samples. Kaufman and Rousseeuw (1990), present a divisive hierarchical clustering algorithm called the DIANA.

4.9.3 Interactive approaches

In many practical applications, the decision tree design criterion is not well-specified at all. It is hoped that the above sections provide helpful approaches at various stages of data analysis and design of decision-making algorithms. Typically, the overall design may require some trial and error approaches and iterative refinements, at least at some stages of development. One example of these is eliminating outliers from consideration. One or a few data samples that lie significantly away from all other samples in the feature space can tilt the resulting solutions unreasonably. They may be eliminated from consideration. A second example of interactive changes is the pruning of decision trees. If a sub-tree covers very few samples, having a subtree for correct decisions within them may result in over-fitting. Such a decision region may be merged with a neighboring subtree. In the feature space, this amounts to combining small distinct regions with larger neighboring regions for simplification of the resulting tree.

4.10 Further reading

The book by Duda, *et al.* (2001), is a standard text and reference for pattern classification. It has detailed treatment of Bayesian decision theory, parameter estimation, linear discriminant functions, and clustering. It also has a section on support vector machines and two sections on decision trees. Anderberg (1973), and Kauffman and Rousseeuw (1990), are books on clustering. Breiman, *et al.* (1984), is a book on developing trees from a purely statistical perspective. Decision trees are commonly covered in the wider topics of artificial intelligence (Russell and Norvig, 2002) and data mining (Han and Kamber, 2006). The book, by Quinlan (1993), is on programs for machine learning and has detailed discussions on many issues in decision tree development.

From time to time, survey articles on decision trees appear in literature. Most of these articles extensively cite and survey research reports on decision trees from numerous different points of view such as tree models from the nature of applications, logical approach to decision-making, problems in directory searching, computational complexity, their relations with neural networks, heuristics, and applications. Moret (1982), and Murthy (1998), are both extensive and scholarly surveys. The main emphasis in Moret (1982), is on representing Boolean functions as decision trees. On the other hand, Murthy (1998), is a more general survey of representation of data with decision trees. Other survey articles include the following. A synthesis of work on tree-structures

in many application areas such as pattern recognition, decision tables, fault location, coding theory, and questionnaire design appears in Payne and Preece (1980). Dattatreya and Kanal (1985) is an overview of decision trees in pattern recognition. Various existing methods for designing decision tree classifiers and their potential advantages over single state classifiers are surveyed in Safavian and Landgrebe (1991).

The treatment of decision tree design in this chapter is introductory. Many specialized techniques for decision tree design have been developed and reported in the research literature. Some of these are application specific and some are general. These approaches are very helpful for an advanced study and for a serious designer in applications areas. Manago and Kodratoff (1991), develop an algorithm called KATE that learns decision trees from complex structured data. Evaluation of which feature to use at which point in the tree design is a well investigated problem. Rules for these are based on information theory, distance measures, or dependence measures (Ben-Bassat, 1982). A more recent reference on feature selection measures is Kononenko and Hong (1997). Neural networks and decision trees are related approaches for decision-making. Such decision trees with parametric classifiers with small experts at decision-making nodes is studied by Jordan and Jacobs (1994). Chai *et al.* (1996), study genetic algorithms to develop linear splitting at decision-making nodes. Various methods for pruning trees have been suggested and investigated. Kim and Koehler (1994), analyze the conditions under which pruning improves the accuracy. Rastogi and Shim (1998), integrate decision tree construction with tree pruning.

Several commercial software packages are available for help with decision tree design, research, and applications. A good list of current software packages is available on the Knowledge Discovery and Nuggets web-page,

<http://www.kdnuggets.com/software.html> (4.36)

Within the class of commercial software for decision trees, there are two widely cited packages. One is CART, developed by Breiman, *et al.* (1984), and upgraded from time to time. The second package is C5.0, developed following the book by Quinlan (1993). The older version of this, C4.5, is available as free software from the same web-site. The web-site also has several other commercial packages as well as free software packages.

Following are recent articles that use decision tree principles for applications related to environmental sciences. Cannon and Whitfield (2002), use tree-based recursive partitioning models to develop mappings between synoptic-scale circulation fields and the leading linear and nonlinear

principal components of weather elements observed at a surface station. Goel, *et al.* (2003), use decision trees and artificial neural networks to identify weed stress and nitrogen status of corn. They use hyperspectral data from a compact airborne spectrographic imager. Li and Claramunt (2006), design decision trees for the classification of geographical information. Their approach takes into account spatial autocorrelation phenomena in the classification process.

4.11 Conclusion

The practice of environmental sciences frequently encounters the problem of decision making in uncertainty. The task of decision making typically involves the measurement of some physical attributes and the assignment of a decision. The desired objective of decision making is to minimize some overall cost criterion. The mathematical relationships among the various aspects of the measurement and the decision making process govern the cost incurred through the process. The class of mathematical relationships includes the subclass of statistical relationships. In reality, the availability of such relationships are limited in three ways. All the aspects of relationships may not be available. Among those that are available, some may be represented ambiguously, instead of being represented accurately. Some of the relationships may be unambiguously represented but the representations may differ slightly from the corresponding real relationships. In addition to these drawbacks, development of good decision making algorithms may pose the following additional challenge. A conceived decision making algorithm may require prohibitive amount of computer time and/or memory. Artificial intelligence approaches help us to strike a compromise between (1) approximate representation of relationships, (2) computational complexities, and (3) the quality of the final solution to the problem at hand. Decision making through the use of decision trees is one such artificial intelligence approach.

Decision trees are useful in many data analysis and classification applications, including medical diagnosis, meteorology, agriculture, pollution monitoring, and remote sensing. They are useful in applications in which hierarchical cause-effect relationships occur naturally. They are useful in applications with the peculiarity of a naturally recursive division of classes within classes. They are useful in applications in which expensive measurements are to be undertaken only if other readily available data so indicate. Finally, even in applications without any such model-driven reason for the use of decision trees, it may simply turn out to be convenient to design and im-

plement decision-making algorithms as trees. Therefore, there are no global decision tree design approaches. A designer should use all available information about the nature of the problem, cause-effect relationships, and the structure of the training data for decision tree design. Ideas and approaches developed in this chapter can be used at any stage of the development of the decision-making algorithms, directly, or with modification. If decision regions are explicitly given, the design of a decision tree is conceptually simple. If possible decision boundaries are suggested, this information will help in the joint task of decision (pattern class label) assignment and decision tree design. If labeled or unlabeled training samples are given, the goal of designing a decision tree classifier influences the structure of decision boundaries. Approaches and techniques to jointly determine such decision boundaries and design decision trees are developed here.

The literature on Decision Trees is vast. Some recent books on pattern classification, clustering, artificial intelligence, and data mining are suggested for the interested reader. There are many survey articles on decision trees, some very general, and some with special emphasis. A few such articles are also cited. These are also invaluable resources of annotated bibliographies. Several research articles dealing with specific methods for decision tree design are included for readers interested in exploring advanced techniques. Recent research articles with applications of decision trees in areas related to environmental sciences are listed and suggested for the reader to gain an experimental perspective. Finally, many commercial software packages are available for the applied researcher. Location of these services on the world-wide web are given.

Acknowledgment

The author thanks the reviewers for many constructive comments and Professor S. Lakshmivarahan for helpful suggestions throughout the preparation of this chapter.

References

Anderberg, M. R.: 1973, *Cluster Analysis for Applications*, Academic Press, New York.

Ben-Bassat, M.: 1982, 'Use of distance measures, information measures, and error bounds on feature evaluation', in P. R. Krishnaiah, & L. N. Kanal, (eds.), *Classification, Pattern Recognition, and Reduction of Dimensionality. Volume 2, Handbook of Statistics*, Elsevier Science

Publishers, pp. 773-791.

Breiman, L., Friedman, J., Olshen, R., & Stone, C.: 1984, *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA.

Cannon, A. J., & Whitfield, P. H.: 2002, 'Synoptic map-pattern classification using recursive partitioning and principal component analysis', *Monthly Weather Review* **130**, 1187–1206.

Chai, B.-B., Zhuang, X., Zhao, Y., & Sklansky, J.: 1996, 'Binary linear decision tree with genetic algorithm', *Proceedings of 13th International Conference on Pattern Recognition*, Los Alamitos, CA, pp. 530–534.

Dattatreya, G. R., & Kanal, L. N.: 1985, 'Decision trees in pattern recognition', in L. N. Kanal, & A. Rosenfeld (eds.), *Progress in Pattern Recognition 2*, North-Holland, pp. 189-237.

Duda, R. O., Hart, P. E., & Stork, D. G.: 2001, *Pattern Classification*, Wiley-Interscience.

Goel, P. K., Prasher, S. O., Patel, R. M., Landry, J. A., Bonnell, R. B., & Viau, A. A.: 2003, 'Classification of hyperspectral data by decision trees and artificial neural networks to identify weed stress and nitrogen status of corn', *Computers and Electronics in Agriculture* **39**, 67–93.

Han, J., & Kamber M.: 2006, *Data Mining*, Morgan Kauffman Publishers.

Jordan, M. I., & Jacobs, R. A.: 1994, 'Hierarchical mixtures of experts and the EM algorithm', *Neural Computation* **6**, 181–214.

Kauffman, L., & Rousseeuw, P. J.: 1990, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley and Sons.

Kim, B., & Koehler, G. J.: 1994, 'An investigation on the conditions for pruning an induced binary tree', *European Journal of Operations Research* **77**, 82–95.

Kononenko, I., & Hong, S. J.: 1997, 'Attribute selection for modeling', *Future Generation Computer Systems* **13**, 181-195.

Knowledge Discovery and Nuggets web-page,

<http://www.kdnuggets.com/software.html>

- Li, X., & Claramunt, A.: 2006, 'A spatial entropy-based decision tree classification of geographical information', *Transactions in Geographical Information Systems* **10**, 451–467.
- Manago, M., & Kodratoff, Y.: 1991, 'Induction of decision trees from complex structured data', in G. Piatetsky-Shapiro & W. J. Frawley (eds.), *Knowledge Discovery in Databases*, AAAI/MIT Press, pp. 289-306.
- Moret, B. M. E.: 1982, 'Decision trees and diagrams', *ACM Computing Surveys* **14**, 593–623.
- Murthy, S. K.: 1998, 'Automatic construction of decision trees from data: A multi-disciplinary survey', *Data Mining and Knowledge Discovery* **2**, 345–389.
- Payne, R. W., & Preece, D. A.: 1980, 'Identification keys and diagnostic tables', *Journal of Royal Statistical Society: series A* **143**, 253-292.
- Payne, H., & Meisel, W.: 1977, 'An algorithm for constructing optimal binary decision trees', *IEEE Transactions on Computers* **26**, 905–916.
- Quinlan, J. R.: 1990, 'Decision trees and decision-making', *IEEE Transactions on Systems, Man, and Cybernetics* **20**, 339–346.
- Quinlan, J. R.: 1993, *Programs for Machine Learning*, Morgan Kaufmann Publishers.
- Rastogi, R. & Shim, K.: 1998, 'Public: A decision tree classifier that integrates building and pruning', in *Proceedings of Int. Conference on Very Large Data Bases*, New York, pp. 404-415.
- Russell, S., & Norvig, P.: 2002, *Artificial Intelligence: A Modern Approach*, Prentice Hall.
- Safavian, S. R., and Landgrebe, D.: 1991, 'A survey of decision tree classifier methodology', *IEEE Transactions on Systems, Man, and Cybernetics* **21**, 660–674.
- Wilson, P. F, Dell, L. D., & Anderson, G. F.: 1993, *Root Cause Analysis*, American Society for Quality, Milwaukee, WI.

Glossary

Item	Definition	Page
\mathbf{x}	observation or data vector	9
pattern classification	assignment of a data vector to one of a known set of classes	9
$\Omega = \{\omega_1, \dots, \omega_k\}$	set of classes	9
cardinal variable	takes values over an uncountable ordered set	9
ordinal variable	takes values over a countable ordered set	9
nominal variable	takes values over a finite unordered set	9
P_i	probability of event denoted by i	10
$p(\mathbf{x} \omega_i)$	probability density function of observation \mathbf{x} given that the observation comes from class ω_i	10
$P[\omega_i \mathbf{x}]$	<i>a posteriori</i> probability of class ω_i , based on observation \mathbf{x}	10

Item	Definition	Page
decision boundary	line, surface, or hypersurface between observation sets for different decisions	12
decision region	set of all possible observation points assigned to a particular decision	12
vertex or node	any physical or abstract entity with possible relations to other such entities	16
graph	set of vertices and a set of pairs of vertices chosen from the vertex set	16
edge or link	a pair of vertices in a graph	16
undirected graph	graph with only unordered pairs of nodes for edges	16
directed graph	graph with only ordered pairs of nodes for edges	16
degree of a node	number of edges that the node touches	16
in-degree	number of edges terminating at a node	17

Item	Definition	Page
out-degree	number of edges originating from a node	17
path	sequence of vertices with each overlapping pair of successive vertices being a valid edge in a graph	17
length of a path	number of edges in the path	17
cycle	path in a graph with the same beginning and ending vertex	18
tree	undirected graph with no cycles	18
rooted tree	tree with one vertex identified as the root node	18
leaf node or terminal node	non-root node with a degree of one in a tree	19
graph traversal	systematically visiting nodes in a graph	19
child-node	a node following another node in a path from root node to a leaf node	19

Item	Definition	Page
height of a rooted tree	length of the longest path of all the paths from the root node to all leaf nodes	20
directed acyclic graph (DAG)	directed graph with no cycles	20
binary tree	tree which has exactly two child nodes for every non-leaf node	20
partition	set of mutually exclusive and collectively exhaustive subsets	20
decision tree	tree with a rule for deciding the child node to traverse to, at every non-leaf node	20
rooted directed acyclic graph (RDAG)	DAG with a root node from which paths exist to all other nodes	20
feature space	set of all possible observations of all available measurement variables	26
labeled training sample	data vector with a known decision	29
linear discriminant function	linear function of data minus a threshold	30
unsupervised classification	grouping a set of data vectors with no prior information about the distinguishing nature of different groups	33

List of Figures

Figure Number	Figure caption	First referred on page	Drawn on page
4.1	A simple example illustrating model and statistical training	11	48
4.2	An example of decision regions	12	49
4.3	Piecewise linear decision boundaries	15	50
4.4	An undirected graph	16	51
4.5	A directed graph	17	52
4.6	A tree obtained from the graph of Figure 4.4	19	53
4.7	The tree of Figure 4.6 with a root at the top and branching downwards	19	54
4.8	A DAG with no unique root node, constructed as a subgraph of Figure 4.5	20	55
4.9	An RDAG extracted from the directed graph of Figure 4.5	20	56
4.10	A binary decision tree obtained by modifying the decision tree of Figure 4.1	21	57
4.11	Rectangular decision boundaries	22	58
4.12	A binary decision tree for the example of Figure 4.11	22	59
4.13	A non-binary decision tree for the example of Figure 4.11	22	60
4.14	A binary decision tree for the example of Figure 4.2	24	61
4.15	Universal graph for optimal tree design example	28	62
4.16	A possible optimal decision tree in the form of a rooted acyclic graph, developed from Figure 4.15	29	63
4.17	A scatter plot example with 4 classes	31	64
4.18	A scatter plot of unlabeled samples well separated into two groups along x axis	36	65

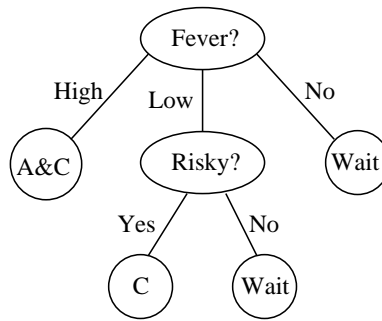


Figure 4.1: A simple example illustrating model and statistical training

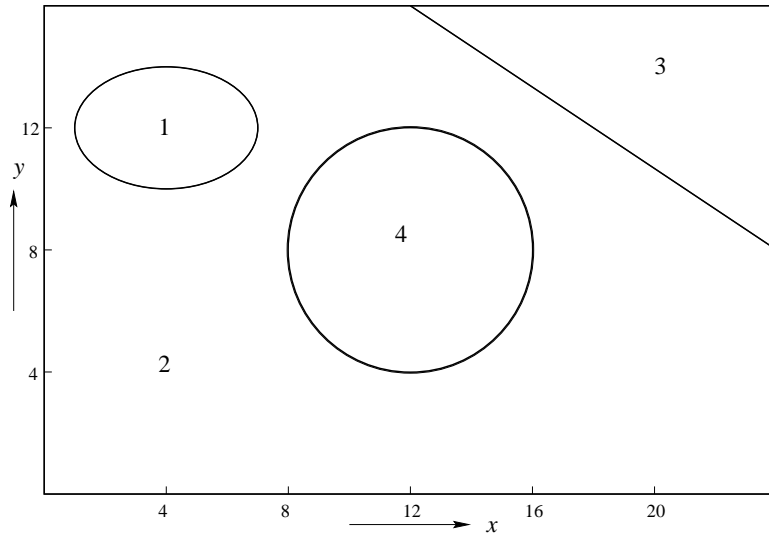


Figure 4.2: An example of decision regions

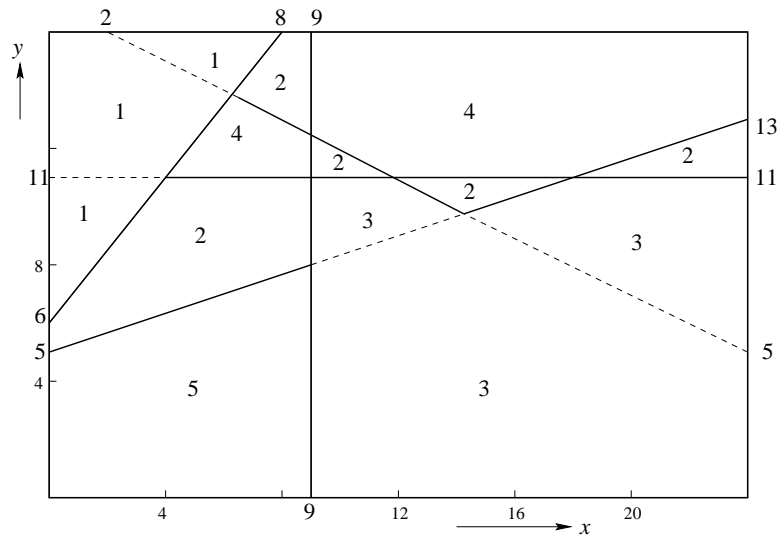


Figure 4.3: Piecewise linear decision boundaries

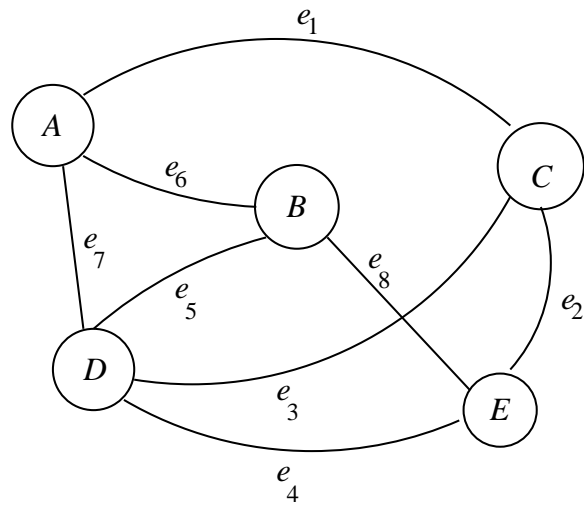


Figure 4.4: An undirected graph

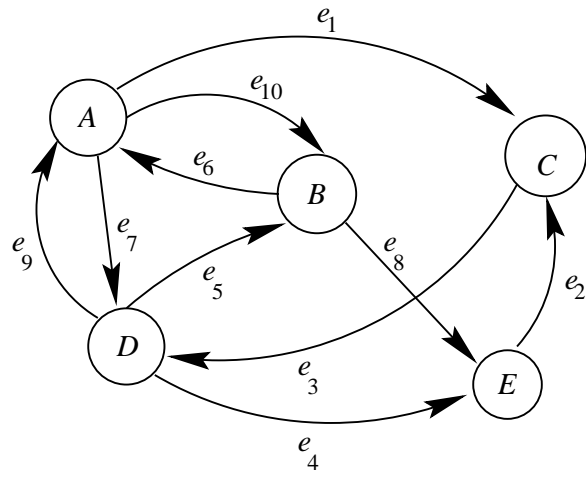


Figure 4.5: A directed graph

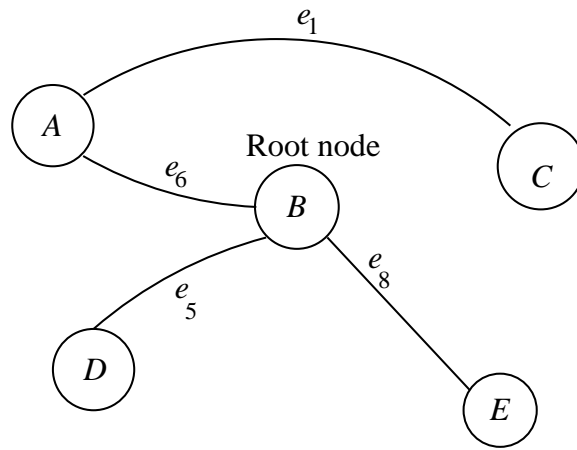


Figure 4.6: A tree obtained from the graph of Figure 4.4

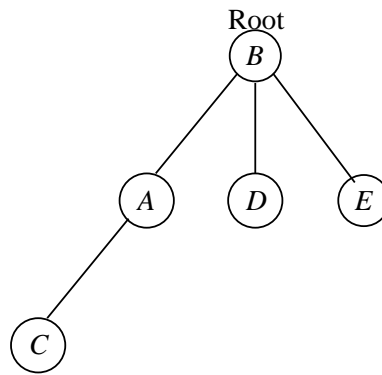


Figure 4.7: The tree of Figure 4.6 with a root at the top and branching downwards

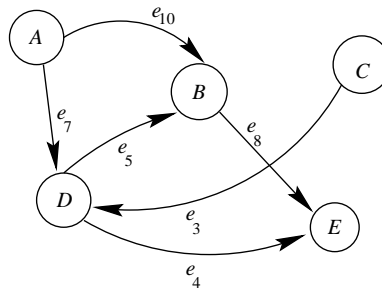


Figure 4.8: A DAG with no unique root node, constructed as a subgraph of Figure 4.5

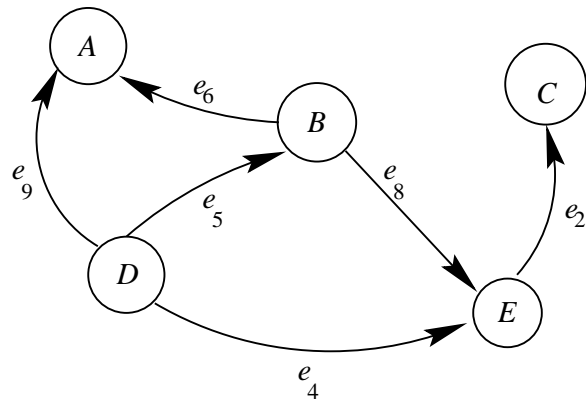


Figure 4.9: An RDAG extracted from the directed graph of Figure 4.5

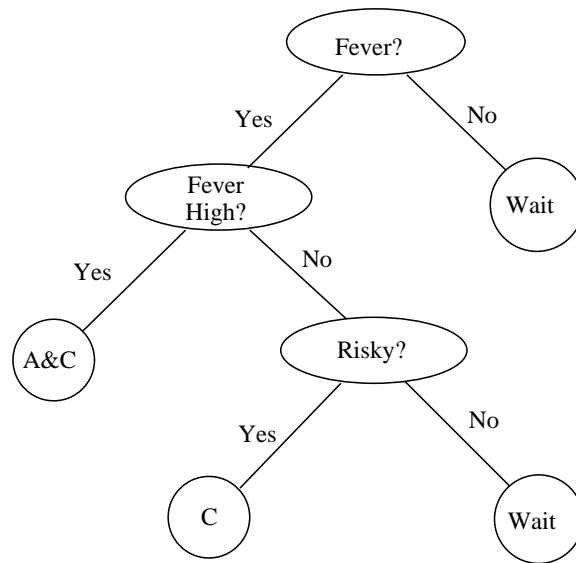


Figure 4.10: A binary decision tree obtained by modifying the decision tree of Figure 4.1

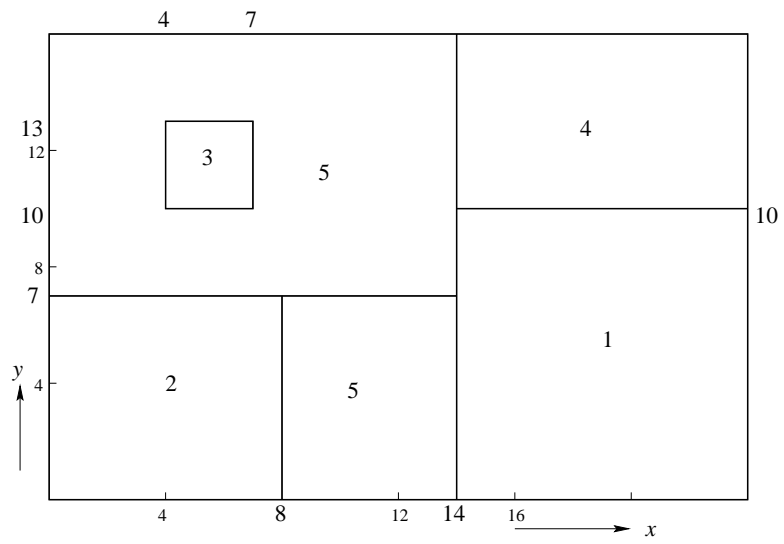


Figure 4.11: Rectangular decision boundaries

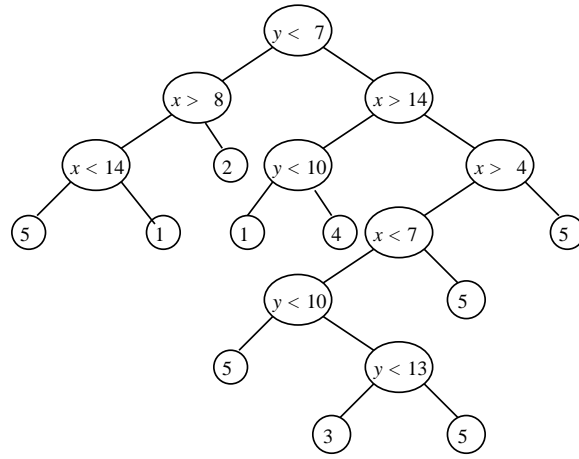


Figure 4.12: A binary decision tree for the example of Figure 4.11

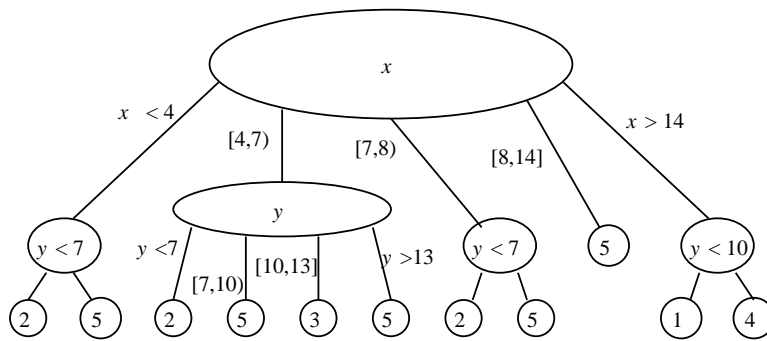


Figure 4.13: A non-binary decision tree for the example of Figure 4.11

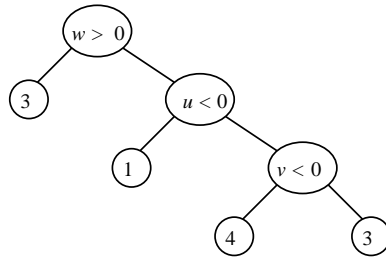


Figure 4.14: A binary decision tree for the example of Figure 4.2

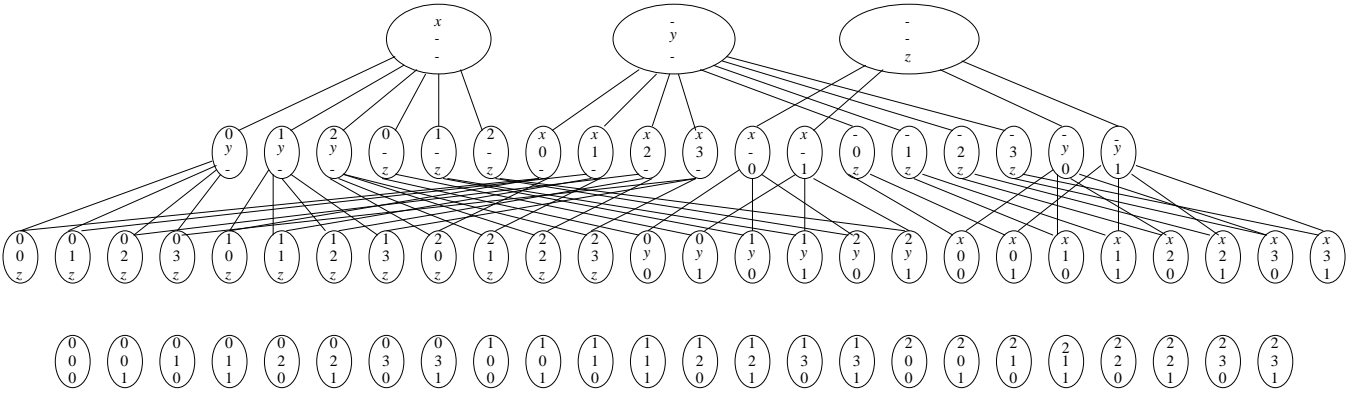


Figure 4.15: Universal graph for optimal tree design example

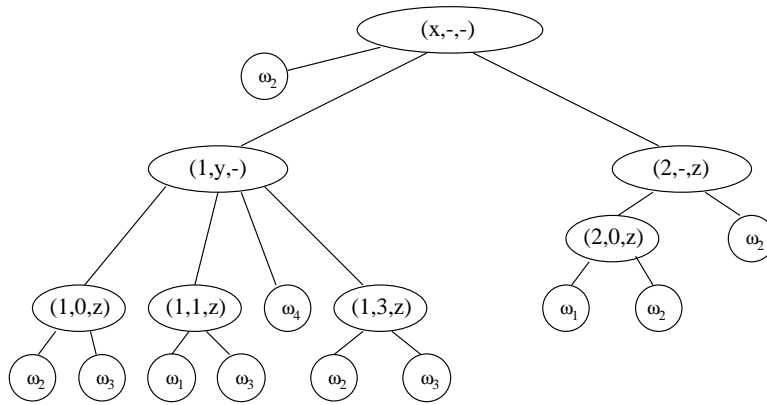


Figure 4.16: A possible optimal decision tree in the form of a rooted acyclic graph, developed from Figure 4.15

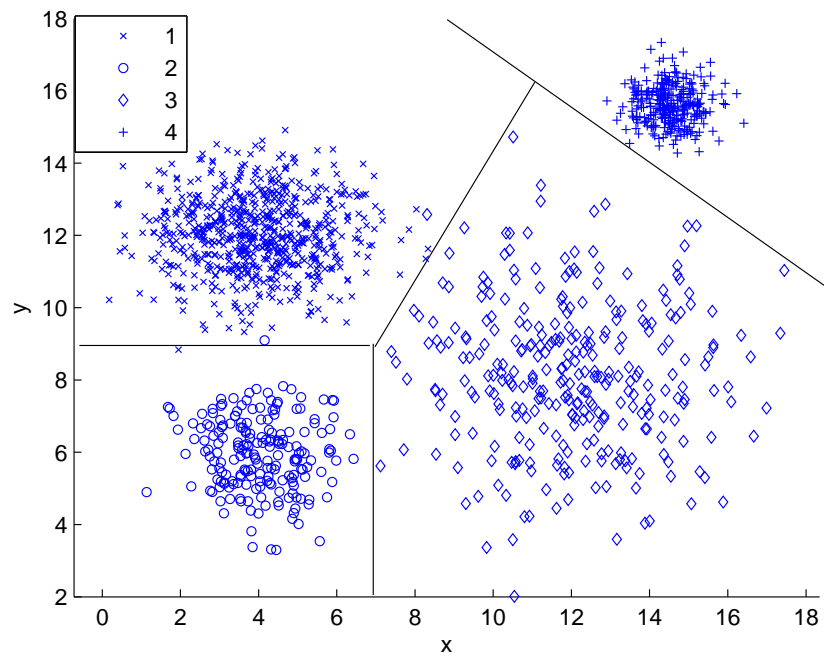


Figure 4.17: A scatter plot example with 4 classes

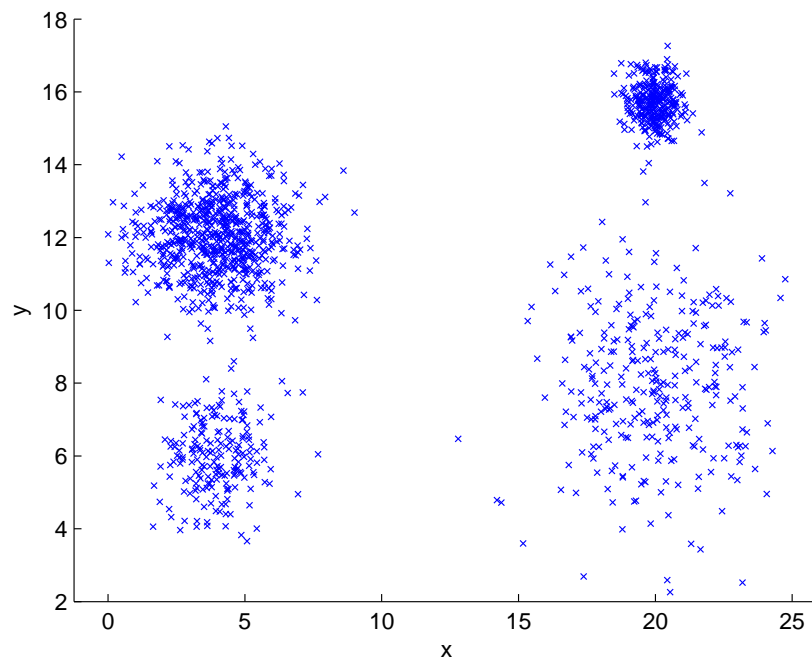


Figure 4.18: A scatter plot of unlabeled samples well separated into two groups along x axis

Index

a posteriori

probability, 10, 14, 25

a priori

knowledge, 8

probability, 10, 12–14

affine transformation, 24

agglomerative clustering, 34, 36

agriculture, 39

algorithm

classification, 14

clustering, 36

decision-making, 9, 12

genetic, 38

Anderberg, 34, 37, 40

arc, 16

Artificial Intelligence, 8

attribute

physical, 7

Bayesian decision theory, 37

Bayes theorem, 10

Ben-Bassat, 38, 40

benefit, 7

binary decision tree, 21, 22, 29

binary tree, 20, 22, 29

Boolean function, 37

bottom-up clustering, 34

branch, 16

Breiman, 37, 38, 41

C4.5, 38

C5.0, 38

Cannon, 38, 41

cardinal, 9

CART, 38

category, 9

Chai, 38, 41

child node, 19

circle, 23

class

assignment, 14

label, 22

classification

cost, 26

matrix, 26

hierarchical, 9

pattern, 14

classifier

design

interactive, 37

linear, 14

piecewise linear, 15

- cluster
 - well separated, 35
- clustering, 34, 37
 - agglomerative, 34, 36
 - algorithm, 36
 - bottom-up, 34
 - data driven, 34
 - distance measure, 38
 - Euclidean, 35
 - inter-class, 36
 - intra-class, 36
 - divisive, 36
 - top-down, 36
- complete model, 8
- composite feature variable, 23
- computational complexity, 37
- connected graph, 18
- continuous segment, 9
- continuous variable, 12
- corrective action, 7
- cost
 - expected, 26
 - feature measurement, 7
 - matrix, 26
- countable set, 10
- cumulative distribution function, 13
- cycle, 18
- DAG, 20
- data
 - analysis, 39
 - driven, 34
 - mining, 37
 - space, 12
 - training, 12
 - statistical, 11
 - vector, 9, 20
- Dattatreya, 3, 26, 38, 41
- decision
 - making, 9, 14
 - algorithm, 11
 - pattern classification, 9
 - statistical, 7
 - boundary, 12, 25
 - final, 12
 - nearly optimal, 8
 - optimal, 7, 8
 - partial, 12, 20
 - region, 12, 14, 15
 - tree, 8, 12, 15, 20
 - binary, 21, 22, 29
 - design, 20, 21, 25, 27, 29
 - example, 21
- decision tree, 9
 - example, 21
- decision tree design
 - unsupervised, 33
- degree, 16
 - in-degree, 17
 - out-degree, 17
- derivative, 13
- deterministic model, 8
- directed acyclic graph, 20

- directed graph, 16
- directory searching, 37
- discrete variable, 12
- discriminant
 - function, 14
 - linear, 14, 30, 37
- distance measure, 38
 - Euclidean, 35
- divisive clustering, 36
- Duda, 9, 14, 32, 33, 37, 41
- dynamic programming, 27
- earth science, 7
- ellipse, 23
 - major axis, 23
 - minor axis, 23
- environmental science, 7
- Euclidean distance, 35
- event, 14
- exact model, 8
- expected cost, 26
- feature
 - measurement, 7, 14
 - cost, 7
 - space, 14, 26
 - variable
 - cardinal, 9
 - nominal, 10
 - ordinal, 10
- final decision, 12
- function
 - Boolean, 37
- Gaussian, 13
- genetic algorithm, 38
- Goel, 39, 41
- graph, 15
 - connected, 18
 - cycle, 18
 - directed, 16
 - directed, acyclic, 20
 - directed acyclic
 - rooted, 20
 - node, 15
 - path, 17
 - traversal, 17, 19
 - tree, 18
 - undirected, 16
 - vertex, 15
- Han, 37, 41
- height, 20
- heuristics, 8, 37
- hierarchical classification, 9
- hyperplane, 26
- hypersurface, 21, 29
- imperfect model, 11
- imprecise relationship, 8
- in-degree, 17
- incomplete model, 11
- induction, 18
- inter-class distance, 36
- inter-relationship, 7

- logical, 10
- interactive approach, 37
- intra-class distance, 36
- iterative refinement, 37
- Jordan, 38, 41
- KATE, 38
- Kauffman, 37, 41
- Kim, 38, 41
- Knowledge Discovery and Nuggets, 38, 41
- Kononenko, 38, 41
- labeled sample, 12
- lattice, 26
- leaf node, 19
- Li, 39, 42
- linear
 - classifier, 14
 - discriminant, 14
 - function, 30, 37
- link, 16
- logical inter-relationship, 10
- loss, 7
- machine learning, 37
- major axis, 23
- Manago, 38, 42
- mathematical space, 7, 12
 - circle, 23
 - ellipse, 23
 - hyperplane, 26
 - hypersurface, 21, 29
 - lattice, 26
 - sublattice, 26
 - surface, 21
- matrix
 - pseudo-inverse, 33
- mean, 13
- measurement
 - feature, 7
 - space, 12
- medical diagnosis, 39
- meteorology, 9, 39
- minor axis, 23
- model, 7
 - complete, 8
 - deterministic, 8
 - exact, 8
 - imperfect, 11
 - incomplete, 11
 - parameter, 8
 - precise, 8
 - representation, 7
 - statistical, 8, 14
- Moret, 37, 42
- multistage classification, 9
- multivariate space, 10
- Murthy, 29, 37, 42
- nearly optimal decision, 8
- neural networks, 37
- node, 15
 - degree, 16
- nominal, 9
- non-leaf node, 20

- nonlinear, 14
 - transformation, 15, 24
- object, 14
- optimal
 - decision, 7
- optimal decision, 7, 8
- ordinal, 9
- out-degree, 17
- over-fit, 31
- parameter, 13
 - estimation, 37
- parametric
 - approach, 14
- parent node, 19
- partial decision, 12
- path, 17
- pattern
 - class, 14
 - classification, 9, 14
 - clustering, 34
 - hierarchical, 9
 - multistage, 9
 - statistical, 9
 - unsupervised, 33
 - classifier
 - design, 14
 - support vector machine, 32
- sample
 - training, 11
 - unlabeled, 33
- pattern classifier, 21
- Payne, 29, 38, 42
- penalty, 7
- physical
 - variable, 9
- physical attribute, 7
- pollution monitoring, 39
- precise model, 8
- principle of optimality, 27
- probability
 - a posteriori*, 10, 14, 25
 - a priori*, 10, 12–14
 - density, 13, 14
 - class conditional, 12
 - function, 10
 - Gaussian, 13
 - parameter, 13
 - distribution, 14
- pruning, 32
- pseudo-inverse, 33
- Quinlan, 37, 38, 42
- Rastogi, 38, 42
- real line, 10
- relationship
 - imprecise, 8
 - inter-, 7
- remote sensing, 39
- risk, 7
- rooted and directed acyclic graph, 20
- rooted and directed tree, 19

- rooted tree, 18
- Russell, 37, 42
- Safavian, 38, 42
- sample
 - labeled, 12
 - pattern
 - training, 11
- scatter plot, 31
- science
 - earth, 7
 - environmental, 7
 - meteorology, 9
- sibling node, 19
- software package
 - commercial, 38
- space
 - mathematical, 7
 - multivariate, 10
- statistical
 - decision-making, 7
 - model, 8, 14
 - pattern classification, 9
 - training data, 11
- sublattice, 26
- support vector machine, 32
- surface, 21
- theorem
 - Bayes, 10
- threshold comparison, 22
- top-down clustering, 36
- training
 - pattern sample, 11
- training data, 12
 - statistical, 11
- transformation
 - affine, 24
 - nonlinear, 15, 24
- traversal, 19
- traverse, 17
- tree, 18
 - binary, 20, 22, 29
 - child node, 19
 - decision, 8, 12, 15, 20
 - binary, 21
 - height, 20
 - leaf, 19
 - non-leaf node, 20
 - parent node, 19
 - pruning, 32
 - rooted, 18, 19
 - sibling node, 19
- unclassified pattern sample, 33
- undirected graph, 16
- unlabeled pattern sample, 33
- unsupervised classification, 33
- variable, 7
 - continuous, 12
 - discrete, 12
 - feature
 - composite, 23

72

physical, 9

transformation, 9

variance, 13

vector

data, 9, 20

vertex, 15

Wilson, 27, 42