Random Graphs for Statistical Pattern Recognition

David J. Marchette
Naval Surface Warfare Center
Dahlgren, Virginia

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Marchette, David J.
Random graphs for statistical pattern recognition / David J. Marchette.
p. cm. — (Wiley series in probability and statistics)
Includes bibliographical references and index.
ISBN 0-471-22176-7 (cloth)
1. Random graphs. 2. Pattern perception—Statistical methods. 3. Pattern recognition systems. I. Title. II. Series.

QA166.17.M37 2004
511'.5—dc22 2003063762

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

Contents

Preface

I predict in the not too distant future, people in academic life are going to define themselves not by one specialty area, but by two sub-specialties that belong to two rather different main specialties. This means that we'll have a web of interests, in which each person will serve as a bridge between different parts of the overall structure. You can see that this is much better than having a tree hierarchy that branches out further and further, with nobody able to talk to the people on the other sub-branches. We'll have people that each belong to two areas, in two different parts of the overall structure. Then we'll be able to have some hope of coping with new knowledge as it comes along.

Donald E. Knuth, *Things a Computer Scientist Rarely Talks About.*

What is *Random Graphs for Statistical Pattern Recognition*? Both topics are extensive: both topics have whole books devoted solely to them, for instance [23] and [103] for random graphs, and [65], [46], and [51], for statistical pattern recognition. I consider the topic of this book to be the intersection of the two topics: *random graphs* $\cap$ *statistical pattern recognition.* In this way I agree with Knuth that disparate fields of knowledge can be used to shed light on each other, and to bridge the gap between them.

I make no attempt to be exhaustive, even in this restricted topic. For example, I see (at least) two sub-categorizations of the intersection of random graphs and statistical pattern recognition: random graph methodologies applied to statistical pattern recognition problems, and the study of random graphs that arise from consideration of statistical pattern recognition problems. It is (primarily) the former sub-category which I address in this book (although I will, on occasion, digress into the latter).

In particular, I will consider almost exclusively **data** random graphs, wherein the vertex set is associated with some collection of random variables (a data set) $\{X_i\}_{i \in \mathcal{I}}$ for some (finite) index set $\mathcal{I}$. The (random) edge set is then determined by the random variables X_i. The distribution of the random graph is induced by the distribution of the random variables. An instantiation of the random graph is defined by the observed data. The X_i are the (class-labeled or unlabeled) observations upon which I wish to perform some type of statistical pattern recognition — classification or clustering.

In addition, our "final result" is most often based on some measure of the performance of a classification or clustering algorithm (which employs some random graph methodology) — for example, probability of misclassification — as opposed to, say, a theorem about some (random) graph invariant. (The latter is of interest only insofar as it aids us on our way toward the former.)

Thus I am interested in exploring the application of random graphs to pattern recognition problems. There are a number of ways that random graphs naturally occur in the analysis of data. I will investigate these graphs, with an eye toward increasing the tools available to the statistician.

It is hoped that by bringing these two disciplines together in this fashion they will both be enhanced. The statistical pattern recognition community will find new tools, plus some old tools in a slightly different guise. The random graph community may find some interesting applications, and some new ways of constructing random graphs. I hope this book will help bridge the gap between the two disciplines.

Another perspective is that this book is about the intersection between statistical pattern recognition and computational geometry. I do not attempt to consider all the ways that computational geometry could arise in pattern recognition tasks, but rather consider that subset of the field that is concerned with or results in random graphs.

The graphs that I am most interested in are proximity graphs and neighborhood graphs. Roughly speaking, proximity graphs are defined in terms of distances between points; that is, there is an edge between two points if they are proximate in some predefined sense. Neighborhood graphs are similar, but defined in terms of properties of neighborhoods of points. These typically involve rules about intersections of proximity regions, rather than the regions themselves. For example, there will be an edge between two vertices if the intersection of their neighborhoods is empty, or contains only the two points at issue. Although these make up the bulk of this work, we do not want to strictly limit ourselves to these graphs. Thus I will discuss several different graphs that arise in pattern recognition, although the main focus will be on proximity and neighborhood graphs.

There is a huge literature on a class of "random graphs for pattern recognition" dealing with graphical models, Bayesian networks, and the like. This book is not a book on graphical models. There are many good books on this subject, and the interested readers are encouraged to investigate them at their leisure. Classification trees can also be considered to be "random graphs for pattern recognition" and there are likewise many books that investigate these. Again, this is not the focus of this book.

This book is written at the level of an advanced undergraduate or first-year graduate course. I have attempted to include the important concepts from both graph theory and

pattern recognition, so, to some extent, the book is self-contained. It is assumed that most readers will have been exposed to basic probability and statistics at an advanced undergraduate level. A course in discrete mathematics would be useful, but may not be required for a mathematically sophisticated reader.

I have included problems throughout the text. The flavor of the problems tends to be experimental rather than theoretical, although I do sprinkle theoretical problems throughout. I suggest that students obtain practical experience with the ideas presented in the text through experimentation on real and simulated data. To this end, students will need to have access to some high-level programming language. I suggest the R language ([98] and [170]), available at cran.r-project.org In addition, the social network analysis contributed package (sna) has some useful functionality and is easily extensible to include the techniques discussed in this book. Delaunay triangularizations and Voronoi tessellation software (for $\mathbb{R}^2$) is available in the deldir contributed package. All simulations discussed in the book have been produced using R, as have all the figures presenting data and results.

References to web pages (as above) are problematic. The web is a very dynamic environment, and pages (or even servers) cannot be relied on to be available in the future. With that said, I will occasionally point out interesting web sites, with the understanding that their existence at the time of the reading is not guaranteed. In the event that the information is no longer available, the reader is encouraged to "google" the topic (if I may be allowed the use of a word that is only now making it into the mainstream), to find where it has moved, or to obtain more information.

I have worked in statistical pattern recognition and computational statistics for more than a decade, and have dabbled in random graphs (as they apply to problems in statistical pattern recognition) since visiting The Johns Hopkins University in 1999.

This book started as a collaboration between myself and Carey Priebe. Unfortunately, Carey was unable to devote the time to the project, and so I took it on myself. Carey contributed in many ways, however, including discussions, suggestions, and some work on early drafts of some of the chapters.

D. J. MARCHETTE

King George, VA, USA

Acknowledgments

The author is a student of Ed Wegman, and it is no exaggeration to say that this book would never have been possible without his help, support, guidance and friendship. This book came out of work done while I was spending a semester at The Johns Hopkins University working with Carey Priebe and John Wierman. Subsequently, they each spent a sabbatical with me at the Naval Surface Warfare Center. Carey worked on some early drafts of this book, and has been involved in much of the original work presented herein. In addition to Carey and John, this book owes itself to many people I have worked with over several years. I would particularly like to thank Jason DeVinney, Diego Socolinsky, Buck McMorris and Godfried Toussaint, for many discussions. Diego was kind enough to provide the images used in the face recognition section. Thanks to Jeff Solka for Figure 3.28 and for many discussions. I could not have done this without the support of Susan, and would also like to thank Glen Moore, for allowing me the flexibility to go where my research interests have guided me. This book is dedicated to my kids: Jeff, Katy and Steven.

D.J.M.

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

1

Preliminaries

The wealth of your practical experience with sane and interesting problems will give to mathematics a new direction and a new impetus.

Leopold Kronecker to Hermann von Helmholtz

Homo Sapiens is about pattern recognition, he said, both a gift and a trap.

William Gibson, *Pattern Recognition.*

This chapter will introduce the concepts of pattern recognition and random graphs that will form the basis for the rest of the book. Much of the notation used in the book, as well as many of the important terms, will be defined here. A reader familiar with these topics may skip ahead, using this section only as necessary to refer to unfamiliar definitions or notation.

We start with a section describing graphs, directed graphs, and random graphs. The concepts and notation introduced here will be used throughout the book without further comment. This is followed by a section introducing the basics of statistical pattern recognition, which serves to define the notation and set up the basic problems in pattern recognition. We end the chapter with a section describing some of the applications that will be used throughout the book. These and similar applications were the real driving force behind our foray into random graphs. As Kronecker suggested, these interesting problems have indeed led us to interesting mathematics.

1.1 GRAPHS AND DIGRAPHS

In simple terms, graphs are structures for the representation of relationships between objects. A graph consists of vertices (objects) and edges (binary relationships).

Graphs are used in many applications, including the analysis of networks; the study of the spread of a liquid through a porous medium; the analysis of social relationships; the design and analysis of algorithms; allocation of resources; and, the topic of this book, pattern recognition. See [59] and [79] for discussions of several different applications of graph theory.

This section will review the basic definitions and notation of graphs and digraphs (graphs with directed edges). This is necessarily a very brief introduction. The interested reader is encouraged to peruse one of the many good books on the subject, such as [34], [24], [23], and [103].

1.1.1 Graphs

The graph is a fundamental structure for analysis in a wide range of discrete mathematics. This section will introduce some of the basic concepts and notation that will be used throughout the book.

Definition 1.1. *A* **graph** G *is a pair* $G = (V, E)$ *for which the non-empty (usually finite)* **vertex** *set is* $V = V(G) = \{v_1, \ldots, v_n\}$ *and the* **edge** *set* $E = E(G)$ *is a collection of* **unordered** *pairs of vertices* (v_i, v_j).

Normally we disallow edges from a vertex to itself (loops); $(v, v) \notin E$ for any $v \in V$. We call $n = |V|$ the **order** of the graph and $|E|$ is called the **size** of the graph, where $|S|$ denotes the cardinality of the set S. If $E(G) = \emptyset$ ($|E(G)| = 0$) we call G the **empty graph**, and if all possible edges are present ($|E(G)| = \binom{n}{2}$) we call G the **complete graph** and denote it K_n. The **trivial graph** consists of a single vertex (and no edges).

We will often write vw for the edge (v, w). Two vertices v and w are said to be **adjacent** if $vw \in E(G)$. A **neighbor** of a vertex v is any vertex adjacent to v. The set of neighbors of v (also called the "(open) neighborhood of v") is denoted $N(v)$. For a set $S \subset V$, the neighborhood is defined as $N(S) = \cup_{s \in S} N(s)$. The closed neighborhood of v is denoted $\bar{N}(v) = N(v) \cup \{v\}$. Similarly, the closed neighborhood of a set $S \subset V$ is defined as $\bar{N}(S) = N(S) \cup S$.

A graph can be represented in a number of different ways. An equivalent representation is as an **adjacency matrix** A_G, which encodes the edges as entries in a matrix whose rows and columns correspond to the vertex set. That is, A_G is an $n \times n$ binary matrix with

$$a_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in E(G) \\ 0 & \text{else} \end{cases} \tag{1.1}$$

where n is the order of the graph G. The **augmented** adjacency matrix is $A_G + I_n$, where I_n is the $n \times n$ identity matrix.

The **degree** of a vertex v is the number of edges **incident** on the vertex; that is, $d(v) = |\{w | vw \in E(G)\}|$. A vertex of degree 0 is called an **isolated** vertex. For a set $S \subset V$, a vertex $v \in S$ is called an **isolate** of S if $N(v) \subset V \setminus S$. In particular, the isolated vertices are the isolates of V, since for such, we have $N(v) = \emptyset = V \setminus V$.

The minimum degree of the vertices of a graph G is denoted δ_G, and the maximum degree is denoted Δ_G.

A vertex is said to **cover** an edge if the edge is incident on the vertex. Similarly, an edge covers a vertex if the edge is incident on the vertex. A **vertex cover** (**edge cover**) is a set of vertices (edges) that cover all the edges (vertices) of the graph. The cardinality of a minimum vertex cover is denoted α_G, and is called the **vertex covering number** of G.

Definition 1.2. *A* **subgraph** *H of a graph G is a graph with $V(H) \subset V(G)$ and $E(H) \subset E(G)$. A subgraph H of G is an* **induced subgraph** *if for every $u, v \in V(H)$, $uv \in E(H) \iff uv \in E(G)$. A* **spanning** *subgraph of G is a subgraph which contains all the vertices of G.*

Definition 1.3. *A* **clique** *in a graph is a maximal complete subgraph (maximal in the sense that no other complete subgraph contains it).*

Definition 1.4. *A set $S \subset V(G)$ is a* **dominating set** *if every vertex in G is either an element of S or is adjacent to an element of S; that is, $\bar{N}(S) = V$. A* **minimal** *dominating set is a dominating set S for which no subset is itself a dominating set. A dominating set is* **minimum** *if it has the minimum cardinality for all dominating sets of G. The cardinality of a minimum dominating set is called the* **domination number** *and is denoted γ_G.*

It is obvious that a minimum dominating set is minimal. The converse is not true (see problem 1.11).

The concept of dominating sets, and the domination number, will play a large part in Chapters 4 and 5. In these chapters I will describe a method of classification and clustering which utilizes dominating sets to reduce the complexity of the classifier and select the number of clusters in a clustering algorithm.

Definition 1.5. *A set $S \subset V(G)$ is* **independent** *if there are no edges between elements of S, that is, $u, v \in S \implies uv \notin E$. Analogous to dominating sets, we have the concept of a maximum independent set: an independent set of vertices is* **maximum** *if it has the maximum cardinality for all independent sets of G. The cardinality of a maximum independent set is called the* **independence number** *and is denoted β_G.*

Note that by definition, every single-vertex set is independent, as is the set of isolated vertices.

Quite a lot is known about dominating sets and independent sets. A few representative theorems are reproduced below. See [88] and [87] for more.

Theorem 1.1 (Johnson). *Determining whether a graph has a dominating set of size $\leq k$ is NP-complete.*

Theorem 1.2 (Ore). *A dominating set S is a minimal dominating set if and only if for each vertex $v \in S$ one of the following holds:*

(a) v is an isolate of S.

(b) There is a vertex $u \in V - S$ *for which* $N(v) \cap S = \{u\}$.

Theorem 1.3. *If* G *is connected and* $\delta_G \geq 3$ *then* $\gamma_G \leq 3n/8$.

Theorem 1.4. $\gamma_G \leq \beta_G$.

See Problem 1.6.

Theorem 1.5. *If* G *is a graph of order* n *that has no isolated vertices, then* $\alpha_G + \beta_G = n$, *and* $\gamma_G \leq \alpha_G$.

Since many applications of graphs involve the concept of traversing the edges of the graph, we have a concept of a path in a graph. A **path** from vertex v_1 to vertex v_p is a set of edges $v_1v_2, v_2v_3, \ldots, v_{p-1}v_p$. This allows us to define the concept of a connected component of a graph:

Definition 1.6. *A* **connected component**, *or simply* **component**, *of a graph* G *is an induced subgraph* H *such that for every pair of vertices* $u, v \in V(H)$, *there is a path in* H *from* u *to* v, *and* H *is maximal with respect to this property.*

A **closed path** is a path that joins itself: $v_1v_2, v_2v_3, \ldots, v_{p-1}v_p, v_pv_1$. A **cycle** is a closed path in which all the vertices (except the first and last) are distinct. A graph is called **acyclic** if it contains no cycles.

A **tree** is a nonempty acyclic connected graph. An **end vertex**, or **leaf**, or **pendant** vertex is a vertex of degree 1. Generally for trees one uses the terminology "end" or "leaf" while for more general graphs one refers to the vertex as a "pendant" vertex. A **forest** is an acyclic graph. Thus the connected components of a forest are trees.

Theorem 1.6. *A graph* G *is a tree if and only if every pair of distinct vertices are connected by a unique path.*

Proof. Let u and v denote two distinct vertices. If they are connected by two distinct paths then these paths determine a cycle, and G is not a tree.

Let G be a graph where every pair of vertices is connected by a distinct path. G is clearly connected, and any cycle in G would define two paths between any pair of vertices in the cycle. Therefore, G is acyclic, and hence a tree. □

The **complement** of a graph $G = (V, E)$ is the graph $\bar{G} = (V, \bar{E})$ such that $vw \in \bar{E} \iff vw \notin E$.

Definition 1.7. *Two graphs are* **isomorphic** *if there is a bijection between their vertices which respects the edge set. Thus graphs* G_1 *and* G_2 *are isomorphic if there exists a bijection* $\phi : V(G_1) \to V(G_2)$, *such that* $vw \in E(G_1) \iff \phi(v)\phi(w) \in E(G_2)$.

When two graphs are isomorphic we will refer to them as the same graph. Sometimes, however, we wish to make a distinction between isomorphic graphs. Such **labeled** graphs are defined by associating a labeling of the vertices with the graph. In this case, we also require that the isomorphism in Definition 1.7 respect the labeling.

Definition 1.8. *Let $G = (V, E)$ be a graph, with vertex set $V = \{v_1, \ldots, v_n\}$. A* **labeling** *of the graph G is a bijection $l : \{1, \ldots, n\} \to V$.*

Thus a labeling is an association of a set of labels with the vertices of the graph. Generally, we will identify v_i with $l(i)$ and simply note whether the graph is to be considered a labeled graph or not. One can also define labellings on the edges of the graph as well as on the vertices. In this case, for digraphs (see Section 1.1.2) we have a label for each directed edge.

Problems

1.1 To what do the row and column sums of the adjacency matrix of a graph correspond?

1.2 Prove that in any nontrivial graph, there exist two vertices with the same degree.

1.3 (Euler) Prove that the sum of the degrees of the vertices of a graph equals twice the size of the graph.

1.4 Show that the number of vertices of odd degree is always even.

1.5 Prove Theorem 1.2.

1.6 Must a maximum independent set also be a dominating set? Prove, or give a counterexample. If true, then show that Theorem 1.4 follows. Otherwise, prove Theorem 1.4 another way.

1.7 Prove Theorem 1.5.

1.8 Prove that every connected graph of order $n \geq 2$ has a dominating set whose complement is also a dominating set.

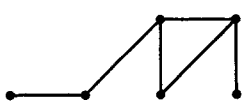
1.9 A graph is **self-complementary** if it is isomorphic to its complement. Prove that there is a graph of order n that is self-complementary if and only if $n \equiv 0$ or $1 \pmod 4$.

1.10 If H is an induced subgraph of G, must $\bar{H}$ be an induced subgraph of $\bar{G}$? Prove, or provide a counterexample.

1.11 Give an example of a graph G on n vertices with $\gamma = 1$ and a minimal dominating set of $n - 1$ elements.

1.12 What is γ for a path of length n?

1.13 Find all minimum dominating sets of the graph



1.14 Prove that every nontrivial tree has at least two end vertices.

1.1.2 Digraphs

The edges of a graph are in a sense bi-directional: any path in the graph can be traversed in any direction. Sometimes one wishes to impose a direction to edges. This will be important in Chapters 4 and 5. A graph with directed edges is called a **digraph**.

Definition 1.9. *A* **digraph** D *is a pair* $D = (V, A)$. *For a digraph, the edges (now called* **arcs***) are* **ordered** *pairs* (v_i, v_j) *with* $i \neq j$; $A \subset V \times V$. *We will write the arcs of* D *as* $A(D)$ *and the vertices* $V(D)$. *An arc from* u *to* v *is called* **bidirected** *if both* uv *and* vu *are in* A.

The definitions of **subdigraph** and **induced subdigraph** are as in the case of graphs (Definition 1.2) except with the constraint that the directions of the arcs be maintained.

The **in degree** of a vertex v is the number of arcs **incident to** v (from some vertex u to v). Similarly the **out degree** is the number of arcs **incident from** v (from v to some u). We denote these $d_{in}(v)$ and $d_{out}(v)$, respectively. The **degree** of a vertex v is $d(v) = d_{in}(v) + d_{out}(v)$.

Any digraph D has an underlying graph G_D corresponding to D with all arcs changed to undirected edges. Similarly, one has definitions of **directed paths**. We will call a subdigraph H of D a **connected component** if the underlying graph G_H is a connected component of G_D. One can define weakly and strongly connected components in digraphs, using directed paths, but we will not consider these. See [34] for details.

A digraph D on vertex set V is called **complete** if for every $u, v \in V$, either $uv \in A(D)$ or $vu \in A(D)$; that is, G is complete if $G_D = K_n$, where $n = |G|$. The **complete symmetric** digraph has both arcs for every pair of vertices, and is denoted $K_n^\star$. If D has the property that $uv \in A \implies vu \notin A$, then D is called **asymmetric**. A complete asymmetric digraph is called a **tournament**.

A **directed cycle**, or simply **cycle**, if the context is clear, is a path $v_1v_2, \ldots, v_{p-1}v_p$, v_pv_1 of arcs in $A(D)$ with distinct vertices v_i. The cycle is a **simple cycle** if none of the arcs are bidirected. If we eliminate all the bidirected arcs from $A(D)$, we obtain the digraph D^0. This digraph will be of interest in Section 4.1.1.

The **adjacency matrix** A_D of the digraph D of order n is the $n \times n$ binary matrix with

$$a_{ij} = \begin{cases} 1 & \text{if } v_iv_j \in A(D) \\ 0 & \text{else.} \end{cases} \tag{1.2}$$

This is exactly analogous to the adjacency matrix of a graph. However, note that unlike the case of graphs, the adjacency matrix of a digraph is not necessarily symmetric. The augmented adjacency matrix is defined analogously with that for graphs.

Fig. 1.1 Two distinct labeled digraphs. The unlabeled versions of these digraphs are isomorphic.

We have isomorphisms of digraphs in much the same way we had isomorphisms of graphs (see Definition 1.7). Similarly, we may wish to label the vertices, resulting in **labeled digraphs**. For example, the labeled graphs in Figure 1.1 are not isomorphic.

Problems

1.15 To what do the row and column sums of the adjacency matrix of a digraph correspond?

1.16 How many distinct labeled digraphs are there of order n?

1.17 Show that the sum of the in degrees of the vertices of a digraph is equal to the sum of out degrees, and these are equal to the number of arcs.

1.18 Show that for adjacency matrix A of a digraph, the ijth entry of A^n corresponds to the number of arc directed paths from v_i to v_j of length n.

1.19 A **source** in a directed acyclic graph is a vertex of in degree 0. Similarly, a **sink** is a vertex of out degree 0. Show that every directed acyclic graph has at least one source and at least one sink. Hence, conclude that if every vertex of a digraph D has nonzero in and out degrees, then D contains a cycle.

1.20 Digraphs permit definitions of dominating sets and independent sets just as with undirected graphs; give their formal definitions.

1.21 If $d_{in}(v) = 0$, how many dominating sets contain v?

1.1.3 Random Graphs

A **random graph** (or digraph) is a (measurable) function

$$\begin{aligned} \mathbb{G}: &\quad \Omega \to \{\text{the set of graphs}\} \\ &\quad \text{or} \\ \mathbb{D}: &\quad \Omega \to \{\text{the set of digraphs}\}. \end{aligned}$$

The sample space Ω will be considered only implicitly. If the range space is finite ($|V|$ is finite) then our random graph is a **graph-valued multinomial**. Occasionally the order of the graph itself is a random variable; for example, we may consider graphs

on N vertices where $N \sim$ Poisson, in which case the range space for $\mathbb{G}$ is countably infinite.

The two most common models are the **uniform** random graph and the **binomial** random graph. The uniform random graph is obtained by specifying a (finite) collection $\mathcal{G} = \{G_1, \cdots, G_m\}$ of graphs as the range space. Then $\mathbb{G}$ takes value $G \in \mathcal{G}$ with probability $1/m$. Thus a uniform random graph is defined by specifying the probability distribution on the range space. This idea is generalized by letting $\mathbb{G}$ take value $G \in \mathcal{G}$ with probability p_G ($\sum_{G \in \mathcal{G}} p_G = 1$) for some (not necessarily finite) collection of graphs $\mathcal{G}$. We call this method of specifying a random graph model **range space probability distribution specification.**

Binomial random graphs, on the other hand, are an example of **edge random graphs**; rather than specify a probability distribution on the range space directly, one may fix the order of a graph and specify a probability distribution on the **edges** (implicitly specifying a probability distribution on the range space). The binomial random graph (also referred to as the **Erdös-Rényi random graph**) is defined by first specifying the range space to be the set of graphs of order n and then specifying that each edge be present, independently, with probability $p \in [0, 1]$. Thus

$$P[\mathbb{G} = G] = p^{|E(G)|}(1-p)^{\binom{n}{2}-|E(G)|}$$

for all G such that $|V(G)| = n$. Generalizing edge random graphs so that edge (v_i, v_j) is in the graph with probability $p_{i,j}$, and relaxing the independence assumption on the edges, quickly leads to an edge random graph model for which the direct range space probability distribution specification is tricky, at best.

An alternative random graph model, the **vertex random graph** (or digraph), specifies randomness associated with the vertices. In the **random intersection graph** ([110]) model $\mathbb{G}(n, m, p)$, for instance, with each vertex $v_i \in \{v_1, \cdots, v_n\}$ we associate a random subset S_i of the integers $\{1, \cdots, m\}$, each chosen independently with probability $p \in [0, 1]$. For $i \neq j$, edge (v_i, v_j) is present if and only if $S_i \cap S_j \neq \emptyset$. In this way the random graph model is defined by specifying a probability distribution associated with the vertices. We will refer to this particular vertex random graph model as the **KSS** model.

We shall be concerned with the case in which the vertices V correspond to **data** (random variables) $\mathcal{X}_n = \{X_1, \cdots, X_n\}$ (we will always assume the X_i are independent and identically distributed, unless otherwise noted) and an edge (or arc) (X_i, X_j) is in the graph (or digraph) according to some function of the data; $(X_i, X_j) \in E$ (or A) if and only if $h(X_i, X_j; \{X_1, \cdots, X_n\}) = 1$. For example, the **nearest-neighbor digraph** (see Section 3.2) has an arc from each vertex X_u to its (assumed unique) nearest neighbor in $\mathcal{X}_n = \{X_1, \ldots, X_n\}$;

$$(u, v) \in A(G) \iff h(X_u, X_v; \mathcal{X}_n) \equiv I\{X_v \in \operatorname*{argmin}_{X_w \in \mathcal{X}_n} \rho(X_u, X_w)\} = 1$$

for some distance or dissimilarity function ρ. (Hereinafter, $I\{E\}$ denotes the indicator function of the event E.)

Since the data are random, the edges too are random. However, the probability distribution on the edges is induced by the probability distribution of the data $\mathcal{X}_n$. For

a **data random (di)graph**, obtained by specifying directly a probability distribution for $V \equiv \mathcal{X}_n = \{X_1, \cdots, X_n\}$, it may be difficult to determine the specification for the vertex random graph or the edge random graph, or the specification for the range space probability distribution. However, since statistical pattern recognition begins with a training data set $\mathcal{D}_n$ (classification) or a set of feature vectors $\mathcal{X}_n$ (clustering), random graph methodologies which are applicable to statistical pattern recognition problems or which arise from consideration of statistical pattern recognition problems are often data random (di)graphs.

Problems

1.22 Prove or disprove: the binomial random graph model with $p = \frac{1}{2}$ is equivalent to the uniform random graph model. That is, each graph in the range space of the binomial random graph model is equiprobable.

1.23 If a binomial random graph model has probability p, show that its complement is a binomial random graph model with probability $1 - p$.

1.24 Specify a data random graph by specifying the distribution function F of the data and an edge function h. Discuss the range space probability distribution specification and/or edge probability distribution specification for your data random graph.

1.25 Investigate the expected size of a Bernoulli random graph with edge probability p and order n through Monte Carlo simulation for various values of p and n. Hypothesize the formula for the expected size as a function of p and n. Can you prove that your formula is correct?

Application: User Profiling Consider the problem of determining whether web surfers group together into interest groups, or whether their surfing is random. Let's apply the KSS graph ideas to this problem.

Data for 28 users was collected over a period of five months. Each user was monitored to determine which web server the user was accessing. Note that this is at the web server level, not the level of individual web pages. The main reason for this is that the monitoring was done at the network level, where the data for individual web pages were not available.

For each week, we let the set S_i for user i be the set of web servers visited. The intersection graph defined by these sets will then be used to investigate user behavior. We want to find the edges that are "significant"; that is, those that have an unexpectedly large intersection, and which persist over time. We will use the KSS graph as our null hypothesis: there is a pool of web servers of size m from which users draw, and they draw each with probability p (equal for all users for all web servers; we will discuss more realistic extensions below).

The KSS model requires a set of m items from which each user draws, with probability p for each item. We thus need to estimate m and p for each week (note that we are allowing both m and p do change from week to week). Once we have

this, we can test whether the intersections of the servers for two users are too large to be likely under the null hypothesis. We can also ask whether other graph invariants, such as clique number, are "unusual" under the null hypothesis.

There are several methods for estimating m and p described in [124]. We will describe two of them here. The first is essentially a method of moments estimator, the second a maximum likelihood estimator.

In [110], it is shown that the expected size is

$$\binom{n}{2}(1-(1-p^2)^m),$$

where n is the order of the graph. Using this, [124] derives the estimator for m:

$$\arg\min\left(\sum |S_i|/(nm) - \sqrt{1-(1-d)^{\frac{1}{m}}}\right)^2,$$

where the minimization is over m and d is the density of the graph.

An alternative approach is to maximize the likelihood function. Using results from the field of animal abundance estimation (see [26]). Set

$$\begin{aligned} k_i &= |S_i| \\ M_i &= |\bigcup_{j=1}^{i} S_j| \\ u_i &= M_i - M_{i-1}, \end{aligned}$$

the likelihood is easily shown to be

$$L = \prod_{j=1}^{n} \binom{M_{j-1}}{k_j - u_j}\binom{m - M_{j-1}}{u_j} p^{k_j}(1-p)^{m-k_j}.$$

The idea is to process each user, in turn. The first user "captures" k_1 severs, which are marked and returned to the wild. The second user captures k_2 servers, and the number of previously marked servers is noted, the new ones are marked, and returned. This is repeated through the final user. This results in the likelihood above.

Using the above estimators, the estimates of m and p are obtained and those edges with an unusually large intersection (as measured by a Bonferroni correction applied to the hypergeometric distribution, as described in [124]) are retained. The edges that are thus marked the most times during the period considered are shown in Figure 1.2.

There are two triangles in Figure 1.2, both of which correspond to physicists working on fluid dynamics problems. The users in triangle BCD are also all of the same ethnic background. User E is the system administrator for one of the fluid dynamics projects, and user A is also a system administrator (for another group). User D is also a system administrator for his own small network. Thus there is some reason to believe that the relationships we have discovered are interesting.

The model considered is admittedly simplistic. It is reasonable to assume that users have their own values of p, for example, and there is evidence that this is the

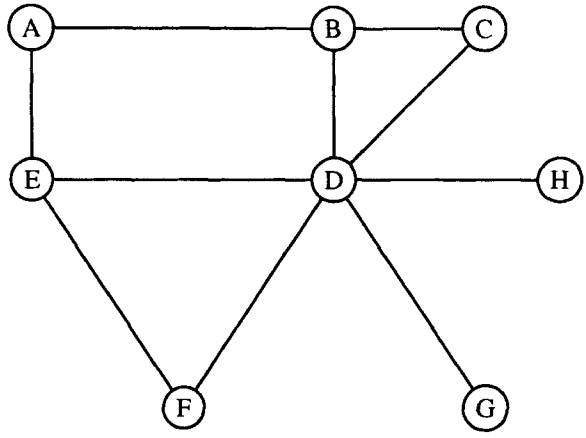


Fig. 1.2 A graph of the users with significantly large intersections. The edges for which the intersection size was statistically significant for 95% of the weeks are shown.

case (see [124] for more information on this). Further, it is reasonable to assume that all web sites are not equal. We cannot give each server its own individual p, since there would be no way to estimate this large a number of parameters. However, it might be possible to group the servers into those visited by (nearly) all users, those which are often visited, and those which are rarely visited. This would result in three p parameters per user. One could also eliminate servers that are common, and only consider the uncommon or rare servers. There are certainly many directions for investigation of this work.

Problems

1.26 Write down the likelihood equation for the case where each user has their own distinct probability.

1.2 STATISTICAL PATTERN RECOGNITION

Pattern recognition is the science (and art) of inferring the nature of an object from the "pattern" of observations made on the object. Thus, given an observation, say a set of measurements from an object, one goal of pattern recognition is to categorize the object into one of several predefined categories.

This basic idea is fundamental to much of science. One of the fundamental precepts of science is that generalizations can be made from observations. In order to make these generalizations, scientists group objects with similar properties, and try to determine the rules that apply to these groups.

Pattern recognition starts with observations and returns "classes". Statistical pattern recognition applies statistical methods to this problem. This section will describe the basic set up of pattern recognition, and give some examples of statistical pattern recognition methods.

Let $(X, Y) \sim F$. The **feature vector** X is a $\mathfrak{R}$-valued random variable. Usually $\mathfrak{R} = \mathbb{R}^d$ or some subset thereof. Occasionally $\mathfrak{R} = \mathcal{C}(\mathbb{R})^d$, where $\mathcal{C}(\mathbb{R})$ denotes the collection of continuous real-valued functions on $\mathbb{R}$, for example, feature observations may be multivariate time series. For **categorical** data $\mathfrak{R}$ is simply a set (unordered).

The **class label** Y is a $\{0, 1\}$- or $\{1, \cdots, J\}$-valued random variable. $J > 1$ is usually finite. (For the two-class case ($J = 2$) we usually consider Y to be $\{0, 1\}$-valued instead of $\{1, 2\}$-valued.) Y is used to indicate the category (or "class") to which X belongs.

The restriction of a unique value of Y for every X seems unnecessarily confining, but is appropriate for many applications. Many statistical pattern recognition systems actually produce a set of posterior probabilities associated with the categories, and thus can be used for problems in which an object may belong to more than one category. For cases in which the categories are hierarchical (the Linean classification system is an example of such), hierarchical pattern recognition methods exist.

We will partition statistical pattern recognition into two main categories: **supervised** and **unsupervised**. The distinguishing feature between these two categories is that for supervised pattern recognition the class labels Y are observed, while these class labels are unobserved in the unsupervised case. The supervised case is referred to as "classification", while the unsupervised case is called "clustering".

1.2.1 Classification

In the supervised case, **training data** are available. The training data set is given by $\mathcal{D}_n := \{(X_i, Y_i)\}_{i=1,\cdots,n}$. That is, we have available observation for which the true categorization is known. The goal is to develop a **classifier** g that will take an unlabeled feature vector X, with true but unobserved class label Y, and estimate its class label $\widehat{Y} = g(X)$ so that $\widehat{Y} = Y$ with high probability. Obviously, g should use the available training data; thus $g : \mathfrak{R} \times (\mathfrak{R} \times \{1, \cdots, J\})^n \to \{1, \cdots, J\}$. The use of training data to build the classifier is referred to as **training**.

In order for statistical pattern recognition methodologies to have any guarantee of success, we must assume that the training data are **representative**. Usually this means that $(X_i, Y_i) \overset{i.i.d.}{\sim} F$ and that the unlabeled feature vector X together with its (unobserved) class label Y are also distributed according to F and are independent of the training data set. Given a training data set $\mathcal{D}_n$, the **probability of misclassification** for classifier g is given by $L(g|\mathcal{D}_n) := P[g(X; \mathcal{D}_n) \neq Y|\mathcal{D}_n]$.

Definition 1.10. *The* **Bayes optimal probability of misclassification** *is given by*

$$L^\star = \inf_{g:\mathfrak{R}\to\{1,\cdots,J\}} P[g(X) \neq Y]; \tag{1.3}$$

the **Bayes rule** *is any map* $g \in \arg\inf_{g:\mathfrak{R}\to\{1,\cdots,J\}} P[g(X) \neq Y]$ *and is denoted* $g^\star$.

The goal of classification, then, is to devise a methodology for taking training data $\mathcal{D}_n$ and constructing a classifier g such that $L(g|\mathcal{D}_n)$ is as close to $L^\star$ as possible. In particular, we desire **consistency**: $L(g|\mathcal{D}_n) \to L^\star$ as $n \to \infty$ (in probability or with probability one).

We denote by F_j the **class-conditional distributions** of $X|Y=j$, and let us assume for discussion that the class-conditional probability density functions f_j exist. Given *i.i.d.* training data, then, the Bayes rule can be obtained from the f_j and the prior probabilities $\pi_j := P[Y = j]$ as the plug-in estimator; $g^\star(x) = \text{argmax}_j \pi_j f_j(x)$. See [46], for instance.

The class-conditional sample sizes $N_j := \sum_{i=1}^n I\{Y_i = j\}$ yield estimates of the priors; $\widehat{\pi}_j = N_j/n$. Any **density estimator** $\widehat{f}_j$ for the f_j yields a plug-in classification rule:

$$\widehat{g}(x) = \underset{j}{\text{argmax}}\, \widehat{\pi}_j(\mathcal{D}_n)\widehat{f}_j(x;\mathcal{D}_n). \tag{1.4}$$

We have $\widehat{\pi}_j \to \pi_j$ almost surely; if in addition a density estimator is employed for which $\widehat{f}_j \to f_j$ in L_1 or L_2 a.s., for instance, then $L(\widehat{g}|\mathcal{D}_n) \to L^\star$ a.s.

This assumes the training data were drawn with the correct priors, which is often not the case. When collecting data for a given pattern recognition task, one often collects (approximately) equal numbers of observations from each class. Alternatively, one class may be much more important, and one may collect a much larger number of examples of this one class. This is typical in medical problems. For example, in digital mammography it is important to have a representative sample of tumors, and so there may be far more such in the training sample than would occur in a similar sized random sample. In this case, there is an incentive to err in one direction: it may be much more important to detect a tumor than it is to correctly score a breast as healthy. Thus practical considerations may adjust the classifier by using weights on the different errors.

A popular method of density estimation that is often applied to classification is the mixture model. Modeling the density as a mixture of "kernel" densities (e.g., the Gaussian or Normal density) allows one to model arbitrarily complex densities. The mixture density is defined to be

$$f(x) = \sum_{k=1}^{K} p_k f_k(x;\theta_k), \tag{1.5}$$

with $p_k \geq 0$ and $\sum_k p_k = 1$, where the component densities have parameters θ_k which, along with the mixing coefficients p_k, must be estimated from the data.

The most common method of estimation in mixture models is the EM algorithm (see [133] and [134]). Data drawn from a mixture distribution can be considered a missing data problem: the missing data for each observation is the component from which that observation was drawn. The basic idea is to first estimate the missing data (the E step) and then, given this estimate, to estimate the parameters of the mixture (the M step). One then repeats this until some convergence criterion is satisfied.

A potentially difficult problem in mixture modeling is the selection of K. This can be thought of as simply a parameter determining the "complexity" of the model,

or one may believe that the mixture components f_k correspond to "latent subclasses" that were not specified in the training data. The discovery of these subclasses may result in interesting and important insight into the underlying process responsible for the data.

Density estimation is, however, quite expensive in high dimensions. Silverman ([189]) illustrates this with a table depicting the number of observations needed to obtain a given performance for a kernel estimator evaluated at a point. The kernel estimator is the nonparametric estimator

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{1.6}$$

where $h > 0$ is called the **bandwidth** and corresponds to a parameter controlling the smoothness of the estimator. Multivariate versions are constructed by using a Mahalanobis distance as the argument to the kernel K. Silverman shows that for a kernel estimator to obtain a fixed given performance at estimating the value of the density at the origin, 4 observations are needed for univariate observations, 19 for bivariate, and for ten dimensional data, 842,000 observations are required to achieve the same level of accuracy. This has been termed the "curse of dimensionality" ([19]), and we will consider this in more detail in Section 1.2.2.

Thus, for multivariate feature vectors in particular, there is quite a lot of interest in developing applicable classification methodologies. One approach involves preprocessing to yield a reduced dimensionality without seriously degrading classification performance. Thus, one might choose a projection $\mathbb{P} : \mathfrak{R} \to \mathbb{R}^{d'}$, where $d' = 1$ or 2, say, and consider classification, as above, using $\mathbb{P}(\mathcal{D}_n)$ as the transformed training data. See, for instance, principal component analysis, independent component analysis, Fisher's linear discriminant, and projection pursuit. These can be found in standard multivariate statistics texts such as [184], [127], [105], and pattern recognition texts such as [65], [51], and [85].

Instead of trying to estimate the probability densities, one might choose to estimate the decision region directly. The simplest decision region is a linear one, and several methods involve either estimating the best linear separator of the data, or extending to piecewise linear discriminators. See, for example, [190] or other pattern recognition books.

A nonparametric method that will play a large part in several chapters in this book is the nearest-neighbor classifier (and its extension, the k-nearest-neighbor classifier). First let us consider the nearest-neighbor classifier.

Definition 1.11. *Given a training set* $\mathcal{D}_n := \{(X_i, Y_i)\}_{i=1,\cdots,n}$, *the nearest-neighbor classifier* g_{nn} *is defined to be*

$$g_{nn}(x) = Y_{\underset{i}{\operatorname{argmin}}\{\rho(x, X_i)\}}, \tag{1.7}$$

where ρ *is a distance function.*

The idea is simple, yet powerful: choose the category associated with the nearest element of the training set. This classifier has been studied widely, due in part to its simplicity, and it is a standard against which new classifiers are often compared.

It is well known that the nearest-neighbor rule has asymptotic error bounded above by $2L^\star$ (see [46]). This means that if the classes are separable ($L^\star = 0$) then the nearest-neighbor classifier is consistent.

The k-nearest-neighbor classifier is an obvious extension. Instead of looking only at the closest point, look at the k closest elements of the training set. A simple vote is taken among the categories. (More complicated voting schemes have been investigated.)

Denoting the k-nearest-neighbor classifier as g_k, the following theorem shows the consistency of this classifier ([46]):

Theorem 1.7 (Stone, 1977). *If* $k \to \infty$ *and* $k/n \to 0$ *then*

$$EL(g_k) \to L^\star,$$

for all distributions.

Another (not necessarily disjoint) approach to dealing with high-dimensionality involves the application of random graphs. This is the main topic of subsequent chapters.

One of the issues in pattern recognition is the estimation of $\hat{L}$ for a given classifier g. One such method is the training/test set method; that is, one selects a training set from which to build the classifier, and an independent test data set (for which the class labels are also known) upon which to evaluate the classifier. One problem with this approach is that it requires the collection of additional data beyond that which is used to build the classifier. One might want to use all the data, with the assumption that this will be a better classifier. Often, it is expensive to collect data, in either time or resources, and so one wishes to make the best use of the data possible.

The method in which one uses all the data to build the classifier and then uses the same data to test the classifier is called **resubstitution**. This is obviously a biased (optimistic) estimate of the error. However, it may be appropriate in some cases, for example, to evaluate models with a small number of parameters when a large amount of data is available, and thus there is reason to believe the bias is small.

An improvement on the resubstitution method, with some of the flavor of the training/test method, is **leave m-out cross-validation**. In this, m observations are withheld from training and subsequently used to test the resultant classifier. This is repeated with the next m observations, until all observations have been in a test set (each observation is used in only one test set). If $m = 1$, this is simply referred to as cross-validation.

Problems

1.27 Implement a nearest-neighbor classifier. Note that a simple change to the code allows one to compute the cross-validated error in essentially the same time required to compute the resubstitution error.

1.28 What is the resubstitution error for a nearest-neighbor classifier? Does your answer depend on how the data are distributed?

1.29 Consider the two-class problem where each class is distributed as a multivariate Normal with (equal) covariances Σ. Derive the discriminant function and show that it is the equation of a hyperplane.

1.30 Construct a data set for which the cross-validated error of the nearest-neighbor classifier is 1.

1.2.2 Curse of Dimensionality

We saw above that the number of observations needed for density estimation increases rapidly in the dimensionality of the data. There are several consequences of this phenomenon that we consider here.

One illustration of this fact is given by [203] (reproduced in [101]). Consider the problem of classification in the two class case, where the classes are each normally distributed with (known) covariance matrices equal to the identity matrix. For $i = 1, 2$ the means are given to be

$$m_i = (-1)^i \left(1, \frac{1}{\sqrt{2}}, \ldots, \frac{1}{\sqrt{d}}\right). \tag{1.8}$$

Note that as d increases, the additional discrimination power added by the new variates decreases.

In the case where the means are known, [203] showed that as $d \to \infty$, the misclassification error $L^\star \to 0$. Thus perfect classification is obtained in the limit.

Surprisingly, in the case where the means are unknown, the plug-in estimator results in an error that goes to $\frac{1}{2}$ as d goes to infinity. Thus, we have an example wherein adding features that have, individually, discriminatory information nonetheless degrades the performance of the classifier, to the point that in the limit one has a classifier that is no better than a coin flip, or guess. We conclude that adding features — even features with value for discrimination — is not always a good idea.

There are a number of other ways to see the curse of dimensionality at work. David Scott ([183] (pp. 29–31)) illustrates this with a number of examples. He shows that as the dimensionality increases, the volume of a hypercube concentrates in the corners, the volume of a sphere concentrates near the boundary. See Problems 1.35–1.38.

An amusing discussion of this effect is given in [109]. Imagine a d-dimensional cubic prison holding a similarly d-dimensional prisoner. In each corner of the hypercube is a guard, who is spherical, and as large as possible, conditional on the guard remaining inside the jail, and with the further condition that no guards intersect inside their boundaries. The prisoner is in the center of the cube, and is also spherical, as large as possible conditional on intersecting the guards only on their boundary. For $d = 2$, the prisoner is held securely (see Figure 1.3). When $d = 3$, the prisoner can see light out between the guards. For $d = 10$, some of the prisoner is actually outside the jail, and as $d \to \infty$ the ratio of the volume of the prisoner to that of the jail increases without bound. This means, among other things, that eventually, nearly all of the prisoner has escaped!

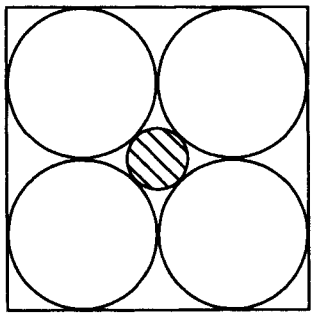


Fig. 1.3 A spherical prisoner guarded by four spherical guards inside a square. As the dimensionality of the prison, guards and prisoner increases, very strange things happen.

Due to the curse of dimensionality, many approaches have been developed for reducing the dimensionality of high-dimensional data. These include principal components, in which the data are projected onto a small subset of the major axes; Fisher's linear discriminant, which is the classification analog of principal components, where the projection is onto the axes that produce the largest linear class separation; projection pursuit, in which projections are searched for using a criterion of "goodness", such as non-normality; and multidimensional scaling, where a projection is found that retains (as much as possible) the original inter-point distances. The interested reader is directed to general books on pattern recognition, such as [65], [177], [51] and [85]; books on multivariate analysis such as [184] or [127]; or specific books on the subjects, such as [99], [107] for principal components, or [27] for multidimensional scaling.

Problems

1.31 Show that $L^\star \to 0$ as $d \to \infty$ for the problem described in Equation (1.8) with known means.

1.32 Assuming we estimate $m = m_0 - m_1$ in Equation (1.8) using the maximum likelihood estimator, show that the classifier error, for n (fixed) training observations, goes to $\frac{1}{2}$ as $d \to \infty$.

1.33 Illustrate the results of Trunk via simulation. Run a series of Monte Carlo replications for various values of d and n and estimate the errors for the two cases. Plot these estimated errors for the two cases as a function of d. Discuss the pattern(s) you observe.

1.34 Let $m_0 = (0, 0, \ldots, 0)$ and $m_1(1, 0, \ldots, 0)$. Find $L^\star$ as a function of dimension d. Repeat the calculation in Problem 1.32. What does the addition of the "noise" variables do to the classifier error?

1.35 Consider a hypersphere inscribed in a hypercube. Show that the ratio of the volume of the hypersphere to that of the hypercube goes to zero as the dimensionality $d \to \infty$.

1.36 Show that the volume of the annulus between a hypersphere of radius r and one of radius $r - \epsilon$ goes to 1 as $d \to \infty$ for every $0 < \epsilon < 1$.

1.37 Consider the spherical multivariate Gaussian density centered at the origin. Show that the probability mass of a sphere centered at the origin of fixed radius $0 < r < \infty$ goes to 0 as $d \to \infty$. Thus conclude that "all the data is in the tails" for high-dimensional normal data.

1.38 Consider the hypercube $[-1, 1]^d$, and let v be any of the 2^d vectors from the origin to one of the corners of the hypercube. Show that the angle between any diagonal v and any coordinate axis e goes to zero. Thus any canonical projection to two dimensions (for the purpose of producing a scatter plot of the observations, for example) results in plotting the observations near a diagonal essentially onto the origin.

1.39 Work out the details of the "prisoner" story of the text. Consider the hypercube $[-1, 1]^d$ with hyperspheres of radius 1 at each corner. Consider the sphere in the center of the hypercube which kisses the corner spheres. Show that for $d > 4$ the center sphere has radius greater than 1. Thus the "interior" sphere contains points outside the cube. Show that the ratio of the volume of the prisoner to that of the jail increases without bound.

1.40 *How is a cube like a porcupine?* For a d-dimensional unit hypercube, compute the length of a diagonal. Can you answer the question?

1.2.3 Clustering

In the unsupervised case, we have available to us feature vectors $\mathcal{X}_n := \{X_1, \cdots, X_n\}$. The goal is to **cluster** these data in such a way as to provide clusters $C_k \subset \mathcal{X}_n$, $k = 1, \cdots, K$ which correspond to some underlying (interesting or useful) unobserved class labels. A fundamental difficulty in clustering is determining K, the number of clusters. Once K is determined, one proceeds to group the observations. Note that it is not always desirable to produce disjoint groups. Usually it is desirable to have each $X_i \in \mathcal{X}_n$ placed in (at least) one of the clusters $C_1, \cdots, C_K$. If not, we may think of a $K + 1^{st}$ cluster — none-of-the-above — into which all unallocated observations are placed; $C_{K+1} := \mathcal{X}_n \setminus (\cup_{k=1}^{K} C_k)$.

There are a total of K^n possible clusterings of n observations into K clusters; even given a criterion function for ranking clusterings, it is obviously desirable to approach the problem from the standpoint of some surrogate criterion applied to $\mathcal{X}_n$.

One may approach clustering from a density estimation viewpoint. For instance, a common approach is to model the density as a mixture of K components (again, choosing K is can be difficult) and then use these components to determine clusters.

A related method is k-means clustering. This is essentially the mixture approach, recast into a clustering algorithm. The idea is to cluster the data into clusters centered on k centers. The centers are initialized arbitrarily, and points are assigned to the cluster associated with their closest center. The centers are then recomputed using the assigned points, and this continues until convergence. The k-means algorithm is given in Algorithm 1.1. Variations on the algorithm replace the mean with other measures of location, such as median or trimmed mean.

input : $\{x_1, \ldots, x_n\}, k$
output : the centers for the k clusters, $\{c_1, \ldots, c_k\}$
initialize: choose initial centers arbitrarily
Do
  for $i = 1, \ldots, k$ **do**
    $S_i = \{x_l | \operatorname{argmin}_j(\rho(x_l, c_j) = i\}$;
  end
  for $i = 1, \ldots, k$ **do**
    $c_i = mean(S_i)$;
  end
Until *there is no change in the clusters*;

Algorithm 1.1: k-means clustering algorithm

The k-means algorithm, unlike the EM algorithm for mixture models, makes a "hard decision" about cluster assignment for each point. There are various so-called "fuzzy" versions of the k-means algorithm that relax this condition, essentially implementing a version of the EM algorithm.

Besides the problem of selecting the value of K, the k-means algorithm, like the EM method in mixture models, suffers from sensitivity to the initial cluster centers. For this reason, some practitioners advise trying several initializations, with various methods for selecting or combining the resultant clusters. Others suggest various methods for selection of initial centers. The interested reader is directed to the appropriate literature, some of which is discussed at the end of this chapter.

Hierarchical clustering procedures produce a (data random) **dendrogram**, or binary tree $\mathbb{T}(\mathcal{X}_n)$. The leaves of this tree are the (random) observations $X_1, \cdots, X_n$. Hierarchical agglomerative clustering of $\mathcal{X}_n$ involves $n-1$ cluster merges. To begin, each observation is considered to constitute a cluster of its own. At each stage, the two "closest" clusters are merged into one.

This requires a definition of "closest" that can be applied to sets. Several methods have been proposed, and each is suited for the discovery of a different kind of clustering.

For pairs of observations, "closest" is defined in terms of a distance or dissimilarity measure. A dissimilarity d only requires that $d(x,y) > 0$, $d(x,y) = d(y,x)$ and $d(x,x) = 0$. Note that the triangle inequality is not assumed, and further that it

is possible to have $d(x,y) = 0$ when $x \neq y$. Hierarchical clustering usually uses distances to define "closest", however we will see situations in later chapters where a dissimilarity measure is more natural.

For **complete linkage** clustering the distance between two clusters is given by the maximum distance between pairs of points:

$$\rho_{cl}(C_i, C_j) := \max_{\substack{X_i \in C_i \\ X_j \in C_j}} \rho(X_i, X_j),$$

while for **single linkage** clustering we use the distance between the two closest points:

$$\rho_{cl}(C_i, C_j) := \min_{\substack{X_i \in C_i \\ X_j \in C_j}} \rho(X_i, X_j),$$

where $\rho : \Re \times \Re \to \mathbb{R}_+$ is a distance (or dissimilarity) function. (Ties are broken arbitrarily or by randomization.) Different choices of ρ result in different characteristics of the clustering algorithm. Other distances include the distance between the means or medians of the sets, again using a distance ρ, with various definitions of multivariate median possible.

At each stage, the distance between the clusters that are merged is larger than (or equal to, but under continuous models equality occurs with probability zero) that of the stage before. Thus we can associate with each merge (each non-leaf vertex in the dendrogram) a height attribute, defined to be the distance between clusters whose merge is represented by that vertex.

Figure 1.4 presents an example of hierarchical clustering. The scatter plot on the left depicts nine observations. The associated dendrogram is shown on the right. We see that the first merge (the merge with the smallest height attribute in the dendrogram) is associated with the two closest points (1 & 6) in the scatter plot. Moving up the dendrogram, we see that the final merge (with a height attribute greater than 0.8) combines the cluster consisting of the three observations in the lower-right of the scatter plot with the cluster consisting of the six observations in the upper-left of the scatter plot.

The cluster labels $\widehat{Y}_i$ are obtained by cutting the dendrogram at height h. This yields $K(h)$ subtrees. All leaves in subtree $\mathbb{T}_k$, $k = 1, \cdots, K(h)$, defined by cutting $\mathbb{T}(\mathcal{X}_n)$ at height h are assigned to cluster k; that is, $\widehat{Y}_i = k$ if observation X_i is associated with a leaf in subtree $\mathbb{T}_k$. In our case, with K given, any h such that cutting at height h yields K subtrees is satisfactory.

Returning to Figure 1.4, the dashed horizontal line on the dendrogram represents a cut such that $K = 2$ subtrees result. This completes the clustering, and we see that the two resultant clusters agree with our intuition from visual inspection of the scatter plot.

Different link functions result in different clusters. Results of hierarchical clustering are presented in Figure 1.5. The figure shows the results of hierarchical clustering using different link functions: complete linkage (top left), average distance (top right) centroid distance (bottom left) and single linkage (bottom right). It is clear from these pictures that the different link functions have very different properties and result in very different clusterings.

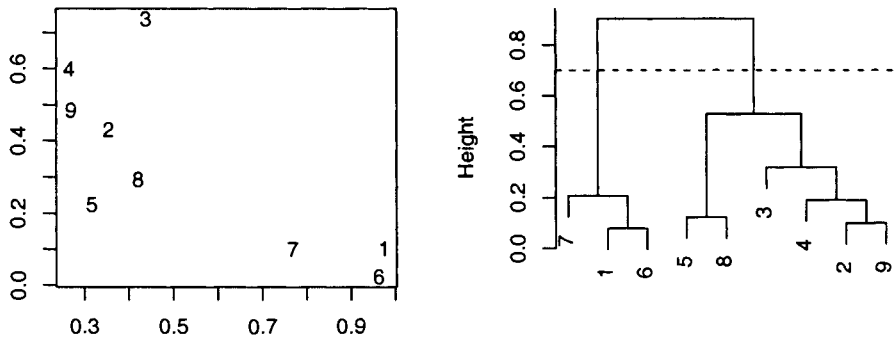


Fig. 1.4 Example of hierarchical clustering.

This is further illustrated in Figure 1.6. Here we cut the tree at $k = 3$ in an attempt to discover the three clusters in the data. The first three methods — complete linkage (upper left), average distance (upper right), and centroid (lower left) — do a good job of finding the clusters, and return nearly identical results. The single linkage does a poor job, placing the one outlier in the middle of the graph into its own cluster. In other simulations having fewer outliers the single linkage cluster does find the three clusters, but the algorithm is not designed for this kind of clustering. As seen in Figure 1.5, single linkage clustering is designed to find clusters made of paths of short segments rather than round clumps such as in Figure 1.6.

A random variable of interest is the invariant depth($\mathbb{T}(\mathcal{X}_n)$). This gives a measure of the computational complexity of the clustering; the maximum number of decisions that must be made in assigning a point to a cluster. In general, $\log_2 n \leq$ depth($\mathbb{T}(\mathcal{X}_n)$) $\leq n - 1$. Random variables of great import are the cluster sizes $[N_1, \cdots, N_K]'$.

When K is the number of classes, and we have labeled data to use for evaluation, we can evaluate clustering in a straightforward manner, analogous to $\widehat{L}$ in the classification setting. Consider the (unobserved, latent) class labels to be $Y_i, i = 1, \cdots, n$. Let

$$\tilde{L} := \min_{\sigma(1),\ldots,\sigma(K)} \frac{1}{n} \sum_{i=1}^{n} I\{\sigma(\widehat{Y}_i) \neq Y_i\},$$

where the minimum is taken over permutations $\sigma(1), \cdots, \sigma(K)$ of $1, \cdots, K$. Note, however, that for this evaluation one is concerned with the labeled tree, as opposed to being concerned solely with graph properties that are independent of the labeling.

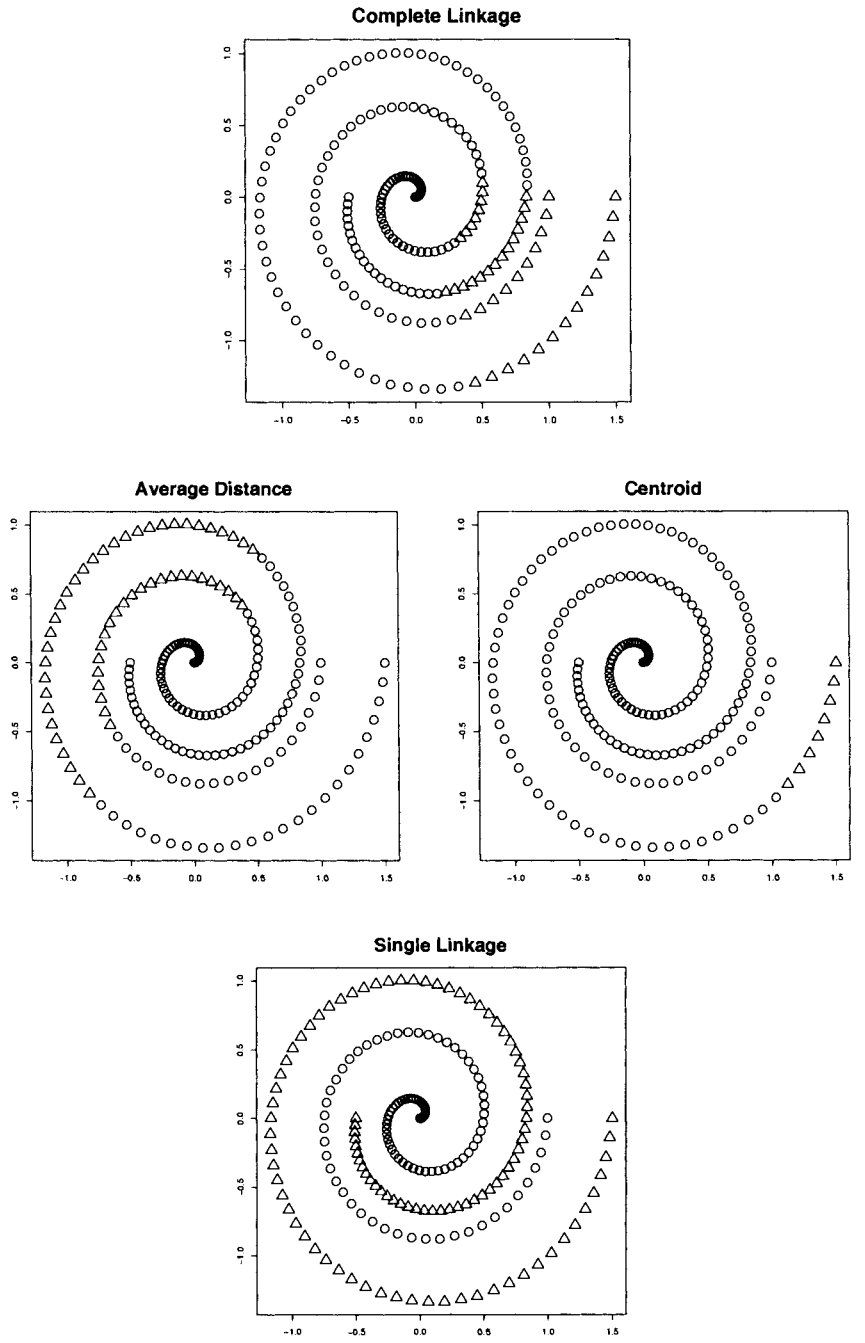


Fig. 1.5 Results of the hierarchical clustering techniques for four link functions. The tree is cut at $k = 2$ and the two clusters are indicated by the different plotting symbols.

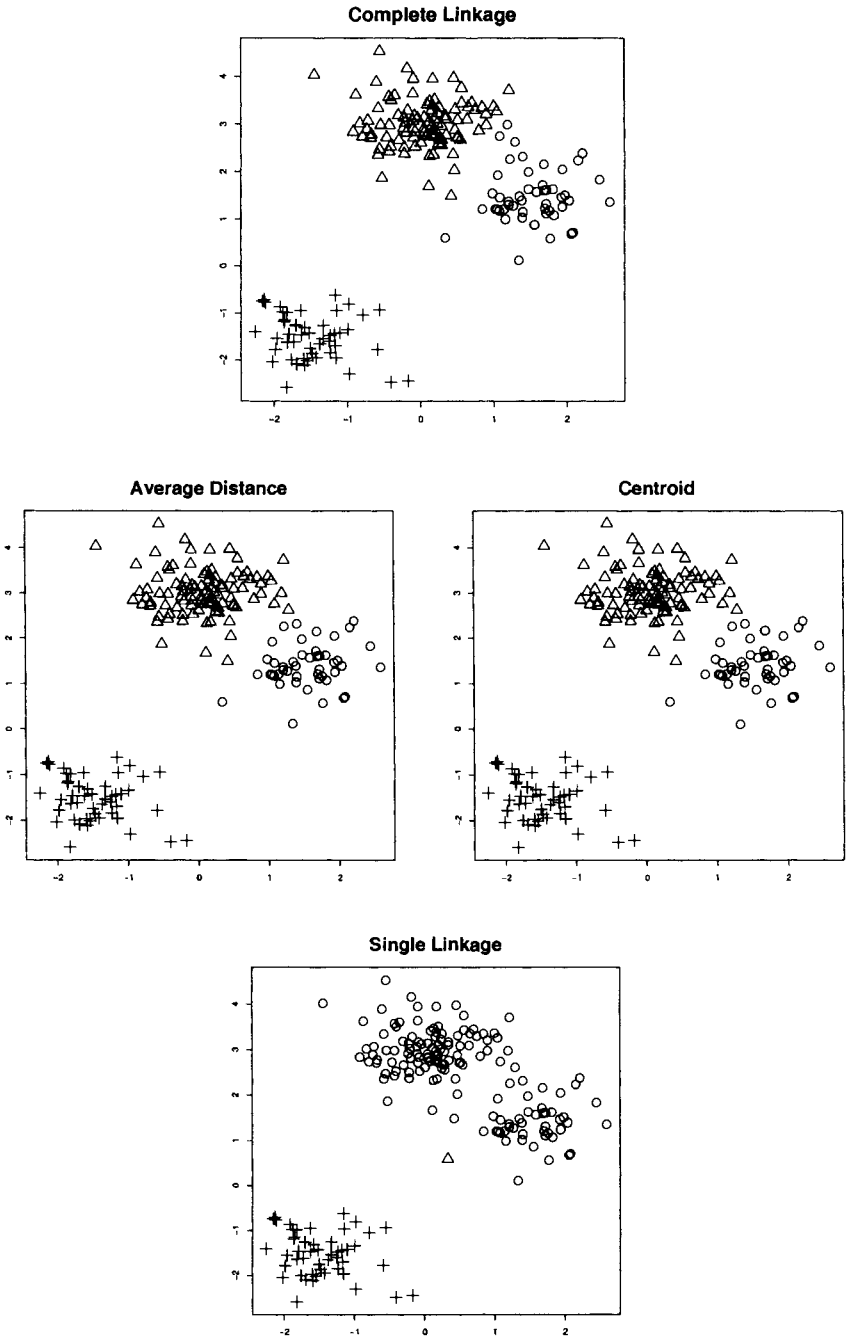
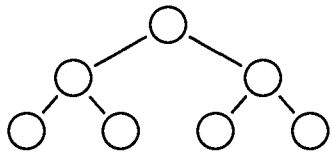
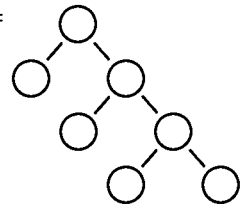


Fig. 1.6 Results of the hierarchical clustering techniques for four link functions. The tree is cut at $k = 3$ and the three clusters are indicated by the different plotting symbols.

Consider now the case of $n = 4$. In this case, there are only two possible (unlabeled) binary trees; $\mathbb{T}(\mathcal{X}_4) : \Omega \to \{T_1, T_2\}$, with

$T_1 :=$ 

and

$T_2 :=$ 

Thus the random graph $\mathbb{T}(\mathcal{X}_n)$ is completely specified by the scalar $p \in [0, 1]$, with $P[\mathbb{T}(\mathcal{X}_n) = T_1] = p$ and $P[\mathbb{T}(\mathcal{X}_n) = T_2] = 1 - p$. However, different distributions F on the data $\mathcal{X}_n$ produce different values of p. For any F, there is a value of $p(F)$ that fully describes $\mathbb{T}(\mathcal{X}_n)$. However, since these and other data random graphs are defined in terms of the data distribution F, it is of interest (but often challenging) to find a map taking $F \in \mathcal{F}$ to $p(F) \in [0, 1]$.

With $n = 4$ and $K = 2$, there are two possibilities for the random cluster sizes $[N_1, N_2]' = [N_1, n - N_1]', N_1 \in \{1, 2\}$. For a tree T and a cut-defining number of clusters K, denote the largest cluster size by $N_{max}(T, K)$;

$$N_{max}(T, K) := \max_{1 \le k \le K} N_k.$$

The random variable $N_{max}(\mathbb{T}(\mathcal{X}_4), 2)$ takes its value in $\{2, 3\}$ and, in fact,

$$N_{max}(\mathbb{T}(\mathcal{X}_4), 2) \equiv \text{depth}(\mathbb{T}(\mathcal{X}_4)).$$

This example illustrates the importance of the labeled graphs. Consider, for instance, tree T_2. Cutting such that $K = 2$ yields two clusters; one containing a single observation and one containing three observations. Letting the leaves be labeled from left to right, we see that a labeling (X_1, X_2, X_3, X_4) puts observation X_1 alone in its own cluster and observations X_2, X_3, X_4 are grouped together in a cluster. If the leaves are labeled (X_4, X_3, X_2, X_1), the tree yields a different clustering all together.

Consider now the case in which

$$X_1, \cdots, X_4 \overset{i.i.d.}{\sim} F = \frac{1}{2}\mathcal{U}\text{niform}(0, 1) + \frac{1}{2}\mathcal{U}\text{niform}(2, 3), \tag{1.9}$$

cutting at $K = 2$. For this simple case, results are available. First, we see that $E_F[N_{max}] > 2$. In addition, much can be said about the performance of the clusterer

for this simple example. With the (unobserved) class label Y_i given by the mixture component from which observation X_i is drawn ($Y_i = 0$ if $X_i \in (0,1)$; $Y_i = 1$ if $X_i \in (2,3)$) we have $P_F[\tilde{L} = 0] = 0.875$.

Problems

1.41 Generate uniform data on the unit square. Using the k-means algorithm, cluster the data into two clusters using random initial centers. Repeat this several times (keeping the data set fixed, changing only the initial centers). Explain the results.

1.42 Generate data from the mixture

$$f(x) = \frac{1}{2}N((-2,0), I_2) + \frac{1}{2}N((2,0), I_2),$$

where I_2 is the two dimensional identity matrix. Run k-means with $k = 2$ on the data with a number of starting points. Estimate (through Monte Carlo simulations) the probability that a k-means algorithm will produce the "correct" split into two clusters for these data as a function of n, the number of observations.

1.43 Repeat the experiment of Problem 1.42 in higher dimensions. Fix n at 100, and consider the results as the dimension d increases.

1.44 For $n = 4$, how many unlabeled trees are there, if we do not restrict to binary trees? Draw them. Does extending to general trees change the number of ways we can cluster the data?

1.45 For $n = 5$, how many unlabeled binary trees are there? Draw them.

1.46 Consider the data in the following table:

(0.27, 0.30)	(0.57, 0.97)	(0.60, 0.58)	(0.97, 0.64)
(0.95, 0.26)	(0.88, 0.71)	(0.62, 0.92)	(0.20, 0.40)
(0.93, 1.00)	(0.45, 0.79)	(0.85, 0.44)	(0.59, 0.95)
(0.00, 0.86)	(0.97, 0.28)	(0.69, 0.71)	(0.34, 0.45)

How many clusters are there in these data?

1.47 Consider the following data:

$(-0.28, 0.96)$	$(0.86, 0.51)$	$(-0.29, -0.96)$	$(0.94, 0.35)$
$(0.62, -0.78)$	$(-0.89, -0.45)$	$(0.77, 0.63)$	$(0.36, -0.93)$
$(0.22, -0.98)$	$(0.80, -0.60)$	$(0.18, -0.98)$	$(0.70, 0.72)$
$(0.28, -0.96)$	$(-0.85, 0.53)$	$(-0.21, 0.98)$	$(-0.90, 0.43)$
$(-0.07, -1.00)$	$(-0.87, -0.49)$	$(-0.09, -1.00)$	$(-0.84, -0.54)$
$(0.11, -0.49)$	$(0.04, 0.50)$	$(0.14, -0.48)$	$(-0.45, -0.23)$
$(0.40, 0.30)$	$(0.17, 0.47)$	$(-0.50, 0.04)$	$(0.00, 0.50)$
$(-0.21, 0.45)$	$(0.05, 0.50)$	$(0.48, 0.14)$	$(-0.32, 0.38)$
$(-0.41, 0.29)$	$(-0.49, 0.07)$	$(-0.44, 0.24)$	$(0.44, -0.23)$
$(-0.20, 0.46)$	$(-0.50, -0.04)$	$(0.11, 0.49)$	$(0.29, -0.41)$

How many clusters are there in these data? Run k-means clustering with k set to your answer. Does it agree with your assessment? Try a hierarchical method, and discuss the results. Can you suggest a method that will result in clusters that will agree with your assessment?

1.48 Consider a mixture of two 2-dimensional normals with equal priors and identity covariances, and a distance between the means of 2. Using Monte Carlo replicates of size $n = 100$, estimate the expected depth of the dendrogram using complete linkage and single linkage. Repeat this experiment for several values of $|\mu_1 - \mu_2|$.

1.3 STATISTICAL ISSUES

Statistics provides tools for three main areas in data analysis. Statistical models are used extensively to provide an understanding of the basic process underlying the data, and to provide a predictor for future data. In statistical pattern recognition, this usually takes the form of either a classifier, which provides a predictor of class association for future data, or a clusterer, for clustering future data. These are also used simply to understand the data set at hand. This is particularly true for clustering, where often the purpose is simply to determine the groups in the data in order to understand some underlying phenomenon, or to suggest further areas for exploration.

Basic data summary, exploratory data analysis, and data visualization provide methods for getting an understanding of the data and the underlying process that produced it. These can also be used to understand the models fit to the data, and to get an appreciation of the goodness-of-fit of the models.

Finally, statistics provides a method for assessing the significance of the associations or models found, or the statistics computed. Without some measure of significance, it is very difficult to distinguish between a true discovery and a coincidence. As an illustration, consider the following experiment: plot data from a unimodal two dimensional distribution, with half the data colored red, the other half blue, in such a way that the colors do not overlap. Show this plot to a group of students (a class on statistical pattern recognition is an excellent venue for this experiment) and ask them

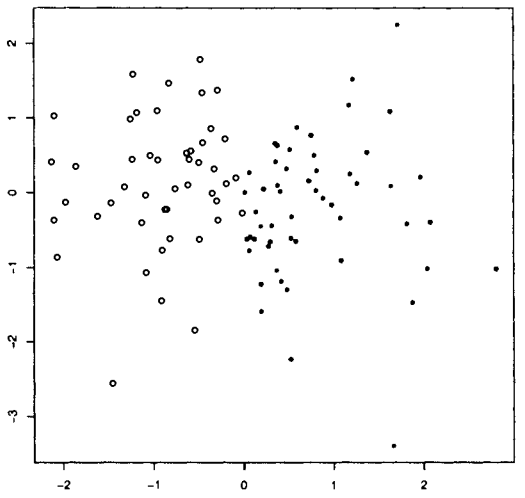


Fig. 1.7 Two dimensional standard normal points, plotted using two plotting symbols: those to the right of the origin are plotted using filled dots. How many groups do you think there are in the plot?

how many groups they see in the data. Overwhelmingly, it seems, they will say 2, even if you tell them to ignore the colors. An example of this is shown in Figure 1.7. The data were drawn from a standard normal, and points to the right of the origin are filled in. It should be noted that in this experiment the data were drawn a single time: no effort was made to find a data set that provided a good illustration of the effect.

The human visual system is very good at finding groups, and if you give it a little nudge, it is very easy to convince yourself of things that are not so. Science is full of examples of scientists discovering patterns that turn out not to be reproducible, which is one of the reasons it insists on reproducibility of experiments (as well as statistical significance tests, where appropriate). Statistical significance tests provide a method of checking to ensure that we are not fooled by spurious patterns.

There are three main ways to assess significance of some statistic. If the distribution of the null hypothesis is known, one can compare the value obtained on the data with the null distribution, to determine how "unusual" the observed value is if it did come from the null distribution. Often, the exact null distribution is unavailable, either because the theory is unknown or because the computations are impractical. In this case, it may be possible to compute asymptotic results, and compare the statistic with the asymptotic (large n) distribution. Finally, one can resort to Monte Carlo simulations. In the classic example of this, one simulates from the null distribution many times, each time computing the desired statistic, then compares the observed statistic with those produced by the simulation.

This latter is very popular in model building and pattern recognition problems, where it is often the case that neither exact nor asymptotic results are available. In particular, classification performance is often assessed through cross-validation, as described above, and models can be assessed via bootstrapping (see [132] for an example of this in normal mixture modeling). A related approach, bootstrap resampling, allows the estimation of variance and confidence intervals by repeated sampling from the emprical distribution function. We will not discuss this further, but there are many references to this approach in the statistics literature. Reference [54] is a good place to start.

As we will see, there are a number of random graphs that provide methods for modeling data, and exploring and visualizing data. They can provide statistics that can be used to test various hypotheses about the data. We will explore a number of these, with a bias more toward exploratory data analysis than theoretical results. The reader is encouraged to implement these ideas and try them on data.

Problems

1.49 Watch *Pink Oz* and report. To do this, play the movie *The Wizard of Oz* with the sound off. At the second lion roar, start Pink Floyd's *Dark Side of the Moon* (be sure to set it to repeat). Note the times when the music appears to track the action on the screen. Discuss this in light of the experiment of p. 26.

1.4 APPLICATIONS

We will consider a number of interesting applications throughout this book. In this section we describe some of them. Others will be introduced as they are used. These applications have in common the fact that they are very high-dimensional, complex data sets.

Much of the analysis in this book is done through the open-source statistics package R. R is a public-domain package similar in most respects to the commercial S/Splus package. It can be found at www.r-project.org and lib.stat.cmu.edu/R/CRAN/ Several books are available describing R and Splus, including [210] and [39], and [209].

1.4.1 Artificial Nose

This section presents a description of the Tufts artificial nose sensor data. These data were taken from a fiber-optic system constructed at Tufts University. The sensor consists of a 19-fiber-optic bundle, with a CCD camera to measure the intensities at each fiber. The fibers are chemically doped with a solvatochromic dye (see [217]), which results in a change in fluorescence intensity in response to interactions of the dye in each fiber with the chemical environment ([47]). An observation is obtained by passing an analyte (a single compound or a chemical mixture) over the fiber bundle

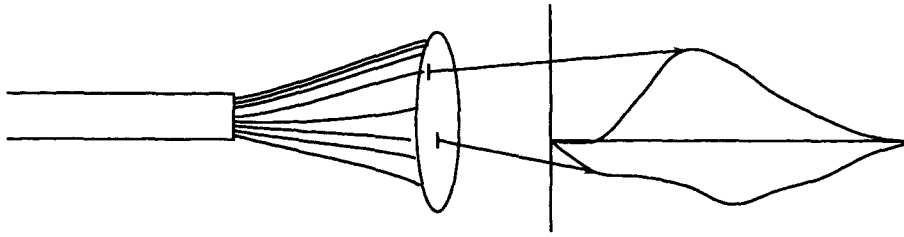


Fig. 1.8 A depiction of the Tufts artificial nose. Optical fibers are doped with a solvatochromic dye. Reaction of the polymer matrix with an analyte produces photons, which are sampled at two wavelengths to produce a response for each fiber, illustrated on the right. The two curves correspond to the intensities measured, as a function of time, from two distinct fibers.

in a four-second pulse, or "sniff." The information of interest is the change over time in emission fluorescence intensity of the dye molecules for each of the 19 fiber-optic sensors (see Figure 1.8).

The data set consists of recordings of sensor responses to various analytes at various concentrations. Each observation is a measurement of the fluorescence intensity response at each of two wavelengths ($\lambda_1 = 620$ nm and $\lambda_2 = 680$ nm) for each sensor in the 19-fiber bundle as a function of time. This results in time series (functional) observations $x_i^{\phi,\lambda}$ for fibers $\phi \in \{1,\ldots,19\}$ and wavelengths $\lambda \in \{1,2\}$. The index $i = 1,\ldots,n$ represents the observation number. While the process is naturally described as functional with t ranging over a 20-second interval, the data as collected are discrete with the 20 seconds recorded at 60 equally spaced time steps for each response. Construction of the database involves taking replicate observations for the various analytes in various concentrations. Thus each observation consists of 2280 values: 19 fibers at 2 wavelengths sampled 60 times.

The sensor responses are inherently aligned due to the "sniff" signifying the beginning of each observation, and the response for each sensor for each observation is normalized by subtracting the background sensor fluorescence (the intensity prior to exposure to the analyte) from each response to obtain the the change in fluorescence intensity for each fiber at each wavelength. No other preprocessing has been performed on these data.

The classification problem is the detection of trichloroethylene (TCE) in complex backgrounds. TCE is a carcinogenic industrial solvent, and is of interest as the target due to its environmental importance as a ground water contaminant. It gained notoriety in the book *A Civil Action*, by Jonathan Harr, and the movie of the same name starring John Travolta.

In addition to TCE in air, eight diluting odorants are considered: BTEX (a mixture of benzene, toluene, ethylbenzene, and xylene), benzene (Ben), carbon tetrachloride (CTe), chlorobenzene (ClB), chloroform (Clf), kerosene (Ker), octane (Oct), and Coleman fuel (WGa). Dilution concentrations of 1:10, 1:7, 1:2, 1:1, and saturated

vapor are considered. In addition, there are 40 observations of TCE alone, with no confusers.

The database contains $n_0 = 352$ observations from class 0, the TCE-absent class. These consist of 32 observations of pure air and 40 observations of each of the eight diluting odorants at various concentrations in air. There are likewise $n_1 = 760$ class 1 (TCE-present) observations, consisting of 40 observations of pure TCE, 80 observations of TCE diluted to various concentrations in air, and 80 observations of TCE diluted to various concentrations in each of the eight diluting odorants in air. Thus there are $n = n_0 + n_1 = 1112$ observations in the training database.

The time series data can be used as-is, or can be smoothed. We use a cross-validated smoothing spline method for smoothing the fibers (the function smooth.spline in R, selecting the smoothing parameter via cross-validation). Examples of the smoothed curves are given in Figure 1.9. These show the curves for several different fibers, under different analyte mixtures. These curves were chosen arbitrarily, and do not necessarily represent the best fibers for discriminating between the different analytes. What is clear in the figure is that different fibers result in different responses, and that different analytes produce different responses in individual fibers. Note that any distance measure on the smoothed curves will be a dissimilarity measure when considered on the unsmoothed curves (since smoothing can take two different observations to the same curve).

The Tufts nose is not a linear sensor. That is to say, the response to the superposition of two chemicals is not the sum of the individual responses. Nor is there any theory to guide the model for the response. Thus we are required to produce a nonparametric classifier.

1.4.2 Hyperspectral Image

Modern imaging systems can now collect images in many wavelengths, both within and above/below those of visible light. These systems are called multi- or hyperspectral, depending on the number of bands simultaneously collected by the sensor. Each band corresponds to an image taken in a separate frequency range. This is similar to an RGB camera, that can be thought of as producing three images, in the red, green and blue wavelengths of light. We consider here a data set from a single hyperspectral sensor, the HyMap (see www.intspec.com), which in this case collected a 126-band image. This is illustrated in Figure 1.10.

Figure 1.11 shows a typical band of this image. A runway is discernible in the upper half of the image, and a stream winds its way through the lower half. (In this image the dark patch corresponding to the stream consists of the marshy area surrounding the stream as well as the stream itself. In other bands the water is much more distinct.) In the lower left corner is the Potomac river. Below the runway are two small wooded areas, consisting of mostly pine trees to the northwest, and oak to the southeast.

Several data sets have been extracted from this hyperspectral image cube, corresponding to subsets of the pixels associated with the classes indicated above. The

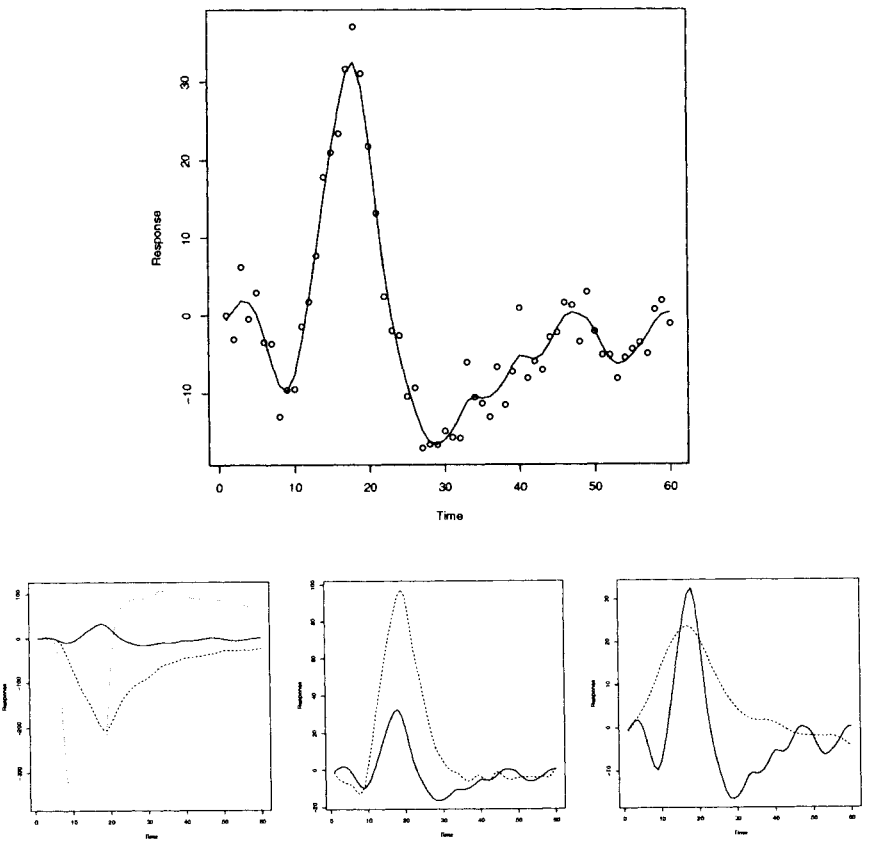


Fig. 1.9 Examples of responses of the fibers in the Tufts artificial nose. The top shows the smoothed curve plotted against the unsmoothed response (dots) for a single fiber. Below are the responses of three fibers to the same analyte (TCE in air), the same fiber to different analytes (TCE in air and TCE in BTE), and the same fiber to TCE in air and pure air. In all the plots, the solid curve corresponds to TCE in air.

problem at hand is to construct a classifier that will assign to each pixel the correct ground-cover class.

1.4.3 Gene Expression

Gene expression data consist of measurements on the expression levels of an array of genes upon presentation of a sample. This can be used both for diagnosis, providing information as to the disease or condition suffered by the patient from which the

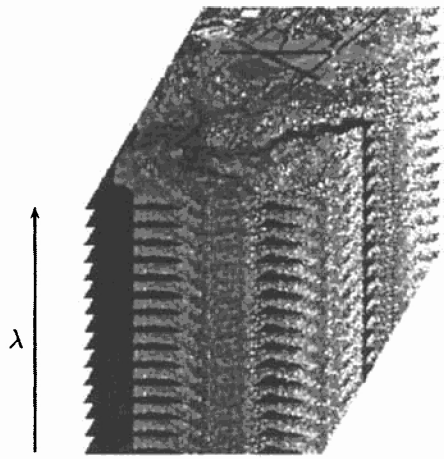


Fig. 1.10 Illustration of a hyperspectral data cube. The cube consists of a sequence of images (bands) taken at different wavelengths λ.

sample is taken, or for research, to determine which genes are active in or relevant to a particular disease.

Typical data sets are on the order of a few dozen patients and many thousands of genes. Thus the problem tends to be very high-dimensional, and thus quite challenging from a statistical point of view. We will consider one oft-used data set to illustrate several techniques in later chapters.

The Golub et al. data set, ([74]) produced by Affymetrix DNA microarrays, involves two general classes of leukemia, ALL (acute lymphoblastic leukemia) and AML (acute myeloid leukemia). Each observation corresponds to a patient, with $n_{ALL} = 47$, $n_{AML} = 25$ for a total of 72 observations. There are 7129 genes, and so each observation is a point in 7129-dimensional Euclidean space; there are 6817 unique human genes monitored, augmented with some redundant probes and control elements. (See also [71].) It is important from a clinical perspective to be able to tell which type of leukemia a patient has, for the treatment regimen is quite different for the different types. As of 1999, there was no single test that would distinguish between the two types.

Note that this is not a "simple" data set. For example, a principal component analysis scree plot suggests that as many as ten or more dimensions are necessary to adequately account for the variability in the data set. Gene selection techniques such as the t-test suggest anywhere from dozens to hundreds of genes are significant, depending on the value of p chosen to define significance (from 0.01 to 0.05).

The ALL class has two (latent) subclasses, T-cell and B-cell, with $n_T = 9, n_B = 38$. The detection of such subclasses is an important goal in many applications, and we will use these data to illustrate some techniques to accomplish this.

Fig. 1.11 One band from a 126-band hyperspectral image of Dahlgren, Virginia.

1.5 FURTHER READING

There are a number of good books on pattern recognition. Some that have been mentioned earlier in this chapter include [65], [46], [51] ([50] is a classic and well worth hunting down) and [85]. Others include [190], [177] and [81]. See [84], [3] [56] and [76] for more information on clustering. Many of the basic techniques of computational statistics relevant to statistical pattern recognition are covered in [128].

There have been several survey papers written about pattern recognition, a recent one being [101]. See also [174] and [116]. The journals *Pattern Recognition* and *IEEE Transactions in Pattern Analysis and Machine Intelligence (PAMI)* are also good sources for current work, as is the journal *Machine Learning*.

There are also a number of good books on graphs and digraphs. Mentioned above were [34] and [24]. In addition to these, the interested reader is directed to the new books [48], [12], [211], [136], [15], and [216]. There are many books that deal with special topics in graph theory, or specific classes of graphs, such as [212]. Several books focus on applications of graph theory, including [59] and [79]. Graphs are used

in a wide range of applications, as can be seen in these references. Pattern recognition applications (to image processing) are given in [106].

Graph theory is a subset of discrete mathematics, and so most (if not all) books on discrete mathematics contain discussion of graphs. See, for example, [13], [5] and [213]. Reference [129] provides a very readable discussion of discrete mathematics in which graphs play a large part. A very different approach is taken in [78]. This book approaches discrete math from the perspective of mathematical logic. Graph theory is covered in a chapter late in the book.

Many books provide discussions of algorithms for graphs and digraphs. See, for example, [185], which provides code in C, as well as a discussion of the algorithms. Another book with extensive discussion of algorithms is [108]. Reference [207] considers graph and tree isomorphisms and traversals in some depth.

For random graphs, the classic [23] has been issued in a new edition, [25]. Two other recent books are [114] and [103].

Graphs also show up in many recreational mathematics books. A very interesting, and slightly unusual, application of graphs (and several other fields of mathematics) is provided in [154], where the mathematics of juggling and bell ringing is discussed. While graphs play a very small part in this book, it is still worth investigating, particularly if you are a juggler.

The two volume set [88] and [87] is the definitive reference on domination in graphs. These books include basic expository articles, as well as more advanced ones, and the first contains an appendix listing a large number of different types of domination and related ideas.

The term "curse of dimensionality" was coined in [19] and is covered in most books on pattern recognition and computational statistics such as those already mentioned. Donoho ([49]) provides a very nice exposition of the curse, as well as the blessings, of dimensionality. See also [109] for another look at the curses and blessings of dimensionality. Reference [214] discusses the curse of dimensionality from a number of perspectives, although most of the papers in this book are more focused on describing new techniques than in discussing the curse *per se*.

The nose data is described in some detail in [158] and [125], as well as the previously cited [47] and [217]. There is also some discussion of the data in [165] and [164]. A similar sensor is discussed in [11].

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

2

Computational Geometry

Geometry (which is the only science that it hath pleased God hitherto to bestow on mankind).

Thomas Hobbes, *Leviathan*

Possession of anything new or expensive only reflected a person's lack of theology and geometry; it could even cast doubts upon one's soul.

John Kennedy Toole, *A Confederacy of Dunces*

2.1 INTRODUCTION

Computational geometry is the study of algorithms for the solution of geometric problems. Since graphs are often used in the formulation of efficient algorithms, they often occur in computational geometry. Furthermore, they often occur naturally in many geometric problems, as we shall see. We will describe some of these graphs, focusing on those that are relevant to statistical pattern recognition. There are a number of good books on computational geometry. These include [156], [148], [41], [75] and [178].

To a large extent, pattern recognition reduces to questions about distances between points. There are a number of constructs in computational geometry that encode important aspects of the notion of nearness. We will consider Voronoi regions, which are defined in terms of nearest neighbors, their dual the Delaunay triangularization, minimum spanning trees, which are defined in terms of a shortest path through the points, and various extensions and generalizations of these ideas. Their utility for pattern recognition will be illustrated on various data sets, both real and simulated.

To illustrate some of the graphs discussed in this and subsequent chapters, we will consider a planar point set, given in Table 2.1 and depicted in Figure 2.1. In this example, points are drawn from two spherical (normal) clusters with different covariance matrices. The larger cluster, consisting of 25 points, is drawn from the two-dimensional standard normal, $N((0,0), I)$, while the smaller, consisting of 10 points, is drawn from $N((2,2), 0.5I)$. Here, I denotes the 2×2 identity matrix, and as always, $N(\mu, \Sigma)$ denotes the normal density with mean vector μ and covariance matrix Σ. Other data sets will be used to illustrate different types of clustering and classification techniques as needed.

Table 2.1 Data used for illustration of various graph methods: A set of points in the plane.

X	Y	X	Y	X	Y
−0.162	−1.149	0.0733	−0.144	2.351	1.779
0.409	1.0150	−1.020	−0.219	2.645	1.428
0.185	0.900	0.223	−0.175	1.384	1.724
1.165	0.0954	1.698	−1.922	1.624	1.479
−1.736	0.876	−0.210	0.944	2.328	1.651
−1.095	−1.984	0.860	0.565	2.237	2.520
−1.164	−1.185	0.030	0.257	1.724	2.145
−0.075	0.706	−0.392	1.339	1.766	1.830
−0.070	−0.755	1.252	−0.867	1.700	1.740
−1.520	−0.335	−0.571	−0.160	−0.273	0.785
0.924	1.188	−0.736	−0.304	2.188	2.116
0.292	0.697	1.412	0.082		

It is worth spending a moment looking at the points in Figure 2.1. We "know" that there are only two groups here, spherically symmetric and normally distributed, since we generated the points. Yet, it is possible to detect quite a lot of apparent "structure" in the spatial pattern of the points. This is because our eyes are the culmination of millennia of evolutionary pressure designed to detect patterns (food, lions) before the patterns detect us. Thus, we are very good at finding patterns, even when they don't exist. (The reader may be excused for thinking the dots in Figure 2.1 look like a giraffe.) This is one reason why automatic pattern recognition techniques (and statistical methods in general) are important areas of research. Two other reasons are the difficulty of visualizing data of higher dimensionality, and the need for reliable techniques that can be used in situations that humans would find tedious.

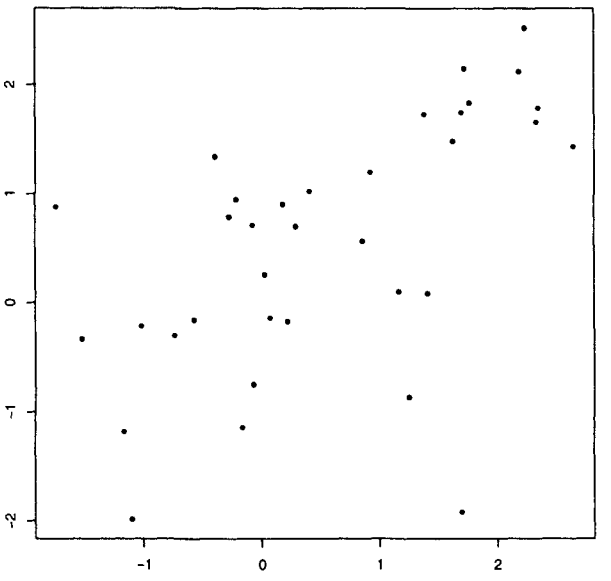


Fig. 2.1 Data used for illustration of various graph methods: A set of points in the plane.

2.2 VORONOI CELLS AND DELAUNAY TRIANGULARIZATION

Voronoi diagrams are used for many problems in computational geometry and its applications. See [145] for an extensive discussion. Essentially, the Voronoi diagram for a set of points encodes nearest-neighbor information for that set, delineating the region around each point that consists of points closer to that point than to any other in the set. This is made precise in the following definition.

Definition 2.1. *Given a set of points* $\mathcal{X} = \{x_1, \ldots, x_n\}$ *in* $\mathbb{R}^d$, *the* **Voronoi cell** *(or* **Voronoi region***) of the point* x_i *is the set of points in* $\mathbb{R}^d$ *which are closer to* x_i *than to the other points in* $\mathcal{X}$. *That is,*

$$V_i = V(x_i; \mathcal{X}) = \{x \in \mathbb{R}^d : d(x, x_i) < \min_{j \neq i} d(x, x_j)\}.$$

Usually the d is Euclidean distance. However other distances have been used. The **Voronoi tessellation** *or* **Voronoi diagram** *of the points is the set of points in* $\mathbb{R}^d$ *which have more than one nearest neighbor in* $\mathcal{X}$.

Many authors prefer *diagram* to *tessellation*. A **tessellation** of a set S is a collection $\{S_1, \ldots, S_n\}$ of subsets of S such that $\cup_{i=1}^n S_i = S$ and $(S_i \setminus \partial S_i) \cap (S_j \setminus \partial S_j) = \emptyset$ for $i \neq j$, where ∂S denotes the boundary of the set S. Sometimes one wishes to

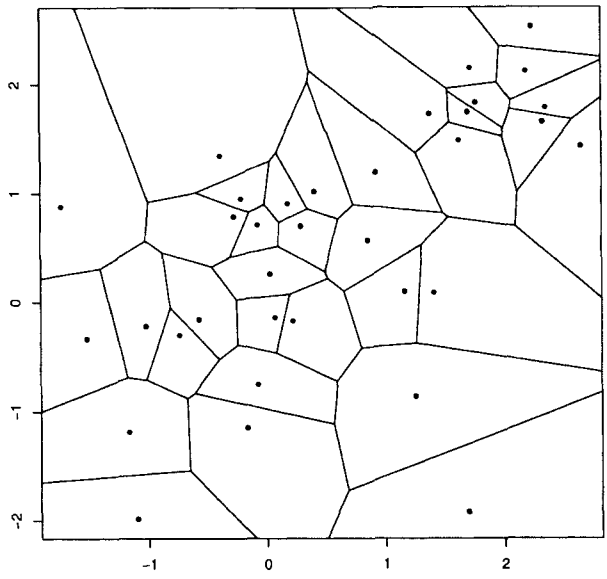


Fig. 2.2 A portion of the Voronoi tessellation defined by a set of points in the plane (Figure 2.1).

reserve the word "tessellation" for those cases where the S_i have some fixed geometric properties (such as being regular polygons, or chosen from a small set of geometric objects). For example, David Kendall argues for this stricter definition in the Forward to [145]. We will use the broader definition, with the understanding that we could be faulted for a "lack of theology and geometry".

Figure 2.2 depicts the Voronoi tessellation for our example points. Note that only a subset of the tessellation is actually shown here. Some of the cells are infinite, and some of the finite outer cells are quite large, so any plot containing them tends to make the rest of the plot indecipherable.

The intersections of Voronoi edges are called **Voronoi vertices**. The points that define the Voronoi tessellation are called **generators**. If every vertex in the Voronoi tessellation has exactly three edges the tessellation is called **non-degenerate**.

Definition 2.2. *The* **convex hull** *of a set of points is the smallest convex set containing the points. Another way to say this is that the convex hull is the intersection of all convex sets containing the set of points.*

We will occasionally blur the distinction between the convex hull and the boundary of the convex hull, using the same name for either (another "lack of theology and geometry"). In particular, when referring to the "convex hull graph" we mean the

graph corresponding to the vertices and edges of the boundary of the convex hull. It should be clear from context which is meant.

We will assume throughout that the points $\mathcal{X}$ defining the Voronoi tessellation are distinct. The Voronoi tessellation has several properties of interest. These include:

V1 The Voronoi cells tessellate $\mathbb{R}^d$. Hence the name "Voronoi tessellation" is appropriate.

V2 Each Voronoi cell is convex.

V3 A Voronoi cell is unbounded if and only if its generating point is on the boundary of the convex hull of the point set $\mathcal{X}$.

V4 All Voronoi edges are infinite straight lines if and only if the defining points $\mathcal{X}$ are collinear.

V5 A Voronoi edge defined by points x_i and x_j is a half line if and only if

- $\mathcal{X}$ is not collinear and
- x_i and x_j are consecutive points on the boundary of the convex hull.

V6 A Voronoi edge defined by points x_i and x_j is a line segment if and only if the line segment they define is not an edge of the convex hull graph.

V7 For each Voronoi vertex v there is a unique circle centered at v through any three points in $\mathcal{X}$. Further, the circle contains no generators, and it is the largest circle centered at v which contains no generators. If the tessellation is non-degenerate then the circle intersects exactly three points in $\mathcal{X}$.

Voronoi diagrams can be generalized in a number of ways. One such is the **higher order** Voronoi diagram:

Definition 2.3. *Given a point set* $\mathcal{X} = \{x_1, \ldots, x_n\}$ *and* $1 \leq k < n$, *the* **Voronoi diagram of order** k, $V_k(\mathcal{X})$ *is made up of regions of points all of which share the same* k *nearest neighbors in* $\mathcal{X}$.

Note that $V_1(\mathcal{X})$ is the standard Voronoi diagram of $\mathcal{X}$. We call $V_{n-1}(\mathcal{X})$ the **furthest site** Voronoi diagram.

Definition 2.4. *The* **Delaunay triangularization** *is the dual to the Voronoi tessellation. That is, it is the graph whose vertices are* $\mathcal{X}$, *with an edge between* x_i *and* x_j *if and only if the Voronoi cells of* x_i *and* x_j *share an edge.*

Some of the properties of the Delaunay triangularization are:

D1 If no four points of $\mathcal{X}$ are cocircular, then the Delaunay triangularization is actually a triangularization (made up exclusively of triangles).

D2 The Delaunay triangularization (viewed as the set defined by the triangles) is the convex hull of the points.

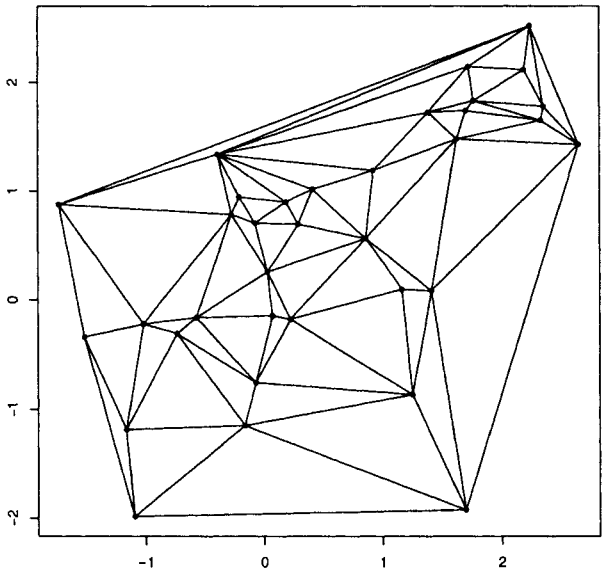


Fig. 2.3 The Delaunay triangularization defined by a set of points in the plane (Figure 2.1).

D3 The interior of each triangle contains no elements of $\mathcal{X}$.

D4 For any triangle, the circle defined by the three points contains no elements of $\mathcal{X}$ (see V7 above).

Many more properties of the Voronoi tessellation and the Delaunay triangularization can be found in [145]. There are numerous generalizations of Voronoi tessellations and Delaunay triangularizations, see [145] for an extensive discussion of many of these generalizations.

Figure 2.3 shows the Delaunay triangularization for the set of points of Figure 2.1. The eye tends to pick up the two clusters in this picture, but it is unclear whether this is more a result of prior knowledge biasing the interpretation than of an inherent quality of the triangularization. Figure 2.4 shows the Delaunay triangularization with the corresponding Voronoi tessellation.

Theorem 2.1. *The Delaunay triangularization of a set of n points in the plane can be computed in $O(n \log n)$ time.*

Similar results are available for higher dimensions. The order increases in d and the problem of constructing Voronoi tessellations or Delaunay triangularizations quickly becomes intractable in high dimensions.

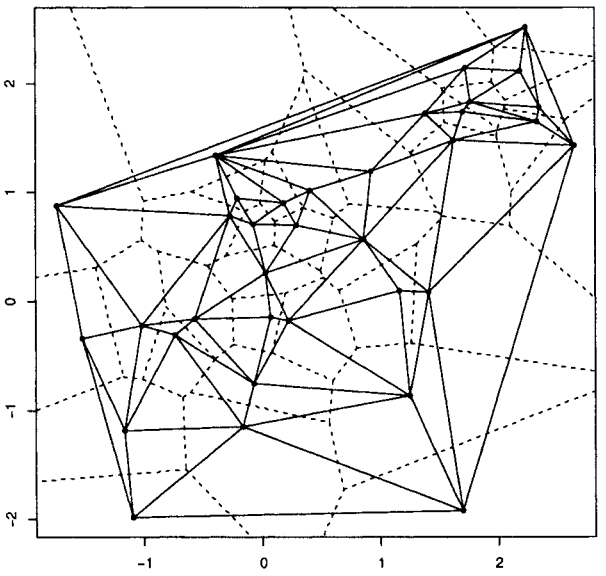


Fig. 2.4 The Delaunay triangularization and corresponding Voronoi tessellation (dotted lines).

Problems

2.1 Prove that the Voronoi cells in $\mathbb{R}^2$ are a tessellation.

2.2 Prove that the intersection of convex sets is convex. Prove that the intersection of all half planes containing a set of points in the plane defines the convex hull of the points.

2.3 Prove that any Voronoi cell is convex.

2.4 Show that a Voronoi cell is unbounded if and only if its generator is in the convex hull graph of the point set $\mathcal{X}$.

2.5 Prove Voronoi properties 4-6.

2.6 Prove Voronoi property 7.

2.7 Prove that for $n \geq 3$ non-collinear points in $\mathbb{R}^2$, the degree of the Voronoi graph is at least 3.

2.8 Let $x, y \in \mathbb{R}^m$ be distinct. Define the **dominance region** of x over y by $H(x,y) = \{z \in \mathbb{R}^m | \ ||x - z|| \leq ||y - z||\}$. For a finite set of distinct points

$\mathcal{X} = \{x_1, \ldots, x_n\}$, define

$$V(x_i) = \cap_{j \neq i} H(x_i, x_j).$$

Prove that $V(x_i)$ is the Voronoi cell of x_i.

2.9 Prove that if no four points of $\mathcal{X}$ are cocircular, then the Delaunay triangularization is actually a triangularization.

2.10 Prove that the Delaunay triangularization of the points $\mathcal{X}$ is the convex hull of $\mathcal{X}$.

2.11 Prove that the interior of each Delaunay triangle contains no elements of $\mathcal{X}$.

2.12 Show that for any Delaunay triangle, the circle defined by the three points contains no points of $\mathcal{X}$.

2.13 Let $D_n = D(x_1, \ldots, x_n)$ be the Delaunay triangularization of the points $(x_1, \ldots, x_n)$. Consider adding a point x_{n+1} to the set. Show that the new edges, those that are in D_{n+1} but not in D_n, are incident to x_{n+1}.

2.14 Consider the Delaunay triangulation for data generated uniformly in the unit square in $\mathbb{R}^2$. For several values of n, generate n such observations and compute the average area of the triangles. What is the apparent relationship between the average area and n?

2.2.1 Poisson Voronoi Cells

Definition 2.5. *A* **Poisson point process** *on $S \subset \mathbb{R}^d$ is the process for which, for any subset $A \subset S$,*

$$P(N(A) = x) = \frac{\lambda |A| e^{-\lambda |A|} x}{x!},$$

for $x = 0, 1, \ldots$, where $N(A)$ is the number of points in the set A and $\lambda \geq 0$ is a parameter called the **intensity** *of the process.*

A way to think about Poisson processes is that, conditional on the number of observations in a region, these observations are distributed uniformly over the region. When integrating out the event on which we have conditioned, $\{N(A) = x\}$, λ determines the probability $P[N(A) = x]$. [145] devotes a full chapter to the properties of the Voronoi tessellation and Delaunay triangularization for Poisson processes. Note that in this context the Voronoi tessellation and Delaunay triangularization are data random graphs.

Poisson point processes (and their sample-size conditional versions, uniform point processes) are widely used as a "null" model for statistical pattern recognition. Since classification and clustering are attempts to identify structure in data, a hypothesis testing approach identifies the absence of structure with the null hypothesis. The term **complete spatial randomness** is often used to identify this null hypothesis [37]. For instance, Penrose and Yukich [153] prove central limit theorems for functionals on

many data random graphs of interest in pattern recognition under Poisson and uniform process models (see Section 3.9).

Problems

2.15 Perform the following experiment:

- Generate $n = 100$ points from a uniform point process in the unit square. Compute the Delaunay triangularization.
- Retain the minimum and maximum area for the triangles.
- Repeat this 100 times.
- Investigate the distributions of the minima and maxima constructed.

2.16 Using the same methodology of Problem 2.15, consider the following test of complete spatial randomness against the alternative hypothesis of "clumping" (points clustering together): given $\beta > 1$, let τ be the number of Delaunay triangles with area less than βA, where A is the area of the smallest triangle. The test will reject in favor of clumping if τ is large. For several values of β and $n = 100$ as in Problem 2.15, use Monte Carlo simulation to investigate the distribution of τ.

2.17 Using the results from Problem 2.16 consider the following Cox process alternative:

- Draw 10 points uniformly from the unit square, $\{y_1, \ldots, y_{10}\}$.
- Using each point in turn draw 10 points uniformly from the ball $B(y_i, r)$.
- Discard the y_i, retaining the 100 points drawn from the balls.

Investigate the power of the test against this alternative as a function of β and r.

2.18 In Problems 2.16 and 2.17 there are two possibilities for defining τ: one can determine the value of A from the given data set, or one can use the expected value of A in the case of complete spatial randomness. Which is "better" (and what do you mean by "better")?

2.3 ALPHA HULLS

Alpha hulls are a generalization of the Delaunay triangularization (and the boundary of the convex hull). There are several definitions in the literature; we will use the following:

Definition 2.6. *For $\alpha > 0$, the* **alpha hull** *(also called the* **alpha shape***) of a set of points $\mathcal{X} = \{x_1, \ldots, x_n\}$ is the graph with vertices $V = \mathcal{X}$ and edge set $E = \{v_i v_j : \exists D_\alpha$ with $v_i, v_j \in \partial D_\alpha$ and $\mathcal{X} \cap D_\alpha = \emptyset\}$, where D_α is a disk of radius α, and ∂S*

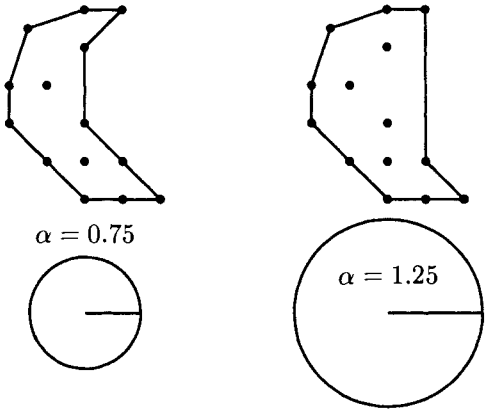


Fig. 2.5 Alpha hulls for two different values of alpha (indicated by the circles). They can be constructed by rolling the circles around the data, placing edges between points that touch the circle.

denotes the boundary of the set S. For $\alpha < 0$, *we require that the disk be of radius* $-\alpha$, *and* $\mathcal{X} \cap D_\alpha = \mathcal{X}$. *Finally, for* $\alpha = 0$, *the graph is the empty graph on* $\mathcal{X}$.

Note: Some authors use $1/\alpha$ in the definition. In this case, the definition for $\alpha = 0$ uses halfplanes: there is an edge between two points if the halfplane they define contains all (or none) of the points. In this case $\alpha = 0$ corresponds to the graph defined by the convex hull, while in our definition this graph results from any sufficiently large value of α. Generally alpha hulls are used for points in the plane. Replacing "disk" with "ball" in the definition extends the concept of alpha hulls from the plane to any dimension.

Figure 2.5 shows two alpha hulls for different values of alpha. One could argue that the figure on the left accurately depicts the shape of the data. The figure on the right is close to the convex hull of the data, and it is clear from this picture that a larger value of alpha will produce the convex hull.

Alpha hulls were originally defined to provide a method for delineating shapes in the plane. They work best when the points are sufficiently close together that the disks do not overlap other regions of the shape. They also provide a generalization of the convex hull graph to allow for relaxation of the requirement of convexity. As will be seen below, unlike the convex hull, the alpha hull is not guaranteed to be connected.

Figure 2.6 depicts the alpha hull for an "S" shaped data set, for various values of α. Note that the alpha hull graph need not be connected. However, for a judicious choice of α the "S" is quite nicely delineated. Note also that various choices of α do, in this example, provide something akin to the convex hull, where convexity is no longer strictly enforced.

Under the assumption that the point set corresponds to a single shape (and hence the alpha hull should be connected) one could define

$$\alpha_c = \underset{\alpha}{\text{argmin}}\{\text{the alpha hull is connected}\}.$$

Then α_c is a data random graph invariant measuring the "shape" of the distribution of the data. For example, Figure 2.6 shows that for the "S" data $\alpha_c \in (0.1, 0.7]$ (in fact, for these data $\alpha_c \approx 0.24$).

Consider a Poisson process (with known intensity $\lambda > 0$) on an "S"-shaped support set such as in Figure 2.6. Then the random variable α_c is of interest, and one can ask questions about its distribution, such as $E[\alpha_c]$.

For $\alpha < 0$, the disks are required to contain the set. Another way to say this is that the complement of the disk does not intersect the set. Figure 2.7 shows three alpha hulls compared with the convex hull for the data set of Table 2.1. Several things are noteworthy in these plots. First, note that the alpha hull is not really a hull at all in this case, as there can be points exterior to the polygon. As α decreases, this starts to correct, but there is still a point outside the hull for $\alpha = -10$ (one of the triangles indicated in the figure). In the upper right and lower left plots the appropriate circles are drawn for the points indicated by triangles, to illustrate this phenomenon.

One way to think about alpha hulls, at least for positive α, is to imagine placing a disk of radius α next to the point set, and rolling the disk around the set without covering any of the points. This can easily be done with Figure 2.5, using a coin for the disk.

Alpha hulls (α-shapes) were originally defined in [53], and some discussion and further references can be found in [178]. Several generalizations have been proposed, and the interested reader is encouraged to investigate the literature for more discussion of these. See also the web pages:

cgm.cs.mcgill.ca/~godfried/teaching/projects97/belair/alpha.html

and

n.ethz.ch/student/fischerk/alphashapes/as/

for examples and alpha hull code.

The question of the utility of alpha hulls for pattern recognition presents itself. Clearly, they are of value for providing a method of delineating shapes (at least in the plane, as indicated by Figure 2.6). There are other obvious uses, however. One that comes instantly to mind is that of clustering. One can imagine several ways to utilize alpha hulls for clustering. The simplest (perhaps) is depicted in Figure 2.8. Here we plot the number of components of the alpha hull against α and look for an "elbow" in the curve. This seems to indicate a clustering into three clusters, as indicated in the right hand plot of the figure. It would be hard to fault this choice of clusters (recall, however, the experiment described on p. 26). Figure 2.9 depicts the results of several standard clustering techniques on these data, using the same number of clusters as was discovered by the alpha hull algorithm described above. As can be seen, the alpha hull clusters are identical, in this case, to the single linkage clusters, which,

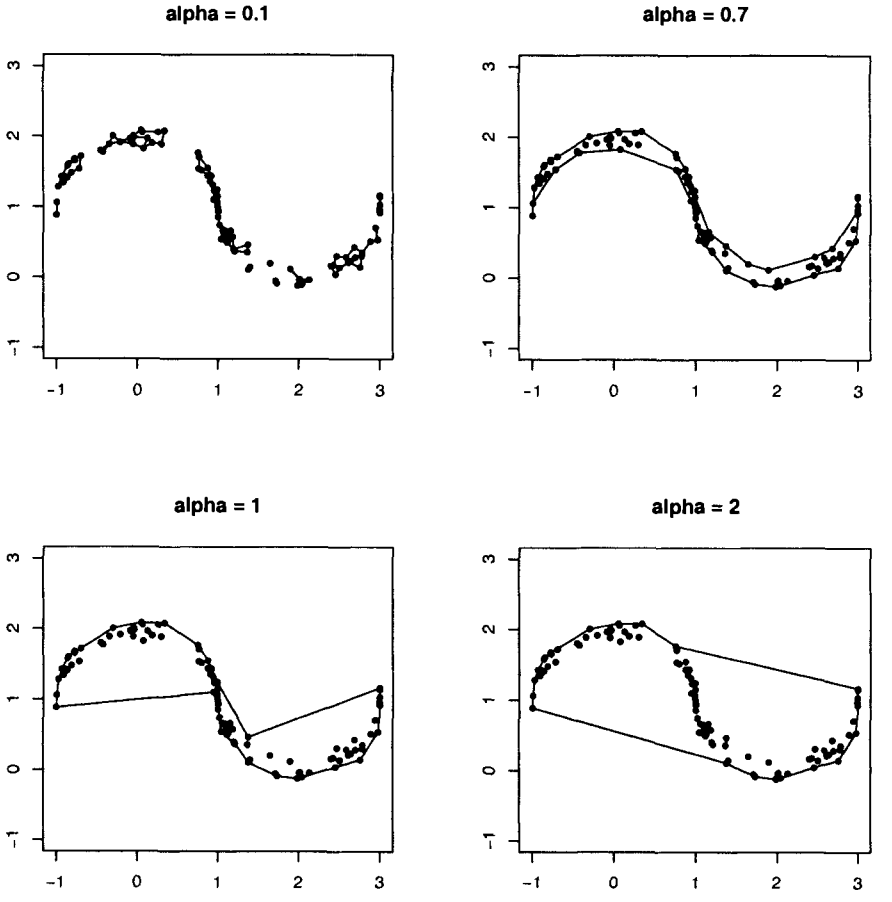


Fig. 2.6 Alpha hulls for four values of α for an "S" shaped data set.

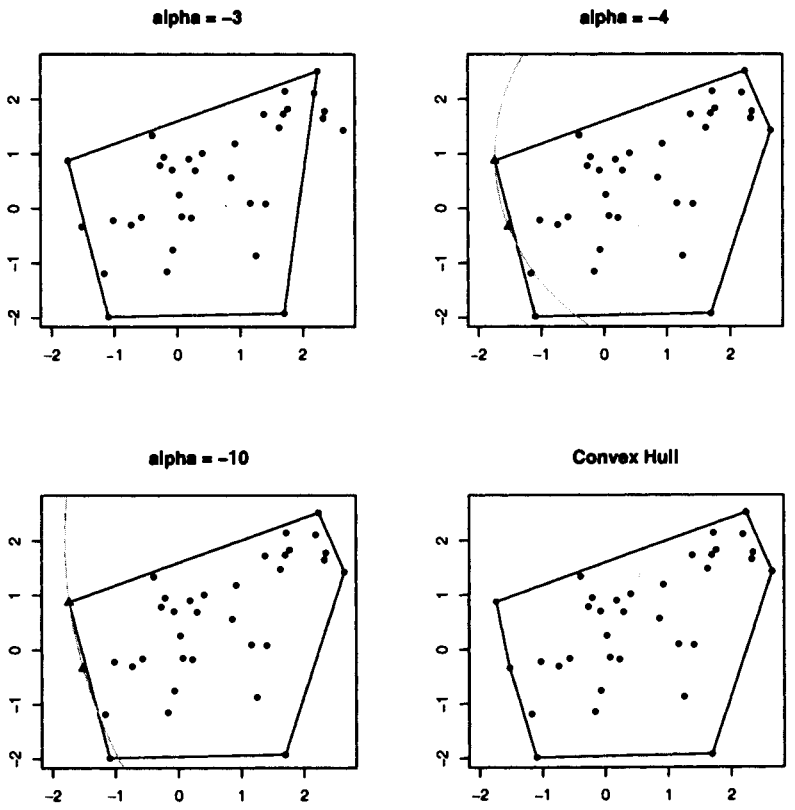


Fig. 2.7 Alpha hulls for $\alpha = -3, -4, -10$, and the convex hull of the data. In the panels for $\alpha = -4$ and -10, the circle corresponding to the two points indicated by triangles is depicted.

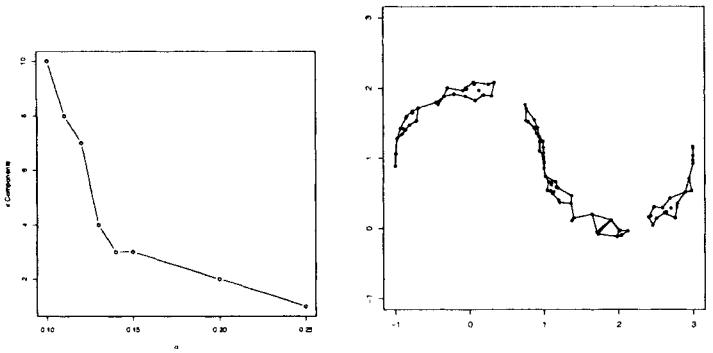


Fig. 2.8 Number of components plotted against α for the "S" shaped data. On the right is a plot for $\alpha = 0.14$ (near the "elbow" of the curve on the left), for which there are 3 components.

it can be argued, is the best choice of clustering among the three algorithms. The alpha hull algorithm has the advantage that (with the caveat that "finding the elbow" in the curve was not precisely defined) it provided the number of clusters as well as the clustering, something that none of the other methods is designed to do.

The alpha hull clustering algorithm described above determined that there are three clusters in the "S". As mentioned above, one could argue that this is reasonable for the specific data set investigated. Given that we have decided that $\alpha = 0.14$, one could ask about the distribution of the number of components of the "S" distribution. Thus, ν_α, the number of clusters in the alpha hull for a given value of α is a data random graph invariant of interest. In order to investigate the distribution of ν_α for this example, we need to be explicit about the distribution of the data.

Let $\theta \sim U(0, \pi), \epsilon \sim U(0.9, 1.1)$ and $X = (\theta - 1, \sin(2\theta) + \epsilon)$. This corresponds to the distribution from which the "S" data was drawn. We can now run an experiment to estimate the distribution of the number of clusters obtained by the alpha hull clustering technique. The results are depicted in Figure 2.10. In each experiment, 100 points were drawn from the distribution of X, and the number of components was determined for a fixed value of $\alpha = 0.14$. As can be seen in the figure, the value obtained initially is the mode, but there is quite a bit of variation. Note in particular that the "correct" answer (one cluster) was found less than 5% of the time.

Problems

2.19 For what value(s) of α is the alpha hull equal to the convex hull? Try to be precise, in terms of some statistic of the data set.

2.20 Show that for $\alpha > 0$, the alpha hull is a subgraph of the Delaunay triangularization.

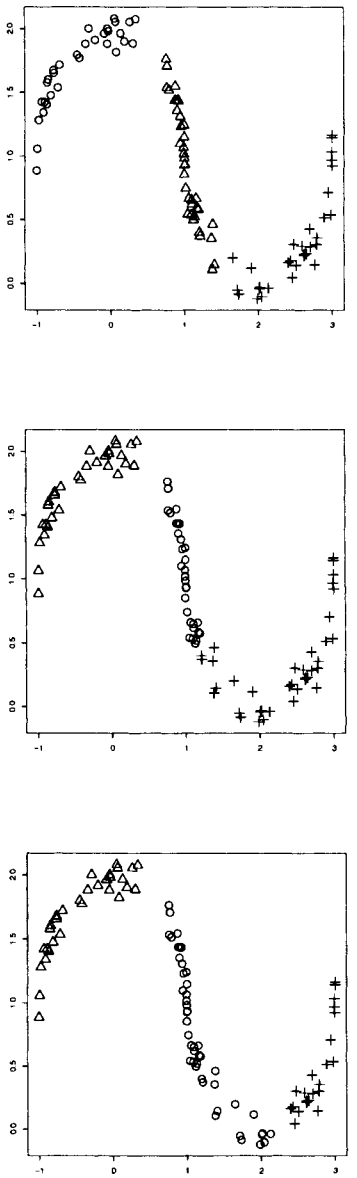


Fig. 2.9 Clusters found by the *k-means* algorithm (top) and hierarchical clustering using complete linkage (middle) and single linkage (bottom). The different clusters are denoted by different symbols.

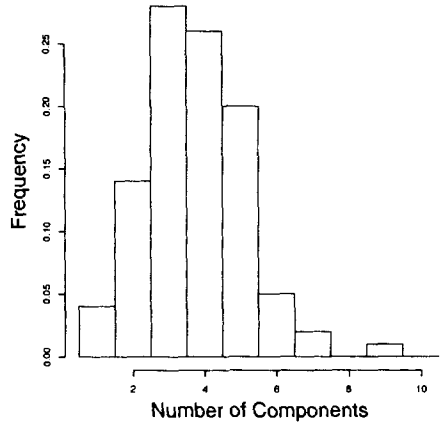


Fig. 2.10 Histogram of the number of clusters for 100 simulations of the "S" data set. In each case 100 observations were drawn, and α was fixed at 0.14.

2.21 For $\alpha < 0$ show that the alpha hull is a subgraph of the dual of the furthest site Voronoi diagram.

2.22 For the "S" distribution described in the text, use Monte Carlo replications to determine the distribution of the smallest value of α such that a single cluster results.

2.4 MINIMUM SPANNING TREES

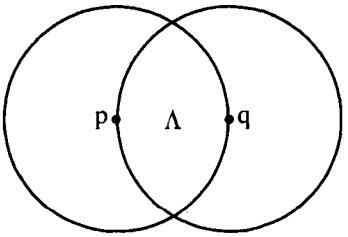
Definition 2.7. *Given a set of points $\mathcal{X} = \{x_1, \ldots, x_n\}$, a* **spanning tree** *is a tree on the vertices $\{x_1, \ldots, x_n\}$. Recall that this requires that every element of the set of vertices be in the tree. The* **minimum spanning tree**, *$T(\mathcal{X}_n)$, for the points is the spanning tree on the x_i such that the sum of the distances between adjacent vertices is minimum. This then is the minimum cost tree spanning the vertices.*

Note: Some authors use the terminology **minimal** spanning trees. In keeping with the distinction of minimal/minimum dominating sets (Definition 1.4) we prefer **minimum**. Also, to be a bit pedantic, the distance used in the definition is the distance between points in $\mathcal{X}$, not the graph distance. We are identifying the vertices and the underlying points that define the data random graph.

Usually the distances in the above definition are taken to be Euclidean distances, but this is not necessary. In fact, the definition can be made completely general, by considering any connected graph with weighted edges, and defining the minimum spanning tree to be the spanning tree with the minimum total weight. In the above case, the graph would be the complete graph, with edge weights equal to the inter-point distances – the distances between incident vertices.

Figures 2.11 and 2.12 depict the minimum spanning tree for the data of Figure 2.1. As can be seen, the minimum spanning tree is a subgraph of the Delaunay triangularization.

Definition 2.8. *For any pair of points p, q in the plane, the lune between them is defined by $\Lambda_{p,q} = B(p, d(p,q)) \cap B(q, d(p,q))$, where $B(x,r)$ is the open ball of radius r centered at x.*



Lemma 2.1. *If uv is an edge in the minimum spanning tree, then the lune between u and v contains no other vertex of the tree.*

There are a number of theorems providing complexity results for constructing minimum spanning trees. A typical one is:

Theorem 2.2. *The minimum spanning tree of a set of n points in the plane can be found in $O(n \log n)$ time.*

Two popular algorithms for constructing the minimum spanning tree are Kruskal's algorithm and Prim's algorithm. Kruskal's algorithm is a greedy algorithm that adds edges in order of their length, so long as they don't result in a cycle. Prim's algorithm starts with a tree consisting of a single vertex, and keeps adding the smallest edge to the tree.

input : A set of points $X = \{x_1, \ldots, x_n\}$
output : A minimum spanning tree for X

initialize: $T = \emptyset$. Order the m edges of the complete graph from smallest to largest
for $i = 1, \ldots, m$ **do**
  if $T \cup \{edge_i\}$ *does not contain a cycle* **then**
    $T = T \cup \{\text{edge}_i\}$;
  end
end

Algorithm 2.1: Kruskal's algorithm for finding a minimum spanning tree

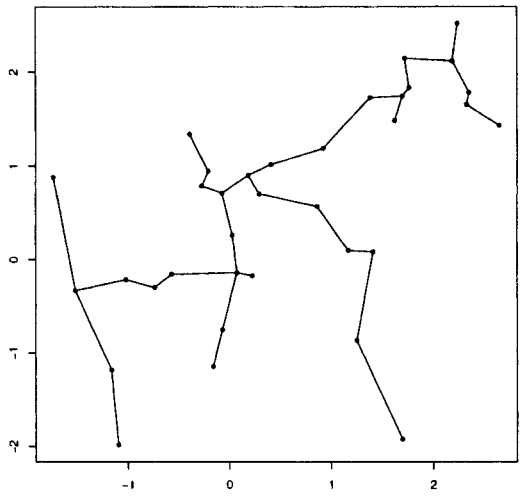


Fig. 2.11 A minimum spanning tree defined by a set of points in the plane (Figure 2.1).

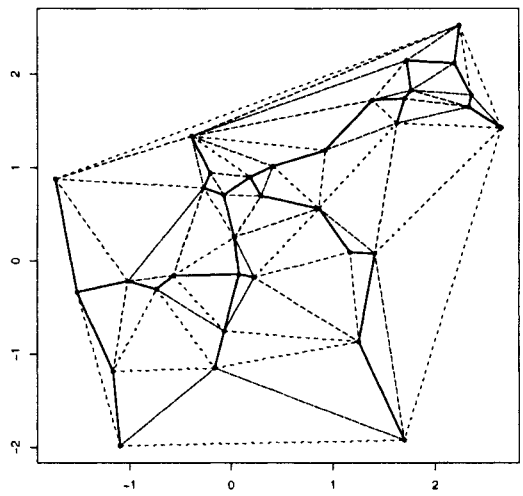


Fig. 2.12 The minimum spanning tree of Figure 2.11 depicted as a subgraph of the Delaunay triangularization of Figure 2.3.

input : A set of points $X = \{x_1, \ldots, x_n\}$
output : A minimum spanning tree for X

initialize: $S = \{v\}$ for any vertex v, $T = \emptyset$
while $|T| < n - 1$ **do**
  $w = \underset{z \in X \setminus S}{\text{argmin}}\{d(z, S)\}$;
  $S = S \cup \{w\}$;
  $T = T \cup \{\text{edge from } T \text{ to } w\}$;
end

Algorithm 2.2: Prim's algorithm for finding a minimum spanning tree

Problems

2.23 Show that the minimum spanning tree is a subgraph of the Delaunay Triangulation.

2.24 Prove Lemma 2.1.

2.25 Compare the running time of Kruskal's algorithm (Algorithm 2.1) and Prim's algorithm (Algorithm 2.2) for constructing the minimum spanning tree using Monte Carlo replication. Does it depend on the distribution of the data?

2.26 Select a branch from an (organic) tree or bush, and denote each leaf and branching point a vertex. Is the branch a minimum spanning tree for these vertices? Should it be? Can it be?

2.4.1 Alpha Hulls and the MST

Alpha hulls, described in Section 2.3, are used to define the shape of a set of points (usually a set of points in the plane). One issue that needs to be addressed in the use of alpha hulls is the selection of α. As shown above, for example in Figure 2.6, different values of α result in very different shapes. In [123], a method for selecting α using minimum spanning trees is described.

Given a set of points $\mathcal{X} = \{x_1, \ldots, x_n\}$, let l_n be the sum of the edge lengths in the minimum spanning tree of $\mathcal{X}$. Then the proposed choice for α is

$$\alpha^\star = \sqrt{\frac{l_n}{n}}. \tag{2.1}$$

This is illustrated in Figure 2.13. Here we have four different two dimensional sets and the corresponding alpha hulls, as chosen by the rule of Equation (2.1). The values of $\alpha^\star$ in this figure are (from left to right, top to bottom) $0.309, 0.242, 0.146$ and 0.175.

This illustrates the fact that, in some situations, a single value of α is not appropriate. Consider the values for the "H", the "annulus" and the "circled H" in Figure 2.13.

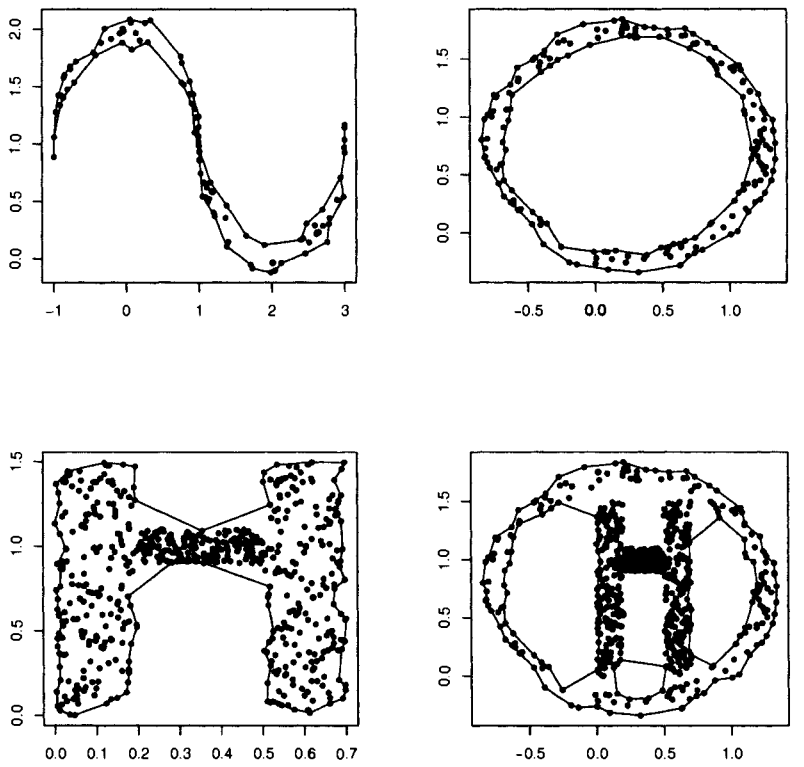


Fig. 2.13 Alpha hulls for four point sets, with α chosen using Equation (2.1).

Clearly, different values of α are appropriate in different regions. Unfortunately, if we use multiple αs, we cannot guarantee that the curves will be closed, and hence that we will obtain a "shape" for the point set. Instead, it seems reasonable to attempt to replace selection based on Equation (2.1) with more robust approaches.

Figure 2.14 shows the results for one such approach. In this case, we sum over only the middle 60% of the edges in the minimum spanning tree. (The value 60% is arbitrary, and chosen simply for illustration.) The αs in this case tend to be smaller: 0.281, 0.232, 0.138 and 0.157 (presented in the same order as above).

Note that the only figure substantially changed using this robust method is the circled H in the bottom right. The internal structure of the H is better modeled, at the cost of more edges between the circle and the H.

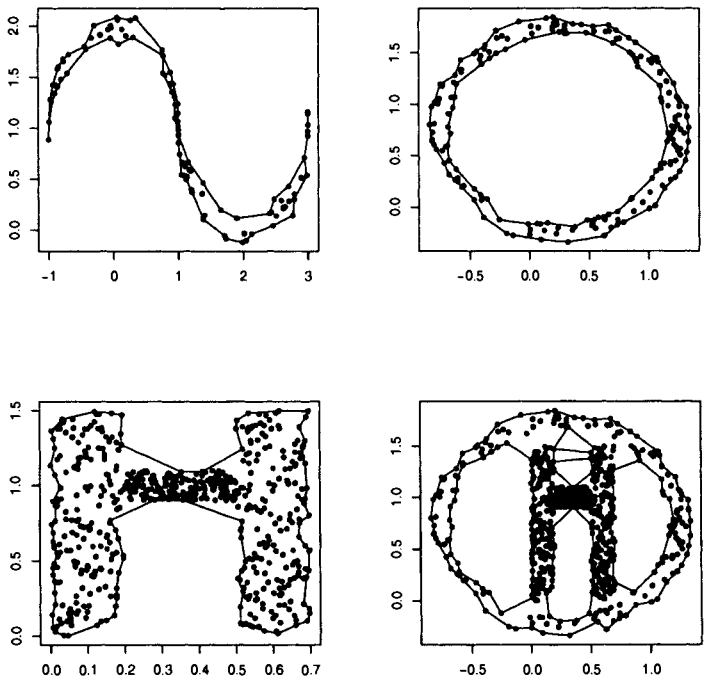


Fig. 2.14 Alpha hulls for four point sets. α was chosen using a robust version of Equation (2.1), where the only the middle 60% of the edges of the minimum spanning tree were used in the sum l_n.

Other choices are clearly possible, such as removing only the largest (or smallest) edge lengths. This does not solve the fundamental problem, however, which is that different regions may have fundamentally different scales.

Further work is clearly indicated. It might be possible to utilize local values of α provided that some "continuity" is maintained (close points should have close αs) in order to ensure that the curves remain closed.

It is worth noting that $\alpha^\star$ is a random variable. To illustrate this, we return to the "S" data, and run a simulation analogous to that of Figure 2.10. The set up is the same, except that $\alpha^\star$ is computed. In this case, out of 100 Monte Carlo replications, all but two produced a single cluster. (Two times the alpha hull was composed of two clusters.) The histogram of observed $\alpha^\star$s is depicted in Figure 2.15.

Another version of an adaptive α will be investigated in Section 3.5.1.

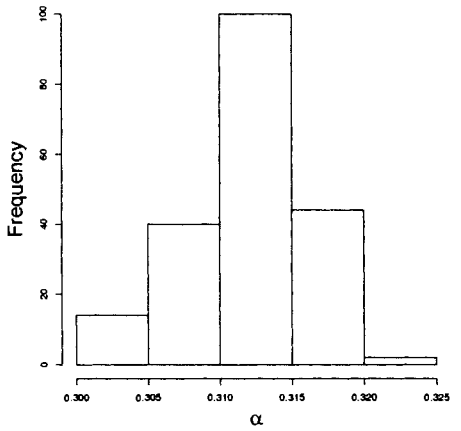


Fig. 2.15 Histogram of the values of α chosen by Equation (2.1).

Problems

2.27 Investigate the application of alpha-hulls and Equation (2.1) to the data in Table 2.1. Does this choice of α correctly elucidate the "shape" of the data?

2.28 For the data of Table 2.1, consider the use of multiple values of α as follows: cluster the data using the k-means algorithm (Algorithm 1.1), where $k = 2$. For each cluster, use Equation (2.1) to choose a value of α and compute the alpha hull for the points in the cluster, ignoring the points not in the cluster. Discuss the pros and cons of this idea.

2.4.2 Clustering

The minimum spanning tree can be used for clustering, using a local criterion for defining clusters. This idea is described in some detail in [221]. The idea is to break (remove edges from) the minimum spanning tree at edges that are "inconsistent". This results in a collection of connected graphs, one for each cluster.

Many definitions of "inconsistent" are possible. One could compute the standard deviation of the edge lengths incident on a vertex and eliminate edges which are "large" relative to this scale. We present one such example in Figure 2.16. In this case, we subtract the mean of the edge lengths incident to a vertex from the largest edge, then divide by the mean absolute deviation, to get a statistic for whether the edge is significantly greater than the others or not. Figure 2.16 depicts the result when the three largest deviations are deleted.

Perhaps a more compelling example is presented in Figure 2.17. The data here corresponds to observations drawn from two normals, one a standard normal, the other

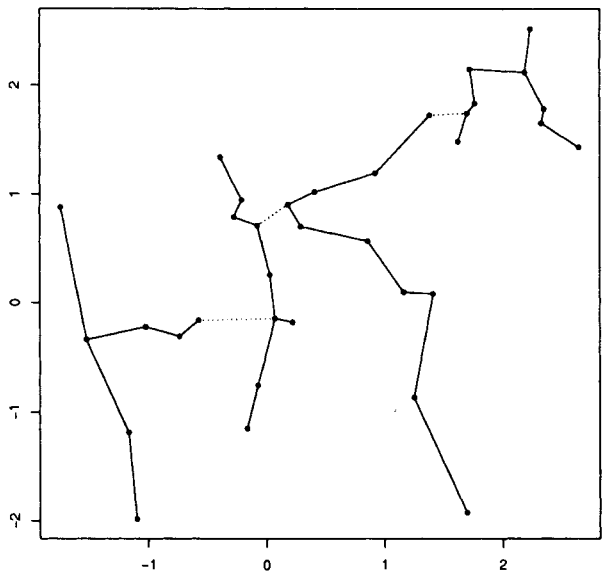


Fig. 2.16 Clustering result for the minimum spanning tree of Figure 2.11. The three deleted edges are indicated as dotted lines.

a normal with mean $(2, 2)$ and covariance $0.25I$. The algorithm correctly chooses the edge between the two components as the first edge to delete.

One might imagine that the approach discussed here is rather sensitive to outliers, and this is indeed the case. In order to help alleviate this problem, one generally eliminates from consideration any edge incident on a leaf of the tree. In order to illustrate the properties of this approach, we present in Figure 2.18 the results of a Monte Carlo simulation investigation. Here we draw two dimensional observations uniformly with support in two disjoint disks, of radius 1.0 and 0.5. In all cases there were 50 points drawn from each disk, the minimum spanning tree was computed, and the edge was removed according to the rule used for Figure 2.16, except that in this case we chose not to divide by the mean absolute deviation. Removing the edge produces two components to the graph, which define the clusters. The number of points incorrectly grouped was computed by comparison with the "truth" obtained from the disks. The experiment was run 100 times and the histogram of the error is shown on the right in Figure 2.18. An example of the data is depicted on the left. In each run the large disk was centered at $(1, 1)$. In the top row the small disk was centered at $(2.1, 2.1)$ and in the bottom row it was centered at $(2.2, 2.2)$. For a disk centered at $(2.3, 2.3)$ the MST algorithm was always perfect for the 100 Monte Carlo replications.

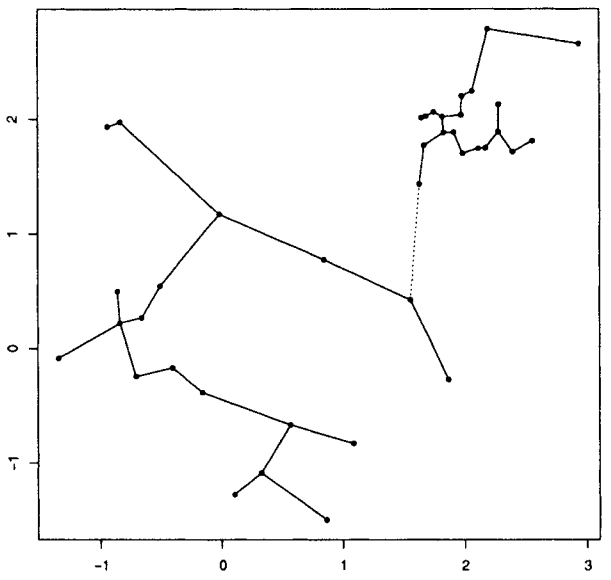


Fig. 2.17 Clustering result for the minimum spanning tree on a second dataset. The deleted edge is indicated as a dotted line.

One can extend these ideas to neighborhoods within a fixed depth from the candidate edge. Another inconsistency measure considered in [221] is the ratio of the edge length to the average edge length in a neighborhood of the candidate edge.

Many other variations and extensions of these ideas are possible. The literature on minimum spanning trees and clustering is large, and we will not attempt completeness here. Instead, we point out a few different approaches to the use of minimum spanning trees in clustering.

One simple approach is to use the minimum spanning tree in concert with a cluster quality measure, such as the ratio of the between cluster variance to the within cluster variance. One selects the edge to delete such that the chosen measure is optimized. Thus, to split the data into two clusters one needs check only $n-1$ possible clusterings.

Analogously to the definition of minimum spanning trees, we can define the **maximum spanning tree** as the spanning tree with the largest sum of edge distances. [10] describe using both minimum and maximum spanning trees for clustering. They define clustering as follows:

Definition 2.9. *A* k**-partition** *of a set of points* $\mathcal{X}$ *is a decomposition of* $\mathcal{X}$ *into* k *disjoint subsets. If* $k = 2$ *this is called a* **bi-partition**. *A* k*-partition is* **minimum diameter** *if the maximum diameter of the sets is minimal over all* k*-partitions. It is*

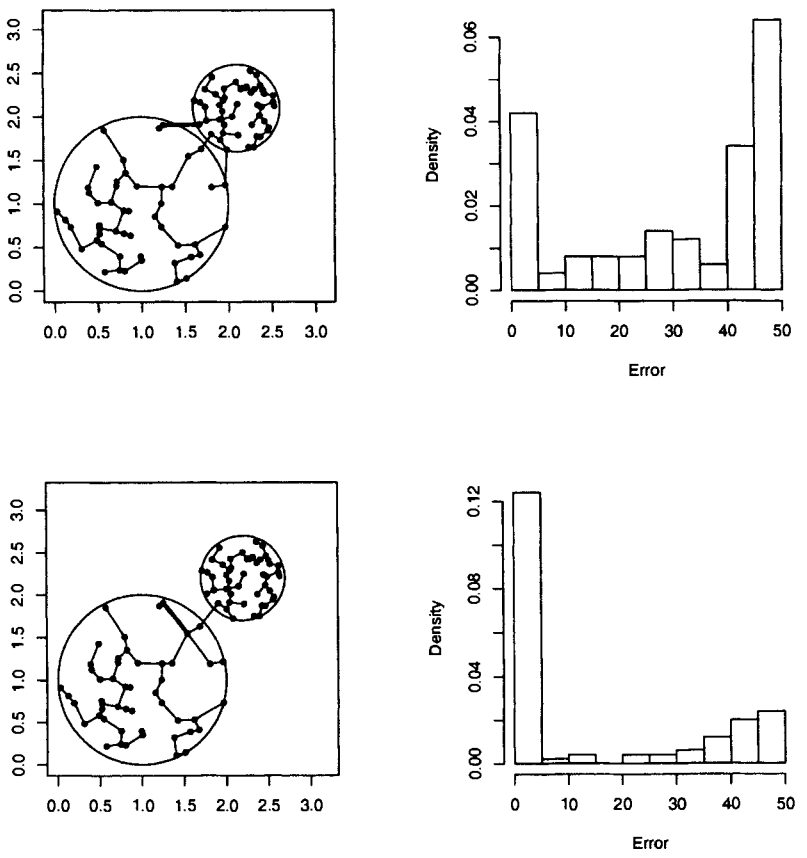


Fig. 2.18 Simulations of clustering data drawn from two uniform distributions. In each case 50 observations were drawn uniformly from each of the disks. Examples of the data set (with circles drawn for comparison) are shown on the left, and histograms of the number of points incorrectly classified are presented on the right.

a **farthest** *k-partition if the minimum inter partition distance is maximized over all k-partitions.*

First, they provide an $O(n \log n)$ algorithm for constructing a minimum diameter bi-partition of a set using the maximum spanning tree for the points. This is done by finding a line that intersects the maximum number of edges in the maximum spanning tree. This line separates the two clusters. It is then shown that the k components of the minimum spanning tree formed by removing the $k-1$ largest edges is a farthest k-partition.

Reference [35] uses the minimum spanning tree to merge clusters. They generate multiple seed points in a k-means clustering, then merge two clusters if their seed points share an edge of the MST.

Reference [111] discusses incorporating multidimensional scaling with the minimal spanning tree to aid in visualizing hierarchical clusterings. They project the data using multidimensional scaling, then connect points within clusters via the minimum spanning tree, and connect clusters via their two closest points.

Problems

2.29 Generate data from a mixture of two normals in $\mathbb{R}^2$ with identity covariances and equal priors but different means. Use the minimum spanning tree to determine the clusters. Be explicit as to the specific algorithm chosen to select the edge to delete. Investigate the effectiveness of the algorithm as a function of the distance between the means.

2.30 Fix the means of the components in Problem 2.29 at $(0,0)$ and $(2,0)$ and let one of the component covariances be σI. Investigate the algorithm as a function of σ.

2.31 Let $\mathcal{X}_n^0 = \{x_1^0, \ldots, x_n^0\}$ be n points distributed uniformly on the unit circle in $\mathbb{R}^2$. For $0 < r < 1$, let $\mathcal{X}_m^1 = \{x_1^1, \ldots, x_m^1\}$ be m points distributed uniformly on the circle of radius r. Consider the data set $\mathcal{X}_0 \cup \mathcal{X}_1$. How does the MST clustering algorithm perform as a function of r, n and m?

2.4.3 Classification Complexity

Minimum spanning trees can also be used in classification, as a measure of the complexity, or difficulty, of the classification problem (see [63] and [95]). Consider the data in a classification problem, $(X, Y) = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ is a feature vector and y_i is a class label. Construct the minimum spanning tree on the x_i without regard to the y_i, then count the number of edges between two points of different classes:

$$C_{\text{MST}} = \frac{|\{x_i x_j \in E : y_i \neq y_j\}|}{n}, \tag{2.2}$$

where E is the edge set of the minimum spanning tree.

Figure 2.19 illustrates this idea with data drawn from two normal distributions in $\mathbb{R}^2$, each with covariance I_2, the identity matrix. In each case, 50 observations are drawn from each normal distribution. One normal always has mean equal to the origin, while the other has mean $(3,3)$ in the upper left, $(2,2)$ in the upper right, $(1,1)$ in the lower left and $(0,0)$ in the lower right. The complexity measures (using Equation (2.2)) for these are, in order of decreasing separation of the means, $0.01, 0.04, 0.20, 0.48$. Figure 2.20 depicts a simulation showing that these results are "typical". This is the same experiment, in which 50 observations are drawn from each of the two normal distributions, for various values of the mean for the second distribution. The complexity, as in Equation (2.2), is averaged over 100 simulations, and plotted against the distance between the means.

As another comparison, the Iris data ([4], [57]) has complexity 8/150. This agrees well with the observation that the classes are well separated. For example, a cross-validated nearest-neighbor classifier incorrectly classifies 6 of the 150 observations.

While determining the complexity of the classification problem is interesting, it is secondary to the main utility of this idea. A more important application is as an index for dimensionality reduction. It is well known that high-dimensional data causes difficulty for classification and clustering applications. This is known in the statistics literature as the curse of dimensionality, see for example Section 1.2.2, [19], [203], [101] or any pattern recognition text. Thus, one of the important tasks of a pattern recognition practitioner is the selection of suitable projections of the data to reduce the dimensionality. For each candidate projection, one must assess its suitability. This can be done by applying a classifier to the result and seeing how well it performs, but this is strongly dependent on the assumptions made by the chosen classifier. Using a nonparametric method is clearly desirable, and one such as Equation (2.2) which is relatively quick to compute and is not tied to a specific classifier is quite attractive.

Consider the hyperspectral data (Oak vs. Pine) where, for the purposes of illustration, we have taken a subset of $n = 706$ pine observations and $m = 294$ oak observations. Equation (2.2) gives a value of 0.182 for this data set. Recall from Section 1.4.2 that these data are 126 dimensional, and it is reasonable to think that not all 126 bands are useful for classification in this problem. We can apply Equation (2.2) to the problem of dimensionality reduction on these data as follows: for each band b, remove the band from the data and compute the complexity using the 125 dimensional data. Bands whose removal results in a negative change in the complexity are presumed noise and should be removed. Bands resulting in a small positive change can also be removed, as they presumably are not important to the classification. Bands resulting in a large positive change should be kept.

This is, of course, a sub-optimal strategy. In fact, as [199] showed, in spite of our intuition to the contrary, the best single feature need not even be amongst the set of the best k features, so any approach that looks at single variates for dimensionality reduction is doomed to be sub-optimal. This is further illustrated, in a slightly different context, in [164].

Further, this step-wise procedure must be done carefully because of correlations. If two "good" features are positively correlated, then leaving either of them out will have little effect on the classifier, and hence, presumably, on the complexity. Thus

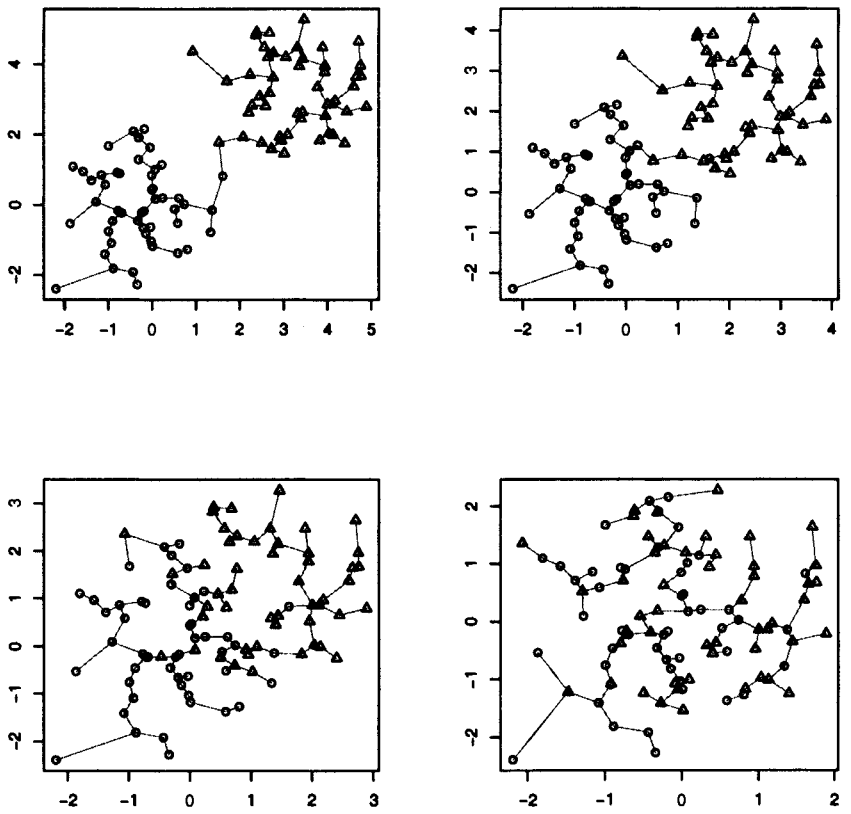


Fig. 2.19 Minimum spanning tree for data drawn from two normals, 50 points each. One normal (denoted by circles) has mean at $(0, 0)$, while the other (denoted by triangles) has mean (from left to right, top to bottom):$(3, 3)$, $(2, 2)$, $(1, 1)$, $(0, 0)$.

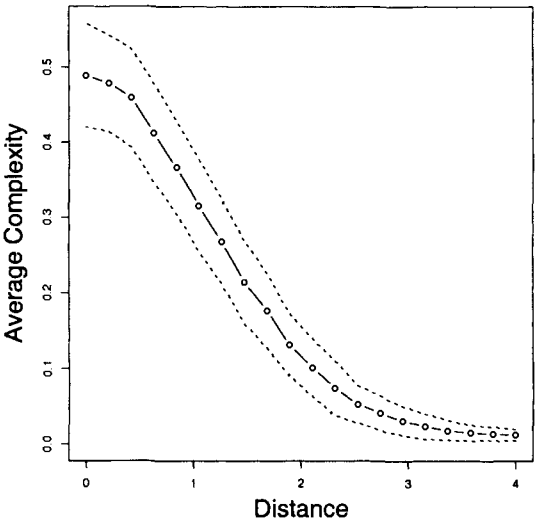


Fig. 2.20 Simulation of observations drawn from two bivariate spherical normals. The horizontal axis corresponds to the distance between the means, while the vertical axis corresponds to the average value of the complexity as measured by Equation (2.2). The dashed curves correspond to one standard deviation on either side of the mean.

the strategy of removing bands with no or only a small increase in the complexity is potentially hazardous.

These admonitions notwithstanding, we proceed with the above plan, yielding the following results. The leave-one-band-out complexities are depicted in Figure 2.21, with the horizontal line denoting the complexity value obtained using all the bands. The images correspond to the best and worst bands as determined from the values of C_{MST}. It is easy to see that the worst band is terrible, while a careful investigation of the best band shows that the two forested regions do have a different gray value.

Running a cross-validated nearest-neighbor classifier, we obtain the results in Table 2.2. The first column corresponds to the bands retained. The numbers refer to the complexity depicted in Figure 2.21. "All-Worst" refers to the removal of the worst band, and similarly "All-Best" ignores the best band. The last entry in the table, "Best", corresponds to just using the best single band. The best result displayed corresponds to the removal of the worst band, while the worst result comes from retaining only the very best single band.

Another look at this problem is presented in Figure 2.22. Here we have sequentially removed the worst bands (according to the value of C_{MST} as depicted in Figure 2.21), and computed the error for all bands remaining. Thus, for example, the entry at $x = 10$ corresponds to removing the 10 worst bands, and using the remaining 116 bands for classification. The figure shows that one can reduce the dimensionality of

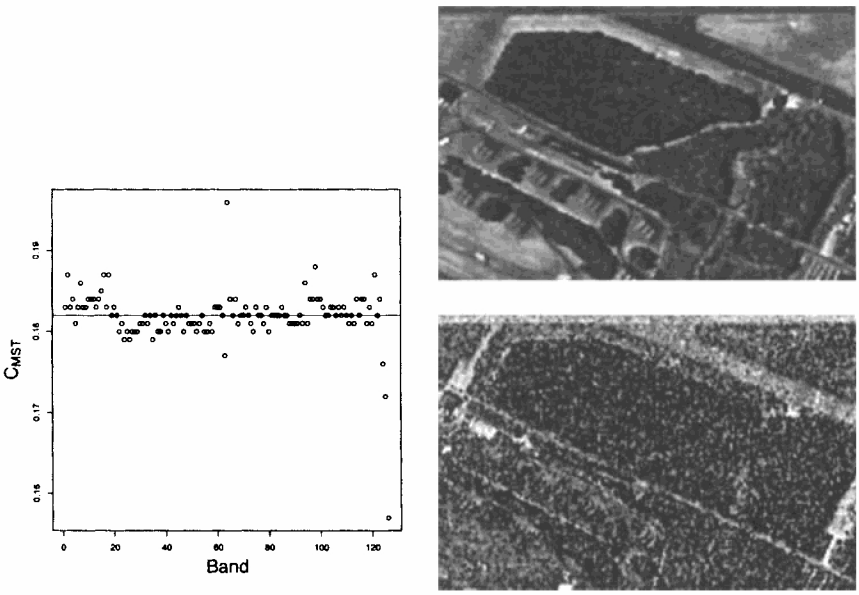


Fig. 2.21 Left: For each band, the leave-one-band-out complexity C_{MST} of the remaining 125 dimensional data. The horizontal line corresponds to the complexity C_{MST} of the full 126 dimensional data. Right: The best single feature ($C_{MST} = 0.196$) above the worst single feature ($C_{MST} = 0.157$).

the data to 37 and still have an error no more than that of the full-dimensional problem (in this case, the error is 0.173).

We can perform the same experiment with the artificial nose data (Section 1.4.1). The complexity of the problem is 0.157 (175 mixed edges). We can reproduce the dimensionality reduction experiment on these data, by successively removing a band and recomputing C_{MST}. The results are shown in Figure 2.23, top curve. Note that several of the bands have no effect on the complexity when removed. In one case, band 7, this is a result of an error in the data collection: this band is always identically 0. The results of a nearest-neighbor classifier are shown in the lower curve in the figure. The horizontal line here corresponds to the nearest-neighbor classifier error when all the fibers are used. The two curves roughly track each other, showing that they are extracting essentially the same information.

Figure 2.24 depicts a result analogous to that of Figure 2.22, using the nose data. In this case we can select five fibers (fibers 2, 12, 13, 24 and 36) and still have an error less than that of the full fiber set. Once again, this may not be the best 5-fiber set;

Table 2.2 Nearest-neighbor errors for several choices of bands for the hyperspectral data.

Bands	Dimensionality	Error
All-Worst	125	0.152
All	126	0.177
> 182	46	0.185
= 182	33	0.185
All-Best	125	0.193
<= 182	80	0.201
≥ 185	9	0.25
Best	1	0.294

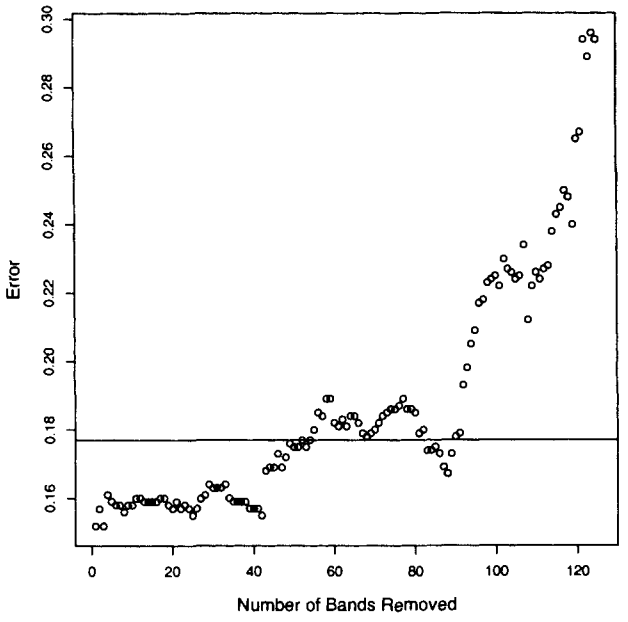


Fig. 2.22 Classification error obtained with a nearest-neighbor classifier using all but the x worst variables, as computed using C_{MST}, as depicted in Figure 2.21. The horizontal line corresponds to the error when using all the variables.

however, since there are 501,942 possible 5-fiber sets to check, this approach obtains a reasonable result using a relatively small number of computations.

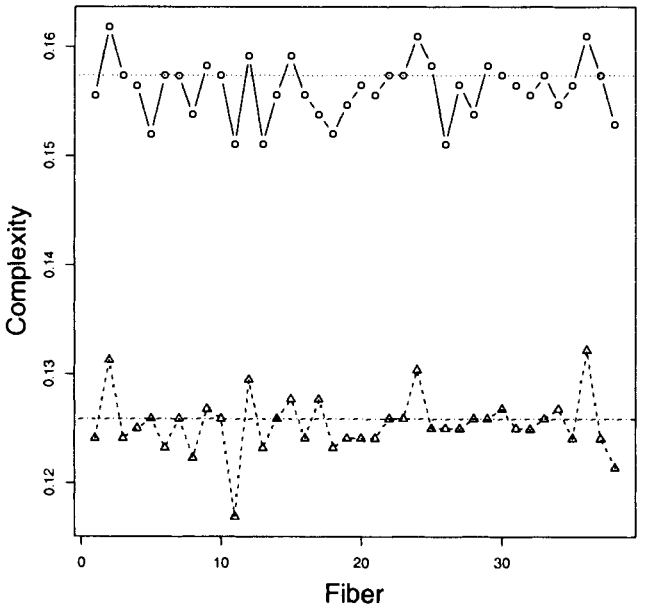


Fig. 2.23 The complexity C_{MST} computed on the artificial nose data, with each fiber left out, successively (dots). The horizontal line corresponds to the value of C_{MST} computed using all the fibers. The proportion of points missed by a nearest-neighbor classifier on the same data is depicted by the triangles. The horizontal line in this case corresponds to the nearest-neighbor error using all the fibers.

Problems

2.32 What can you say about the distribution of C in Equation (2.2) when $F_{X|Y=0}$ is equal to $F_{X|Y=1}$?

2.33 Reproduce the curve in Figure 2.20. Find a function that fits the curve. Can you prove that this function is the expected value represented by the curve?

2.4.4 Application: Rényi Divergence

The Rényi divergence generalizes some common distances between densities, in particular, the Hellinger and Kullback-Leibler divergences. Given two densities f_0 and f_1, with f_0 dominating f_1, and both having compact support (say on the unit hypercube), the Rényi divergence of f_1 with respect to f_0 is defined to be

$$I_v(f_1, f_0) = \frac{1}{1-v} \ln \int \left(\frac{f_1(x)}{f_0(x)} \right)^v f_0(x) \mathrm{d}x. \tag{2.3}$$

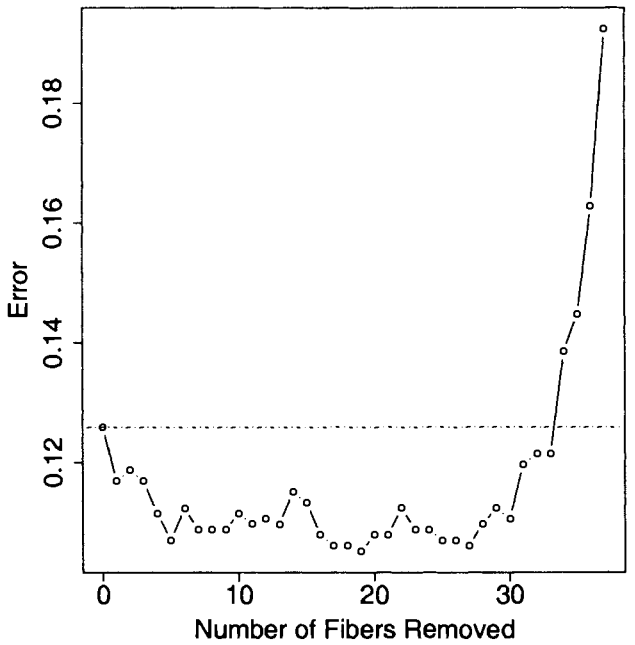


Fig. 2.24 Plot of the nearest-neighbor error against number of fibers removed for the artificial nose data. The fibers are removed in the order indicated by their complexity. The horizontal line corresponds to the error or a nearest-neighbor classifier on all 38 fibers.

For $v = \frac{1}{2}$, the Rényi divergence is the log of the Hellinger distance squared:

$$I_{\frac{1}{2}}(f_1, f_0) = \ln\left(\int \sqrt{f_1(x)f_0(x)}\mathrm{d}x\right)^2. \tag{2.4}$$

For the limit as $v \to 1$, one obtains the Kullback-Leibler divergence:

$$KL(f_1, f_0) = \int f_0 \ln \frac{f_0(x)}{f_1(x)}\mathrm{d}x. \tag{2.5}$$

These measures are used in a wide variety of statistical problems. For example, when choosing a model amongst a large family, one often makes the selection of the complexity of the model based on the least complex model such that a (penalized) estimate of the divergence from a more complex model is not significant. See, for example, [17], [18], [102].

The power weighted minimum spanning tree is the tree for which the power weighted sum of the edge lengths $\sum |e|^\gamma$ is minimized, for a fixed power $\gamma \in (0, d)$. Here d is the dimension of the data and e is the length of an edge. The sum is taken over all the edges in the tree. [93] define the k-MST to be the minimum spanning tree

on the subset of k points for which the power weighted sum is minimized. Denoting this minimum length $L_{n,k}$, they show that for $v = (d - \gamma)/d$, the statistic

$$\hat{H} = \frac{1}{1-v} \ln(n^{-v} L_{n,k}) + \beta(v, d), \tag{2.6}$$

where $\beta(v, d)$ is the Rényi entropy for the uniform density on the unit hypercube, is a consistent and robust estimator of the Rényi entropy

$$\frac{1}{1-v} \ln \int f^v(x) \mathrm{d}x. \tag{2.7}$$

By making a probability integral transformation for a given density f_0, [93] produce a robust consistent estimator for the Rényi divergence between the unknown density f_1 and the fixed (known) density f_0. Note that Equation (2.6) does not require an estimate of the density f_1.

The α-divergence is a further generalization that does not require the domination of f_0 over f_1, or the restriction to densities of compact support. Hero, Ma, Michel and Gorman [91] discuss this at length, providing a similar approach to estimation of the α-divergence, and an example application to geo-registration of imagery. [151] use the Rényi divergence to test for the equality of coefficients of variation. These have been used for various applications, including as features in image recognition tasks (see, for example, [194, 126]). Further work on using minimal graphs to calculate Rényi divergence can be found in [90]. Applications of Rényi entropy, mentioned in [92], are outlier detection, clustering, and image registration.

Thus we see another demonstration of the use of a data random graph functional for estimating a quantity of interest. While not directly a technique for pattern recognition, the above references demonstrate that these robust entropy estimators are of use in pattern recognition applications.

2.4.5 Application: Image Segmentation

The image segmentation problem consists of segmenting an image into disjoint regions such that the pixels within a region R are homogeneous (in some predefined manner) while the pixels in adjacent regions differ from the pixels in region R.

Thus, one needs a way to compare pixels (or groups of pixels) to determine whether they are "similar" or not. That is, one needs a dissimilarity measure. We shall assume that this dissimilarity has been provided by the application, but simple ones are easy to imagine, such as differences in gray-scale value of pixels (or standardized mean differences if one is considering sets of pixels).

Reference [181] describes an image segmentation method utilizing maximum spanning trees. Using their dissimilarity measure, the maximum spanning tree T is computed on the pixels. For each edge, a cost is computed based on the partition defined by removing the edge. For internal edges, find the edge with the minimum cost, and remove it, partitioning the pixels into two subsets. Continue this process until the minimum cost exceeds a user specified threshold δ.

The result is a partitioning of the pixels into subsets, corresponding to a segmentation of the image. Several examples are provided in [181], showing reasonably good results. Other graph-based techniques for image segmentation can be found in [106].

Segregation and Aggregation In many applications a fundamental question is: do the points cluster? In particular, it is often desirable to know whether one group tends to be closer to each other than to another group. This is referred to as **segregation**. When one group tends to be closely associated with the other, this is referred to as **aggregation**. The minimum spanning tree will give us a method for testing for segregation or aggregation of spatial data.

Consider the data in Figure 2.25. The data are from [140], and correspond to the positions of oak and maple trees in a forest. We have indicated the minimum spanning tree in the figure.

A question of interest is whether these trees are distributed randomly (complete spatial randomness) or whether the trees of each species tend to be grouped together. To test this, we use Equation (2.2). For sufficiently small values of C_{MST} we reject spatial randomness in favor of segregation; for large values we reject in favor of aggregation.

Let $X = \{x_1, \ldots, x_n\}, Y = \{y_1, \ldots, y_n\}$ and $N = n + m$. [63] show that under the null hypothesis of complete spatial randomness:

$$\begin{aligned} E[C_{\text{MST}}] &= \frac{2mn}{N} \\ Var[C_{\text{MST}}] &= \frac{2mn(2mn - N)}{N^2(N-1)} + \\ &\quad \frac{2mn(C - N + 2)}{N(N-1)(N-2)(N-3)} (N(N-1) - 4mn + 2), \end{aligned}$$

where C is the number of edge pairs sharing a common vertex. This results in an asymptotically normal distribution for

$$\frac{C_{\text{MST}} - E[C_{\text{MST}}]}{\sqrt{Var[C_{\text{MST}}]}}.$$

As can be seen for the data in Figure 2.25, $C_{\text{MST}} = 16/44$ (Equation (2.2)). To see if this is unusually small (indicating that the trees do tend to group together) one can use the above formula, resulting in a p-value of 0.033. As a further test we ran a simulation of 10,000 Monte Carlo replications from uniform data on $[0,1]^2$, and obtained a value of 16/44 or less 45 times, resulting in $p = 0.045$, and we feel fairly confident that we can reject complete spatial randomness. Note that the formula of [63] is conditioned on the observed value of C, while our simulation is not.

Problems

2.34 A model for aggregation can be defined in terms of a Cox process. Let $Y \sim F$. Draw m observations $y_1, \ldots, y_m$. For each y_i, draw n observations from $X_i \sim G_{y_i}$,

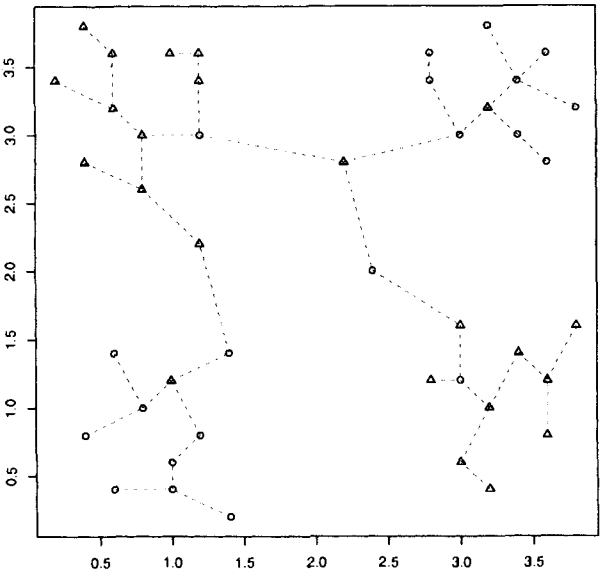


Fig. 2.25 Positions of oak (triangles) and maple (circles) trees in a forest, with the minimum spanning tree indicated with dotted lines.

$x_{i1}, \ldots, x_{in}$, where $E[X_i] = y_i$ (n may be a fixed value, or a random variable itself). The observations are the x_{ij}, y_i. Set F to be the uniform distribution on $[0, 1]^2$ and G_y the uniform distribution on a ball of radius r centered at y, and let $n \sim \text{Poisson}(\lambda)$. Investigate the distribution of C_{MST} (through Monte Carlo experimentation) as a function of m, r and λ.

2.35 Repeat Problem 2.35 using only the data x_{ij}, assigning the data associated with even values of i to class 0 and the odd to class 1. Investigate the distribution of C_{MST} (through Monte Carlo experimentation) as a function of m, r and λ.

2.5 FURTHER READING

There are many fine books on computational geometry, and we make no attempt at an exhaustive coverage here. A classic text is [156]. Another good text is [41], which presents many applications as well as algorithms. The handbooks [75] and [178] contain over seventy articles covering a wide range of topics.

For programmers, [148] provides algorithms and code for computational geometry. Minimum spanning tree algorithms can also be found in [185].

Reference [1] discusses using minimum spanning trees to provide efficient multicast algorithms on the Internet. [219] describes using minimal spanning trees to cluster gene expression data. In [192], a multidimensional analog of the median is defined via geodesic distances on a graph, in particular the minimum spanning tree of a data set. These are just a few illustrative application-oriented articles; readers are encouraged to browse the relevant literature and compile a list of such methodologies applied in their own application areas.

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

3

Neighborhood Graphs

Recent advances in computational geometry make nearest-neighbor rules the choice for pattern recognition in the coming decades.

Godfried Toussaint, *Lecture given at Interface 2002*

3.1 INTRODUCTION

Most techniques for statistical pattern recognition involve measurements on distances. The nearest-neighbor classifier is an extreme version of this point of view. In this chapter we will consider several types of data random graphs that use concepts of "nearness," or neighborhoods of points, to define the graph. We will start by giving a general definition of such neighborhood graphs, and then we will consider some special cases.

Definition 3.1. *Let $V \subset \mathbb{R}^d$. Associate to any pair $(p, q) \in V \times V$ a set $U_{p,q} \subset \mathbb{R}^d$. $U_{p,q}$ will be referred to as the* **neighborhood** *of p and q. Let $\mathcal{P}$ be a property defined on $\mathcal{U} = \{U_{p,q} \mid (p, q) \in V \times V\}$. The* **Neighborhood Graph** *$G_{\mathcal{U},\mathcal{P}}$ is the graph on the vertices V with edge set $E(G_{\mathcal{U},\mathcal{P}})$ defined to be the set of pq such that $U_{p,q}$ has property $\mathcal{P}$.*

While we use the term "neighborhood graph" in the definition, we will be interested in digraphs defined in the same manner. We will not be pedantic and require a distinction between graphs and digraphs in the definition, but will instead leave it up to the specific examples to make this distinction as necessary.

Definition 3.2. *A* **representation** *of the neighborhood graph in* $\mathbb{R}^d$ *is called a* **neighborhood skeleton** *or* **(geometric) realization** *of the graph.*

A common way to define the neighborhoods is through **regions of influence**. Informally, one defines a region around a point that corresponds to the "influence region" of the point. The neighborhood of a point, or a pair of points, is then defined in terms of these influence regions. We will see several examples of this basic approach in this chapter.

The regions of influence are almost always constructed from spheres or intersections of spheres (under some predefined metric). Essentially, the neighborhood of p is defined as points "close to" p, and the neighborhood $U(p, q)$ consists of those points "close to" both p and q. This agrees with our intuitive conception of neighborhood, where the regions of influence correspond to neighborhoods.

An early discussion of neighborhood graphs is found in [117].

3.1.1 Application: Image Processing

One kind of neighborhood graph derives from a fixed lattice of points on the plane, with edges defined by a given neighborhood structure. [16] discuss morphological operators that are characterized by neighborhood graphs of this type. In order to understand this idea, we will briefly describe mathematical morphology.

Definition 3.3. *A* **lattice** *is a nonempty partially ordered set* $\mathcal{L}$ *such that every finite subset has a least upper bound (supremum) and a greatest lower bound (infimum). A lattice is* **complete** *if every infinite subset has a supremum and an infimum. An* **operator** *is a mapping* $\phi : \mathcal{L} \to \mathcal{L}$.

Mathematical morphology is a methodology for the analysis of signals (usually images) in terms of shapes. It can be defined in terms of operators on lattices. See [89] for more information. Let $\mathcal{R}$ be a finite rectangle on the lattice $\mathbb{Z}^2$. A binary image is a subset $\mathcal{B} \subset \mathcal{R}$. That is, the image can be thought of as a copy of $\mathcal{R}$ where the pixels correspond to the integer-valued coordinates, and they take on the values 1 or 0 depending on if they are in the subset or not. A function from $\mathcal{R}$ to a finite subset $\mathcal{G} \subset \mathbb{Z}^+$ is a gray-scale image, in an analogous fashion.

A binary image is illustrated in Figure 3.1. The lattice is drawn as open dots, with the image shown as solid dots. In keeping with our intuition of an image as consisting of a finite rectangle of both 0s and 1s, we have indicated this in the figure with a dotted rectangle, although technically the image consists only of the solid dots. Fundamental to mathematical morphology is the concept of neighborhood, and it is this idea which leads to the use of neighborhood graphs. A neighborhood of a point in a lattice is simply a set of points. This is illustrated in Figure 3.1 as a digraph. Thus, the neighbors of a point in this figure are the points directly to the north, and east of the point.

Definition 3.4. *An* **erosion** *is an operator on the lattice which commutes with the infimum. A* **dilation** *is an operator which commutes with the supremum.*

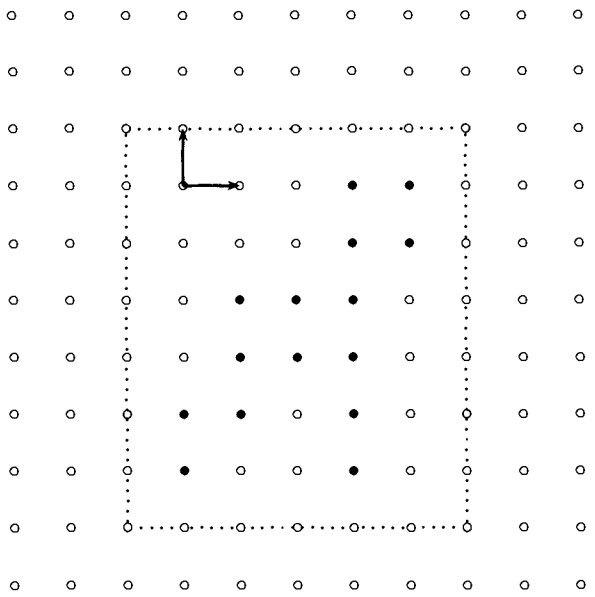


Fig. 3.1 A subset of the lattice $\mathbb{Z}^2$ with a binary image indicated by the dotted rectangle. The solid dots correspond to pixels with value 1 while the empty dots correspond to pixels of value 0. One neighborhood structure is indicated in the upper left corner of the image: each vertex is adjacent to its neighbors above and to the left.

Definition 3.5. *A pair of operators* (α, β) *is an* **adjunction** *provided that for all* $x, y \in \mathcal{L}$, $\beta(y) \leq x \iff y \leq \alpha(x)$.

It turns out that the only pairs of operators that are adjunctions are erosions and dilations, that is, α is an erosion and β is a dilation.

Definition 3.6. *A* **sup-generating family** *is a subset* $l \subset \mathcal{L}$ *such that every* $x \in \mathcal{L}$ *is the supremum of elements of* l. *A function* $a : l \to \mathcal{L}$ *is called a* **structuring function.** *A structuring function* a *has the* **property of dilation** *if there exists a dilation* δ *such that* $a(y) = \delta(y)$ *for all* $y \in l$.

Erosions and dilations are built up from structuring functions. An important aspect of mathematical morphology is the generation and characterization of structuring functions, and their applications to image analysis. [16] describe a class of structuring functions defined by a neighborhood graph on $\mathcal{R}$, and illustrate their use with applications to the detection of fracture lines in porous material, and the analysis of microscopic pulpwood images.

Given $\mathcal{R}$ and $\mathcal{G}$ as above, an **impulsive function** is a function $f_{u,v} : \mathcal{R} \to \mathcal{G}$ such that $f_{u,v}(u) = v$ and is zero elsewhere. Given a neighborhood graph G on $\mathcal{R}$, define the disk $D_G(u, r)$ to be the set of vertices in the graph that are within a (graph) distance of r of the vertex u. Given a function $g : \mathcal{R} \to \mathcal{G}$, write the restriction of g to $D_G(u, r)$ as $g|_{D_G(u,r)}$ (we are, as usual, identifying the graph vertices with the

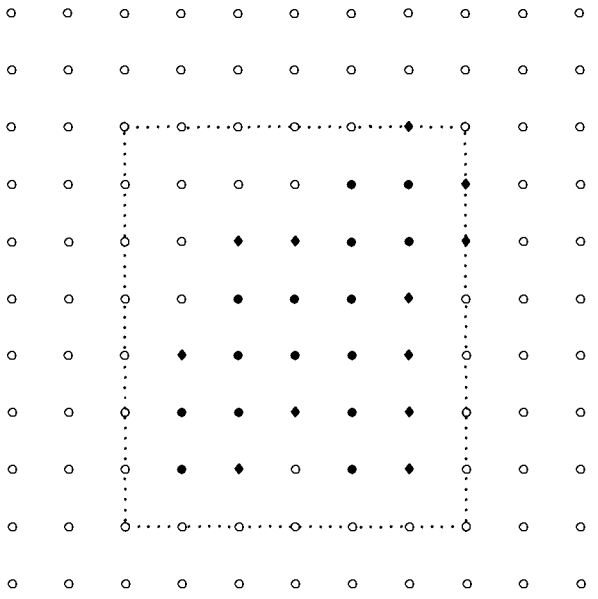


Fig. 3.2 A dilation of the image of Figure 3.1 by the structuring element defined by the neighborhood graph of north and east neighbors. The changed pixels are indicated by diamonds.

underlying points in $\mathcal{R}$). Then [16] define a class of structuring functions as

$$a(f_{u,v}) = g|_{D_G(u,r)} + v.$$

If g is the constant zero function, then a dilation of Figure 3.1 by the neighborhood graph defined as in the figure (connect north and east neighbors) is shown in Figure 3.2.

The erosion and dilation operators are the basic building blocks for many morphological operators. Further, [16] describe using morphological operators (the watershed, in particular) to define the vertices of the neighborhood graph used in the algorithms. This results in a very tight integration between morphological and graph theory operations. For more information on mathematical morphology, including the definition of the watershed and other morphological concepts, see [187] and [?].

In [16], these ideas are exploited to detect fracture lines in porous material, building a neighborhood graph using the holes in the material as the vertices (the hypothesis being that the most likely lines are those which go through the minimum number of these holes). By computing the geodesic distance, fracture lines are computed. A second application is the analysis of the celulosis production capacity in microscopic images of Eucalyptus pulpwood. Small "blobs" are detected in the images, and regions of low density of blobs are nonproductive. Using the blobs as edges in the graph, as segmented via watershed and other morphological operations, regions of large edge lengths can be detected and segmented out as nonproductive regions.

Problems

3.1 Investigate the properties of different neighborhoods using the image in Figure 3.1.

3.2 NEAREST-NEIGHBOR GRAPHS

Definition 3.7. *Given a set* $V \subset \mathbb{R}^d$ *and a point* $v \in V$, *define* $nn(v)$ *to be the* **nearest neighbor** *of* v *in* V; *that is,* $nn(v) = \underset{w \in V}{\operatorname{argmin}}\{d(w, v)\}$. *In order to simplify special cases, ties are broken arbitrarily, say by selecting the vertex with the largest index.*

Definition 3.8. *Define the* **nearest-neighbor digraph** *on* V, $NN(V)$, *to be the digraph on the vertex set* V *with edge set* $pq \in E(NN(V)) \iff q = nn(p)$.

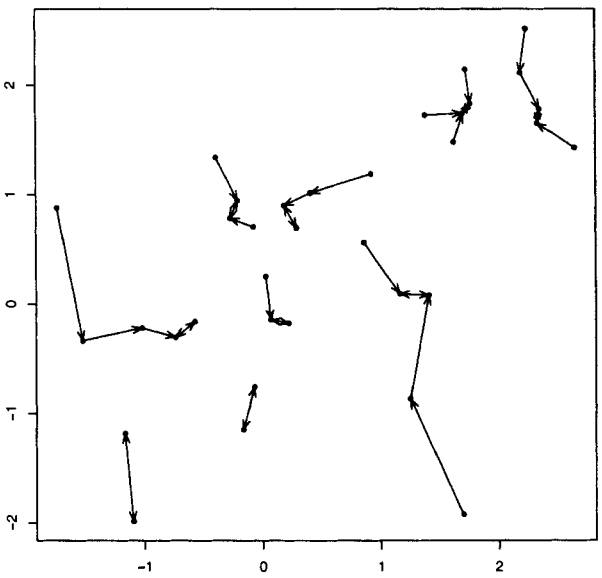


Fig. 3.3 A nearest-neighbor digraph defined by a set of points in the plane (Figure 2.1).

An example of a nearest-neighbor digraph is depicted in Figure 3.3. It can be seen in this example that the nearest-neighbor digraph can consist of a large number of components, unlike, for example, the minimum spanning tree (Figure 2.11), which is constrained to have a single connected component by definition.

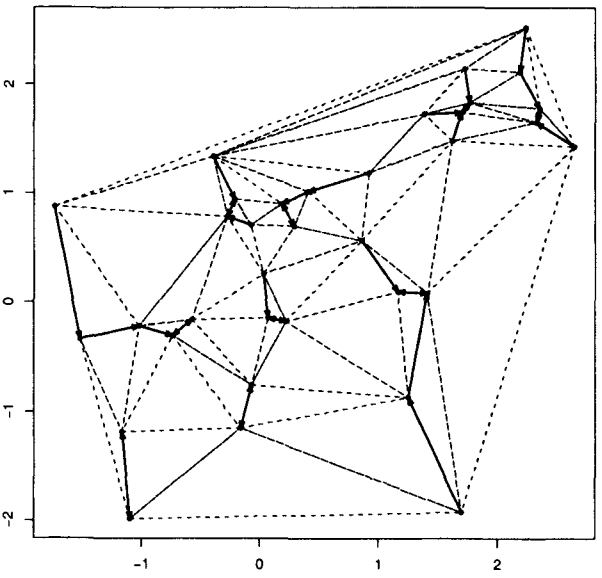


Fig. 3.4 The nearest-neighbor digraph plotted with the Delaunay triangularization of the points.

Some simple observations on the nearest-neighbor digraph can be found in the introduction of [152] and the expanded and improved version [55]. These are

1. Along any directed path the edges have non-increasing lengths.

2. The only cycles are 2-cycles. Ignoring the trivial case of a single point, each weakly connected component contains exactly one 2-cycle. This pair of vertices is called the **bi-root** of the component.

3. In $d = 2$ the graph is planar, and the maximum degree of a vertex is 6 (the kissing number). If the points are in general position the maximum degree is 5. Thus, if the points are observations from a continuous random variable then the maximum degree is 5 almost surely.

4. If the digraph is considered as a undirected graph, with bi-roots replaced by a single vertex, then this undirected nearest-neighbor graph is a subgraph of the Delaunay triangularization. See Theorem 3.3 and Figure 3.4.

The reader is encouraged to examine Figures 3.3 and 3.4 as illustrations of some of the preceding observations.

We can investigate the number of components of the nearest-neighbor digraph through simulations. Figure 3.5 shows the results for one such simulation. In this, X_n is distributed uniformly on the unit square, and 100 Monte Carlo samples are drawn, with $n = 10, 50, 100, 200, 500, 750, 1000$. For each sample, the nearest-neighbor digraph is computed and the number of weakly connected components divided by n is computed. As can be seen from the figure this is essentially constant, implying the number of components is linear in n (see the Problems for more discussion of this).

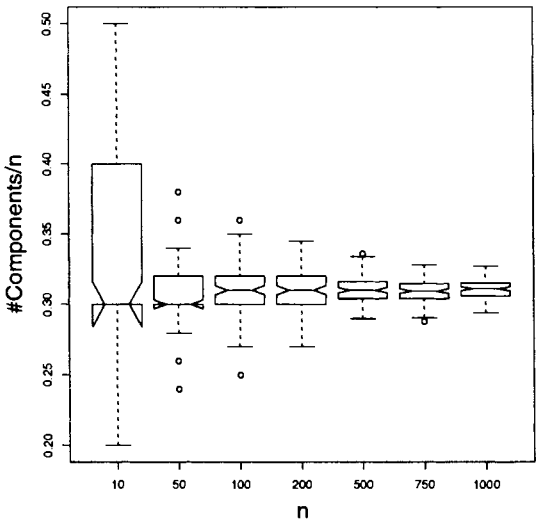


Fig. 3.5 The proportion of weakly connected components of the nearest-neighbor digraph for 100 Monte Carlo replications of two dimensional uniform random variates for various values of n.

Note that there are many ways to frame the nearest-neighbor digraph in terms of the neighborhood graph formulation of Definition 3.1. One such would be as follows. Let the neighborhood $U_{p,q}$ be defined as the ball centered at p of radius $d(p, q)$: $U_{p,q} = B(p, d(p, q))$. Define the property $\mathcal{P}$ to be $U_{p,q} \setminus \{p, q\} = \emptyset$. Then the nearest-neighbor digraph is the neighborhood graph $G_{\mathcal{U},\mathcal{P}}$ so defined.

Problems

3.2 Prove that the edges along any directed path in a nearest-neighbor graph have non-increasing lengths.

3.3 Show that the bi-root of a weakly connected component is unique.

3.4 Can you argue it is "obvious" that for data drawn uniformly on the unit square the number of weakly connected components of the nearest-neighbor digraph is linear in n (see Figure 3.5).

3.5 Generate data from a bivariate standard normal for various numbers of observations n. Construct the nearest-neighbor digraph, and investigate the number of weakly connected components as a function of n. How does this compare with the results in Figure 3.5?

3.3 K-NEAREST-NEIGHBOR GRAPHS

Definition 3.9. *Given a set $V \subset \mathbb{R}^d$ and a point $v \in V$, define $knn(v)$ to be the set of* **k-nearest neighbors** *of v in V. That is, $knn(v)$ consists of the k points in V closest to v.*

Definition 3.10. *The* **k-nearest-neighbor graph (KNN)** *on V is defined to be the graph with edge set defined by $pq \in E(KNN) \iff p \in knn(q)$ or $q \in knn(p)$. The* **Mutual k-nearest-neighbor graph (MKNN)** *requires the points to be k-nearest neighbors of each other; $pq \in E(MKNN) \iff p \in knn(q)$ and $q \in knn(p)$.*

As in the nearest-neighbor graph (Section 3.2) one can formulate the definition of the KNN as a neighborhood graph using Definition 3.1. We leave this as an exercise for the reader.

Examples of k-nearest-neighbor graphs for the points from Figure 3.3 are depicted in Figure 3.6. As can be seen in these plots, the graph tends to be connected as k increases, provided the clusters are not widely separated. Also note that the 1-nearest-neighbor graph is the same as the nearest-neighbor digraph where the arcs are all replaced with (two way) edges.

Note that for $k = 2$ the KNN has "discovered" the correct clustering of the data. For $k = 1$ there are 9 components to the graph and for $k \geq 3$ there is only one.

Figure 3.7 depicts the mutual k-nearest-neighbor graph for the same values of k as in Figure 3.6. It is obvious from the definitions that MKNN < KNN for fixed k, and these figures illustrate this fact.

The number of components of the MKNN and KNN graphs are plotted against k in Figure 3.8. Looking at these curves, one can see an elbow at either $k = 2$ (KNN) or $k = 3$ (MKNN). This would give a suggestion of either 2 or 6 clusters, if one were to use one of these statistics to estimate the number of clusters. Investigating the upper right panel of Figure 3.6 and lower left panel of Figure 3.7, one can see the clusters this analysis produces. The KNN provides the "correct" clustering. It is amusing to stare at the MKNN clusters and then look at the points in Figure 2.1. Humans are excellent pattern detectors, and it is easy to "detect" patterns that are spurious, especially if properly prompted. We will investigate another approach to clustering via these graphs in Section 3.3.3

Both the KNN and MKNN are graphs rather than digraphs. One could define the KNN digraph by restricting the definition to be one sided: $pq \in E(KNN) \iff$

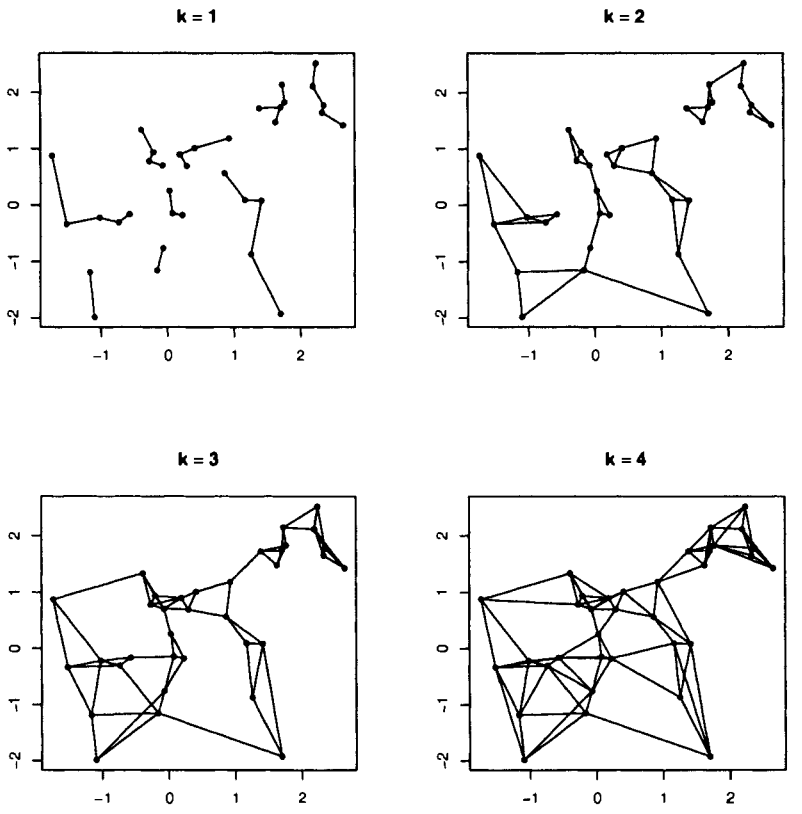


Fig. 3.6 k-nearest-neighbor graphs for four values of k for the set of points of Figure 2.1.

$p \in \text{knn}(q)$. We will always mean the graph defined in Definition 3.10 when we refer to the KNN, or k-nearest-neighbor graph, and will explicitly refer to the digraph version as the KNN digraph.

3.3.1 Application: Measures of Association

Consider the problem of determining how predictable one random variable is from another. Several statistics have been proposed for this (in some sense, this is what a large body of the statistical literature is concerned with). The theory of generalized correlation coefficients has been advanced to place the field on a firm theoretical foundation.

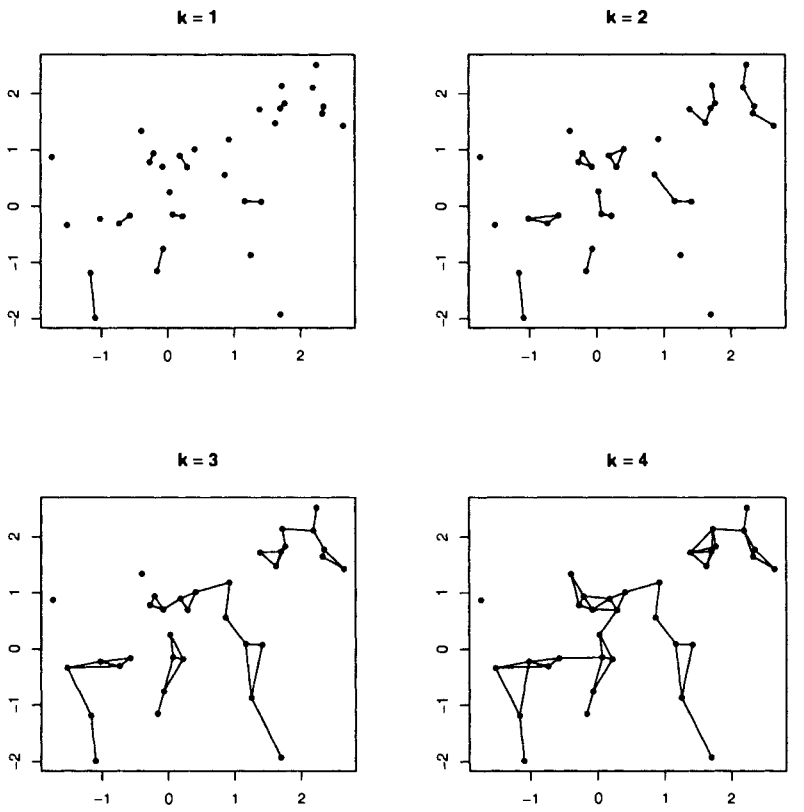


Fig. 3.7 Mutual k-nearest-neighbor graphs for four values of k for the set of points of Figure 2.1.

Let a sample of ordered pairs, $(x_1, y_1), \ldots, (x_N, y_N)$ be given. Let a_{ij} and b_{ij} be scores for X and Y. For example, these might be

$$a_{ij} = x_i - x_j, \tag{3.1}$$

$$a_{ij} = \text{rank}(x_i) - \text{rank}(x_j), \tag{3.2}$$

$$a_{ij} = \text{sign}(x_i - x_j), \tag{3.3}$$

and similarly for b_{ij}. These correspond to the Pearson, Spearman, and Kendall measures of association, respectively. Given the scores, assumed to be the same rule for each random variable, the generalized correlation coefficient is defined to be

$$\Gamma = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} b_{ij}. \tag{3.4}$$

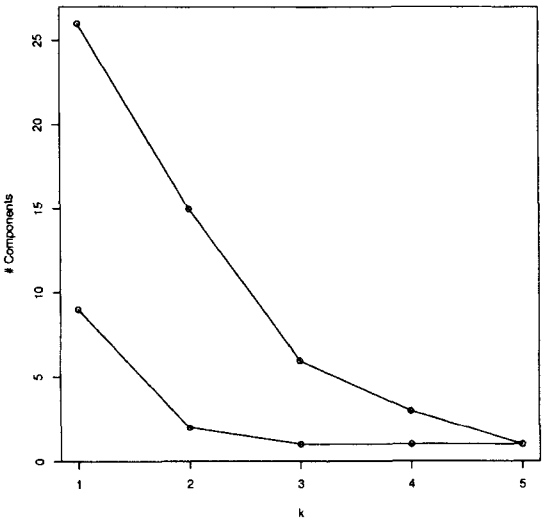


Fig. 3.8 Number of components of the k-nearest-neighbors graph (triangles) and mutual k-nearest-neighbor graphs (circles) as a function of k for the points of Figure 2.1.

The test for correlation (or, more properly, no correlation) is defined with reference to the distribution of Γ under the null hypothesis. That is, the distribution of

$$\Gamma(\pi) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} b_{\pi(i)\pi(j)}, \tag{3.5}$$

conditioned on the $\{x_i\}$ and $\{y_i\}$, where π is a permutation on the N indices and each permutation is equally probable.

Reference [64] considers the problem of extending this idea of a generalized correlation coefficient, and its associated test, to multivariate data. This is done through the KNN and the k minimum spanning tree (KMST). We will focus on the results for the KNN; the interested reader can consult [64] for the definition and results for the KMST.

The KNN used is slightly different than the one defined previously. Define KNN_X to be the graph on vertices $V = \{x_i, y_i\}$ such that there is an edge between (x_i, y_i) and (x_j, y_j) if and only if x_j is one of the K nearest-neighbors of x_i. A similar definition, reversing the roles of X and Y, holds for KNN_Y.

The statistic used to test for correlation is defined in terms of the number of edges that the graphs KNN_X and KNN_Y share. Let

$$a_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \in \text{KNN}_X \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

$$b_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \in \text{KNN}_Y \\ 0 & \text{otherwise,} \end{cases} \tag{3.7}$$

and

$$\Gamma_1 = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} b_{ij}. \tag{3.8}$$

Then the generalized correlation coefficient Γ_1 is the number of edges that the two graphs share. This is similar in flavor to the complexity measure of Equation (2.2). It is illustrated in Figure 3.9.

Consider Figure 3.10, where we revisit the sinusoidal "S" shaped data. Note that the high local correlation of these data appears to be discovered by the statistic. The edges in the plot seem to provide local information, that is, regions of locally high correlation seem to have a larger number of edges than regions of low local correlation. Consider Figure 3.11. In this the two "balls" each have the same number of observations as the "handle", and yet the handle represents nearly half the edges. To see if this is an artifact of the fact that the handle is parallel to one of the axes, see Problem 3.7.

A similar result is shown in Figure 3.12. The correlation is 0.05 for these data, and yet $\Gamma_1 = 123$ for 75 points. The statistic is finding the local correlation of the data; the edges of intersection are indicative of the regions in which the highest correlation is found.

Problems

3.6 Simulate data as in Figure 3.9, and determine empirically the distributions for Γ_1 for the two cases.

3.7 What happens if the "handle" of the barbell is rotated by 45^o? Does the contribution of the "handle" still outweigh that of either "ball"? What about if the handle is moved above the balls?

3.8 How does the value of Γ_1 change as we change the number of cycles in the sinusoidal "S" data? Investigate what happens with both the Γ_1 statistic and the correlation as the curve is extended with subsequent cycles.

3.9 If G_x is the k-nearest-neighbor graph on X and G_y is the one on Y, with intersection graph (the graph defined by the shared edges) G, and $G_{x,y}$ is the k-nearest-neighbor graph on the (x, y) pairs, show that G is a subgraph of $G_{x,y}$. Is it necessarily an induced subgraph?

3.10 Consider data generated uniformly on a "plus" (two rectangles of length 1 and width $\frac{1}{4}$ meeting perpendicularly at their centers). For $n = 100$ devise and run an experiment to see if the value of Γ_1 is significantly greater than expected. Be explicit as to what "expected" means.

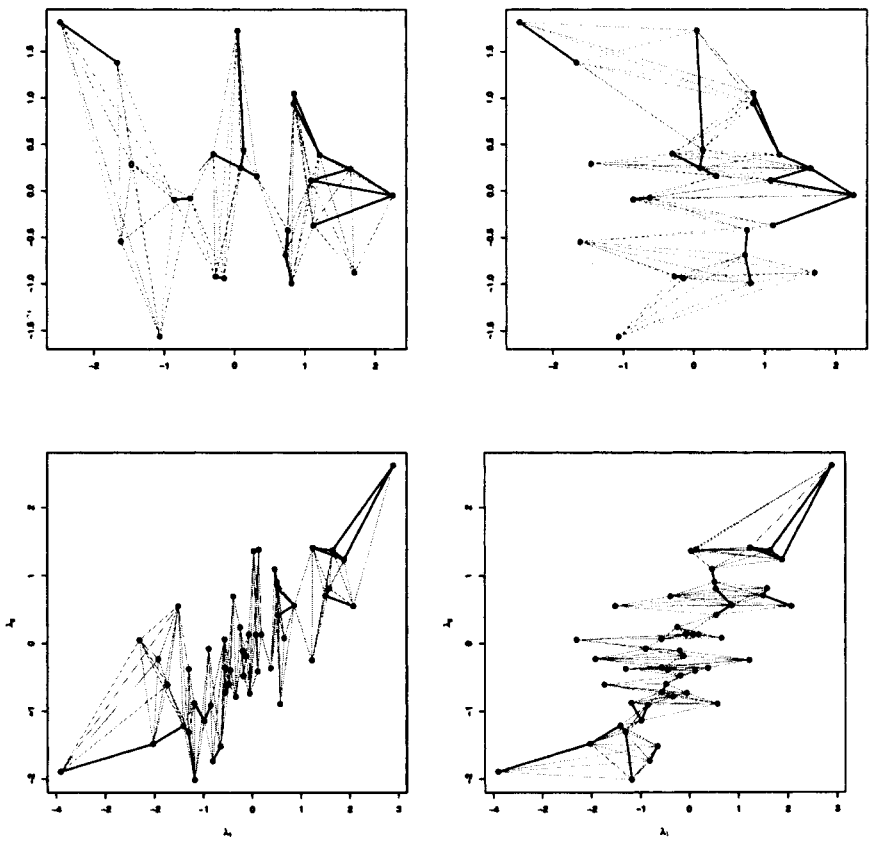


Fig. 3.9 Standard normal data (top) and normal data with a $\text{cov}(x, y) = 0.6$ (bottom) are shown with their associated KNN_X (left) and KNN_Y (right) graphs (edges in gray). The edges common to both graphs are indicated as solid lines. There are 17 and 23 of these common edges, respectively.

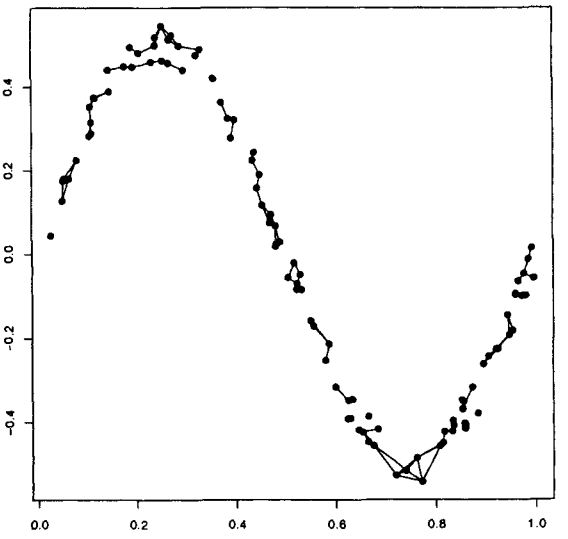


Fig. 3.10 Observations from an "S"-shaped distribution, with the edges common to the KNN_X and KNN_Y graphs.

Moment Distributions In order to utilize the statistic, we need to know the distribution of the statistic under the null hypothesis. One way to do this would be empirically, whereby one estimates the rejection intervals through simulations. This is generally not the preferred method if theoretical results are obtainable, and so in this and the next section we illustrate moment calculations and asymptotic normality results that can be obtained for use in estimating the rejection intervals.

Define two graphs on the pairs $(x_1, y_1), \ldots, (x_N, y_N)$, using one of the graphs discussed above. These graphs, G_x and G_y, then have N vertices in common, as does their intersection. Let e_x be the number of edges in the graph G_x, and similarly define e_y. Let d_i denote the degree of node i. It is easy to see that the average degree is related to the number of edges e in a graph by $\bar{d} = 2e/N$. A second concept that will be needed in the moment calculation is the number of edge pairs that share a common vertex. We denote this as C. Note that

$$C = \sum_{i=1}^{N} d_i(d_i - 1)/2. \tag{3.9}$$

The moments can be represented by functions of the number of edges, e_x and e_y, in the graphs G_x and G_y, and the number of pairs sharing vertices, C_x and C_y. Lemma 3.1 gives the formula for the mean of Γ_1 from Equation (3.8), in terms of these graph parameters.

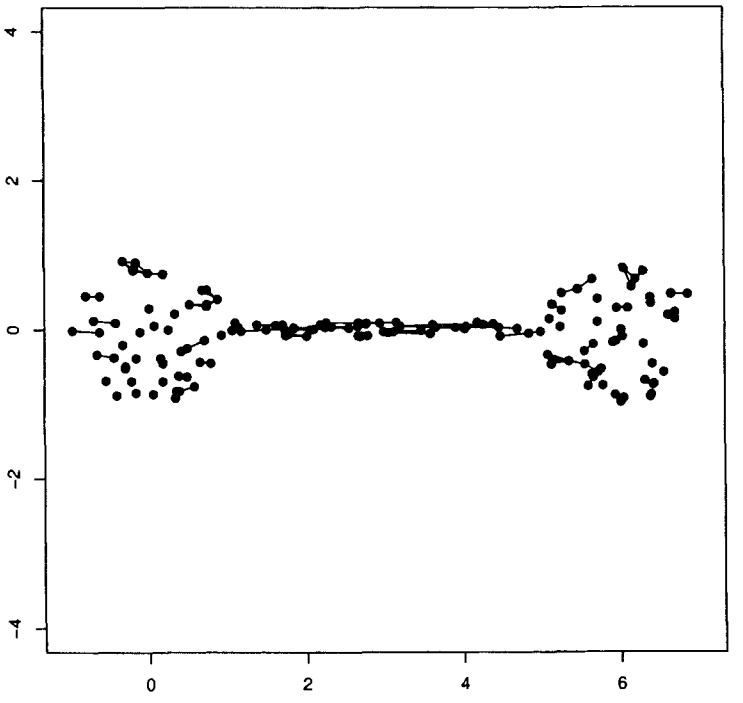


Fig. 3.11 Data drawn uniformly in two balls (50 observations each) and uniformly on the thin rectangle between the balls (50 observations). The edges used in the computation of Γ_1 are shown, with 34 and 37 in the two balls and 61 in the "handle". There are a total of 132 edges in the graph.

Lemma 3.1 (Friedman and Rafsky).

$$E[\Gamma_1 | e_x, e_y] = \frac{2e_x e_y}{N(N-1)} \equiv \mu_{\Gamma_1}. \tag{3.10}$$

Proof. The calculation is simple once one makes the following observation. Label the edges of G_x arbitrarily and define the random variable z_i, for $1 \leq i \leq e_x$, as

$$z_i = \begin{cases} 1 & \text{if i} \in G_y \\ 0 & \text{otherwise,} \end{cases} \tag{3.11}$$

Then clearly

$$\Gamma_1 = \sum_{i=1}^{e_x} z_i, \tag{3.12}$$

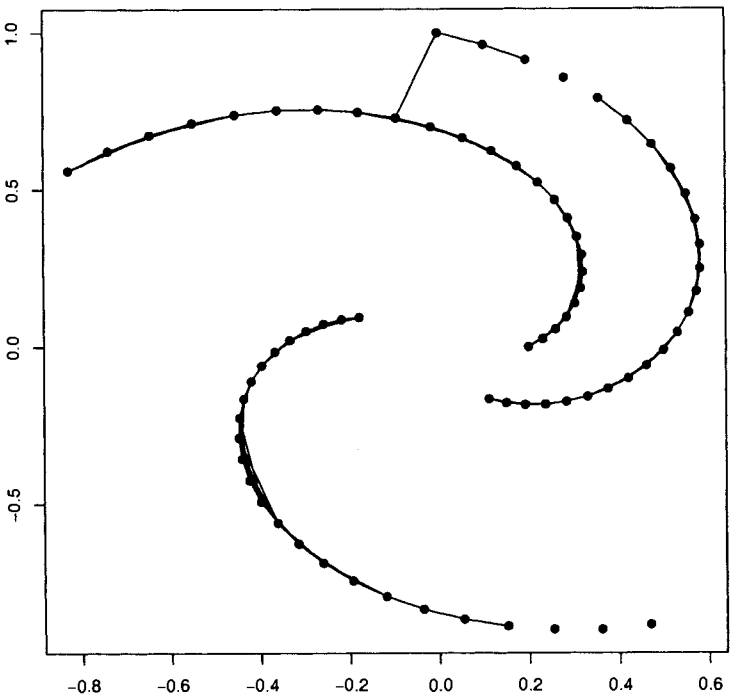


Fig. 3.12 A dataset of spirals, with the edges that define Γ_1.

which implies that

$$E[\Gamma_1] = \sum_{i=1}^{e_x} E[z_i] = e_x P[z_i = 1]. \tag{3.13}$$

Now,

$$p \equiv P[z_i = 1] = \frac{\#\text{ edges in } G_y}{\text{Total possible edges}} = \frac{2e_y}{N(N-1)}. \tag{3.14}$$

Substituting (3.14) in (3.13) completes the proof. □

Lemma 3.2 (Friedman and Rafsky).

$$\begin{aligned} Var[\Gamma_1 | e_x, e_y, C_x, C_y] \;=\; & \mu_{\Gamma_1}(1-\mu_{\Gamma_1}) + \\ & \frac{4C_xC_y}{N(N-1)(N-2)} + \\ & \frac{4(e_x(e_x-1)-2C_x)(e_y(e_y-1)-2C_y)}{N(N-1)(N-2)(N-3)}. \end{aligned} \tag{3.15}$$

Proof. Since the z_i are Binomial with probability p, we have

$$Var[z_i] = p(1-p), \tag{3.16}$$

with p as in Equation (3.14). From Equation (3.12) and the standard formula for the variance of a sum of random variables, we have

$$Var[\Gamma_1] = \sum_{i=1}^{e_x} Var[z_i] + 2\sum_{i<j} cov(z_i, z_j). \tag{3.17}$$

From Equation (3.16) the first term is $e_x p(1-p)$. The second term can be computed using

$$cov(z_i, z_j) = E[z_i z_j] - p^2, \tag{3.18}$$

which gives

$$2\sum_{i<j} cov(z_i, z_j) = 2\sum_{i<j} E(z_i z_j) - e_x(e_x - 1)p^2. \tag{3.19}$$

It suffices to compute $E[z_i z_j]$. This takes on two values, depending on whether (i, j) shares a common defining node in G_y. Recall that Equation (3.9) allows us to calculate the number of edge pairs sharing a node in the complete graph. Thus

$$\begin{aligned} E[z_i z_j | (i,j) \text{ shares a node in } G_y] &= \frac{\#\text{ edge pairs sharing node in } G_y}{\#\text{ edges sharing node in the complete graph}} \\ &= \frac{C_y}{N(N-1)(N-2)/2} \\ &= \frac{2C_y}{N(N-1)(N-2)}. \end{aligned} \tag{3.20}$$

Similarly,

$$\begin{aligned} E[z_i z_j | (i,j) \text{ don't share in } G_y] &= \frac{\#\text{ edge pairs not sharing in } G_y}{\#\text{ edges not sharing in the complete graph}} \\ &= \frac{e_y(e_y-1)/2 - C_y}{N(N-1)(N-2)(N-3)/8} \\ &= \frac{4e_y(e_y-1) - 8C_y}{N(N-1)(N-2)(N-3)}. \end{aligned} \tag{3.21}$$

Combining Equations (3.16)–(3.21) together delivers the result. □

The permutation distribution of Γ_1 is asymptotically normal, under some assumptions about the scores (see [64] for details). Thus one can use the normality to construct tests, without requiring an explicit calculation of the exact distribution, or Monte Carlo simulations to approximate it.

In addition to the work discussed above, [64] define a statistic for answering the question of how well X can be used to predict Y. To do this, they redefine b_{ij} as

follows. For each observation, rank the other observations by their distance (in Y space). This defines the rank $R_i(j)$ of each observation j from observation i. These are then summed over the edges of G_x to produce a new statistic:

$$\Gamma_2 = \sum_{(i,j)\in E(G_x)} R_i(j).$$

Similar results to those discussed here are available for Γ_2.

Problems

3.11 Is k necessary in the definitions of the Γs? What would happen if one used a graph, such as the minimum spanning tree, which had no parameter like k to adjust?

3.12 How do the results obtained in Problem 3.10 compare with the theoretical values?

3.13 Consider the following generalization to higher dimensions. First, reduce to two dimensions via the first two principal components. Then compute Γ_1. Investigate the properties of this, either theoretically or via simulations.

3.3.2 Application: Artificial Nose

The artificial nose data of Section 1.4.1 consists of 1112 observations from a total of 19 chemical mixtures: Air, BTE, Ben, CTe, ClB, Clf, Ker, Oct and WGa alone, with TCE, and TCE alone (see Section 1.4.1 for more detail). Rather than consider the full 1112 vertex KNN graph, we instead consider the 19 vertex graph, where each vertex corresponds to a chemical mixture (all the observations from that mixture), and the distance is defined as the minimum distance between observations in each set:

$$d(S, S') = \min_{s\in S, s'\in S'} \{d(s, s')\}.$$

The graphs for $k = 1, \ldots, 4$ are shown in Figure 3.13.

The graphs are drawn so that the TCE vertices and the no-TCE vertices are separated, since we are interested in how the addition of TCE effects the nose's response. We are primarily interested in the edges that go between the TCE and no-TCE vertices.

In the first graph ($k = 1$), we see that the edges between the two groups join chloroform to chloroform, kerosene to kerosene, benzene to chloroform or kerosene to benzene. These then are the closest chemicals, the ones on the decision boundary between TCE and no-TCE.

Considering two neighbors ($k = 2$) adds a number of within-class edges, and three new edges between classes: kerosene to octane, kerosene to air, and chlorobenzene to chloroform. As k increases, the number of edges between classes increases.

From this, we infer that the decision region is straddled by chloroform and kerosene (with and without TCE). Note that air is close to kerosene and octane (presumably the low concentration observations of these latter). Also, every chemical combination

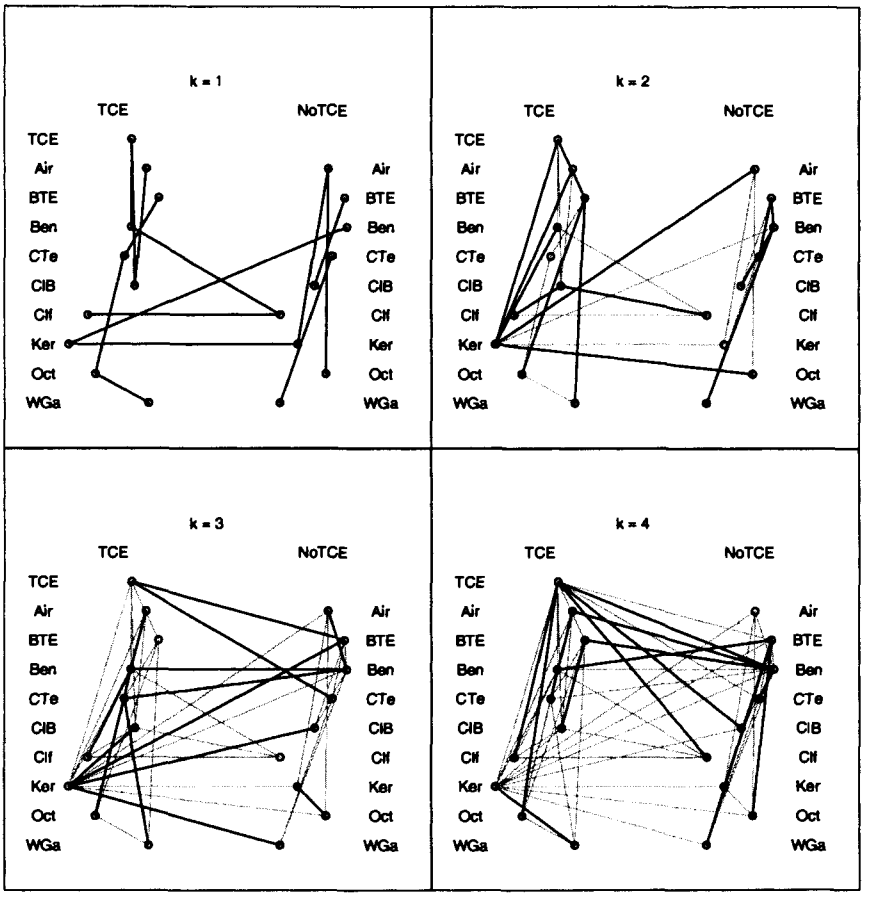


Fig. 3.13 KNN graphs for the artificial nose data of Section 1.4.1. In these plots each node corresponds to all observations from a specific chemical mixture, with the distance computed as the minimum distance between observations in the set. In each successive plot the new edges are drawn in dark lines, while the edges from the previous graph are drawn in gray.

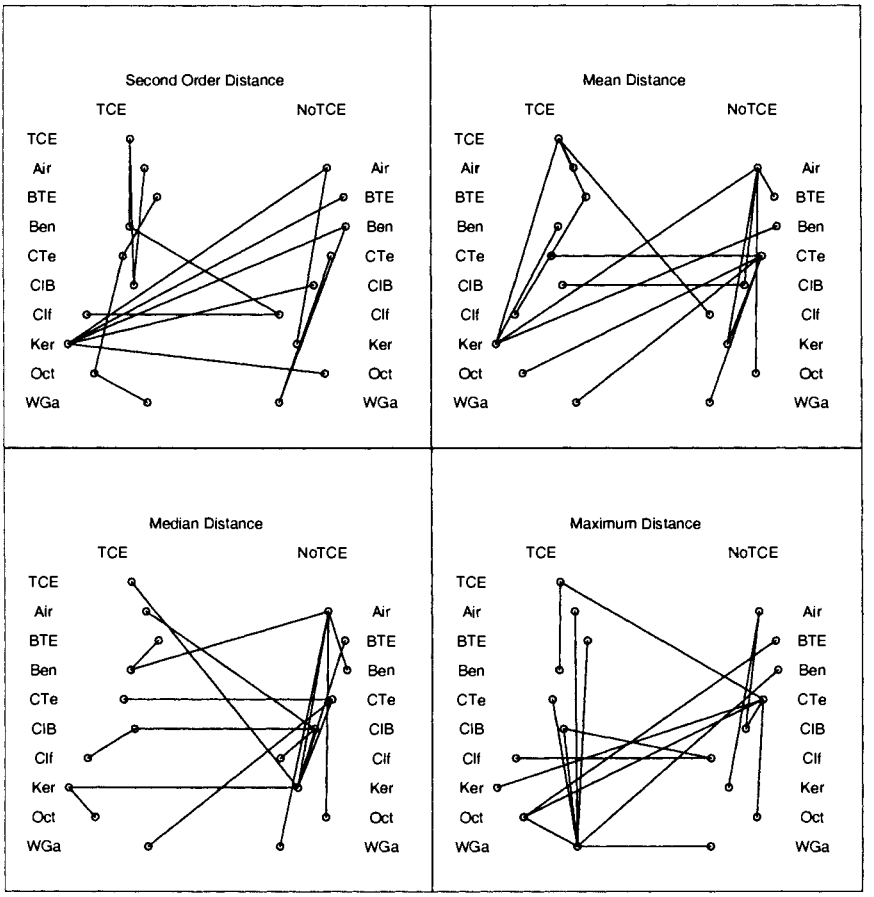


Fig. 3.14 Nearest-neighbor graphs for the artificial nose data of Section 1.4.1, using different distance metrics between sets. In these plots each node corresponds to all observations from a specific chemical mixture, with the distance between sets computed in four different ways: Upper Left: second smallest distance between observations in each set; Upper Right: average distance between observations in each set; Lower Left: median distance between observations in each set; Lower Right: maximum distance between observations in each set. In each plot the nearest-neighbor graph under the associated distance is plotted.

except Coleman fuel + TCE is within 4 neighbors of the other class. Analysis of this type can lead to some limited insight into the structure of the high-dimensional data, and can provide hypotheses for further investigation.

Similarly, we can analyze the data using different set-distance functions. For example, in Figure 3.14 we have the nearest-neighbor graphs under several different distances. The upper left shows the distance defined as the second smallest distance between, observations in the sets, while the other distances are the average, median, and maximum of the distances between observations in each set. These can be compared with the upper left plot of Figure 3.13, which uses the minimum distance. From this type of analysis one can discover some of the structure of the decision boundary.

3.3.3 Application: Outlier Detection

Let $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ be independent, identically distributed with distribution function F and density f (assumed to exist). Let G_k be the mutual k-nearest-neighbor graph (MKNN) on X. Define the support of F to be

$$\mathcal{S} = \{x \in \mathbb{R}^d : f(x) > 0\}. \tag{3.22}$$

Reference [29] defines a statistic to use to determine whether $\mathcal{S}$ is connected. This statistic is defined to be

$$\hat{k}_{d,n}(\mathcal{S}) = \underset{k \in \mathbb{Z}}{\operatorname{argmin}}\{G_k \text{ is connected}\}. \tag{3.23}$$

Theorem 3.1 (Brito et al.). *Assume $\mathcal{S}$ is connected and grid compatible with grid constant α and that there exist a_1 and a_2 such that $0 < a_1 \leq f(x) \leq a_2$ for all $x \in \mathcal{S}$.*

Then, there exists a positive constant c (depending only on a_1, a_2, and α) such that $\hat{k}_{d,n}(\mathcal{S}) \leq c \ln N$ almost surely for large N.

The theorem assumes the support is grid compatible, which is a technical regularity assumption, essentially that the support can be covered by small cubes in a nice way. The details are in [29], and we will not dwell on them.

Theorem 3.2 (Brito et al.). *Let $\mathcal{S}_1, \ldots, \mathcal{S}_m$ be disjoint grid compatible compact sets in $\mathbb{R}^d$. Let F_i have support in $\mathcal{S}_i$, $F = \sum_{i=1}^{m} \beta_i F_i$ for some fixed $\beta_i > 0$. Suppose the independent sample $X \sim F$.*

Then, there exists a positive constant c such that for $k = c \ln n$ and large enough n, the connected components of G_k are $X \cup \mathcal{S}_i$ almost surely.

The theorems suggest a log linear model for the expected value of the statistic $k_{d,n}$, and so [29] performed a series of simulation experiments to determine the parameters of the model

$$E(k_{d,n}) = a_d + b_d \ln n. \tag{3.24}$$

The authors provide tables for various values of d, but for illustration purposes we will only need the values for $d = 2$: $E(k_{2,n}) \sim 4.341 + 0.430 \ln n$. In order to determine significance, the authors propose the estimated standard deviation of the statistic, which for $d = 2$ is 1.313.

Reference [29] proposes the following test for the null hypothesis that the data consists of a single cluster, with no outliers:

1. Estimate $\hat{E}_{d,n}$ using Equation (3.24) and coefficients from their table (in the case $d = 2$, these are $a = 4.341$, $b = 0.430$ and $\hat{\sigma}_2 = 1.313$). Compute $z = \hat{E}_{d,n} + 3\hat{\sigma}_d$. Set $k_{\max} = \lceil z \rceil$.

2. Build the MKNN graph for $k = k_{\max}$.

3. If the graph is connected, conclude that there is insufficient evidence (at approximately the 0.1 level) to reject the null hypothesis of no clustering/outliers.

We will argue below that the test is actually more appropriate for the detection of outliers than clusters; it is quite sensitive to outliers.

By this algorithm, our example in Figure 2.1 produces a result of $z = 10$, thus indicating that we do not have evidence that the data clusters or contains outliers (recall that the MKNN for these data is connected from $k = 5$ on).

A similar approach is the purely nonparametric one: collect many samples from H_0 and compute some statistic (for example, the smallest k such that the MKNN is connected). Then, if the value observed is greater than, say, 90% of the values from the simulation, one rejects the hypothesis.

One problem is that one must be able to draw from H_0, and it is unclear what this might mean in this case. A reasonable choice, similar to the one made by [29], is to draw from a uniform distribution. One might also draw from a number of other unimodal distributions to see how sensitive the statistic is to the null hypothesis distribution.

We performed this experiment for our points in Figure 2.1. The observed value of the smallest k such that the MKNN is connected is 5. A Monte Carlo replication of 100 data sets of size 35 drawn from a 2-dimensional uniform distribution determined that 90% of the values were less than or equal to 8 (agreeing fairly well with the value provided by the above algorithm of [29]). See Figure 3.15.

It could be argued that the null hypothesis of a uniform distribution is not appropriate for these data. In this case, a reasonable hypothesis might be that the data are normally distributed. We performed the same type of simulation using data drawn from a normal distribution, using the sample mean and variance of the data in place of the mean and variance for the normal distribution. In this case, the 90% quantile is 14. The histogram for this simulation is depicted in Figure 3.16. There were two cases out of 100 in which the simulation returned $k \geq 20$. These were placed in the final bin in the histogram.

An example of a data set in which k is large ($k = 20$) is shown in Figure 3.17. If one deletes the point in the lower left corner, the value of k is reduced to 9. Thus this "outlier" causes the statistic to grow from 9 to 20.

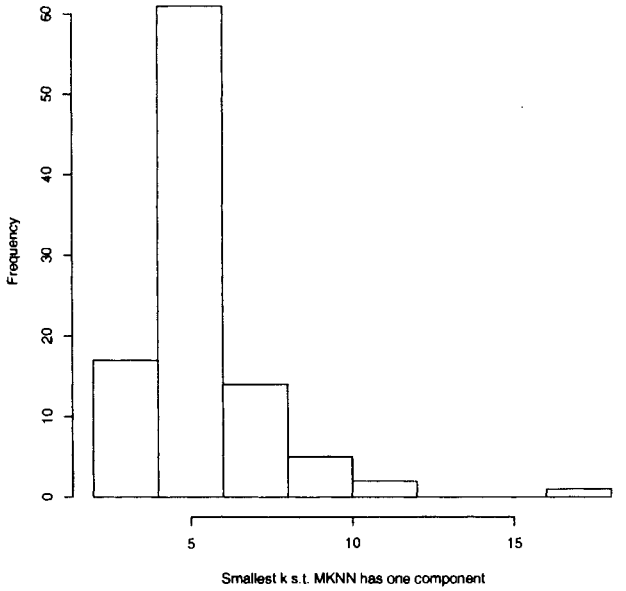


Fig. 3.15 A histogram from 100 simulations. In each simulation, 35 2-dimensional uniform random variates were generated, and the simulation recorded the smallest k such that the resulting MKNN was connected. These are the values plotted in the histogram.

The problem, in this case, is partly a result of the tails of the normal distribution, and partly the elliptical shape of the distribution. Using the Mahalanobis distance, the MKNN graph for the data in Figure 3.17 has only two components at $k = 6$, instead of 9, and a single component at $k = 14$, compared to 20. Reference [29] actually performs similar simulations, but they truncate the tails of the normal distribution to eliminate outliers. The interested reader is encouraged to try this to determine whether this results in a different result for these data.

This illustrates that the method of Brito et al. is indeed an algorithm for the detection of outliers. The value of the statistic is dramatically effected by the existence of observations in the tails of the distribution. To the extent that clusters are defined to be groups of observations which are far from the rest of the data, the method will detect the clustering. However, "observations which are far from the rest of the data" is a reasonable layman's definition of "outlier".

There are two examples in [29] of data sets for which traditional clustering methods tend to perform poorly, and the MKNN method produces the "correct" clustering. This does argue against our claim that the method is really an outlier detection algorithm. It also illustrates the difficulty in defining "cluster," which is one of the reasons cluster analysis is so difficult. In cluster analysis, one (explicitly or implicitly through the

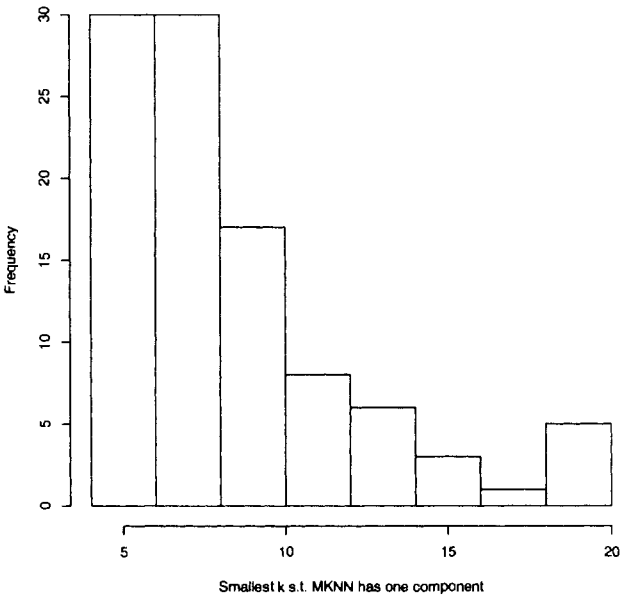


Fig. 3.16 A histogram from 100 simulations. In each simulation, 35 2-dimensional normal random variates were generated, and the simulation recorded the smallest k such that the resulting MKNN was connected.

choice of the algorithm used) defines what is meant by "cluster" and then proceeds to try to find groups of observations that fit one's definition of cluster. This can be a bit of an art, since a given application may not provide sufficient insight into what definition of "cluster" is appropriate.

Problems

3.14 Redo the simulation of Figure 3.15, drawing from a d-dimensional uniform distribution and investigate the dependence of the distribution on the dimensionality of the data.

3.15 How does the distribution of the experiment of Figure 3.16 depend on the covariance of the normal distribution?

3.16 Redo the simulation of Figure 3.16, drawing from a d-dimensional normal distribution and investigating the dependence of the distribution on the dimensionality of the data.

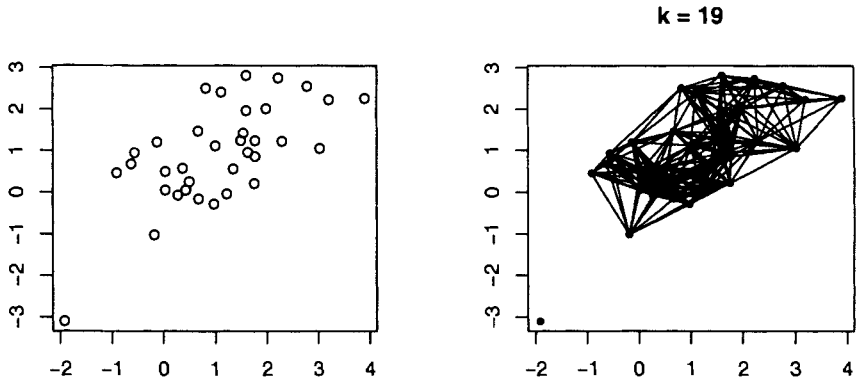


Fig. 3.17 A data set, drawn from a bivariate normal distribution, for which the smallest value of k for which the MKNN is connected is 20. The data are plotted on the left, and the graph for $k = 19$ is shown on the right.

3.3.4 Application: Dimensionality Reduction

Isomap is a methodology for dimensionality reduction described in [196]. It can be thought of as a form of nonlinear multidimensional scaling that utilizes a k-nearest-neighbor graph to define the distances.

Given a point set $\mathcal{X} = \{x_1, \ldots, x_n\}$, one first constructs the k-nearest-neighbor graph on $\mathcal{X}$. This is used to define the inter-point distance matrix as follows. If x_i and x_j are not in the same graph component then set $d(x_i, x_j) = \infty$. Otherwise, define $d(x_i, x_j)$ to be the sum of the lengths of the edges on the shortest path (in the graph) from x_i to x_j. Note that, as usual, we are identifying the graph vertices with the underlying points.

This idea is illustrated in Figure 3.18. A spiral data set is shown along with its nearest-neighbor graph. The dotted line connects two points that are relatively close in Euclidean distance (0.63) yet are very far in graph distance (4.26).

Once the inter-point distance matrix is computed, multidimensional scaling (see, for example, [127, 27]) is applied to reduce the dimensionality. Other techniques which operate on inter-point distance matrices could be used as well, such as various clustering or classification algorithms.

Figure 3.19 illustrates one such algorithm with another spiral data set. The graph distance is computed using 1- and 3-nearest-neighbor graphs. The resultant inter-point distance matrix is processed through a singular value decomposition to find the dominant eigenvectors, which are then plotted on the right. As can be seen, the nearest-neighbor graph pulls the two spirals apart, as would be expected. The

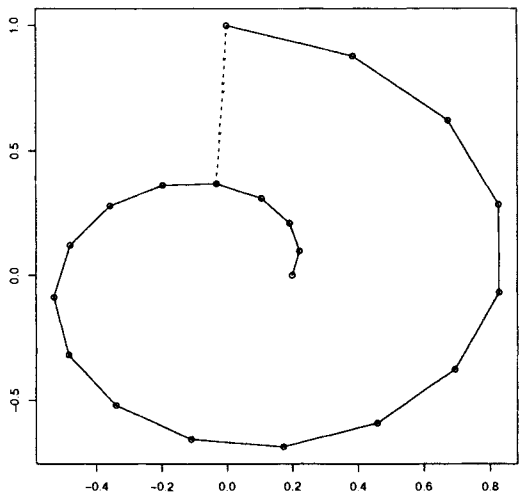


Fig. 3.18 Spiral data set illustrating the idea of the Isomap procedure. The nearest-neighbor graph for the data is shown in solid lines. The dotted line connects two points that are relatively close in Euclidean distance, but very far in graph distance.

3-nearest-neighbor graph puts the spirals together, as a result of the edges between them.

Other graphs than the k-nearest-neighbor graph can be used in the isomap procedure. For example, [196] suggest a sphere graph (see Sections 3.6, 3.7 and 4.1.1 for some examples), and one could just as easily use mutual k-nearest-neighbor, or any other graph that we investigate in this book.

Another issue is the possibility of infinite distances in the inter-point distance matrix. These correspond to different components in the graph which defines the distances. These must be treated separately; a component with few observations could be treated as outliers, while larger components would be viewed as distinct clusters. Increasing the value of k can result in a connected graph, although as Figure 3.17 shows, this may not be desirable in some cases.

Figure 3.20 shows the isomap results for the nose data, using a 4-nearest-neighbor graph for to define the inter-point distance matrix; the value 4 was chosen as the smallest k such that the k-nearest-neighbor graph was connected. The dimensionality reduction method is classical multidimensional scaling (the function "cmdscale" in R, see [184, 127, 27]). Figure 3.21 shows a zoom of the region of maximum overlap. One can see that the two classes have a strong relationship in these plots, although they are not perfectly overlapped. Recall that the data consist of 38 function-valued features, and so this is a considerable reduction in dimensionality.

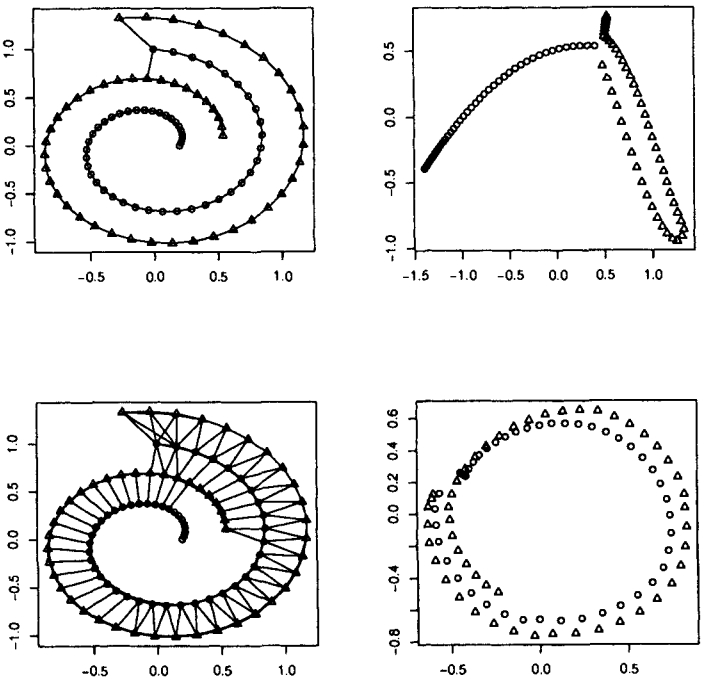


Fig. 3.19 Isomap for spiral data using k-nearest-neighbor graphs, for $k = 1, 3$. The scatter plots on the right correspond to the first two principal components of the inter-point distance matrix, where the distance is computed using the graph distance method described in the text. The two spirals are indicated by different plotting symbols.

These results can be compared to multidimensional scaling applied to the original inter-point distance matrix (Figures 3.22 and 3.23). These two sets of figures have pretty much the same character; it is not clear that one of these approaches has anything to offer over the other, for this data set. However, they do provide different information, and can provide suggestions for areas of further investigation.

Consider, for example, the looping structure in the upper left of Figure 3.20, within the region defined by the gray polygon. The same observations are indicated by the gray polygon in Figure 3.22. These observations correspond to chloroform. The different structures in the two plots indicate different interpretations of the topology of the high-dimensional data. Further investigation of this structure might bear fruit.

As can be seen in Figure 3.24, these chloroform observations have essentially the same degree distribution as the full graph. At least by this measure, they are not noteworthy. The 4-nearest-neighbor graph is difficult to visualize, due to the large number of vertices and high amount of interconnection. However, we make an attempt

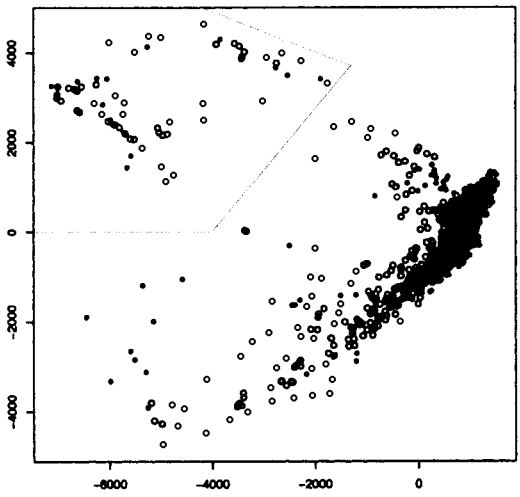


Fig. 3.20 Isomap of the nose data, using a 4-nearest-neighbor graph to define the distances.

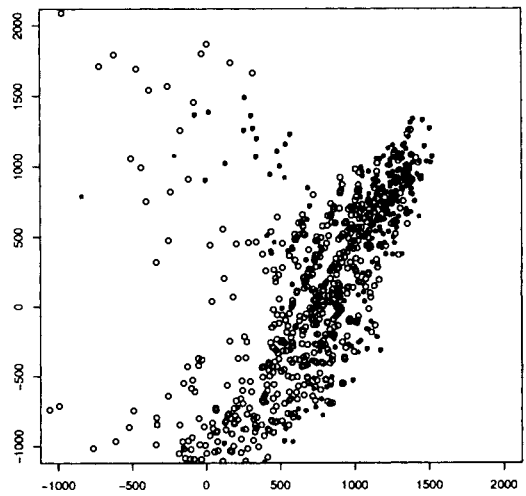


Fig. 3.21 A zoom of the data in Figure 3.20.

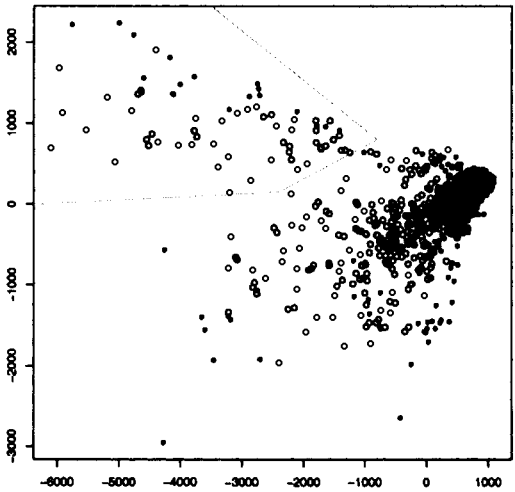


Fig. 3.22 Multidimensional scaling for the nose data.

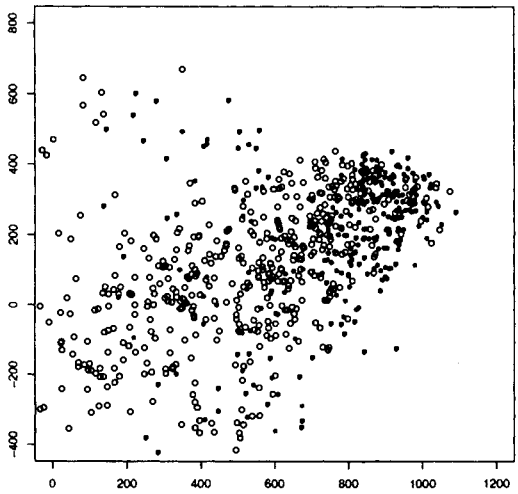


Fig. 3.23 A zoom of the data in Figure 3.22.

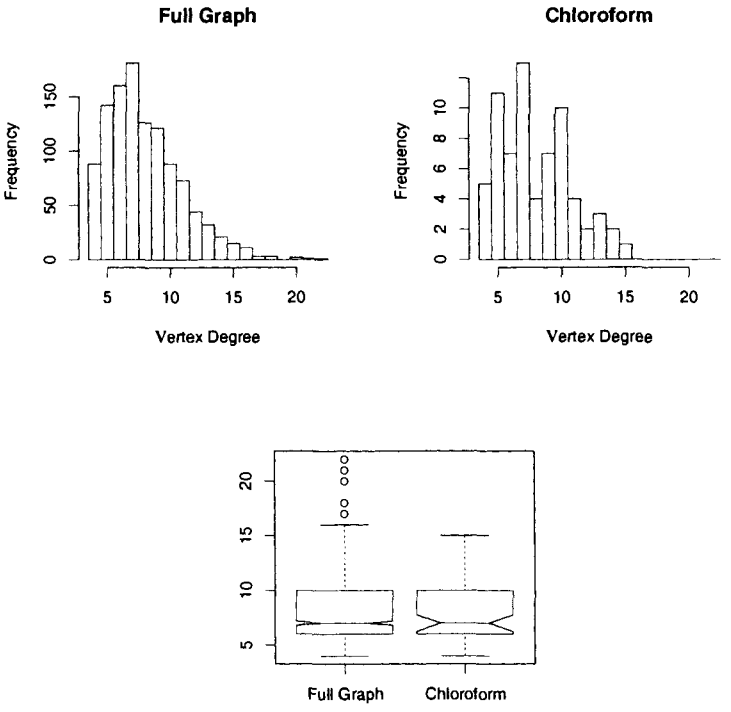


Fig. 3.24 Distributions of the vertex degrees for the 4-nearest-neighbor graph of the nose data, and the "loop structure" indicated by the polygon in Figure 3.20, corresponding to chloroform.

to do this in Figure 3.25. The graph consists of a large, highly interconnected group, another highly interconnected group of chloroform nodes, and a group of eight vertices that connect the two groups. These are pictured in the figure. The edges between the intermediary group and the others are emphasized in this plot, and the important nodes are plotted with symbols indicating the chemical compound associated with the node. Very little if any of the graph structure is discernible within the main groups. Better graph display techniques might elicit other interesting structure. For example, Figure 3.26 displays the same graph using the points defined by the isomap procedure. While interesting, the utility of displays of this type are limited.

A final view of the graph is given in Figure 3.27. We have utilized multidimensional scaling on the isomap inter-point distance matrix to produce a three dimensional projection, rendered as a stereo pair. By bringing the two pictures together, one can discern the third dimension, allowing the elucidation of further structure in the graph.

This exercise indicates some of the utility of graph methods for this problem. The separation of the chloroform from the main group is discoverable without graph

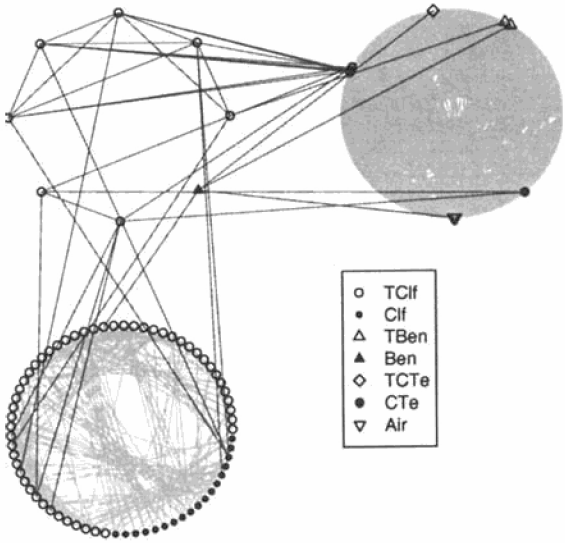


Fig. 3.25 The 4-nn graph for the nose data. The lower left group of vertices are the chloroform nodes discovered through the multidimensional scaling. The group of eight vertices in the upper left are vertices that connect this group to the rest of the graph, indicated in the upper left.

methods, but by analyzing the graph we have discovered that there is a tenuous link between the main group of chloroform and the rest of the observations. This link, consisting of seven chloroform and one benzene vertex, might be worthy of further study.

Finally, we consider the observations consisting of the lowest concentrations of confusers. This data set, consisting of 80 TCE and 80 non-TCE observations, is depicted in Figure 3.28. In this figure the isomap procedure has been applied to reduce the data to two dimensions (to facilitate visualization via a scatter plot). The different chemical confusers are indicated by the symbols (note that there is no benzene in this dataset, due to a lack of sufficient low-concentration exemplars of this chemical). As can be seen, this projection does a very good job of separating the different chemical confusers, and of discovering the unique geometry of chloroform and kerosene,

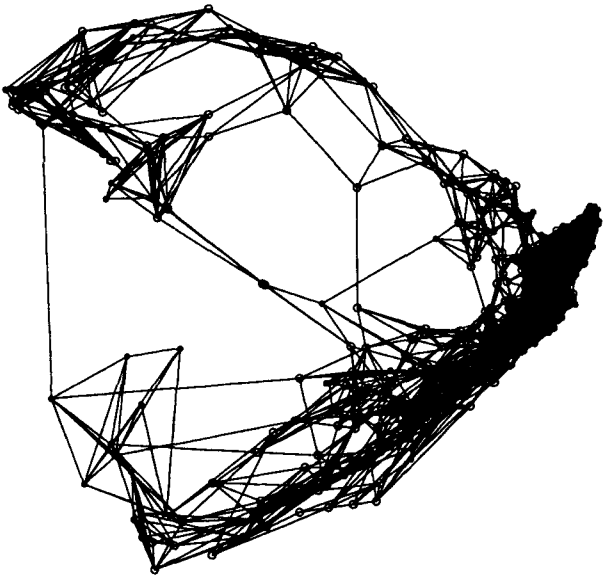


Fig. 3.26 The 4-nn graph for the nose data, using the points defined by the isomap procedure, Figure 3.20.

although it is not optimal for the problem of discriminating between TCE presence and absence.

Problems

3.17 Generate 1000 observations uniformly on the unite sphere in $\mathbb{R}^7$. Compute the 5-nearest-neighbor graph, and use this to compute the inter-point distances as in the isomap procedure. Project to $\mathbb{R}^2$ using multidimensional scaling. Discuss the results.

3.18 Investigate the isomap distance calculation procedure on various 2-dimensional shapes. Can you suggest rules-of-thumb for the choice of k in the k-nearest-neighbor graph?

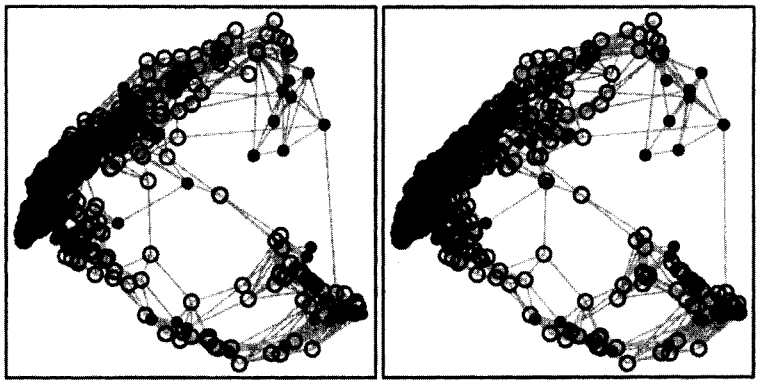


Fig. 3.27 A stereo pair of the 4-nn graph for the nose data, using the points defined by the isomap procedure, Figure 3.20. Here, the multidimensional scaling has provided a projection to $\mathbb{R}^3$, which is used to define the placing of the vertices.

3.4 RELATIVE NEIGHBORHOOD GRAPHS

Relative neighborhood graphs have been investigated by a number of researchers. See in particular [104], [144] and [155]. The idea is to connect points if they are "relatively close" to each other. The concept of "relative closeness" is defined in terms of the lune between the points. This was defined for planar points in Definition 2.8, but the general definition is the same:

Definition 3.11. *Let $V \subset \mathbb{R}^d$. For any pair $(p, q) \in V \times V$, the set $\Lambda_{p,q} = B(p, d(p,q)) \cap B(q, d(p,q))$ is called a* **lune**. *See Figure 3.29.*

Definition 3.12. *The* **relative neighborhood graph** *with vertex set $V \subset \mathbb{R}^d$, denoted $RNG(V)$, is defined to be the graph on V with edge set defined by the lunes; $pq \in E(RNG(V)) \iff \Lambda_{p,q} \cap V = \emptyset$.*

For this definition, it is important that the lune be open (the intersection of open balls). Otherwise p and q would always be in the lune, and the definition would need to be modified appropriately.

The RNG for the points of Figure 2.1 is depicted in Figure 3.30. This also illustrates the relative neighborhood graph calculation. The balls are drawn for two pairs of vertices, indicated as circles and triangles in the plot. In the first case, the relative neighborhood, the intersection of the two balls, contains no points of the data set, and so there is an edge in the relative neighborhood graph, as indicated by the heavy line segment. In the second case, the intersection of the balls contains data points, and so the potential edge, indicated by the dotted line segment, is missing from the relative neighborhood graph. Figure 3.31 shows the RNG as a subgraph of the Delaunay triangularization.

The relative neighborhood graph was first discussed in the context of pattern recognition in [200], and further illustrated in [202]. The idea was to use the graph to outline

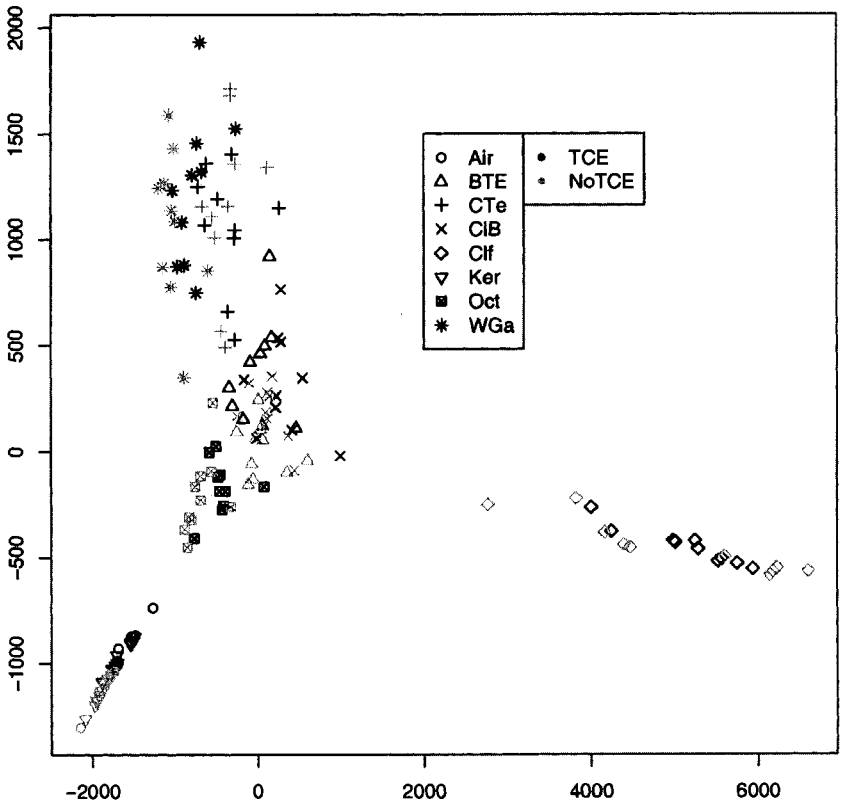


Fig. 3.28 An isomap projection for the low concentration nose data. (Figure courtesy of Jeff Solka.)

planar figures; in a sense to produce the "sketch" associated with the points in the plane. This is illustrated for a few simple figures in Figure 3.32. A much larger collection is provided in [202].

Using the terminology of Definition 3.1, we see that for a relative neighborhood graph RNG(V), we have $U_{p,q} = \Lambda_{p,q}$, the lune between p and q, and $\mathcal{P}$ is the property that the lune contains no elements of V.

It is immediate from the definition that pq is an edge in RNG(V) if there is no triangle pqv with $v \in V \setminus \{p, q\}$ such that pq is strictly the longest edge in the triangle. This is the definition first given in [200]. Two points x and y in V are called "relatively close" if $d(x, y) \leq \max_{z \in V}(d(x, z), d(y, z))$.

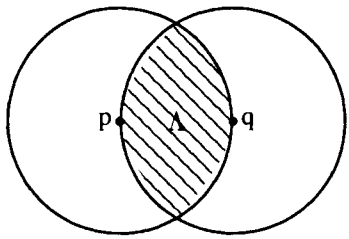


Fig. 3.29 The region of influence for a relative neighborhood graph (shaded region), corresponding to the lune between the two points.

Reference [200] proves two theorems, which we reproduce here as a single theorem. These both assume that the points lie in the plane, that is that $d = 2$, although the theorem is true in general.

Theorem 3.3 (Toussaint). *The relative neighborhood graph is a superset of the minimum spanning tree. The RNG is a subset of the Delaunay triangularization (DT). In other words, MST < RNG < DT.*

Proof. We first show that MST $<$ RNG. Consider Figure 3.33. Let a and b be arbitrary points of the MST. Let $R_a = B(a, d(a,b))$ and $R_b = B(b, d(a,b))$. R denotes the lune $R_a \cap R_b$, and $B = \partial R$. Let c be any point (vertex in V). If $c \in B$, then $d(a,b) = d(c,b)$ or $d(a,b) = d(c,a) \Rightarrow$ the MST is not unique.

If $c \in R$ then $d(a,c) < d(a,b)$ and $d(b,c) < d(a,b) \Rightarrow ab \notin$ MST. Thus if $ab \in$ MST then all other vertices must be in R^c. But then, by definition, ab is an edge in RNG. $\therefore$ MST $\subset$ RNG.

Now we show that RNG $<$ DT. Let $ab \in$ E(RNG) and $R = R_a \cap R_b$. Then no other points of V lie in R. Let T_a and T_b denote the tiles of Voronoi regions for a and b, respectively (see Figure 3.34). Let c and d line in B. Clearly, if all the points on B are on one side of the line segment ab, then T_a and T_b share a side, and hence $ab \in$ DT. So assume there exist at least one point on either side of ab. The Voronoi lines associated with ca and ad are perpendicular bisectors of ca and ad, respectively. Since the three points a, c and d all lie on a circle centered at b, T_a and T_b always share a side of nonzero length. If c or d are in R^c, the side shared by a and b will increase. Hence $ab \in$ DT. □

Theorem 3.3 allows us to put bounds on the size of the RNG, at least for points in the plane. This is presented in the following corollary.

Corollary 3.1. *If $N_e = |E(RNG)|$ for a relative neighborhood graph on n points in the plane, then $n - 1 \leq N_e \leq 3n - 6$ and is thus $O(n)$.*

Proof. The MST contains $n - 1$ edges, $\therefore$ by Theorem 3.3 we have $n - 1 \leq N_e$. It is well known (see [216]) that a planar graph (a graph that can be represented in the

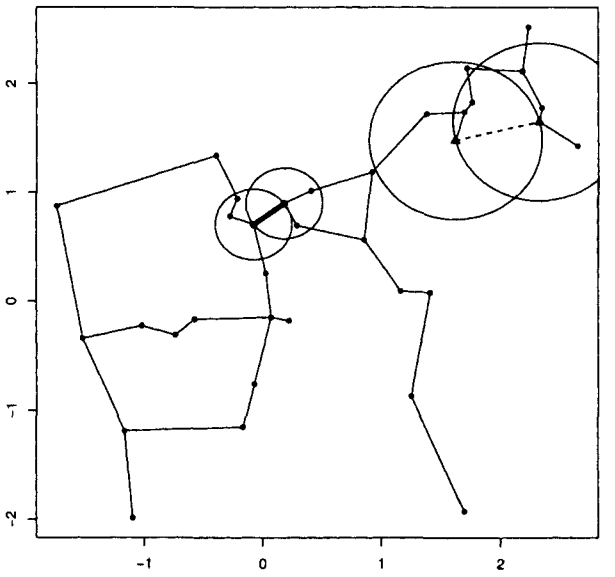


Fig. 3.30 The relative neighborhood graph defined by the set of points in Figure 2.1. The balls defining the edges are shown for two cases: one where there is an edge in the relative neighborhood graph (shown in bold), and one where there is no edge (indicated by a dotted line segment).

plane without crossing edges) with $n \geq 3$ vertices can contain at most $3n - 6$ edges. Further, this bound is attained for the Delaunay triangularization. Thus, by Theorem 3.3, $N_e \leq 3n - 6$. □

Theorem 3.3 provides an efficient algorithm for the calculation of relative neighborhood graphs, given that the Delaunay triangularization has already been calculated. This is given in Algorithm 3.1.

Some properties of the planar RNG can be found in [206]. Discussions of relative neighborhood graphs in different metrics are provided in [147] and [118]. Relative neighborhood graphs in problems where the variables are mixed (continuous, categorical and qualitative) are discussed in [97].

Problems

3.19 Compute the area of the lune between two points, as a function of their distance.

3.20 What is the distribution of the distance between two points drawn independently from $U[0, 1]^2$.

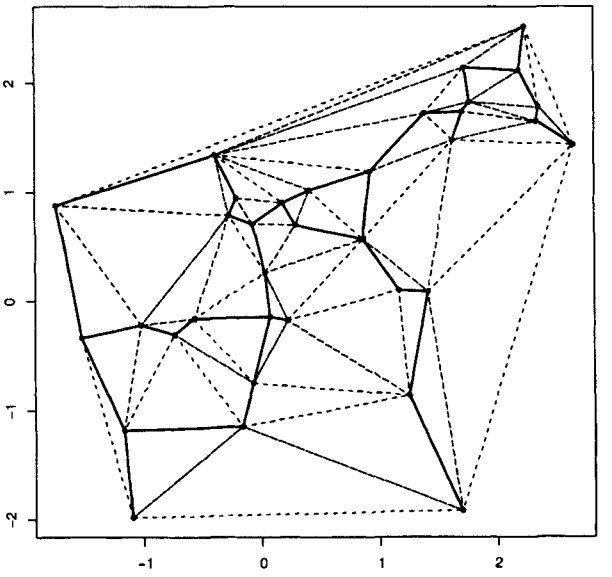


Fig. 3.31 The relative neighborhood graph plotted with the Delaunay triangularization of the points.

input : The Delaunay triangularization $\mathcal{D}$ of a set of points
output : A relative neighborhood graph

initialize: $T = \emptyset$.
foreach *edge* $vw \in E(\mathcal{D})$ **do**
  if $\Lambda(v, w) \cap V(\mathcal{D}) = \emptyset$ **then**
    $T = T \cup \{e\}$;
  end
end

Algorithm 3.1: Relative neighborhood graph calculation

3.21 Let $x_1, \ldots, x_n$ be drawn independently from $U[0, 1]^2$. Using the results of Problems 3.19 and 3.20, what is the probability that there is an edge from x_1 to x_2?

3.22 Are the edges in problem 3.21 independent?

3.23 Prove that a cycle of n vertices is a relative neighborhood graph for $n \geq 3$.

3.24 Prove that a planar graph with $n \geq 3$ vertices contains at most $3n - 6$ edges.

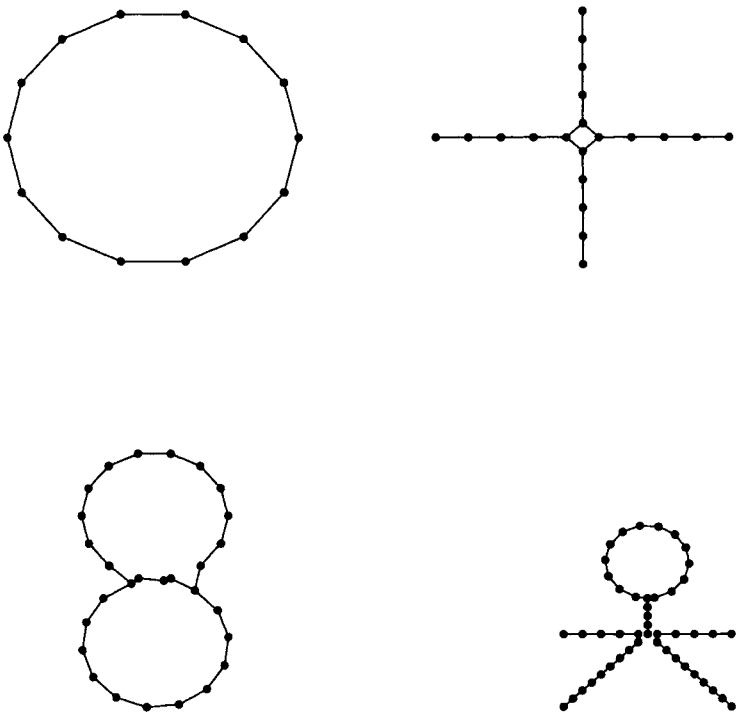


Fig. 3.32 Some examples of relative neighborhood graphs.

3.5 GABRIEL GRAPHS

Gabriel graphs (GG), also known as Least Squares Adjacency Graphs, were introduced by [67], in which they were used for geographic analysis. They are defined by the region of influence depicted in Figure 3.35.

Definition 3.13. *A* **Gabriel Graph (GG)** *on a vertex set* $V \subset R^d$ *is defined to be the graph on* V *whose edges are defined by the region of influence corresponding to the circle separating the two points. That is,* $pq \in E(GG) \iff B\left(\frac{p+q}{2}, \frac{d(p,q)}{2}\right) \cap V = \emptyset$ *(see Figure 3.35).*

Theorem 3.4. $RNG < GG < DT$

Proof. The first part is trivial, see Figure 3.38. We leave the rest as a problem for the reader.

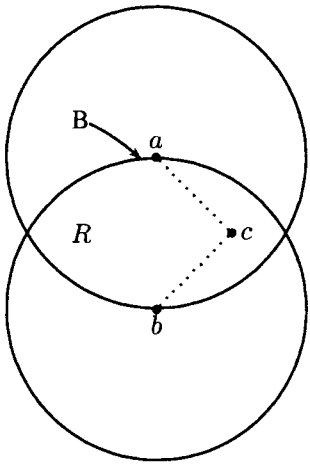


Fig. 3.33 Figure used in the proof of the first part of Theorem 3.3.

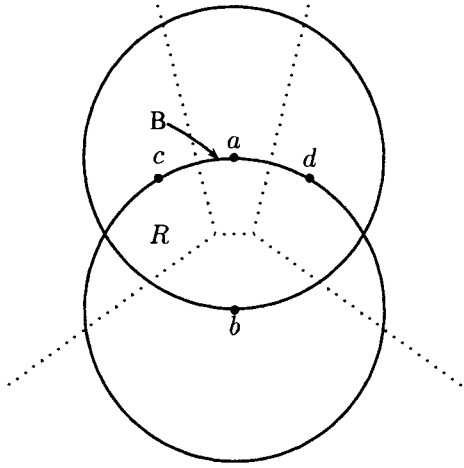


Fig. 3.34 Figure used in the proof of the second part of Theorem 3.3. The dotted lines correspond to the Voronoi diagram.

□

Reference [130] derives some properties of Gabriel Graphs. [97] suggests some generalizations of Gabriel graphs to other metrics. They begin by writing the region of influence in a slightly different form.

$$\text{DISK}(x_i, x_j) = \left\{ x \in V : \left[\rho_2^2(x, x_i) + \rho_2^2(x, x_j)\right]^{\frac{1}{2}}, \leq \rho_2(x_i, x_j) \right\}, \qquad (3.25)$$

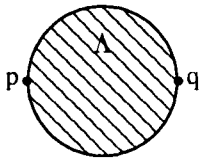


Fig. 3.35 Region of influence, Λ, for a Gabriel graph.

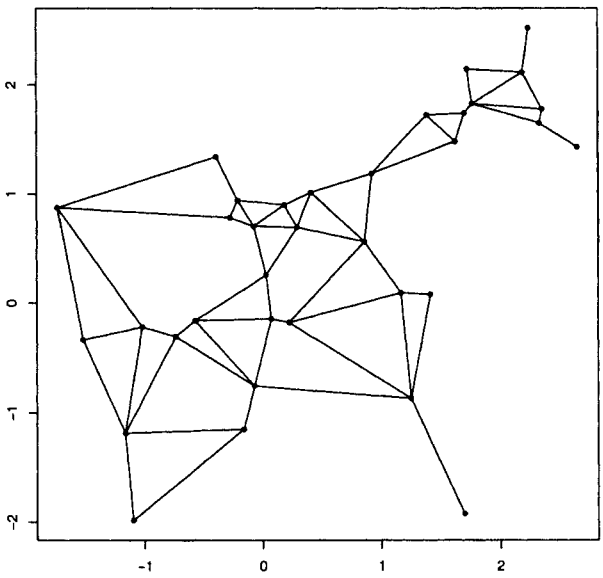


Fig. 3.36 The Gabriel graph for the points of Figure 2.1.

where ρ_2 is the Euclidean (L_2) distance. They then extend this idea to a p-disk as follows.

$$\text{DISK}_p(x_i, x_j) = \left\{ x \in V : \left[\rho_p^p(x, x_i) + \rho_p^p(x, x_j)\right]^{\frac{1}{p}} \leq \rho_p(x_i, x_j) \right\}, \qquad (3.26)$$

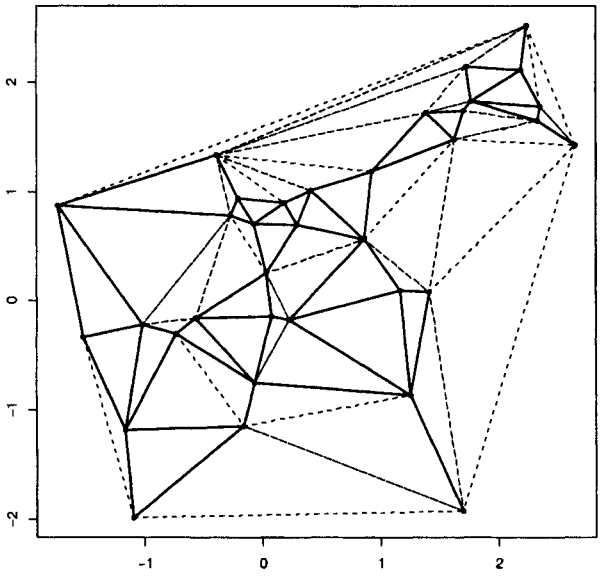


Fig. 3.37 The Gabriel graph plotted with the Delaunay triangularization.

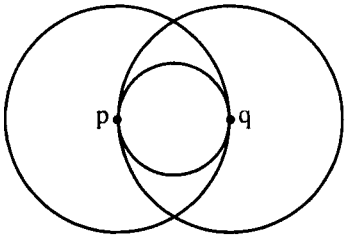


Fig. 3.38 A "proof without words" that RNG < GG.

where

$$\rho_p = \left[\sum_{k=1}^{d} |x_{ik} - x_{jk}|^p \right]^{\frac{1}{p}} \tag{3.27}$$

is the L_p distance. For $p = \infty$ this is taken to be the sup-norm. When $p = 1$ (and $d = 2$) the region of influence is a rectangle with the two points in opposite corners;

when $p = \infty$ the region of influence is a square with the points bisecting opposite sides. See Figure 3.39 for a picture when $d = 2$.

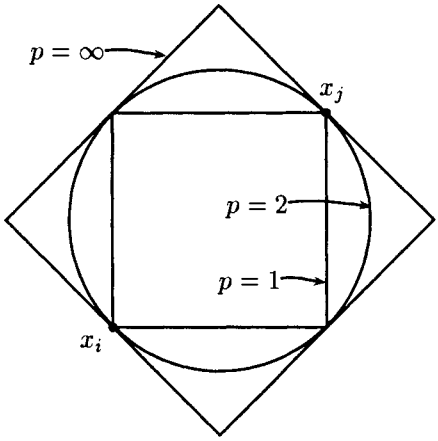


Fig. 3.39 Regions of influence defined by DISK_p for various values of p.

The case $p = 1$ is given a special name, RECT, and can be alternatively defined as the smallest coordinate-oriented rectangle that contains x_i and x_j.

Lemma 3.3. *$RECT(x_i, x_j) \subseteq DISK_p(x_i, x_j)$ for every x_i and x_j.*

Definition 3.14. *The* **Rectangular Influence Graph (RIG)** *is the influence graph whose regions of influence are RECTs.*

Proposition 3.1. *$GG < RIG$.*

Proof. This is immediate from Lemma 3.3. □

From Theorems 3.3 and 3.4 we have MST < RNG < GG < DT. Proposition 3.1 gives MST < RNG < GG < RIG. However, [97] show that it need not be the case that either RIG < DT or DT < RIG.

Another generalization of neighborhood graphs is defined as follows. Let the β-lune $L_\beta(p, q)$ be defined by

$$L_\beta(p,q) = B\left(p\left(1-\frac{\beta}{2}\right) + q\frac{\beta}{2}, \frac{\beta}{2}d(p,q)\right) \cap B\left(q\left(1-\frac{\beta}{2}\right) + p\frac{\beta}{2}, \frac{\beta}{2}d(p,q)\right). \tag{3.28}$$

Then the **β-skeleton** is the graph $G_\beta(V)$ with edges pq iff $L_\beta(p, q) \cap V = \emptyset$. We then have ([149]):

Theorem 3.5 (O'Rourke and Toussaint). *For any $1 \leq \beta \leq 2$ and any L_p metric, $NNG < MST < RNG < G_\beta < GG < DT$.*

Note that when $\beta = 1$ the β-skeleton is the Gabriel graph, and when $\beta = 2$ it is the relative neighborhood graph. This is illustrated in Figure 3.40, in which the

transitions from the relative neighborhood graph to the Gabriel graph is depicted. See [171] for more discussion of β-skeletons. So β provides a natural family of graphs between these two graphs. In Section 3.5.1 we will see another natural generalization of the Gabriel graph that takes its inspiration from alpha hulls.

Problems

3.25 Fill in the details of the proof of Theorem 3.4.

3.26 Prove Lemma 3.3.

3.27 What happens when $\beta < 1$? Is the β-skeleton still a subgraph of the Delaunay triangularization? Draw the β-lune for $\beta = 1, \frac{2}{3}, \frac{1}{2}, \frac{1}{3}$.

3.28 What happens when $\beta > 2$? Is the β-skeleton still a subgraph of the Delaunay Triangularization? Draw the β-lune for $\beta = 2, 2.5, 3, 4$.

3.5.1 Gabriel Graphs and Alpha Hulls

The Gabriel graph provides a natural way to define a "variable α" alpha hull. Fix $\beta \geq 1$ and for each pair of vertices v, w, compute the radius r_{vw} of the disk used in the Gabriel graph. Then define $\alpha = \beta r_{vw}$. For $\beta = 1$ we have the Gabriel graph. For other values of β we have a graph similar to the alpha hull, with a different value of alpha defining each edge in a natural way. Figure 3.41 depicts four values of β on the "circled H" data from Section 2.4.1. When $\beta = 1$ we have the Gabriel graph (upper left) and we see edges between the circle and the H, as well as a lot of structure within the H. As β increases, we see the H and the circle becoming disjoint (distinct components of the graph) and we see less internal structure of the H, and a somewhat better defined circle. The internal structure is inevitable in this approach, unless we place a threshold on the size of the radius used (forcing points that are close together to use a larger disk than they otherwise would).

This idea produces a type of variable α alpha hull, where the circle radius is defined by the relative distances of the points. It is also nice from the standpoint that it is a natural extension of the Gabriel graph.

Problems

3.29 Investigate the idea of thresholding the radius of the disk: replace $\alpha = \beta r_{vw}$ with $\alpha = \max(\beta r_{vw}, T)$, for a user provided threshold T. Using data similar to the "circled H" data, can you select β and T to get a good representation of the circle and the H?

3.30 Prove that for $\beta \leq 1$ the graph described in this section is a subgraph of the Delaunay triangularization.

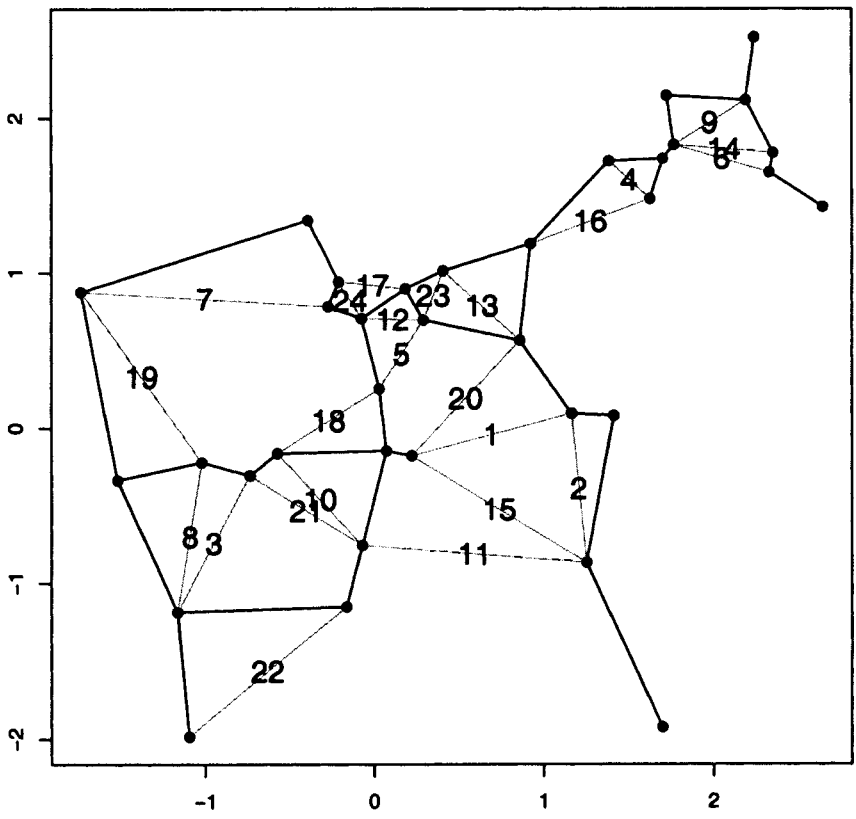


Fig. 3.40 The transition from the relative neighborhood graph ($\beta = 2$) to the Gabriel graph ($\beta = 1$). As β decreases, edges are added to the graph, in the order indicated by the numbers.

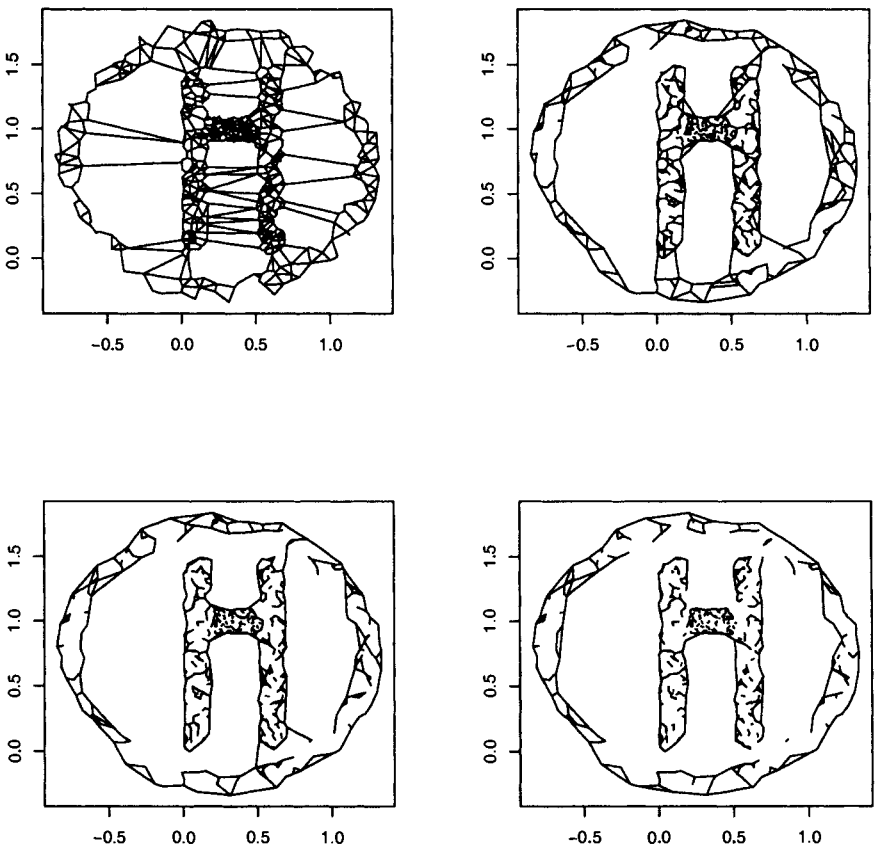


Fig. 3.41 Examples of the Gabriel graph version of the alpha hull algorithm for $\beta = 1, 1.5, 1.75, 2$ (top left, right, bottom left, right, respectively). The observations are not plotted to better show the graphs.

3.5.2 Application: Nearest-Neighbor Prototypes

One of the most popular classes of classifiers is the k-nearest-neighbor classifier. This has several nice properties:

- It is consistent (under appropriate assumptions); that is, its error goes to Bayes optimal.
- It is very easy to implement and use.
- It is easy to understand and explain.
- In our experience, it does quite well most of the time, and is a good classifier to try before attempting more complicated methods.
- It is very often used in comparison to other techniques, and so it has become a standard of comparison in pattern recognition.

Nearest-neighbor classifiers have one big drawback, however, particularly if one is dealing with very large data sets. All the (training) data must be retained, and upon presentation of a new observation for classification, the observation must be compared with all the retained training data. (This last is not quite true, as we will see in Chapter 6; however, even with the techniques discussed there, the computational burden of nearest-neighbor classifiers can be quite high.)

Consider the points in Figure 3.42. The two classes are indicated by the different symbols. Note that for a 1-nearest-neighbor classifiers, most, if not all, of the points in the upper left quadrant are not important. If they are removed, any point that would have been classified correctly due to proximity to one of these points would still be classified correctly. This is the idea behind nearest-neighbor editing. This is defined in Algorithm 3.2.

input : A set of points $X = \{x_1, \ldots, x_n\}$
output : A subset of the points with the same decision region

initialize: Construct the Voronoi regions for the points
for $i = 1, \ldots, n$ **do**
  if *any Voronoi neighbor of* x_i *has a different class label than* x_i **then**
    mark x_i;
  end
end
return *the marked points*

Algorithm 3.2: Nearest-neighbor editing using the Voronoi Tessellation

An alternative way of stating Algorithm 3.2 is that we retain only those points which are connected by Delaunay edges to points of different classes. This has much the same flavor as the complexity measure discussed in Section 2.4.3. See Figure

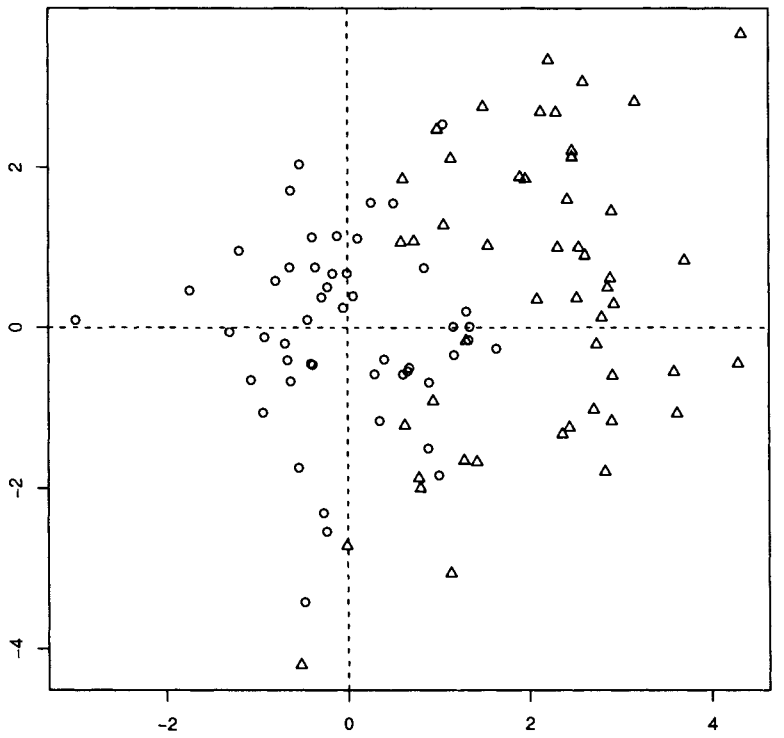


Fig. 3.42 An example two class data set. The classes are indicated by plotting symbols. The coordinate axes are drawn in dotted lines.

3.43 for the results of the algorithm on the example of Figure 3.42. This kind of data reduction is also referred to as **condensing**.

As can be seen by Figure 3.43, this approach tends to keep only those points that are close to the decision boundary. Another perspective is to remove points that are "confusing", that is points that will tend to be errors. This is the approach taken by training set editing, as done, for example by Cocosco. See

www.bic.mni.mcgill.ca/users/crisco/pgedit/

The idea here is to remove incorrectly labeled points from the data set. The algorithm is given in Algorithm 3.3.

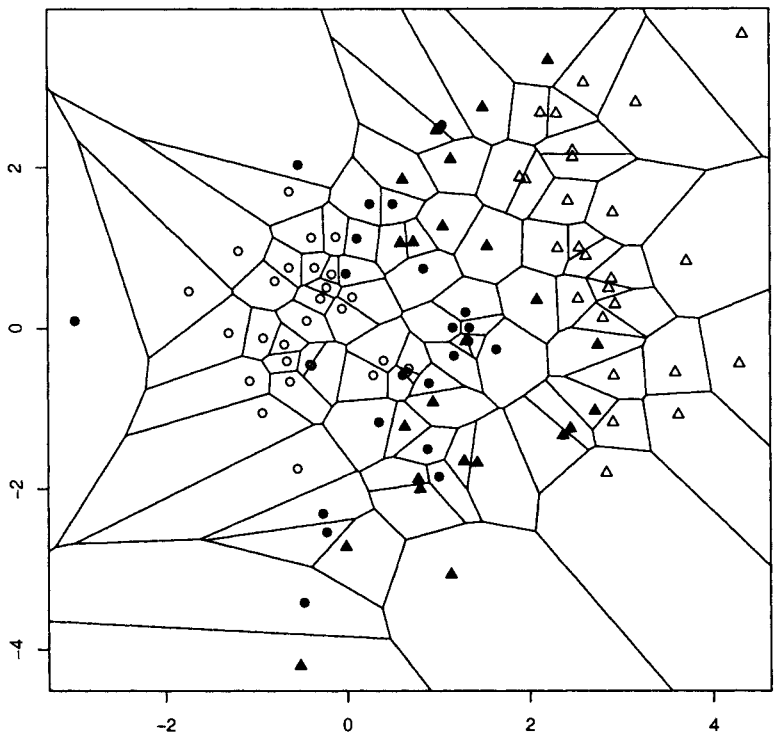


Fig. 3.43 The Delaunay triangularization of the points from Figure 3.42, with the points retained by Algorithm 3.2 depicted by filled symbols.

The results of running Algorithm 3.3 are depicted in Figure 3.44. As can be seen, the points removed tend to be those along the decision boundary between the classes, although the precise definition of this boundary depends on the proximity graph used.

References [180] and [179] discuss Algorithm 3.3 and variations. They combine condensing and editing, so that they simultaneously remove points that are not useful for the classification, and "clean" the data of (potentially) misclassified, or confusing, points.

Problems

3.31 Repeat the experiment shown in Figures 3.42 and 3.43 with two dimensional data of your own choosing. Discuss the results.

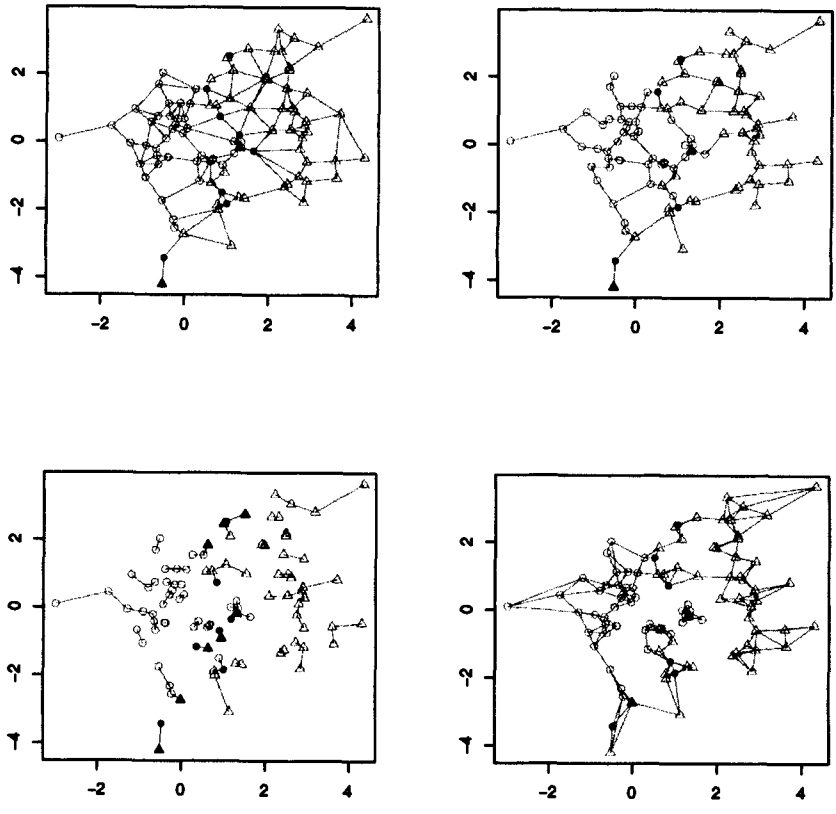


Fig. 3.44 The result of running Algorithm 3.3 on the points in Figure 3.42. The marked points are indicated by solid symbols. The proximity graphs are, from top left to bottom right, the Gabriel graph, the relative neighborhood graph, 1-nearest-neighbor graph and 3-nearest-neighbor graph. The proximity graphs are shown in light gray.

input : A set of points $X = \{x_1, \ldots, x_n\}$
output : A subset of the points with "incorrectly classified" points removed
initialize: Construct the proximity graph for the points
for $i = 1, \ldots, n$ **do**
  Determine the most common class c of the graph neighbors of x_i;
  if *c differs from the class of x_i* **then**
    mark x_i;
  end
end
return *the unmarked points*

Algorithm 3.3: Proximity Graph editing

3.6 SPHERE-OF-INFLUENCE GRAPHS

Sphere-of-influence graphs were introduced into the pattern recognition literature by [201]. These are neighborhood graphs where the neighborhood is a sphere, with radius defined by the nearest-neighbor distance. More precisely, we have the following:

Definition 3.15. *For each x_i let*

$$r_i = \min_{j \neq i} \{d(x_i, x_j)\}, \tag{3.29}$$

and

$$B_i = \{x : d(x, x_i) \leq r_i\}. \tag{3.30}$$

Then the graph defined on the vertices $V = X$ with edges $x_i x_j \iff B_i \cup B_j \neq \emptyset$ is called the **closed sphere-of-influence graph**, *denoted CSIG If the $\leq$ is replaced with $<$ in the definition, we refer to the graph as a* **sphere-of-influence graph** *or SIG.*

Reference [82] points out that in the plane, an equivalent definition is that the circles bounding the disks have two points in common (for SIG) or at least one point in common (CSIG). Also, that $d(v_i, v_j) < (\leq)\ r_i + r_j$. They prove a number of SIG and CSIG results, some of which are contained in the following theorem, in which the spheres in question are disks in the plane (with Euclidean distance).

Theorem 3.6 (Harary et al). *For points in the plane:*

1. *The union of nontrivial SIGs (CSIGs) is a SIG (CSIG).*
2. *The complete graph K_n for $n \leq 8$ is a SIG (CSIG).*
3. *Paths and cycles are SIGS. Nontrivial paths are CSIGs only if they have an even number of vertices. Only the cycle C_3 and all even cycles are CSIGs.*

Edge density and clique number for CSIGs is discussed in [138] and [137]. In particular, they show that

$$|E| \leq \left(5^d - \frac{3}{2}\right)|V|$$

for any CSIG in any $\mathbb{R}^d$ with the L_2 (Minkowski) metric. [139] extends these results, and gives some characterizations of CSIGS in various metric spaces. Reference [80] gives combinatorial bounds on the number of edges in a SIG, and [52], [66], [191] and [33] give results on the expected size of a random SIG.

Reference [94] proves the following theorem on the variance of the size of the associated random sphere of influence graph.

Theorem 3.7 (Hitczenko et al.). *Let $x_1, \ldots, x_n \overset{iid}{\sim} U[0,1]^d$ and let e_n denote the number of edges in their sphere-of-influence graph. Then for any $n \geq 4$ there exist positive constants c_d and C_d such that*

$$c_d n \leq var(e_n) \leq C_d n.$$

An extension of the idea of the sphere-of-influence graph is to require that the overlap of the spheres be larger than a fixed amount. This results in the so called *tolerance sphere-of-influence graph* (see [119] and [96]).

Note that the distinction between SIGs and CSIGS is (generally) irrelevant in pattern recognition. Often one assumes the data are continuous, in which case the graphs defined by open or closed balls are the same almost surely. In some cases, for example with categorical data or mixed continuous and categorical variates, the distinction may be important. For the data we have discussed in the book it is not.

Problems

3.32 Prove Theorem 3.6.

3.33 Given a set of points $\{x_1, \ldots, x_n\} \in \mathbb{R}^2$, prove that the SIG on the points is a subgraph of the CSIG. Provide a "proof without words" to illustrate this. Give an example where the graphs are not the same.

3.7 SPHERE-OF-ATTRACTION GRAPHS

A relative of the sphere-of-influence graph discussed in Section 3.6 is the sphere-of-attraction graph described in [135].

Definition 3.16. *Let $C = (c_1, \ldots, c_n)$ and $P = (p_1, \ldots, p_m)$ be given subsets of R^d. In the original paper ([135]) the C represent consumer's ideal product preferences and the P represent actual products. Analogous to Definition 3.15, define*

$$r_i = \min_{1 \leq j \leq m} \{d(c_i, p_j)\}, \tag{3.31}$$

and

$$B_i = \{x : d(x, c_i) \leq r_i\}. \tag{3.32}$$

The **closed sphere-of-attraction graph** *(CSAG) is the graph on vertices $V = C$ with edges $c_i c_j \iff B_i \cup B_j \neq \emptyset$. If the $\leq$ is replace with $<$ in the definition, the graph is referred to as a sphere-of-attraction graph, or SAG.*

One way to think about the CSAG is that each ball centered at a customer is defined to be just large enough to cover the nearest product. It thus corresponds to all potential products that are as close to the customer's ideal as the nearest existing product. Any novel product that falls in the ball will be "better" than any existing product (for that customer).

Reference [135] characterizes CSAGs in several ways. First they note that any planar graph is a CSAG. This uses a result of Koebe ([113], see [150]).

Definition 3.17. *A* **coin graph** *is defined as the intersection graph of disks in the plane such that no two disks intersect except (possibly) on their boundary.*

Koebe proved that a graph is planar if and only if it is a coin graph. Thus we have the following:

Theorem 3.8 (Mcmorris and Wang). *Every planar graph is a CSAG.*

Further characterizations are discussed in [135].

3.8 OTHER RELATIVES

In this section we will look at a couple of graphs related to the neighborhood graphs already discussed. k-nearest-neighbor graphs are generalized in [197].

Definition 3.18. *Given points $\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^q$, a* **$k$-ply neighborhood system** *is a set of closed balls $\Gamma = \{B_1, \ldots, B_n\}$, with B_i centered on x_i, such that no point in $\mathcal{X}$ is contained in the interior of more than k balls.*

Given a ball B of radius r centered at x, define for $\alpha > 0$, αB to be the ball centered at x of radius αr.

Definition 3.19. *The α-overlap graph of a k-ply neighborhood system Γ is the graph with vertices $V = \{x_1, \ldots, x_n\}$ and edges $x_i x_j \in E$ if and only if $B_i \cap (\alpha B_j) \neq \emptyset$ and $B_j \cap (\alpha B_i) \neq \emptyset$.*

When $\alpha = 1$ the α-overlap graph is the intersection graph of the neighborhood system.

Reference [205] describes a variation on the region of influence graphs which extends the RNG and GG as follows. Let $\sigma > 0$ be given, and set $D = \min_{x \in V}[d(x, p), d(x, q)]$. Then define

$$R_1(p, q, \sigma) = R_{GG}(p, q) \cup \{x : \sigma D < d(p, q)\} \tag{3.33}$$

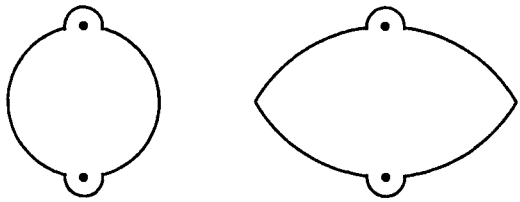


Fig. 3.45 Neighborhood regions $R_1(p, q, 0.25)$ and $R_2(p, q, 0.25)$.

$$R_2(p, q, \sigma) = R_{RNG}(p, q) \cup \{x : \sigma D < d(p, q)\}, \tag{3.34}$$

where R_{GG} and R_{RNG} are the regions of influence for the Gabriel and relative neighborhood graphs, respectively. [169] modify the graph defined by $R_1(p, q, \sigma)$ to incorporate weights on the edges. These weights are used to define "fuzzy" clusters.

The basic idea is to create a hierarchy of clusterings by varying σ in Equations (3.33) or (3.34) and considering the connected components of the resultant graphs.

As with nearly all clustering algorithms, in the final analysis a subjective assessment must be made as to the number of clusters in the data. This requires some kind of assessment of the discovered clusters. In two or three dimensions, this is fairly simple, since we can plot the data and associated clusters, and determine by eye whether the discovered clusters agree with our intuition (please note the warning discussed on p. 26). In higher dimensions, this becomes problematic. Several approaches are possible, such as projecting to lower dimensions via some technique such as principal components or multidimensional scaling; defining an objective measure of cluster fit, such as the ratio of within-class covariance to between class covariance; prediction strength as in [198]; or using multiple clustering algorithms and comparing the results to obtain a consensus. This latter is one reason for considering multiple variations on the various themes, as we have been doing in this book.

Problems

3.34 Investigate the methods of this section on the data in Table 2.1.

3.35 Investigate through simulation the properties of these techniques on data drawn from the distribution of the plots in Figure 2.13. The arms and bars of the "H" are made up of uniform data in rectangles, the circle consists of data uniform on an annulus. The "S" distribution is defined on page 48.

3.9 ASYMPTOTICS

Many of the invariants of the graphs we have studied satisfy a central limit theorem, and this can be useful in devising tests in pattern recognition. The definitive reference for this section is [153]. We reproduce their main result below, without proof. The interested reader is directed to the paper for more details.

First, some notation. For a set $S \subset \mathbb{R}^d$, let $|S|$ denote the Lebesgue measure of S. For $x \in \mathbb{R}^d$, the translate $S \pm x = \{y \pm x : y \in S\}$. $Q_r(x) = [-r, r]^d + x$, the translate of the cube. Similarly, for $x > 0$, let $aS = \{ax : x \in S\}$, the **scaling** of S. Denote the r-neighborhood of the boundary of S by $\partial_r S = \cup_{x \in \partial S} Q_r(x)$. The **diameter** of S is denoted $\text{diam}(S) = \sup\{|x - y| : x, y \in S\}$, where in this $|\cdot|$ denotes Euclidean distance.

Fix $\lambda > 0$. A sequence bounded of Borel subsets of $\mathbb{R}^d$, $(B_n)_{n \geq 1}$, will be called a **sequence of windows** of $\mathbb{R}^d$ if they satisfy:

1. $|B_n| = \frac{n}{\lambda}$.
2. $\cup_{n \geq 1} \cap_{m \geq n} B_m = \mathbb{R}^d$, so B_n "tends to" $\mathbb{R}^d$ as $n \to \infty$.
3. $\lim_{n \to \infty}(|\partial_r B_n|/n) = 0$ for all $r > 0$. This is called the *vanishing relative boundary* condition.
4. For some β_1, $\text{diam}(B_n) \leq \beta_1 n^{\beta_1}$. This is called the *polynomial boundedness* condition on B_n.

Fix B_0 a bounded Borel set in $\mathbb{R}$ with Lebesgue measure 1 and a boundary of Lebesgue measure 0. A typical example of this would be the unit cube. Note that a sequence of cubes, properly scaled, is an obvious choice for a sequence of windows on $\mathbb{R}$ and this is the sequence one should be thinking of as we proceed: the sequence of hypercubes centered at the origin with side of length $\sqrt[d]{n/\lambda}$.

For such a sequence of windows, the translation of the sequence is denoted $\mathcal{B} = (A)$ such that $A = B_n + x$ for some n and x, that is, the collection of all translates of elements of the sequence. Similarly, $\mathcal{B}_0$ is the collection of all translates of scalings of B_0.

We are interested in considering graphs resulting from observations drawn from uniform random variables on the sequence of windows, so let $\mathcal{U}_{m,n} = \{U_{1,n}, \ldots, U_{m,n}\}$ be i.i.d. uniform random variables on B_n, and $\mathcal{P}_{\lambda,n}$ be a homogeneous Poisson process on B_n of intensity λ. Finally, $\mathcal{X} = \{X_1, \ldots, X_n\}$ are i.i.d. uniform random variables on B_0.

The quantity of interest is some invariant of a graph defined by the random observations, such as the number of components in the sphere-of-influence graph. Let H be a real valued functional defined for all finite subsets of $\mathbb{R}$, corresponding to the invariant of interest. Since H can really be thought of as operating on the graph, we require that H be translation invariant: $H(A + x) = H(A)$. One can equivalently think of H as operating on a subset of $\mathbb{R}$ or on the corresponding graph defined by the subset. Note that all the graphs in this chapter are translation invariant, in the sense that the underlying observations can be translated without changing the graph.

Several further technical constraints must be placed on H. These basically come down to requiring that H not change too much with the addition of a single point. Let $\mathcal{P}_\lambda$ be a Poisson process of intensity λ on $\mathbb{R}$. Define the **add one cost** of H as the change in H caused by adding a vertex at the origin; $\Delta(\mathcal{X}) = H(\mathcal{X} \cup \{0\}) - H(\mathcal{X})$.

Definition 3.20. *H is* **strongly stabilizing** *if there exists almost surely finite random variables S and Δ_∞ such that with probability 1, $\Delta((\mathcal{P} \cup B_S(0)) \cup A) = \Delta_\infty$ for all finite $A \subset \mathbb{R}^d \setminus B_S$, where $B_S(0)$ is the ball of radius S at the origin. S is called the* **radius of stabilization** *of H.*

In words, H is strongly stabilizing if the add one cost does not depend on points outside a finite region of the origin. We also will require a constraint on the moments of the add one cost. For a set $A \in \mathcal{B}$, let $\mathcal{U}_{m,A}$ be a point process of m i.i.d. uniform variates on A.

Definition 3.21. *H satisfies the* **uniform bounded moments** *condition on $\mathcal{B}$ if*

$$\sup_{0 \in A \in \mathcal{B}} \sup_{m \in I_\lambda(A)} E[\Delta(\mathcal{U}_{m,A})^4] < \infty,$$

where $I_\lambda(A) = [\frac{\lambda|A|}{2}, \frac{3\lambda|A|}{2}]$, and $|\cdot|$ denotes the cardinality of the set.

Finally, we require a uniform bound on H.

Definition 3.22. *H satisfies the* **uniform bounded** *condition if there exists a constant β_2 such that for all finite sets $\mathcal{X} \subset \mathbb{R}^d$,*

$$|H(\mathcal{X}|) \leq \beta_2 (diam(\mathcal{X}) + |\mathcal{X}|)^{\beta_2}.$$

Theorem 3.9 (Penrose and Yukich). *Let H satisfy definitions 3.20-3.22. Assume the distribution of Δ_∞ is non-degenerate. Then given λ, there exist constants σ, τ independent of the choice of (B_n), with $0 < \tau^2 \leq \sigma^2$ such that as $n \to \infty$,*

$$n^{-1} Var(H(\mathcal{P}_n)) \to \sigma^2 \tag{3.35}$$

and

$$n^{-1}(H(\mathcal{P}_n) - E(H(\mathcal{P}_n)) \xrightarrow{\mathcal{D}} N(0, \sigma^2). \tag{3.36}$$

Similarly,

$$n^{-1} Var(H(\mathcal{U}_{n,n})) \to \tau^2 \tag{3.37}$$

and

$$n^{-1}(H(\mathcal{U}_{n,n}) - E(H(\mathcal{U}_{n,n})) \xrightarrow{\mathcal{D}} N(0, \tau^2). \tag{3.38}$$

See [153] for a proof of the theorem.

Theorem 3.9 provides for central limit theorems for several of the graphs and digraphs we have investigated. For example, [153] provide the following (see the paper for precise statements of the theorems).

Theorem 3.10 (Penrose and Yukich). *Under appropriate conditions,*

1. *The total edge length of the k-nearest-neighbor graph*
2. *The number of edges in a sphere-of-influence graph*
3. *The total edge length in the Voronoi tessellation (considering only the edges of finite length)*

each satisfy a central limit theorem. Similar results are available for the Delaunay triangularization, Gabriel graph and relative neighborhood graph.

3.10 FURTHER READING

Reference [131] discusses intersection graphs, including sphere-of-influence graphs, from a graph theoretic perspective, and provide a number of references to results. Reference [69] utilize intersection graphs in the analysis of survival data. They show how using the set of maximum cliques of an intersection matrix defined by the censored data can be used to produce a maximum likelihood estimation for the distribution function of the data. The censored data considered is time-to-event data, for example, time to failure or time to death, where in some cases failure has not yet occurred at the time of the data collect.

Reference [20] discusses breast lesion analysis. Utilizing various image processing algorithms, ducts are extracted and highlighted. The cell nuclei are used to construct a neighborhood graph. This is used to extract statistics about the duct, and the dual graph results in statistics about the area and perimeter of the duct. These are used to characterize benign and malignant lesions.

In [46], Chapter 19, and [85], Chapter 13, various types of nearest-neighbor prototyping are discussed. These are good references for both practical and theoretical considerations.

References [188] and [2] use neighborhood graphs to describe the topological properties of cortical structures. In this, graphs (actually, multigraphs) are defined in terms of the adjacency of voxels (three-dimensional versions of pixels) for foreground and background, and it is shown that if these are trees then the cortical region is homeomorphic to a sphere. This is important for correcting anomalies introduced in the magnetic resonance images by noise and other mechanisms.

Path planning is an important part of many systems. It is essential for many robotic applications, such as planning the motion of a robotic arm, to navigating through a field of obstructions. Reference [220] discusses a method using what they call **random neighborhood graphs**. This is a sphere intersection graph. The spheres are used to define a potential field, with the graph providing a link from high-potential spheres to low-potential spheres. This moves the robot from high-cost areas to low-cost areas, providing a path of "least resistance" through the obstacles.

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

4

Class Cover Catch Digraphs

There is nothing that man fears more than the touch of the unknown. He wants to see what is reaching towards him, and to be able to recognize or at least classify it. Man always tends to avoid physical contact with anything strange.

Elias Canetti, "The Fear of Being Touched," *Crowds and Power.*

4.1 CATCH DIGRAPHS

We have seen in Chapter 3 graphs defined by the intersection of sets. We discussed this in terms of neighborhoods, since that was appropriate to the applications we wished to study. This idea, though, is completely general. Given a collection $\mathcal{S}$ of sets, we can construct the intersection graph exactly analogously to the neighborhood graph.

Definition 4.1. *Given a collection of sets $\mathcal{S}$ we define the* **intersection graph** *to be the graph with vertices the sets of $\mathcal{S}$ and edges $\{vw \mid v \cap w \neq \emptyset\}$.*

It has been shown ([215]) that all (finite) graphs can be represented as intersection graphs for some family of convex sets in $\mathbb{R}^3$. We are interested in a slight modification to this concept, which is more in keeping with the neighborhood graphs of Chapter 3.

A **pointed set** is a set with a distinguished element, called the **base point**. This leads to a slightly different idea from that of intersection graphs. Instead of considering general intersections of the sets, we only consider those which contain (catch) the base point. This leads to the definition of the catch digraph.

Definition 4.2. *Let $\{(S_i, p_i)\}$ be a family of pointed sets, S_i denoting the set and $p_i \in S_i$ denoting the points. The* **catch digraph** *for the family is the digraph with vertices S_i (or, equivalently, p_i) and edges $ij \iff p_j \in S_i$. The set S_i is said to "catch" the point p_j.*

We will view the vertices as represented by the points rather than the sets. The idea is that there is an edge from p_i to p_j if the set associated with p_i "catches" (contains) p_j. This is (generally) truly a digraph (i.e., edges are not necessarily bi-directed), since there are no a priori constraints on the sets S_i.

The references [167] and [168] provide an extensive discussion of catch digraphs where the sets are intervals. A characterization of interval catch digraphs is given, and algorithms for recognizing whether a digraph is an interval catch digraph, and constructing the pointed sets, is given.

4.1.1 Sphere Digraphs

A particular kind of catch digraph, similar to the neighborhood graphs of Chapter 3 was introduced by [122]:

Definition 4.3. *A catch digraph where the sets are spheres (in $\mathbb{R}^n$) and the points are the sphere centers is called a* **sphere digraph.**

The spheres in the definition are called **balanced spheres**.

Theorem 4.1 (Maehara). *A digraph is a sphere digraph if and only if D^0 (see p. 6 for the definition) is acyclic.*

Proof. Suppose D is a sphere digraph. Let $v_1, v_2, \ldots, v_k = v_1$ be a directed cycle in D^0, and denote by r_i the radius of the sphere S_i associated with vertex v_i. Note that $r_i > r_{i+1}$, since otherwise the edge would be bidirectional. Thus $r_1 > r_2 > \ldots > r_{k-1} > r > r_k = r_1$, which is a contradiction.

Now assume D^0 is acyclic. Then there is an ordering of the vertices such that the adjacency matrix is upper triangular (see Problem 4.1). Let $v_1, \ldots, v_n$ be such an ordering, where n is the order of the digraph. We have $v_i v_j \in A(D^0) \to i < j$. We construct the balanced spheres as follows. Let $x \gg n$ be sufficiently large, and let B be the matrix obtained by replacing the 1's in the ith row of the adjacency matrix of D^0 with i's. Let S_i be the sphere with center equal to the ith row of $B + xI_n$, and radius

$$r_i = \sqrt{(2x^2 - (2i-1)x)}.$$

Then the sphere digraph for these spheres is isomorphic to the digraph D (see Problem 4.2). □

Sphere digraphs are a kind of proximity graph. That is, they are defined in terms of the "proximity" of points to one another. The spheres define regions of "proximity", and points inside these regions are connected via edges.

In the next section, we will be interested in a particular type of sphere digraph. The centers and radii will be defined by observations, as was the case with neighborhood

graphs in Chapter 3. However, we will be considering a classification problem, and the observations corresponding to the centers will be from a different class than those that define the radii. This will allow us to construct sphere digraphs whose underlying spheres cover the observations from one of the classes, at the exclusion of the observations from the other class. From this, we will build a classifier, and discuss some methods for investigating the relationships between the two classes.

Problems

4.1 Prove that if a digraph with no bidirectional edges is acyclic, then there is an ordering of the vertices such that its adjacency matrix is upper triangular.

4.2 Complete the proof of Theorem 4.1 as follows. Denote by $y \sim z$ the statement that $|y - z| \ll x$, and by d_{ij} the Euclidean distance between the centers of S_i and S_j. Show that

1. If $v_iv_j \in A(D)$ and $v_jv_i \in A(D)$, then $d_{ij}^2 \sim 2x^2 - 2(i+j)x$ and thus $d_{ij} < r_i$ and $d_{ij} < r_j$.
2. Similarly, if $v_iv_j \in A(D)$ and $v_jv_i \notin A(D)$, then $d_{ij} < r_i$ and $d_{ij} > r_j$.
3. Finally, if $v_iv_j \notin A(D)$ and $v_jv_i \notin A(D)$, then $d_{ij} > r_i$ and $d_{ij} > r_j$.

Conclude that the sphere digraph on the S_i is isomorphic to D.

4.2 CLASS COVERS

We now turn to the problem of classification. We will consider the two-class problem throughout, although it will be clear that the generalizations to multi-class problems are straightforward. Thus, as in Section 1.2, we have (X, Y) with Y a $\{0,1\}$-valued random variable denoting the class label. We denote by X_0 the training data associated with the random variable $X|Y=0$, and similarly for X_1. In a slight abuse of notation, we will write $X_0 = \{x_1, \ldots, x_n\}$ and $X_1 = \{y_1, \ldots, y_m\}$. The possibility of confusion with Y is mitigated by the avoidance of extra subscripts. We will refer to the X_0 observations as "target" points, while the X_1 points are "non-target" points.

4.2.1 Basic Definitions

Definition 4.4. *Let $d(\cdot,\cdot) : \Re \times \Re \to [0,\infty)$ be any distance or dissimilarity measure. The* **class cover problem** *for a distinguished target class X_0 is to find a minimal collection of open balls $B_i := B(c_i, r_i) := \{x : d(x, c_i) < r_i\}$, such that*

A. $X_0 \subset \cup_i B_i$.

B. $X_1 \cap (\cup_i B_i) = \emptyset$.

A collection of balls satisfying these conditions is termed a **class cover**. Condition A defines a **proper** cover of class X_0 while B defines a **pure** cover with respect to

class X_1. Thus a proper cover is one which covers all the X_0 points, while a pure one covers none of the X_1 points. We will see some cases below where we will wish to relax these constraints slightly.

A **constrained** class cover requires the balls to be centered at target class observations; $c_i \in X_0 \; \forall i$. An **unconstrained** class cover is one in which the ball centers are not required to be centered on observations. A **homogeneous** class cover requires the ball radii to be the same for all balls; $r_i = r \; \forall i$. If the ball radii are unequal, the class cover is **inhomogeneous**.

This class cover problem is a generalization of the set cover problem (see for example [68]). It is motivated by applications in machine learning and statistical pattern recognition wherein the methodology requires the selection of a (small) set of representative exemplars from X_0. In fact, many statistical pattern recognition techniques, such as the reduced nearest-neighbor classifier and support vector machines (see for example [38] and [208]) can be viewed as methodologies for the selections of representative exemplars. The class cover problem has been previously considered in [31] and [162].

Definition 4.5. *The random* **class cover catch digraph** D *for* $X_0 = \{x_1, \ldots, x_n\}$ *against* $X_1 = \{y_1, \ldots, y_m\}$, *is the digraph of order* n *with vertex set* X_0 *and a directed edge from* x_i *to* x_j *if and only if* $x_j \in B(x_i, \min_{y \in X_1} d(x_i, y))$.

That is, there is an edge from the ith vertex to the jth vertex if and only if there exists an open ball centered at X_i which is "pure", or contains no elements of class 1, and simultaneously contains, or "catches", point x_j. We write $V(D) = \{v_i, \cdots, v_n\}$ with each v_i corresponding to x_i; thus $v_i v_j \in A(D)$ if and only if $x_j \in B_i$. We call such a digraph a random $\mathcal{C}_{n,m}$-graph.

Our $\mathcal{C}_{n,m}$-graphs can be seen to be a special case of both the intersection digraphs of [186] and the covering sets or transversals of [204]. Clearly, class cover catch digraphs are a kind of sphere digraph (Section 4.1.1). In addition, significant similarities are apparent between $\mathcal{C}_{n,m}$-graphs and neighborhood graphs (Chapter 3), in particular, sphere-of-influence (Section 3.6) and sphere-of-attraction graphs (Section 3.7).

This is a vertex random graph model ([110], see also p. 8 in Section 1.1.3) and is not one of the standard models. Rather, the randomness in our $\mathcal{C}_{n,m}$ model resides in the vertices X_0 and the existence of an edge $v_i v_j$ is a deterministic function of the random variable X_1 and the random set B_i. Thus our random graph model is *data-driven* — a function of the joint distribution F_{X_0,X_1}.

Problems

4.3 Generate $n = 20$ bivariate "target" observations uniformly on a disk centered at the origin. Designating the origin to be "non-target", compute the class cover catch digraph, and plot it using the generated points as the vertex positions.

4.4 Generate data as in Problem 4.3 and compute the number of (weakly) connected components for the CCCD. Repeat this for different values of n, for at least 100 Monte

Carlo replications, and investigate (empirically) the probability that these CCCD's are (weakly) connected as a function of n.

4.3 DOMINATING SETS

A simple class cover is defined by the collection of all the balls defining the class cover catch digraph. This class cover provides us with a classifier, defined in terms of the interior and exterior of the class cover, or equivalently, the balls that define it. In general, this is a highly redundant cover, however, and so for reasons of computational complexity as well as issues related to the curse of dimensionality (see Section 1.2.2), we would like to reduce the number of balls in the cover. So we would like a solution to the class cover problem.

The solution is to use a minimum dominating set of the class cover catch digraph. This covers all the observations (as a consequence of domination) and is of minimum cardinality. Since the selection of a minimum dominating set is NP-complete ([88]), even for the subclass of digraphs that we are considering ([43]), we need an efficient method to obtain an approximately minimum dominating set. The greedy algorithm for this is given in Algorithm 4.1.

input : A graph or digraph $G = (V, E)$
output : A dominating set S

initialize: $\mathcal{J} = V, \mathcal{S} = \emptyset$
while $\mathcal{J} \neq \emptyset$ **do**
  $v = \underset{v' \in V}{\operatorname{argmax}}\, d_{out}(v')$;
  $\mathcal{S} = \mathcal{S} \cup \{v\}$;
  remove from E all edges incident on v;
  $\mathcal{J} = \mathcal{J} \setminus \bar{N}(\mathcal{S})$;
end

Algorithm 4.1: Greedy dominating set algorithm

An equivalent way to state the greedy algorithm is to repeatedly select the vertex which covers (is adjacent to) the largest number of uncovered vertices. Note that the vertex so chosen may be one already covered by an already-selected vertex. Eliminating the covered vertices from consideration tends to produce larger dominating sets than otherwise.

The greedy algorithm does not guarantee a minimum dominating set. It does produce a dominating set, and in many practical applications seems to do a good job of producing nearly minimum sets. In high-dimensional problems, we have observed (anecdotally) that it often seems to produce minimum dominating sets; see, for example, the discussion of the gene expression data (Section 4.6.1).

input : A class cover catch digraph $G = (V, A)$ and associated radii $\mathcal{R} = \{r_1, \ldots, r_n\}$
output : A dominating set $\mathcal{S}$

initialize: $\mathcal{J} = V, \mathcal{S} = \emptyset, \mathcal{R}' = \mathcal{R}$
while $\mathcal{J} \neq \emptyset$ **do**
  $i = \text{argmax}_j\{r_j : r_j \in \mathcal{R}'\}$;
  $\mathcal{S} = \mathcal{S} \cup \{v_i\}$;
  $\mathcal{J} = \mathcal{J} \setminus \bar{N}(\mathcal{S})$;
  remove from $\mathcal{R}'$ all radii associated with elements of $\mathcal{S}$;
end

Algorithm 4.2: Greedy by-radius dominating set algorithm

input : A digraph $D = (V, A)$

initialize: Use Algorithm 4.1 to find a dominating set $\hat{S}$ with $\hat{\gamma} = |\hat{S}|$. Set $\gamma^\star = \hat{\gamma} - 1$. Let $\mathcal{Z}$ be the set of vertices with in-degree 0.
foreach $S \subset V \setminus \mathcal{Z}$ *of size* $\gamma^\star - |\mathcal{Z}|$ **do**
  if $S \cup \mathcal{Z}$ *is a dominating set* **then**
    $\hat{S} = S \cup \mathcal{Z}$;
    $\hat{\gamma} = \gamma^\star$;
    $\gamma^\star = \hat{\gamma} - 1$;
    restart the algorithm with this initialization;
  end
end

Algorithm 4.3: Minimum dominating set algorithm

An alternative algorithm, which utilizes the radii of the balls, is given in Algorithm 4.2. This algorithm seems to generally produce larger dominating sets than the standard greedy algorithm. However, if the application is such that larger balls are preferred, this algorithm may be a better choice.

An alternative to Algorithm 4.2 is to use the radii as tie breakers in Algorithm 4.1. Thus, when there are multiple vertices covering the same number of points, one selects the one with the largest (maximum volume) or the smallest (maximum density) radius.

In general, to find a dominating set of size s in a digraph of order n, one may have to check all $\binom{n}{s}$ subsets of size s. Thus, the problem of finding a true minimum dominating set can be computationaly intractable. However, sometimes we reduce the necessary computations by taking advantage of some simple observations. Algorithm 4.3 uses the observation that any dominating set must contain all observations with in-degree zero. For graphs, these vertices are the isolated vertices, and one would naturally separate them before running the greedy algorithm.

Algorithm 4.3 starts with a greedy approximation to the minimum dominating set, then checks for sets of size one fewer vertices. If it finds one which dominates, it then resets and starts over. Note that since all vertices of in-degree zero must be in any dominating set, these are automatically added, reducing the number of sets that must be checked from $\binom{n}{\gamma^*}$ to $\binom{n-|\mathcal{Z}|}{\gamma^*-|\mathcal{Z}|}$, which can be a significant reduction in sparse digraphs. Further improvements can be made, for instance by noting that for each vertex v_i of in-degree one and $wv_i \in A$, either v_i or w must be in any dominating set. Clearly, any algorithm for computing a dominating set should be run on each connected component of the digraph independently. The problem of finding a minimum dominating set is NP-Hard, however for some digraphs, Algorithm 4.3 can find minimum dominating sets quite quickly. It is also easy to modify the algorithm to return all minimum dominating sets.

Problems

4.5 Redo Problem 4.4 computing instead the (approximate) domination number for the digraph using the two greedy algorithms (Algorithm 4.1 and 4.2). Investigate the distributions of the dominating set sizes produced by the two algorithms.

4.6 Using the data generated in problem 4.5, determine the exact domination number (Hint: It will be less than 6 in all cases). Compare this with the results of the greedy algorithms.

4.7 Modify Algorithm 4.3 to return all minimum dominating sets.

4.4 DISTRIBUTIONAL RESULTS FOR $\mathcal{C}_{N,M}$-GRAPHS

Before considering the problem of constructing classifiers from $\mathcal{C}_{n,m}$-graphs, it is important to understand some of their properties. In this section we will consider various calculations concerning the distributions of various aspects of these graphs. The calculations are considerably more tractable in the univariate case, and so we will consider this case first, even though from a pattern recognition standpoint, this is the less interesting case.

One of the most important graph invariants that we will be concerned with is the domination number γ. This is because, as has been discussed above, dominating sets allow us to reduce the computational complexity of the classifier constructed from the class cover catch digraph. Knowing the domination number tells us the number of exemplars that must be retained, and hence the complexity of the resulting classifier.

4.4.1 Univariate Case

Consider now class cover catch digraphs in one dimension – the case $\mathfrak{R} = \mathbb{R}$. As mentioned above, class cover catch digraphs on $\mathbb{R}$ are a special case of interval catch digraphs. See [167] and [168].

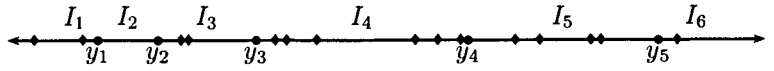


Fig. 4.1 The one-dimensional class cover catch digraph divides naturally into intervals between class X_1 (y_i) points. The balls associated with the x_i (diamonds) can't cover X_1 observations, and so no edge can cross a y.

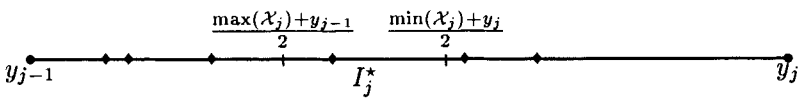


Fig. 4.2 The critical region, $I_j^\star$, for a particular interval.

Consider the collection of $m+1$ intervals based on the order statistics

$$-\infty =: y_{(0)} < y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(m)} < y_{(m+1)} := +\infty.$$

Let $I_j := (y_{(j-1)}, y_{(j)})$ for $j = 1, \cdots, m+1$. See Figure 4.1. Let $\mathcal{X}_j = I_j \cap X_0$ and $\mathcal{Y}_j = \{y_{(j-1)}, y_{(j)}\}$. This gives us $m+1$ disconnected subgraphs D_j, each of which may be null or may itself be disconnected. Note that it is only in the one-dimensional case that we can break the catch digraph into disconnected subgraphs in this simple way. Define $N_j := |\mathcal{X}_j|$, and let $\gamma_j(N_j)$ denote the cardinality of a minimum dominating set for the random $\mathcal{C}_{n,m}$–graph induced by $\mathcal{X}_j$, $\mathcal{Y}_j$. Then $\gamma(D; n, m) = \sum_{j=1}^{m+1} \gamma_j(N_j)$. Thus the study of $\gamma(D; n, m)$ is carried out via the investigation of the simpler random variables $\gamma_j(N_j)$.

Lemma 4.1. *For $j = 1$ or $m+1$, we have $\gamma_j(N_j) = 1\{N_j > 0\}$, where $1\{\cdot\}$ is the indicator function.*

The proof of the lemma is left as an exercise (Problem 4.10).

For $j = 2, \cdots, m$ and $N_j \geq 1$, $\gamma_j(N_j)$ takes values in $\{1, 2\}$ with distribution-dependent probabilities $\{\kappa_j(N_j), 1 - \kappa_j(N_j)\}$, respectively, where

$$\kappa_j(N_j) := P[\mathcal{X}_j \cap I_j^\star \neq \emptyset]$$

with

$$I_j^\star := \left(\frac{\max(\mathcal{X}_j) + y_{(j-1)}}{2}, \frac{\min(\mathcal{X}_j) + y_{(j)}}{2} \right) \subset I_j.$$

See Figure 4.2.

Lemma 4.2. *For $j = 2, \cdots, m$, $\gamma_j(N_j) =^d 1 + Bernoulli(1 - \kappa_j(N_j))$ for $N_j \geq 1$.*

Proof. Clearly $\gamma_j(0) = 0$. If $N_j = 1$ ($\mathcal{X}_j = \{X\}$) then there exists an $\epsilon > 0$ such that the ball $B(X, \epsilon)$ suffices to demonstrate that $\gamma_j(1) = 1$, and $X \in I_j^\star$ so $\kappa_j(1) = 1$ as desired. Suppose now that $N_j \geq 2$. Let $X_j^- := \max\{X \in \mathcal{X}_j : X \leq \frac{Y_{(j-1:m)}+Y_{(j:m)}}{2}\}$ and $X_j^+ := \min\{X \in \mathcal{X}_j : X \geq \frac{Y_{(j-1:m)}+Y_{(j:m)}}{2}\}$. ($X_j^- < X_j^+$ *a.s.* if both exist.) Let $B_j^- := B(X_j^-, X_j^- - Y_{(j-1:m)})$ and $B_j^+ := B(X_j^+, Y_{(j:m)} - X_j^+)$. ($B_j^-$(respectively B_j^+)$= \emptyset$ if $X_j^-(X_j^+)$ does not exist.) Since $\mathcal{X}_j \subset (B_j^- \cup B_j^+)$ and $\mathcal{Y}_j \cap (B_j^- \cup B_j^+) = \emptyset$, it follows that $\gamma_j(N_j) \in \{1, 2\}$. Finally, observe that $\gamma_j(N_j) = 1 \iff$ there exists $X \in \mathcal{X}_j$ such that (i) $X - \min(X_j) < Y_{(j:m)} - X$ and (ii) $\max(X_j) - X < X - Y_{(j-1:m)}$, and (i) and (ii) hold if and only if there exists $X \in I_j^\star$. □

Clearly $P[\mathcal{X}_j \cap I_j^\star \neq \emptyset]$ depends on the conditional distribution $F_{X|Y}$ on the interval I_j; if we know this distribution, we can calculate κ_j.

As an immediate consequence of the preceding two Lemmas the upper bound of n for $\gamma(D; n, m)$ can be tightened for $\mathcal{F}(\mathbb{R})$–random $\mathcal{C}_{n,m}$–graphs.

Theorem 4.2. *Let D be an $\mathcal{F}(\mathbb{R})$-random $\mathcal{C}_{n,m}$-graph. Then* $1 \leq \gamma(D; n, m) \leq \min(n, 2m)$.

The following Lemma provides the exact result for the distribution of the cardinality of a minimum dominating set for a simple random $\mathcal{C}_{n,2}$-graph.

Lemma 4.3. *Let $-\infty < a < b < +\infty$. Let $\mathcal{X} = \{X_1, \cdots, X_n\}$ with $X_i \sim^{i.i.d.}$ Uniform(a, b). Let $\mathcal{Y} = \{a, b\}$. Let D be the random $\mathcal{C}_{n,2}$-graph for $\mathcal{X}$, $\mathcal{Y}$. Then $\gamma(D; n, 2) =^d 1 +$ Bernoulli$(1 - \kappa(n))$ with*

$$\kappa(n) := P[\gamma(D; n, 2) = 1] = \frac{5}{9} + \frac{4}{9} \cdot 4^{-(n-1)}$$

for $n \geq 1$.

Note that some consequences of this lemma are that $\kappa(1) = 1$, $\kappa(2) = \frac{6}{9}$, and $\kappa(n) \searrow \frac{5}{9}$ as $n \nearrow \infty$.

Proof. The proof is take from [159]. Let $I^\star := (\frac{\max(\mathcal{X})+a}{2}, \frac{\min(\mathcal{X})+b}{2}) \subset (a, b)$. The proof proceeds in two steps. (i) $\gamma(D; n, 2) = 1 \iff \mathcal{X} \cap I^\star \neq \emptyset$. (ii) $P[\mathcal{X} \cap I^\star \neq \emptyset] = \frac{5}{9} + \frac{4}{9} \cdot 4^{-(n-1)}$. (i) follows from the proof of Lemma 4.2. For (ii) note first that, since the X_i are independent and identically distributed uniform over (a, b), the random variable $1\{\mathcal{X} \cap I^\star = \emptyset\}$ is independent of a, b, and the size of the interval $b - a$. Thus, to simplify our calculations, we set $a = 0$ and $b = 1$. Let E be the event $\{\min(\mathcal{X}), \max(\mathcal{X})\} \cap I^\star \neq \emptyset$, and let $\bar{E}$ be the complement of E. Then

$$P[\mathcal{X} \cap I^\star \neq \emptyset] = P[\mathcal{X} \cap I^\star \neq \emptyset | E]\, P[E] \; + \; P[\mathcal{X} \cap I^\star \neq \emptyset | \bar{E}]\, P[\bar{E}].$$

Clearly $P[\mathcal{X} \cap I^\star \neq \emptyset | E] = 1$. To calculate $P[E]$ we use the equivalence

$$E \iff \min(\mathcal{X}) > \frac{\max(\mathcal{X})}{2} \text{ or } \min(\mathcal{X}) > 2\max(\mathcal{X}) - 1$$

and condition on $\min(\mathcal{X})$ and $\max(\mathcal{X})$, writing $\min(\mathcal{X}) = x_1$ and $\max(\mathcal{X}) = x_n$, to obtain

$$P[E] = \int_0^1 \int_{max\{min\{\frac{x_n}{2}, 2x_n - 1\}, 0\}}^{x_n} f_{x_1,x_n}(x_1, x_n) dx_1 dx_n,$$

where $f_{x_1,x_n}(x_1, x_n) = n(n-1)(x_n - x_1)^{n-2}$ is the joint probability density function of $\min(\mathcal{X})$ and $\max(\mathcal{X})$ (see, e.g., [40]). The previous expression reduces to

$$\begin{aligned} P[E] &= \int_0^{\frac{1}{2}} \int_0^{x_n} f_{x_1,x_n}(x_1, x_n) dx_1 dx_n \\ &\quad + \int_{\frac{1}{2}}^{\frac{2}{3}} \int_{2x_n - 1}^{x_n} f_{x_1,x_n}(x_1, x_n) dx_1 dx_n \\ &\quad + \int_{\frac{2}{3}}^{1} \int_{\frac{x_n}{2}}^{x_n} f_{x_1,x_n}(x_1, x_n) dx_1 dx_n \\ &= 2^{2-n} - 3^{1-n}. \end{aligned}$$

To complete our evaluation of $P[\mathcal{X} \cap I^\star \neq \emptyset]$ we give the conditional probability that $\mathcal{X} \cap I^\star \neq \emptyset$ given $\bar{E}$ with $\min(\mathcal{X}) = x_1$ and $\max(\mathcal{X}) = x_n$ as

$$\rho := P[\mathcal{X} \cap I^\star \neq \emptyset | \bar{E}] = 1 - \left(1 - \left(\frac{1 + x_1 - x_n}{2(x_n - x_1)}\right)\right)^{n-2}.$$

Conditioning on $\min(\mathcal{X})$ and $\max(\mathcal{X})$ yields

$$\begin{aligned} P[\mathcal{X} \cap I^\star \neq \emptyset | \bar{E}] P[\bar{E}] &= \int_{\frac{1}{2}}^{\frac{2}{3}} \int_0^{2x_n - 1} \rho f_{x_1,x_n}(x_1, x_n) dx_1 dx_n \\ &\quad + \int_{\frac{2}{3}}^{1} \int_0^{\frac{x_n}{2}} \rho f_{x_1,x_n}(x_1, x_n) dx_1 dx_n \\ &= \frac{5}{9} + \frac{4}{9} \cdot 4^{-(n-1)} - (2^{2-n} - 3^{1-n}). \end{aligned}$$

The desired result follows immediately. □

A moment of reflection on this result is in order. The result says that for any n, the probability that no single ball covers the observations is approximately $\frac{4}{9}$. Although for any fixed interval the probability of an observation falling in the interval goes to 1, the "magic interval" $I_j^\star$ shrinks (in n) at a rate that ensures that there is always a large probability that the interval is empty.

Lemma 4.3 allows the derivation of the distribution of $\gamma(D; n, m)$ for a particular class of random $\mathcal{C}_{n,m}$–graphs on $\mathbb{R}$. Define $\mathcal{U}(\mathbb{R}) \subset \mathcal{F}(\mathbb{R})$ as

$$\begin{aligned} \mathcal{U}(\mathbb{R}) := \{F_{X,Y} : & X \text{ and } Y \text{ are independent,} \\ & X_i \sim^{i.i.d} F_X, Y_j \sim^{i.i.d} F_Y, \text{ and} \end{aligned}$$

$$F_X = F_Y = \text{Uniform}(I), \text{ where}$$
$$I = (a,b) \subset \mathbb{R} \text{ with } -\infty < a < b < +\infty\}.$$

The following Theorem gives the exact distribution of the cardinality of minimum total dominating sets for $\mathcal{U}(\mathbb{R})$-random class cover catch digraphs.

Let Z_m denote the integers modulo m; $Z_m := \{0, \cdots, m-1\}$. Define

$$\Delta_{z,b}^{S} := \left\{ (z_1, \cdots, z_b) : \sum_{i=1}^{b} z_i = z \; ; \; z_i \in S \; \forall i \right\}.$$

Theorem 4.3. *Let D be a $\mathcal{U}(\mathbb{R})$-random $\mathcal{C}_{n,m}$-graph. Then the probability mass function for the random variable $\gamma(D;n,m)$ is given by*

$$P[\gamma(D;n,m) = d] = \frac{n!m!}{(n+m)!} \times \sum_{\vec{n} \in \Delta_{n,m+1}^{z_{n+1}}} \sum_{\vec{d} \in \Delta_{d,m+1}^{z_3}} \alpha(d_1,n_1)\alpha(d_{m+1},n_{m+1})\Phi(d,n)$$

where

$$\alpha(d,n) = \max(1\{n = d = 0\}, 1\{n \geq d = 1\}),$$
$$\Phi(d,n) = \prod_{j=2}^{m} \beta(d_j, n_j),$$

and

$$\beta(d,n) = \max(1\{n = d = 0\}, 1\{n \geq d \geq 1\}) \cdot \kappa(n)^{1\{d=1\}} \cdot (1-\kappa(n))^{1\{d=2\}}.$$

This is proved in [159].

While the expected value of $\gamma(D;n,m)$ can be obtained from Theorem 4.3, a more straightforward derivation is provided in [159]. We present the theorem here, and leave the derivation as an exercise.

Theorem 4.4. *Let D be a $\mathcal{U}(\mathbb{R})$-random $\mathcal{C}_{n,m}$-graph. Then*

$$E[\gamma(D;n,m)] = \frac{2n}{n+m} + \frac{n!m(m-1)}{(n+m)!} \sum_{i=1}^{n} \frac{(n+m-i-1)!}{(n-i)!} \cdot (2-\kappa(i)),$$

where $\kappa(i)$ is given by Lemma 4.3.

Corollary 4.1.

$$e_{n,n} = E[\gamma(D;n,n)] \to \infty. \tag{4.1}$$

Proof. From Theorem 4.4 we have:

$$E[\gamma(D;n,n)] = 1 + \frac{n!n(n-1)}{(2n)!} \sum_{i=1}^{n} \frac{(2n-i-1)!}{(n-i)!} \cdot (2-\kappa(i))$$

$$
\begin{aligned}
&= 1 + \frac{(n!)^2(n-1)}{(2n)!} \sum_{i=1}^{n} \binom{2n-i-1}{n-i} (2 - \kappa_j(i)) \\
&\geq 1 + \frac{(n!)^2(n-1)}{(2n)!} \sum_{i=1}^{n} \binom{2n-i-1}{n-i} \\
&= 1 + \frac{(n!)^2(n-1)}{(2n)!} \frac{2^{n-1}(2n-1)!!}{n!} \\
&= \frac{n+1}{2}.
\end{aligned}
$$

□

Limiting results for $\gamma(D; n, m)$ are also available.

Theorem 4.5. *(i) Let $D_{n,m}$ be a $\mathcal{U}(\mathbb{R})$–random $\mathcal{C}_{n,m}$–graph. Then for n fixed and finite, $\lim_{m\to\infty} \gamma(D_{n,m}) = n$ a.s.*
(ii) Let $G_{n,m}$ be a $\mathcal{U}(\Re)$–random $\mathcal{C}_{n,m}$–graph. Then for m fixed and finite,

$$\lim_{n\to\infty} \gamma(G_{n,m}) =^d m + 1 + B,$$

where $B \sim$ Binomial$(m-1, \frac{4}{9})$.

There are also strong law of large numbers results. For example, [45] (see also [43]) prove:

Theorem 4.6. *For $a \in (0, \infty)$, and $m = \lfloor an \rfloor$,*

$$\lim_{n\to\infty} \frac{\gamma}{n} = \frac{a(13a+12)}{3(a+1)(3a+4)}$$

almost surely.

In [218] the following extension is proved:

Theorem 4.7. *Assume the densities for X and Y are bounded with a finite number of discontinuities and $m/n \to r$. Then*

$$\lim_{n\to\infty} \frac{\gamma}{n} = \int g\left(r \frac{f_Y(u)}{f_X(u)} f_Y(u)\right) du$$

almost surely, where

$$g(a) = \frac{a(13a+12)}{3(a+1)(3a+4)}.$$

As we have seen, in the one-dimensional case the graph is determined by the disconnected regions defined by the $\mathcal{Y}$ observations. Once again, we consider the case where $m = 2$ and the n observations $\mathcal{X}$ fall between the two $\mathcal{Y}$ observations. We will further assume that the $\mathcal{Y}$ observations are the points 0 and 1. We will denote the corresponding class cover catch digraph C_n^*. The digraph corresponding to the

observations on either side of the range of the $\mathcal{Y}$ observations is easily analyzed and will not be considered here.

An example is shown in Figure 4.3. A C_5^* is shown above, with a representation of the digraph in the interval shown below. Note that CCCDs are almost always true digraphs, in the sense that their edges are typically not all bidirectional. Also, there are obviously infinitely many representations for any given CCCD digraph.

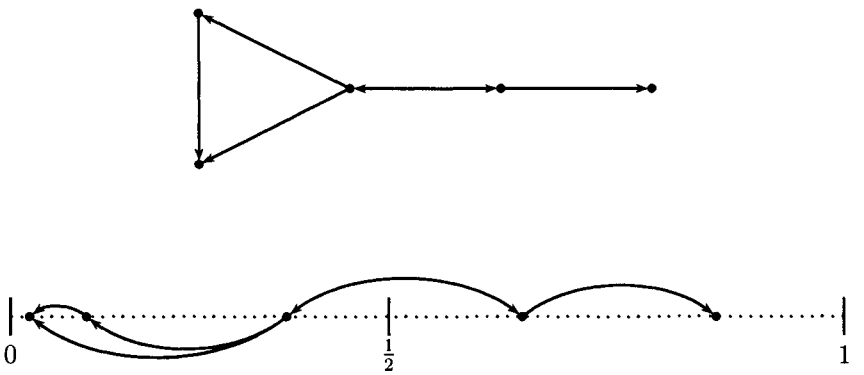


Fig. 4.3 A C_5^* digraph and its realization.

Let $\mathcal{X} = \{X_1, \cdots, X_n\}$ be i.i.d. Uniform(0,1), and let D be the C_n^* digraph for $\mathcal{X}$. We will investigate the properties of D. In particular, we are interested in calculating the probabilities for given situations, such as the probability that the digraph is complete, and in determining the distributions for various random quantities, such as the number of outgoing edges at a vertex.

Problems

4.8 Consider the target class X to be drawn uniformly (iid) on $(0, 1)$, and the non-target class drawn iid from $\frac{1}{2}U(-1, -\epsilon) + \frac{1}{2}U(1 + \epsilon, 2)$ for some $\epsilon > 0$. Show that $P(\gamma = 1) \to 1$.

4.9 Now consider the target class X to be drawn uniformly (iid) on $(0, 1)$, and the non-target class drawn iid from $\frac{1}{2}U(-1, 0) + \frac{1}{2}U(1, 2)$. What is $P(\gamma = 1)$?

Connectedness In this section we will consider the connectedness of the C_n^* digraph. In particular, we are interested in the probability that the digraph is connected, and that it is complete. Recall that a digraph is complete symmetric if every pair of vertices $\{v, w\}$ has an edge vw. Let K_n^* denote the complete symmetric digraph. Then for the C_n^* digraph D, we have

Theorem 4.8. $P[D = K_n^*] = \frac{1}{3^{n-1}}$.

Proof. Consider three cases.

Lemma 4.4. *Let* $C_1 = \{X_{(1)} < \frac{1}{2}, X_{(n)} > \frac{1}{2}, X_{(n)} \leq 2X_{(1)}$ *and* $X_{(1)} \geq 2X_{(n)} - 1\}$. *Then*
$P[C_1] = \frac{1}{3^{n-1}} - \frac{1}{4^{n-1}}$.

Proof. The proof follows by noting that the probability can be calculated via

$$n(n-1)\int_{\frac{1}{2}}^{\frac{2}{3}}\int_{\frac{y}{2}}^{\frac{1}{2}}(y-x)^{n-2}dxdy$$

$$+n(n-1)\int_{\frac{2}{3}}^{\frac{3}{4}}\int_{2y-1}^{\frac{1}{2}}(y-x)^{n-2}dxdy$$

$$= \frac{1}{3^{n-1}} - \frac{1}{4^{n-1}}.$$

This can in turn be derived by noting that the probability in question is derived by integrating the joint distribution of the extreme order statistics over the appropriate region. □

Lemma 4.5. *Let* $C_2 = \{X_{(1)} < \frac{1}{2}, X_{(n)} < \frac{1}{2}, X_{(n)} \leq 2X_{(1)}\}$ *Then*
$P[C_2] = \frac{2}{4^n}$.

Proof. The proof follows by noting that the probability can be calculated via

$$n(n-1)\int_{0}^{\frac{1}{2}}\int_{\frac{y}{2}}^{y}(y-x)^{n-2}dxdy = \frac{2}{4^n}.$$

□

Lemma 4.6. *Let* $C_3 = \{X_{(1)} > \frac{1}{2}, X_{(n)} > \frac{1}{2}, X_{(1)} \geq 2X_{(n)} - 1\}$ *Then*
$P[C_2] = \frac{2}{4^n}$.

The proof is similar to that of Lemma 4.5.

The proof of the theorem now follows from the lemmas, noting that these are precisely the cases in which the digraph D is complete. □

Obviously, C_n^* has either 1 or 2 components. This follows immediately from the geometry of the problem. The largest vertex less than 1/2 (denoted x^-) has an edge to all vertices less than 1/2, and similarly, the smallest vertex greater than 1/2 (denoted x^+) has an edge to all those greater. Thus the question of connectedness comes down

to the question of whether one of the two vertices closest to 1/2 on either side covers the other or not.

Clearly, as $n \to \infty$, the probability that C_n^* has 2 components goes to zero. Intuitively, the reason is that in order for it to have 2 components, x^- must not cover x^+ and vice versa. Thus we have

Theorem 4.9.

$$P[C_n^* \text{ is connected}] \to 1$$

as $n \to \infty$.

Proof. Consider the probability that there are two components. If all the observations are on one side of 1/2, then there is a single component, and so for there to be two components the data must straddle 1/2. Further, as mentioned above, we must have

$$\begin{aligned} 2x^- &< x^+ \\ x^- &< 2x^+ - 1. \end{aligned}$$

Adding these inequalities, and simplifying, results in the condition $x^+ - x^- > 1/3$. In particular, this means that the largest gap between order statistics must be larger than 1/3. Clearly the probability that the maximal spacing is greater than 1/3 goes to zero. See [40] for the details. □

We can actually compute the probability of a connected digraph explicitly. The probability that the digraph has exactly 1 component (e.g., that the digraph is connected) is given in the following theorem.

Theorem 4.10.

$$P[C_n^* \text{ is connected}] = 1 - \sum_{k=1}^{n-1} \left(\frac{1}{2^k} + \frac{n!}{k!(n-k-1)!} B(n,k) \right),$$

where

$$B(n,k) = \frac{1}{2^{n-k}} \beta\left(\frac{1}{3}, k+1, n-k\right) - \frac{1}{2^k} \beta\left(\frac{2}{3}, k+1, n-k\right)$$

and $\beta[z,a,b]$ *is the incomplete beta function,*

$$\beta[z,a,b] = \int_0^z t^{a-1}(1-t)^{b-1} dt. \tag{4.2}$$

Proof. Recall that the hypergeometric function is defined by

$$ {}_2F_1(a,b,c,z) = \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)} \int_0^1 \frac{t^{b-1}(1-t)^{c-b-1}}{(1-tz)^a} dt \tag{4.3}$$

$$= \sum_{j=0}^{\infty} \frac{(a)_j (b)_j}{(c)_j j!} z^j, \tag{4.4}$$

where

$$(a)_j = a(a+1)(a+2)\cdots(a+j-1) \tag{4.5}$$

is the Pochhammer symbol. Note that when one of a, b or c is a negative integer, Equation (4.4) is actually a finite sum.

If all the observations fall on one side of the midpoint of the interval, then the digraph is connected. Otherwise, the digraph is connected if either the largest observation less than the midpoint covers the smallest greater than the midpoint, or vice versa. So we calculate the probability that the digraph is *not* connected. This is the probability that there are exactly k observations less than 1/2, times the probability that the greatest less than 1/2 does not cover the least greater than 1/2 and vice versa. Let E be the event that C_n^* is not connected. Then

$$\begin{aligned}
P[E] &= \sum_{k=1}^{n-1} k(n-k)\binom{n}{k}\left(\int_{\frac{1}{2}}^{\frac{2}{3}}\int_{0}^{2y-1} x^{k-1}(1-y)^{n-k-1}\,dx\,dy\right. \\
&\quad \left. + \int_{\frac{2}{3}}^{1}\int_{0}^{\frac{y}{2}} x^{k-1}(1-y)^{n-k-1}\,dx\,dy\right) \\
&= \sum_{k=1}^{n-1}(n-k)\binom{n}{k}\left(\int_{\frac{1}{2}}^{\frac{2}{3}}(2y-1)^k(1-y)^{n-k-1}\,dx\,dy\right. \\
&\quad \left. + \frac{1}{2^k}\int_{\frac{2}{3}}^{1} y^k(1-y)^{n-k-1}\,dx\,dy\right) \\
&= \sum_{k=1}^{n-1}(n-k)\binom{n}{k}\left(\frac{{}_2F_1(k+1,1+k-n,k+2,\frac{1}{3})}{k(k+1)2^{n-k}3^{k+1}}\right. \\
&\quad + \frac{(k-1)!(n-k-1)!}{2^k n!} \\
&\quad \left. - \frac{2n!}{3^{k+1}k(k+1)n!}{}_2F_1\left(k+1,1+k-n,k+2,\frac{2}{3}\right)\right)
\end{aligned}$$

The result follows by simplification, followed by subtraction from 1. □

Theorems 4.9 and 4.10 result in the following corollary:

Corollary 4.2.

$$\sum_{k=1}^{n-1} \frac{n!}{k!(n-k-1)!}\left(\frac{1}{2^k}\beta\left(\frac{2}{3},k+1,n-k\right) - \frac{1}{2^{n-k}}\beta\left(\frac{1}{3},k+1,n-k\right)\right) \to 1$$

as $n \to \infty$.

The probability that C_n^* has a single component is plotted in Figure 4.4. The minimum value for the probability is obtained for $n = 3$.

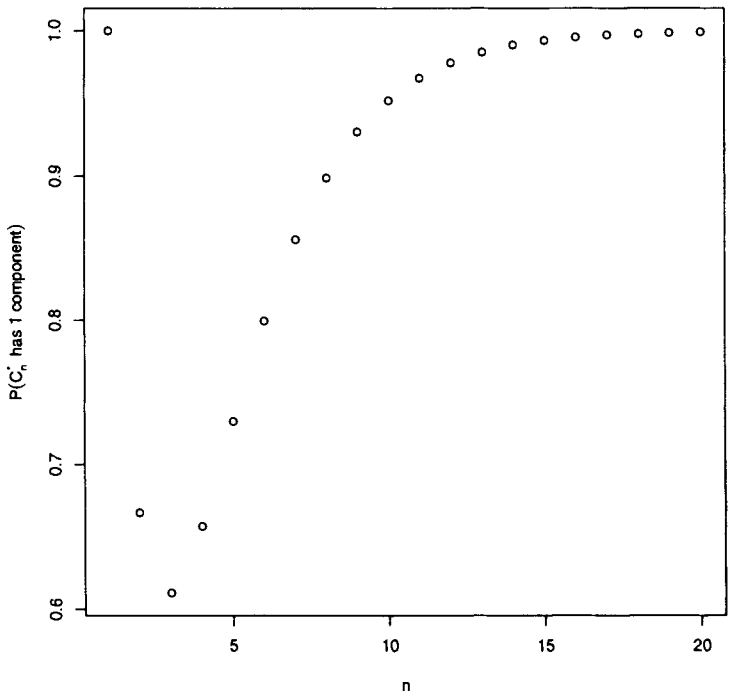


Fig. 4.4 The probability that C_n^* is connected (has a single component) as a function of n, plotted for $n = 1, \ldots, 20$.

Distribution of Edges We now focus more closely on the distributions of the edges in a C_n^*. Since this is a digraph, we are concerned with directed edges, and so treat the edges incident on (incoming) and incident from (outgoing) separately.

$$d_{in}(X) = \sum_{i=1}^{n} I\{|X - X_i| < min\{X_i, 1 - X_i\}\}; \tag{4.6}$$

$$d_{out}(X) = \sum_{i=1}^{n} I\{|X - X_i| < min\{X, 1 - X\}\}. \tag{4.7}$$

From this it is easy to see that:

Theorem 4.11. *With the above notation*

(i) $d_{in}(X) \sim Bin(n-1, \frac{1}{2})$,

(ii) $d_{out}(X) \sim U(0, ..., n-1)$.

(iii) $size = E[\#edges] = \frac{n(n-1)}{2}$.

Problems

4.10 Prove Lemma 4.1

4.11 Prove Theorem 4.4.

4.12 Prove Theorem 4.5.

4.13 Prove Theorem 4.11.

Number of Dominating Sets Recall that a set of vertices is called *independent* if there are no edges between them. We have:

Theorem 4.12. *For any C_n^*, there always exists an independent minimum dominating set.*

Proof. This is trivially true if $\gamma = 1$. Assume $\gamma = 2$. Clearly if $n = 2$ the minimum dominating set is independent (otherwise $\gamma = 1$). Assume $n > 2$. Let $\{x, y\}$ be the dominating set where $x < y$ and $y - x$ is maximal over all minimum dominating sets. Clearly $x < 1/2$ and $y > 1/2$. Assume there is an edge from x to y, i.e. that x covers y. Then there is at least one observation greater than y. Let z be the smallest observation greater than y. Then the set $\{x, z\}$ is a dominating set and $z - x > y - x$ which is a contradiction. A similar argument works for an edge from y to x. □

A minimal dominating set is one which has no nontrivial subset which is itself a dominating set. Note that for $\gamma = 1$ there can exist minimal dominating sets of order 2. For example, consider the observations $\{x_1, x_2, x_3\} = \{\frac{1}{4}, \frac{2}{5}, \frac{7}{10}\}$. Then $\{x_2\}$ is a minimum (and minimal, trivially) dominating set, while $\{x_1, x_3\}$ is a minimal dominating set.

Let N_D denote the number of distinct minimum dominating sets. If $\gamma = 1$ we have four cases:

1. If $x_{(n)}$ and $x_{(1)}$ have degree n then there are n dominating sets, and the probability of this happening is given by Theorem 4.8 as $\frac{1}{3^{n-1}}$.

2. If x_1 has degree n but x_n does not, then

$$P(n,k) = \binom{n-2}{k-1} n(n-1)$$

$$\left(\int_{\frac{1}{2}}^{1}\int_{\frac{x+1}{2}}^{1}\left(\frac{1-x}{2}\right)^{k-1}\left(y-\frac{x+1}{2}\right)^{n-k-1}dydx\right.$$

$$\left.+\int_{\frac{1}{3}}^{\frac{1}{2}}\int_{\frac{x+1}{2}}^{2x}\left(\frac{1-x}{2}\right)^{k-1}\left(y-\frac{x+1}{2}\right)^{n-k-1}dydx\right)$$

$$= \binom{n-2}{k-1}n(n-1)$$

$$\left(\int_{\frac{1}{2}}^{1}\frac{(1-x)^{k-1}(3x-1)^{n-k}}{2^{n-1}(n-k)}\left(y-\frac{x+1}{2}\right)^{n-k-1}dx\right.$$

$$\left.+\int_{\frac{1}{3}}^{\frac{1}{2}}\frac{(1-x)^{k-1}(3x-1)^{n-k}}{2^{n-1}(n-k)}\left(y-\frac{x+1}{2}\right)^{n-k-1}dx\right)$$

$$= \frac{2(n-1)!}{(n-k)!(k-1)!}\left(\frac{1}{4^n}+\frac{n}{3^k}\beta\left(\frac{1}{4},n-k-1,k\right)\right).$$

3. Similarly, if x_n has degree n but x_1 does not, then

$$P(n,k) = \binom{n-2}{k-1}n(n-1)\left(\int_{0}^{\frac{1}{2}}\int_{0}^{\frac{y}{2}}\left(\frac{y}{2}\right)^{k-1}\left(\frac{y}{2}-x\right)^{n-k-1}dxdy\right.$$

$$\left.+\int_{\frac{1}{2}}^{\frac{2}{3}}\int_{2y-1}^{\frac{y}{2}}\left(\frac{y}{2}\right)^{k-1}\left(\frac{y}{2}-x\right)^{n-k-1}dxdy\right)$$

$$= \frac{2(n-1)!}{(n-k)!(k-1)!}\left(\frac{1}{4^n}+\frac{n}{3^k}\beta\left(\frac{1}{4},n-k-1,k\right)\right).$$

4. If neither x_1 nor x_n have degree n, then

$$P(n,k) = \binom{n-2}{k}\frac{n(n-1)}{2^{n-2}}\times$$

$$\left(\int_{0}^{\frac{1}{3}}\int_{\frac{x+1}{2}}^{1}(1+x-y)^k(3(y-x)-1)^{n-k-2}dydx\right.$$

$$\left.+\int_{\frac{1}{3}}^{\frac{2}{3}}\int_{2x}^{1}(1+x-y)^k(3(y-x)-1)^{n-k-2}dydx\right)$$

$$
\begin{aligned}
&= \binom{n-2}{k} \frac{n(n-1)}{2^{n-k-2}3^{k+1}(n-k-1)} \\
&\quad \left(\int_0^{\frac{1}{3}} \left((2-3x)^{n-k-1} {}_2F_1\left(n-k-1,-k,n-k,1-\frac{3x}{2}\right) \right.\right. \\
&\quad \left. -2^{k-n+1}(1-3x)^{n-k-1} {}_2F_1\left(n-k-1,-k,n-k,\frac{1-3x}{4}\right) \right) + \\
&\quad \int_{\frac{1}{3}}^{\frac{2}{3}} \left((2-3x)^{n-k-1} {}_2F_1\left(n-k-1,-k,n-k,1-\frac{3x}{2}\right) \right. \\
&\quad \left.\left. -(3x-1)^{n-k-1} {}_2F_1\left(n-k-1,-k,n-k,\frac{3x-1}{2}\right) \right) dx \right) \\
&= \binom{n-2}{k} \frac{n(n-1)}{2^{n-k-2}3^{k+1}(n-k-1)} \\
&\quad \left(\sum_{j=0}^{k} \frac{(n-k-1)_j(-k)_j(2^{n-k-2j}-2^j-2^{k-n+1})}{(n-k)_j j! 4^j 3(n-k+j)} \right. \\
&\quad \left. + \sum_{j=0}^{k} \frac{(n-k-1)_j(-k)_j(1-2^{k-n-j+1})}{(n-k)_j j! 2^j 3(n-k+j)} \right) \\
&= \frac{4(k+1)}{3^{k+2}} - \frac{{}_2F_1(-k,n-k-1,n-k+1,\frac{1}{4})n!}{3^{k+2}4^{n-k-2}k!(n-k)!}. \qquad (4.
\end{aligned}
$$

Putting the above together, we have

Theorem 4.13. *The probability of the event $E_{k,1}$, that there are k distinct minimum dominating sets of size 1 is:*

$$
P[E_{k,1}] = \begin{cases} \frac{4(n-1)!}{(n-k)!(k-1)!}\left(\frac{1}{4^n} + \frac{n}{3^k}\beta(\frac{1}{4}, n-k+1, k)\right) + \\ \frac{4(k+1)}{3^{k+2}} - \frac{n!}{4^{n-k-1}} {}_2F_1(-k,n-k-1,n-k+1,\frac{1}{4}) & 1 \le k < n \\ \frac{1}{3^{n-1}} & k = n \\ 0 & else \end{cases}
$$

Theorem 4.14. *For any fixed k, $P[N_D = k \wedge \gamma = 1] \to \frac{4(k+1)}{3^{k+2}}$ as $n \to \infty$.*

Proof. As n increases, clearly the only case which contributes significantly to the probability is the last, where neither x_1 nor x_n have degree n. From Equation (4.8), consider the second term:

$$
\begin{aligned}
f(n,k) &:= \frac{n!}{3^{k+2}4^{n-k-2}k!(n-k)!} {}_2F_1\left(-k,n-k-1,n-k+1,\frac{1}{4}\right) \\
&= \frac{n!}{3^{k+2}4^{n-k-2}k!(n-k)!} \frac{(n-k)!}{(n-k-2)!} \times
\end{aligned}
$$

$$
\begin{aligned}
& \int_0^1 t^{n-k-2}(1-t)^2(1-t/4)^k dt \\
< \; & \frac{n!}{3^{k+2}4^{n-k-2}k!(n-k-2)!} \int_0^1 t^{n-k-2}(1-t)^2 dt \\
= \; & \frac{2n!}{3^{k+2}4^{n-k-2}k!(n-k+1)!} \\
< \; & \frac{2^{n+1}4^{k+2}}{3^{k+2}4^n(n-k+1)} \\
\to \; & 0.
\end{aligned}
$$

This completes the proof. □

From a pattern recognition standpoint the number of minimum dominating sets is important for two reasons. First, there is the desire to represent the data with the smallest possible set of "representative exemplars". This is the vector quantization problem, also seen in the reduced nearest neighbor classifier. The number of minimum dominating sets gives an indication of the difficulty of finding an appropriate exemplar set. Second, while any minimum dominating set covers the class, these sets are not necessarily equal for other purposes, such as classification or within-class clustering. For example, consider the data in Figure 4.5. Two minimum dominating sets have been identified, denoted $D_{1\cdot}$ and $D_{2\cdot}$. The second might be preferable over the first, for some applications, as it places the exemplars more centrally relative to the observations they cover.

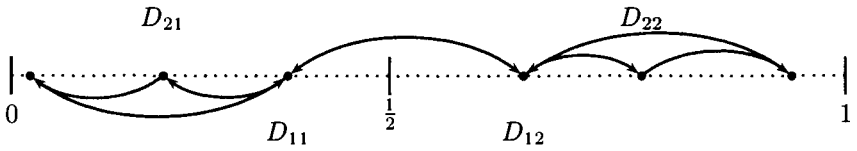


Fig. 4.5 A C_6^* digraph with several minimum dominating sets of size two.

Problems

4.14 Investigate the distribution of the number of dominating sets empirically. Draw Monte Carlo samples from $U(0,1)$ and determine the number of distinct minimum dominating sets of the corresponding CCCD (with $Y = \{0,1\}$).

Infinite CCCD This result characterizes the domination number of a class of class cover catch digraphs of order $\aleph_0$, the cardinality of the integers, and a class of class cover catch digraphs of order c, the cardinality of the continuum.

Theorem 4.15. *Consider* $\mathcal{C}(V, \{a, b\})$ *for* $-\infty < a < b < \infty$.
If conditions
(a) $V \subset (a, b)$,
(b) $\inf V = a$,
(c) $\sup V = b$
are met, then
(i) $\gamma(\mathcal{C}(V, \{a, b\}) = 1 \iff (b - a)/2 \in V$,
(ii) $\gamma(\mathcal{C}(V, \{a, b\}) = 2 \iff (b - a)/2 \notin V$.

Proof. $\gamma(\mathcal{C}(V, \{a, b\}) \leq 2$ since $D = \{\sup\{y \in V : y < (a + b)/2\}, \inf\{y \in V : y > (a + b)/2\}\}$ is a dominating set. The only open ball B for which $V \subset B$ and $B \cap \{a, b\} = \emptyset$ is $B = \{x : |x - (a + b)/2| < (b - a)/2\}$; thus a dominating set D of cardinality one, if such a set exists in V, must be $D = \{(a + b)/2\}$. □

Corollary 4.3. *Let* $-\infty < a < b < \infty$, *and let* $\mathbb{Q} \subset \mathbb{R}$ *denote the rationals.*
(i) $\gamma(\mathcal{C}(\mathbb{Q} \cap (a, b), \{a, b\}) = 1 \iff b - a \in \mathbb{Q}$.
(ii) $\gamma(\mathcal{C}(\mathbb{Q} \cap (a, b), \{a, b\}) = 2 \iff b - a \in \mathbb{R} \setminus \mathbb{Q}$.
(iii) $\gamma(\mathcal{C}((\mathbb{Q} + (b - a)) \cap (a, b), \{a, b\}) = 1$.
(iv) $\gamma(\mathcal{C}((\mathbb{R} \setminus \mathbb{Q}) \cap (a, b), \{a, b\}) = 2 \iff b - a \in \mathbb{Q}$.
(v) $\gamma(\mathcal{C}((\mathbb{R} \setminus \mathbb{Q}) \cap (a, b), \{a, b\}) = 1 \iff b - a \in \mathbb{R} \setminus \mathbb{Q}$.
(vi) $\gamma(\mathcal{C}(((\mathbb{R} \setminus \mathbb{Q}) + (b - a)) \cap (a, b), \{a, b\}) = 2$.

This has no practical consequences whatsoever (as far as we can tell).

Problems

4.15 Let p_k be the kth prime. Let $X_k = \{p_1, \ldots, p_k\}$ and $Y_k = \{1, p_k + 1\}$. For the first 100 primes, what is γ_k, the domination number for the CCCD on X_k, Y_k? What do you conjecture is the answer for all k, as a function of k? Can you prove it?

Characterization Conjecture We will see in Section 4.5 that CCCDs can be characterized in fairly simple terms, provided we are allowed to consider CCCDs generated by multivariate data. Such characterizations are currently unavailable for the CCCDs restricted to one dimension. In this section, we will describe a conjecture by DeVinney that would give a nice characterization of C_n^* digraphs.

Definition 4.6. *An augmented adjacency matrix has the* **consecutive ones property** *for rows and columns if within any row or column the ones in the matrix all appear consecutively.*

Definition 4.7. *For an augmented adjacency matrix* M, *define the* **middle ones square property** *as follows. A vertex* v_i *is a* **right vertex** *if* $M_{i,1} = 1$. *Similarly a*

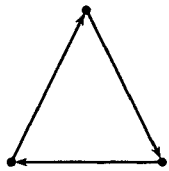


Fig. 4.6 A digraph with independence number $\beta = 1$ and domination number $\gamma = 2$.

left vertex v_j *has* $M_{j,n} = 1$. *A left or right vertex* v_i *is a* **reach vertex** *if there exists a right or left (respectively) vertex* v_j *such that* $M_{i,j} = 1$ *and* $M_{j,i} = 0$. *The middle ones property then holds if for any reach vertices* v_i *and* v_j, $M_{i,j} = M_{j,i} = 1$.

Conjecture 4.1 (DeVinney). *A digraph* D *with* n *vertices is a* $C_n^\star$ *digraph if and only if its augmented adjacency matrix* M *has the following properties:*

1. *The consecutive ones property for rows and columns.*
2. *The middle ones property.*
3. $M_{i,1} + M_{i,m} \geq 1$.

Problems

4.16 Prove or provide a counterexample for Conjecture 4.1.

4.4.2 Multivariate CCCDs

We have seen that $\gamma \in \{1, 2\}$ for univariate observations drawn between two Y points. We now turn to the problem of providing bounds for γ in higher dimensions.

What is the expected size of the minimum dominating set in $\mathbb{R}^q$? In order to explore this question we have taken two paths. First, we provide an upper bound for the size of the dominating set in terms of the kissing number. Then, we explore some simulations to estimate γ.

First, we note that the domination number is bounded above by the independence number ([43]). Note that while this is always true for graphs, it is not true for digraphs in general. See Figure 4.6.

Theorem 4.16 (DeVinney). *For any class cover catch digraph* D, *the independence number,* $\beta_D \geq \gamma_D$.

Definition 4.8. *A class cover catch digraph induced by points in* $\mathbb{R}^q$ *under the Euclidean metric (*L_2*) is called a* **Euclidean CCCD**. *Similarly, any CCCD isomorphic to a Euclidean CCCD is also called a Euclidean CCCD.*

Definition 4.9. *A* **kissing set** *in* $\mathbb{R}^q$ *is a set of centers of nonintersecting open balls of radius 1 whose boundaries intersect the boundary of a ball of radius one centered*

at the origin. The **kissing number**, $\tau(q)$, *is the size of the largest possible kissing set in* $\mathbb{R}^q$.

With this definition in hand, [43] proves

Theorem 4.17 (DeVinney). *For any Euclidean CCCD* D *in* $\mathbb{R}^q$, *where* $Y = \{0\}$, $\beta_D \leq \tau(q)$.

Clearly the Y point need not be the origin; any singleton Y set will do. Thus putting Theorems 4.16 and 4.17 together, it is easy to see that $\gamma_D \leq m\tau(q)$ for any Euclidean CCCD D.

Consider drawing X and Y uniformly in the unit square in $\mathbb{R}^2$. Let $n = |X|$ be fixed, and consider varying $m = |Y|$. We will assume that $m << n$. In Figure 4.7 we present the results for 100 runs, with $n = 1500$ and $m = 1, \ldots, 10$. We see that $\gamma \approx 2m$.

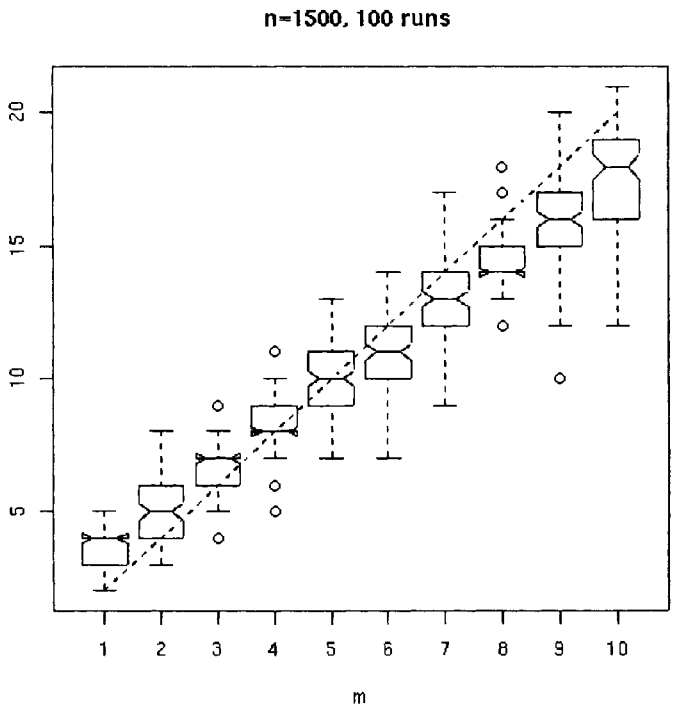


Fig. 4.7 $X, Y \sim U(0,1)^2$. 100 simulations were performed and $\hat{\gamma}$ was calculated using the greedy algorithm. Box plots depict the results with the line $\hat{\gamma} = 2m$ plotted for comparison.

This example shows that the bounds we have are, in general, much too large. Note that we cannot loosen the bounds (see Problem 4.17). So this simulation shows that

the result $\gamma = m\tau$ is a rare event (at least for these distributions) and, in general, γ is much smaller.

Problems

4.17 Show that the bound $\gamma \leq \tau(2)$ is tight for a Euclidean CCCD with a single non-target point; that is, construct $\tau(2) = 6$ observations in $\mathbb{R}^2$, with non-target point at the origin, so that the resultant CCCD is the empty digraph.

4.18 Redo the simulation depicted in Figure 4.7 with $q = 3$. What do you conjecture about the domination number for larger values of q based on this (admittedly small) sample?

4.5 CHARACTERIZATIONS

First, we have a very elegant characterization for sphere digraphs.

Theorem 4.18. *A sphere digraph contains no simple cycles.*

See [43] for a proof. Note that we have not stated the theorem as an if and only if. However we obtain this stronger result as a corollary of the following characterization of Euclidean CCCDs.

Theorem 4.19 (DeVinney). *A CCCD is Euclidean if and only if it has no simple cycles.*

Note that the theorem does not state the value of q, the dimension in which a representation exists, nor does it specify the number of Y observations required (although, in fact, only a single non-target observation is needed). A more interesting result would be the minimum q such that the CCCD was Euclidean. No results are available for this, at the time of this writing.

Clearly any CCCD is a sphere digraph. At first it may seem obvious that a sphere digraph can be represented as a CCCD: simply add points on the boundaries of the spheres to correspond to the defining non-target observations. The problem with this is that a given representation of the sphere digraph may have spheres that are completely internal. See Figure 4.8. Clearly the balls in the figure cannot correspond to a cover as defined by a CCCD, and so one must work a little harder to show the other direction of the theorem. Of course, the fact that this particular representation of the digraph is not one resulting from a CCCD does not imply that there is no other such representation. Also, it is important to note that there are sphere digraphs in a given dimension q that are not class cover catch digraphs in $\mathbb{R}^q$ (see Problem 4.20).

Problems

4.19 Produce a set of points, $X, Y \subset \mathbb{R}^2$ having the CCCD shown in Figure 4.8.

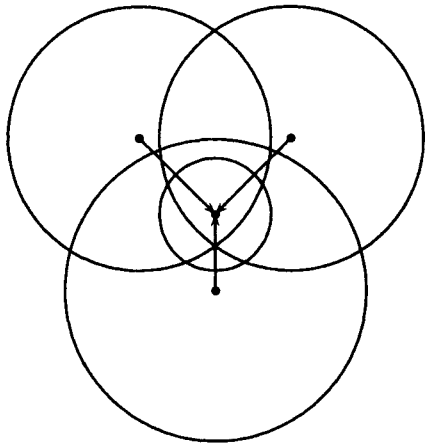


Fig. 4.8 Example sphere digraph. The spheres in this cover cannot have come from a CCCD cover, since the middle sphere is entirely interior to the others.

4.20 Exhibit a one-dimensional sphere digraph that is not a one-dimensional CCCD.

4.6 SCALE DIMENSION

The CCCD can be used to provide a nonlinear projection of the data to a (possibly lower) dimension in which to perform classification. In its simplest form, this projection is defined by the distance to each ball in the dominating set of the CCCD. This is illustrated in Figure 4.9. For each element of the dominating set, a new variable is defined as the distance of the observations to this element. Thus, we obtain a projection (depicted in the bottom figure) where each axis corresponds to the distance to a dominating element. In this way, points close to X observations fall in the rectangles near the axes (indicated by the dotted lines), while Y observations fall into the upper right quadrant.

Of course, if the domination number is large, this may result in an increase in the dimensionality. Thus, we might ask if the dominating elements might be grouped in some way to determine, in some sense, the best low-dimensional projection. This is the scale dimension problem considered in [125].

Given a set $\mathcal{W} \subset X_0$, the preclassifier defined by the CCCD is given by:

$$g_{\mathcal{W}}(x) = \begin{cases} 0 & \text{if} \quad x \in \cup_{w_i \in \mathcal{W}} B(w_i, r_i) \\ 1 & \text{else,} \end{cases} \tag{4.9}$$

where r_i is the radius associated with vertex w_i. Observations interior to balls in $\mathcal{W}$ are classed as target, the rest as nontarget.

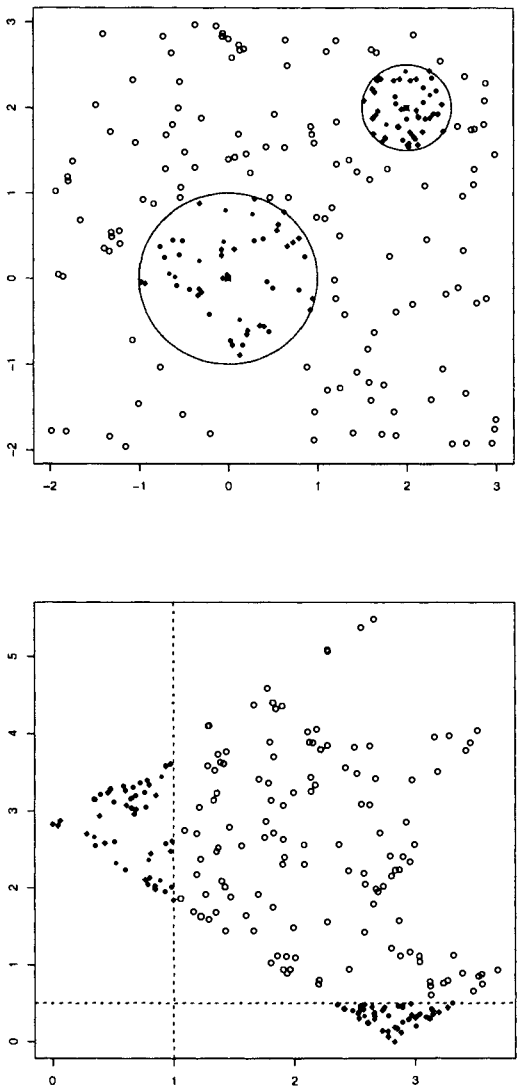


Fig. 4.9 An illustration of a CCCD-defined projection. The data are depicted on the top, with the X observations indicated by solid dots. The two elements of the dominating sets are indicated by the "x" symbols. On the bottom are the projections defined by the distances to the elements of the dominating set.

Now, for a set $\mathcal{W}_i \subset X_0$ with associated radii $\mathcal{R}_{\mathcal{W}_i} = \{r_j : v_j \in \mathcal{W}_i\}$, choose a method for selecting a single radius $r_{\mathcal{W}_i}$. We will use $r_{\mathcal{W}_i} = \min(\mathcal{R}_{\mathcal{W}_i})$; other reasonable choices are to use the maximum, mean or median. Now, for a collection of (disjoint) sets $W_k = \mathcal{W}_1, \ldots, \mathcal{W}_k$ we define

$$g_{W_k}(x) = \begin{cases} 0 & \text{if } x \in \cup_{w_i \in \mathcal{W}_i} B(w_i, r_{\mathcal{W}_i}) \\ 1 & \text{else.} \end{cases} \tag{4.10}$$

This allows us to make a trade-off between the projection dimension and the performance of the preclassifier, and hence, presumably, the performance of any subsequent classifier applied to the projected data. Let $\hat{L}_{g_{W_k}}$ be the resubstitution error of the preclassifier $g_{W_k}(x)$. We seek to find $d^* = min\{\text{argmin}_k \hat{L}_{g_{W_k}} + \alpha k\}$. This is the scale dimension, and is interpreted as the number of distinct scales (ball radii) necessary to cover the data.

The parameter α determines the definition of the "elbow" of the dimension/error curve, giving the trade-off between dimensionality and performance. A pedagogical illustration of this is shown in Figure 4.10. The "elbow" in the scale dimension curve is at $d^* = 2$. This groups the five smaller balls together, with a common radius indicated by the dotted circles. The associated projection is shown in the bottom right figure. Note that in this case the resubstitution error is not 0 (as it would be in the 6 dimensional space defined by the dominating set). In fact, the resubstitution error is approximately 2.7%. However, the data are correctly summarized as being of two distinct scales.

Problems

4.21 What is the scale dimension of the Y observations in Figure 4.10? You may want to perform a Monte Carlo simulation to answer this question.

4.22 Generate data as in Figure 4.10. Instead of using dominating sets, use the full class cover catch digraph to investigate the scale dimension. Do you obtain the same answer? What are the advantages and disadvantages of using the dominating set?

4.6.1 Application: Latent Class Discovery

In this section we will investigate the gene expression data of Section 1.4.3. Recall that these data consist of 72 observations in 7129 dimensional space. The data are split into observations from two different types of leukemia, X_0 = ALL ($n_0 = 47$) and X_1 = AML ($n_1 = 25$).

We will focus on the CCCD constructed from X_0 against X_1. The greedy dominating set algorithm (Algorithm 4.1) returns a set of size 21.

The scale dimension suggested by this dominating set is 5 (see [166] for more details). Clustering the covering balls into five clusters, using agglomerative hierarchical clustering, we discover that the cluster containing the smallest balls, those closest to the AML leukemia observations, contain 8 of the 9 T-cell observations.

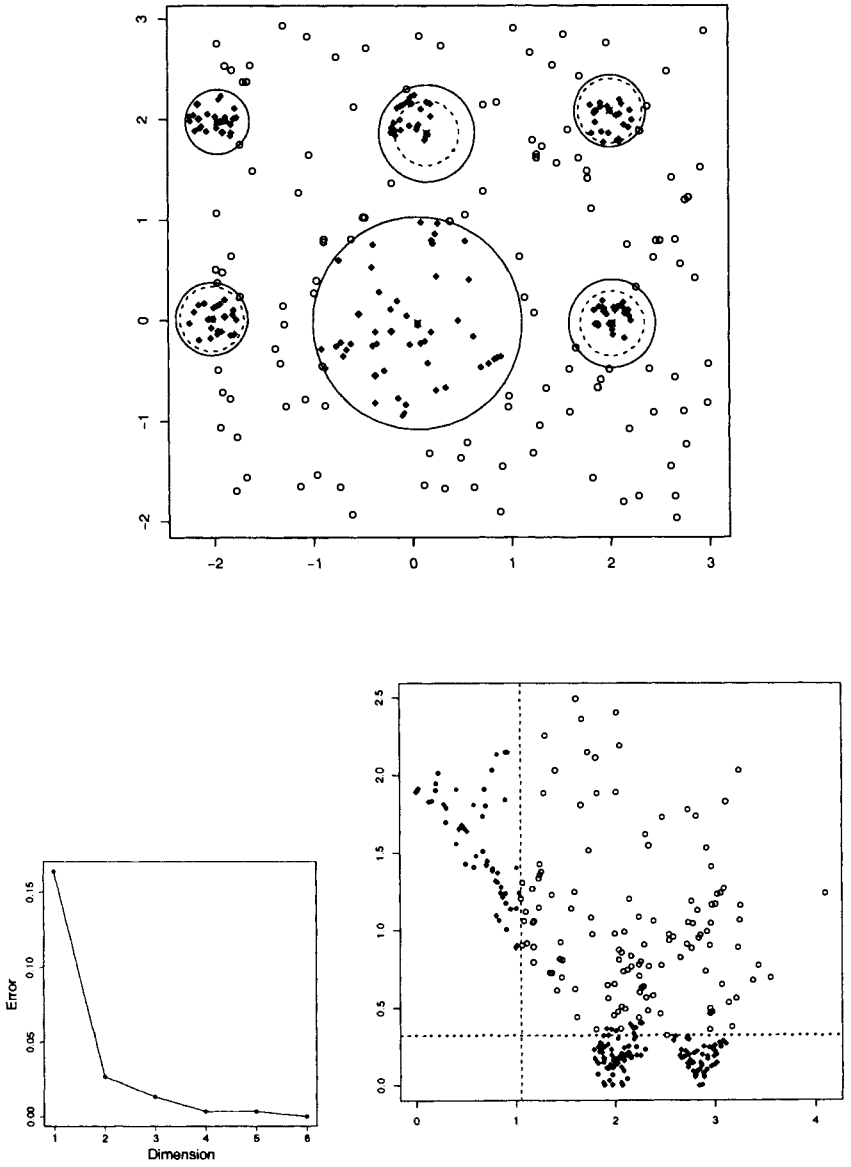


Fig. 4.10 An illustration of scale dimension. The upper plot shows the data, with the solid circles corresponding to the radii of the dominating elements of the CCCD. The bottom left plot shows the scale dimension plot, which indicates a dimension of 2. The right plot shows the corresponding projection to 2 dimensions.

From this, we hypothesize that ALL T-cell leukemia is closer, in gene-expression space, than the B-cell subclass.

This study was done on the full-dimensional data, without the scaling and dimensionality reduction that is typical for these problems. However, further experimentation has shown that the results hold after scaling and various types of dimensionality reduction, and so, at least for this data set, we feel confident that the results are valid.

This analysis falls under the label **latent class discovery** since the T-cell/B-cell distinction was not used in the analysis, and was discovered after the fact. Generally, a latent class is a subclass which was discovered via classification or clustering. The standard method for latent class discovery is to cluster the data from the individual classes. The CCCD allows us to define a different kind of latent class: those that are a similar distance from the decision boundary. We cluster the dominating set cover by radius, and look for structure within these clusters.

It is legitimate to question whether the choice of dominating set effects the outcome of this experiment. We used the greedy algorithm to find an approximately minimum dominating set. What happens if we use a truly minimum dominating set? What if there are many such sets? How does the result depend on the choice of dominating set? We now look at these questions.

It turns out that 16 of the vertices have in-degree 0, and so we can apply Algorithm 4.3 and discover that the greedy solution is in fact a minimum dominating set. In fact, there are precisely 180 distinct minimum dominating sets for this CCCD digraph, 14 of which could result from the greedy algorithm, depending on how ties are broken.

Of the 31 vertices that are not of in-degree 0, 17 of them (slightly more than half) show up in one of the minimum dominating sets. Table 4.1 shows the vertices and the number of minimum dominating sets the vertex falls in. Vertices 10, 36 and 38 fall in exactly half the dominating sets.

Table 4.1 The number of minimum dominating sets containing each vertex, for the gene expression data. Vertices which are not elements of any minimum dominating set are not listed.

Vertex	# Dominating Sets	Vertex	# Dominating Sets
3	54	4	36
5	24	6	18
8	60	9	60
10	90	12	72
24	36	23	18
15	24	25	54
29	60	33	60
36	90	37	54
38	90		

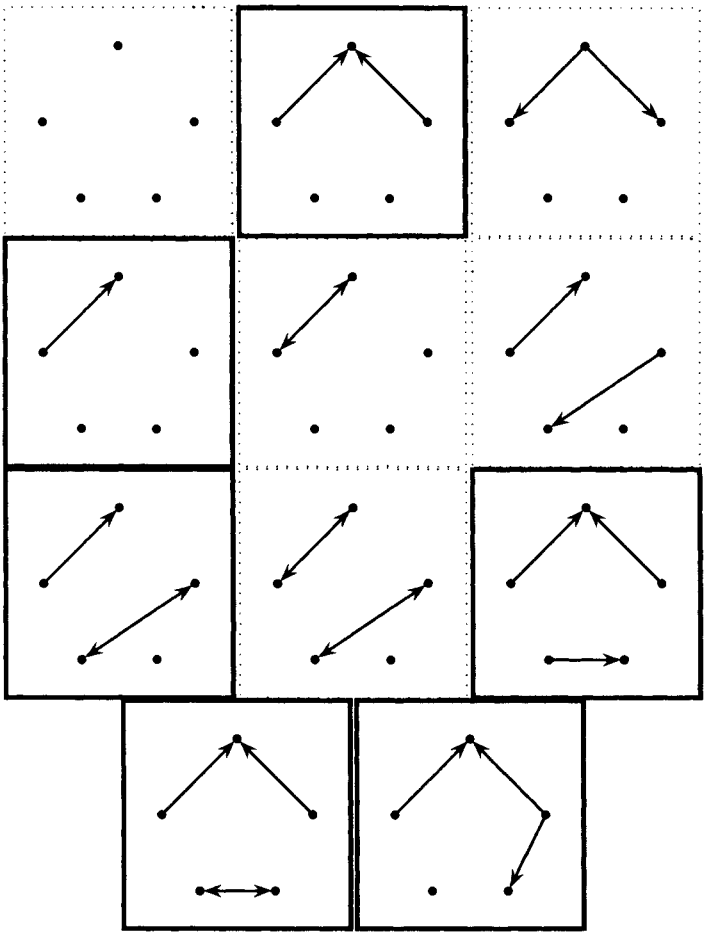


Fig. 4.11 Unique induced subraphs on the 5 changing vertices in the 180 dominating sets. Dark boxes correspond to those graphs found among the 14 greedy solutions.

We are interested in how redundant this cover is, and so we consider the subgraphs induced by these vertices. The induced graphs of the 5 vertices that differ between dominating sets are shown in Figure 4.11. This shows how interconnected the 5 vertices can be. In Figure 4.11, a bold box indicates that the corresponding graph was associated with one of the 14 greedy solutions.

Figure 4.12 shows the distributions of the radii for the vertices of in-degree 0 compared with the other vertices that appear in at least one minimum dominating set. There is no significant difference between these. Further, a hierarchical clustering of these radii has the in-degree 0 vertices well mixed throughout the clusters.

Do the different dominating sets have any effect on the latent class discovery process? The answer is no, at least for this data set. To detect the T-cell latent class,

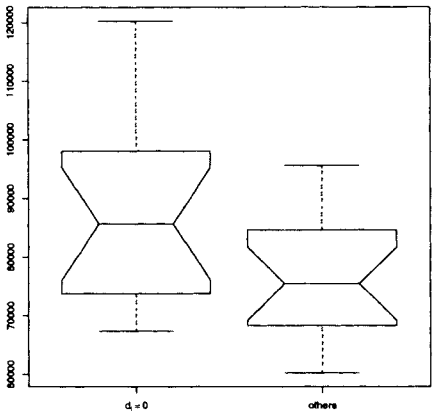


Fig. 4.12 Box plots of the radii for the in-degree 0 vertices compared to those of vertices of in-degree > 0 which appear in a minimum dominating set.

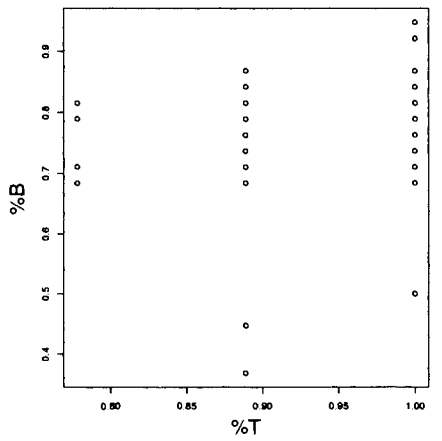


Fig. 4.13 Plot of the percent of T-cells that are in the largest cluster against the percentage of the B-cells that are in pure clusters for each of the 180 dominating sets.

we must have a cluster that contains nearly all of the T-cell observations. Thus we consider the cluster with the largest number of T-cells, and measure the percentage of T-cells in that cluster. Similarly, the B-cell balls should be pure, and so we measure the percentage of B-cells that are in pure balls. These values are plotted in Figure 4.13 for all 180 dominating sets.

Note that all the dominating sets detect the T-cell latent class, in that 80% or more of the T-cells are covered by a single ball. The B-cell purity is less consistent, but nearly all the dominating sets have over 70% of the B-cell observations covered by pure (B-cell only) balls. Thus there is reason be believe that, at least in this case, the choice of dominating set does not have a significant effect on the discovery of the latent class.

This experiment gives more support to the claim that reducing the complexity through a minimum (or approximately minimum) dominating set can be used for useful and interesting pattern recognition tasks. Further, it appears that in at least some applications, the problem of finding a true minimum dominating set is tractable. Clearly, for very large graphs this may not be the case. See the problems for some further examples, where perhaps the reduction to the dominating set is not desirable.

Problems

4.23 Draw observations from

$$X_0 \sim U(B((0,0),2) \setminus B((0,0),1)),$$

$$X_1 \sim U(B((0,0),1))$$

and perform latent class analysis, using $d^* = 2$. What do the "latent classes" correspond to? Are they real, or illusionary?

4.24 Repeat Problem 4.23 using

$$X_0 \sim \frac{1}{2}U(B((0,0),2) \setminus B((0,0),1)) + \frac{1}{2}U(B((0,0),3) \setminus B((0,0),2+\epsilon)).$$

Should the analysis be done on the dominating set or the original digraph? Can you construct a test for the existence of a latent class in this situation? What is the power of this test as a function of ϵ?

4.25 Following on from Problem 4.24, let $B_r^+ = \{(x,y) \in B((0,0),r) : x,y > 0\}$, and $B_r^- = \{(x,y) \in B((0,0),r) : x,y < 0\}$, and set

$$X_0 \sim \frac{1}{2}U(B_2^+ \setminus B_1^+) + \frac{1}{2}U(B_3^- \setminus B_2^-).$$

Redo Problem 4.24 with this distribution. Do you come to a different conclusion?

4.26 Consider the data

$$X_0 = \{(0.5,-0.1),(0.5,0.0),(0.5,0.1),(0.5,0.9),(0.5,1.0),(0.5,1.1),$$
$$(0.4,0.0),(0.6,0.0),(0.4,1.0),(0.6,1.0),(0.5,0.6)\}$$

and $X_1 = \{(0.0, 0.0), (0.0, 1.0)\}$. List all the minimum dominating sets for the CCCD. What percentage of the dominating sets "discover" the two latent classes corresponding to the clusters around the points $(0.5, 0)$ and $(0.5, 1)$? How many dominating sets fail to find both latent classes? Which of the dominating sets are possible greedy solutions?

4.27 Using the CCCD constructed in Problem 4.26, find the maximum cliques in the digraph. How do these compare to the dominating sets as elucidaters of latent classes?

4.7 (α, β) GRAPHS

The constraint that the cover on X be pure (contain no non-target points) and proper (contain all the target points) may not be appropriate for some applications, and can be relaxed in a number of ways. In this section we will look at some possible approaches to relaxing these constraints.

Essentially, we want to be able to allow some balls to cover a few non-target points, if this allows them to cover sufficiently more target points, and we'd like to allow for some of the target points (outliers) to remain uncovered.

As above, we let n correspond to the number of target (X) points, and m the number of non-target (Y) points. Let β be a nonnegative integer. Let $d_\beta(x, Y)$ denote the $(\beta + 1)$th order statistic of the distances from the point x to the set Y. If $\beta \geq m$ we define the distance to be infinite (or any sufficiently large value). Thus, d_0 is the distance used to define the balls in the standard CCCD. If we use d_1 instead, then each ball contains a single non-target point. The CCCD defined by the distance d_m is the complete symmetric digraph.

Let G denote the standard CCCD. Writing $Q \lhd G$ if Q is an induced subgraph of G, we have a sequence of digraphs

$$G = G_0 \lhd G_1 \lhd \ldots \lhd G_m = K_n^\star.$$

By adjusting the value β we can control the amount of "impurity" of the associated balls.

Outliers can be removed by simply removing balls (and their associated vertices) containing fewer than, say, α observations. An alternative approach, which we prefer, is to use the dominating set algorithm for this purpose: stop the greedy dominating set algorithm when all but α points have been covered. Thus, by adjusting α and β we can construct covers with varying "purity" and "properness".

Focusing on the purity issue for the moment, it may seem that having all balls contain β non-target points may not be desirable. We'd like to only add non-target points if by doing so the ball covers sufficiently many new target points. Thus we want to optimize some formula of the form

$$\sum |B(x, r_x) \cap X| - \lambda |B(x, r_x) \cap Y|. \tag{4.11}$$

The simple approach discussed here is illustrated in Figures 4.14 and 4.15. Figure 4.14 shows target data generated uniformly on the unit disk, with a single outlier at $(-1.5, -1.5)$, and non-target data uniform on the square $[-2, 2]^2$. The cover corresponding to the dominating sets are shown in each plot for various values of α and β. The singleton outlier is easily handled by a value of $\alpha = 1$ as shown in the figure. Different values of β provide different levels of coverage for the support of the target class, with fewer balls needed as β increases. Finally, for $\beta = 18$ (the number of non-target observations within the disk) a single ball suffices to cover the target class observations.

It is easy to compute the "correct" value for β in this example. Given that the Y observations are uniform, the proportion of these expected in the support of the X random variable is the ratio of the areas, or approximately 20%. Thus we should choose $\beta \approx 0.2m$. In this sample a value of 18 worked.

This example illustrates that even though the requirement that each ball cover β non-target observations is clearly sub-optimal, the method can produce interest results nonetheless. Figure 4.15 shows another example on real data. These data consist of six spectral bands collected from buried mines and mine-like objects. We have plotted the first two principle components in order to visualize the data and the class cover catch digraphs. The CCCD is plotted for $\beta = 0, 1, 2, 7$. This last is the smallest value of β for which the digraph is weakly connected. By analyzing the resultant graphs, we can get some idea about the geometry of the decision region in the higher-dimensional space, and the relationships between the different objects.

For example, for $\beta = 2$ the digraph (minus the outlier) is weakly connected. Further, the central points are highly connected, even for $\beta = 1, 2$. Thus, the central points seem likely to be solidly in the support of the mine distribution, while the three points that are above this region in the plots correspond to points near the boundary. There is reason to believe that the outlier to the left is a misclassification of a mine-like object, and the fact that it shows up as an outlier in every analysis we are aware of bears this out.

We will consider one approach to the optimization of Equation (4.11) in Section 4.11. Other approaches are discussed and illustrated in [125] and [43].

Problems

4.28 The mine data are shown in Table 4.2. Redo the experiments in this section on these data using mine-like objects as the target class.

4.8 CCCD CLASSIFICATION

We have already seen a simple classifier built on the CCCD given by Equations (4.9) and (4.10). This is a one-sided classifier, in that only the X observations form a CCCD. We could just as easily have used the Y observations to construct a CCCD. Consider doing just this, and call the resulting covers (after computing dominating sets, if desired) $\mathcal{C}_X$ and $\mathcal{C}_Y$. These sets correspond to X (Y) observations and their

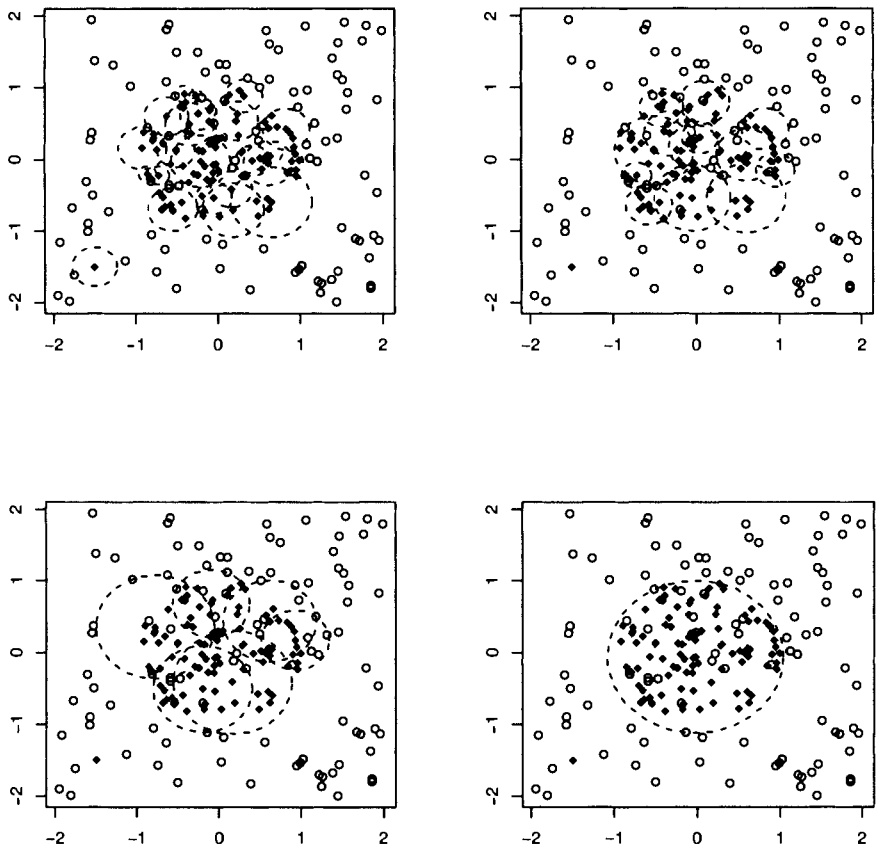


Fig. 4.14 Dominating sets defined by the (α, β) graph with parameters (left to right, top to bottom): $\alpha = \beta = 0$; $\alpha = 1, \beta = 1$; $\alpha = 1, \beta = 5$; $\alpha = 1, \beta = 18$.

Table 4.2 The mine data. The first column corresponds to class, with 0 corresponding to mine and 1 to mine-like object.

0	0.678	0.352	0.123	0.276	0.365	0.641
0	0.854	0.09	−0.962	0.558	0.449	0.433
0	0.456	0.675	0.367	0.661	0.383	0.706
0	1.106	1.523	2.33	2.939	3.118	2.369
0	−1.149	−0.139	0.619	0.148	0.097	1.478
0	1.487	0.211	0.337	0.382	0.442	0.262
0	0.708	−1.027	0.352	0.121	0.375	0.414
0	−0.95	−0.455	−1.221	0.551	0.354	0.609
0	−0.985	−1.202	−1.351	0.774	0.252	0.531
0	−0.153	−0.583	−0.648	0.176	0.061	0.973
0	−0.61	1.832	0.291	0.258	0.175	0.251
0	−1.442	−1.276	−0.236	0.597	0.48	0.451
1	−0.654	−0.082	−0.963	0.527	0.938	0.493
1	0.367	1.075	1.97	0.901	1.132	0.08
1	−0.583	−0.57	−0.425	0.556	1.167	0.496
1	0.087	−0.66	−1.008	1.133	0.636	0.834
1	−0.193	0.003	−0.628	2.785	0.843	0.949
1	1.631	2.814	1.535	0.99	0.971	1.795
1	2.416	2.176	0.557	1.38	1.239	0.636
1	−0.671	−0.795	−0.791	1.022	0.821	0.538
1	−0.402	−0.839	1.97	0.552	0.323	0.612
1	0.95	−0.127	1.12	0.287	0.094	0.544
1	−0.841	−0.914	−1.062	0.872	1.134	0.791
1	−0.462	−0.899	−0.879	0.884	1.064	0.56
1	−0.023	−0.271	−0.025	0.492	0.161	0.93
1	−0.127	0.482	−0.292	0.208	2.803	0.977
1	−0.088	0.571	0.903	0.888	0.338	0.496
1	2.064	0.99	1.288	0.096	1.599	0.54
1	−0.94	−1.009	−0.968	1.271	1.029	3.222
1	−0.506	−0.301	0.335	0.479	0.282	0.795
1	−0.786	−1.034	−0.825	0.435	0.932	1.047
1	−0.819	−0.276	0.029	0.699	0.593	0.633
1	0.23	−0.107	0.652	0.568	0.388	0.91
1	−0.143	−0.346	0.394	0.61	0.353	0.429
1	2.136	1.658	0.345	0.297	0.417	0.853
1	−1.253	−0.934	1.092	1.91	1.197	0.748
1	−0.934	−0.854	0.959	1.089	0.813	0.996
1	0.109	0.272	0.978	0.463	0.375	0.278
1	−0.566	−0.022	1.101	0.52	0.287	0.382

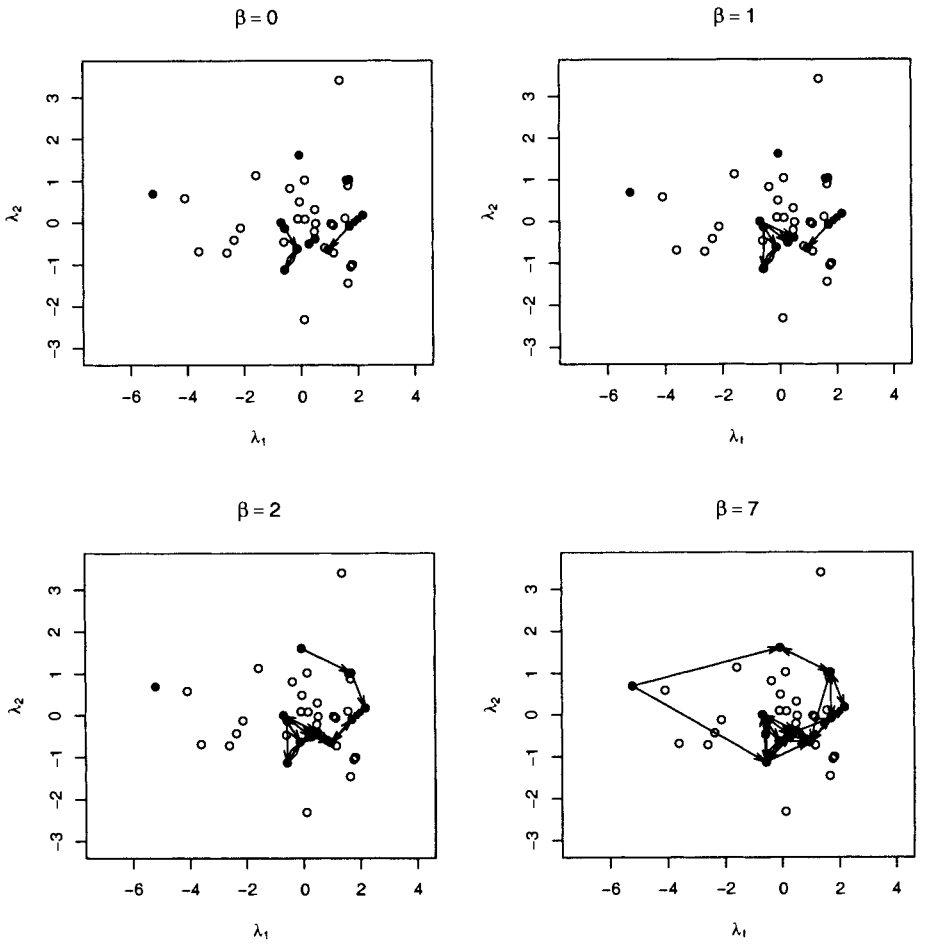


Fig. 4.15 Multispectral mine data. (α, β) graphs for several values of β. The data have been projected to two dimensions via principal components.

radii, chosen (for now) so that the balls are pure, containing only X (Y) observations. The classifier defined by these covers is then

$$g(z) = \begin{cases} 0 & \text{if} \quad d(z, \mathcal{C}_X) < d(z, \mathcal{C}_Y) \\ 1 & \text{if} \quad d(z, \mathcal{C}_Y) < d(z, \mathcal{C}_X) \\ -1 & \text{otherwise.} \end{cases} \tag{4.12}$$

The value -1 indicates no decision. The function d could be the Euclidean distance, or it could be a scaled distance. For example, we prefer to scale by the radius. Thus,

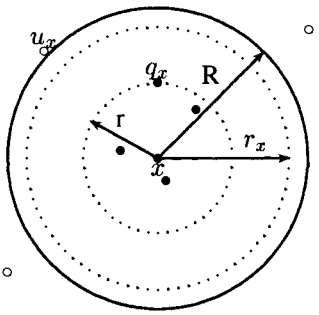


Fig. 4.16 The ball radius r_x is adjusted to fall between the minimum (r) and maximum (R) covering radii.

setting ρ to be Euclidean distance,

$$d(z, B(x, r_x)) = \rho(z, x)/r_x \tag{4.13}$$

$$d(z, C) = \min_{x \in C} \rho(z, x)/r_x. \tag{4.14}$$

A further modification is often made, to adjust the ball radius. Recall that the radius is defined as the largest value such that the ball contains no non-target observations. However, any value between the last target covered and the first non-target point will work for the cover. Thus we define an adjustment factor as follows. Let

$$u_x = \underset{z \in Y}{\text{argmin}}\{d(x, z)\}$$

be the closest non-target point to x and

$$q_x = \underset{z \in X}{\text{argmax}}\{d(x, z) : d(x, z) < d(x, u_x)\}$$

be the target point in x's ball that is furthest from x. Then we set the new radius of x to be

$$r_x = (1 - \tau)d(x, q_x) + \tau d(x, u_x). \tag{4.15}$$

τ is a user-settable parameter (which may, in principle, be set separately for each x). This is illustrated in Figure 4.16.

We can use the same basic idea for (α, β) CCCDs, with appropriate adjustment to the definition of u_x and q_x. This can allow us to ignore outliers, of either class, and better model the decision region.

A relatively simple problem is illustrated in Figure 4.17. The data are plotted in the top plot, with the dotted circles indicating the dominating sets for the class cover catch digraphs on each of the classes. The decision region of the resultant classifier is presented in the lower plot, with the true decision region represented by the circle. In this case the classifier uses the distances scaled by the ball radii, and both CCCDs have $\alpha = \beta = 0$.

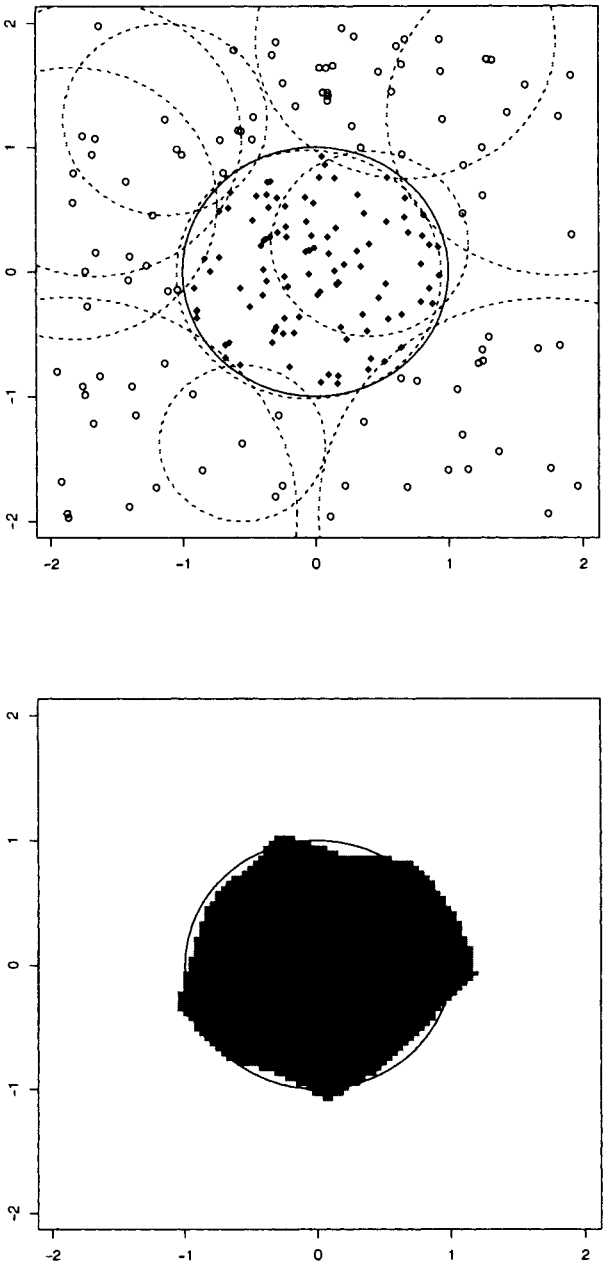


Fig. 4.17 A simple two-class simulation. The top plot shows the data and the dominating sets for the class cover catch digraphs on each class. The bottom plot shows the decision region of the CCCD classifier.

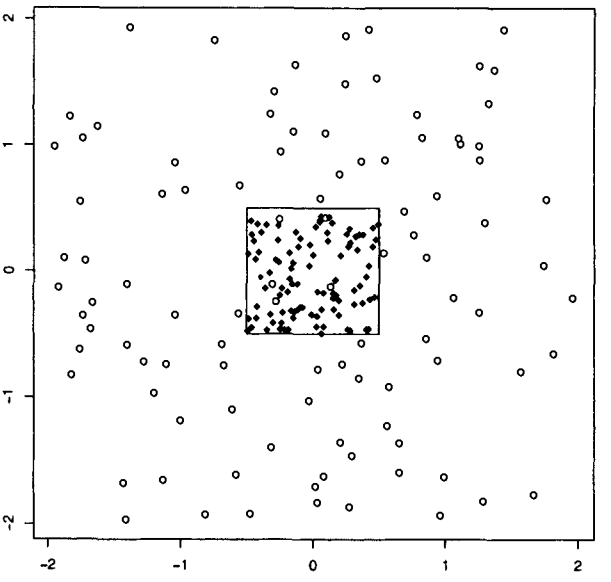


Fig. 4.18 A simple two-class simulation. 100 observations each were drawn from $X_0 \sim U([-\frac{1}{2}, \frac{1}{2}])$ and $X_1 \sim U([-2, 2]^2)$.

A slightly more interesting example is given in Figure 4.18. In this, class 0 is uniform on a unit square, and class 1 is uniform on the square $[-2, 2]^2$. This experiment is interesting for two reasons. First, the decision region is no longer a circle, and hence the CCCD classifier is not necessarily well matched to the problem, the way it was in Figure 4.17. Second, the classes overlap, allowing us to investigate the effect of α. Figure 4.19 shows the decision regions for three versions of the CCCD classifier, and for a nearest-neighbor classifier. As can be seen, the CCCD classifier, with $\alpha_1 = 6$ is the best of the four, and does quite a good job of determining the decision region. Note that the number 6 was not chosen arbitrarily, but rather from the ratio of the areas of the supports of the two classes. It would need to be estimated in a real problem.

In the examples of Figures 4.17–4.19 we have chosen to set $\tau = 1$ throughout.

The main difference between the $\alpha = 0$ case and $\alpha = 6$ case is the filling in of the central gaps caused by class 1 points in the interior of the smaller square. These gaps are also evident in the nearest-neighbor classifier.

In Figure 4.20 we vary the βs. In all these experiments, $\alpha_0 = 0$ and $\alpha_1 = 6$. In the top two plots the values of β are equal for the two classes. Increasing β in this case causes the balls for the denser class (class 0) to increase more rapidly than those of the less dense class. Note that the class 1 balls do not increase significantly, due

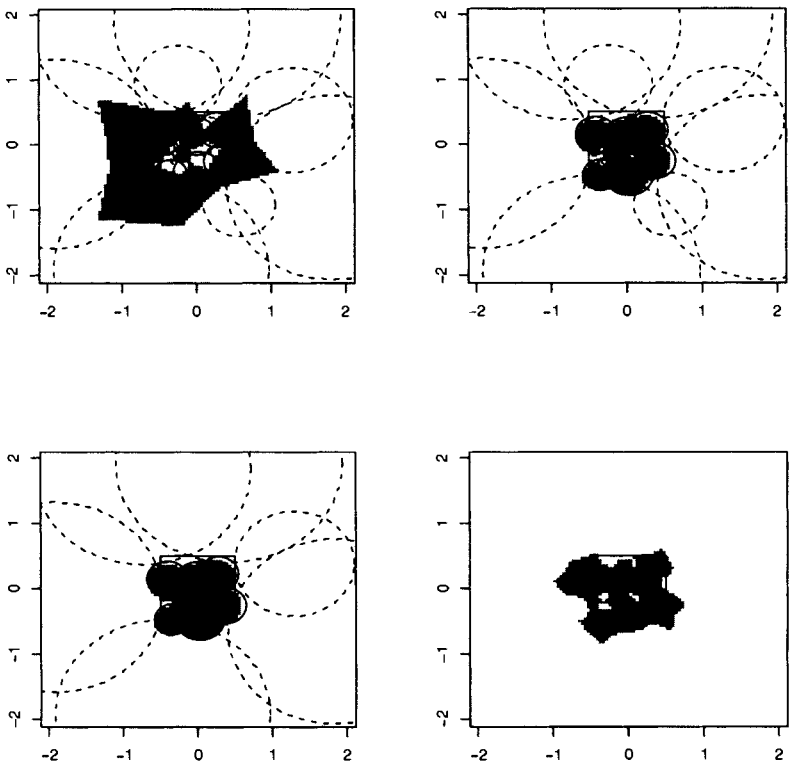


Fig. 4.19 The the decision regions for several classifiers constructed from the data in Figure 4.18. The top left shows the naive CCCD classifier. The top right shows the CCCD classifier, using scaling by radius, $\alpha = 0$ for both classes. In the bottom left, $\alpha_0 = 0, \alpha_1 = 6$. The circles indicate the covers associated with the dominating sets. The bottom right shows the decision region for a 1-nearest-neighbor classifier.

Table 4.3 Results of CCCD classifiers compared with nearest-neighbor classifiers on the artificial nose data.

Classifier	Error
1-NN	0.171
3-NN	0.137
5-NN	0.133
CCCD	0.099
(α, β) CCCD	0.094

to the relative denseness of class 0, as can be seen by comparing the regions in the top right ($\beta_0 = \beta_1 = 4$) with the bottom left ($\beta_0 = 4, \beta_1 = 0$). The bottom right plot ($\beta_0 = 0, \beta_1 = 4$) shows the effect of only increasing the class 1 balls: the class 0 region shrinks, as would be expected.

Let's revisit the nose data of Section 1.4.1. We split the data into two sets of equal size, a training and a test set; in each set there are 380 TCE observations and 176 non-TCE observations. Using the naive classifier of Equation (4.12), without scaling by the radii, we obtain an error of 0.262. Table 4.3 presents the results of several nearest-neighbor classifiers, compared to the CCCD classifier (using scaling by radius). The (α, β) classifier is presented for illustration only. In this, the performance is computed for all choices of αs and βs for the two classifiers. Since this optimization is done using the test set, it is biased, and should not be taken too seriously. The parameters chosen were $\alpha_0 = 5, \beta_0 = 1, \alpha_1 = 0, \beta_1 = 1$.

As can be seen, the CCCD classifier performs quite well on these data. The (α, β) classifier did not do significantly better, and since the error estimate is biased, it would seem that it is not worth considering for these data.

The dominating sets are of size 61 and 84 for TCE and no-TCE, respectively. All chemicals have representatives among both the TCE dominating set and the no-TCE dominating set. The number of exemplars of each chemical are tabulated in Table 4.4. The numbers are fairly similar for both TCE and no-TCE for most chemicals, although one might look into the disparities in ClB and Ker.

Let's look a little deeper into the two dominating sets, considering the subgraphs induced by them. The class 0 set has 27 isolated vertices, three components consisting of two vertices and a single arrow, and a 28 vertex component consisting of 76 arrows. The class 1 set has 67 isolates, 5 components consisting of two vertices and a single arrow, a component of the form $a \leftarrow b \rightarrow c$ and one of the form $a \leftarrow b \rightarrow c \rightarrow d$. The fact that these graphs are so sparse, indicates that the dominating elements are spread out over the support of their respective data, and one might infer that the covers are not overly redundant.

A stereo plot of the dominating sets is presented in Figure 4.21. The coordinates are taken from a three dimensional multidimensional scaling of the original inter-point distance matrix. The majority of the vertices cluster in the clump of (mostly

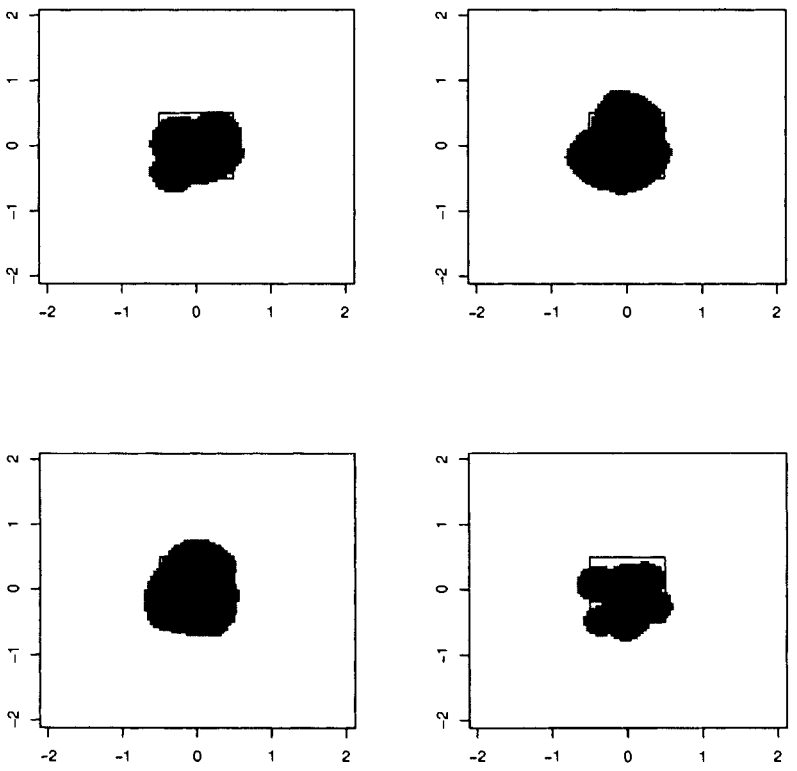


Fig. 4.20 The the decision regions for several classifiers constructed from the data in Figure 4.18. In all plots, $\alpha_0 = 0$ and $\alpha_1 = 6$. The top left shows the CCCD classifier with $\beta_0 = \beta_1 = 2$. The top right shows the CCCD classifier with $\beta_0 = \beta_1 = 4$. In the bottom left, $\beta_0 = 4$ and $\beta_1 = 0$, while in the bottom right $\beta_0 = 0$ and $\beta_1 = 4$.

Table 4.4 Number of exemplars of each chemical within the dominating sets for the nose data.

Chemical	TCE	No-TCE
Air	9	7
Ben	7	13
BTE	6	13
ClB	1	12
Clf	10	10
CTe	5	11
Ker	14	4
Oct	3	7
Wga	5	7
TCE	1	

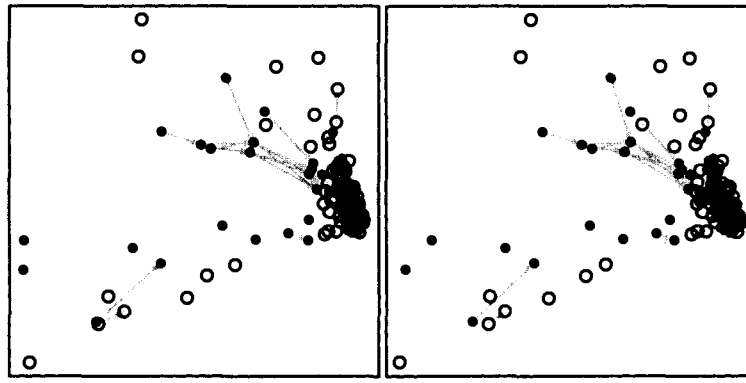


Fig. 4.21 Stereo pairs plot of the dominating sets of the nose data. Class 0 (TCE present) is plotted as solid dots, class 1 (TCE absent) is plotted as circles.

isolated) vertices, indicating that most of the dominating elements are close to the decision boundary and relatively close to each other (at least in this projection). It is clear that the TCE dominating set induces a graph with some structure, indicating some of the structure we elucidated in previous sections (Sections 2.4.2 and 3.3.2, in particular).

A further illustration of the representativeness of the dominating sets is given in Figure 4.22, where histograms of the radii are plotted for the full graphs and the dominating sets. Under this measure, the dominating sets appear quite representative as well. Note that the distances from no-TCE to the boundary are generally less than those from TCE to the boundary. One hypothesis that might explain this is that the

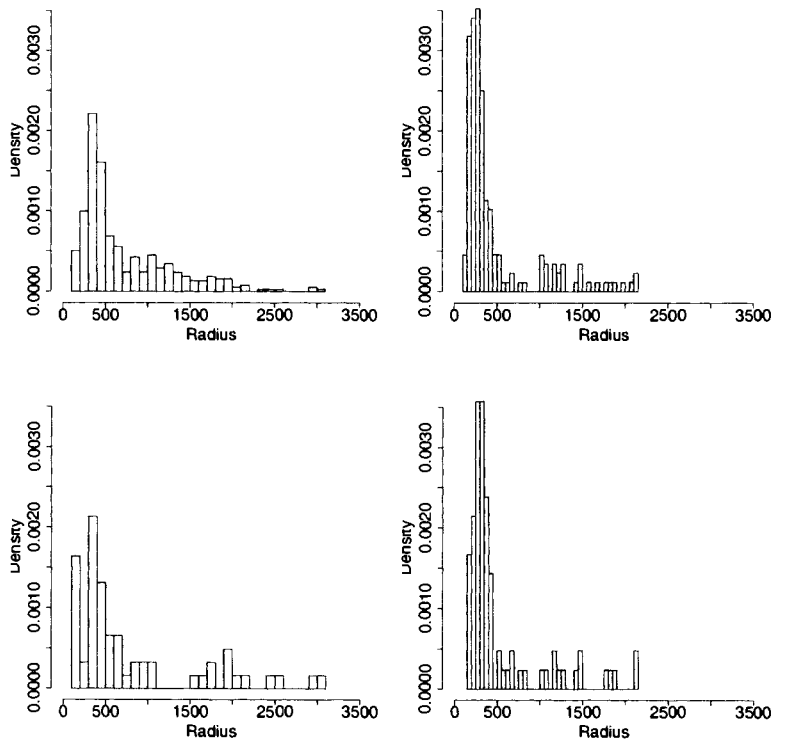


Fig. 4.22 Histograms of the radii for the CCCD (top) and dominating sets (bottom). TCE is on the left, no-TCE on the right.

Table 4.5 Results of CCCD classifiers compared with nearest-neighbor classifiers on the hyperspectral data.

Classifier	Error
1-NN	0.164
3-NN	0.134
5-NN	0.130
CCCD	0.154
(α, β) CCCD	0.134

chemical reaction of TCE with confusers causes TCE to spread out more. We have not investigated this hypothesis further.

The results of a similar experiment on the hyperspectral data (Section 1.4.2) is shown in Table 4.5. Again, the data were split into training and test sets of equal size. The CCCD classifier is comparable to the nearest-neighbor classifiers. However, instead of saving 500 observations, as with the nearest-neighbor classifiers, the CCCD requires 119 observations from the two dominating sets.

As with the nose example, the (α, β) classifier results are biased due to the method of selecting the parameters. In this case the best parameters found are $\alpha_0 = \beta_0 = 2$, $\alpha_1 = 4$ and $\beta_1 = 3$. The fact that this appears to produced a somewhat improved classifier suggests that the flexibility of the (α, β) classifier might be justified in this problem.

The problem of finding a reduced set of points to use as prototypes has a long history in pattern recognition. An early algorithm is the condensed nearest-neighbor classifier of [83]. [42] describe another technique for use with nearest-neighbor classifiers. In some sense, this is one of the purposes of support vector machines; see [208], [182] and [38].

Problems

4.29 Build a CCCD classifier for the mine data of Table 4.2. Use cross-validation to determine your error. Compare this with a nearest-neighbor classifier.

4.9 HOMOGENEOUS CCCDS AND OTHER VARIANTS

Several variations on the basic idea of the CCCD are possible. A simple homogeneous CCCD would result from setting

$$r = \min_{x \in X_0} (d(x, X_1)) \tag{4.16}$$

and letting $r_x = r$ for all $x \in X_0$. This results in a constrained homogeneous class cover that is pure and proper, with radius chosen to be maximal under these conditions.

Variations similar to the (α, β) CCCD can be obtained by replacing the minimum in Equation (4.16) with a β-order statistic.

It is clear that the homogeneous CCCD defined by Equation (4.16) is a subgraph of the standard CCCD. Thus we have $\gamma_{CCCD} \leq \gamma_{\text{homogeneous CCCD}}$. In the case of the nose data (considering only the subset of the data used for training in the experiment described in Section 4.8) we have $\hat{\gamma}_{\text{TCE}} = 61 < \hat{\gamma}_{\text{homogeneous TCE}} = 380$ and $\hat{\gamma}_{\text{no-TCE}} = 84 < \hat{\gamma}_{\text{homogeneous no-TCE}} = 175$, a considerable difference (in fact, the homogeneous dominating sets result in a total reduction of complexity of just 1 vertex). This is one reason that homogeneous class covers are (generally) less desirable than non-homogeneous ones. Note the the scale dimension of Section 4.6 provides a method for producing a compromise.

Another approach to the homogeneous class cover problem, is to take the cover to be the dominating set $\mathcal{S}$ from the non-homogeneous CCCD, with radii

$$r = \min_{x \in \mathcal{S}} r_x. \tag{4.17}$$

This has the advantage that the size of the cover is equal to that of the non-homogeneous one, but the disadvantage that the cover may no longer be proper. By replacing the minimum with a maximum the cover is proper but (almost certainly) not pure. Algorithm 4.4 gives an algorithm for finding a pure proper homogeneous cover. [31] provide another. In Algorithm 4.4 a ball of radius 0 does not cover any vertices, even its center, and so it is not available as a potential dominating set element.

input : A non-homogeneous CCCD
output : A homogeneous dominating set

initialize: $R = \min r_i$, S to be the greedy dominating set for the CCCD with all radii set to R
while *TRUE* **do**
  set $i = \underset{r_i > 0}{\operatorname{argmin}} r_i$;
  if $v_i \in \tilde{N}(V \setminus \{v_i\})$ **then**
    set $r_i = 0$;
    set $R = \min_{r_i > 0} r_i$;
    set S to be the greedy dominating set for the CCCD with all radii set to R;
  end
  else
    return S;
  end
end

Algorithm 4.4: An algorithm for constructing an (approximately minimum) homogeneous cover.

One could also relax the constraint that the cover be centered on X_0 observations. One way to do this would be to construct the (homogeneous or non-homogeneous) constrained CCCD, then treat this as an in initial position for a mixture model.

Problems

4.30 Assume that the intersection of the supports of X_0 and X_1 is non-empty. Let n_0 be fixed. Prove that $\lim_{n_1 \to \infty} \gamma_0 = n_0$ for the homogeneous CCCD. Give an example of distributions where $\gamma_0 \to 1$ for the non-homogeneous CCCD.

4.10 VECTOR QUANTIZATION

Vector quantization seeks to represent a full data set by a small subset of representative points (vectors). These representative points act as prototypes for the data, and may be used in place of the full data set, for example, in a nearest-neighbor classifier. See [70], [36] and [77] for more information. In particular, the latter contains a bibliography consisting of more than 500 references.

Vector quantization is concerned with two issues: how many bits are required to represent the data, and the quantization error introduced. In the case of classification, the error is defined in terms of the classification error; how well does the reduced model classify the observations.

Clearly, the selection of the minimum dominating set withing the CCCD is a type of vector quantization. By restricting ourselves to actual observations within the data, we are (from the perspective of vector quantization) being unnecessarily restrictive. Perhaps, by allowing further flexibility in the cover, we might be able to further reduce the complexity. This leads to the more general version of the class cover problem, in which we do not restrict the centers of the balls.

Vector quantization leads to another use for class cover catch digraphs in pattern recognition, which is summary of the data via a small number of prototypes. We have seen some of this type of analysis in the examples (Section 4.8).

In [163] it is proved that the CCCD classifier is Bayes optimal in the case where the supports of the densities of the classes are δ-separable. Two compact sets are δ-separable if the distance between the sets is at least δ for some $\delta > 0$. Thus, in this case, the quantization produced by the CCCD dominating set has an asymptotically zero error. This is also true for most types of reduced nearest-neighbor classifiers.

4.11 RANDOM WALK VERSION

The fact that the β in the (α, β) classifier of Section 4.7 is global is clearly undesirable. Rather than having every ball cover a few extra observations, we'd like only those balls for which this is advantageous to do so. This section describes a method for implementing this goal.

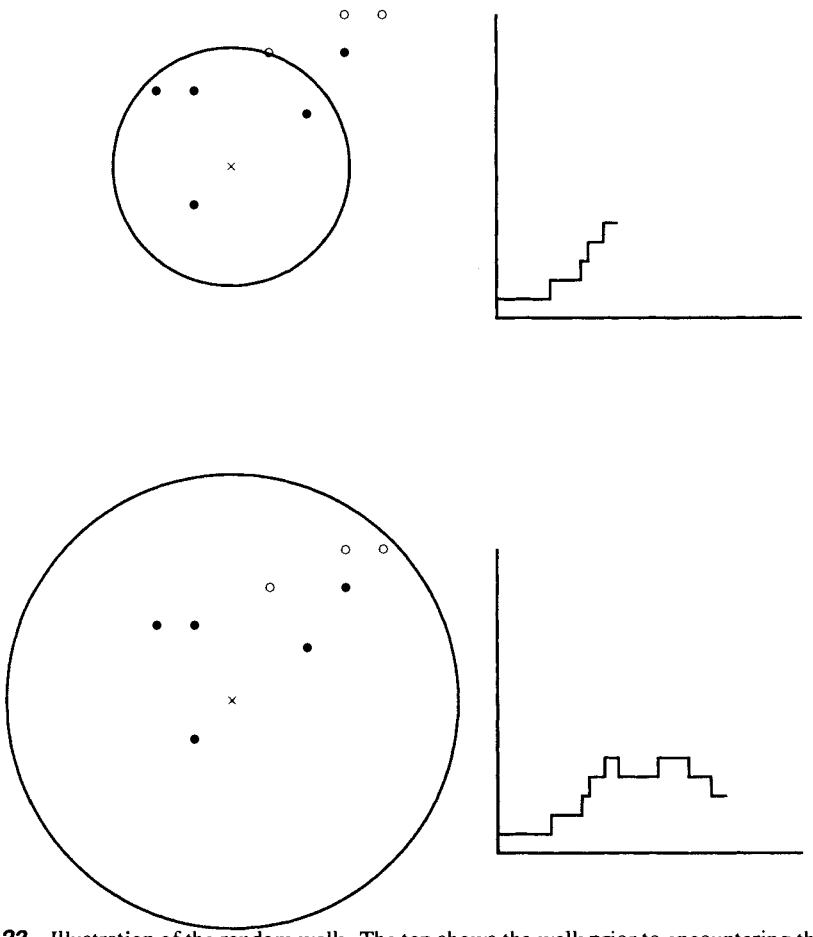


Fig. 4.23 Illustration of the random walk. The top shows the walk prior to encountering the first non-target point. The walk increases by a step at each target observation. In the bottom plot, the random walk has continued, decreasing a step at each non-target observation.

Figure 4.23 illustrates the idea. To determine the radius for an observation, a walk is started whereby a step up is taken each time a target observation is encountered by an expanding sphere centered at the point, and a step down is taken for each non-target point encountered.

Formally, the random walk is defined as

$$R_x(r) = \frac{1}{n}|\{z \in \mathcal{X}_0 : \rho(x,z) \leq r\}| - \frac{1}{n}|\{z \in \mathcal{X}_1 : \rho(z,z) \leq r\}|. \qquad (4.18)$$

Given a random walk, we need a method for selecting the radius. If we choose the point at which the walk first decreases, we have the original algorithm. Our choice is to select the radius to be the point at which the random walk is maximized. Alterna-

tively, one could penalize a walk that decreases "too far". This can be implemented by subtracting a penalty curve from the walk, and choosing the point at which the difference is maximized. We will see a similar approach in Chapter 5.

The walk R_x provides both a method of selecting the radius and of scoring the "suitability" of a ball for incorporation in the classifier. Larger values of $R_x(r_x)$ indicate "better" balls, in the sense that they tend to be purer and cover more than ones with lower values.

The dominating set algorithm used for the random walk version of the CCCD is to repeatedly select the ball with the largest score. We wish to penalize overly large balls in favor of smaller ones, and so we use the score:

$$T_x = R_x(r_x) - \frac{n_u r}{2d_m}, \tag{4.19}$$

where n_u is the number of points not covered by the current cover and $d_m = \max\{\rho(X_0, X_1)\}$. This is made explicit in Algorithm 4.5.

input : A CCCD along with the random walks that defined the radii
output : A dominating set S

initialize: $\mathcal{J} = V, \mathcal{S} = \emptyset, n_u = |V|$
while $\mathcal{J} \neq \emptyset$ **do**
  Compute the scores T_x using Equation (4.19);
  $v = \underset{v' \in V}{\operatorname{argmax}} T_{v'}$;
  $\mathcal{S} = \mathcal{S} \cup \{v\}$;
  $\mathcal{J} = \mathcal{J} \setminus \bar{N}(\mathcal{S})$;
  $n_u = |\mathcal{J}|$;
end

Algorithm 4.5: Random Walk dominating set algorithm

The scores T_x can also be used in the classifier. Instead of using the scaled distance of Equation (4.13), we us the slightly modified

$$d(z, B(x, r_x)) = \left(\frac{\rho(z, x)}{r_x}\right)^{T_x^\epsilon}, \tag{4.20}$$

where $\epsilon \in [0, 1]$ is a control parameter determining how closely the classification regions follow the boundary of high scoring balls.

Problems

4.31 Implement the random walk CCCD and test the resulting classifier on the mine data of Table 4.2. Compare your results with those obtained in Problem 4.29.

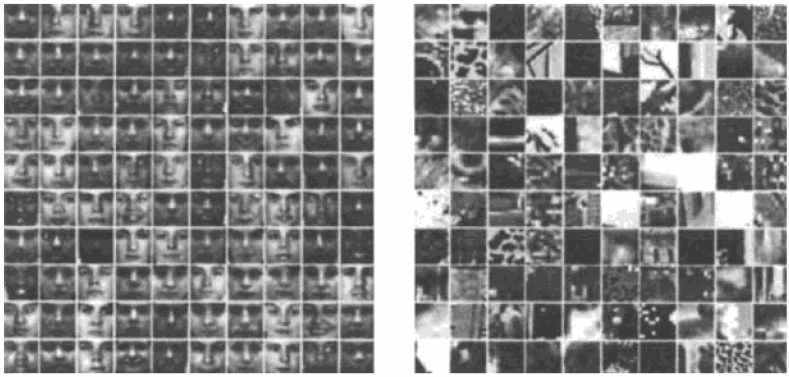


Fig. 4.24 Example training data for the face detection algorithm. Face chips are on the left, non-face on the right.

4.11.1 Application: Face Detection

An area of great interest in image processing is the automatic detection and recognition of faces in digital images. See [115] for a survey of techniques. Here we will discuss a method of [193], which uses class cover catch digraphs. The idea is similar in flavor to that of [58], in the sense that the algorithm allocates most of its processing power to those regions which are "most face-like".

The task is to find faces in an image, without restriction on the number or positions of faces that may be in the image, or the scale (within reasonable limits). In this work the faces are assumed to be in standard position, that is facing the camera without significant rotation or tilting. Figure 4.24 shows examples of training data from the two classes, face and non-face. Approximately 2,000 face observations were used, each one cropped and scaled to a 21×21 pixel image chip, encompassing the part of the face from the just above the eyes to just below the mouth vertically, and the two cheeks horizontally. Over a million non-face chips were collected from a large collection of images. The images in this section are courtesy of Diego Socolinsky.

In some ways this is a one-class classification problem; we wish to classify an observation (a window in an image) as either face or any of a huge class of other objects (non-face).

The CCCD was modified in some important ways. Since the number of face observations is much smaller than non-face, we wish to be able to quickly eliminate from consideration the vast majority of non-target observations. This is accomplished through a tree structure, which implements a kind of boosted classifier (see [61] and [85] for more information on boosting). At each stage of the tree a small number of balls (usually one or two) are selected from each class, and the radii of the balls are increased or decreased in order to ensure that a given (empirical) misclassification error is obtained on the face class. The tree then is built by repeating the algorithm on the misclassified observations.

The algorithm has two stages; it selects a (small) set of faces, and repeatedly selects non-face exemplars to compare against the face exemplars. Once this fails to provide sufficient performance, a new set of faces is selected, and the procedure continues.

The resulting tree classifier is illustrated in Figure 4.25. Note that in this (pedagogical) example we have chosen to use two faces and two non-faces at each stage, and that the faces are retained for several (in this case, two) stages.

At each stage, a novel observation (a window on the image) is compared to the exemplars using the CCCD classifier. If it is deemed non-face (goes down the right branch) no further processing is done, and it is classed as non-face. Otherwise, it continues down the tree on the left, and the comparison repeats with the new exemplars. Only if the observation makes it all the way through the tree does it get classed as face. Thus, most of the processing is done only for those windows which are most "face-like". Multiple windows, of different sizes, are used, rescaled to 21×21 to account for differences in scale of the faces. In principle, resistance to changes in orientation could be implemented through a combination of enhanced training data and some affine transformations of the windows as well.

Figure 4.26 shows the processing time, indicating that the algorithm does focus on the regions containing "face-like" structure. The background of this image is relatively benign, but this type of result is typical for this algorithm.

Multiple detections are made for every face, since slight translations of a window on a face also appear quite face-like. These multiple detections are merged into a single one in a fairly standard way. See [193], and also [14]. The results are illustrated in Figure 4.27. (The images used in these tests are from the CMU/MIT test set.)

The results are quite promising. One example is given in Figure 4.28, where all but one of the faces is detected, with one false alarm. (It is not clear whether this error is one of the detection algorithm or the merging algorithm.) The algorithm can run up to 12 frames a second on a PC using a 320×240 pixel camera. Classification errors are not reported, as the work is still in-progress. However, the algorithm does seem to find most of the the faces in the test images, with very few false alarms. Further work is ongoing.

4.12 FURTHER READING

The Reduced Coulomb Energy (RCE) neural network was first described in [176], and further discussed in [175]. The technique is framed in terms of a neural network architecture, but it can be understood in terms of the class cover problem. Essentially, balls are placed at observations, with fixed radii. As new observations are presented, the radii are modified to ensure that no balls from other classes contain the observation. If no balls cover the new observation, a new ball is initialized.

This approach is also reminiscent of the adaptive mixtures algorithm of [160, 161, 157]. The idea here is to fit a normal mixture to each class, in which new components are added to the mixture when observations are presented that are not sufficiently covered by existing components. A separate mixture is computed for each class, and the likelihood ratio is used to classify new observations.

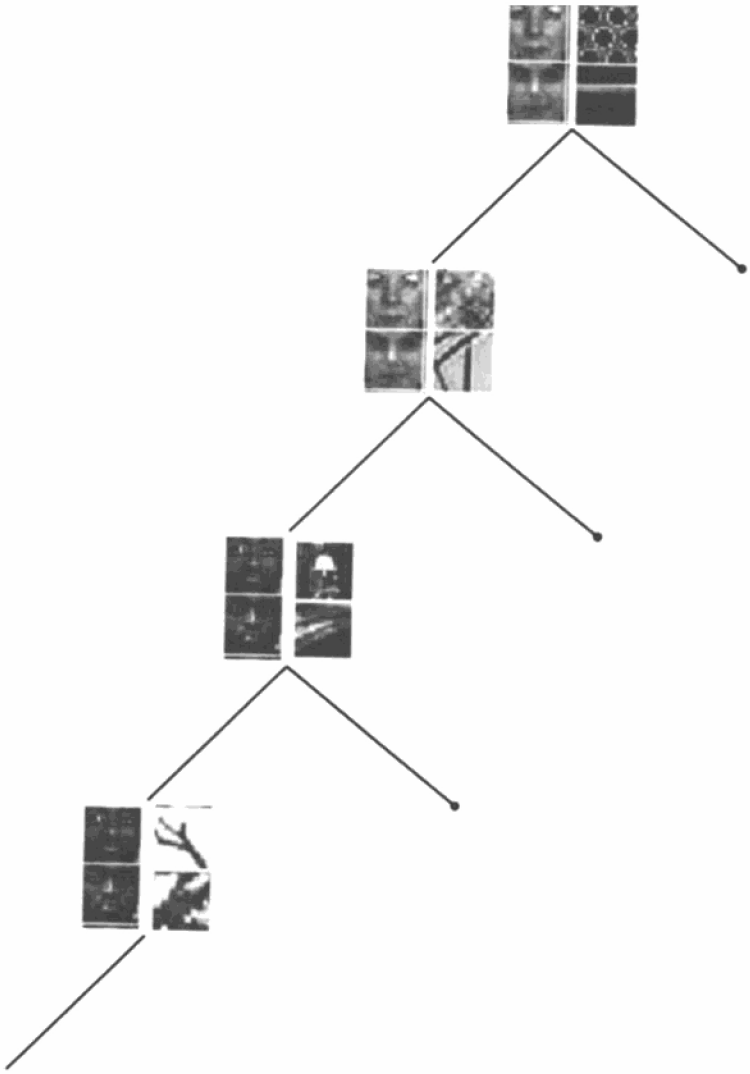


Fig. 4.25 An illustration of the boosted CCCD face classification tree. The exemplars at each stage (two faces, two non-faces in this example) are presented in the boxes.

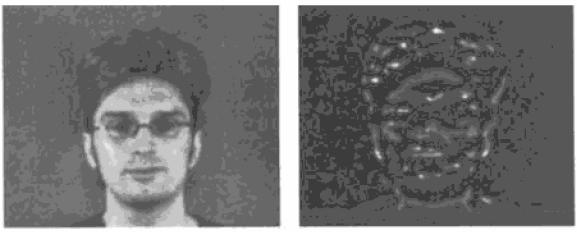


Fig. 4.26 An example illustrating the processing time spent within a window. The right-hand image shows the processing time as a gray-scale image, with white indicating the longest time to process the associated window.

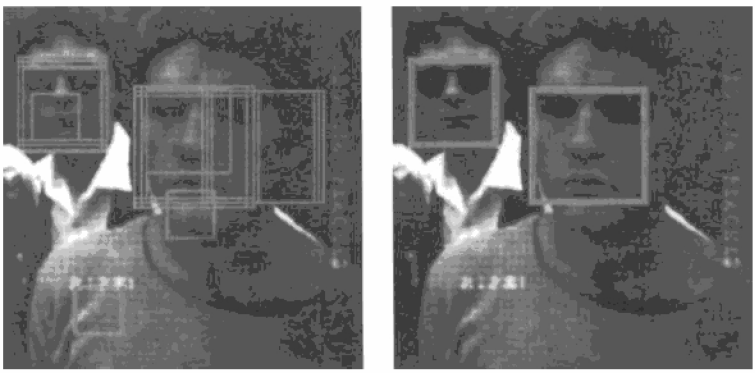


Fig. 4.27 The original detections and the resultant merged detections for an example image.

There are many other approaches that have a similar flavor (radial basis functions, for example, see [86], [85], and [22], as well as several the fall into the general category of neural networks, see [177]). Few if any of these take a graph-theoretic approach to the fitting of the model, and most use ad hoc methods for determining the ultimate complexity of the classifier.

The classifier resulting from the RCE model is of the form

$$g(z) = \begin{cases} 0 & \text{if} \quad z \in \cup_{x \in X} B_x \cup (\cup_{y \in Y} B_y)^c \\ 1 & \text{if} \quad z \in \cup_{y \in Y} B_y \cup (\cup_{x \in X} B_x)^c \\ -1 & \text{otherwise.} \end{cases} \tag{4.21}$$

Here B_x is the largest ball centered at x containing no points from the other class. See also [60] for a similar approach.

Reference [100] discusses using proximity graphs to segment images according to their local texture characteristics. The idea is to set regions (or pixels) as the vertices, and place an edge between two such if their distance (in feature or gray-scale space) is small. Manipulation of the resulting graph (for example, graph components) results in segmentation of the image.

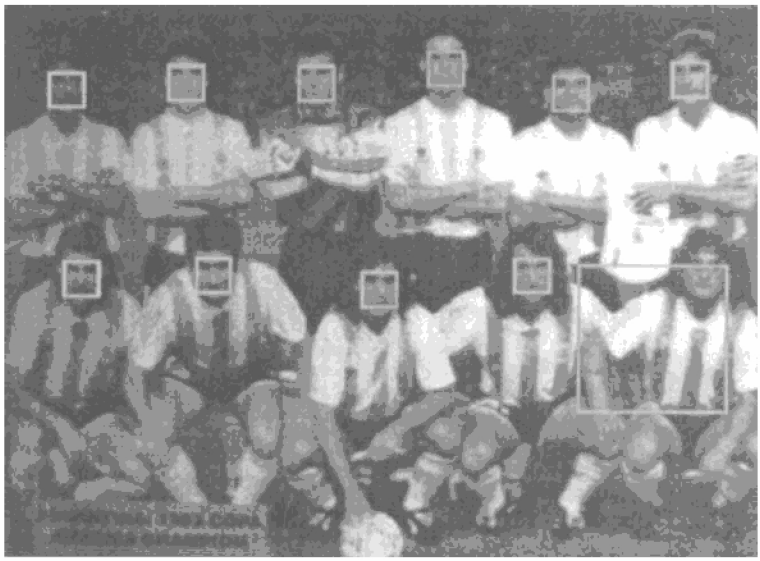


Fig. 4.28 An example of the face detection algorithm, with a missed detection and a false alarm. The detections are indicated by the boxes.

There are several discussions of catch digraphs from a graph theoretic perspective. See [167], [168] and [131].

The papers [163] and [44] provide several variations on the CCCD for classification. A different kind of proximity region is described in [32], for which calculations similar to those in Section 4.4 are possible in higher dimensions.

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

5

Cluster Catch Digraphs

The strategy of cluster analysis is structure seeking although its operation is structure imposing.

Aldenderfer and Blashfield, *Cluster Analysis.*

The class cover catch digraph (Chapter 4) seems to be fundamentally tied to classification problems. The question arose as to whether it could be modified for use with clustering. In clustering, there is no "other class" to be used to define the radii of the balls, which is the fundamental requirement for defining the graph. Therefore, we need a method to define the radii in the absence of distinct known classes. This chapter will define and investigate one solution to this problem.

5.1 BASIC DEFINITIONS

Instead of using a second class to define the radius of the ball at an observation, the cluster catch digraph uses a measure of "clustered-ness" as defined by a test against a hypothesis of "no clustering".

Figure 5.1 shows the basic idea (see also Section 4.11). We consider the problem of defining the radius for a single vertex, denoted by the circled "x" in the figure. Define a random walk on the plane by considering a growing circle centered at the observation. Every time a new observation intersects the circle, the walk moves up one unit. Otherwise, it continues horizontally at the same rate as the radius of the circle. This results in a walk with vertical steps for each observation, at the distance from the central observation. If the upward step is of height $1/n$, then this corresponds to the empirical distribution of the distance from the central point. This

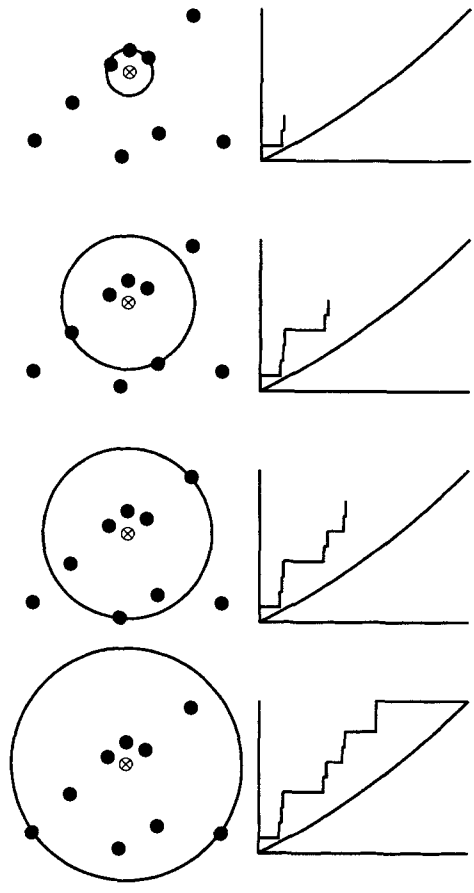


Fig. 5.1 Random walk defining the calculation of the radius for the cluster catch digraph.

is the key observation. We define the radius in terms of a hypothesis test, comparing this empirical distribution function with a "no clustering" null hypothesis. This is a sequential test, defining the radius in terms of the first time the test rejects the "no clustering" null hypothesis.

If we denote the empirical distribution function defined by the random walk as $RW(r)$, and the null distribution as F_0, then the non-sequential Kolmogorov-Smirnov (K-S) test would suggest we should choose the radius to be

$$r^* = \underset{r \geq 0}{\operatorname{argmax}} \{RW(r) - F_0(r)\}. \tag{5.1}$$

This selects the radius for which the data in the ball are farthest from the null hypothesis of complete spatial randomness. This deviation from "non-clustering" is required to be in the positive, or "more clustered" sense. Thus, we pick the radius so that the

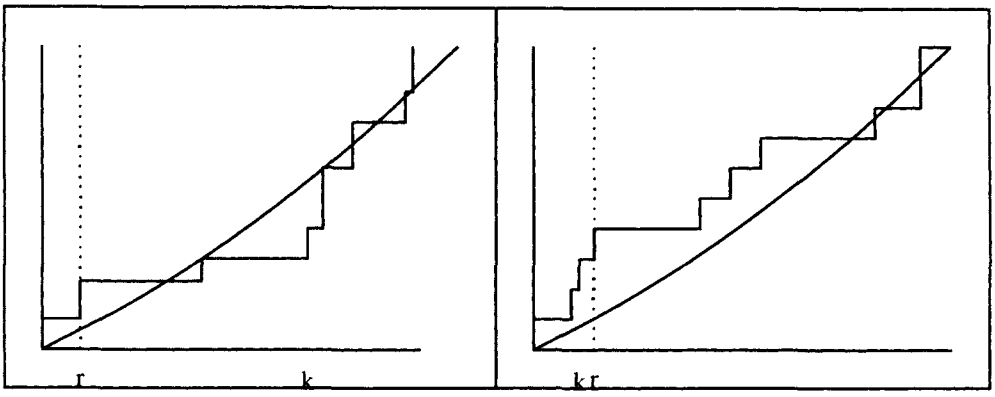


Fig. 5.2 Radius determination in the random walk. k indicates the first time the K-S test rejects the null hypothesis (indicated by the curve). In the left figure, the test rejects as being less clustered than the null, and so the radius is chosen to be the value at which the test was largest prior to the rejection. In the right, the rejection indicates the data are more clustered than the null, and so the radius is increased so long as the test statistic does not decrease.

corresponding ball covers the most clustered group of observations possible, by the definition from the K-S test.

For the sequential version, define

$$\begin{aligned} l(r) &= \min_{q \le r}\{RW(q) - F_0(q)\} \\ u(r) &= \max_{q \le r}\{RW(q) - F_0(q)\}. \end{aligned}$$

Let k_α be the value of the K-S statistic at which one would reject with level α. Let R be the first radius such that either $-l(R) \ge k_\alpha$ or $u(R) \ge k_\alpha$. Set R to be the first point at which rejection occurs due to $l(R)$ (if no such rejection occurs, then set R to be the maximum possible). Then set

$$r_x = \operatorname*{argmax}_{q<R}\{RW(q) - F_0(q)\}.$$

The K-S statistic associated with x would then be

$$k_x = RW(r_x) - F_0(r_x).$$

See Figure 5.2 for an illustration. Compare this with the random walk CCCD of Section 4.11. Throughout this chapter we will use the value of $\alpha = 0.05$ without further comment.

Other rules are possible. For example, if the first rejection is for $u(R)$, one could continue the walk until the first point at which the statistic starts to decrease. Alternatively, take the maximum between the first point at which $u(R)$ rejects and the next place at which it fails to reject. We will not consider these variations further.

We now need to consider the definition of the null distribution. If we define "not clustered" as complete (local) spatial randomness, then the appropriate distribution is the spatial Poisson, or uniform, distribution. In the case of a d-dimensional uniform distribution defined on a d-dimensional ball, the empirical distribution function, as a function of the distance from the central point, is proportional to r^d. This suggests a null distribution of the form mr^d.

Other null distributions are possible, leading to different digraphs, but we will not pursue these here. Note that in particular, since we are only considering inter-point distances, we do not truly have a null hypothesis of complete spatial randomness, but rather of isometry.

The value of d is generally set by the data, although we will discuss some variation of this in Section 5.4. The slope of the distribution, m, corresponds to the intensity of the Poisson distribution.

Consider Figures 5.3 and 5.4. Here 1000 observations were drawn from a two dimensional spherical normal distribution. In Figure 5.3 the data are depicted with three representative balls computed using Equation (5.1). As can be seen, the observation near the mean does a good job of covering most of the data. The observation half way out from the mean has a larger radius, indicating that the data continue to be more clustered as we move toward the mean, and similarly the observation farthest from the mean has the largest ball of these three.

From the walks in Figure 5.4 we can see that the third walk dips below the null hypothesis. With an alpha level of $\alpha = 0.05$ this dip is significant, and so using the sequential version this ball would have a much smaller radius (in fact, it would only go out to the point nearest the center). The statistics computed for these three balls ($\max_{r \geq 0}\{RW(r) - F_0(r)\}$) are (in order of distance from the mean) 0.527, 0.531 and 0.350. Using the sequential algorithm, the third is reduced to 0.001.

Typical results can be illustrated through simulations. The radii calculated by the sequential method are depicted in Figure 5.5. For this figure, 100 samples of 1000 observations from a spherical normal distribution were used to generate the picture. As can be seen, the average radius at the mean of the distribution is two standard deviations. The radius grows as one moves out from the mean, until one reaches the range at which the normal density drops below that of the uniform, at which point the radius shrinks to the distance to the nearest observation. The average radii ranged between 0.83 and 4.34 in this simulation.

Figure 5.6 shows the statistics computed in the simulations of Figure 5.5. This is a section of the sample along the horizontal line through the origin. As can be seen, the maximum is found near the center, and the statistic falls off fairly rapidly to zero by four standard deviations from the mean.

Problems

5.1 For the K-S test, one needs only check the values of the statistic at the observations. Is this true for the sequential radius selection algorithm?

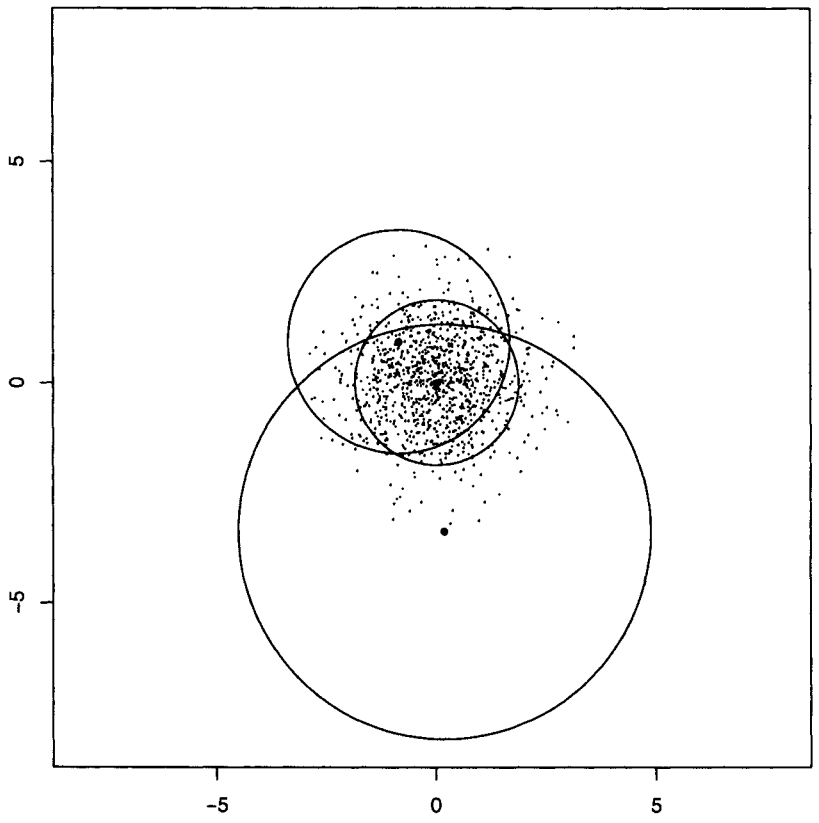


Fig. 5.3 Uncorrelated normally distributed data. Three example balls are shown, for the observation nearest the mean, half way out, and farthest from the mean (indicated by the solid dots). The radius calculation uses the rule of Equation (5.1).

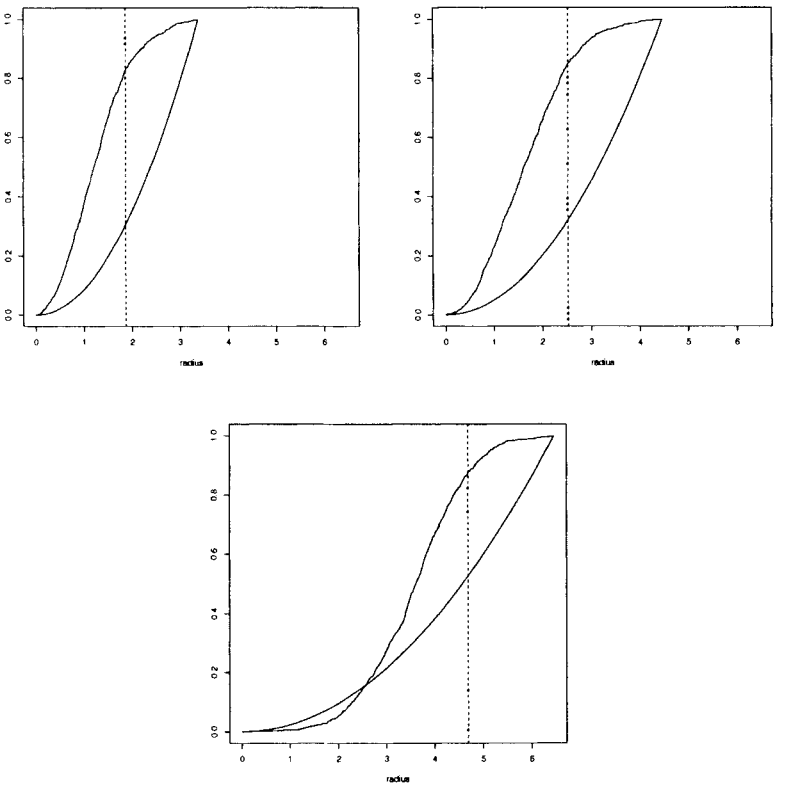


Fig. 5.4 The random walks for the balls from Figure 5.3. The radius is indicated by a dotted vertical line, and the x-axis has been scaled consistently for the three graphs.

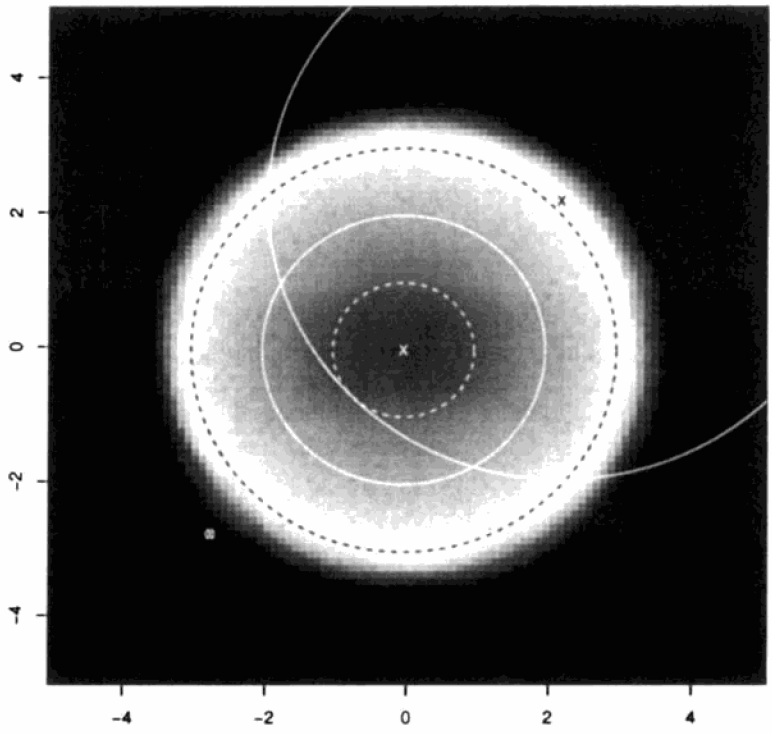


Fig. 5.5 A data image of the radius for uncorrelated normal data, using the sequential method. Radius is indicated by gray level, with black indicating small radii and white large. The solid circles depict the radii for three different points (indicated by the x's). The dotted circles indicate a distance of 1 and 3 standard deviations from the mean. (Circle and "x" color is chosen only for ease of display.)

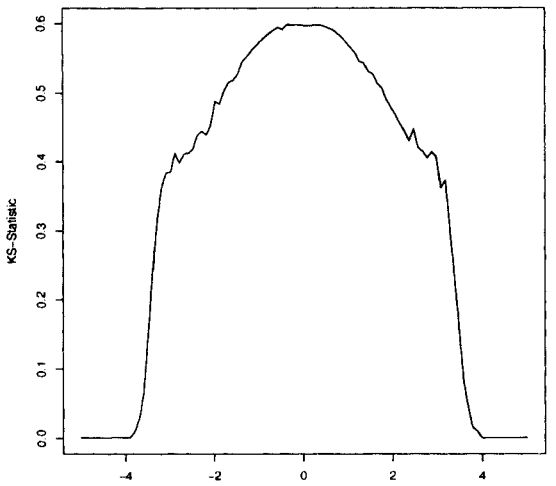


Fig. 5.6 The (average) statistics computed for the simulations of Figure 5.5.

5.2 Consider a single observation x at the origin, and k observations on a sphere of radius R centered at the origin. Fix m and α, and investigate the properties of r_x as R increases. Note that the answer is independent of the distribution of the points on the sphere.

5.2 DOMINATING SETS

One way to select the clusters from the catch digraph is by computing a dominating set of the digraph. As we have seen, there are often several minimum dominating sets, and we wish to choose ones that have properties that make them good candidates for cluster centers. We may wish to trade off the cardinality with clustering performance. For example, we may choose an independent dominating set, one for which no edges exist between dominating set elements.

This approach does not guarantee a single observation for each cluster. In particular, non-spherical clusters, and those with tails of smaller-than-uniform intensity will tend to have more than one observation per cluster. Thus, the dominating set technique is more properly considered a vector quantization ([70], [77]; see also Section 4.10) than a clustering, and further processing is required before clusters can be assigned.

The greedy algorithm presented in Section 4.3 does a good job of finding approximately minimum cardinality dominating sets. It uses out-degree, or number of points covered, as the sole criterion for selecting the vertices in the set. While this is a

reasonable criterion, we have more information in the case of the CCD, which we may want to take into account. For example, each vertex has both a radius and a K-S statistic associated with it. Thus, we can modify the greedy algorithm in several ways. First, we can use either the K-S statistic or local density (number of points covered divided by an appropriate function of the radius) to break ties in the greedy algorithm. This has the advantage that the algorithm retains all the same properties of the standard greedy algorithm. Algorithm 5.1 provides one version of such an algorithm.

input : A CCD on $\mathcal{X}$, with vertex set $V = \mathcal{X}$ and K-S statistics from the random walk $\mathcal{K}[\{k_1, \ldots, k_n\}$

output : A dominating set S

initialize: $\mathcal{J} = V, \mathcal{S} = \emptyset$

while $\mathcal{J} \neq \emptyset$ **do**

$z = \max_{\{v' \in \mathcal{J}\}} \{d_{out}(v')\}$;

$v = \operatorname{argmax}_{\{v_j \in \mathcal{J}: d_{out}(v_j) = z\}} \{k_j\}$;

$\mathcal{S} = \mathcal{S} \cup \{v\}$;

remove from E all edges incident on v;

$\mathcal{J} = \mathcal{J} \setminus \bar{N}(\mathcal{S})$;

end

Algorithm 5.1: Greedy dominating set algorithm with ties broken by the K-S statistic

Alternatively, we can modify the algorithm as in Algorithm 5.2. In words, the algorithm selects the vertex with the largest K-S statistic. This is the "most clustered" observation, under our chosen metric. The observations covered by the vertex are then removed from the statistics of all remaining vertices, and the algorithm proceeds until all observations have been covered. The K-S recomputation involves adjusting the random walk for each vertex x as if the covered observations were not there. The new K-S statistic is the difference between this new random walk and the null distribution. Note that the radii are kept fixed throughout; all that needs to be recomputed is the value of the random walk at the fixed radius.

This is illustrated in Figure 5.7. The solid curve indicates the original random walk, with r chosen as indicated. The observations labeled "x" are removed from the computation, and the new random walk is indicated by the dotted curve. As can be seen, this amounts to simply removing one step from the height for each observation removed.

Another variation, which we will not discuss here involves recomputing the radii as well as the statistic. See Problem 5.3 for more information.

The two algorithms are illustrated in Figure 5.8. The left-hand column uses Algorithm 5.1, the right uses Algorithm 5.2. As can be seen, for $m = 1$, where the null hypothesis of no clustering corresponds to observations uniformly in the square,

input : A CCD on $\mathcal{X}$, with vertex set $V = \mathcal{X}$ and random walk $\mathcal{K}$
output : A dominating set for the CCD

initialize: $\mathcal{J} = V, \mathcal{S} = \emptyset, \mathcal{R}' = \mathcal{R}$
while $\mathcal{J} \neq \emptyset$ **do**
  $i = \text{argmax}_j\{k_j \in \mathcal{K}\}$;
  $\mathcal{S} = \mathcal{S} \cup \{v_i\}$;
  $\mathcal{J} = \mathcal{J} \setminus \bar{N}(\mathcal{S})$;
  recompute $\mathcal{K}$ with observations in $\mathcal{J}$ removed.
end

Algorithm 5.2: K-S greedy algorithm

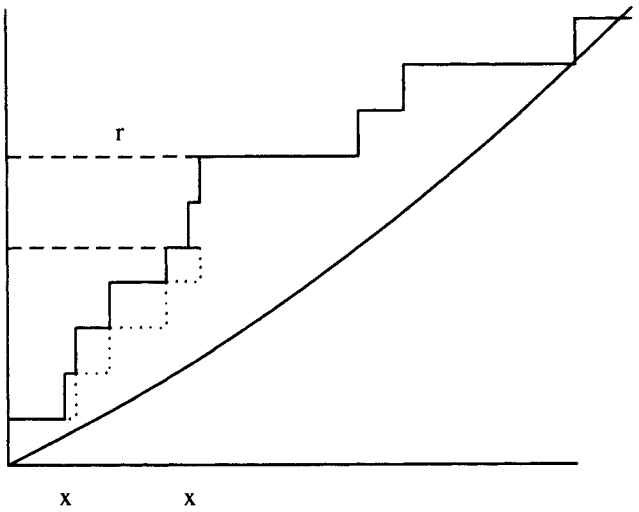


Fig. 5.7 Illustration of the recomputation of the K-S statistic. The observations labeled "x" are removed from the walk, resulting in the random walk indicated by the dotted curve. The dashed lines indicate the difference between the original K-S statistic and the recomputed K-S statistic.

the two algorithms perform comparably, finding three of the four clusters. There is still work to be done deciding how the data are to be assigned to the three clusters, however. For $m = 2$, corresponding to a null hypothesis of higher density, we see that Algorithm 5.1 does a very good job of finding the four clusters. Algorithm 5.2 also finds the clusters, but also identifies three outliers, and leaves some ambiguity as to the number and constitution of the clusters. Clearly, in this one example, Algorithm 5.1 is preferable.

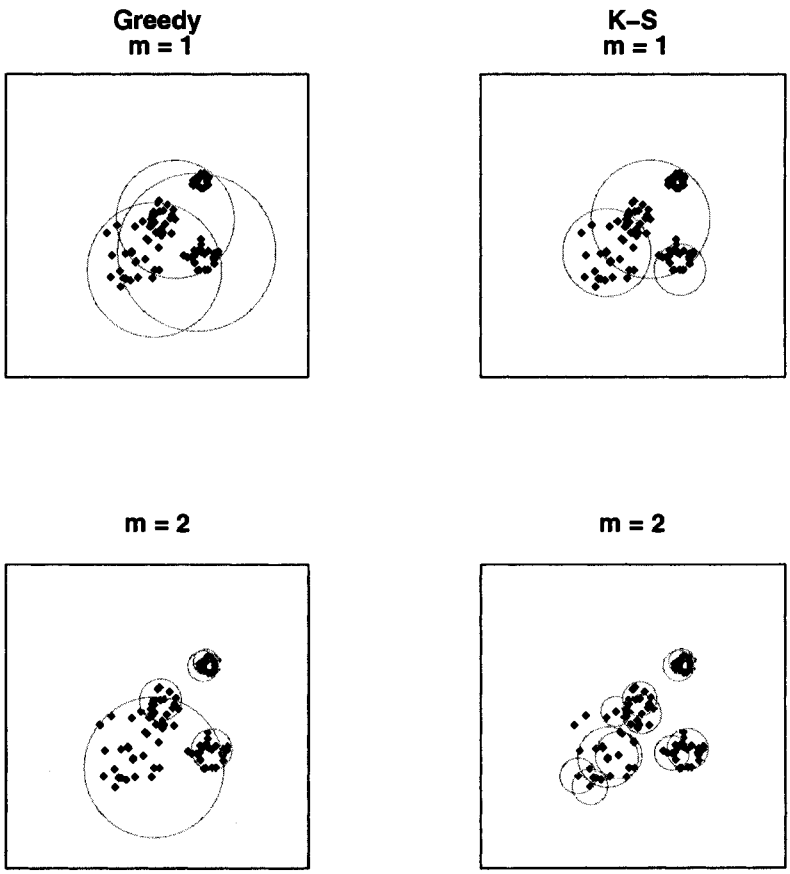


Fig. 5.8 A data set consisting of four clusters, with the results of Algorithms 5.1 (Greedy) and 5.2 (K-S) displayed using the circles corresponding to the dominating cover. The results for two different values of m are shown.

To investigate the relationship between the value of the multiplier m and the radius, consider Figure 5.9. The data consists of 100 observations chosen uniformly from each of two disks, $B((0,0),1)$ and $B(1.4,1.4,1)$. Consider the point at $(0,0)$, central to the first disk. We vary m between 0.1 and 5, and observe that for small values of m this central ball covers all or nearly all of the data; for intermediate values it covers just the one disk, and for large values it covers only the small sub-cluster in the center.

Problems

5.3 Modify the dominating set algorithm (Algorithm 5.2) as follows: after each observation is added to the dominating set, recompute the radii for the remaining uncovered observations as if all currently covered observations were removed. Investigate the properties of this, as compared with Algorithms 5.1 and 2. Does this algorithm produce a dominating set for the original digraph? How does it compare to the other approaches as a method for cluster discovery?

5.3 CONNECTED COMPONENTS

Given the dominating set, as computed via one of the algorithms of the previous section, one needs to process the set to determine the clusters in the data. As can be seen in Figure 5.8, it may not be sufficient to assign a cluster to each element of the set. Some way must be found to group elements together.

We saw in Section 4.6 one way to do a grouping: we could cluster the dominating set elements by radius or K-S statistic. This is not really what we want to do, however. Another idea would be to treat the elements as observations in $\mathbb{R}^q$ and cluster them using some standard clustering technique. This begs the question, and so we look to other approaches in this section.

An obvious grouping is via the components of the induced sub-digraph. A variation on this is to compute the intersection graph of the cover, and compute the components of this. Figure 5.10 shows the catch digraphs and intersection graph for the dominating sets found using Algorithms 5.1 and 5.2. Note that the intersection graph for Algorithm 5.1 is just the digraph with the arcs turned into edges, and so this graph is not drawn in the figure.

As we have seen, the cluster catch digraph depends on our assumption about the intensity of the data, as expressed by the parameter m. Figure 5.11 illustrates how this dependence is reflected in the number of components of the induced catch digraph and intersection graph for the two dominating set algorithms.

Problems

Investigate the properties of using catch digraphs on the results from Algorithm 5.2 for clustering, as discussed in this section through simulation.

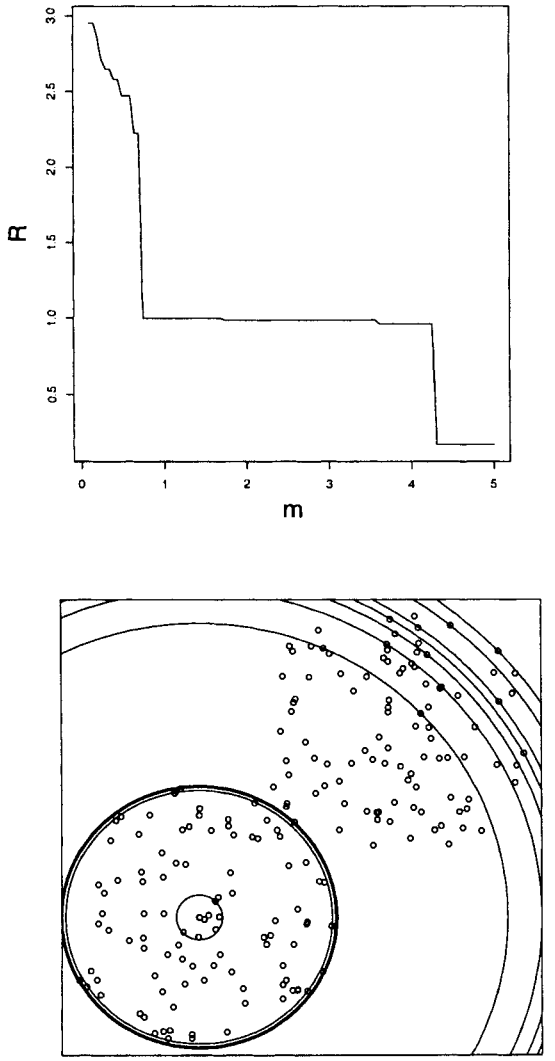


Fig. 5.9 The radius of the central point in the lower left disk as a function of m. The bottom plot shows the data and the circles of the different radii.

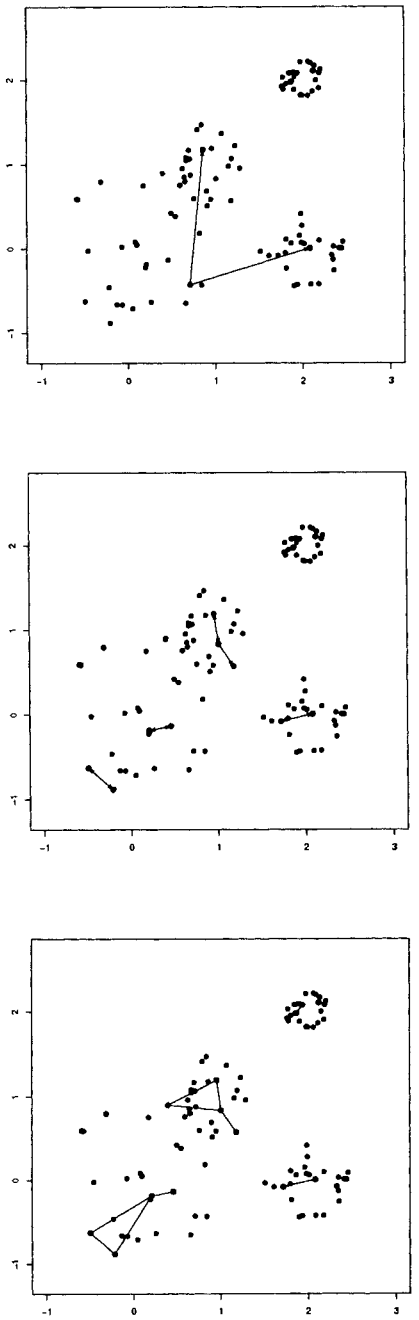


Fig. 5.10 The catch digraphs for the dominating sets from Algorithms 5.1 (top) and 2 (middle) and the intersection graph for the dominating set of Algorithm 5.2 (bottom).

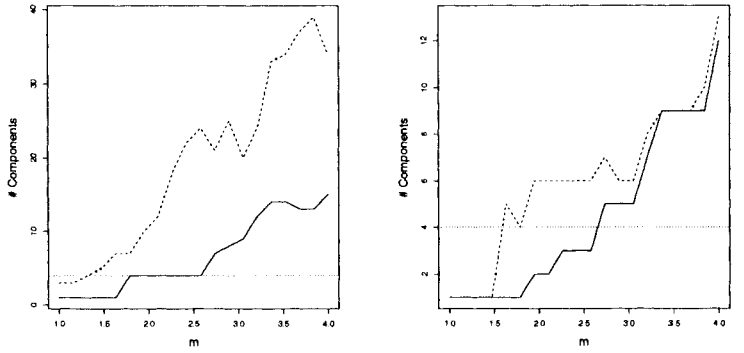


Fig. 5.11 Size of dominating set (left) and number of components in the intersection graph (right) as a function of the parameter m. The true number of clusters is indicated by the dotted horizontal line. The solid curve corresponds to dominating set Algorithm 5.1; the dotted curve corresponds to Algorithm 5.2.

5.4 VARIABLE METRIC CLUSTERING

As mentioned above, the distance metric is critical to the performance of any clustering technique. The local dimensionality of the data is also critical, in the selection of the null hypothesis of the cluster catch digraph. For example, although the data may be d dimensional, it may all lie on a $d' < d$ dimensional subspace, in which case the null hypothesis should reflect this lower dimension. Since the radius calculation is inherently a local calculation, it seems natural to use a local distance in the calculation. In this section we propose a local distance metric that simultaneously selects a data driven metric and a local dimensionality to be used in the radius calculation.

Let K be given. Set x to be the central observation, for which the radius is to be calculated. Let $X = \{x, x_{(1)}, \ldots, x_{(K)}\}$ be the data matrix defined by the $K + 1$ nearest neighbors to x. Let M be the $d \times d$ covariance matrix of X, and S be the singular values of M, with associated eigenvectors U. For fixed $\epsilon > 0$, set the ϵ-rank, denoted d_ϵ, of M to be the number of eigenvalues of M greater than ϵ. Set U_ϵ to be the eigenvectors associated with the eigenvalues greater than ϵ. Then we define the local distance

$$\rho_{K,\epsilon}(x, x_j) = d(U_\epsilon^T x, U_\epsilon^T x_j), \tag{5.2}$$

where d is the Euclidean metric. This implements a d_ϵ-dimensional Mahalanobis distance between observations. Using $\rho_{K,\epsilon}$ as the distance and d_ϵ for the dimension of the null hypothesis, allows for elliptical clusters lying on low dimensional subspaces of the original space. It is important to note that the clusters need not have the same dimension.

Since the vertices now have a dimension associated with them, we can modify the algorithms discussed above by considering digraphs consisting of same-dimensional points. This assumes that one should not cluster observations together if their local

neighborhood does not lie in the same dimensional Euclidean (or other metric or dissimilarity) space.

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

6

Computational Methods

If he once again pushes up his sleeves in order to compute for 3 days and 3 nights in a row, he will spend a quarter of an hour before to think which principles of computation shall be most appropriate.

Voltaire, *Diatribe du docteur Akakia*

6.1 INTRODUCTION

Large and/or high-dimensional data cause considerable trouble for algorithms, particularly those that are of order n^2 in complexity. Ed Wegman (see

www.galaxy.gmu.edu/stats/syllabi/ENAR.html

for a presentation of this work) has provided a taxonomy of data set sizes, expanded from one by Peter Huber, with calculations on the time for analysis of these data sets under different assumptions on the order of the algorithm. The news is not good. We reproduce a subset of these in Tables 6.1 and 6.2. The examples are given to illustrate the types of data that one might consider from the different sizes, but should not be taken too seriously, since data volumes tend to grow exponentially, as new methods for collection come online.

The timings in Table 6.2 are also approximate, and for comparison only. They do not include the time to access the data, which can be considerable for the larger data sizes. Note that the universe is currently on the order of 10^{10} years old.

Table 6.1 Huber/Wegman taxonomy of data set sizes, with examples of data sets of the different sizes.

Descriptor	Bytes	Example
tiny	10	Data set in a statistics book
small	10^4	Medical experiment
medium	10^6	One 1024×1024 image
large	10^8	Moderate sized image database
huge	10^{10}	One day of Internet packet data from a large site
massive	10^{12}	One day from the Earth Observer Satellite
supermassive	10^{15}	Government databases

Table 6.2 The computational complexity of the Huber/Wegman taxonomy of data set sizes. The entries are the time to perform a floating-point operation on all the bytes in the data set (ignoring access time to retrieve the data) using a 10 megaflop computer. Entries without a time designator are in seconds.

	Computational Complexity			
Descriptor	$O(n^{\frac{1}{2}})$	$O(n)$	$O(n\log(n))$	$O(n^2)$
tiny	10^{-6}	10^{-5}	1×10^{-5}	0.001
small	10^{-5}	0.001	0.004	10
medium	10^{-4}	0.1	0.6	1.2 days
large	0.001	10	1.3 minutes	31.7 years
huge	0.01	16.7 minutes	2.8 hours	3×10^6 years
massive	0.1	1.2 days	3.9 months	3×10^9 years
supermassive	3.2	3.3 years	10 years	3×10^{15} years

The problem gets worse as more and more systems are producing **streaming data**. The earth observer and network traffic data are examples of this. Here, there is no "n" corresponding to the number of observations; instead the data are produced in a continuous stream, and techniques are required to produce models "on the fly" as the data are received. We will not discuss streaming data any further, but it is worth keeping the problem in mind as we proceed.

Clearly, methods are needed to reduce the computational complexity of algorithms. For example, the naive nearest-neighbor algorithm is of order n, since it requires the calculation of the distance to each point. Cross-validation of the nearest-neighbor classifier is of order $O(n^2)$, since it requires the distances between every pair of points. Similarly, the CCCD and CCD algorithms of Chapters 4 and 5 are order $n^2 + n * m$ and n^2, respectively. It must be stressed that these order calculations are for naive algorithms. In this chapter we will look at some clever approaches to reducing this computational complexity.

Typical results for algorithm complexity are of the form:

Theorem 6.1 (Agarwal et al.). *The minimum spanning tree of a set of n points in d dimensional Euclidean space can be found in $O(n^{2-2/(\lceil d/2 \rceil + 1)+\epsilon})$ for any fixed $\epsilon > 0$.*

This type of result may or may not come with an explicit algorithm to implement the result. We will be more interested in specific algorithms and techniques in this chapter than in theoretical results. The interested reader is directed to the vast literature on computational geometry and the theory of algorithms for specific results, both theoretical and algorithmic. This chapter will be a very brief look at some interesting and useful techniques.

6.2 KD-TREES

The purpose of kd-trees is to construct a data structure that allows the efficient search of a set of observations to find the one nearest a given observation. A kd-tree is a binary tree constructed from the data that partitions the data into rectangular regions, in a manner similar to CART (classification and regression trees: see [85] or the classic [28]). At each node of the tree, a variate and splitting threshold is chosen. Data with values less than the threshold for the chosen variate are sent to the left, those with values greater are sent to the right. See [21] and [62].

Because the data are partitioned into disjoint regions, the tree allows for a fast determination of which region a new observation falls into, and only this region and those close to it need to be searched for nearest neighbors.

6.2.1 Data Structure

In its simplest form, a node in a kd-tree consists of the following fields:

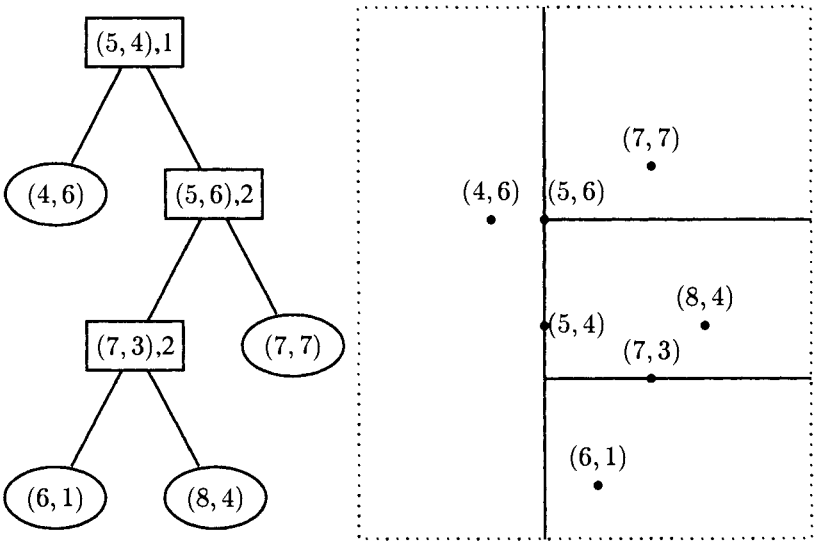


Fig. 6.1 A kd-tree for seven observations. Leaf nodes are denoted by ovals. Nodes are indicated by their observation and split, except for leaves which have no split. The regions defined by the tree are depicted on the right.

- Observation. This defines the splitting threshold, and also corresponds to the value contained in a leaf node. This may be implemented as an index into the data structure holding the observation, or an address associated with the observation.
- Split variate. This is an integer corresponding to the variate on which to split.
- Left node. This is a pointer to a kd-tree.
- Right node. This is also a pointer to a kd-tree.

Other information can be stored at a node. For instance, one could store the defining corners of the bounding rectangle for the data below the node. One can also store summary statistics for the data below the node, the class or other attributes of the observation, and so on.

Figure 6.1 depicts a small tree and the regions defined by the tree in the plane. Consider using this tree to find the nearest neighbor to the point $x = (6, 2)$ among these seven points. First we traverse the tree to find the region in which x lies: right, left, left. So we have done three comparisons of single values. We are now in the lower right section. We need to test this section, the one above, and the one to the left. This can be done via moving up and down the tree. The key point is that we never have to check the distance from x to $(7, 7)$. This trivial example illustrates the basic idea. Further efficiency, in larger trees, can be obtained by geometric reasoning. In this simple example, by noting that the distance between x and $(6, 1)$ is 1, and hence

the circle is contained in the region of the leaf, we need not test any of the other regions. Thus, we have found the nearest neighbor with three floating point tests, a single distance calculation, and a containment test, in place of seven distance tests.

6.2.2 Building the Tree

Ideally, one wants the tree to be balanced, and to have roughly square regions. A balanced tree can be achieved by ensuring that each split divides the data in the node in half. Doing this, however, requires $O(n \log n)$ operations; for example, we can sort on one of the variables. Instead, one can approximately balance the tree by various methods. In this section, we will discuss some of these.

The tree building algorithm requires a method for selecting the split (the variable on which the split will be made) and the observation (which determines the threshold). First let us consider methods for selecting the split.

One method is to simply select the split at random. Clearly one does not want to select the same split for all nodes, as illustrated in Figure 6.2. Splitting on the same variable results in many long thin regions, and poor performance of the search algorithm. The splits in Figure 6.1 were random (more properly, they were selected arbitrarily, simply for the purpose of illustration).

Another choice would be to pick the variate with the largest variance for the split, as is suggested by [146]. Note that the variance of a variate can be computed in $O(n)$ time. Other possibilities are easy to imagine. One could select from the s most spread variates the one which is least skewed, for example, in order to try to make the selection of the observation easier.

The strategy of the observation selection is to try to produce a balanced tree. One can produce a balanced tree by selecting the median, or an approximately balanced tree by selecting the mean or midpoint for the observation. Having observed that the median can produce long thin regions for skewed distributions, Moore in his dissertation ([141]) suggests using the midpoint.

Just as with the split, one could select the observation at random. In this case, the algorithm for tree building is extremely fast. However, since it is assumed that the main computational burden is in the search (which presumably will be performed many times), it is generally worth trading off the speed of tree building for better performance in tree search.

Note that one can stop the growing of the tree at any given level. For example, one could allow leaf nodes to contain 100 observations, instead of a single one, and stop growing the tree when the node reaches this number. Then, when the tree is searched, the 100 observations need to be checked at each leaf node. This is only of use when there is insufficient memory available to hold the full tree (or, for that matter, the full data set). In this case, the data itself would reside on disk, and only pointers to the data would be kept in the tree (for instance, each leaf would point to a different file on the disk). The disk would only be searched for those leaves that the search algorithm visited, and so the tree structure would still result in a considerable savings over a brute force search.

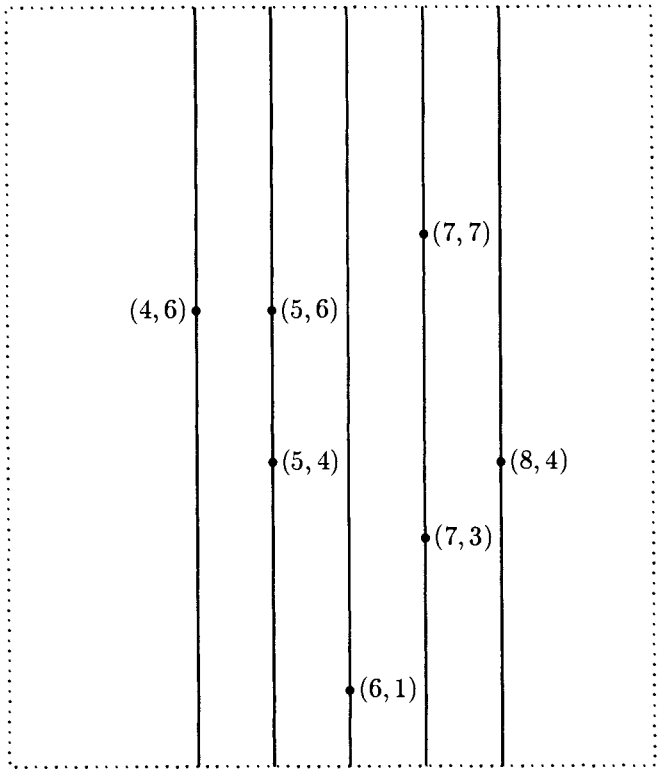


Fig. 6.2 A poor choice for the splits in a kd-tree.

Problems

6.1 Given the data of Figure 6.1, what is the optimal tree for finding the point nearest a given x? What is its worse case time?

6.2.3 Searching the Tree

Searching the kd-tree for a nearest neighbor to a newly presented vector x proceeds as follows. First, the tree is searched to find the leaf node into which x falls. The distance between x and this leaf is computed. The algorithm then proceeds back up the tree. At the parent node to the leaf, it is determined whether the other node could contain a nearer neighbor: if the current nearest distance is smaller than the split difference, then there cannot be a closer point down the other branch; otherwise the other branch is searched. The algorithm then proceeds up the tree, until the root node is reached, at which time the algorithm terminates.

This algorithm can be sped up slightly by retaining the bounding rectangle of the points below the node. This can be used, instead of the split criterion alone, to

determine whether it is possible that the nearest neighbor is found below the node. Other information can be kept to reduce the necessity of searching branches that do not contain potential minima. This adds complexity to the construction of the tree, and can add complexity (and hence time) to the search. Whether this extra complexity is justified depends on the problem at hand.

If we don't insist on an exact nearest neighbor, we can improve the speed of the algorithm, at the expense of not necessarily guaranteeing the nearest neighbor. [8] describe a priority queue approach, in which instead of going through the tree systematically, we choose the branches based on their distance to the point, and thus prioritize the search to check the branches most likely to contain a nearest neighbor first. Thus we can stop the process after a given amount of time, and have an approximation to the nearest neighbor. By searching the most likely branches first, they hope to find good approximations early on in the search, and they provide some empirical evidence that this indeed works in practice.

Problems

6.2 Implement a kd-tree version of a relative neighborhood graph. Test empirically whether it is faster than doing the brute force $O(n^3)$ algorithm.

6.3 CLASS COVER CATCH DIGRAPHS

The class cover catch digraph requires two searches of two different datasets for its construction. First, for every X observation x, the distance to the nearest Y observation must be computed, in order to determine the radius r_x for each x. This can be done by utilizing a kd-tree for the Y observations, and searching it for each X. Second, for each x, r_x pair, the X observations which are covered by the sphere $B_r(x)$ must be determined. This can be done by utilizing a kd-tree for the X observations, and modifying it to use r_x as the minimum distance, retaining those that fall below r_x in their distance to x.

This can dramatically reduce the computational complexity of constructing the class cover catch digraph. If X consists of n d-dimensional observations and Y consists of m observations, the complexity of the brute force approach is $O(n^2d + mnd)$.

In order to implement the greedy algorithm for selection of the (approximately) minimum dominating set, one needs to repeatedly run through the edge list, finding the vertex v with the maximum number of edges, then removing the edges to $\bar{N}(v)$, until there are no edges left. If the graph is too large to fit in memory, this can be accomplished by repeatedly running the modified kd-tree algorithm to find the covered observations. One such algorithm is given in Algorithm 6.1.

input : A kd-tree for the $\mathcal{X}$ observations, and their associated radii
output : A dominating set $\mathcal{S}$

initialize: $\mathcal{S} = \emptyset$
while *the tree is not empty* **do**
  for *each* $x \in X$ **do**
    use the modified algorithm to find $d_{out}(x)$;
    $s = \arg\max\{d_{out}(x)\}$;
    $\mathcal{S} = \mathcal{S} \cup \{s\}$;
    $\mathcal{X} = \mathcal{X} \setminus \{s\}$;
    remove $\tilde{N}(s)$ from the tree;
  end
end

Algorithm 6.1: A fast dominating set algorithm

This algorithm in effect finds the (approximately) minimum dominating set without explicitly calculating the graph. This suggests that for very large data sets, if we use a modified greedy algorithm which removes from consideration any points already covered, we can greatly reduce the running time of the algorithm, at the expense of the size of the dominating set.

Note that in Algorithm 6.1 we need to run through the algorithm for finding the covered points many times, which for large data sets is impractical. For very large data sets, the kd-tree itself may be too large to fit in memory, and so other algorithms are needed.

An approximate algorithm can be obtained via probabilistic methods. The idea is to select candidates for the dominating set at random. Because of the two-part nature of the CCCD algorithm (compute the radii, then the dominating set), there are two issues that must be addressed. If it is practical, using the kd-tree algorithm discussed above, to compute the radii, we can then use Algorithm 6.2.

input : Observations $\mathcal{X} = \{x_1, \ldots, x_n\}$ and associated radii $\mathcal{R} = \{r_1, \ldots, r_n\}$
output : A dominating set $\mathcal{S}$

initialize: $\mathcal{S} = \emptyset$, choose n' such that n' observations can be handled in available memory
while *there are observations to be processed* **do**
  draw n' observations, $\mathcal{X}'$ from $\mathcal{X}$;
  select S' using Algorithm 6.1 on $\mathcal{X}'$;
  $\mathcal{S} = \mathcal{S} \cup \mathcal{S}'$;
end

Algorithm 6.2: A fast dominating set algorithm using multiple subsets

Algorithm 6.2 returns a dominating set. It does not necessarily return a minimum dominating set, and in fact can be expected to be strictly larger than the one the greedy algorithm (Algorithm 6.1) would return, if it were practical.

If computing the radius for every X observation is impractical, then we have two possible algorithms. The first assumes that the problem is with n, that is, that it is possible to compute the radius for a small number of X observations, just not all of them. The second algorithm, which does not return a true dominating set, can be used to produce an approximation even when both n and m are too large to be manageable. The algorithm is given in Algorithm 6.3.

input : Observations $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\}$
output : A dominating set $\mathcal{S}$ for the CCCD on $\mathcal{X}$

initialize: $\mathcal{S} = \emptyset$, $\mathcal{X}' = \mathcal{X}$, and $P = \{\frac{1}{n}, \ldots, \frac{1}{n}\}$, a vector of length n
while $\mathcal{X}' \neq \emptyset$ **do**
  draw x from $\mathcal{X}$ with probability P;
  $r = d(x, Y)$;
  $\bar{N}(x) = \{x' \in \mathcal{X}' : d(x, x') < r\}$;
  $\mathcal{S} = \mathcal{S} \cup \{x\}$;
  $\mathcal{X}' = \mathcal{X}' \setminus \bar{N}(x)$;
  $P_i = d(\mathcal{S}, x_i)$;
  normalize P to sum to 1;
end

Algorithm 6.3: A fast dominating set algorithm using sampling

If necessary, the P vector can be eliminated, resulting in an algorithm that always selects uniformly from the remaining observations. The purpose of the P vector is to try to ensure that regions far from those already investigated are searched, resulting in an algorithm that is more likely to find a small dominating set in some problems. Note that this algorithm, like other greedy algorithms, can be stopped at any time, and returning a set that covers all but $\mathcal{X}'$. Finally, we can put the ideas of the previous algorithms together into one, as given in Algorithm 6.4.

Algorithm 6.4 does not guarantee that the resulting set S is a dominating set. In fact, the radii chosen are not even guaranteed to produce balls that cover no Y observations, and so the graphs that we are using are not necessarily subgraphs of the CCCD. However, for some very large practical problems, the resulting set may be significantly better than the alternative, which may be to subsample the data and use only the subsampled observations.

Problems

6.3 Investigate empirically the algorithms discussed in this section. How much of a trade-off do you find between them?

input : Observations $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\}$
output : A set $\mathcal{S}$ for the CCCD on $\mathcal{X}$

Set n' and m' so that Algorithm 6.1 is feasible with these values;
initialize: $\mathcal{S} = \emptyset$, $\mathcal{X}' = \mathcal{X}$, and $\mathcal{Y}' = \mathcal{Y}$
while $\mathcal{X}' \neq \emptyset$ **do**
  draw n' observations from $\mathcal{X}'$ and m' from $\mathcal{Y}'$;
  Set $\mathcal{S}'$ to be the results of Algorithm 6.1 using these observations;
  $\mathcal{S} = \mathcal{S} \cup \mathcal{S}'$;
  remove the drawn observations from $\mathcal{X}'$ and $\mathcal{Y}'$;
end

Algorithm 6.4: A fast approximation to the minimum dominating set

6.4 CLUSTER CATCH DIGRAPHS

The naive cluster catch digraph requires the calculation of distances between each pair of points. Since the algorithm is sequential, it is well suited to a kd-tree implementation. For each x it is necessary to calculate r_x by finding the closest observation, then the next closest, and so on, until the stopping criterion is met, or all observations have been chosen.

Once the graph has been defined, the selection of the dominating set can proceed as before. Alternatively, one can utilize the algorithms discussed in Section 6.3.

Problems

6.4 Modify the kd-tree search algorithm to return the next closes observation to x each time it is run.

6.5 VORONOI REGIONS AND DELAUNAY TRIANGULARIZATIONS

We saw above a complexity theorem for general Voronoi tessellations. In the plane, one has the following:

Theorem 6.2. *The Voronoi tessellation of a set of n points in the plane can be found in $O(n \log n)$ time. Similarly for the Delaunay triangularization.*

Given the Voronoi regions for the points, the problem of nearest neighbors comes down to finding the polygonal region in which a new observation falls. This can be accomplished in the plane in $O(\log n)$ time ([156]; see also Section 2.2.2). Unfortunately, in high-dimensions the calculation of the Voronoi diagram is prohibitive, and so this is not a practical solution in these cases.

Similarly, given the Delaunay triangularization, the calculation of relative neighborhood graphs and their relatives is greatly reduced, however this is only practical for low-dimensional data.

Finding Voronoi regions in high-dimensional data is difficult. Reference [7] provides a method to efficiently find approximate Voronoi regions (see also [9]). Given the purpose of random graphs for statistical pattern recognition (i.e.: pattern recognition) there may well be uses for approximations to the various types of graphs we have encountered in these chapters.

6.6 FURTHER READING

Many researchers have investigated kd-trees and variants for nearest-neighbor search. A few interesting articles are [172], [173], [120] and [121]. Reference [8] uses kd-trees for vector quantization. Reference [142] discusses using kd-trees to speed up the EM algorithm in mixture models, and has an extremely fast implementation that has been applied to a number of large data problems. There are many structures similar to the kd-trees which may be useful in data random graphs and other pattern recognition techniques. Reference [143] describes using AD-trees for computing contingency tables, used for machine learning rule finding and Bayesian networks. Several articles concerning nearest-neighbor type calculations are available in [73]. Expected case performance for approximate nearest-neighbor searches is given in [6]. They show that one can achieve linear space and logarithmic query time, on average.

Something we have not touched on in this book is the efficient storing of large graphs. Let n be the order of the graph, and s be its size. Again naively, we may consider that there are four different ways to store a graph. We could store (1) the adjacency matrix, an $O(n^2)$ storage requirement; (2) the incidence matrix (the rows are vertices, the columns edges) which is $O(ns)$; (3) the edge adjacency list (a vector of neighbors for each vertex) which is $O(n^2)$; or (4) we could store an $s \times 2$ matrix of vertex pairs, corresponding to the edges, $O(s)$. Obviously, for certain calculations, some of these are more efficient than others.

In the above calculations, we have been a bit cavalier, in that we have not taken into account the storage requirements for the elements of the matrices. For example, the adjacency matrix requires a single bit per entry, while the edge adjacency list requires about $log(n)$ bits per entry. These calculations are done much more carefully in [195], but since we are still being naive, we are not assuming careful allocation (most programmers would allocate either 8 bits (character) or 16- (or 32-) bits (integer) for the adjacency matrix, we are sad to say).

The adjacency and incidence matrices can make use of sparse matrix techniques, since for most graphs these are quite sparse. For typical graphs, $s \gg n$, and so if storage is the only consideration, one would naturally prefer the adjacency matrix to the incidence matrix.

There is an entirely different way of looking at the storage of graphs that is covered in great detail in [195]. We have discussed this implicitly throughout this book, but have not been explicit: the graph representations themselves can be the way to store

the graph. For example, with interval graphs, one need only store the endpoints of the intervals. With these, the graph is completely defined, and the storage is $O(n)$. If the graph is planar, it can be represented as a coin graph. Finding such an efficient representation for a given graph may not be trivial, however.

In fact, for all data random graphs, the graph is completely determined by the data, and so one need only store the data. Usually, this is not enough in practice, as the determination of adjacency is complex given only the points. For example, in the relative neighborhood graphs of Chapter 3, determining whether two points have an edge is quite complex, since one must check that there are no points in the intersection region of the neighborhoods. On the other hand, the catch digraphs of Chapters 4 and 5 are different. Given the points and the radii, a check for adjacency needs a single distance calculation and a few comparisons, which may be acceptable considering the reduction in storage from a complete adjacency matrix.

So, if one could take a given graph and produce its representation as an interval graph, sphere catch digraph, or some other similar class of graphs, one would have a new representation of the graph that may be much more efficient than any of the four naive methods discussed above. In many situations, the representation can be chosen to be one in which tests for adjacency (or whatever calculations need to be done) are also efficient. The interested reader is encouraged to investigate [195] for more information.

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

References

1. H. Abdel-Wahab, I. Stoica, F. Sultan, and K. Wilson. A simple algorithm for computing minimum spanning trees in the internet. *Information Sciences*, 101:47–69, 1997.

2. L. Abrams, D. E. Fishkind, and C. E. Priebe. A proof of the spherical homeomorphism conjecture for surfaces. *IEEE Transactions on Medical Imaging*, 21:1564–1566, 2002.

3. M. Alenderfer and R. Blashfield. *Cluster Analysis.* Sage Publications, Beverly Hills, CA, 1984.

4. E. Anderson. The irises of the gaspe peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935.

5. I. Anderson. *A First Course in Discrete Mathematics.* Springer, New York, 2001.

6. S. Arya and H. Y. Fu. Expected-case complexity of approximate nearest neighbor searching. *SIAM Journal on Computing*, 32:793–815, 2003.

7. S. Arya, T. Malamatos, and D. M. Mount. Space-efficient approximate Voronoi diagrams. In *Proceedings of the 34th ACM Symposium on the Theory of Computing*, pp. 721–730, 2002.

8. S. Arya and D. M. Mount. Algorithms for fast vector quantization. In J. A. Storer and M. Cohn, editors, *Proceedings of the Data Compression Conference*, pp. 381–390. IEEE Press, 1993.

9. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

10. T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. In *Proceedings of the Fourth Annual Symposium on Computational Geometry*, pp. 252–257, 1988.

11. G. A. Bakken, G. W. Kauffman, P. C. Jurs, K. J. Albert, and S. S. Stitzel. Pattern recognition analysis of optical sensor array data to detect nitroaromatic compound vapors. *Sensors and Actuators B*, 79:1–10, 2001.

12. R. Balakrishnan and K. Ranganathan. *A Textbook of Graph Theory*. Springer, New York, 2000.

13. V. K. Balakrishnan. *Introductory Discrete Mathematics*. Dover Publications, New York, 1991.

14. S. Baluja, H. Rowley, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:23–38, 1998.

15. J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, New York, 2001.

16. J. Barrera, F. De Assis Zampirolli, and R. De Alencar Lotufo. Morphological operators characterized by neighborhood graphs. In *Proceedings of the X Brazilian Symposium on Computer Graphics and Image Processing*, pp. 179–186, 1997.

17. M. Basseville. Distance measures for signal processing and pattern recognition. *Signal Processing*, 18:349–369, 1989.

18. J. Beirlant, E. J. Dudewicz, L. Györfi, and E. van der Meulen. Nonparametric entropy estimation: An overview. *Intern. J. Math. Stat. Sci.*, 6(1):17–39, 1997.

19. R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.

20. C. A. Beltrami, V. Della Mea, and N. Finato. Structure analysis of breast lesions using neighborhood graphs. *Analytical and Quantitative Cytology and Histology*, 17:143–150, 1995.

21. J. L. Bentley. Multidimensional binary search trees used for associative searches. *Communications of the ACM*, 18(9):509–517, 1975.

22. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 2002.

23. B. Bollobás. *Random Graphs*. Academic Press, London, 1985.

24. B. Bollobás. *Modern Graph Theory*. Springer-Verlag, New York, 1998.

25. B. Bollobás. *Random Graphs*. Cambridge University Press, Cambridge, 2001.

26. D. L. Borchers, S. T. Buckland, and W. Zucchini. *Estimating Animal Abundance*. Springer, London, 2002.

27. I. Borg and P. Groenen. *Modern Multidimensional Scaling*. Springer, New York, 1997.

28. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, Boca Raton, FL, 1998.

29. M. R. Brito, E. L. Chávez, A. J. Quiroz, and J. E. Yukich. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics and Probability Letters*, 35:33–42, 1997.

30. E. Canetti. *Crowds and Power*. Noonday Press, New York, 1984.

31. A. Cannon and L. C. Cowen. Approximation algorithms for the class cover problem. *Annals of Math. and AI*. To Appear: Available at www1.cs.columbia.edu/~cannon

32. E. Ceyhan and C. E. Priebe. Central similarity proximity maps in Delaunay tessellations. In *Proceedings of the Joint Statistical Meetings*, 2003. To appear.

33. T. K. Chalker, A. P. Godbole, P. Hitczenko, J. Radcliff, and O. G. Ruehr. On the size of a random sphere of influence graph. *Advances in Applied Probability*, 31:596–609, 1999.

34. G. Chartrand and L. Lesniak. *Graphs & Digraphs*. Chapman & Hall/CRC, Boca Raton, FL, 1996.

35. D. Chaudhuri and B. B. Chaudhuri. A novel multiseed nonhierarchical data clustering technique. *IEEE Transaction on Systems, Man, and Cybernetics*, 27(5):871–877, 1997.

36. D. Cohn, E. A. Riskin, and R. Ladner. Theory and practice of vector quantizers trained on small training sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):54–65, 1994.

37. N. Cressie. *Statistics for Spatial Data*. Wiley, New York, 1993.

38. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, 2000.

39. P. Dalgaard. *Introductory Statistics with R*. Springer, New York, 2002.

40. H. A. David. *Order Statistics*. Wiley, New York, 1981.

41. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and applications*. Springer-Verlag, Berlin, 1997.

42. V. S. Devi and M. N. Murty. An incremental prototype set building technique. *Pattern Recognition*, 35:505–513, 2002.

43. J. G. DeVinney. *The Class Cover Problem and its Applications in Pattern Recognition*. Ph.D. thesis, The Johns Hopkins University, 2003.

44. J. G. DeVinney, C. E. Priebe, D. J. Marchette, and D. A. Socolinsky. Classification using random walk class covers. In *Proceedings of the Joint Statistical Meetings*, 2003. To appear.

45. J. G. DeVinney and J. C. Wierman. A SLLN for a one-dimensional class cover problem. *Statistics and Probability Letters*, 2003. To appear.

46. L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

47. T. A. Dickinson, J. White, J. S. Kauer, and D. R. Walt. A chemical-detecting system based on a cross-reactive optical sensor array. *Nature*, 382:697–700, 1996.

48. R. Diestel. *Graph Theory*. Springer, New York, 2000.

49. D. L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality, 2000. American Mathematical Society Lecture, available at www-stat.stanford.edu/~donoho/Lectures/AMS2000/AMS2000.html

50. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

51. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, 2000.

52. R. A. Dwyer. The expected size of the sphere-of-influence graph. *Computational Geometry*, 5:155–164, 1995.

53. H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983.

54. B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, London, 1993.

55. D. Eppstein, M. S. Paterson, and F. F. Yao. On nearest-neighbor graphs. *Discrete and Computational Geom.*, 17:263–282, 1997.

56. B. S. Everitt. *Cluster Analysis, Third Edition*. Wiley, New York, 1993.

57. R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(Part II):179–188, 1936.

58. F. Fleuret and D. Geman. Coarse-to-fine face detection. *IJCV*, 41:85–107, 2001.

59. L. R. Foulds. *Graph Theory Applications*. Springer, New York, 1992.

60. C. Frélicot and F. Lebourgeois. A pretobology-based supervised pattern classifier. In *Proceedings of the Fourteenth International Conference on Pattern Recognition, Volume 1*, pp. 106–109, 1998.

61. Y. Freund and R. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55:249–266, 1997.

62. J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic time. Technical Report CS Rep. 475-482, Stanford Department of Computer Science, 1975.

63. J. H. Friedman and L. C. Rafsky. Multivariate generalizations of the Wald-Wolfowitz and Smirnov two-sample tests. *Annals of Statistics*, 7(4):697–717, 1979.

64. J. H. Friedman and L. C. Rafsky. Graph-theoretic measures of multivariate association and prediction. *Annals of Statistics*, 11(2):377–391, 1983.

65. K. Fukunaga. *Statistical Pattern Recognition*. Academic Press, San Diego, 1990.

66. Z. Füredi. The expected size of a random sphere-of-influence graph. *Bolyai Society Mathematical Studies*, 6:319–326, 1997.

67. K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systemic Zoology*, 18:259–278, 1969.

68. R. S. Garfinkel and G. L. Nemhauser. *Integer Programming*. Wiley, New York, 1972.

69. R. Gentleman and A. C. Vandal. Computational algorithms for censored-data problems using intersection graphs. *Journal of Computational and Graphical Statistics*, 10:403–421, 2001.

70. A. Gersho and R. M. Gray. *Vector Quantization and Signal Processing*. Kluwer Academic Publishers, Boston, 1992.

71. G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Science*, 97(22):12079–12084, 2000.

72. W. Gibson. *Pattern Recognition*. G. P. Putnam's Sons, New York, 2003.

73. M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, editors. *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*. American Mathematical Society, Providence, RI, 2002.

74. T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri et al. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

75. J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, New York, 1997.

76. A. D. Gordon. *Classification*. Chapman and Hall, London, 1999.

77. R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44:2325–2384, 1998.

78. D. Gries and F. B. Schneider. *A Logical Approach to Discrete Math*. Springer, New York, 1993.

79. J. Gross and J. Yellen. *Graph Theory and Its Applications*. CRC Press, Boca Raton, FL, 1999.

80. L. Guibas, J. Pach, and M. Sharir. Sphere of influence graphs in higher dimensions. *Colloquia Mathematica Societatis Janos Bolyai: Intuitive Geometry*, pp. 131–137, 1991.

81. D. J. Hand. *Construction and Assessment of Classification Rules*. Wiley, New York, 1997.

82. F. Harary, M. S. Jacobson, M. J. Lipman, and F. R. McMorris. Abstract sphere-of-influence graphs. *Mathematical and Computational Modelling*, 17(11):77–83, 1993.

83. P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

84. J. A. Hartigan. *Clustering Algorithms*. Wiley, New York, 1975.

85. T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, New York, 2001.

86. S. Haykin. *Neural networks: A Comprehensive Foundation*. Prentice-Hall, Beverly Hills, CA, 1999.

87. T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Domination in Graphs: Advanced Topics*. Marcel Dekker, New York, 1998.

88. T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in Graphs*. Marcel Dekker, New York, 1998.

89. H. J. A. M. Heijmans. *Morphological Image Operators*. Academic Press, London, 1997.

90. A. O. Hero and B. Ma. Convergence rates of minimal graphs with random vertices, 2002. Preprint.

91. A. O. Hero, B. Ma, O. J. J. Michel, and J. Gorman. Alpha-divergence for classification, indexing and retrieval. Technical Report CSPL-328, Communications and Signal Processing Laboratory, University of Michigan, 2001.

92. A. O. Hero and O. J. J. Michel. Asymptotic theory of greedy approximations to minimal k-point random graphs. *IEEE Transactions on Information Theory*, 45(6):1921–1938, 1999.

93. A. O. Hero and O. J. J. Michel. Estimation of rényi information divergence via pruned minimal spanning trees. In *Proceedings of the IEEE Signal Processing Workshop on Higher-Order Statistics*, pp. 264–268, 1999.

94. P. Hitczenko, S. Janson, and J. E. Yukich. On the variance of the random sphere of influence graph. *Random Structures Algorithms*, 14(2):139–152, 1999.

95. T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

96. T. S. Holm and K. P. Bogart. On tolerance sphere-of-influence graphs. *Bulletin of the ICA*, 24:33–46, 1998.

97. M. Ichino and J. Sklansky. The relative neighborhood graph for mixed feature variables. *Pattern Recognition*, 18(2):161–167, 1985.

98. R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

99. J. E. Jackson, editor. *A User's Guide to Principal Components*. Wiley-Interscience, New York, 1991.

100. H. Jahn. A graph structure for grey value and texture segmentation. In Jolion and Kropatsch [106].

101. A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

102. L. F. James, C. E. Priebe, and D. J. Marchette. Consistent estimation of mixture complexity. *Annals of Statistics*, 29(5):1281–1296, 2001.

103. S. Janson, T. Łuczak, and A. Rucinński. *Random Graphs*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, New York, 2000.

104. J. W. Jaromczyk and G. T. Toussaint. Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1517, 1992.

105. R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1998.

106. J.-M. Jolion and W. G. Kropatsch, editors. *Graph Based Representations in Pattern Recognition*. Springer, New York, 1998.

107. I. T. Jolliffe, editor. *Principal Component Analysis*. Springer Verlag, New York, 2002.

108. D. Jungnickel. *Graphs, Networks and Algorithms*. Springer, Berlin, 2002.

109. P. C. Kainen. *Utilizing Geometric Anomalies of High Dimension: When Complexity Makes Computation Easier*. Birkhäuser, Boston, 1997.

110. M. Karonski, K. Singer, and E. Scheinerman. Random intersection graphs: The subgraph problem. *Combinatorics, Probability and Computing*, 8:131–159, 1999.

111. S. Kim, S. Kwon, and D. Cook. Interactive visualization of hierarchical clusters using MDS and MST. *Metrika: Special Issue on Interactive Statistics*, 51:39–51, 2000.

112. D. E. Knuth. *Things a Computer Scientist Rarely Talks About*. CSLI Publications, Stanford, CA, 2001.

113. P. Koebe. Kontaktprobleme der konformen appildung. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften, Leipzig, Mathematische-Pysische Klasse*, 88:141–164, 1936.

114. V. F. Kolchin. *Random Graphs*. Cambridge University Press, Cambridge, 1999.

115. D. J. Kriegman, M. H. Yang, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:34–58, 2002.

116. S. R. Kulkarni, G. Lugosi, and S. S. Venkatesh. Learning pattern classification—a survey. *IEEE Transactions on Information Theory*, 44:2178–2206, 1998.

117. P. M. Lankford. Regionalization: Theory and alternative algorithms. *Geographic Analysis*, 1:196–212, 1969.

118. D. T. Lee. Relative neighborhood graphs in the l_1-metric. *Pattern Recognition*, 18:327–332, 1985.

119. M. J. Lipman. Maximum tolerance sphere-of-influence graphs. *Congressus Numerantium*, 121:195–203, 1996.

120. S. Maeewongvatana and D. M. Mount. An empirical study of a new approach to nearest neighbor searching. In *Proceedings of the 3rd International Workshop on Algorithm Engineering and Experiments*, pp. 172–187, 2001.

121. S. Maeewongvatana and D. M. Mount. On efficiency of nearest neighbor searching with data clustered in lower dimensions. In *International Conference on Computational Sciences*, pp. 842–851, 2001. Spring Lecture Notes LNCS 2073.

122. H. Maehara. A digraph represented by a family of boxes or spheres. *Journal of Graph Theory*, 8:431–439, 1984.

123. D. P. Mandal and C. A. Murthy. Selection of alpha for alpha-hull in $\mathbb{R}^2$. *Pattern Recognition*, 30(10):1759–1767, 1997.

124. D. J. Marchette. Profiling users by their network activity. In *Proceedings of the Joint Statistical Meetings*, 2003. To appear.

125. D. J. Marchette and C. E. Priebe. Characterizing the scale dimension of a high dimensional classification problem. *Pattern Recognition*, 36, 2003.

126. D. J. Marchette, J. L. Solka, R. Guidry, and J. Green. The advanced distributed region of interest tool. *Pattern Recognition*, 31, 1999.

127. K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, New York, 1995.

128. W. L. Martinez and A. R. Martinez. *Computational Statistics Handbook with MATLAB*. Chapman & Hall/CRC, Boca Raton, FL, 2002.

129. J. Matoušek and J. Nešetřil. *Invitational to Discrete Mathematics*. Oxford University Press, Oxford, 2002.

130. D. W. Matula and R. R. Sokal. Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographic Analysis*, 12:205–222, 1980.

131. T. A. McKee and F. R. McMorris. *Intersection Graph Theory*. SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 1999.

132. G. J. McLachlan. On bootstrapping the likelihood ratio test statistic for the number of components in a normal mixture. *Applied Statistics*, 36:318–324, 1987.

133. G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, New York, 1997.

134. G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, New York, 2000.

135. F. R. McMorris and C. Wang. Sphere-of-attraction graphs. *Congressus Numerantium*, 142:149–160, 2000.

136. R. Merris. *Graph Theory*. Wiley, New York, 2001.

137. T. S. Michael and T. Quint. Sphere of influence graphs: A survey. *Congressus Numerantium*, 105:153–160, 1994.

138. T. S. Michael and T. Quint. Sphere of influence graphs: Edge density and clique size. *Mathematical and Computational Modelling*, 20(7):19–24, 1994.

139. T. S. Michael and T. Quint. Sphere of influence graphs in general metric spaces. *Mathematical and Computational Modelling*, 29:45–53, 1999.

140. P. W. Mielke, Jr. and K. J. Berry. *Permutation Methods: A Distance Function Approach*. Springer, New York, 2001.

141. A. Moore. *Efficient Memory-based Learning for Robot Control*. Ph.D. thesis, University of Cambridge, 1991.

142. A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees, 1998. citeseer.nj.nec.com/moore98very.html

143. A. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, 1998.

144. S. Nanda and C. M. Newman. Random nearest neighbor and influence graphs on z^d. *Random Structures Algorithms*, 15(3–4):262–278, 1997.

145. A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. Wiley, New York, 2000.

146. S. M. Omohundro. Efficient algorithms with neural network behavior. *Journal of Complex Systems*, 1:273–347, 1987.

147. J. O'Rourke. Computing the relative neighborhood graph in the l_1 and l_∞ metrics. *Pattern Recognition*, 15:189–192, 1982.

148. J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, Cambridge, 1993.

149. J. O'Rourke and G. T. Toussaint. Pattern recognition. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 43, pp. 797–813. CRC Press, New York, 1997.

150. J. Pach and P. K. Agarwal. *Combinatorial Geometry*. Wiley, New York, 1995.

151. M. C. Pardo and J. A. Pardo. Use of Rényi's divergence to test for the equality of the coefficients of variation. *Journal of Computational and Applied Mathematics*, 116:93–104, 2000.

152. M. S. Paterson and F. F. Yao. On nearest-neighbor graphs. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, pp. 416–426. Springer, 1992.

153. M. D. Penrose and J. E. Yukich. Central limit theorems for some graphs in computational geometry. *Annals of Applied Probability*, 11(4):1005–1041, 2001.

154. B. Polster. *The Mathematics of Juggling*. Springer, New York, 2000.

155. A. Pordt and T. Reisz. Linked cluster expansions beyond nearest neighbor interactions: Convergence and graph classes. *International Journal of Modern Physics A*, 12(21):3739–3757, 1997.

156. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

157. C. E. Priebe. Adaptive mixture density estimation. *Journal of the American Statistical Association*, 89:796–806, 1994.

158. C. E. Priebe. Olfactory classification via interpoint distance analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):404–413, 2001.

159. C. E. Priebe, J. G. DeVinney, and D. J. Marchette. On the distribution of the domination number for random class cover catch digraphs. *Statistics and Probability Letters*, 55:239–246, 2001.

160. C. E. Priebe and D. J. Marchette. Adaptive mixtures: Recursive nonparametric pattern recognition. *Pattern Recognition*, 24(12):1197–1209, 1991.

161. C. E. Priebe and D. J. Marchette. Adaptive mixture density estimation. *Pattern Recognition*, 26(5):771–785, 1993.

162. C. E. Priebe and D. J. Marchette. Characterizing the dimensionality of a classification problem. In *Computing Science and Statistics*, volume 32, pp. 48–62, 2000.

163. C. E. Priebe, D. J. Marchette, J. G. DeVinney, and D. A. Socolinsky. Classification using class cover catch digraphs. *Journal of Classification*, 20:3–23, 2003.

164. C. E. Priebe, D. J. Marchette, and D. M. Healy, Jr. Integrated sensing and processing decision trees. Technical Report 637, Department of Mathematical Sciences, Johns Hopkins University, 2002. submitted for publication.

165. C. E. Priebe, D. J. Marchette, and D. M. Healy, Jr. Integrated sensing and processing for statistical pattern recognition. In D. Rockmore and D. M. Healy, Jr., editors, *Modern Signal Processing*. Cambridge University Press, Cambridge, 2003.

166. C. E. Priebe, J. L. Solka, D. J. Marchette, and T. Clark. Class cover catch digraphs for latent class discovery in gene expression monitoring by DNA microarrays. *Computational Statistics and Data Analysis*, 43(4):621–632, 2003.

167. E. Prisner. A characterization of interval catch digraphs. *Discrete Mathematics*, 71:285–289, 1989.

168. E. Prisner. Algorithms for interval catch digraphs. *Discrete Applied Mathematics*, 51:147–157, 1994.

169. Y. Qian and R. Zhao. Robust clustering based on global data distribution and local connectivity matrix. In *1997 IEEE Conference on Intelligent Processing Systems*, pp. 1629–1633, 1997.

170. R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2003. www.R-project.org.

171. J. D. Radke. On the shape of a set of points. In G. T. Toussaint, editor, *Computational Morphology*, pp. 105–136. Elsevier Science Publishers B.V. (North-Holland), 1988.

172. V. Ramasubramanian and K. K. Paliwal. Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding. *IEEE Transactions on Speech and Audio Processing*, 7:221–226, 1992.

173. V. Ramasubramanian and K. K. Paliwal. Fast nearest-neighbor search based on Voronoi projections and its application to vector quantization encoding. *IEEE Transactions on Speech and Audio Processing*, 7:221–226, 1999.

174. S. J. Raudys and A. K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:252–264, 1991.

175. D. L. Reilly and L. N. Cooper. An overview of neural networks: Early models to real world systems. In S. F. Zornetzer, J. L. Davis, C. Lau, and T. McKenna, editors, *An Introduction to Neural and Electronic Networks*. Academic Press, San Diego, 1995.

176. D. L. Reilly, L. N. Cooper, and C. Elbaum. A neural model for category learning. *Biological Cybernetics*, 45:35–41, 1982.

177. B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.

178. J.-R. Sack and J. Urrutia, editors. *Handbook of Computational Geometry*. North Holland, Amsterdam, 2000.

179. J. S. Sánchez, F. Pla, and F. J. Ferri. On the use of neighborhood-based non-parametric classifiers. *Pattern Recognition Letters*, 18:1179–1186, 1997.

180. J. S. Sánchez, F. Pla, and F. J. Ferri. Prototype selection for the nearest neighbor rule through proximity graphs. *Pattern Recognition Letters*, 18:507–513, 1997.

181. J. Scanlon and N. Deo. Graph-theoretic algorithms for image segmentation. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, pp. 141–144, 1999.

182. B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA, 1999.

183. D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, New York, 1992.

184. G. A. F. Seber. *Multivariate Observations*. Wiley, New York, 1984.

185. R. Sedgewick. *Algorithms in C. Part 5: Graph Algorithms*. Addison Wesley, Boston, 2002.

186. M. Sen, S. Das, A. B. Roy, and D. B. West. Interval digraphs: An analogue of interval graphs. *Journal of Graph Theory*, 13:189–202, 1989.

187. J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1997.

188. D. W. Shattuck and R. M. Leahy. Automated graph-based analysis and correction of cortical volume topology. *IEEE Transactions on Medical Imaging*, 20:1167–1177, 2001.

189. B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, New York, 1986.

190. J. Sklansky and G. Wassel. *Pattern Classifiers and Trainable Machines*. Springer Verlag, New York, 1979.

191. S. R. Slade and A. Godbole. Random sphere of influence graphs in the l_p metrics. *Congressus Numerantium*, 134:175–187, 1998.

192. C. G. Small. Multidimensional medians arising from geodesics on graphs. *Annals of Statistics*, 25:478–494, 1997.

193. D. A. Socolinsky, J. D. Neuheisel, C. E. Priebe, J. G. DeVinney, and D. J. Marchette. Fast face detection with a boosted CCCD classifier. In *CVPR 2003*, 2003. To appear.

194. J. L. Solka, D. J. Marchette, B. C. Wallet, V. I. Irwin, and G. W. Rogers. Identification of man-made regions in unmanned aerial vehicle imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20, 1998.

195. J. P. Spinrad. *Efficient Graph Representations*. American Mathematical Society, Providence, RI, 2003.

196. J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

197. S.-H. Teng. Combinatorial aspects of geometric graphs. *Computational Geometry*, 9:277–287, 1998.

198. R. Tibshirani, G. Walther, D. Botstein, and P. Brown. Cluster validation by prediction strength. citeseer.nj.nec.com/tibshirani01cluster.html

199. G. T. Toussaint. Note on optimal selection of independent binary-valued features for pattern recognition. *IEEE Transactions on Information Theory*, p. 618, 1971.

200. G. T. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.

201. G. T. Toussaint, editor. *Computational Morphology*. Elsevier Science Publications B.V. (North-Holland), 1988.

202. G. T. Toussaint. A graph-theoretic primal sketch. In G. T. Toussaint, editor, *Computational Morphology*, pp. 229–260. Elsevier Science Publishers B.V. (North-Holland), 1988.

203. G. V. Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):306–307, 1979.

204. Z. Tuza. Inequalities for minimal covering sets in set systems of given rank. *Discrete Applied Mathematics*, 51:187–195, 1994.

205. R. Urquhart. Graph theoretical clustering based on limited neighborhood sets. *Pattern Recognition*, 15(3):173–187, 1982.

206. R. Urquhart. Some properties of the planar euclidean relative neighborhood graph. *Pattern Recognition Letters*, 1:317–322, 1983.

207. G. Valiente. *Algorithms on Trees and Graphs*. Springer, Berlin, 2002.

208. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

209. W. N. Venebles and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 2002.

210. W. N. Venebles and D. M. Smith. *An Introduction to R*. Network Theory Limited, Bristol, 2001.

211. W. D. Wallis. *A Beginner's Guide to Graph Theory.* Birkhäuser, Boston, 2000.

212. W. D. Wallis. *Magic Graphs.* Birkhäuser, Boston, 2001.

213. W. D. Wallis. *A Beginner's Guide to Discrete Mathematics.* Birkhäuser, Boston, 2003.

214. K. Warwick and M. Kárný, editors. *Computer-Intensive Methods in Control and Signal Processing.* Birkhäuser, Boston, 1997.

215. G. Wegner. *Eigenschaften der Nerven Hololog*isce-einfacher Familien in R^n. Ph.D. thesis, Göttingen, 1967.

216. D. B. West. *Introduction to Graph Theory.* Prentice-Hall, Upper Saddle River, NJ, 2001.

217. J. White, J. S. Kauer, T. A. Dickinson, and D. R. Walt. Rapid analyte recognition in a device based on optical sensors and the olfactory system. *Analytical Chemistry*, 68:2191–2202, 1996.

218. P. Xiang and J. C. Wierman. Limit theory for the domination number of random class cover catch digraphs. In *Proceedings of the Joint Statistical Meetings*, 2003. To appear.

219. Y. Xu, V. Olman, and D. Xu. Minimum spanning trees for gene expression data clustering. *Genome Informatics*, 12:24–33, 2001.

220. L. Yang and S. M. LaValle. A framework for planning feedback motion strategies based on a random neighborhood graph. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pp. 544–549, 2000.

221. C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20(1):68–86, 1971.

222. S. F. Zornetzer, J. L. Davis, C. Lau, and T. McKenna, editors. *An Introduction to Neural and Electronic Networks.* Academic Press, San Diego, 1995.

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

Author Index

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

Subject Index

Random Graphs for Statistical Pattern Recognition. David J. Marchette

ISBN: 0-471-22176-7

WILEY SERIES IN PROBABILITY AND STATISTICS

ESTABLISHED BY WALTER A. SHEWHART AND SAMUEL S. WILKS

The ***Wiley Series in Probability and Statistics*** is well established and authoritative. It covers many topics of current research interest in both pure and applied statistics and probability theory. Written by leading statisticians and institutions, the titles span both state-of-the-art developments in the field and classical methods.

Reflecting the wide range of current research in statistics, the series encompasses applied, methodological and theoretical statistics, ranging from applications and new techniques made possible by advances in computerized practice to rigorous treatment of theoretical approaches.

This series provides essential and invaluable reading for all statisticians, whether in academia, industry, government, or research.

ABRAHAM and LEDOLTER · Statistical Methods for Forecasting
AGRESTI · Analysis of Ordinal Categorical Data
AGRESTI · An Introduction to Categorical Data Analysis
AGRESTI · Categorical Data Analysis, *Second Edition*
ALTMAN, GILL, and McDONALD · Numerical Issues in Statistical Computing for the Social Scientist
AMARATUNGA and CABRERA · Exploration and Analysis of DNA Microarray and Protein Array Data
ANDĚL · Mathematics of Chance
ANDERSON · An Introduction to Multivariate Statistical Analysis, *Third Edition*
*ANDERSON · The Statistical Analysis of Time Series
ANDERSON, AUQUIER, HAUCK, OAKES, VANDAELE, and WEISBERG · Statistical Methods for Comparative Studies
ANDERSON and LOYNES · The Teaching of Practical Statistics
ARMITAGE and DAVID (editors) · Advances in Biometry
ARNOLD, BALAKRISHNAN, and NAGARAJA · Records
*ARTHANARI and DODGE · Mathematical Programming in Statistics
*BAILEY · The Elements of Stochastic Processes with Applications to the Natural Sciences
BALAKRISHNAN and KOUTRAS · Runs and Scans with Applications
BARNETT · Comparative Statistical Inference, *Third Edition*
BARNETT and LEWIS · Outliers in Statistical Data, *Third Edition*
BARTOSZYNSKI and NIEWIADOMSKA-BUGAJ · Probability and Statistical Inference
BASILEVSKY · Statistical Factor Analysis and Related Methods: Theory and Applications
BASU and RIGDON · Statistical Methods for the Reliability of Repairable Systems
BATES and WATTS · Nonlinear Regression Analysis and Its Applications
BECHHOFER, SANTNER, and GOLDSMAN · Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons
BELSLEY · Conditioning Diagnostics: Collinearity and Weak Data in Regression

*Now available in a lower priced paperback edition in the Wiley Classics Library.

BELSLEY, KUH, and WELSCH · Regression Diagnostics: Identifying Influential Data and Sources of Collinearity
BENDAT and PIERSOL · Random Data: Analysis and Measurement Procedures, *Third Edition*
BERRY, CHALONER, and GEWEKE · Bayesian Analysis in Statistics and Econometrics: Essays in Honor of Arnold Zellner
BERNARDO and SMITH · Bayesian Theory
BHAT and MILLER · Elements of Applied Stochastic Processes, *Third Edition*
BHATTACHARYA and JOHNSON · Statistical Concepts and Methods
BHATTACHARYA and WAYMIRE · Stochastic Processes with Applications
BILLINGSLEY · Convergence of Probability Measures, *Second Edition*
BILLINGSLEY · Probability and Measure, *Third Edition*
BIRKES and DODGE · Alternative Methods of Regression
BLISCHKE AND MURTHY (editors) · Case Studies in Reliability and Maintenance
BLISCHKE AND MURTHY · Reliability: Modeling, Prediction, and Optimization
BLOOMFIELD · Fourier Analysis of Time Series: An Introduction, *Second Edition*
BOLLEN · Structural Equations with Latent Variables
BOROVKOV · Ergodicity and Stability of Stochastic Processes
BOULEAU · Numerical Methods for Stochastic Processes
BOX · Bayesian Inference in Statistical Analysis
BOX · R. A. Fisher, the Life of a Scientist
BOX and DRAPER · Empirical Model-Building and Response Surfaces
*BOX and DRAPER · Evolutionary Operation: A Statistical Method for Process Improvement
BOX, HUNTER, and HUNTER · Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building
BOX and LUCEÑO · Statistical Control by Monitoring and Feedback Adjustment
BRANDIMARTE · Numerical Methods in Finance: A MATLAB-Based Introduction
BROWN and HOLLANDER · Statistics: A Biomedical Introduction
BRUNNER, DOMHOF, and LANGER · Nonparametric Analysis of Longitudinal Data in Factorial Experiments
BUCKLEW · Large Deviation Techniques in Decision, Simulation, and Estimation
CAIROLI and DALANG · Sequential Stochastic Optimization
CHAN · Time Series: Applications to Finance
CHATTERJEE and HADI · Sensitivity Analysis in Linear Regression
CHATTERJEE and PRICE · Regression Analysis by Example, *Third Edition*
CHERNICK · Bootstrap Methods: A Practitioner's Guide
CHERNICK and FRIIS · Introductory Biostatistics for the Health Sciences
CHILÈS and DELFINER · Geostatistics: Modeling Spatial Uncertainty
CHOW and LIU · Design and Analysis of Clinical Trials: Concepts and Methodologies, *Second Edition*
CLARKE and DISNEY · Probability and Random Processes: A First Course with Applications, *Second Edition*
*COCHRAN and COX · Experimental Designs, *Second Edition*
CONGDON · Applied Bayesian Modelling
CONGDON · Bayesian Statistical Modelling
CONOVER · Practical Nonparametric Statistics, *Second Edition*
COOK · Regression Graphics
COOK and WEISBERG · Applied Regression Including Computing and Graphics
COOK and WEISBERG · An Introduction to Regression Graphics
CORNELL · Experiments with Mixtures, Designs, Models, and the Analysis of Mixture Data, *Third Edition*
COVER and THOMAS · Elements of Information Theory

*Now available in a lower priced paperback edition in the Wiley Classics Library.

COX · A Handbook of Introductory Statistical Methods
*COX · Planning of Experiments
CRESSIE · Statistics for Spatial Data, *Revised Edition*
CSÖRGŐ and HORVÁTH · Limit Theorems in Change Point Analysis
DANIEL · Applications of Statistics to Industrial Experimentation
DANIEL · Biostatistics: A Foundation for Analysis in the Health Sciences, *Sixth Edition*
*DANIEL · Fitting Equations to Data: Computer Analysis of Multifactor Data, *Second Edition*
DASU and JOHNSON · Exploratory Data Mining and Data Cleaning
DAVID and NAGARAJA · Order Statistics, *Third Edition*
*DEGROOT, FIENBERG, and KADANE · Statistics and the Law
DEL CASTILLO · Statistical Process Adjustment for Quality Control
DENISON, HOLMES, MALLICK and SMITH · Bayesian Methods for Nonlinear Classification and Regression
DETTE and STUDDEN · The Theory of Canonical Moments with Applications in Statistics, Probability, and Analysis
DEY and MUKERJEE · Fractional Factorial Plans
DILLON and GOLDSTEIN · Multivariate Analysis: Methods and Applications
DODGE · Alternative Methods of Regression
*DODGE and ROMIG · Sampling Inspection Tables, *Second Edition*
*DOOB · Stochastic Processes
DOWDY, WEARDEN, and CHILKO · Statistics for Research, *Third Edition*
DRAPER and SMITH · Applied Regression Analysis, *Third Edition*
DRYDEN and MARDIA · Statistical Shape Analysis
DUDEWICZ and MISHRA · Modern Mathematical Statistics
DUNN and CLARK · Applied Statistics: Analysis of Variance and Regression, *Second Edition*
DUNN and CLARK · Basic Statistics: A Primer for the Biomedical Sciences, *Third Edition*
DUPUIS and ELLIS · A Weak Convergence Approach to the Theory of Large Deviations
*ELANDT-JOHNSON and JOHNSON · Survival Models and Data Analysis
ENDERS · Applied Econometric Time Series
ETHIER and KURTZ · Markov Processes: Characterization and Convergence
EVANS, HASTINGS, and PEACOCK · Statistical Distributions, *Third Edition*
FELLER · An Introduction to Probability Theory and Its Applications, Volume I, *Third Edition,* Revised; Volume II, *Second Edition*
FISHER and VAN BELLE · Biostatistics: A Methodology for the Health Sciences
*FLEISS · The Design and Analysis of Clinical Experiments
FLEISS · Statistical Methods for Rates and Proportions, *Third Edition*
FLEMING and HARRINGTON · Counting Processes and Survival Analysis
FULLER · Introduction to Statistical Time Series, *Second Edition*
FULLER · Measurement Error Models
GALLANT · Nonlinear Statistical Models
GIESBRECHT and GUMPERTZ · Planning, Construction, and Statistical Analysis of Comparative Experiments
GIFI · Nonlinear Multivariate Analysis
GHOSH, MUKHOPADHYAY, and SEN · Sequential Estimation
GIFI · Nonlinear Multivariate Analysis
GLASSERMAN and YAO · Monotone Structure in Discrete-Event Systems
GNANADESIKAN · Methods for Statistical Data Analysis of Multivariate Observations, *Second Edition*
GOLDSTEIN and LEWIS · Assessment: Problems, Development, and Statistical Issues
GREENWOOD and NIKULIN · A Guide to Chi-Squared Testing

*Now available in a lower priced paperback edition in the Wiley Classics Library.

GROSS and HARRIS · Fundamentals of Queueing Theory, *Third Edition*
*HAHN and SHAPIRO · Statistical Models in Engineering
HAHN and MEEKER · Statistical Intervals: A Guide for Practitioners
HALD · A History of Probability and Statistics and their Applications Before 1750
HALD · A History of Mathematical Statistics from 1750 to 1930
HAMPEL · Robust Statistics: The Approach Based on Influence Functions
HANNAN and DEISTLER · The Statistical Theory of Linear Systems
HEIBERGER · Computation for the Analysis of Designed Experiments
HEDAYAT and SINHA · Design and Inference in Finite Population Sampling
HELLER · MACSYMA for Statisticians
HINKELMAN and KEMPTHORNE: · Design and Analysis of Experiments, Volume 1: Introduction to Experimental Design
HOAGLIN, MOSTELLER, and TUKEY · Exploratory Approach to Analysis of Variance
HOAGLIN, MOSTELLER, and TUKEY · Exploring Data Tables, Trends and Shapes
*HOAGLIN, MOSTELLER, and TUKEY · Understanding Robust and Exploratory Data Analysis
HOCHBERG and TAMHANE · Multiple Comparison Procedures
HOCKING · Methods and Applications of Linear Models: Regression and the Analysis of Variance, *Second Edition*
HOEL · Introduction to Mathematical Statistics, *Fifth Edition*
HOGG and KLUGMAN · Loss Distributions
HOLLANDER and WOLFE · Nonparametric Statistical Methods, *Second Edition*
HOSMER and LEMESHOW · Applied Logistic Regression, *Second Edition*
HOSMER and LEMESHOW · Applied Survival Analysis: Regression Modeling of Time to Event Data
HUBER · Robust Statistics
HUBERTY · Applied Discriminant Analysis
HUNT and KENNEDY · Financial Derivatives in Theory and Practice
HUSKOVA, BERAN, and DUPAC · Collected Works of Jaroslav Hajek— with Commentary
IMAN and CONOVER · A Modern Approach to Statistics
JACKSON · A User's Guide to Principle Components
JOHN · Statistical Methods in Engineering and Quality Assurance
JOHNSON · Multivariate Statistical Simulation
JOHNSON and BALAKRISHNAN · Advances in the Theory and Practice of Statistics: A Volume in Honor of Samuel Kotz
JUDGE, GRIFFITHS, HILL, LÜTKEPOHL, and LEE · The Theory and Practice of Econometrics, *Second Edition*
JOHNSON and KOTZ · Distributions in Statistics
JOHNSON and KOTZ (editors) · Leading Personalities in Statistical Sciences: From the Seventeenth Century to the Present
JOHNSON, KOTZ, and BALAKRISHNAN · Continuous Univariate Distributions, Volume 1, *Second Edition*
JOHNSON, KOTZ, and BALAKRISHNAN · Continuous Univariate Distributions, Volume 2, *Second Edition*
JOHNSON, KOTZ, and BALAKRISHNAN · Discrete Multivariate Distributions
JOHNSON, KOTZ, and KEMP · Univariate Discrete Distributions, *Second Edition*
JUREČKOVÁ and SEN · Robust Statistical Procedures: Aymptotics and Interrelations
JUREK and MASON · Operator-Limit Distributions in Probability Theory
KADANE · Bayesian Methods and Ethics in a Clinical Trial Design
KADANE AND SCHUM · A Probabilistic Analysis of the Sacco and Vanzetti Evidence
KALBFLEISCH and PRENTICE · The Statistical Analysis of Failure Time Data, *Second Edition*

*Now available in a lower priced paperback edition in the Wiley Classics Library.

KASS and VOS · Geometrical Foundations of Asymptotic Inference
KAUFMAN and ROUSSEEUW · Finding Groups in Data: An Introduction to Cluster Analysis
KEDEM and FOKIANOS · Regression Models for Time Series Analysis
KENDALL, BARDEN, CARNE, and LE · Shape and Shape Theory
KHURI · Advanced Calculus with Applications in Statistics, *Second Edition*
KHURI, MATHEW, and SINHA · Statistical Tests for Mixed Linear Models
KLEIBER and KOTZ · Statistical Size Distributions in Economics and Actuarial Sciences
KLUGMAN, PANJER, and WILLMOT · Loss Models: From Data to Decisions
KLUGMAN, PANJER, and WILLMOT · Solutions Manual to Accompany Loss Models: From Data to Decisions
KOTZ, BALAKRISHNAN, and JOHNSON · Continuous Multivariate Distributions, Volume 1, *Second Edition*
KOTZ and JOHNSON (editors) · Encyclopedia of Statistical Sciences: Volumes 1 to 9 with Index
KOTZ and JOHNSON (editors) · Encyclopedia of Statistical Sciences: Supplement Volume
KOTZ, READ, and BANKS (editors) · Encyclopedia of Statistical Sciences: Update Volume 1
KOTZ, READ, and BANKS (editors) · Encyclopedia of Statistical Sciences: Update Volume 2
KOVALENKO, KUZNETZOV, and PEGG · Mathematical Theory of Reliability of Time-Dependent Systems with Practical Applications
LACHIN · Biostatistical Methods: The Assessment of Relative Risks
LAD · Operational Subjective Statistical Methods: A Mathematical, Philosophical, and Historical Introduction
LAMPERTI · Probability: A Survey of the Mathematical Theory, *Second Edition*
LANGE, RYAN, BILLARD, BRILLINGER, CONQUEST, and GREENHOUSE · Case Studies in Biometry
LARSON · Introduction to Probability Theory and Statistical Inference, *Third Edition*
LAWLESS · Statistical Models and Methods for Lifetime Data, *Second Edition*
LAWSON · Statistical Methods in Spatial Epidemiology
LE · Applied Categorical Data Analysis
LE · Applied Survival Analysis
LEE and WANG · Statistical Methods for Survival Data Analysis, *Third Edition*
LEPAGE and BILLARD · Exploring the Limits of Bootstrap
LEYLAND and GOLDSTEIN (editors) · Multilevel Modelling of Health Statistics
LIAO · Statistical Group Comparison
LINDVALL · Lectures on the Coupling Method
LINHART and ZUCCHINI · Model Selection
LITTLE and RUBIN · Statistical Analysis with Missing Data, *Second Edition*
LLOYD · The Statistical Analysis of Categorical Data
MAGNUS and NEUDECKER · Matrix Differential Calculus with Applications in Statistics and Econometrics, *Revised Edition*
MALLER and ZHOU · Survival Analysis with Long Term Survivors
MALLOWS · Design, Data, and Analysis by Some Friends of Cuthbert Daniel
MANN, SCHAFER, and SINGPURWALLA · Methods for Statistical Analysis of Reliability and Life Data
MANTON, WOODBURY, and TOLLEY · Statistical Applications Using Fuzzy Sets
MARCHETTE · Random Graphs for Statistical Pattern Recognition
MARDIA and JUPP · Directional Statistics
MASON, GUNST, and HESS · Statistical Design and Analysis of Experiments with Applications to Engineering and Science, *Second Edition*
McCULLOCH and SEARLE · Generalized, Linear, and Mixed Models

*Now available in a lower priced paperback edition in the Wiley Classics Library.

McFADDEN · Management of Data in Clinical Trials
McLACHLAN · Discriminant Analysis and Statistical Pattern Recognition
McLACHLAN and KRISHNAN · The EM Algorithm and Extensions
McLACHLAN and PEEL · Finite Mixture Models
McNEIL · Epidemiological Research Methods
MEEKER and ESCOBAR · Statistical Methods for Reliability Data
MEERSCHAERT and SCHEFFLER · Limit Distributions for Sums of Independent Random Vectors: Heavy Tails in Theory and Practice
*MILLER · Survival Analysis, *Second Edition*
MONTGOMERY, PECK, and VINING · Introduction to Linear Regression Analysis, *Third Edition*
MORGENTHALER and TUKEY · Configural Polysampling: A Route to Practical Robustness
MUIRHEAD · Aspects of Multivariate Statistical Theory
MULLER and STOYAN · Comparison Methods for Stochastic Models and Risks
MURRAY · X-STAT 2.0 Statistical Experimentation, Design Data Analysis, and Nonlinear Optimization
MURTHY, XIE, and JIANG · Weibull Models
MYERS and MONTGOMERY · Response Surface Methodology: Process and Product Optimization Using Designed Experiments, *Second Edition*
MYERS, MONTGOMERY, and VINING · Generalized Linear Models. With Applications in Engineering and the Sciences
NELSON · Accelerated Testing, Statistical Models, Test Plans, and Data Analyses
NELSON · Applied Life Data Analysis
NEWMAN · Biostatistical Methods in Epidemiology
OCHI · Applied Probability and Stochastic Processes in Engineering and Physical Sciences
OKABE, BOOTS, SUGIHARA, and CHIU · Spatial Tesselations: Concepts and Applications of Voronoi Diagrams, *Second Edition*
OLIVER and SMITH · Influence Diagrams, Belief Nets and Decision Analysis
PALTA · Quantitative Methods in Population Health: Extensions of Ordinary Regressions
PANKRATZ · Forecasting with Dynamic Regression Models
PANKRATZ · Forecasting with Univariate Box-Jenkins Models: Concepts and Cases
*PARZEN · Modern Probability Theory and Its Applications
PEÑA, TIAO, and TSAY · A Course in Time Series Analysis
PIANTADOSI · Clinical Trials: A Methodologic Perspective
PORT · Theoretical Probability for Applications
POURAHMADI · Foundations of Time Series Analysis and Prediction Theory
PRESS · Bayesian Statistics: Principles, Models, and Applications
PRESS · Subjective and Objective Bayesian Statistics, *Second Edition*
PRESS and TANUR · The Subjectivity of Scientists and the Bayesian Approach
PUKELSHEIM · Optimal Experimental Design
PURI, VILAPLANA, and WERTZ · New Perspectives in Theoretical and Applied Statistics
PUTERMAN · Markov Decision Processes: Discrete Stochastic Dynamic Programming
*RAO · Linear Statistical Inference and Its Applications, *Second Edition*
RAUSAND and HØYLAND · System Reliability Theory: Models, Statistical Methods, and Applications, *Second Edition*
RENCHER · Linear Models in Statistics
RENCHER · Methods of Multivariate Analysis, *Second Edition*
RENCHER · Multivariate Statistical Inference with Applications
RIPLEY · Spatial Statistics
RIPLEY · Stochastic Simulation
ROBINSON · Practical Strategies for Experimenting

*Now available in a lower priced paperback edition in the Wiley Classics Library.

ROHATGI and SALEH · An Introduction to Probability and Statistics, *Second Edition*
ROLSKI, SCHMIDLI, SCHMIDT, and TEUGELS · Stochastic Processes for Insurance and Finance
ROSENBERGER and LACHIN · Randomization in Clinical Trials: Theory and Practice
ROSS · Introduction to Probability and Statistics for Engineers and Scientists
ROUSSEEUW and LEROY · Robust Regression and Outlier Detection
RUBIN · Multiple Imputation for Nonresponse in Surveys
RUBINSTEIN · Simulation and the Monte Carlo Method
RUBINSTEIN and MELAMED · Modern Simulation and Modeling
RYAN · Modern Regression Methods
RYAN · Statistical Methods for Quality Improvement, *Second Edition*
SALTELLI, CHAN, and SCOTT (editors) · Sensitivity Analysis
*SCHEFFE · The Analysis of Variance
SCHIMEK · Smoothing and Regression: Approaches, Computation, and Application
SCHOTT · Matrix Analysis for Statistics
SCHOUTENS · Levy Processes in Finance: Pricing Financial Derivatives
SCHUSS · Theory and Applications of Stochastic Differential Equations
SCOTT · Multivariate Density Estimation: Theory, Practice, and Visualization
*SEARLE · Linear Models
SEARLE · Linear Models for Unbalanced Data
SEARLE · Matrix Algebra Useful for Statistics
SEARLE, CASELLA, and McCULLOCH · Variance Components
SEARLE and WILLETT · Matrix Algebra for Applied Economics
SEBER and LEE · Linear Regression Analysis, *Second Edition*
SEBER · Multivariate Observations
SEBER and WILD · Nonlinear Regression
SENNOTT · Stochastic Dynamic Programming and the Control of Queueing Systems
*SERFLING · Approximation Theorems of Mathematical Statistics
SHAFER and VOVK · Probability and Finance: It's Only a Game!
SMALL and McLEISH · Hilbert Space Methods in Probability and Statistical Inference
SRIVASTAVA · Methods of Multivariate Statistics
STAPLETON · Linear Statistical Models
STAUDTE and SHEATHER · Robust Estimation and Testing
STOYAN, KENDALL, and MECKE · Stochastic Geometry and Its Applications, *Second Edition*
STOYAN and STOYAN · Fractals, Random Shapes and Point Fields: Methods of Geometrical Statistics
STYAN · The Collected Papers of T. W. Anderson: 1943–1985
SUTTON, ABRAMS, JONES, SHELDON, and SONG · Methods for Meta-Analysis in Medical Research
TANAKA · Time Series Analysis: Nonstationary and Noninvertible Distribution Theory
THOMPSON · Empirical Model Building
THOMPSON · Sampling, *Second Edition*
THOMPSON · Simulation: A Modeler's Approach
THOMPSON and SEBER · Adaptive Sampling
THOMPSON, WILLIAMS, and FINDLAY · Models for Investors in Real World Markets
TIAO, BISGAARD, HILL, PEÑA, and STIGLER (editors) · Box on Quality and Discovery: with Design, Control, and Robustness
TIERNEY · LISP-STAT: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics
TSAY · Analysis of Financial Time Series
UPTON and FINGLETON · Spatial Data Analysis by Example, Volume II: Categorical and Directional Data
VAN BELLE · Statistical Rules of Thumb
VESTRUP · The Theory of Measures and Integration

*Now available in a lower priced paperback edition in the Wiley Classics Library.

VIDAKOVIC · Statistical Modeling by Wavelets
WEISBERG · Applied Linear Regression, *Second Edition*
WELSH · Aspects of Statistical Inference
WESTFALL and YOUNG · Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment
WHITTAKER · Graphical Models in Applied Multivariate Statistics
WINKER · Optimization Heuristics in Economics: Applications of Threshold Accepting
WONNACOTT and WONNACOTT · Econometrics, *Second Edition*
WOODING · Planning Pharmaceutical Clinical Trials: Basic Statistical Principles
WOOLSON and CLARKE · Statistical Methods for the Analysis of Biomedical Data, *Second Edition*
WU and HAMADA · Experiments: Planning, Analysis, and Parameter Design Optimization
YANG · The Construction Theory of Denumerable Markov Processes
*ZELLNER · An Introduction to Bayesian Inference in Econometrics
ZHOU, OBUCHOWSKI, and McCLISH · Statistical Methods in Diagnostic Medicine

*Now available in a lower priced paperback edition in the Wiley Classics Library.