

# A Statistical Perspective on Data Mining

Ranjan Maitra\*

## Abstract

Technological advances have led to new and automated data collection methods. Datasets once at a premium are often plentiful nowadays and sometimes indeed massive. A new breed of challenges are thus presented – primary among them is the need for methodology to analyze such masses of data with a view to understanding complex phenomena and relationships. Such capability is provided by data mining which combines core statistical techniques with those from machine intelligence. This article reviews the current state of the discipline from a statistician’s perspective, illustrates issues with real-life examples, discusses the connections with statistics, the differences, the failings and the challenges ahead.

## 1 Introduction

The information age has been matched by an explosion of data. This surfeit has been a result of modern, improved and, in many cases, automated methods for both data collection and storage. For instance, many stores tag their items with a product-specific bar code, which is scanned in when the corresponding item is bought. This automatically creates a gigantic repository of information on products and product combinations sold. Similar databases are also created by automated book-keeping, digital communication tools or by remote sensing satellites, and aided by the availability of affordable and effective storage mechanisms – magnetic tapes, data warehouses and so on. This has created a situation of plentiful data and the potential for new and deeper understanding of complex phenomena. The very size of these databases however means that any signal or pattern may be overshadowed by “noise”. New methodology for the careful analysis of such datasets is therefore called for.

Consider for instance the database created by the scanning of product bar codes at sales checkouts. Originally adopted for reasons of convenience, this now forms the basis for gigantic databases as large stores maintain records of products bought by customers in any

---

\*Ranjan Maitra is Assistant Professor of Statistics in the Department of Mathematics and Statistics at the University of Maryland, Baltimore County, Baltimore, MD 21250, USA. The author thanks Surajit Chaudhuri for discussions on the practical aspects of data mining from the point of view of a researcher in databases and for help with Figure 4, Rouben Rostamian for providing me with the enrolment data of Table 1 and Devasis Bassu for help with the example in Section 6 of this paper.

transaction. Some businesses have gone further: by providing customers with an incentive to use a magnetic-striped frequent shopper card, they have created a database not just of product combinations but also time-sequenced information on such transactions. The goal behind collecting such data is the ability to answer questions such as “If potato chips and ketchup are purchased together, what is the item that is most likely to be also bought?”, or “If shampoo is purchased, what is the most common item also bought in that same transaction?”. Answers to such questions result in what are called *association rules*. Such rules can be used, for instance, in deciding on store layout or on promotions of certain brands of products by offering discounts on select combinations. Applications of association rules transcend sales transactions data — indeed, I illustrate the concepts in Section 2 through a small-scale class-scheduling problem in my home department — but with millions of daily transactions on millions of products, that application best represents the complexities and challenges in deriving meaningful and useful association rules and is part of folklore.

An oft-stated goal of data mining is the discovery of patterns and relationships among different variables in the database. This is no different from some of the goals of statistical inference: consider for instance, simple linear regression. Similarly, the pair-wise relationship between the products sold above can be nicely represented by means of an undirected weighted graph, with products as the nodes and weighted edges for the presence of the particular product pair in as many transactions as proportional to the weights. While undirected graphs provide a graphical display, directed acyclic graphs are perhaps more interesting — they provide understanding of the phenomena driving the relationships between the variables. The nature of these relationships can be analyzed using classical and modern statistical tools such as regression, neural networks and so on. Section 3 illustrates this concept. Closely related to this notion of knowledge discovery is that of causal dependence models which can be studied using Bayesian belief networks. Heckerman (1996) introduces an example of a car that does not start and proceeds to develop the goal of finding the most likely cause for the malfunction as an illustration for this concept. The building blocks are elementary statistical tools such as Bayes’ theorem and conditional probability statements, but as we shall see in Section 3, the use of these concepts to make a pass at explaining causality is unique. Once again, the problem becomes more acute with large numbers of variables as in many complex systems or processes.

Another aspect of knowledge discovery is *supervised learning*. Statistical tools such as discriminant analysis or classification trees often need to be refined for these problems. Some additional methods to be investigated here are  $k$ -nearest neighbor methods, bootstrap aggregation or *bagging*, and *boosting* which originally evolved in the machine learning literature, but whose statistical properties have been analyzed in recent years by statisticians. Boosting is particularly useful in the context of *data streams* — when we have rapid data flowing into the system and real-time classification rules are needed. Such capability is especially desirable in the context of financial data, to guard against credit card and calling card fraud, when transactions are streaming in from several sources and an automated split-second determination of fraudulent or genuine use has to be made, based on past experience. Modern classification tools such as these are surveyed in Section 4.

Another important aspect of knowledge discovery is *unsupervised learning* or clustering, which is the categorization of the observations in a dataset into an *a priori* unknown number of groups, based on some characteristic of the observations. This is a very difficult problem, and is only compounded when the database is massive. Hierarchical clustering, probability-based methods, as well as optimization partitioning algorithms are all difficult to apply here. Maitra (2001) develops, under restrictive Gaussian equal-dispersion assumptions, a multi-pass scheme which clusters an initial sample, filters out observations that can be reasonably classified by these clusters, and iterates the above procedure on the remainder. This method is *scalable*, which means that it can be used on datasets of any size. This approach, along with several unsurmounted challenges, are reviewed in detail in Section 5.

Finally, we address the issue of text retrieval. With its ready accessibility, the World Wide Web is a treasure-trove of information. An user wishing to obtain documents on a particular topic can do so by typing the word using one of the public-domain search engines. However when an user types the word “car” he likely wants not just all documents containing the word “car” but also relevant documents including ones on “automobile”. In Section 6, we discuss a technique, similar to dimension reduction which makes this possible. Here also, the problem is compounded by databases that are gigantic in size.

Note therefore, that a large number of the goals of data mining overlap with those in statistics. Indeed, in some cases, they are exactly the same and sound statistical solutions exist, but are often computationally impractical to implement. In the sequel, I review some of the facets of the connection between data mining and statistics as indicated above. I illustrate each aspect through a real-life example, discuss current methodology and outline suggested solutions. Except for part of the clustering example — where I used some of my own pre-written software routines in C and Fortran — I was able to use the publicly available and free statistical software package **R** available at <http://www.r-project.org/> to perform necessary data analysis. While not exhaustive, I hope that this article provides a broad review of the emerging area from a statistical view-point and spurs interest in the many unsolved or unsatisfactorily-solved problems in these areas.

## 2 Association Rules

Association rules (Piatetsky-Shapiro, 1991) are statements of the form, “93% of customers who purchase paint also buy paint brushes.” The importance of such a rule from a commercial viewpoint is that a store selling paint will also stock paint brushes, for customer convenience, translating into more buyer visits and and store sales. Further, he can offer promotions on brands with higher profit margins to customers buying paint and subtly dictate customer preference. Such information can additionally be used to decide on storage layout and scheduled dispatch from the warehouse.

Transactions data are used to derive rules such as the one mentioned above. Formally, let  $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n\}$  be the transactions database, where each transaction  $\mathcal{T}_j$  is a member of the set of items in  $\mathcal{I} = \{i_1, i_2, \dots, i_m\}$ . (The transactions data may, for instance, be represented as an  $n \times m$  matrix of 0’s and 1’s called  $\mathcal{D}$  with the  $(k, l)$ ’th entry as an indicator

variable for the presence of item  $i_k$  in transaction  $\mathcal{T}_i$ .) Further, write  $\mathcal{X} \subseteq \mathcal{T}_i$  to denote that a set of items  $\mathcal{X}$  is contained in  $\mathcal{T}_i$ . An association rule is a statement of the form  $\mathcal{X} \Rightarrow \mathcal{Y}$  where  $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{I}$  and  $\mathcal{X}$  and  $\mathcal{Y}$  are disjoint sets. The rule  $\mathcal{X} \Rightarrow \mathcal{Y}$  has confidence  $\gamma$  if  $100\gamma\%$  of the transactions in the database  $\mathcal{D}$  containing  $\mathcal{X}$  also contain the set of items  $\mathcal{Y}$ . It is said to have support  $\delta$  if  $100\delta\%$  of the transactions in  $\mathcal{D}$  contain all elements of the set  $\mathcal{X} \cup \mathcal{Y}$ . The *improvement*  $\iota$  of a rule  $\mathcal{X} \Rightarrow \mathcal{Y}$  is the ratio of the confidence and the proportion of transactions only involving  $\mathcal{Y}$ . Note that while *confidence* provides us with a measure of how confident we are in the given rule, *support* assesses the proportion of transactions on which the rule is based. In general, rules with greater support are more desirable from a business perspective, though rules with lower support often represent small niches who, because of their size are not necessarily wooed by other businesses and may be very loyal customers. Further, improvement compares how much better a rule  $\mathcal{X} \Rightarrow \mathcal{Y}$  is at predicting  $\mathcal{Y}$  than in using no conditional information in deciding on  $\mathcal{Y}$ . When improvement is greater than unity, it is better to predict  $\mathcal{Y}$  using  $\mathcal{X}$  than to simply use the marginal probability distribution of  $\mathcal{Y}$ . On the other hand, if the improvement of  $\mathcal{X} \Rightarrow \mathcal{Y}$  is less than unity, then it is better to predict  $\mathcal{Y}$  using the marginal probability distribution than in using it distribution conditional on  $\mathcal{X}$ . At equality, there is no difference in using either  $\mathcal{X}$  for predicting  $\mathcal{Y}$  or in simply deciding on  $\mathcal{Y}$  by chance. Although, association rules have been primarily applied to sales data, they are illustrated here in the context of scheduling mathematics and statistics graduate classes.

## 2.1 An Application

Like several departments, the Department of Mathematics and Statistics at the University of Maryland, Baltimore County offers several graduate courses in mathematics and statistics. Most students in these classes pursue Master of Science (M. S.) and doctoral (Ph. D.) degrees offered by the department. Scheduling these classes has always been an onerous task. Practical issues do not permit all classes to be scheduled at different times, and indeed, four particular time-slots are most desirable for students. Additionally, it is desirable for commuting and part-time students to have their classes scheduled close to one another. Hitherto, class times have been decided on the basis of empirical perceptions of the scheduling authorities. Deciding on obvious candidates for concurrently scheduled classes is sometimes rather straightforward. For instance, students should have a proper grasp of the elementary probability course (Stat 651) to be ready for the higher-level course (Stat 611) so that these can run simultaneously. Such rules are not always obvious however. Further, the other question – which classes to schedule close to each other – is not always easy to determine. Here I use association rules to suggest classes that should not be offered concurrently, as well as classes that may be scheduled close to each other to maximize student convenience. The database on the enrollment of graduate students in the fall semester of 2001 is small with 11 items (the classes offered) and 38 transactions (students' enrollment in the different classes) – see Table 1. Association rules were built using the above – five rules with the highest support (for both one- and two-class conditioning rules) are reported in Table 2. Since each

Table 1: Enrollment data of departmental graduate students in mathematics and statistics at the University of Maryland, Baltimore County, Fall semester, 2001.

Course	Class list by student's last name
Math 600	Gavrea, Korostyshevskaya, Lu, Muscedere, Shevchenko, Soane, Vdovina
Math 617	Chaillou, Gavrea, Korostyshevskaya, Shevchenko, Vdovina, Waldt
Math 630	Du, Feldman, Foster, Gavrea, Korostyshevskaya, Shevchenko, Singh, Smith, Soane, Taff, Vdovina, Waldt
Math 700	Chaillou, Hanhart, He, Sabaka, Soane, Tymofyeyev, Wang, Webster, Zhu
Math 710	Korolev, Korostyshevskaya, Macura, Osmoukhina
Stat 601	Cameron, Li, Paul, Pone, Siddani, Zhang
Stat 611	Cameron, Li, Liu, Siddani, Zhang
Stat 615	Benamati, Kelly, Li, Liu, Pone, Siddani, Wang, Webb, Zhang
Stat 621	Hang, Osmoukhina, Tymofyeyev
Stat 651	Waldt, Wang
Stat 710	Hang, Liu, Paul, Pone, Safronov

student enrolls in at most three classes, note that all conditioning rules involve only up to two classes. To suggest a class schedule using the above, it is fairly clear that Math 600, Math 617 and Math 630 should be held at different time-slots but close to each other. Note that while the rule “If Math 600 and Math 617, then Math 630” has confidence 100%, both “(Math 600 & Math 630) $\Rightarrow$  Math 617” and “(Math 617 & Math 630) $\Rightarrow$  Math 600” have confidence of 80%. Based on this, it is preferable to suggest an ordering of class times such that the Math 630 time-slot is in between those of Math 600 and Math 617. Similar rules are used to suggest the schedule in Table 3 for future semesters that have a similar mix of classes. The suggested schedule requires at most four different time-slots, while the current schedule derived empirically and ignoring enrollment data has no less than eight time-slots.

Table 2: Some association rules obtained from Table 1 and their confidence ( $\gamma$ ), support ( $\delta$ ) and improvement ( $\iota$ ). Only single- and double-class conditioning rules with the five highest improvement measures are reported here.

Rule	$\gamma$	$\delta$	$\iota$	Rule	$\gamma$	$\delta$	$\iota$
Stat 601 $\Rightarrow$ Stat 611	0.67	0.11	5.07	(Math 700 & Stat 615) $\Rightarrow$ Stat 651	1.00	0.03	19.00
Stat 611 $\Rightarrow$ Stat 601	0.67	0.11	5.07	(Math 630 & Stat 651) $\Rightarrow$ Math 617	1.00	0.03	6.33
Math 600 $\Rightarrow$ Math 617	0.57	0.10	3.62	(Math 600 & Math 630) $\Rightarrow$ Math 617	0.80	0.10	5.07
Math 617 $\Rightarrow$ Math 600	0.67	0.10	3.62	(Stat 611 & Stat 615) $\Rightarrow$ Stat 601	0.75	0.08	4.75
Stat 611 $\Rightarrow$ Stat 615	0.80	0.11	3.38	(Math 617 & Math 630) $\Rightarrow$ Math 600	0.80	0.10	4.34

Table 3: Suggested schedule for graduate classes in mathematics and statistics at the University of Maryland, Baltimore County based on the association rules obtained in Table 2.

Slot	Classes	Slot	Classes	Slot	Classes
1	Stat 621	5	Math 600	1	Math 600, Stat 611
2	Stat 651, Stat 611	6	Math 617	2	Math 630, Math 700, Stat 601, Math 710
3	Math 710	7	Math 700, Stat 601	3	Math 617, Stat 615, Stat 621
4	Stat 615	8	Math 630, Stat 710	4	Stat 651, Stat 710

The above is an interesting and somewhat straightforward illustration of association rules. It is made easier by the limitation that all graduate students register for at most three classes. For instance, the consequences of searching for such rules for all undergraduate and graduate classes taught at the university can well be imagined. There are hundreds of such classes and over 12,000 students, each of who take any number between one and eight classes a semester. Obtaining association rules from such large databases is quite daunting. This problem of course, pales in comparison to the search for such rules in sales databases, where there are several millions of items and hundreds of millions of transactions.

## 2.2 Application to Large Databases

There has been substantial work on finding association rules in large databases. Agarwal *et al.* (1993) focus on discovering association rules of sufficient support and confidence. The basic idea is to first specify large item-sets, described by those sets of items that occur have a pre-specified transaction support (*i. e.* proportion of transactions in which they occur together) is greater than the desired minimum support  $\delta_+$ . By restricting attention to these item-sets, one can develop rules  $X \Rightarrow Y$  of confidence given by the ratio of the support of  $X \cup Y$  to the support of  $X$ . This rule holds if and only if this ratio is greater than the minimum desired confidence  $\gamma_+$ . It may be noted that since  $X \cup Y$  is a large item-set, the given rule  $X \Rightarrow Y$  trivially has pre-specified minimum support  $\delta_+$ .

The algorithms for discovering all large item-sets in a transactions database are iterative in spirit. The idea is to make multiple passes over the transactions database starting with a *seed* set of large item-sets and use this to generate candidate item-sets for consideration. The support of each of these candidate item-sets is also calculated at each pass. The candidate item-sets satisfying our definition of large item-sets at each pass form the seed for the next pass, and the process continues till convergence of the item-sets, which is defined as the stage when no new candidate item-set is found. The idea is conceptually very simple and hinges on the fact that an item-set of  $j$  elements can be large only if every subset of it is large. Consequently, it is enough for a multi-pass algorithm to build the rules incrementally in terms of the cardinality of the large item-sets. For instance, if at any stage, there are  $k$  large item-sets of  $j$  elements each, then the only possible large item-sets with more than  $j$  elements pertain to transactions which have members from these  $k$  item-sets. So, the search-space for large item-sets can be considerably whittled down, even in the context of huge databases.

This forms the basis of algorithms in the database literature.

## 2.3 Additional Issues

The solutions for large item-sets ignore transactions with small support. In some cases, these represent niches which may be worth catering to, especially from a business point of view. Reducing  $\gamma_+$  and  $\delta_+$  would allow for these rules but result in a large number of spurious association rules – defeating the very purpose of data mining. Brin *et al.* (1997a, 1997b) therefore develops the notion of both correlation and implication rules. Instead of confidence measures, their rules have implication strengths ranging from 0 to  $\infty$  – an implication strength of 1 means that the rule is as useful as under the framework of statistical independence. An implication strength greater than 1 indicates a greater than expected presence, under statistical independence, of the itemset. Another notion (Aggarwal and Yu, 1998) called *collective strength* of an itemset  $\mathcal{I}$  involves defining a *violation rate*  $v(\mathcal{I})$  as the fraction of *violations* (transactions in which only some members of an itemset are present) of the itemset over all transactions. The collective strength is then defined as  $\mathcal{C}(I) = \{\{1 - v(\mathcal{I})\} / \{1 - \mathbb{E}v(\mathcal{I})\}\} \cdot \{\{\mathbb{E}v(\mathcal{I})\} \{\mathbb{E}v(\mathcal{I})\}\}$  where  $\mathbb{E}$  denotes expected value under the assumption of statistical independence. This measure regards both absence and presence of item combinations in an itemset symmetrically so that it can be used if the absence of certain item combinations rather than their presence is of interest. It additionally incorporates the correlation measure because for perfectly positively correlated items,  $\mathcal{C}(I) = \infty$  while for perfectly negatively correlated items,  $\mathcal{C}(I) = 0$ . Finally, the collective strength uses the relative occurrences of an itemset in the database: itemsets with insignificant support can then be pruned off at a later stage. DuMouchel and Pregibon (2001) point out however, that the above advantages accrue at considerable loss of interpretability. They propose a measure of “interestingness” for all itemsets and fit an empirical Bayes model to the itemset counts. All lower 95% credible limits for the interestingness measure of an itemset are ranked. This approach has the added advantage of discovering complex mechanisms among multi-item associations that are more frequent than their corresponding pairwise associations suggest.

One of the issues in a multi-pass algorithm as of Aggarwal *et al.* (1993) is that it is not suited for data streams such as web applications. Understanding the structure and nature of such transactions is important and to this extent, some work has been done recently (Babcock *et al.*, 2002). Moreover, none of the above formulations use any time series information, which is very often present in the database. For example, a customer may purchase tea and sugar separately in successive transactions, presumably failing to recall that he needed to purchase the second product also. Such associations would not show up above, and indeed it would perhaps be desirable for the store to provide a friendly reminder to such customers. Additionally, no consideration is given to multiple taxonomies (or hierarchies) in postulating these rules. For example, a refrigerator is a kitchen appliance is a heavy electrical appliance. This is a taxonomy. Given such a taxonomy, it may be possible to infer a rule that “people who buy kitchen appliances tend also to buy dishware”. This rule may hold even if rules such as “people who buy refrigerators also tend to buy dishware” or “people who buy kitchen

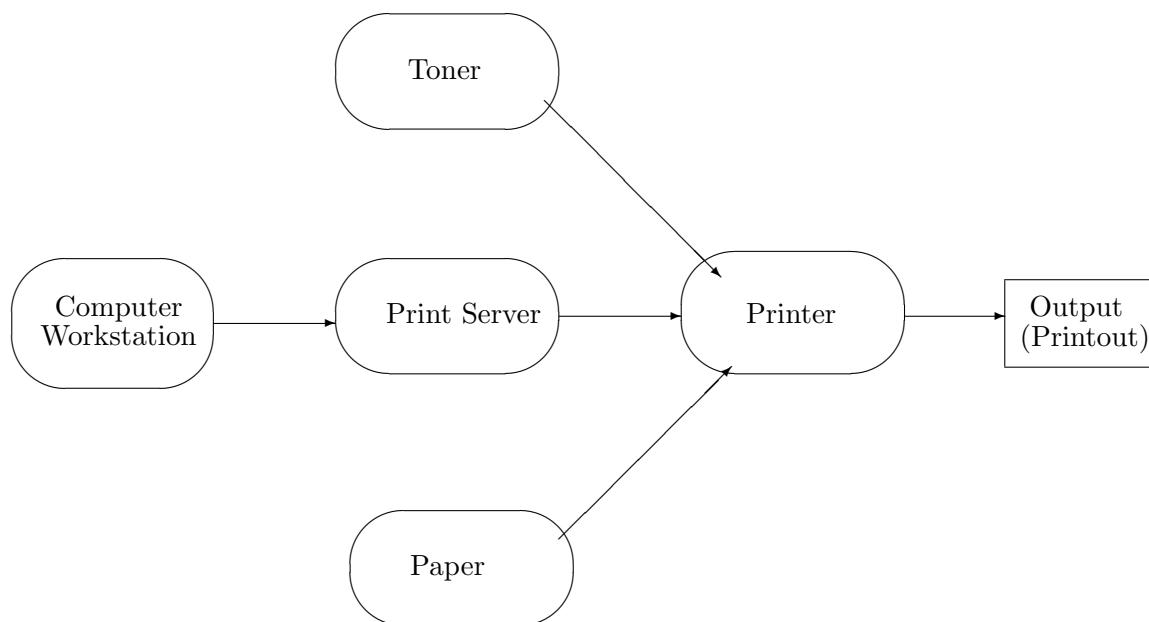


Figure 1: A directed graph representation for the output from a laser printer. Directed edges are drawn in the direction of cause to effect.

appliances also tend to buy dishware” do not hold. It would be useful to construct rules using such hierarchies. These are some of the aspects of association rules requiring attention of both the statistical and the database community.

### 3 Graphical Models and Belief Networks

The pair-wise associations from the enrollment data in Table 1 can be represented using an undirected graph with nodes representing the classes. Weights on the edges between a pair of nodes would indicate the frequencies with which the two classes are taken by the same student. Graphs are frequently used to provide pictorial representations of joint pairwise relationships between variables. From a statistical point of view they additionally help us to learn and specify the joint multivariate distribution of the variables. For suppose we are interested in specifying the joint distribution of  $p$  variables  $X_1, X_2, \dots, X_p$ . The simplest approach is to specify that the joint distribution of the variables is the product of the marginals of each, *i. e.* the variables are all independent. In many complex phenomena and real-life situations, such an assumption is not tenable. On the other hand, multivariate distributions are difficult to specify and subsequent inference is often intractable, especially when  $p$  is large. Representing the joint distribution through an appropriate sequence of marginal and conditional distributions alleviates the problem somewhat. I provide an illustration through a simplified day-to-day example, also graphically represented in Figure 1.

When a print job is submitted from a desktop computer/workstation, control of the job is transferred to the print server which queues it appropriately and sends it to the

printing device. The printer needs toner and paper to print its output. The output may be represented by a variable which may be binary or a continuous variable measuring the degree of satisfaction with the result. For instance, 0% may indicate no output, 100% a perfect result, while 25% may mean faded output as a consequence of an over-used toner. The variables here are the binary variable  $X_1$  representing the correctness of the print command,  $X_2$  for the number of pages transmitted from the print server to the printer,  $X_3$  denoting the quality of the toner,  $X_4$  for the number of pages in the printer paper tray and  $X_5$  indicating the status of the printer (off-line, on-line or jammed) while interacting with the toner, the print server and the paper tray. The output variable is given by  $X_6$ . The distribution of  $X_1, X_2, \dots, X_6$  can be represented by means of the directed graph in Figure 1. The arrows indicate conditional dependence of the descendants to their parents. For instance,  $X_2$  is dependent on  $X_1$ , while  $X_5$  is dependent on its *antecedents*  $X_1, X_2, \dots, X_4$  only through its *parents*  $X_2, X_3, X_4$ . Finally  $X_6$  is dependent on its antecedents  $X_1, X_2, \dots, X_5$  only through  $X_5$ .

The graph of Figure 1 indicates that the joint distribution of  $X_1, X_2, \dots, X_6$  is:

$$\Pr(X_1, X_2, \dots, X_6) = \Pr(X_1)\Pr(X_2 | X_1)\Pr(X_3)\Pr(X_4)\Pr(X_5 | X_2, X_3, X_4)\Pr(X_6 | X_5). \quad (1)$$

Note that conditional and marginal probability distributions may have additional parameters: for instance, both paper and toner may be modeled in terms of the number of pages sent through the print server and the date last replenished. Now suppose that given  $X_6 = 0$ , we want to diagnose the possible cause. If all the probability distributions were completely specified, by using Bayes' Theorem and conditional probability statements one can calculate the probabilities of the other components malfunctioning, given  $X_6 = 0$  and use this to identify the most likely causes, together with a statement on the reliability of our assessments. The graphical structure described above is a *Bayesian belief network*.

In reality of course, we would not be handed down a complete specification of the conditional and marginal probability distributions, even with a completely known network. I address the alternative situation a little later but for now address the case for a scenario with knowledge of network structure such as outlined, but unknown probabilities. Data on the network (such as several observations on the working of the printer in the example above) are used to learn the different probabilities). For each variable  $X_i$ , let  $\Delta_i$  represent the set of its antecedents, and  $\theta_{i\omega}(x_i)$  be the conditional density of  $X_i$  at the point  $x_i$  given that  $\Delta_i = \omega$ . For discrete statespaces, one can put Dirichlet distributions on each probability while for continuous statespaces a Dirichlet Process prior may be used. Assuming *parameter independence* *i. e.* independence of the parameters underlying the unknown probabilities, the probability distributions are updated with newer data. I refer to Heckerman (1996) for a detailed description of the related mechanics but note that uncertainty in our knowledge of the probability distributions governing the network is addressed through these priors.

The use of graphical and conditional structures to specify joint distributions has a long history in statistics. The simplest case is that of a Markov Chain. If  $X_1, X_2, \dots, X_n$  are successive realizations of a first-order Markov Chain the dependence structure may be represented using the directed graph  $X_1 \rightarrow X_2 \rightarrow \dots \rightarrow X_n$ . Note that the above structure

implies that the conditional distribution of any interior  $X_i$  given both the past and the future involves only its immediate antecedent  $X_{i-1}$  and its immediate descendant  $X_{i+1}$ . In other words the conditional distribution of  $X_i$  given  $X_{i-1}$  and  $X_{i+1}$  is independent of the other variables. Thus an equivalent structure may be specified by an undirected acyclic graph with the only edges involving  $X_i$  being those connecting it to  $X_{i-1}$  and  $X_{i+1}$  for all  $i = 2, 3, \dots, n-1$ .  $X_1$  and  $X_n$  have only one connecting edge, to  $X_2$  and  $X_{n-1}$  respectively. The undirected graph representation is important because a Markov Chain running backwards and therefore having the same directed graph  $X_1 \leftarrow X_2 \leftarrow \dots \leftarrow X_n$  has the same conditional dependence structure as the one above.

Specifying joint distributions through conditional distributions is fairly common and well-established in statistics. For instance, it can be shown that under conditions of positivity for the statespace (which may be relaxed somewhat), complete knowledge of the local characteristics or the full conditional distributions  $\Pr(X_i | X_j : j \neq i, j = 1, 2, \dots, n)$  is enough to specify the joint distribution of  $X_1, X_2, \dots, X_n$  (Brook, 1964; Besag, 1974). Indeed a system specified through its conditional probability structure is called a Markov Random Field (MRF) and is commonly used in statistical physics, Bayesian spatial statistics, to name a few areas. Of course, a conditional independence model as described above using a graph is handy here because the local characteristic at a particular  $X_i$  is dependent only on those  $X_j$ 's which share a common edge with  $X_i$ . This property is called the *locally Markov property*. Those  $X_j$ 's sharing an edge with  $X_i$  in the graphical representation of dependence are called neighbors of the latter. Defining the *environment* of a set of random variables to be the union of neighbors of each element in the set, a *globally Markov property* is defined to be the case when the conditional distribution of the subset of random variables given everything else depends only through the environment. Under conditions of positivity, the *locally Markov property* is equivalent to the *globally Markov property*. Further, if we define a *cliquo* to be a set consisting either of a single random variable or a set of random variables all of which are neighbors to each other and a *clique* to be a maximal cliquo (such that there is no superset of the cliquo which is also a cliquo), it can be shown (Hammersley and Clifford, 1971) that the underlying probability density/mass function can be decomposed into a product of functions, each defined on a separate clique.

In most cases, it is impractical to know causal relationships between the different components of the network. Indeed, learning the underlying causal structure is desirable and a much-sought goal of the knowledge discovery process. This can be done by extending the methodology from the case where the network is completely known but the governing probability distributions are not. We specify a prior probability on each possible network and then for each network, proceed as above. The posterior probability of each network is marginalized over the parameter probability distributions and the *maximum a posteriori* (MAP) estimate computed. Note that the computational burden is often severe so that stochastic methods such as Markov Chain Monte Carlo (MCMC) are used in computation. Markov graphs are particularly amenable to such methods. Also, I refer back to the discussion in the previous paragraph and mention that in the terminology of Verma and Pearl (1990), two network structures are equivalent if and only if they have the same structure, ignoring arc

directions and the same  $v$ -structures. (A  $v$ -structure is an ordered triple  $(x, y, z)$  with arcs between  $x$  to  $y$  and  $z$  to  $y$ , but no arc between  $x$  and  $z$ .) Equivalent network structures have the same probability and therefore, our learning methods actually learn about equivalence classes in the network. Even with these simplifications, the number of networks to evaluate grows exponentially with the number of nodes  $n$ . Of interest therefore is to see whether a few network-structures can be used in the learning procedure. Some researchers have shown that even a single “good” network structure often provides an excellent approximation to several networks (Cooper and Herscovits, 1992; Aliferis and Cooper, 1994, Heckerman *et al.*, 1995). The focus then is on how to identify a few of these “good” networks. One approach is to score networks in terms of the goodness of fit and rank them — available methods include Madigan and Raftery’s (1994) Occam’s Razor, Risannen’s (1987) Minimum Description Length (MDL), the more traditional Akaike’s (1974) An Information Criterion (AIC) or Schwarz’s (1978) Bayes Information Criterion (BIC). Finally, consider the issue of learning new variables in the network whose existence and relevance is not known *a priori* – they are therefore *hidden*. Such variables (both their number and statespace) can be incorporated in the prior model during the network-learning process. I refer to Cheeseman and Stutz (1996) for a detailed description of a network with hidden variables. Finally, belief networks have generated a lot of interest in the knowledge discovery community – a few references in this regard are Charniak (1991), Spiegelhalter *et al.* (1993), Heckerman *et al.* (1995), Lauritzen (1982), Verma and Pearl (1990), Frydenberg (1990), Buntine (1994, 1996). I conclude this discussion of graphical models and belief networks by noting that most of the network-learning methods discussed here are computer- and memory-intensive, especially in the case of very complex processes with lots of variables. Developing methodology in this context is therefore desirable and useful.

## 4 Classification and Supervised Learning

Supervised learning or classification has long been an important area in statistics, and can arise with regard to a number of applications. For instance, a bank may like to identify long-term depositors by their attributes and provide them with incentives for conducting business at that bank. Other applications also exist: here we discuss a case study in the emerging area of bioinformatics.

### 4.1 Protein Localization

Identifying protein localization sites is an important early step for finding remedies (Nakai and Kanehisa, 1991). The *E. coli* bacteria, has eight such sites: “cp” or the cytoplasm, “im” or the inner membrane without signal sequence, “pp” or periplasm, “imU” or inner membrane with an uncleavable signal sequence, “om” or the outer membrane, “omL” or the outer membrane lipoprotein, “imL” or the inner membrane lipoprotein, “imS” or the inner membrane with a cleavable signal sequence. Each protein sequence has a number of numeric attributes: these are “mcg” or measurements obtained using McGeoch’s method

for signal sequence recognition, “gvh” or measurements via von Heijne’s method for signal sequence recognition, “acd” or the score of a discriminant analysis of the amino acid content of outer membrane and periplasmic proteins, “alm1” or the score of the ALOM membrane spanning region prediction program, and “alm2” or the score using ALOM program after excluding putative cleavable signal regions from the sequence. In addition, there are two binary attributes – “lip” or the von Heijne’s Signal Peptidase II consensus sequence score and the “chg” indicating the presence of a charge on the N-terminus of predicted lipoproteins. (See Horton and Nakai, 1996 and the references therein for a detailed description of these attributes.) Data on 336 such protein sequences for *E. coli* are available for analysis at the University of California Irvine’s Machine Learning Repository (Horton and Nakai, 1996). The “lip” and “chg” are binary attributes. These attributes are ignored for classification because some of the methods discussed here do not admit binary attributes. As a consequence, protein sequences from the sites “imL” and “imS” can no longer be distinguished from each other. Sequences from these sites are also eliminated. Two cases from the site “omL” are discarded because the number of cases is too few for quadratic discriminant analysis, leaving for analysis a subset of 324 sequences from five protein localization sites. Further, a set of 162 random sequences was kept aside as a test set — the remaining 162 formed the training data.

## 4.2 Decision-Theoretic Approaches

The dataset is in the form  $\{(\mathbf{X}_i, \mathcal{G}_i): i=1, 2, \dots, n\}$ , where  $1 \leq \mathcal{G}_i \leq K$  represents the group to which the  $i$ ’th observation  $\mathbf{X}_i$  belongs. The density of an arbitrary observation  $\mathbf{X}$  evaluated at a point  $\mathbf{x}$  may be written as  $f(\mathbf{x}) = \sum_{k=1}^K \pi_k f_k(\mathbf{x})$ , where  $\pi_k$  is the prior probability that an observation belongs to the  $k$ ’th group, and  $f_k(\cdot)$  is the density of an observation from the  $k$ ’th group. Our goal is to specify the group  $\mathcal{G}(\mathbf{x})$ , given an observation  $\mathbf{x}$ . Applying Bayes theorem provides posterior membership probabilities of  $\mathbf{x}$  for each group:

$$\Pr(\mathcal{G}(\mathbf{x}) = k \mid \mathbf{X} = \mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{k=1}^K \pi_k f_k(\mathbf{x})}.$$

The group with the highest posterior probability is identified as the predicted class for  $\mathbf{x}$ . Note that having the individual class densities is equivalent to obtaining the posterior class probabilities, and hence the predicted classification (assuming that the  $\pi_k$ ’s are known).

When the densities  $f_k$ ’s are multivariate Gaussian, each with different mean vectors  $\boldsymbol{\mu}_k$ ’s but with common dispersion matrix  $\boldsymbol{\Sigma}$ , the logarithm of the ratio between the posterior class probabilities of classes  $k$  and  $j$  is a linear function in  $\mathbf{x}$ . This is because

$$\log \frac{\Pr(\mathcal{G}(\mathbf{x}) = k \mid \mathbf{X} = \mathbf{x})}{\Pr(\mathcal{G}(\mathbf{x}) = j \mid \mathbf{X} = \mathbf{x})} = \log \frac{\pi_k}{\pi_j} - \frac{1}{2}(\boldsymbol{\mu}_k + \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k + \boldsymbol{\mu}_j) + \mathbf{x}' \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_j).$$

This is the basis for Fisher’s (1936) linear discriminant analysis (LDA). Linear discriminant functions  $\delta_k(\mathbf{x}) = \mathbf{x}' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log \pi_k$  are defined for each group. Then the decision

rule is equivalent to  $\mathcal{G}(\mathbf{x}) = \operatorname{argmax}_k \delta_k(\mathbf{x})$ . In practice, the  $\pi$ 's,  $\boldsymbol{\mu}$ 's and  $\boldsymbol{\Sigma}$  are unknown and need to be estimated. The class proportions of the training data are used to estimate the  $\pi_k$ 's, the individual group means are used to estimate the  $\boldsymbol{\mu}_k$ 's, while the common estimate of  $\boldsymbol{\Sigma}$  is estimated using the pooled within-groups dispersion matrix. Operational implementation of LDA is very straightforward: Using the spectral decomposition  $\hat{\boldsymbol{\Sigma}} = \mathbf{V} \mathbf{D} \mathbf{V}'$  and  $\tilde{\mathbf{X}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{V}' \mathbf{X}$  as the projection of the *sphered* data-point  $\mathbf{X}$  into the transformed space, a new observation  $\mathbf{X}$  is classified to closest class centroid in the transformed space, after scaling “closeness” appropriately to account for the effect of the class prior probabilities  $\pi_k$ 's.

For arbitrary Gaussian densities, the discriminant function  $\delta_k(\mathbf{x}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}_k| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)' \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \log \pi_k$  includes a quadratic term in  $\mathbf{x}$ , so is called quadratic discriminant analysis (QDA). The estimation of the parameters is similar as before, with the  $k$ th group dispersions from the training data used to estimate  $\boldsymbol{\Sigma}_k$ .

For the protein sequence data, the training dataset was used to devise classification rules using both LDA and QDA. With LDA, the test set had a misclassification rate of 30.5% while with QDA, it was only 9.3%. Clearly, LDA was not very successful here. In general, both methods are reputed to have good performance despite being fairly simple. This is perhaps because most decision boundaries are no more complicated than linear or quadratic. Friedman (1989) suggested a compromise between LDA and QDA by proposing regularized discriminant analysis (RDA). The basic idea is to have a regularized estimate for the covariance matrix

$$\hat{\boldsymbol{\Sigma}}_k(\gamma) = \gamma \hat{\boldsymbol{\Sigma}}_k + (1 - \gamma) \hat{\boldsymbol{\Sigma}},$$

where  $\hat{\boldsymbol{\Sigma}}$  is the pooled dispersion matrix from LDA and  $\hat{\boldsymbol{\Sigma}}_k$  is the  $k$ 'th group variance-covariance matrix obtained as per QDA. Other regularizations exist, and we address these shortly. But first, a few additional words on the popularity of LDA. As discussed in Hastie *et al* (2001), one attraction of LDA is an additional restriction that allows substantial reduction in dimensionality through informative low-dimensional projections. The  $K$ -dimensional centroids obtained using LDA span at most a  $(K - 1)$ -dimensional subspace, so there is substantial reduction in dimensionality if  $p$  is much more than  $K$ . So,  $\tilde{\mathbf{X}}$  may as well be projected on to a  $(K - 1)$ -dimensional subspace, and the classes separated there without any loss of information. A further reduction in dimensionality is possible by projecting  $\tilde{\mathbf{X}}$  onto a smaller subspace in some optimal way. Fisher's optimality criterion was to find projections which spread the centroids as much as possible in terms of variance. The problem is similar to finding principal component subspaces of the centroids themselves: formally, the problem is to define the  $j$ 'th *discriminant coordinate* by the corresponding principal component of the matrix of centroids (after scaling with  $\hat{\boldsymbol{\Sigma}}$ ). The solution is equivalent to that for the generalized eigenvalue problem and for Fisher's problem of finding the projection of the data that maximizes the between-class variance (hence separates the different classes as widely as possible) relative to the within-class covariance and is orthogonal to the previous projection coordinates already found. As with principal components, a few discriminant coordinates are often enough to provide well-separated classes. We display LDA using only the first two *discriminant variables* — the projected datapoint using the corresponding discriminant coordinates — for the *E. coli* protein sequence dataset in Figure 2.

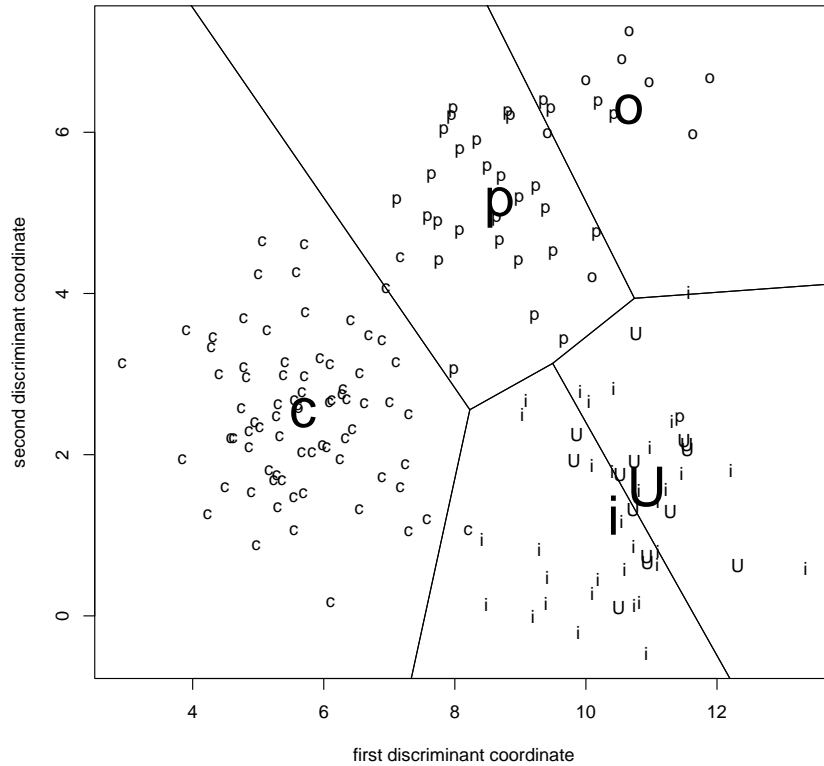


Figure 2: Decision boundaries for the classifier based on the first two linear discriminants. The larger font “c”, “p”, “o”, “u” and “i” represent the projected means in the first two discriminant coordinates for the “cp”, “pp”, “om”, “im” and “imU” categories. Smaller font characters represent the projected observations for each group.

One drawback of LDA arises from the fact that linear decision boundaries are not always adequate to separate the classes. QDA alleviates the problem somewhat by using quadratic boundaries. More general decision boundaries are provided by flexible discriminant analysis (FDA) (Hastie *et al.*, 1994). The basic idea is to recast the classification problem as a regression problem and to allow for nonparametric approaches. To do this, define a score function  $\varphi : \mathcal{G} \rightarrow \mathbb{R}^1$  which assigns scores to the different classes and regress these scores on a transformed version of the predictor variables, perhaps allowing for regularization and other constraints. In other words, the problem reduces to finding scorings  $\varphi$ 's and transformations  $\zeta(\cdot)$ 's that minimize the different classes and regress these scores on a transformed version of the predictor variables, perhaps allowing for regularization and other constraints. In other words, the problem reduces to finding scorings  $\varphi$ 's and transformations  $\zeta(\cdot)$ 's that minimize

the constrained mean squared error

$$MSE(\{\varphi_m, \zeta_m\}_{m=1}^M) = \frac{1}{n} \sum_{m=1}^M \left[ \sum_{i=1}^n \{\varphi_m(\mathcal{G}_i) - \zeta_m(x_i)\} + \lambda J(\zeta_m) \right].$$

Here  $J$  is a regularization constraint corresponding to the desired nonparametric regression methodology such as smoothing and additive splines, kernel estimation and so on. Computations are particularly simplified when the nonparametric regression procedure can be represented as a linear operator  $\varpi_h$  with smoothing parameter  $h$ . Then the problem can be reformulated in terms of the multivariate adaptive regression of  $\mathbf{Y}$  on  $\mathbf{X}$ . Here  $\mathbf{Y}$  is the  $(n \times K)$  indicator response matrix with  $i$ 'th row corresponding to the  $i$ 'th training data-pair  $(\mathbf{X}_i, \mathcal{G}_i)$  and having exactly one non-zero entry of 1 for the column corresponding to  $\mathcal{G}_i$ . Denote  $\varpi_h$  as the linear operator that fits the final chosen model with fitted values  $\hat{\mathbf{Y}}$  and let  $\hat{\zeta}(\mathbf{x})$  be the vector of fitted functions. The optimal scores are obtained by computing the eigen-decomposition of  $\mathbf{Y}'\hat{\mathbf{Y}}$  and then projecting the  $\hat{\zeta}(\mathbf{x})$  on to this normalized eigen-space. These optimal scores are then used to update the fit. The above approach reduces to LDA when  $\varpi_h$  is the linear regression projection operator. FDA can also be viewed directly as a form of penalized discriminant analysis (PDA) where the regression procedure is written as a generalized ridge regression procedure (Hastie *et al.*, 1995).

For the protein sequence dataset, PDA provided us with a misclassification rate of 13.59% on the test set. FDA used with multivariate adaptive regression splines (MARS) (Friedman, 1991) gave us a test set misclassification rate of 16.05%. With ‘BRUTO’, the rate was 16.67% while with a quadratic polynomial regression scoring method, the test set misclassification error was 14.81%.

Another generalization is mixture discriminant analysis (MDA). This is an immediate extension of LDA and QDA and is built on the assumption that the density for each class is a mixture of Gaussian distributions with unknown mixing proportions, means and dispersions (Hastie and Tibshirani, 1996). The number of mixtures in each class is assumed to be known. The parameters are estimated for each class by the Expectation-Maximization (E-M) algorithm. MDA is more flexible than LDA or QDA in that it allows for more prototypes than the mean and variance to describe a class: here the prototypes are the mixing proportions, means and dispersions that make up the mixture dispersion in each class. This allows for more general decision boundaries beyond the linear or the quadratic. One may combine FDA or PDA with MDA models to allow for even more generalizations.

More flexible options for decision-theoretic approaches to the classification problem involve the use of nonparametric density estimates for each class, or the special case of *naive Bayesian* approaches which assume that the inputs are conditionally independent within each class. The full array of tools in density estimation can then be put to work here.

### 4.3 Distribution-free Predictive Approaches

The methods discussed in the previous section are essentially model-based. Model-free approaches such as tree-based classification also exist and are popular for their intuitive appeal.

In this section, we discuss in detail a few such predictive approaches – these are the nearest-neighbor methods, support vector machines, neural networks and classification trees.

### 4.3.1 Nearest-neighbor Approaches

Perhaps the simplest and most intuitive of all predictive approaches is  $k$ -nearest-neighbor classification. Depending on  $k$ , the strategy for predicting the class of an observation is to identify the  $k$  closest neighbors from among the training dataset and then to assign the class which has the most support among its neighbors. Ties may be broken at random or using some other approach. Despite the simple intuition behind the  $k$ -nearest neighbor method, there is some similarity between this and regression. To see this, notice that the regression function  $f(\mathbf{x}) = \mathbb{E}(Y \mid \mathbf{X} = \mathbf{x})$  minimizes the expected (squared) prediction error. Relaxing the conditioning at a point to include a region “close” to the point, and with a 0-1 loss function leads to the nearest-neighbor approach.

The choice of  $k$  or the number of neighbors to be included in the classification at a new point is important. A common choice is to take  $k = 1$  but this can give rise to very irregular and jagged regions with high variances in prediction. Larger choices of  $k$  lead to smoother regions and less variable classifications, but do not capture local details and can have larger biases. Since this is a predictive problem, a choice of  $k$  may be made using cross-validation. Cross-validation was used on the protein sequence training dataset to obtain  $k = 9$  as the most optimal choice in terms of minimizing predicted misclassification error. The distance function used here was Euclidean. The 9-nearest neighbor classification predicted the protein sequences in the test set with a misclassification rate of 15.1%. With  $k = 1$ , the misclassification rate was 17.3%.

On the face of it,  $k$ -nearest neighbors have only one parameter – the number of neighbors to be included in deciding on the majority-vote predicted classification. However, the effective number of parameters to be fit is not really  $k$  but more like  $n/k$ . This is because the approach effectively divides the region spanned by the training set into approximately  $n/k$  parts and each part is assigned to the majority vote classifier. Another point to be noted is that the *curse of dimensionality* (Bellman, 1961) also impacts performance. As dimensionality increases, the data-points in the training set become closer to the boundary of the sample space than to any other observation. Consequently, prediction is much more difficult at the edges of the training sample, since some extrapolation in prediction may be needed. Furthermore, for a  $p$ -dimensional input problem, the sampling density is proportional to  $n^{\frac{1}{p}}$ . Hence the sample size required to maintain the same sampling density as in lower dimensions grows exponentially with the number of dimensions. The  $k$ -nearest-neighbors approach is therefore not immune to the phenomenon of degraded performance in higher dimensions.

### 4.3.2 Support Vector Classifiers and Machines

Consider a two-class problem with two input dimensions in the feature space as represented in Figure 3. Without loss of generality, let the class indicators  $\mathcal{G}_i$  be -1 (represented by filled  $\diamond$ 's) and +1 (denoted by filled  $\circ$ 's). Figure 3a shows the case where the two classes

are linearly separable (the line  $\boldsymbol{\omega}^t \boldsymbol{x} = \gamma$  is one possible line). Note that for points classified correctly,  $\mathcal{G}_i(\boldsymbol{\omega}^t \boldsymbol{x} - \gamma) \geq 0$ . Rosenblatt (1958) devised the perceptron learning algorithm with a view to finding a separating hyperplane (line in two dimensions) minimizing the distance of misclassified points to the decision boundary. However Figure 3a shows that there can be an infinite number of such solutions. Vapnik (1996) provided an elegant approach to this problem by introducing an additional requirement: the revised objective is to find the *optimal separating hyperplane* separating the two classes while also maximizing the shortest distance of the points in each class to the hyperplane. Assuming without loss of generality that  $\|\boldsymbol{\omega}\| = 1$ , the optimization problem is to maximize  $\delta$  over  $\|\boldsymbol{\omega}\| = 1, \gamma$  subject to the constraint that  $\mathcal{G}_i(\boldsymbol{\omega}^t \boldsymbol{x}_i - \gamma) \geq \delta$  for all  $i = 1, 2, \dots, n$ . The problem can be written in the equivalent but more convenient form as: minimize  $\|\boldsymbol{\omega}\|$  over all  $\boldsymbol{\omega}$  and  $\gamma$  subject to the constraint  $\mathcal{G}_i(\boldsymbol{\omega}^t \boldsymbol{x}_i - \gamma) \geq 1$  for all  $i = 1, 2, \dots, n$ . Here, we have dropped the unit-norm requirement on  $\boldsymbol{\omega}$  and set  $\delta = \|\boldsymbol{\omega}\|^{-1}$ . Standard solutions to this problem exist and involve quadratic programming. The classifier for a new point  $\boldsymbol{x}$  is given by  $\text{sign}(\hat{\boldsymbol{\omega}}^t \boldsymbol{x} - \hat{\gamma})$  where  $\hat{\boldsymbol{\omega}}, \hat{\gamma}$  are the optimal values.

Figure 3b shows the case where the two classes overlap in the input space so that no hyperplane can completely separate the two groups. A reasonable compromise is to allow for some variables to be on the wrong side of the margin. Let us define additional non-negative slack variables as  $\boldsymbol{\vartheta} = \{\vartheta_i, i = 1, 2, \dots, n\}$ . The original constraint in the problem of finding the optimal separating hyperplane is then modified to be  $\mathcal{G}_i(\boldsymbol{\omega}^t \boldsymbol{x}_i - \gamma) \geq \delta(1 - \vartheta_i)$  for all  $i = 1, 2, \dots, n$  with  $\sum \vartheta_i$  bounded above by a constant. The slack variables  $\vartheta_i$  in the constraint denote the proportional amount by which any predicted observation is on the wrong side of the classification hyperplane. Slapping the upper bound on their sum limits the total proportional amount by which predictions are made on the wrong side of the margin. Note that misclassifications occur only when  $\vartheta_i > 1$  so that  $\sum \vartheta_i \leq M$  automatically limits the number of training set misclassifications to at most  $M$ . Once again, the optimization problem can be rephrased as that of minimizing  $\|\boldsymbol{\omega}\|$  subject to the constraints  $\mathcal{G}_i(\boldsymbol{\omega}^t \boldsymbol{x}_i - \gamma) \geq (1 - \vartheta_i)$  with  $\delta = \|\boldsymbol{\omega}\|^{-1}, \sum \vartheta_i \leq M, \vartheta_i \geq 0$  for all  $i = 1, 2, \dots, n$  so that similar to the separable case, quadratic programming techniques can again be used. It can be shown (Hastie *et al.*, 2001) that the solution  $\hat{\boldsymbol{\omega}}$  is in the form  $\hat{\boldsymbol{\omega}} = \sum_{i=1}^n \hat{\alpha}_i \mathcal{G}_i \boldsymbol{x}_i$ , with non-zero values of  $\hat{\alpha}_i$  only when  $\mathcal{G}_i(\boldsymbol{\omega}^t \boldsymbol{x}_i - \gamma) = (1 - \vartheta_i)$ . The solution is called the *support vector classifier*. Notice that points with non-zero values for the slack variables  $\vartheta_i$  play a major role — these are called the *support vectors*. This is a major distinction with LDA where the solution depends on all data points, including those far away from the decision boundary. It does this through calculations on the class covariance-matrices and the class centroids. The support vector classifier on the other hand uses all data-points to identify the support vectors, but focuses on the observations near the boundary for the classification. Of course, if the underlying classes are really Gaussian, LDA will perform better than SVM because of the latter's heavy reliance on the (noisier) observations near the class boundaries.

Support vector machines (Cristianini and Shawe-Taylor, 2001) generalize the above scenario in a spirit similar to the extension of FDA over LDA. The support vector classifier finds linear boundaries in the input feature space. The approach is made more flexible by

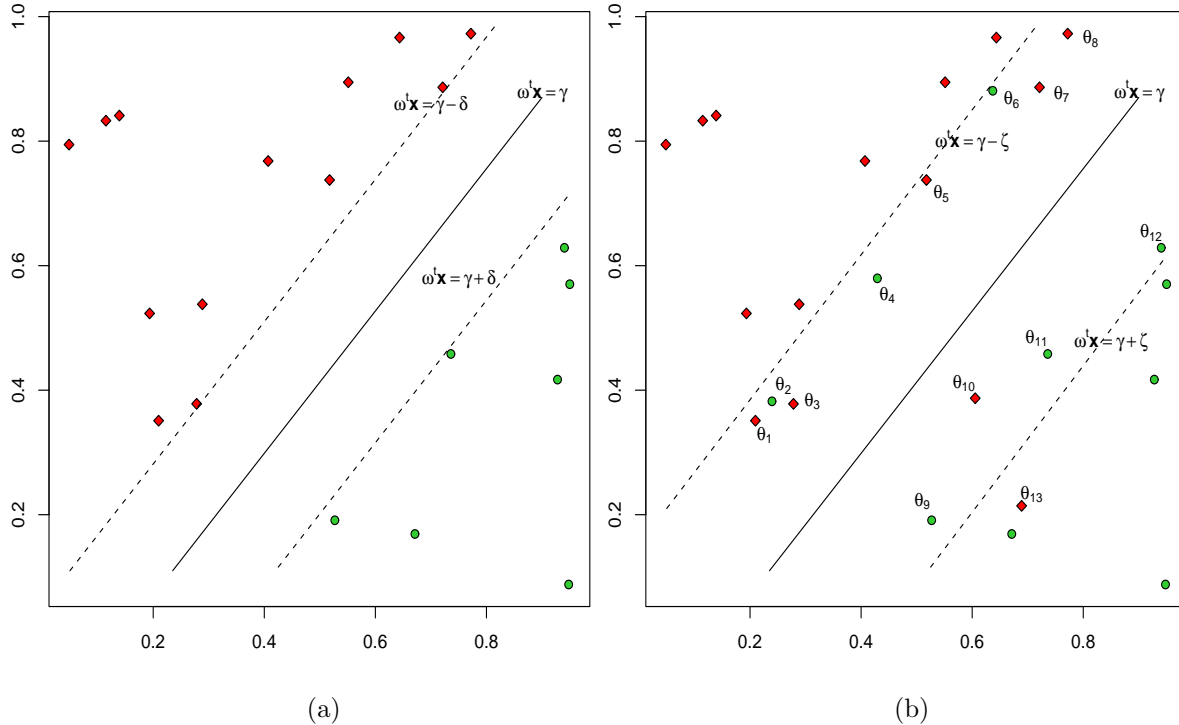


Figure 3: The support vector classifier for a two-class problem in a two-dimensional input space. (a) In the separable case, the decision boundary is the solid line while broken lines bound the maximal separating margin of width  $2\delta$ . (b) In the overlapping case, points on the wrong side of their margin are labeled  $\theta_j = \delta\vartheta_j$ , which is the distance from their margin. All unlabeled points have  $\theta_j = 0$ . The maximal margin itself is obtained within the constraint that  $\sum \vartheta_j$  does not exceed a certain allowable “budget”.

enlarging the feature space to include basis expansions such as polynomials or splines. Once the basis functions are selected to be, say,  $\zeta(\mathbf{x}) = \{\zeta_i(\mathbf{x}), i = 1, 2, \dots, m\}$ , the problem is recast in the same frame-work to obtain the classifier  $\mathcal{G}(\mathbf{x}) = \text{sign}(\hat{\omega}^t \zeta(\mathbf{x}) - \hat{\gamma})$ . Similar to the case of linear boundaries, the solution is now  $\hat{\omega} = \sum_{i=1}^n \hat{\alpha}_i \mathcal{G}_i \zeta(\mathbf{x}_i)$ . This representation makes computations practical. This is because the fitted function  $\hat{\mathcal{G}}(\mathbf{x})$  now has the form  $\sum_{i=1}^n \hat{\alpha}_i \mathcal{G}_i \langle \zeta(\mathbf{x}), \zeta(\mathbf{x}_i) \rangle$  which means that knowledge of the *kernel function*  $\Psi(\mathbf{x}, \mathbf{y}) = \langle \zeta(\mathbf{x}), \zeta(\mathbf{x}_i) \rangle$  rather than the exact transformation  $\zeta(\mathbf{x})$  is enough to solve the problem. Popular choices for  $\Psi(\mathbf{x}, \mathbf{y})$  include the  $p$ th degree polynomial  $\Psi(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^t \mathbf{y})^p$ , the radial basis function  $\Psi_h(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / h)$  or the sigmoidal function denoted by  $\Psi(\mathbf{x}, \mathbf{y}) = \tanh(\beta_1 \mathbf{x}^t \mathbf{y} + \beta_0)$ .

Support vector machines with different kernels were used on the protein sequence dataset. Support vector classification using 35 support vectors, yielded a test set misclassification rate of 62.34%. A slightly higher misclassification rate of 63.5% was obtained with the sigmoidal

kernel with  $\beta_1 = 0.2$  and  $\beta_0 = 0$ . This last application used 33 support vectors. Using a polynomial kernel of degree 6 improved misclassification a bit — prediction errors on the test set were on the order of 56.8%. The number of support vectors was then 45. Finally using the radial basis function with  $h = 5$  gave a test set misclassification rate of 17.9%. The number of support vectors was identified to be 51. Clearly, the performance is the best with the radial kernel, even though it is worse than that using either QDA, FDA, PDA or  $k$ -nearest neighbors.

I conclude with a brief discussion on recent work in this fast-developing area. As with several similar problems, support vector machines are also challenged by massive databases. One approach, developed by Platt (1998, 1999) is to break the quadratic programming into a series of smallest possible problems. This is called *chunking*. The smallest possible chunk size is 2. Since the component problems are small, analytic solutions may be obtained, speeding up computations. Another area of interest is to formulate the problem in a probabilistic framework and to obtain not a prediction but the posterior probability of a class given an input. This is done, for example, by formulating the classification model through a logistic link function  $\text{Prob}(\mathcal{G}_i = 1 \mid \mathbf{x}) = (1 + \exp\{-f(\mathbf{x})\})^{-1}$  (Wahba, 1999). Other approaches also exist — see, for example, Hastie and Tibshirani (1998), Platt (1999), Vapnik (1996) and the references therein.

### 4.3.3 Artificial Neural Networks

Consider a classification problem with two input variables  $\mathbf{X}_1$  and  $\mathbf{X}_2$ . A single-hidden-layer artificial neural network is obtained by defining a layer  $\mathbf{Z}$  with components  $Z_l = \sigma(\alpha_{0l} + \boldsymbol{\alpha}_l^t \mathbf{X})$ ,  $l = 1, 2, \dots, L$ . The output  $\mathcal{G}$  is then assumed to be a perturbed version of a function of a linear combination of this layer. Here,  $\sigma(\cdot)$  is the *activation function* and  $\mathcal{G}_i = F_j(\mathbf{X}) \star \epsilon$  with  $F(\mathbf{X}) = g_j(T_j)$  where  $T_j = \beta_{0j} + \boldsymbol{\beta}_j^t \mathbf{X}$  and  $\star$  represents the degradation operator on the signal  $F(\mathbf{X})$  with  $\epsilon$  and  $j = 0, 1$  for the two-class problem (more generally  $0, 1, \dots, K - 1$  for the  $K$ -class problem). This is illustrated in Figure 4, with  $m = 3$ . The  $\mathbf{X}$ 's form the input layer, the  $\mathbf{Z}$ 's are the hidden layer and the  $\mathbf{Y}$ 's are the output layer. The components  $\alpha_0$ 's and  $\beta_0$ 's are the *bias* units for each layer. In general there can be several hidden layers. The most common choice for the activation function is the sigmoid  $\sigma(u) = (1 + \exp(-u))^{-1}$  though a Gaussian radial basis function  $\sigma(u) = \exp\{-u^2\}$  has also been used in what is called a *radial basis function network*. The function  $g_j(\cdot)$  is called the *output function* and is usually the *softmax function*  $g_j(v) = \exp(T_j) / \sum_j \exp(T_j)$ . This is of course the transformation also used in logistic regression which like other regression models such as multiple linear regression or projection pursuit (Friedman and Stuetzle, 1984) are special cases within this framework: choosing the identity function for  $\sigma(\cdot)$ ,  $m = 1$ ,  $\beta_0 = 0$  and the binomial distribution for instance, takes us to the logistic model. Note that there may be any number of hidden layers in the model.

Fitting the neural network means estimating the unknown parameters  $\boldsymbol{\alpha}$ 's and  $\boldsymbol{\beta}$ 's to obtain the classifier  $\mathcal{G}(\mathbf{x}) = \text{argmax}_j F_j(\mathbf{X})$  optimizing the error function - usually chosen to be least-squares or entropy error. Because of the strong possibility of over-fitting, given the

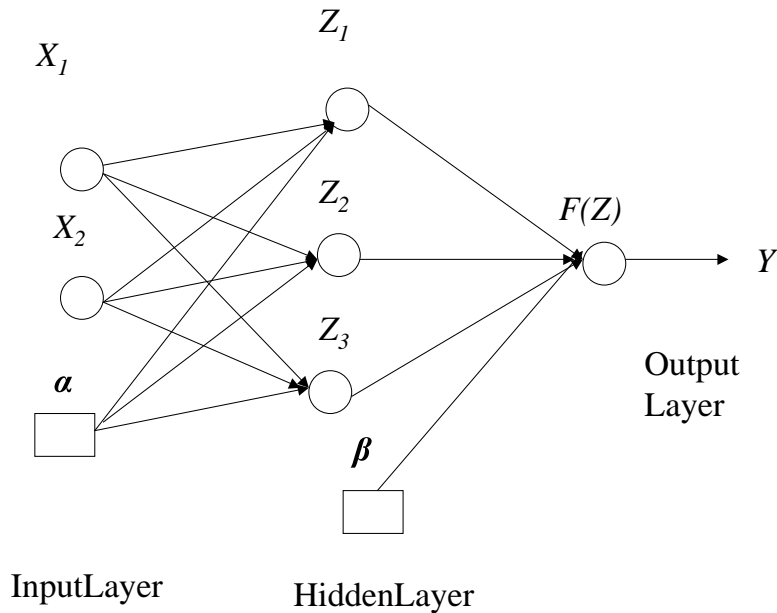


Figure 4: A two-variables-input, one-hidden layer, feed-forward neural network.

model's enormous flexibility, a penalty component is usually incorporated: early stopping is also employed. The most common method used to fit a neural network model is *back-propagation* proposed by Hopfield (1982, 1984) — essentially a gradient descent approach which works by setting the parameters to some initial values, building  $F$  from these values in the network and then computing the error in the fitted  $F$  and the observations. This error is then propagated back to the previous hidden layer where it is apportioned to each component. The process is in turn repeated down to the previous layer and on down to the input layer. The input parameters are adjusted to minimize these apportioned errors and fed forward to form a new updated version of  $F$ . This procedure is iterated till convergence. Back-propagation can be quite slow operationally and is sped up using conjugate gradients and variable metric methods.

Artificial neural networks have been wildly popular in the engineering and artificial intelligence communities, partly because of an effective naming strategy that has linked the methodology to an imitation of the brain. The enormous flexibility provided by any number of hidden layers is often mentioned as its great virtue. However, from a statistical perspective, they are really nothing more than non-linear statistical models with the inherent

dangers of over-fitting presented by the possibly unrestrained number of parameters and layers. Over-fitting of the weight parameters is sought to be prevented by incorporating penalty functions and by using cross-validation to decide on the weight of the penalty. The number of hidden units and layers is best left to knowledge and understanding of the system being modeled and/or to experimentation. Additionally, the initial weights supplied in the fitting algorithms have a good deal of impact in the rate of convergence and the final solution — small weights mean that the algorithm hardly moves, while large weights often lead to poor solutions. Also, since scaling the inputs impacts the quality of the solution, it is perhaps best to start with initializers all standardized to zero mean and unit standard deviation. Finally, as with many nonlinear problems, multiple local minima exist — the solution is either to try out a number of runs with different starting points and choose the optimal from among the solutions, or to average over predictions from different solutions (Ripley, 1996).

Applying the single-hidden-layer neural network on the protein sequence training sample with an entropy error function and several random initial weights yielded a network with two nodes in the hidden layer and a test set misclassification rate of 22.22%.

#### 4.3.4 Classification Trees

We conclude this description of nonparametric classification with a brief discussion on classification trees (Breiman *et al.*, 1984). These are supervised learning methods which result in a tree-like structure, effectively partitioning the input feature space into sets of rectangles and then taking a majority vote of the training set observations in each set to come up with a classification rule. A common approach is to use only binary splits at each node, though multi-split approaches are also used. The tree is built by finding the split in the predictor variable which best splits the data into heterogeneous groups. Thus it is inductively built through future splits that are wholly dependent on past choices (such algorithms are called greedy algorithms). The tree-depth is usually found by cross-validation. We used the training sample for the protein sequence dataset to come up with a seven-node tree as the best leave-out-one cross-validated tree. This tree yielded a 19.75% test set misclassification rate.

Classification trees are attractive because of their interpretability and their ability to handle missing observations – as a new value or as a surrogate variable for splitting at a node. Furthermore, they are easy to use in higher dimensions since the splitting is often done one feature at a time. Moreover both categorical and quantitative input variables can be incorporated in the model. However, there are some disadvantages. Some algorithms require discrete (binned) input variables. More importantly, a tree model invariably has high-order interactions, given that all splits depend on previous ones. Also classification is very often done across several databases and there needs to be a way for unifying such trees. I discuss methods for aggregating different trees in the following section, but conclude this section with an outline of a different approach to this problem. The strategy is to take Fourier Transforms of the different classification trees from the different databases (Kargupta and Park, 2001). Indeed, the coefficients of these representations decay exponentially, so that storing only a few for computations is adequate. This is especially useful for applications such as mobile

computing where memory and bandwidth are limited and need to be conserved. The stored coefficients can then be combined to build a tree-based structure based on all the databases.

## 4.4 Aggregation Approaches

The classification approaches perform differently on different datasets. No method uniformly performs better than any other on all datasets, and this is indeed expected because each technique addresses specific situations. Moreover, some methods such as classification trees have high variance and are not very robust. Perturbing the data slightly can result in a very different tree structure. In this section, we discuss approaches which combine different models as well as aggregate over bootstrap samples of the data.

### 4.4.1 Bootstrap Aggregation

The concept of bootstrap aggregation or *bagging* was first introduced by Breiman (1996). Suppose that as before, we have training data of the form  $\mathbf{Z} = \{(\mathbf{X}_i, \mathcal{G}_i) : i = 1, 2, \dots, n\}$ , and let us assume that this yields the classification rule  $\hat{\mathcal{G}}(\mathbf{x})$ . Bagging computes a composite rule over a collection of bootstrap samples with variance reduction as the goal. For each bootstrap sample  $\mathbf{Z}^{b^*}$ ,  $b = 1, 2, \dots, B$  a classifier  $\hat{\mathcal{G}}^{b^*}(\mathbf{x})$  is obtained. The test set is classified from each bootstrapped rule  $\hat{\mathcal{G}}^{b^*}(\mathbf{x})$  and each observation in the test set is classified to that group which has the highest representation among the bootstrap predictions. The scheme outlined here is a majority voting rule. (An alternative scheme is to average the posterior probability for each class over the bootstrap predictions.)

Bagging is a methodology also used in regression — indeed it can be shown quite easily that within this framework and under squared-error loss, the expected error in prediction is reduced. The same is not necessary in classification — bagging a good classifier can make it better, but bagging a bad one can actually make it worse! To see this, consider the simple two-class scenario where the observation  $\mathcal{G}_i \equiv 1$  for all  $\mathbf{X}$ . Any classifier  $\hat{\mathcal{G}}(\mathbf{x})$  that predicts an observation into class 0 with probability  $\alpha$  and class 1 otherwise will have a misclassification rate  $\alpha$  while the error rate for a bagged rule will be  $1.0$ , for  $\alpha > 0.5$ .

Using bootstrap replications of size 500, we illustrate bagging on the protein sequence dataset with some of the methods described earlier — these are  $k$ -nearest neighbors, the single-layer four-hidden-nodes neural network and the classification tree. For the  $k$ -nearest neighbor case,  $k$  was chosen for each bootstrap sample to be the one with the lowest misclassification rate on the complete training sample. For tree classification, the best tree size was obtained by leave-out-one cross-validation of the bootstrap sample. In all cases, the final rule was decided from the bootstrapped classification rules  $\hat{\mathcal{G}}^{b^*}(\mathbf{x})$  by majority voting. For the  $k$ -nearest neighbors classification approach, bagging resulted in an error rate of 17.28%. Bagging the neural network produced a test set misclassification rate of 20.99% while the bagged tree classifier produced an error rate of 16.67%. Interestingly therefore, bagging degraded the performance of the  $k$ -nearest neighbor classifier but improved performance of the tree and neural network classifiers.

#### 4.4.2 Bumping

Bumping is really a method of stochastic search that tries to find a better model. The method uses bootstrap sampling to search the space for a good model. This method can be used effectively to get out of the problems associated with local minima. Operationally, the method is implemented by taking bootstrap samples  $\mathbf{Z}^{b^*}$ ,  $b = 1, 2, \dots, B$  and deriving the corresponding classification rule  $\hat{\mathcal{G}}^{b^*}(\mathbf{x})$ . Each rule is then used to fit the training set. The classification rule with the least error rate on the training set is chosen to be the optimal rule found by bumping. We use bumping on both the classification tree and the neural network rules to classify the protein sequence localization dataset. The test set misclassification rate of the bumped leave-out-one cross-validated tree classifier was 20.37% while that of the two-node bumped neural network classifier was 25.31%.

#### 4.4.3 Boosting

I finally address a very powerful method originally introduced in the machine learning literature, called *boosting*. This method as introduced by Freund and Schapire (1997) is intended to improve performance of “weak” classifiers by combining several of them to form a powerful “committee” – in that sense, this is similar to committee methods such as bagging (Friedman *et al.*, 2000). A weak classifier is one that performs only marginally better than random guessing. Boosting applies the weak classifier successively to modified portions of the dataset. The modification introduced in each iteration is to increase representation from misclassified observations while decreasing participation from parts of the dataset where the classification rule does well. Thus, as the methodology progresses, the algorithm concentrates on those parts of the feature space that are difficult to classify. The final step is to take a weighted vote of all the classification rules that come out from every iteration. Operationally, the algorithm called *AdaBoost.M1* introduced by Freund and Schapire (1997) has the following steps:

1. Assign initial weights  $\omega_j$  to each observation in the training set,  $i = 1, 2, \dots, n$ .
2. For  $j = 1$  to  $J$  do the following:
  - (a) Fit a classifier  $\hat{\mathcal{G}}^j(\mathbf{x})$  to the training dataset where the  $i$ th observation is assigned the weight  $\omega_i$ ,  $i = 1, 2, \dots, n$ .
  - (b) Let  $\alpha_j = \log \frac{1-\varepsilon_j}{\varepsilon_j}$ , where  $\varepsilon_j = \frac{\sum_{i=1}^n \omega_i \mathbf{1}[\hat{\mathcal{G}}^j(\mathbf{x}_i) \neq \mathcal{G}_i]}{\sum_{i=1}^n \omega_i}$ .
  - (c) Re-weight  $\omega_i$  to be  $\omega_i \exp \{ \alpha_j \mathbf{1}[\hat{\mathcal{G}}^j(\mathbf{x}_i) \neq \mathcal{G}_i] \}$  for  $i = 1, 2, \dots, n$ .
3. Compute the final boosted classifier  $\hat{\mathcal{G}}(\mathbf{x}) = \operatorname{argmax}_k \sum_{j=1}^J \alpha_j \mathbf{1}[\hat{\mathcal{G}}^j(\mathbf{x}) = k]$ .

Note that for the two-class problem with classes designated as  $\{-1, 1\}$ , the final classifier reduces to  $\hat{\mathcal{G}}(\mathbf{x}) = \operatorname{sign} \left[ \sum_{j=1}^J \alpha_j \hat{\mathcal{G}}^j(\mathbf{x}) \right]$ . Also the above can be viewed as a way of fitting

an additive expansion if we consider the individual classifiers  $\hat{G}^j(\mathbf{x})$  to be a set of elementary “basis” functions. This equivalence of the boosting procedure to forward stage-wise modeling of an additive model is key to understanding its performance in a statistical framework (Friedman *et al.*, 2000). We illustrate boosting on the tree classifier, the PDA classifier and the neural network classifier for the protein sequence localization data. The improvement in the test set misclassification rate was marginal (to 17.9%) for the boosted leave-out-one cross-validated tree classifier. Boosting the 2-node neural network classifier had the best improvement, with a decrease of misclassification rate to 20.37% on the test set while boosting the PDA classifier had no effect – the test set error remained at 13.58%.

We end by noting that unlike the other methods discussed here, boosting can be adapted to infinite data streams, such as credit card transactions data. In this two-class problem, the goal is real-time identification of fraudulent transactions. Fraud is usually as newer and newer such schemes are invented and put to practice. At the same time, any classifier should guard against past kinds of aberrant behavior. One way of applying boosting is to keep portions of the past data and to include new transactions data weighted by cases where the current rule performed poorly to derive a classifier that can then be applied and fine-tuned on new streaming data. This methodology can also be applied to derive classifiers from massive databases by first obtaining a rule from a small sample, using that to get a weighted sub-sample from the remainder where the rule does not perform well, recomputing the rule and so on until the final weighted rule does not show further improvement.

## 5 Clustering and Unsupervised Learning

The ability to group data into a previously unknown number of categories is important in a variety of contexts. Applications include grouping species of bees in taxonomy (Michener and Sokal, 1957), clustering corporations on the basis of size, assets, and so on in business (Chen *et al.*, 1974), locating Iron Age settlements from broaches found in archaeology (Hodson *et al.*, 1966), or demarcating polar regions using characteristics of sea ice and fern in glaciology (Rotman *et al.*, 1981). In recent times, the importance of clustering has been made more acute by the huge databases created by automated data collection methods. For example, identifying niche groups from gigantic customer databases for targeted advertising is useful (Hinneburg and Keim, 1999). In industrial manufacturing Hinneburg and Keim (1999) homogeneous groups may be identified on the basis of “feature” vectors — such as Fourier or wavelet coefficients — from a large computer-aided-design (CAD) database of parts. Standard parts may then be manufactured for each representative group in lieu of specialized parts reducing costs and the number of kinds of parts to be produced. Many more applications exist: here I present the software metrics example of Maitra (2001).

### 5.1 Clustering Software Metrics

The software industry is a very dynamic sector with new releases and upgrades becoming available in very short time-frames. The challenge for any software industry is to offer

both maintenance and upgrade services. However because this is also a sector with very high personnel turnover rates, product maintenance or upgrade are rarely carried out by the same set of developers. The current personnel must therefore have a proper grasp of the complexities of the product. Traditionally, they have relied on system source code documentation and manuals. With the increasingly short time-lines associated with software production, comprehensive manuals have all but disappeared. Automated methodology to reveal a comprehensive view of the inner organization of source code is therefore desirable. Since modularity characterizes good software packages (Myers, 1978; Yourdon, 1975), one may cluster procedures and functions using measurements on different aspects of code, such as number of blank lines, non-commented source lines, calls to itself, calls to external routines and so on. Many software packages contain a huge number of components – the example in Maitra (2001) that we revisit here has 32 measurements on 214,982 procedures. Clustering algorithms for such datasets are therefore desirable and important for the software industry.

## 5.2 An Overview of Clustering Methods

There is a large body of methodological work in clustering (Banfield and Raftery, 1993; Beckett, 1977; Brossier, 1990; Can and Ozkaran, 1984; Chen *et al.*, 1974; Cormack, 1971; Celeux and Govaert, 1995; Eddy, 1996; Everitt, 1988; Fowlkes, 1988; Gnanadesikan, 1977; Good, 1979; Hartigan, 1975, 1985; Mojena and Wishart, 1980; Murtagh, 1985; Ramey, 1985; Ripley, 1991; Symons, 1981; Van Ryzin, 1977). Broadly, clustering algorithms can be divided into two kinds: hierarchical clustering techniques and optimization partitioning algorithms (Mardia *et al.*, 1979). The former algorithms partition the dataset in a hierarchy resulting in a tree structure, with the property that all observations in a group at some branch node remain together higher up the tree. Most of these techniques are based on merging or splitting groups of data-points on the basis of a pre-defined between-groups similarity measure. Splitting or top-down methods start with one node representing the single group to which the entire dataset belongs. At each stage, the group splits into two portions such that the distance between the two groups is maximized. There are several criteria for calculating inter-group distances. The *single linkage* criterion specifies the distance between any two groups to be the minimum of all distances between observations in either group, and results in lean, skinny clusters. The *complete linkage* criterion specify the distance between any two groups to be the maximum of all pairwise distances between points in the two groups and results in compact clusters. *Average linkage* defines the distance between the groups as the average of all the pairwise distances between every pair of points in the two groups and results in groups that are round but also that may have some extreme observations. Several other criteria also exist. Agglomerative or bottoms-up approach start with each observation as its own group. Groups which are closest to each other are successively merged, using one of the same sets of criteria outlined above, and form a hierarchy right up to the very top where every observation in the dataset is a member of the same group.

Optimization-partitioning algorithms divide the dataset into a number of homogeneous clusters based on an optimality criterion, such as the minimization of some aspect of the

within-sums-of-squares-and-products matrix (Friedman and Rubin, 1967; Scott and Symons, 1971). Some common approaches are  $K$ -means clustering,  $K$ -medoids, self-organizing maps and probability clustering. We discuss each method briefly in the following paragraphs.

The  $K$ -means algorithm (Hartigan and Wong, 1979) is widely used in a number of applications. The procedure finds a locally optimal partition of the input space close to an initial set of  $K$  cluster centers using Euclidean distance. The algorithm iteratively updates the partitions and cluster centers till convergence. The partitions are updated by assigning each point in the dataset to the center closest to it, and the cluster centers are calculated at each step by calculating the center of the data-points in each partition. The  $K$ -means algorithm performs best when the underlying clusters are reasonably spherical, well-separated and homogeneous. The method has some optimality properties — Pollard (1981) provides an elegant proof for the strong consistency of the cluster center estimates.

A generalization of the above is the  $K$ -medoids algorithm, which also finds a locally optimal partition in the vicinity of the initial cluster values called “medoids”. The algorithm is flexible enough to allow for any distance measure. Similar to  $K$ -means, the algorithm iteratively updates the partitions and the “medoids” alternately, till convergence. The partitions are updated at each step by assigning each observation to the class with the closest medoid (as per the distance measure). The medoid for each class is updated by setting it to be the observation in the group having the smallest total distance to all its other fellow cluster members. The  $K$ -medoids algorithm is clearly more computationally demanding than  $K$ -means because of the medoids-update step. Simplified strategies have been suggested by Kaufman and Rousseeuw (1990) and earlier by Massart *et al.* (1983). The latter suggests a branch-and-bound combinatorial method for the problem that can be practically implemented only when the size of the dataset is very small.

A constrained version of the  $K$ -means algorithm is provided by Self-Organizing Maps (SOM) (Kohonen, 2001). The idea behind this algorithm, originally provided by Kohonen (1990) is to map the original input-space to a lower-dimensional manifold and then to find  $K$  prototypes representing the  $K$  groups in the lower-dimensional space. Each of the  $K$  prototypes  $\{\mu_j; j = 1, 2, \dots, K\}$  are parameterized with respect to a coordinate system  $\ell_j \in \mathcal{Q}$ . For instance, the  $\mu_j$  may be initialized to lie in the  $p$ -dimensional principal component plane of the data. The SOM algorithm tries to bend the plane so that these prototypes approximate the data points as well as possible. A final step maps the observations onto the  $p$ -dimensional grid. The updates on the  $\mu$ 's can be done by processing each observation  $\mathbf{x}$  one at a time. The process involves first finding the closest  $\mu_i$  to the  $\mathbf{x}$  and then updating the  $\mu$ 's using  $\mu_j = \mu_j + \alpha\gamma(\|\ell_i - \ell_j\|)(\mathbf{x} - \mu_j)$ . Here,  $\gamma(\cdot)$  is a decreasing *neighborhood* function which effectively implies that prototypes  $\mu_j$ 's have more weight if their corresponding coordinates  $\ell_j$ 's in the lower-dimensional map are close to  $\ell_i$ . The parameter  $\alpha$  is the learning rate and is decreased gradually from 1 to zero as the algorithm progresses. The function  $\gamma(\cdot)$  also usually becomes more rapidly decreasing as the algorithm proceeds. Note that using a neighborhood function such that every prototype has only one neighbor in the  $\mathcal{Q}$ -coordinate system yields an online version of the  $K$ -means algorithm. When applied in this framework, experiments show that the algorithm does stabilize at one of the local min-

ima of the  $K$ -means algorithm. Batch versions of SOM also exist: each  $\boldsymbol{\mu}_j$  is updated to  $\sum_j \gamma(\|\boldsymbol{\ell}_i - \boldsymbol{\ell}_j\|)(\boldsymbol{x} - \boldsymbol{\mu}_j) / \sum_j \gamma(\|\boldsymbol{\ell}_i - \boldsymbol{\ell}_j\|)$ .

Finally, we discuss a model-based clustering approach, which is implemented through the expectation-maximization (E-M) algorithm. The dataset  $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$  is assumed to be a random sample from the mixture distribution

$$f(\boldsymbol{x}) = \sum_{k=1}^K \pi_k \phi_k(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

where  $\pi_k$ 's sum to unity, and  $\phi_k(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$  represents the multivariate normal density with mean  $\boldsymbol{\mu}_k$  and dispersion matrix  $\boldsymbol{\Sigma}_k$ . The goal is to estimate the mixing proportions  $\pi_k$ 's, as well as  $\boldsymbol{\mu}_k$  and  $\boldsymbol{\Sigma}$ . With a known  $K$ , the problem can be set up as a likelihood-maximization problem and solved iteratively using an E-M approach. The proposal (Day, 1969; Dick and Bowden, 1973; Wolfe, 1970) updates the class proportions  $\pi_k$ 's in the E-step and the cluster means  $\boldsymbol{\mu}_k$ 's and dispersions  $\boldsymbol{\Sigma}_k$ 's in the M-step. The E-M algorithm has been shown to exhibit strong consistency (Render and Walker, 1985) — the result holds for the wider class of a known number of mixtures of exponential families. In the E-step,  $\pi_k$ 's are updated given the current values of  $\pi_k$ 's,  $\boldsymbol{\mu}_k$ 's,  $\boldsymbol{\Sigma}_k$ 's by classifying the observations into each of the  $K$  groups and then estimating the current value of  $\pi_k$  by the proportion of observations classified into the  $k$ 'th group. The M-step then uses the updated classification to obtain updated estimates for the class means and dispersions. The procedure starts with initial estimates for all parameters and continues until convergence. For unknown  $K$  however, the E-M algorithm is not necessarily consistent (see the report of the Panel on Discriminant and Clustering, 1989). I now mention some issues in the choice of the number of clusters  $K$ .

Determining the number of clusters  $K$  has been quite a challenge: there are many suggestions — some of them were made by Marriott (1971), Mojena and Wishart (1980) and more recently, by Tibshirani *et al.* (2001). Everitt (1974) performed a simulation analysis of then-available criteria for choosing  $K$  and reported that Marriott's (1971) criterion of minimizing  $(k^2 | \mathbf{W}_k |)$  with respect to  $k$  and subject to non-singular  $\mathbf{W}_k$  to be most useful. This criterion assumes a common dispersion matrix for  $\boldsymbol{\Sigma}_k$  and denotes the pooled within-clusters sums-of-squares-and-products matrix as  $\mathbf{W}_k$ . Tibshirani *et al.* (2001) recently proposed another heuristic approach for estimation  $K$ . They propose the *Gap statistic* which compares the curve for  $\log \Omega_k$  obtained from the dataset to that obtained using uniform data over the input space. Here  $\Omega_k$  is any within-cluster dissimilarity measure obtained by partitioning the dataset into  $k$  clusters. For instance, one may use  $|\mathbf{W}_k|$  as the measure when the underlying groups are assumed to be homogeneous and Gaussian. The Gap statistic estimates the optimal number of clusters  $K$  to be that value for which the difference or the “gap” is the largest. Unlike several other competing methods, their illustrations show that this measure works well even when there is only one underlying cluster in the population.

### 5.3 Application to Massive Datasets

Most clustering algorithms are not feasible to implement when the dataset is massive. Among the methods described above, SOM is the only approach largely unaffected but even then we need to know the lower-dimensional manifold on which to project the data. In many cases, such as in data visualization, the dimensionality is set to be 2 or 3. The coordinate system can then be decided, perhaps by taking a sample from the dataset and computing the principal components plane. Online clustering can then proceed. However, this depends on the objective of clustering and is not always the best approach for a given situation. We discuss here approaches to clustering such massive datasets.

The earliest statistical attention to the need for clustering large datasets was in the context of a hierarchical algorithm developed by McQuitty and Koch (1975) for a dataset of a thousand observations. A subsequent attempt was made by formulating the problem in terms of data reduction (Bruynooghe, 1978; Zupan, 1982). These papers however address the problem using definitions of a large dataset that are now largely obsolete. More recent attention has been in the context of clustering massive databases in the computer sciences literature (a few references are Agarwal *et al.*, 1998; Bradley *et al.*, 1997; Bradley *et al.*, 1998; Charikar *et al.*, 1997; Eddy *et al.*, 1996; Ester *et al.*, 1996; Ester *et al.*, 1998; Ganti *et al.*, 1999; Goil *et al.*, 2000; Huang, 1997; Guha, *et al.*, 1998; Han, *et al.*, 1998; Sheikholeslami *et al.*, 1998; Zhang, *et al.*, 1996). A number of algorithms try to obtain approximate partitions of the dataset and are not really practical for large datasets. I focus discussion here on the few statistical approaches developed.

Bradley *et al.* (1997, 1998) suggest an approach under the assumption that the number of clusters  $K$  is known in advance and that the group densities are multivariate normal. They first take a sample of observations and set up the likelihood-maximization problem in terms of an E-M algorithm. After clustering the first sample, the algorithm updates the estimated  $\pi_k$ 's via somewhat crude refinements derived from the remaining dataset in samples. The method is scalable, but inefficient. Further, the algorithm just identifies clusters based on a first-stage sample, and at subsequent stages only updates the estimated class proportions. This means that the benefits from such a huge number of observations are not fully utilized, and since the initial sample is necessarily small (for practicality of the above procedure), minority clusters are unlikely to be included and therefore identified.

Gordon (1986) provided a suggested road-map for clustering massive datasets. The basic thrust of his suggestion was to split the clustering problem into clustering and classification components and to devise a scheme whereby one would be able to figure out whether the groups had been adequately identified at the clustering stage. Although his paper has very few clear directions on implementation, Maitra (2001) developed and extensively tested a strategy similar in spirit. Under the assumption that all clusters have common dispersion  $\Sigma$ , the methodology first takes a subsample  $\mathcal{S}_1$  of size  $n$  from the dataset  $\mathcal{D}$  (decided by the available resources). This first sample is clustered and the cluster means and dispersion estimated, using  $K$ -means with  $K$  estimated via Marriott's (1971) criterion. Writing  $K_0$  as the estimate for  $K$ ,  $\pi_k^0$ 's,  $\mu_k^0$ 's and  $\Sigma^0$ 's for the estimates of cluster proportions, means and the common dispersion, the next step is to perform a hypothesis test on each of the remaining

members of the dataset where the hypothesis to be tested is whether the observation can reasonably be classified into the clusters so far identified. Formally therefore, the null and the alternative hypotheses to be tested for a given observation  $\mathbf{X}_i$  are

$$H_0 : \mathbf{X}_i \sim \sum_{k=1}^{K_0} \pi_k^0 \phi(\mathbf{x}; \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}^0) \quad vs. \quad H_1 : \mathbf{X}_i \sim \sum_{l=1}^{L_0} \pi_l^* \phi(\mathbf{x}; \boldsymbol{\mu}_l^*, \boldsymbol{\Sigma}^0), \quad (2)$$

where  $L_0 = K_0 + 1$ ,  $\pi_l^*$ 's and  $\boldsymbol{\mu}_l^*$ 's are the class probabilities and class centers under  $H_1$ ,  $\{\boldsymbol{\mu}_k^0; k = 1, 2, \dots, K_0\} \subset \{\boldsymbol{\mu}_l^*; l = 1, 2, \dots, L_0\}$ . In this setting, a likelihood-ratio-type test statistic is given by

$$\Lambda_0 = \min(1, |\boldsymbol{\Sigma}^0|^{-\frac{1}{2}} \sum_{k=1}^K \pi_k^0 \phi(\mathbf{x}; \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}^0)). \quad (3)$$

The null distribution of this test statistic can be obtained by simulation. The above hypothesis is tested at significance levels from a set of candidate values of  $\alpha$ . For each  $\alpha$ , if the proportion of observations rejected is less than  $\alpha$ , then it is clear that random chance can explain the extreme observations in the dataset so all clusters are declared to have been identified and the *cluster identification stage* is terminated. Otherwise, the observations for the largest significance level  $\alpha$  for which the proportion of rejects is greater than  $\alpha$  form the set  $\mathcal{D}_1$  under consideration at the next stage. The  $\pi_k^0$ 's are scaled to sum to the proportion accepted, and a sample  $\mathcal{S}_1$  of size  $n$  drawn from the second stage dataset  $\mathcal{D}_1$ . The sample is again clustered and the entire procedure iterated until every observation in the dataset has either been included in a sample to be clustered at some stage or has been in the acceptance region for one of the hypothesis tests. Let  $K = \sum_i K_i$ ,  $\{(\boldsymbol{\mu}_k^*, \pi_k^*); k = 1, 2, \dots, K\} = \bigcup_i \{(\boldsymbol{\mu}_k^i, \pi_k^i); k = 1, 2, \dots, K_i\}$  be the number of clusters, the cluster means and the cluster proportions at the end of this stage. This ends the cluster-identification stage of the algorithm. A classification of all observations into these clusters provides final estimates of the class proportions and the empty clusters are eliminated. A final step classifies the observations into the clusters based on these final estimates of class proportions. The algorithm is scalable and when tested on a range of examples, of datasets with  $N$  ranging from 500,000 to over 77 million, terminated in no more than five stages.

Maitra (2001) illustrated and tested this methodology extensively through a set of simulation experiments. Before discussing an experiment, I mention a measure to evaluate performance. One measure with known densities is the Kullback-Leibler-type divergence measure  $\kappa(f, \hat{f}) = \int_{\mathbf{x}} |\log f(\mathbf{x}) - \log \hat{f}(\mathbf{x})| f(\mathbf{x}) d\mathbf{x}$ , with  $f(\mathbf{x})$  and  $\hat{f}(\mathbf{x})$  as the true and estimated mixture densities respectively. Another option is the dissimilarity measure between the original true partition and the partition formed by the clustering algorithm, adapted from Mirkin and Chernyl (1970), and Arabie and Boorman (1973). This partition-based measure is constructed for a total number of  $N$  objects as follows: Out of the  $N(N - 1)/2$  possible pairs of objects, let the disagreement  $\Delta$  be the total number of pairs which are either classified together in the true partition but in different groups by the clustering algorithm or vice-versa. Then  $\mathcal{M} = 2\Delta/N(N - 1)$  is a normalized measure of disagreement between two partitions.  $\mathcal{M}$  is complementary to the similarity measure (equal to  $1 - \mathcal{M}$ )

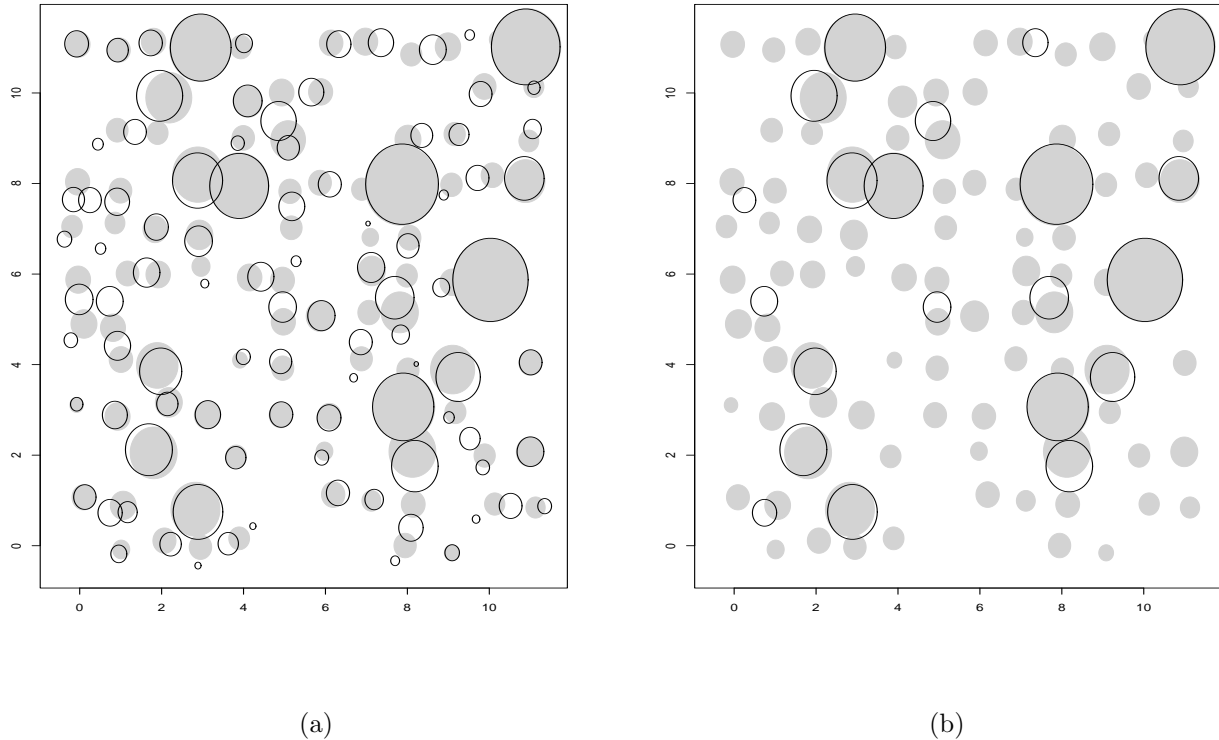


Figure 5: True cluster means and proportions (filled circles) and the estimates (unfilled circles) for test case 1, obtained using (a) the multi-stage clustering algorithm, and (b) clustering a sample from the dataset. The circles are proportional in area to the square root of the class proportions.

proposed by Rand (1971). The experiment (Maitra, 2001) I discuss here is on a sample of size 500,000 from a mixture of 94 bivariate normal populations with different means and class proportions but equal dispersions ( $\sigma_1 = \sigma_2 = 0.1$ ;  $\rho = 0.5$ ). The mixing proportions ranged from the very small ( $1.8 \times 10^{-4}$ ) to the moderate (0.17). The clusters are graphically shown in Figure 5 (filled circles). The initial starting points for the  $K$ -means algorithm at each stage were chosen to be the  $K$  highest local modes of the given stage sample. The multi-stage algorithm terminated in only four stages of sampling and clustering. Ninety-five groups were identified and the final estimated standard deviations were 0.225 and 0.238 with correlation estimated at 0.305. Figure 5a (unfilled circles) provides a graphical display of the performance of the algorithm: note that a large number of the clusters have been correctly identified. (The area of the circles in the figure is proportional to the square root of the class probabilities in order to highlight the smaller clusters). Figure 5b (unfilled circles) displays the 21 clusters identified by just clustering a small sample of size 2,500 from the original dataset. Interestingly, the strategy fails to identify even a few medium-sized clusters. The Kullback-Liebler-type divergence measure was computed to be 0.48 for the multi-stage clustering scheme and 1.25 for the alternative. Partitioning of the observations

as per the former disagreed with the truth for only  $\Delta = 188,037,120$  pairs of cases, so that  $\mathcal{M} = 0.0015$ . Disagreement was more emphatic for the case of the alternative algorithm with  $\Delta = 1,468,538,880$  and  $\mathcal{M} = 0.0089$ . Thus, relative to the latter algorithm, the multi-stage clustering strategy improved classification rates by about 83.1%.

This above approach provides the first real attempt to address the problem of clustering massive datasets in a comprehensive statistical framework. It is not necessary to use  $K$ -means while clustering the sample at each stage: other approaches can also be used. The assumption on common dispersion for all clusters is important in the derivation of the likelihood ratio test of (2) for testing the representativeness of the identified clusters based on each individual observation. Some very limited generalization (such as a known relationship of the cluster dispersions with the cluster means) is possible and exhibited but those are useful only in some special cases. We discuss possible extensions and detail areas requiring further attention in a short while.

## 5.4 Application to the Clustering of Software Records

This illustration is also provided in Maitra (2001). Once again,  $K$ -means was used in clustering the sub-samples. The dataset was first pre-processed by removing those procedures without any metrics. These procedures had perhaps not yet been integrated into the package. This reduced the total number of records to 214,982. Non-informative metrics (or metrics with common readings for all procedures) were then ignored from our consideration, thus bringing down the dimensionality from 70 to 32. All metrics were then standardized to the same scale and the multi-stage clustering algorithm applied. At each stage, the sample size was set to be  $n = 2,500$ . Initial guesses for the  $K$ -means algorithm were chosen for each  $k$  from the means of the  $k$  most homogeneous groups obtained through hierarchical clustering with Euclidean distance, and the optimal  $K$  was determined by Marriott's criterion. The multi-stage clustering algorithm terminated after two stages and identified 94 clusters. With this grouping, the 94 classes for the procedures were used for software maintenance and upgrades. Moreover, some of these procedures were themselves found to be defective, and the grouping was also used to identify other components that needed to be updated while fixing those bugs.

## 5.5 Unresolved Issues

There are several aspects of clustering that need further attention. Some of these issues are general and pertain to all sizes of datasets: others are more specific and arise because of the size of the dataset or because of the type of data. Indeed, the breadth of the unresolved issues underlines the difficult nature of the problem.

Hartigan's (1975) implementation of the  $K$ -means algorithm is sub-optimal and known to be unstable. One of the issues in the  $K$ -means algorithm is the choice of initial values for the cluster centers. The algorithm then finds a local minimum of the chosen penalty function. Bradley and Fayyad (1998) develop an algorithm in an attempt to provide a robust way of

obtaining the cluster centers. Assuming  $K$  known, the strategy consists of obtaining several subsamples of the dataset and clustering each sub-sample via  $K$ -means and an arbitrary set of initialization values. The cluster centers,  $\boldsymbol{\mu}_c$ 's obtained from each sub-sample are then clustered again via  $K$ -means, with initial values of the cluster centers given by cluster centers from each sub-sample. Each exercise with initial points  $\boldsymbol{\mu}_c$ 's gives a set of cluster means  $\boldsymbol{\mu}_c^*$ 's and the final initial points for the  $K$ -means algorithm are chosen from amongst this set as the one which is closest in a least-squares sense to all the  $\boldsymbol{\mu}_c$ 's. The goal of these suggestions is to obtain initial values for the  $K$ -means algorithm. When the sub-sample has the same size as the sample, the above method is identical to running the  $K$ -means algorithm with randomly chosen start-points, and then replicating this experiment several times. Note that the sub-samples are not necessarily independent and it is not clear how this affects the clustering strategy of the initially chosen cluster means  $\boldsymbol{\mu}_c$ . Additionally, knowledge of  $K$  is critical. Further, this strategy only addresses the issue of initial points for the  $K$ -means algorithm. The important issue of noise and variability in the observations is not addressed, and there is need for robust clustering methodology against noise/random perturbations.

The multi-stage clustering scheme developed by Maitra (2001) is severely restricted by the assumption of underlying homogeneous Gaussian clusters. Extending the methodology is important, and we note that the major stumbling block is the estimation of the dispersion matrix under the global condition for the likelihood ratio test for testing each observation.

Clustering datasets on the basis of indirectly observed data is another important problem that has not received much attention, even though the Panel on Discriminant Analysis and Clustering (1989) mentioned this as an important area requiring statistical attention. The context is that  $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$  are not directly observed, but a projection is observed. Therefore, we have a sample  $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_N$ , where  $\mathbf{Y}_i = \mathbf{A}\mathbf{X}_i$ . For this proposal,  $\mathbf{A}$  is a linear transformation with full column rank. The objective then is to obtain estimates for  $K$ ,  $\boldsymbol{\mu}_K$ ,  $\boldsymbol{\Sigma}_K$ , without going through a potentially computationally expensive inverse problem, as would happen for very large  $N$ . Such functionality would be very useful in the context of segmenting images that are indirectly observed, for example, in Positron Emission Tomography (PET) — see Maitra and O'Sullivan (1998) or Maitra (2001) for detailed discussions on the application.

Another area of interest with regard to clustering is when all coordinate-values are not all available together, but become available incrementally. That is, we first have observations only on the first coordinate  $X_{11}, X_{21}, \dots, X_{N1}$  followed by observations on the second coordinate  $X_{12}, X_{22}, \dots, X_{N2}$  and so on up to time  $p$ . Writing  $\mathbf{X}_i = \{X_{i1}, X_{i2}, \dots, X_{ip}\}; i = 1, 2, \dots, p$ , our goal is to cluster the time series  $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N$  without actually having to wait for all the observations to be recorded. In other words, we propose to cluster the  $\mathbf{X}_i$ 's based on partially available information and then to sequentially update our clusters with new available data on the other coordinates. This would be desirable in the context of the PET set-up also. This is because when the duration of the study is long, it is not feasible to segment voxels based on the complete set of observations. In such a case, it would be desirable to segment observations based on the first few time-points and then to update the segmentation sequentially with the observations from subsequent time-bins.

Another area of interest is the case when the observations arrive sequentially. This is important in the context, for example, of streaming electronic transactions. Businesses need to identify and categorize transactions because marketing campaigns can identify and tailor programs designed for the different groups. The context then is that we have an infinite stream of observations  $\mathbf{X}_1, \mathbf{X}_2, \dots$  from where we want to estimate the number of clusters  $K$ , the cluster centers  $\boldsymbol{\mu}$ 's and the dispersions  $\boldsymbol{\Sigma}$ 's, and also categorize the observations into these clusters.

Thus, we see that several issues in clustering datasets need attention in the statistical literature. Many of the approaches that exist are empirical and somewhat ad hoc, pointing to the difficult nature of the problem, while at the same time underlining the absence of as well as the need for coordinated statistical methodology.

## 6 Information Retrieval and Latent Semantic Indexing

Search engines available on the Worldwide Web (WWW) very nicely illustrate the need for information retrieval. For instance, it is very common to use a web-based search engine to find articles pertaining to a particular topic. Typing in the word “laptop” for instance should bring up all articles in the database containing that word. This is what one gets by a lexicographic search. However, when searching for documents on the word “laptop”, the user is most likely interested not only in articles containing the word “laptop” but also in articles on portable computers which choose to use some other word (such as “notebook”) to refer to portable computational equipment. This aspect would be missing in a simple lexical-match-based search strategy.

How would one provide a search strategy incorporating such synonyms? One approach would be to put together an exhaustive list of synonyms and refer to all articles containing at least one of every synonym for the desired word. However, the database has a large number of words (and with particular reference to the WWW, this list is dynamic) and besides, the same word can be used in more than one context. To elucidate the last point, note that an user who types in the word “Jaguar” would get an article on cars, the Anglo-French fighter aircraft as well as the member of the feline family. This aspect of a word being used in many different concepts is called polysemy and may be alleviated somewhat by using multiple-search words such as “Jaguar cars” to promote context. Once again, note that a lexical match of the word pair would not yield any information on cars similar to any of the Jaguars in terms of performance and/or price. An approach, called *relevance feedback* (Rocchio, 1971) seeks to provide the user with a list of documents and ask for feedback on the relevance of the list to the query.

A statistical approach called Latent Semantic Indexing (LSI) was developed (Deerweuster *et al.*, 1990) to allow for retrieval on the basis of concept, rather than an exact lexical match. The approach uses statistically derived conceptual indices to search the database. The basic idea is to write a term-document matrix, containing the frequency of occurrences of each term in each document. Let us assume that there are  $D$  documents in the database, and the documents all contain a total of  $T$  unique keywords (or the words of interest in the database).

For instance, keywords could be all words in the database, excluding articles, prepositions, and conjugations. Denote the term-document frequency matrix by  $\mathbf{A} = ((a_{td}))$  where  $a_{td}$  is the number of times that the  $t$ 'th keyword occurs in the  $d$ 'th document. LSI uses a  $k$ -rank approximation of  $\mathbf{A}$  using its singular value decomposition (SVD). To see this, suppose that  $\text{rank}(A) = r$  and that the SVD of  $A$  is given by  $\mathbf{A} = \mathbf{U}\mathbf{D}_\lambda\mathbf{V}'$  with  $\mathbf{U}$ ,  $\mathbf{V}$  being  $T \times r$  and  $D \times r$  matrices respectively. Here  $\mathbf{U}'\mathbf{U} = \mathbf{I}_r$ ,  $\mathbf{V}'\mathbf{V} = \mathbf{I}_r$  and  $\mathbf{D}_\lambda$  is a diagonal matrix of the singular values  $\lambda = \{\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r\}$  of  $A$ . Alternatively, write  $\mathbf{U} = [\mathbf{u}_1 : \mathbf{u}_2 : \dots : \mathbf{u}_r]$  and  $\mathbf{V} = [\mathbf{v}_1 : \mathbf{v}_2 : \dots : \mathbf{v}_r]$ , then  $\mathbf{A} = \sum_{i=1}^r \lambda_i \mathbf{u}_i \mathbf{v}_i'$ . Then the  $k$ -rank approximation to  $\mathbf{A}$  is given by  $\mathbf{A}_k = \sum_{i=1}^k \lambda_i \mathbf{u}_i \mathbf{v}_i'$ . The goal of the reduced-rank representation  $\mathbf{A}_k$  is to capture the salient association structure between the terms and the documents while at the same time eliminating a substantial part of the noise and variability associated with word usage. Note that SVD decomposes  $\mathbf{A}$  into a set of  $k$  uncorrelated indexing variables, with each term and document represented by a vector in a  $k$ -dimensional space using the elements of the left or right singular vectors.

A query search for documents pertaining to a particular set of terms can be represented as a vector of 0's and 1's, with 1 indicating presence of a keyword in the given query and 0 otherwise. A query can be considered to be a document denoted by the  $T \times 1$  vector  $\mathbf{q}$ . To find out documents close to this query, we need to transform both query and documents into the same space and pick the transformed documents closest to the query in the reduced space. Note that the reduced-rank representation of  $\mathbf{A}$  means that any document  $\mathbf{d}$  in the database can be represented as  $\mathbf{U}'_k \mathbf{\Lambda}_k \mathbf{d}_k$  so that the document maps to the reduced-document space as  $\mathbf{d}_k = \mathbf{\Lambda}_k^{-1} \mathbf{U}'_k \mathbf{d}$ . This is the row of  $\mathbf{V}_k$  corresponding to the document  $\mathbf{d}$ . Similarly, any query  $\mathbf{q}$  transforms to the vector  $\tilde{\mathbf{q}} = \mathbf{\Lambda}_k^{-1} \mathbf{U}'_k \mathbf{q}$ , where  $\mathbf{\Lambda}_k = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_k)$ . To calculate documents close to the query, we calculate the cosine between the vectors  $\tilde{\mathbf{q}}$  and  $\mathbf{d}_k$  for each document  $\mathbf{d}$  and use that to provide the documents most relevant to a search. The search returns all those documents with cosine measure above a user-defined threshold value  $\alpha$ . We now illustrate this feature using a simple example.

## 6.1 An Illustration

Consider a database containing the titles of the 16 statistics books given in Table 6.1. In our database, we have used as keywords all those words that appear more than once in each title. Thus, we have the keywords: Generalized (T1), Linear (T2), Model (T3), Theory (T4), Inference (T5), Application (T6), Regression (T7), Introduction (T8), Analysis (T9), Multivariate (T10) and Nonparametric (T11). Note that we have associated words such as "Applied" and "Applications" or "Multivariate" and "Multivariable" to the same keywords. Also we have not included "statistics" as a keyword since all the titles here pertain exclusively to that subject. Table 6.1 yields the term-document matrix provided in Table 6.1. The matrix is decomposed using SVD. For the reduced-rank representation, we use  $k = 2$  so that the application of LSI on this example can be represented graphically. Figure 6 is a two-dimensional plot of the terms and the documents in the reduced space. The two columns of  $\mathbf{U}_2 \mathbf{\Lambda}_2$  are plotted against each other on the  $x$ - and  $y$ -axes respectively. Similarly

Table 4: List of statistics books used in our LSI example. The underlined terms are the keywords used for searching the documents (book titles).

---

D1	<u>Generalized Linear Models</u>
D2	<u>Theory of Statistical Inference</u>
D3	<u>Applied Linear Regression</u>
D4	<u>Linear Statistical Inference</u> and Its Applications
D5	<u>Applied Multivariate Statistical Analysis</u>
D6	<u>Theory and Applications of the Linear Model</u>
D7	<u>Introduction to Linear Regression Analysis</u>
D8	<u>Multivariate Analysis</u>
D9	<u>An Introduction to Multivariate Statistical Analysis</u>
D10	<u>Theory of Multivariate Statistics</u>
D11	<u>Applied Linear Regression Analysis</u> and <u>Multivariable Methods</u>
D12	<u>The Analysis of Linear Models</u>
D13	<u>Applied Regression Analysis</u>
D14	<u>Applied Linear Regression Models</u>
D15	<u>Nonparametric Regression</u> and <u>Generalized Linear Models</u>
D16	<u>Multiple and Generalized Nonparametric Regression</u>

---

Table 5: The  $11 \times 16$  term-document matrix corresponding to the database in Table 6.1.

<b>Terms</b>	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16
Generalized	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Linear	1	0	1	1	0	1	1	0	0	0	1	1	0	1	1	0
Model	1	0	0	0	0	1	0	0	0	0	0	1	0	1	1	0
Theory	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0
Inference	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Application	0	0	1	1	1	1	0	0	0	0	1	0	1	1	0	0
Regression	0	0	1	0	0	0	1	0	0	0	1	0	1	1	1	1
Introduction	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
Analysis	0	0	0	0	1	0	1	1	1	0	1	1	1	0	0	0
Multivariate	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0
Nonparametric	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

for the transformed documents, the two columns of  $\mathbf{V}_2\mathbf{\Lambda}_2$  are plotted on the  $x$ - and  $y$ -axes respectively. Note the natural clustering of terms and documents. Thus, the documents D5, D8, D9 and D10 all cluster together and the keyword “multivariate” may be used to describe these titles. The documents D1, D15 and D16 all pertain to “generalized” and “models” while the documents D3, D4, D6, D12 and D14 all cluster together and are close to the keywords “linear” and “regression”. The terms “theory” and “inference” are close to each other, as also “nonparametric” and “generalized” or “linear” and regression”. I now exhibit query search using LSI. Suppose we want to choose titles pertaining to “linear models”. The query vector  $\tilde{\mathbf{d}}$  in the two-dimensional transformed-space is given by (0.211,.122) — illustrated by the “x” in Figure 6. To find documents in the database most relevant to our search, we calculate the cosine of the angle formed by  $\tilde{\mathbf{d}}$  and  $\mathbf{d}_k$  for each document vector  $\mathbf{d}$ . Only the transformed document vectors for D1, D3, D4, D6, D12, D14, D15 and D16 have cosines of over 0.9 with  $\tilde{\mathbf{q}}$ . Note that these include titles on regression when the query did not include this as a keyword. Note also that lexical matching would not have included D16, for instance. It is often common to scale the term-document matrix in order to down-weight common terms. The SVD is then performed and LSI implemented on this weighted term-document matrix. Although we did not use such a strategy in our illustration here, this strategy can be readily implemented within the framework mentioned above.

## 6.2 Updating the Database

In a dynamic database such as on the WWW, new documents and terms are being added over time. As new documents and terms come in, the database and indeed the reduced- $k$ -rank representation needs to be updated. Of course, one may repeat the above exercise for every update of the database. However, this is a very expensive process requiring constant recomputation. Moreover for large databases, the SVD decomposition is in itself very computationally expensive, even for extremely sparse matrices. One option is to use what is called *folding-in* of terms and documents. This approach assumes that the SVD done on the original database is adequate for the expanded database. In other words, the new term and document vectors are just “folded in” into the existing database. To illustrate, a new document vector  $\mathbf{d}^*$  is mapped, using the arguments given above, to  $\mathbf{d}_k^* = \mathbf{\Lambda}_k^{-1}\mathbf{U}'_k\mathbf{d}^*$ , while a new term vector  $\mathbf{t}^*$  is similarly projected to  $\mathbf{t}_k^* = \mathbf{\Lambda}_k^{-1}\mathbf{V}'_k\mathbf{t}^*$  in the originally reduced term-space. The new vectors thus formed are however not orthogonal to the others or to each other. Folding-in also has no effect on existing term and document vectors and therefore, the approach does not learn about context from new terms and documents in the expanded database. All these factors mean that while folding-in may work for a few additional terms and documents, it would be inefficient with substantially new database.

Another approach to this problem is called SVD-updating (Berry *et al.*, 1994; Berry *et al.*, 1999). This approach assumes that the term-document matrix for the existing database  $\mathbf{A}$  is represented through  $\mathbf{A}_k$ . Matrix operations are then used to compute the SVD of  $\mathbf{A}_k$  augmented with the new columns from the additional documents  $\mathbf{D}$  and the new terms  $\mathbf{T}$ . To elucidate, suppose for the sake of simplicity that the new term-document matrix only

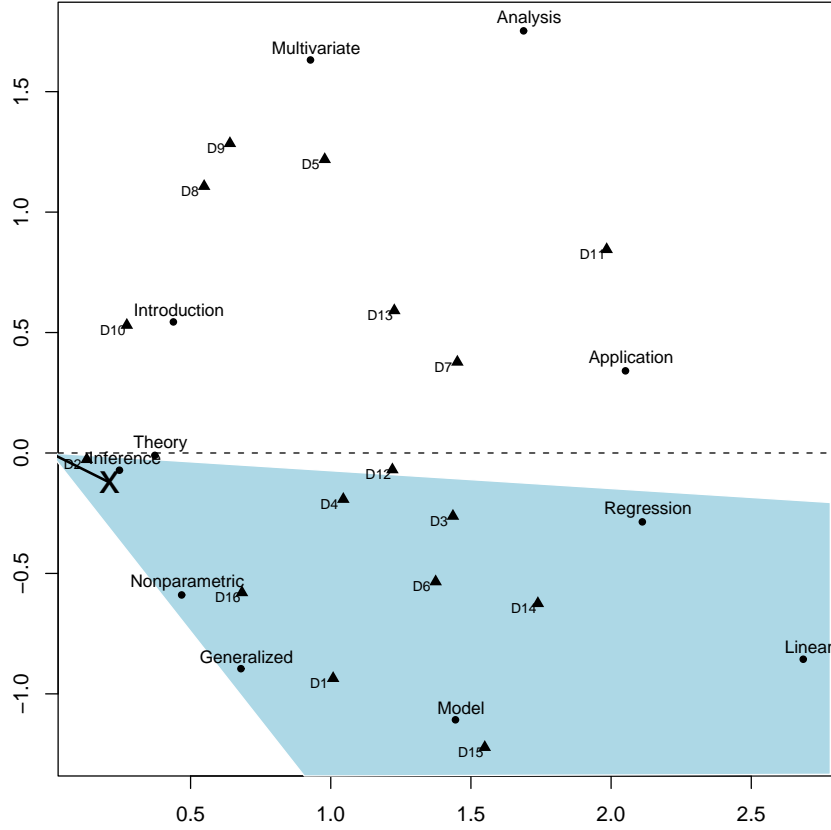


Figure 6: The two-dimensional plot of terms and documents for the example in Table 6.1. The point “x” denotes the location of the query “linear models” in the transformed space. The shaded region represents the region containing the documents which in the reduced document-space have cosines of over 0.9 with the transformed query vector.

includes a new set of documents (and no new terms) and is given by  $[\mathbf{A}|\mathbf{D}]$ . Here  $\mathbf{D}$  is a  $T \times P$  matrix. Replacing  $\mathbf{A}$  by its reduced-rank version  $\mathbf{A}_k$ , the SVD of  $\mathbf{A}^* = [\mathbf{A}_k|\mathbf{D}]$  is computed in terms of  $\mathbf{A}^* = \mathbf{U}^* \mathbf{\Lambda}^* \mathbf{V}^{*t}$ . It can be shown (Berry *et al.*, 1995; Berry *et al.*, 1999) that  $\mathbf{U}^* = \mathbf{U}_k \mathbf{U}_*$ ,  $\mathbf{V}^* = \mathbf{V}_\ominus \mathbf{V}_*$  and  $\mathbf{\Lambda}^* = \mathbf{\Lambda}_*$  where  $\mathbf{A}_* = [\mathbf{\Lambda}_k | \mathbf{U}_k^t \mathbf{D}]$ ,  $\mathbf{A}_* = \mathbf{U}_* \mathbf{\Lambda}_* \mathbf{V}_*^t$  in terms of its SVD and  $\mathbf{V}_\ominus$  is the block-diagonal matrix formed by the matrices  $\mathbf{V}_k$  and  $\mathbf{I}_P$ . This reduces computations for the SVD but note that the rank of the new matrix is limited by at most  $k$ . A similar approach may be used for adding new terms. Once again, suppose that we are only incorporating new terms and no new documents in the expanded database (as a consequence, say, of documents that have been updated). Then the new term-document matrix is given by  $\mathbf{A}^\bullet = \begin{bmatrix} \mathbf{A}_k \\ \mathbf{D} \end{bmatrix}$ . Here  $\mathbf{D}$  is a  $Q \times D$ . Then we replace  $\mathbf{A}$  by its reduced-rank version  $\mathbf{A}_k$  and compute the SVD of  $\mathbf{A}^\bullet = \begin{bmatrix} \mathbf{A}_k \\ \mathbf{D} \end{bmatrix}$  in terms of  $\mathbf{A}^\bullet = \mathbf{U}^\bullet \mathbf{\Lambda}^\bullet \mathbf{V}^{\bullet t}$ . Again, it is easy

to show that  $\mathbf{V}^\bullet = \mathbf{V}_k \mathbf{V}_\bullet$ ,  $\mathbf{V}^\bullet = \mathbf{V}_\circ \mathbf{V}_\bullet$  and  $\mathbf{\Lambda}^\bullet = \mathbf{\Lambda}_\bullet$  where  $\mathbf{A}_\bullet = \left[ \frac{\mathbf{\Lambda}_k}{\mathbf{T}\mathbf{V}_k} \right]$ ,  $\mathbf{A}_\bullet = \mathbf{U}_\bullet \mathbf{\Lambda}_\bullet \mathbf{V}_\bullet$  in terms of its SVD and  $\mathbf{V}_\circ$  is the block-diagonal matrix formed by the matrices  $\mathbf{U}_k$  and  $\mathbf{I}_Q$ . For the case when both new terms and documents are added to the database, the above can be implemented in a two-step approach. Note that while the computations here are considerably more than for *folding-in*, they are far less than for computing the SVD of the entire database. A primary contribution to the expense in SVD-updating computations as described above is the fact that the  $\mathbf{U}$ 's and  $\mathbf{V}$ 's are dense matrices.

### 6.3 Other Issues

We conclude this section with a discussion on some of the many interesting statistical issues unresolved. A primary unresolved issue pertains to the choice of  $k$ . In our application above, we chose  $k = 2$  for ease of representation of our results. There is really no known method for choosing  $k$ . One of the issues in the choice of  $k$  is the absence of a proper statistical representation of the LSI framework, even though Ding (1999) took the first steps to providing a probability model representation of LSI. A appropriate statistical model may be used together with a penalty function to decide on  $k$ . Another issue concerns noise and variability. There are two aspects here: one pertains to terms that are corrupted as a result of typographical errors. For instance, “nonparametric” may be mis-spelt as “nonparametric” in one of the documents. This is noise in the database and needs to be modeled separate from variability which arises when one has a choice of synonyms (for example, “war” and “conflict”) and chooses to use one of them. It may be useful to incorporate within LSI, a probability model on the basis of usage of these synonyms. Another area in which there has been considerable work, and where LSI has been used, concerns the issue of retrieval across multi-lingual databases (Landauer and Littman, 1990; Littman *et al.*, 1998). SVD-updating has been modified to accommodate the above context. Finally, computation continues to be very serious constraint. An approach may be to update the SVD of a database incrementally, using a combination of folding-in and SVD-updating, perhaps using a statistical model and a multi-stage approach. Thus, we see that there are substantial issues in information retrieval requiring statistical attention.

## 7 Discussion

This paper discusses some of the major areas associated with the fast emerging discipline of data mining. As we can see, a number of these aspects have received statistical attention in a variety of ways. In some cases, methodology has been built, perhaps somewhat unconventionally, atop existing statistical machinery. However I note that there are avenues for substantial further research in almost all the aspects discussed in this paper. A common theme in many of these cases pertains to mining and knowledge discovery in massive databases: however, as we have mentioned in this paper, these are not the only issues. Knowledge discovery, learning of the different processes, the understanding of causal and

associative relationships, whether in small or large databases, is an important goal. Sound statistical procedures exist, but quite often, under very restrictive procedures which are not always appropriate for modern databases. There is a need to extend such methodology for more general situations.

In passing, I discuss an area not traditionally associated with data mining, but which has been very much an issue in multivariate statistics. I believe the issue of dimensionality reduction has a greater role in analyzing large, irregular or noisy databases. In the statistical literature, we have available several approaches to this problem: some commonly used techniques are Principal Components Analysis (PCA), Canonical Correlation Analysis (CCA) and Multi-dimensional Scaling (MDS) (Mardia *et al*, 1979), Principal Curves (Hastie and Stuetzle, 1989), Projection Pursuit (Huber, 1985). In recent years, there has been development of a technique called Independent Components Analysis (ICA) which separates signals into independent components (Hyvarinen and Oja, 2000). The approach is similar to PCA for Gaussian data but finds independent components in non-Gaussian data, thereby removing a major shortcoming in the application of PCA to non-Gaussian data.

At some level, data mining ultimately needs automated tools to gain insight into relationships from databases. A large number of tools are commercially available and often sold as a panacea for all problems. Indeed, these tools are often touted as a substitute for expert human interaction and analyses. In substance, many of these tools are front-ends for very common and existing statistical tools packaged under effective naming strategies: it is unlikely — and in my experience erroneous to believe — that human expertise can ever be substituted by a set of automated learning tools, especially in the analysis of complex processes and complicated datasets. I believe that data mining tools will be invaluable in accessing records, especially in complex databases. However from the point of view of understanding and gaining insights into databases, knowledge discovery would perhaps be better served by considering these tools (as we do with statistical software) as some more useful aid in the task ahead.

## References

- [1] Agarwal R., Imielinski T., and Swami A. (1993). Mining association rules between sets of items in large databases. In *Proc. of ACM SIGMOD Conference on Management Data*, 207-16, Washington, DC.
- [2] Agarwal, R., Gehrke, J., Gunopulos, D. and Raghavan, P. (1998). Automatic subspace clustering of high-dimensional data for data mining applications. In *Proc. of ACM SIGMOD International Conference on Management of Data*.
- [3] Aggarwal C. C. and Yu, P. S. 1998. Mining Large Itemsets for Association Rules. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*. 2. 1, 23-31.
- [4] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Trans. Automatic Control*, 19:716-723.

- [5] Aliferis, C. and Cooper, G. (1994). An evaluation of an algorithm for inductive learning of Bayesian belief networks using simulated datasets. In *Proc. Tenth Conf. on Uncertainty in Artificial Intelligence*, 8-14. San Francisco: Morgan Kaufmann.
- [6] Arabie, P. and Boorman, S. A. (1973). Multi-dimensional Scaling of Measures of Distance Between Partitions. *J. Math. Psych.* 10:148-203.
- [7] Babcock B., Babu S., Datar M., Motwani, R. and Widom, J. 2002. Models and Issues in Data Stream Systems. *Invited paper in Proc. of the 2002 ACM Symp. on Principles of Database Systems (PODS 2002), to appear.*
- [8] Banfield, J. D. and Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics* 49:803-21.
- [9] Beckett, J. (1977). Clustering Rank Preference Data. *ASA Proc. Soc. Stat.* 983-86.
- [10] Bellman, R. E. (1961). *Adaptive Control Processes*. Princeton University Press.
- [11] Berry, M. W., Dumais, S. T. and O'Brien, G. W. (1994). Using linear algebra for Intelligent Information Retrieval. *SIAM Review*, 37:4:573-595.
- [12] Berry, M. W., Drmac, Z., Jessup, E. R. (1999). Matrices, vector spaces, and information retrieval. *SIAM Review*, 41:2:335-362.
- [13] Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems (with discussion). *J. Roy. Stat. Soc. B* 36:192-236.
- [14] Bozdogan, H. and Sclove, S. L. (1984). Multi-sample Cluster Analysis Using Akaike's Information Criterion. *Ann. Inst. Stat. Math.* 36:163-180.
- [15] Bradley, P. S. and Fayyad, U. M. (1998) Refining Initial Points for K-Means Clustering. *Proceedings of the Fifteenth International Conference on Machine Learning ICML98, pages 91-99. Morgan Kaufmann, San Francisco.*
- [16] Bradley, P., Fayyad, U. and Reina, C. (1997) Scaling EM(Expectation-Maximization) Clustering to Large Databases. *Technical Report, MSR-TR-98-35, Microsoft Research.*
- [17] Bradley, P., Fayyad, U. and Reina, C. (1998) Scaling clustering algorithms to large databases. In *The Fourth International Conference on Knowledge Discovery and Data Mining, New York City.*
- [18] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:51-64.
- [19] Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth.
- [20] Brin S., Motwani R. and Silverstein C. (1997a). Beyond market baskets: generalizing association rules to correlations. *Proc. of the ACM SIGMOD*, 265-276.

- [21] Brin S., Motwani R., Ullman J. D. and Tsur S. (1997b). Dynamic itemset counting and implication rules for market basket data. *Proc. of the ACM SIGMOD*, 255-264.
- [22] Brook, D. (1964). On the distinction between the conditional probability and the joint probability approaches in the specification of nearest-neighbor systems. *Biometrika*, 51:481-3.
- [23] Brossier, G. (1990) Piece-wise hierarchical clustering. *J. Classification* 7:197-216.
- [24] Bruynooghe, M. (1978). Large Data Set Clustering Methods Using the Concept of Space Contraction. In *COMPSTAT, Third Symp. Comp. Stat. (Leiden)* 239-45. Physica-Verlag (Heidelberg)
- [25] Buntine, W. (1994). Operations for learning with graphical models. *J. Artificial Intelligence Research*, 2:159-225.
- [26] Buntine, W. (1996). Graphical models for discovering knowledge. In *Advances in Knowledge Discovery and Data Mining (Fayyad, Piatetsky-Shapiro, Smyth and Uthurusamy, Eds.)*, 59-82.
- [27] Can, F. and Ozkaran, E. A. (1984). Two partitioning-type clustering algorithms. *J. Amer. Soc. Inform. Sci.* 35:268-76.
- [28] Charniak, E. (1991). Bayesian networks without tears. *AI Magazine*, 12:50-63.
- [29] Charikar, M., Chekuri, C., Feder, T. and Motwani, R. (1997) Incremental Clustering and Dynamic Information Retrieval. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 144-155.
- [30] Cheeseman, P. and Stutz, J. (1996) Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining (Fayyad, Piatetsky-Shapiro, Smyth and Uthurusamy, Eds.)*, 153-182.
- [31] Chen, H., Gnanadesikan, R. and Kettenring, J. R. (1974). Statistical methods for grouping corporations. *Sankhya Ser. B* 36:1-28.
- [32] Cormack R. M. (1971). A review of classification. *J. Roy. Stat. Soc. Ser. A* 134:321-367.
- [33] Celeux, G. and Govaert, G. (1995) Gaussian parsimonious clustering models. *Patt. Recog.* 28:781-93.
- [34] Cooper G. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309-47.
- [35] Cristianini, N. and Shawe-Taylor, J. (2001). *An Introduction to Support Vector Machines (and other kernel-based learning methods)*, Cambridge University Press.

- [36] Day, N. E. (1969) Estimating the components of a mixture of two normal distributions. *Biometrika* 56:463-474.
- [37] Deerweester, S., Dumais, S., Furnas, G., Landauer, T. and Harshman, R. Indexing by latent semantic analysis. *J. Amer. Soc. Information Science*, 41:391-407.
- [38] Dick, N. P. and Bowden, D. C. (1973) Maximum likelihood estimation of mixtures of two normal distributions. *Biometrics* 29:781-790.
- [39] Ding, C. H. Q. (1999) A similarity-based probability model for Latent Semantic Indexing. *In Proc. of 22nd ACM SIGIR'99 Conf.*, 59-65.
- [40] *Discriminant Analysis and Clustering: Report of the Panel on Discriminant Analysis, Classification and Clustering.* In *Statistical Science* 4:34-69.
- [41] DuMouchel W. and Pregibon D. (2001). Empirical bayes screening for multi-item associations in massive datasets. In *Proceedings of Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA. ACM Press 67-76.
- [42] Eddy, W. F., Mockus, A. and Oue, S. (1996). Approximate single linkage cluster analysis of large datasets in high-dimensional spaces. *Comp. Stat. and Data Analysis* 23:29-43.
- [43] Ester, M., Kriegel, H. P., Sander, J. and Xu, X. (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD-96), Portland, OR, pp. 226-231.*
- [44] Ester, M., Kriegel, H. P., Sander, J. and Xu, X. (1996) Clustering for mining in large spatial databases. *Special Issue on Data Mining, KI-Journal, ScienTec Publishing, No. 1.*
- [45] Everitt, B. (1974). *Cluster Analysis*. Heineman Educational, London.
- [46] Everitt, B. S. (1988) A finite mixture model for the clustering of mixed-mode data. *Stat. Prob. Let.* 6:305-309.
- [47] Fisher R. A. (1936) The use of multiple measurements in taxonomic problems. *Eugenics*, 7:179-188.
- [48] Fowlkes, E. B., Gnanadesikan, R. and Kettenring, J. R. (1988). Variable selection in clustering. *J. Classification* 5:205-28.
- [49] Freund, Y. and Schapire, R. (1997). A decision-theoretic generalization of online learning and an application to boosting. *J. Computer and System Sciences*, 55:119-139.
- [50] Friedman, J. (1989) Regularized discriminant analysis. *J. Amer. Stat. Assoc.*, 84:165-175.

- [51] Friedman, J. (1991) Multivariate adaptive regression splines (with discussion). *Ann. Stat.*, 19(1):1-141.
- [52] Friedman, J., Hastie, T. and Tibshirani, R. (2001). Additive logistic regression: a statistical view of boosting (with discussion). *Ann. Stat.*, 28:337-307.
- [53] Friedman, J. and Stuetzle, W. (1981). Projection pursuit regression. *J. Amer. Stat. Assoc.*, 76:817-823.
- [54] Friedman, H. P. and Rubin, J. (1967). On some invariant criteria for grouping data. *J. Amer. Stat. Assoc.* 62:1159-78.
- [55] Frydenberg, M. (1990) The chain graph Markov property. *Scand. J. Stat.*, 17:333-353.
- [56] Ganti, V., Ramakrishnan, R., Gehrke, J., Powell, A. and French, J. (1999) Clustering large datasets in arbitrary metric spaces. In *Proc. of International Conference on Data Engineering*.
- [57] Gnanadesikan, R. (1977). *Methods for Statistical Data Analysis of Multivariate Observations*. John Wiley & Sons (New York; Chichester).
- [58] Goil, S., Harsha, A. C. (1999) Parallel subspace clustering for very large datasets. *Technical report CPDC-TR-9906-010, Northwestern University*.
- [59] Good, I. J. (1979). The clustering of random variables. *J. Stat. Computation and Simulation*, 9:241-3.
- [60] Gordon, A. D. (1986). Links between clustering and assignment procedures. *Proc. Comp. Stat.* 149-56. Physica-Verlag (Heidelberg).
- [61] Guha, S., Rastogi, R. and Shim, K. (1998) CURE: An efficient algorithm for clustering large databases. In *Proceedings of ACM-SIGMOD 1998 International Conference on Management of Data, Seattle*.
- [62] Hammersley, J. and Clifford P. (1971) Markov fields on finite graphs and lattices. *Unpublished manuscript*.
- [63] Han, E. H., Karyapis, G., Kumar, V. and Mobasher, B. (1998) Clustering in a high-dimensional space using hypergraph models. *Technical Report 97-019, Department of Computer Science, University of Minnesota*.
- [64] Hartigan, J. (1975) *Clustering Algorithms*. Wiley, New York.
- [65] Hartigan, J. (1985). Statistical Theory in Clustering. *J. Class.* 2:63-76.
- [66] Hartigan, J. A. and Wong, M. A. (1979). [Algorithm AS 136] A  $k$ -means clustering algorithm. *Applied Statistics*, 28:100-108.

- [67] Hastie, T., Buja, A., and Tibshirani, R. (1995). Penalized discriminant analysis. *Ann. Stat.*, 23:73-102.
- [68] Hastie, T. and Stuetzle, W. (1989). Principal curves. *J. Amer. Statist. Assoc.*, 84(406):502-516.
- [69] Hastie, T. and Tibshirani, R. (1996). Discriminant analysis by Gaussian mixtures. *J. Roy. Stat. Soc. Ser. B*, 58:155-176.
- [70] Hastie, T., Tibshirani, R. and Buja, A. (1994). Flexible discriminant analysis by optimal scoring. *J. Amer. Stat. Assoc.*, 89:1255-1270.
- [71] Hastie, T., Tibshirani, R. and Friedman, J. (2001) *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, New York.
- [72] Heckerman, D. Geiger D. and Chickering D. (1995) Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*.
- [73] Heckerman, D. (1996) Bayesian networks for knowledge discovery. In *Advances in Knowledge Discovery and Data Mining (Fayyad, Piatetsky-Shapiro, Smyth and Uthurusamy, Eds.)*, 273-205.
- [74] Hinneburg, A. and Keim, D. (1999) Cluster discovery methods for large databases: From the past to the future. *Tutorial Session, In Proc. of ACM SIGMOD International Conference on Management of Data*.
- [75] Hodson, F. R., Sneath, P. H. A. and Doran, J. E. (1966) Some Experiments in the Numerical Analysis of Archeological Data. *Biometrika* 53:311-324.
- [76] Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational properties. *Proc. National Academy of Sciences of the USA*, 79:2554 - 2588.
- [77] Hopfield, J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. National Academy of Sciences of the USA*, 81:3088 - 3092.
- [78] Horton, P. and Nakai, K. (1996) A probabilistic classification system for predicting the cellular localization sites of proteins. *Intelligent Systems in Molecular Biology*, 109-115.
- [79] Huang, Z. (1997) A fast clustering algorithm to cluster very large categorical datasets in data mining. In *Proc. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*.
- [80] Huber, P. J. (1985). Projection pursuit. *Ann. Stat.*, 13:435-475.

- [81] Hyvarinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411-430.
- [82] Kargupta, H. and Park, B. (2001). Mining time-critical data stream using the Fourier spectrum of decision trees. *Proc. IEEE Intl. Conf. on Data Mining*, 281-288, 2001.
- [83] Kaufman, L. and Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York.
- [84] Kohonen, T. (1990). The self-organizing map. *Proc. IEEE*, 78:1464-1479.
- [85] Kohonen, T. (2001). *Self-organizing maps (Third edition)*. Springer, Berlin.
- [86] Landauer, T. K., and Littman, M. L. (1990). Fully automatic cross-language document retrieval using latent semantic indexing. In *Proc. Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research, Waterloo Ontario*.
- [87] Lauritzen, S. (1982) *Lectures on Contingency Tables*. University of Aalborg Press, Aalborg, Denmark.
- [88] Littman, M., Dumais, S. T. and Landauer, T. (1998). Automatic cross-language information retrieval using latent semantic indexing. In *Cross Language Information Retrieval, Grefenstette, G., (Ed.)*. Kluwer.
- [89] Madigan, D. and Raftery, A. (1994) Model selection and accounting for model uncertainty in graphical models using Occam's window. *J. Amer. Stat. Assoc.*, 89:1535-1546.
- [90] Maitra, R. (2001) Clustering massive datasets with applications to software metrics and tomography. *Technometrics*, 43:3:336-46.
- [91] Maitra, R. and O'Sullivan, F. (1998). Variability assessment in positron emission tomography and related generalized deconvolution problems. *J. Amer. Stat. Assoc.* 93:44:1340-55.
- [92] Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979). *Multivariate Analysis*. Acad. Pr. (New York; London)
- [93] Marriott, F. H. (1971). Practical problems in a method of cluster analysis. *Biometrics* 27:501-14.
- [94] Michener, C. D. and Sokal, R. R. (1957) A quantitative approach to a problem in classification. *Evolution* 11:130-162.
- [95] Mirkin, B. G. and Chernyl, L. B. (1970) Measurement of the distance between distinct partitions of a finite set of objects. *Automation and Remote Control* 31:786-92.
- [96] McQuitty, L. L. and Koch, V. L. (1975). A method of hierarchical clustering of a matrix of a thousand by thousand. *Ed. Psych. Meas.* 35:239-54.

- [97] Mojena, R. and Wishart, D. (1980). Stopping Rules for Ward's Clustering Method. In *COMPSTAT 1980, Proc. Comp. Stat.* 454-59. Physica-Verlag (Heidelberg).
- [98] Murtagh, F. (1985). *Multi-dimensional clustering algorithms*. Springer-Verlag (Berlin; New York).
- [99] Myers, G. J. (1978) *Composite Structured Design*. Van Nostrand (New York).
- [100] Nakai, K. and Kanehisa, M. (1991) Expert sytem for predicting protein localization sites in gram-negative bacteria. *PROTEINS: Structure, Function, and Genetics*, 11:95-110.
- [101] Piatetsky-Shapiro G. (1991). Discovery, analysis, and presentation of strong rules. In *Inductive Logic Programming, S. Muggleton (Eds.)*, 281-298. Academic Press, London.
- [102] Platt, J. (1998). How to implement SVMs. In *IEEE Intelligent Systems Magazine, Trends and Controversies, Marti Hearst, (ed.)*, 13(4).
- [103] Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning, B. Schopf, C. Burges, and A. Smola, (Eds.)*, MIT Press.
- [104] Pollard, D. (1981). Strong consistency of  $k$ -means clustering. *Ann. Stat.* 9:135-40.
- [105] Ramey, D. B. (1985). Nonparametric clustering techniques. In *Encycl. Stat. Sci.* 6:318-9. Wiley (New York).
- [106] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *J. Amer. Stat. Assoc.* 66:846-50.
- [107] Render, R. A. and Walker, H. F. (1984) Mixture Densities, Maximum Likelihood and the EM algorithm. *SIAM Rev.* 26:195-239.
- [108] Ripley, B. D. (1991). Classification and clustering in spatial and image data. In *Analysis and Modeling of Data Knowledge* 85-91. Springer-Verlag (Berlin; New York).
- [109] Ripley, B. D. (1996). *Pattern recognition and neural networks*. Cambridge University Press.
- [110] Rissanen, J. (1987) Stochastic complexity (with discussion). *J. Roy. Stat. Soc. Ser. B*, 49:223-265.
- [111] Rocchio, J. J. (1971). Relevance feedback in information retrieval. In *The SMART Retrieval System, G. Salton (Ed.)*. Prentice-Hall, Inc, NJ, 313-323.
- [112] Rotman, S. R., Fisher, A. D. and Staelin, D. H. (1981). Analysis of multiple-angle microwave observations of snow and ice using cluster analysis techniques. *J. Glaciology* 27:89-97.

- [113] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386-408.
- [114] Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6:461-464.
- [115] Scott, A. J. and Symons, M. J. (1971). Clustering methods based on likelihood ratio criteria. *Biometrics* 27:387-97.
- [116] Sheikholeslami, G., Chatterjee, S. and Zhang, A. (1998)- WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *24th International Conference on Very Large Data Bases, August 24-27, New York City*.
- [117] Spiegelhalter, D., Dawid, A., Lauritzen, S. and Cowell, R. (1993) Bayesian analysis in expert systems. *Statistical Science*, 8:219-282.
- [118] Symons, M. J. (1981). Clustering criteria and multivariate normal mixtures. *Biometrics* 37:35-43.
- [119] Tibshirani, R., Walther, G. and Hastie, T. (2001). Estimating the number of clusters in a dataset via the gap statistic. *J. Roy. Stat. Soc. Ser. B*, 63:411-423.
- [120] Van Ryzin, J. (1977). *Classification and clustering*. Acad. Pr. (New York; London).
- [121] Vapnik, V. (1996). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.
- [122] Verma, T. and Pearl, J. (1990) Equivalence and synthesis of causal models. In *Proc. Sixth Conf. on Uncertainty in Artificial Intelligence*, 220-7. San Francisco: Morgan Kaufmann.
- [123] Wahba, G. (1999). Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In *Advances in Kernel Methods – Support Vector Learning (B. Scholkopf, C. J. C. Burges, and A. J. Smola, Eds.)*, 69–88. MIT Press.
- [124] Wolfe, J. H. (1970) Pattern Clustering by multivariate mixture analysis. *Multivariate Behavioral Research* 5:329-350.
- [125] Yourdon, E. (1975). *Techniques of Program Structure and Design*. Prentice-Hall (Englewood Cliffs, NJ).
- [126] Zhang, T., Ramakrishnan, R. and Livny, M. (1996) BIRCH: An efficient data clustering method for very large databases. In *Proc. of ACM SIGMOD International Conference on Management of Data*.
- [127] Zupan, J. (1982). *Clustering of Large Data Sets*. John Wiley & Sons New York; Chichester.