DySect: An Incremental Clustering Algorithm

Peter Andreae, Brian Dawkins and Paul O'Connor [1]

Abstract

Incremental learning techniques have been studied and applied in the Artificial Intelligence community for some time. An example is the Classit algorithm as discussed in Gennari, Langley, and Fisher(1989). This article introduces a modified version of that algorithm called DySect (The name is derived from "dynamic sectioning") which is used as the basis of a technique to obtain clusters from multivariate numerical data. The technique is essentially of order between n and $n\log(n)$ where n is the number of observations and is applied to various real and artificial data sets. It has been run effectively on a data set with roughly a million objects.

It appears that if the technique is applied in conjunction with CART (Breiman, Friedman, Olshen and Stone 1984) useful insight can be gained into the structure of multivariate data sets. The possibility of automatic diagnostic procedures based on the technique is discussed.

[1]Peter Andreae is a Senior Lecturer in the Computer Science Department, Brian Dawkins is a Senior Lecturer in the Institute of Statistics and Operations Research, and Paul O'Connor is a graduate student in the same Institute, at Victoria University of Wellington, Box 600, Wellington, New Zealand. They wish to thank Telecom (NZ) for financial support in the course of the project within which some of the software was developed.

Keywords: CART, concept formation, dynamic sectioning, automatic diagnosis Wisconson Breast Cancer, Cleveland Heart Disease

1 INTRODUCTION

This article has three principal objectives :

- to disseminate to the statistical community a clustering method based on ideas of incremental learning developed in the Artificial Intelligence community.
- to begin development of clustering techniques suitable for very large data sets.
- to investigate an automatic diagnostic procedures apparently suited to many areas of medical research.

We begin with a brief survey of classical clustering, firstly noting with Kaufman and Rousseeuw (1990) that

> ... cluster analysis is mostly used as a descriptive or exploratory tool, in contrast with statistical tests which are carried out for inferential or confirmatory purposes. (p37)

2 CLASSICAL METHODS

Clustering techniques have been part of classical data analysis for many decades, and much effort has been expended in producing algorithms and suitable software to implement them. All major statistical software packages have some form of clustering algorithm built in and there are indeed specific packages whose sole purpose is to expedite generating clusters. (See for example the suite of programs developed by Kaufman and Rousseeuw (1990). Chapter 1 of that work contains a good general discussion of the types of clustering and issues involved.)

SAS (1985) has the CLUSTER procedure which allows eleven different methods of clustering, ranging from average linkage through Ward's. All methods are agglomerative in that initially all points are considered to be distinct, and then clusters are created iteratively by merging extant clusters using the selected criterion. However it is noted (SAS 1985, p. 270) that the computational resources are at least order $n \log n$ where n is the number of observational units, and some methods are even of order n^3. This puts severe limitations on the size of the data sets that can be clustered with these techniques.

Divisive techniques are much less common, and little attention seems to have been paid to them in the statistical community, although in many contexts they would seem to be in some ways more appropriate. See for example Kershaw and Looney (1985). Also the work by Kaufman and Rousseeuw (1990) contains several divisive techniques.

The FASTCLUS technique as implemented in SAS is designed to overcome some of the problems of the classical clustering techniques, producing an order n algorithm. It is recommended for use on data sets with 100 to 100000 observations (SAS 1985, p. 377). However it does not produce a hierarchical structure, which is often itself informative, and the user is required to specify in advance the maximum number of clusters required. An additional tuning parameter relating to distances between seeds can have a marked effect on the results. See also the KMEANS algorithm of Spath (1980). This is closely related to FASTCLUS. Note that the results are dependent upon the initial seed selection and/or order of presentation. (Milligan 1980, Spath 1980).

Additional general references are Everitt (1980), Digby and Kempton (1987) and Gordon (1987). Kaufman and Rousseeuw (1990) also present a technique called CLARA designed to work around the problem of large data sets. It uses ideas similar in spirit to those of FASTCLUS and KMEANS in order to facilitate the clustering but as implemented has a number of limitations, including that of random selection of seeding values.

This paper presents a divisive clustering algorithm called DySect which is intended to compete most directly with FASTCLUS. It is hierarchical in nature in that the output is a multi–way tree, and one of the underlying principles is that of developing an algorithm that will allow "backing–out" of "bad" splits, thus addressing the problem noted by Kaufman and Rousseeuw (1990, p. 44) where they observe

> A heirarchical method suffers from the defect that it can never repair what was done in previous steps. Indeed once an agglomerative algorithm has joined two objects, they cannot be separated any more. Also, whatever a divisive algorithm has split up cannot be reunited.

We will discuss several experiments in which the algorithm was applied to various real and artificial data sets. DySect is based on the COBWEB and CLASSIT algorithms described in Gennari, Langley, and Fisher (1989). The incremental nature of the algorithm suggests that it is of potential use in such areas as medical diagnosis where cases are presented in sequence, and are required to be classified on the basis of the previous instances. We illustrate this with applications to two medical data sets in the public domain, Cleveland Heart Disease data and Wisconson Breast Cancer data. The former comes from Dr. Robert Detrano of the V.A. Medical Center in Long Beach, California. and has been analysed in Detrano et al (1989) as well as Genari et al (1989). The Breast Cancer data is from Dr. William H. Wolberg at the University of Wisconsin Hospitals, Madison and has been discussed in several previous publications, including Mangasarian and Wolberg (1990), Wolberg and Mangasarian (1990), Mangasarian, Setiono and Wolberg (1990) and Bennett and Mangasarian (1992).

3 CLASSIFICATION HIERARCHIES

The goal of clustering is to partition a set of data elements into clusters representing classes of "similar" data elements, where similarity may be defined in many different ways. Some clustering methods generate a single set of disjoint classes (eg FASTCLUS); other methods generate a hierarchy (tree) of classes, as in Figure 1. In such a hierarchy, each node of the tree represents a class of data elements. The topmost node (the "root" of the tree) represents the class containing the entire set of data elements. Nodes at the base of the hierarchy ("leaves" of the tree) represent the smallest classes; in many methods, these are singleton classes corresponding to individual data elements. Every non-leaf node has two or more nodes immediately below it (its "children"). The children of a node represent disjoint subclasses of the class represented by the parent node. For example, nodes B_1, B_2, and B_3 of Figure 1 represent disjoint subclasses of class B. Some methods produce hierarchies in which each non-leaf node has exactly two children (a "binary tree"); other methods produce hierarchies in which non-leaf nodes may have more than two children.

Each level of a classification hierarchy embodies a different clustering of the data elements. For example, classes A, B, and C in Figure 1 represent one clustering of the data, and B_1, B_2, B_3, C_1, and C_2 represent another clustering. (In fact, any cut through the hierarchy represents a clustering, e.g. A, B,C_1 and C_2, would also be a clustering). An important issue in using a class hierarchy is to determine which sets of classes in the hierarchy represent a useful clustering of the data. Methods that generate binary trees (most traditional clustering methods) give very little guidance on which possible clustering in the hierarchy is likely to be meaningful. The DySect algorithm is not restricted to binary class hierarchies, and the display method advocated seems to give reasonable indicatations of meaningful associations.

Most of the clustering methods, including DySect, assume that all the data elements are described by a fixed number of variables. Most methods only allow numerical variables (including the current version of DySect); other methods allow categorical variables, or a combination of categorical and numerical variables.

Classification hierarchies should be contrasted with the "decision trees" gener-

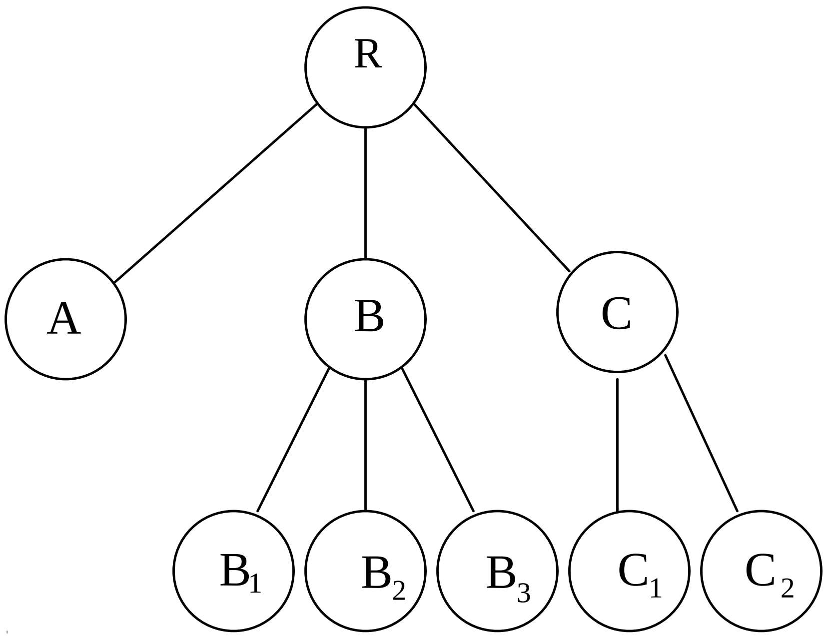

Figure 1: Diagramatic Representation of part of a hierarchical structure.

ated by data analysis techniques such as CART. (Breiman, Friedman, Olshen and Stone 1984) Both CART and the clustering methods generate hierarchical structures describing a set of data elements. However, the data elements must be preclassified for CART, and the decision tree represents a description of the different classes that can be used either to characterize these known classes or to classify a new, unclassified data element. The decision tree nodes represent decision points (based on a binary decision criterion using only one variable) in a process for determining what class a data element should belong to. The clustering methods, on the other hand, assume that the data elements are unclassified, and the class hierarchies that they construct represent possible classifications of the data, where each node represents one class.

There is a natural way in which the CART algorithm can be applied to the classification structure obtained by some hierarchical techniques such as DySect and we exploit this connection later. For example in Figure 1 if we choose a classification based on the three children of the parent node, then the CART algorithm, as implemented for example in S (Chambers and Hastie 1992) could be applied with appropriate classifications, in order to obtain a decision tree suitable for classifying future instances.

4 COMPUTATIONAL COMPLEXITY

An important criterion for evaluating a clustering algorithm is its computational complexity. This is in part determined by how the algorithm processes the data elements. Most of the traditional clustering methods process all the data elements at once, using the distances between all pairs of elements or some surrogate such as dissimilarities to determine the clustering. This leads to $O(n^2)$ complexity in the simple algorithms, and $O(n \log n)$ complexity for the more sophisticated algorithms. The $O(n^2)$ algorithms are not feasible for very large data sets. DySect is an incremental algorithm, processing just one data element at a time. On each data element, it modifies its current version of the hierarchy to accomodate the new element in the best way it can. The current version of DySect takes at most $O(\log n)$ time to modify the hierarchy on each new data element giving DySect an $O(n \log n)$ complexity, allowing it to handle very large data sets. DySect could be modified to reduce this to an $O(n)$ complexity, but we have not yet done so.

A consequence of DySect's incremental nature is that the resulting classification hierarchy depends on the order in which the data elements are presented much as FASTCLUS behaves as noted previously. However, our experience to date has indicated that while there is some degree of non–invariance of the hierarchical structures obtained, the clusterings are nonetheless useful, and the different hierarchies obtained from differing orderings are seldom radically different. Furthermore, there cannot be a unique "correct" clustering for any set of data since there will be many different clusterings that are all justifiable on different grounds and each provides a different way of explaining or elucidating the relationships between the data elements. DySect's ability to generate several different clusterings can actually be turned to an advantage, since it provides the analyst with alternative clusterings from which to select, based on additional knowledge about the domain.

5 THE DySect ALGORITHM

The fundamental idea of the DySect algorithm (based on the COBWEB algorithm, Fisher 1987) is to present a series of data elements to the algorithm, one at a time, incrementally constructing a class hierarchy. The algorithm decides, on the basis of some suitably defined criterion, which classes in the hierarchy the new element should be added to and whether any changes to the current hierarchy are required. The changes it can make include adding a new class to contain the new element, merging two existing classes, and splitting an existing class. The criterion it uses to decide what actions to take involve a local evaluation of the goodness of the partition represented by the set of child nodes of any node it is considering. The criterion is therefore a property of a set of classes, rather than based on distances between elements or classes. The Partition Evaluation Criterion will be discussed in the next section.

Every new element is first added to the root class (since this is the class of all data elements). DySect must then decide how to incorporate the element into the children of that class. A new element is incorporated by adding it to an existing subclass of P, or to a new subclass of its own, or into a new subclass formed out of the existing subclasses. (More explicit details will be presented later.) Whichever class the element is added to, the algorithm will then attempt to incorporate the new element into the children of that class, continuing until it reaches a leaf class or satisfies some other stopping criterion. At present we grow the full hierarchy leading to an order $n \log n$ algorithm, but suitable stopping rules could be devised in order to make the process order n

At each non-leaf node P, DySect chooses whichever of the following actions results in the set of subclasses of P with the best value of the "Partition Evaluation Criterion". It then performs the action and repeats the process until the appropriate stopping criterion is met. The four actions are:

1. Add the element into an existing subclass of P

2. Create a new subclass of P and add the element into the subclass as its sole member.

3. Merge two of the subclasses of P into a new subclass (with the two subclasses as its children). This reduces the number of children of P, but adds a new level to the hierarchy.

4. Split a subclass of P into several smaller classes (by replacing the class with its children). This increases the number of children of P, but removes a level from the hierarchy below the subclass chosen.

To evaluate the choices, it does not have to actually modify the hierarchy — it merely computes what the Partition Evaluation Criterion would be if the action were performed. If P has m children, then there are $(m+1)^2$ options to evaluate. Given that the number of children of a node is almost guaranteed to be less than 10, and usually between 2 and 5, this is not expensive to compute.

When DySect has chosen a subclass (action 1), it then recursively incorporates the element into that subclass until it reaches a leaf of the hierarchy. It then replaces the leaf class by a new class whose children are the old leaf class and a new leaf class containing the new element (action 2). The third and fourth actions allow a form of recovery from a bad choice earlier in the processing, and appear to be one of the reasons why useful results can be obtained, despite the order-dependent nature of the structures built by the algorithm.

The final hierarchy constructed by this algorithm is large — it contains one leaf for every element. In general, the lower levels of the tree tend not to contain useful information, and it is only classes in the top several levels that represent useful clusters.

DySect maintains a descriptions of each class in the hierarchy in terms of simple statistics on the elements in the class and as a list of all the data elements in the class. However, these descriptions are not very informative or useful to a data analyst. One way of obtaining more useful descriptions is to assign suitable classification values to the elements in each cluster and then to apply CART software as noted previously, to obtain useful descriptions of the classes in terms of the variables.

Simplified IF-THEN rules for classifying data elements into these classes can then be obtained straightforwardly from the CART trees. A second way of obtaining useful descriptions is to construct box plots of all the variables for the elements in each class. The experimental results in section 8 are presented using this latter method, though we have also generated the alternative descriptions using CART.

6 THE PARTITION EVALUATION CRITERION

The Partition Evaluation Criterion (PEC) is defined on a partition of a class — a set of disjoint subclasses of the class whose union is the class. The children of any class in the classification tree constitute a partition of that class. The PEC that DySect uses is:

$$\mathrm{PEC} = \frac{\sum_{k=1}^{K} P(C_k) \sum_{i=1}^{I} (s_{ik}^2 - s_{ip}^2)}{K}$$

where K = number of subclasses in the partition, I = number of variables, s_{ik}^2 = the variance of variable i in the kth subclass, s_{ip}^2 = the variance of variable i in the parent class, and $P(C_k)$ is the proportion of the data elements in the kth subclass. DySect attempts to maximise the value of the PEC.

This is not the only possible PEC that could be used, and we have experimented with other PEC's. One way of viewing the current PEC is a data–based estimate of the average expected decrease, per subclass, of the variances summed over all the variables. It can also be viewed as a trade-off between the number of subclasses in the partition and the homogeneity of the subclasses, where the homogeneity is measured by the sum over the variables of the variances in each class. The trade-off is required because allowing more classes in the partition always allows the classes to be more homogeneous, but partitions with a very large number of very specific classes are generally not useful for clustering purposes.

The PEC is only a local evaluation criterion in that it is applied independently to the children of each node in the classification tree; it is not an evaluation criterion on the tree as a whole, and DySect does not guarantee to produce a global maximum of a sum of all the PEC's at each node.

The form of the PEC that DySect uses allows very efficient computation of the PEC since the variances can be easily computed incrementally as new data items are added, and the contributions of each subclass in a partition can be computed independently, and do not need to be recalculated for each option.

7 PSUEDO–CODE REPRESENTATION OF THE ALGORITHM

Figure 2 gives a block diagram structure for the program which is given in psuedo–code representation as follows.

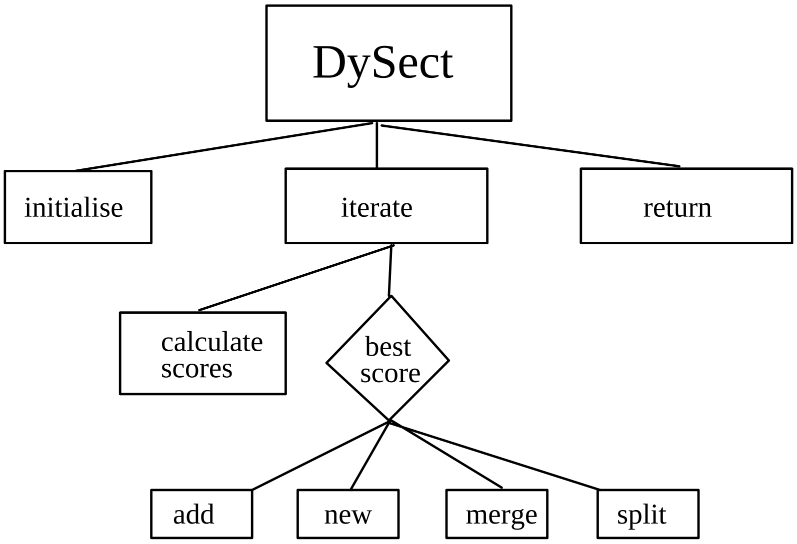

Figure 2: A Block diagram structure of the DySect algorithm.

Function DySect (*instance*, *node*,)

If *node* is a leaf

Then Make a *new node* and replace *node* with *new node*

Attach *node* as a child of the *new node*

Construct a new leaf node containing *instance* and

attach as a child of the *new node*

Else

Repeat

Calculate PEC *new* when a new child with sole member

instance is created

For each *child* of *node*

Calculate PEC *childscore* if *instance* is added

to the child

Find the minimum *current* of the *childscore* values

For all pairs of children of *node*

Compute PEC *pairscore* when the pair are merged

into a single class

Find the minimum *merge* of the *pairscore* values

For each *child* of *node*

Compute PEC *replace* if *child* is replaced by its children

Find the minimum *split* of the *replace* scores.

Choose the least of *new, current, merge, split*

new: Create a new child of *node* containing *instance*;

current: Add *instance* in appropriate child

merge: Remove the nodes being merged, create a new child of *node* with the merged nodes as its children

split: Replace the appropriate node by its children

Until instance has been added to a child of *node*.

Call DySect on the *instance* and the child it was added to.

Fi

8 THE EXAMPLES

We have applied DySect to a variety of data sets. The three examples we discuss here were chosen to be an informative mix of classical and new data, both real and artificial, with the intention of demonstrating the general nature and utility of the procedure.

8.1 THE IRIS DATA

The first example is the time–honoured Anderson/Fisher iris data. A typical tree grown by the DySect algorithm is presented in Figure 3. Each of the enclosing rectangles represents a node; there are thus eight nodes in the bottom row. The children of a given node are plotted beneath it. Thus the left hand node in the second row has three children, as does the root node, represented by the top rectangle. The boxplots within a given enclosing rectangle represent each of the variables restricted to members of the node. The central "layer" of the figure thus presents boxplots of the data elements in each of the three subclasses that DySect constructed at this level of the hierarchy. The extreme left hand panel labelled Class 2 depicts the boxplot representations of the values of the four variables associated with the 49 elements of the group. The three panels exhibit one of the more intriguing aspects of the DySect approach to clustering in that it is evident from the comparison that there are different "profiles" associated with the groups. By "profile" we refer to the relative positions of the boxplots for the different variables within a given node. Thus it is quite evident that Class 3 and Class 4 have similar profiles and Class 2 is quite different.

The bottom set of panels illustrates the distributions of the subgroups obtained by decomposing the middle level groups. Thus the three panels under Class 2, labelled *c21*, *c24* and *c25* respectively represent the three groups determined from the 49 elements of class 2.

The software allows the user to identify the data elements of any class in the hierarchy, and consequently further analyses can be based on such information. Although it is not evident from Figure 3, it can be determined that the first level split into three categories parallels the three species known to be present in the data set. Indeed Class 2 contains 49 members of the first species, and class 3 contains 48 elements of the third species. That in some orders of presentation DySect obtains

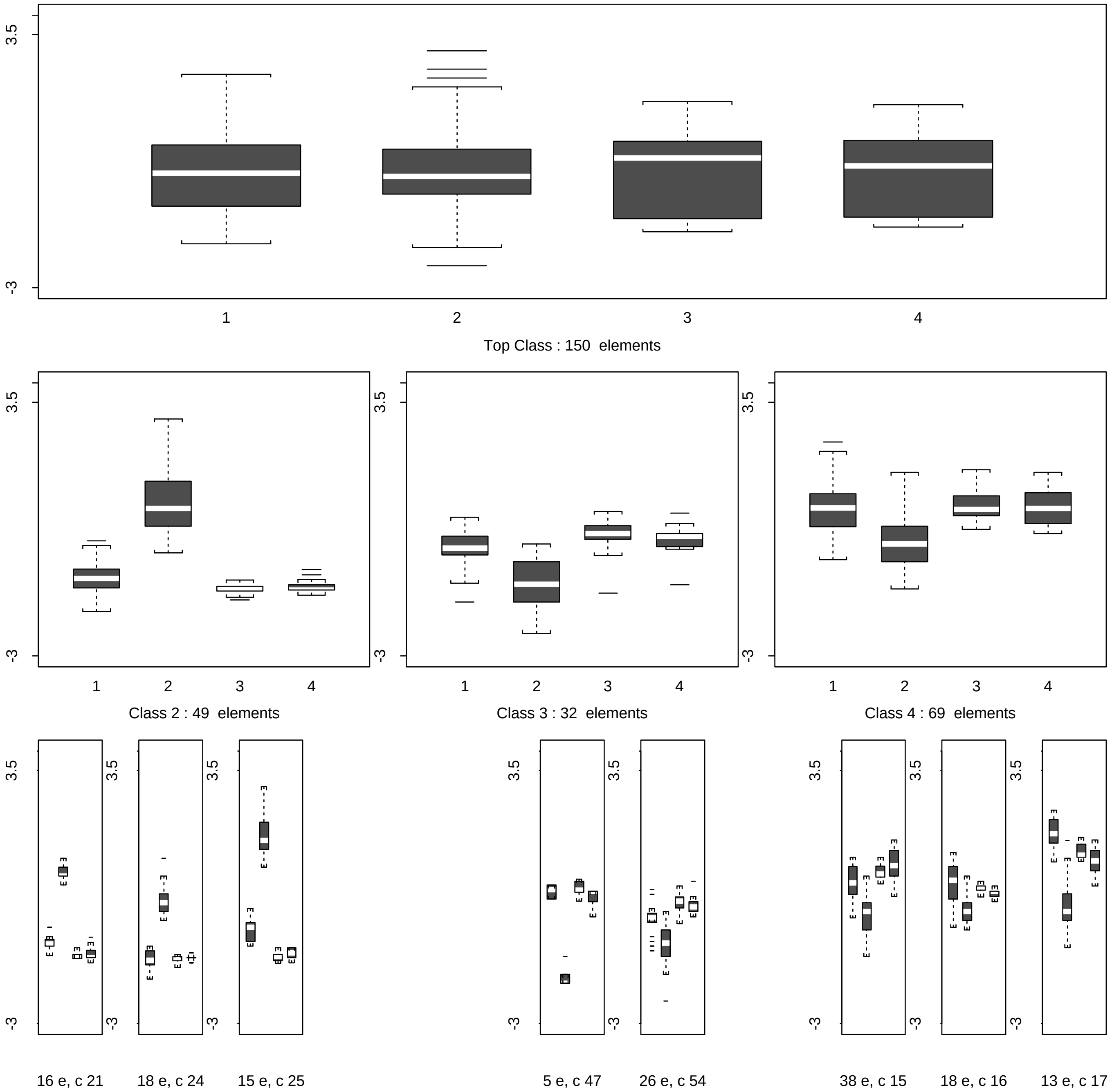

Figure 3: The standard boxplot display of an application of DYSECT to the iris data. The boxplot for each of the variables based on the membership at a node is display for a given node. In the bottom set of panels, e denotes elements and c the class number. Thus the extreme left hand set of boxplots represents class 21 with 16 elements. See text for further discussion of the display.

the correct classification at the first level.

8.2 CANCER DATA

We will now consider the breast cancer database as detailed above. As of 15th July 1992 it consisted of 699 data elements on which nine numerical "diagnostic" variables are present together with a 10th binary variable denoting presence or absence of disease. The diagnostic variables are essentially ordered categorical ones and are on a scale of 1 to 10 and measure such things as uniformity of cell size and shape, marginal adhesion etc. The data have been accumulated over several years and are presumably still being added to. Figure 4 is the result of one run of DySect on the data (excluding the classification variable).

DySect has made a clear separation at the first level into benign and malignant growths, Class 7 with 239 elements has 14 diagnosed as free of disease whereas Class 9 with 460 elements has only 16 classified as diseased for an overall misclassification rate of about 4%. The next level down is more interesting, at least to the naive user: a point which seems of some potential interest is that the node Class 17 is clearly the repository of the cases least easily distinguishable by an automatic selection procedure. Presumably a closer study of these cases is in order.

In any case it would seem that the hierarchy as presented can form the basis of an automatic diagnostic screening procedure which could be used to classify further instances. On the basis of the information available, various nodes are selected as a partition of the extant cases, and these are classified as favourable or unfavourable.

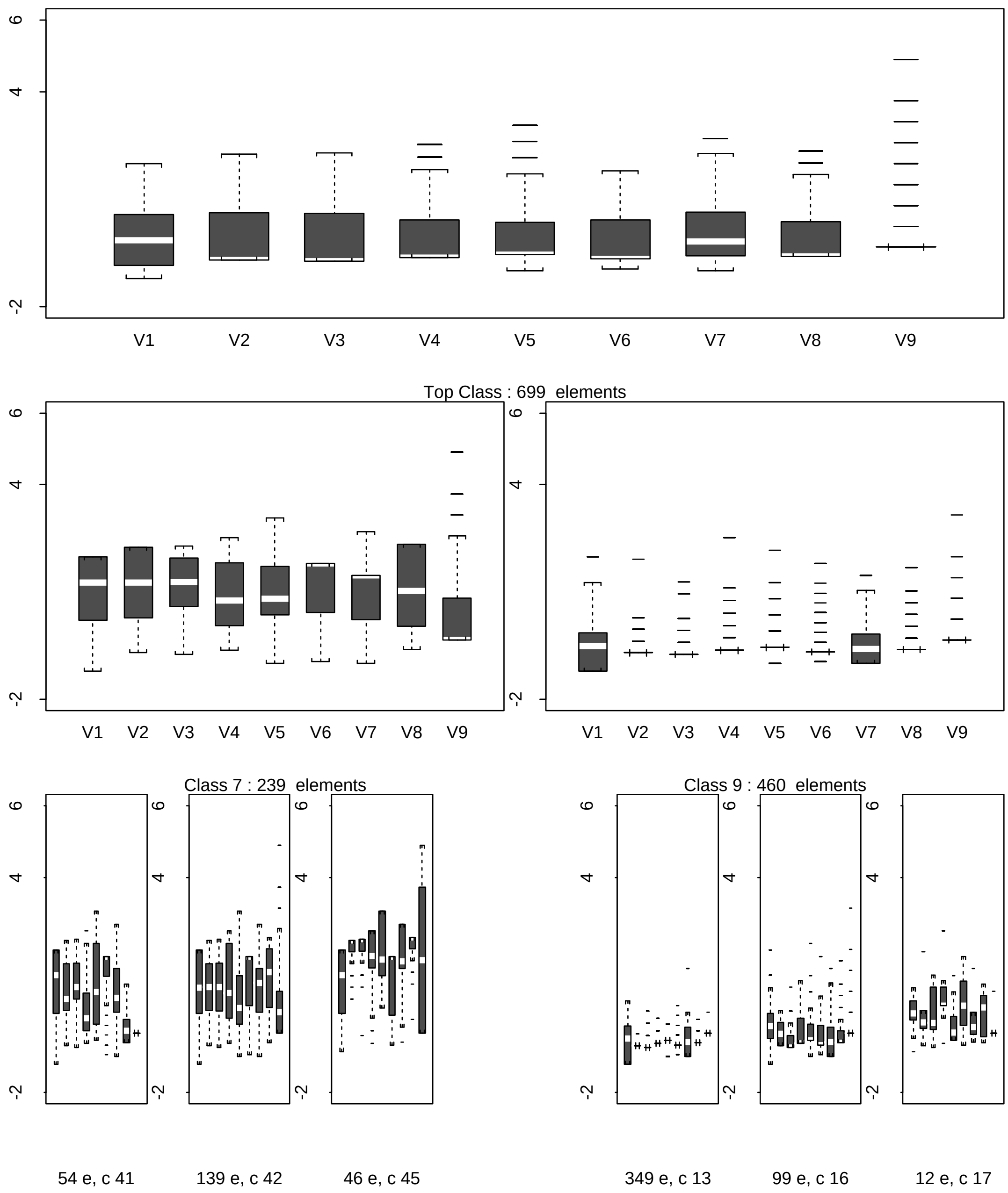

Figure 4: DySect structure obtained from the Wisconsin Breast Cancer Data, using the nine diagnostic variables.

Then any further instance could be presented to the algorithm and incorporated into the tree. Depending on what node of the partition it was incorporated in, the instance could then be classified. For example using a decision tree based on Classes 7 and 9, it would be classified as diseased or not depending on which class it was incorporated into. Of course a new instance **could** in principle lead to a radical rearrangement of these top level children of the parent node, but it is extremely unlikely, and would generally be an indication of something very unusual about the instance. This aspect of DySect has not been explored in any detail.

8.3 HEART DISEASE DATA

As a further instructive example we will use the Heart Disease Database obtained from Dr Robert Detrano of the V.A. Medical Center in Long Beach, California. This has been analysed in several publications, notably Detrano et al (1989) and Gennari et al (1989). In the original data there were 303 instances with observations on a large number of variables, but in the publicly available data only 14 are used. The variables are a mix of categorical and numerical data and include a severity variable which is the dependent variable of interest. Although this severity variable is an ordered categorical variable from 0 to 4, where 0 denotes absence of disease, the principal research interest has been in the simple presence/absence of disease.

We shall restrict attention to a subset of six of the variables including a mix of numerical and ordered categorical. The interest in this example lies in the application of the DySect algorithm to the restricted data with a view to determining how homogeneous the classes are, particularly at the first "child" level. The more homogeneous the classes, presumably the better the diagnostic tool that can be obtained from the hierarchy.

Figure 5 shows the results of one run of DySect on the restricted data. The three children of the root node have 79 "misclassified" instances: the classes have 87, 24 and 53 free of disease respectively, starting with Class 2, and on the basis that the predominant type determines the nominal classification into diseased and undiseased, there are thus 79 misclassified, with roughly 74% correctly assigned. This compares very favourably with the analyses of the full set referenced in the literature.

8.4 DATA WITH NOISE VARIABLES

It is interesting to note how robust the procedure is to contamination by noise variables. We added four pure noise variables to the iris data and ran the algorithm on the augmented data. As noted in Fowlkes, Gnanadesikan and Kettenring (1988) such contamination can lead to considerable confusion in classical methods. A typical run of a DySect analysis is shown in Figure 6. The split between the first and the other two varieties has been clearly identified.

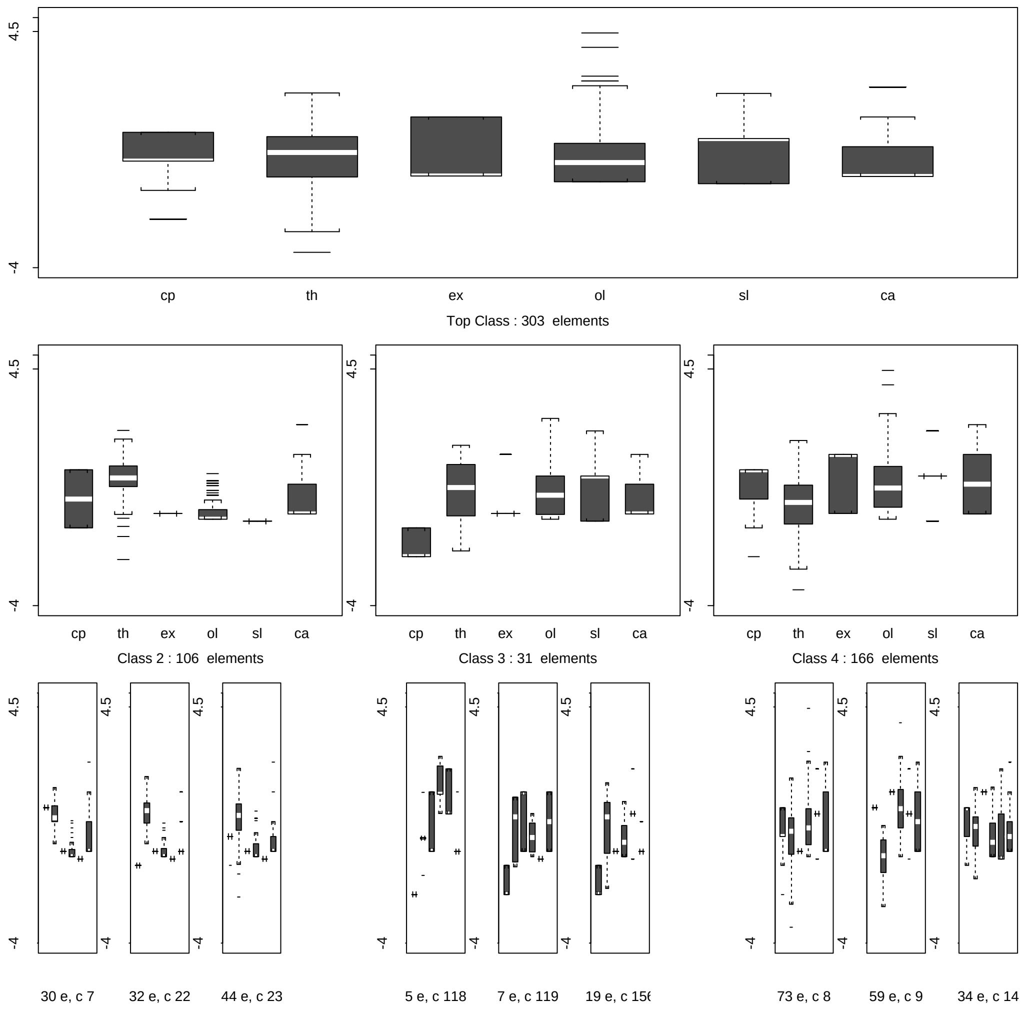

Figure 5: Top–level display of a DySect analysis of the Heart Disease Data. See text for details.

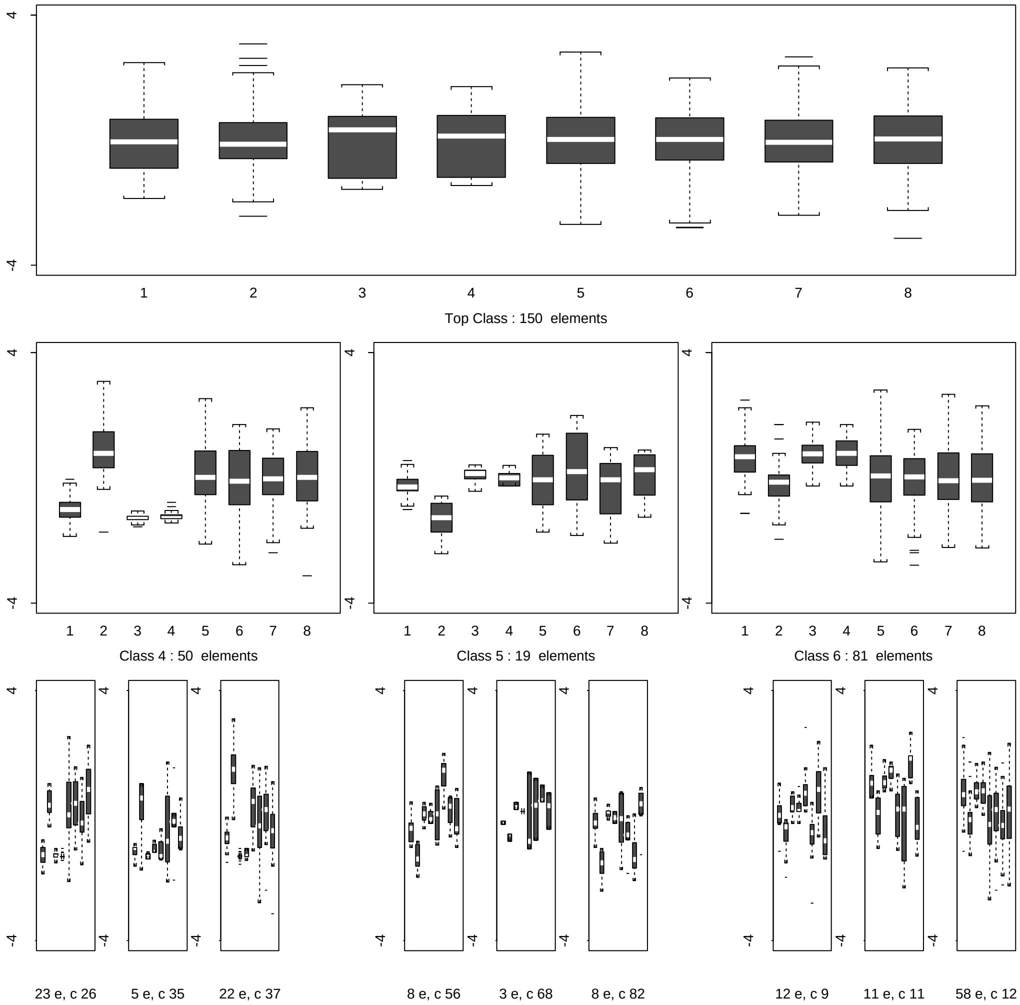

Figure 6: An example of the ability of DySect to filter out noise variables. See text for further details.

9 DISCUSSION

We have intended that the evidence produced in the previous sections should have gone some distance towards meeting the objectives detailed in the INTRODUCTION. It appears to us that incremental techniques, and indeed machine learning ideas in general, are of sufficient interest to provoke much associated research on a purely applied statistical level. On a distinct conceptual level is the burgeoning problem of handling huge amounts of data, and it is imperative that methods are developed that are at most order n, the number of instances of observations. One of the major virtues of the approach would seem to be that it offers the prospect of coping with large data sets, in that it seems clear that there are modifications of the algorithm that should lead to order n analysis.

Finally it appears that some form of diagnostic tool ought to be able to be developed using a conjunction of hierarchical clustering tools and CART type methods to produce decision trees of the sort required in such diagnosis. The examples above are intended only as primitive steps in that general direction.

9.1 LIMITATIONS

A limitation at present would appear to be the non–robustness to order of presentation of the cases. While that is undeniably the case, it should be observed that much the same problem can affect FASTCLUS, the "nearest neighbour" to DySect amongst the clustering algorithms in common use by the statistical community, or indeed any technique which makes essential use of a random set of seeding values. It is also the case that no clustering technique should ever be used "in isolation", and that two or three alternatives should be tried. Taking a fairly pragmatic/empirical approach it appears that useful groupings can result from the application of DySect in practically any case and it is of course open to the analyst to build further investigations on the results. We agree with Spath (1982) who asserts

> Primarily, what makes an application of cluster analysis successful is the significant practical interpretation of the clusters it produces. For this reason it frequently makes sense to apply various methods, one after another and independently. Nevertheless one is sometimes happy enough to a obtain reasonable subdivisions of the objects. (p12)

There are several possible approaches that could be employed here. These include such things as selection of a subsample analogous to the seeding sample of FASTCLUS and growing the tree with this sample as initial elements. Indeed a suitable research topic would be that of investigating various strategies aimed at limiting the variability of the output: it is interesting to note that in some orders of presentation, DySect has obtained the exact division of the iris data into its three species.

We like to think of this order–dependence as reflecting the actual process involved in periodic reviews of biological taxa as new evidence comes to hand or the gradual accumulation of medical cases leads to a deeper understanding of the epidemiology of a particular disease.

One practical limitation to the use of the algorithm is the non–availability of suitable code. The programming problems inherent in an implementation are non–trivial and run–time efficiency requires the application to be built in a relatively low–level language such as C. At the time of writing we are planning on being able to submit to *statlib* suitable code. This will be in the form of C source code for the main algorithm, and a set of S(plus) functions suitable for calling the compiled code and massaging the output with CART etc.

9.2 FURTHER WORK

In addition to the order dependency noted above there are other topics of interest regarding the algorithm as implemented. Some of the more obvious include the following:

- The current implementation of the search algorithm has $O(n \log n)$ time complexity and $O(n \log n)$ space requirements. This is primarily due to keeping all the lower levels of the class hierarchy. If these levels, which are not useful to the interpretation process, were not created, then the complexity could be reduced to $O(n)$.

- What kind of clusters does the method favour? Could different types of clusters be determined by changing the PEC?

- What alternative search strategies could be implemented without unduly increasing the resources necessary?

- What guidelines could be developed in order to "automate" suitable selection of suitable clusters?

- Missing values are not handled; these could be handled in a number of more or less obvious way.

- More seriously, categorical variables have not been incorporated into the procedure in any way. Cobweb handled categorical variables, but it is not obvious how its PEC can be combined with DySect's PEC for numerical variables in a straightforward way.

- The PEC deals with each variable independently, and therefore cannot find classes "defined" by relationships between the variables.

Current research we are undertaking is aimed at resolving some of the issues involved.

10 SUMMARY AND CONCLUSIONS

We have developed an algorithm, DySect, from an incremental learning algorithm published in the Artificial Intelligence literature, and have applied it to numerous data sets, many more than those discussed above. It seems evident on empirical grounds that useful groupings of experimental units can be made on the basis of the tree–ordered structures grown.

It appears that DySect, as an order $O(n \log n)$ procedure where n is the sample size, can produce some informative and interesting groupings of the data. Its incremental nature certainly allows for some variation in the hierarchical structures produced, but experience to date indicates that this is not a serious problem, and is more of academic rather than applied interest, and is in the nature of a trade–off with the order of the process, a problem shared with approaches such as FASTCLUS and CLARA. Kaufman and Rousseeuw (op. cit. p41) attempt to circumvent the problem by repeating the analysis several times with randomly generated seeds, choosing the "best" solution from amongst those generated.

The ideas certainly seem to warrant more research effort, since the potential gains seem to be considerable, particularly as relates to the sizes of the datasets that can be managed.

11 COMPUTING

All computing was carried out on the Sun SPARCstation network of the Institute of Statistics and Operations Research of Victoria University. The algorithm was implemented by Paul O'Connor in C, who also devised the boxplot representation of the tree structures obtained from DySectt. All graphics and CART trees were obtained using Splus.

12 REFERENCES

Andrews, D. F., and Herzberg, A. M. (1985), Data, New York:Springer–Verlag.

Bennett, K. P., and Mangasarian, O. L. (1992), "Robust linear programming discrimination of two linearly inseparable sets", Optimization Methods and Software1, 1992, 23–34.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984) Classification and Regression Trees, Belmont, California:Wadsworth.

Chambers, J. M., and Hastie, T. J. (1992) Statistical Models in S, Pacific Grove, California:Wadsworth and Brooks.

Digby, P. G. N., and Kempton, R. A. (1987), Multivariate Analysis of Ecological Communities, London:Chapman and Hall.

Everitt, B. S. (1980), Cluster Analysis, 2nd. Ed., London:Heinemann Education Books.

Fisher, D. (1987), "A hierarchical conceptual clustering algorithm," Machine Learning, 2, 139–172.

Fowlkes, E. B., Gnanadesikan, R. and Kettenring, J. R. (1988), "Variable Selection in Clustering," Journal of Classification, 5, 205–228.

Gennari, J. H., Langley, P. and Fisher, D. (1989), "Models of Incremental Concept Information," Artifical Intelligence, 40, 11–61.

Gordon, A. D. (1987), " A review of hierarchical classification," Journal of the Royal Statistical Society, Ser. A, 150, 119–37.

Kaufman, L. and Rousseeuw, P. J. (1990), Finding Groups in Data, New York:Wiley.

Kershaw, K. A., and Looney, J. H. H. (1985), Quantitative and Dynamic Plant Ecology, London:Edward Arnold.

Mangasarian, O. L., Setiono, R., and Wolberg W. H. (1990), "Pattern recognition via linear programming: Theory and application to medical diagnosis", in: Large-scale numerical optimization, eds. T. T. Coleman and Yuying Li, Philidelphia:SIAM Publications pp. 22–30.

Mangasarian, O. L., and Wolberg, W. H. (1990), "Cancer diagnosis via linear programming", SIAM News, 23, 5, 1–18.

Milligan, G. W. (1980), "An Examination of the Effect of Six types of Error Perturbation on Fifteen Clustering algorithms," Psychometrica, 45, 325–342.

SAS Institute (1985), SAS User's Guide: Statistics, Cary,NC: SAS Institute Inc.

Spath, H. (192) Cluster Analysis Algorithms, Chichester:Ellis Horwood.

Wolberg W. H. and Mangasarian O. L. (1990) "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", Proceedings of the National Academy of Sciences, 87, 9193–9196.