How does something arise from nothing?

– John Archibald Wheeler, July 9, 1911 – April 13, 2008.

University of Alberta

High-Dimensional Data Mining: Subspace Clustering, Outlier Detection and Applications to Classification

by

Andrew Philip Ogilvie Foss

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

Examining Committee

Osmar Zaïane, Computing Science

Raymond Ng, Computer Science, University of British Columbia

Dale Schuurmans, Computing Science

Jörg Sander, Computing Science

Mauricio Sacchi, Physics

To my mother, Margaret Scott Foss and father, Gr. Cpt. Patrick Shaw Foss

Abstract

Data mining in high dimensionality almost inevitably faces the consequences of increasing sparsity and declining differentiation between points. This is problematic because we usually exploit these differences for approaches such as clustering and outlier detection. In addition, the exponentially increasing sparsity tends to increase false negatives when clustering.

In this thesis, we address the problem of solving high-dimensional problems using low-dimensional solutions. In clustering, we provide a new framework MAXCLUS for finding candidate subspaces and the clusters within them using only two-dimensional clustering. We demonstrate this through an implementation GCLUS that outperforms many state-of-the-art clustering algorithms and is particularly robust with respect to noise. It also handles overlapping clusters and provides either 'hard' or 'fuzzy' clustering results as desired. In order to handle extremely high dimensional problems, such as genome microarrays, given some sample-level diagnostic labels, we provide a simple but effective classifier GSEP which weights the features so that the most important can be fed to GCLUS. We show that this leads to small numbers of features (e.g. genes) that can distinguish the diagnostic classes and thus are candidates for research for developing therapeutic applications.

In the field of outlier detection, several novel algorithms suited to high-dimensional data are presented (T*ENT, T*ROF, FASTOUT). It is shown that these algorithms outperform the state-of-the-art outlier detection algorithms in ranking outlierness for many datasets regardless of whether they contain rare classes or not. Our research into high-dimensional outlier detection has even shown that our approach can be a powerful means of classification for heavily overlapping classes given sufficiently high dimensionality and that this phenomenon occurs solely due to the differences in variance among the classes. On some difficult datasets, this unsupervised approach yielded better separation than the very best supervised classifiers and on other data, the results are competitive with state-of-the-art supervised approaches.The elucidation of this novel approach to classification opens a new field in data mining, classification through differences in variance rather than spatial location.

As an appendix, we provide an algorithm for estimating false negative and positive rates so these can be compensated for.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Osmar R. Zaïane, for constant support, encouragement and insight.

I also much appreciate the time and efforts of my examining committee in reading and considering my candidacy proposal and dissertation and attending the candidacy and defence examinations.

I would also like to thank Dr Jörg Sander, Dr Robert Holte and Dr Sandra Zilles for reading some of my work and helpful comments and suggestions. The authors who provided us with the implementation of their algorithms that were utilised in this work are greatly appreciated.

Finally, this work has been made possible in part by the generous financial support of the Natural Sciences and Engineering Council of Canada (NSERC), the iCORE Circle of Research Excellence, AICML and the University of Alberta. I thank them.

Table of Contents

List of Tables

List of Figures

List of Publications

Related Refereed and Invited Publications

Andrew Foss, Sandra Zilles and Osmar R. Zaïane. Unsupervised Class Separation of Multivariate Data through Cumulative Variance-based Ranking. In *Proceedings of IEEE International Conference on Data Mining ICDM'09*, pp. 139-148, 2009.

Andrew Foss and Osmar R. Zaïane. The Estimation of True and False Positive Rates in Higher Dimensional Problems and its Data Mining Applications. In *Proc. of Foundations of Data Mining Workshop, in conjunction with IEEE International Conference on Data Mining*. pp. 673-681, 2008

Hongqin Fan, Osmar R. Zaïane, Andrew Foss and Junfeng Wu, Resolution-Based Outlier Factor: Detecting the Top-n Most Outlying Data Points in Engineering Data. *Knowledge and Information Systems, An International Journal*, pp. 31-51, 2009.

Hongqin Fan, Osmar R. Zaïane, Andrew Foss, and Junfeng Wu. A Nonparametric Outlier Detection for Effectively Discovering Top-N outliers from Engineering Data. In *Proc. of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'06)*, 2006.

Andrew Foss and Osmar R. Zaïane. Effective Subspace Clustering with Dimension Pairing. Technical Report TR06-23, University of Alberta, Oct 2006.

List of Symbols

CLIQUE	‘Projected’ clustering algorithm [11]
FASTOUT	Ensemble based outlier detection algorithm (binary measure)
FASTOUT-R	Ensemble based outlier detection algorithm (real-valued measure)
GCLUS	Implementation of the MAXCLUS framework
GSEP	Algorithm for fast classification of very high dimensional binary labelled data
LOF	Local Outlier Factor [39]
MAXCLUS	A framework for efficient and effective subspace clustering
ORCA	High efficiency, full-dimensional outlier detection algorithm
RMD	Robust Mahalanobis Distance based outlier algorithm [153]
ROF	Resolution based Outlier Factor scheme [63] based on TURN*
SERA	Statistical Error Rate estimation Algorithm
SOE1	Single dimension ensemble outlier algorithm [85]
T*	Outlier detection algorithmic framework
T*ENT	Implementation of T* using an entropy measure
T*ROF	Implementation of T* using ROF
TURN*	2D clustering algorithm [66]

Chapter 1

Introduction

High dimensional databases are becoming the norm as data collection becomes increasingly automated. Knowledge discovery from these datasets presents several important challenges. As will be discussed more extensively later, increasing sparsity typically makes full-dimensional manipulation of the data fruitless. This motivates analysis of subspaces. However, the number of subspaces increases exponentially with dimensionality. On the other hand, since there are effective tools for data mining in low dimensionality, it is of interest to see if applying these tools in low dimensional spaces can yield useful information for a high dimensional dataset.

It has been observed [31] that the relative differences between data points declines for most types of data with increasing dimensionality. This is a consequence of the phenomenon of what is termed concentration of measure in the mathematical community [55] , which is a generalisation of the law of large numbers. This is problematic for many techniques such as tree indices as well as any algorithm exploiting distance or similarity. On the other hand, concentration of measure is described as a blessing by the mathematical community as it gives strong bounds on the deviation of a broad class of functions from their mean. This is widely exploited in machine learning (e.g. [91]) and we show how this phenomenon can be exploited for using outlier detection for classification.

Three important areas of data mining are classification, clustering and outlier detection. Clustering is an unsupervised approach to classification while outlier detection seeks to find the most unusual data points. To date, there has been a tendency to see these three areas as separate with their own approaches and algorithms. As it will become obvious in this thesis, outlier detection in high dimensional datasets can not only use clustering techniques internally but can produce output that amounts to unsupervised classification (clustering) or semi-supervised classification through the use of some labelled data. That is, we can leverage varying outlier tendencies to separate and yield information about underlying classes.

Another area, much neglected by data miners, is the proper estimation and accommodation of underlying error rates. Data has inherent error and this 'noise' amplifies strongly with increasing dimensionality. Without taking this into account, our results may be meaningless. On the other hand, proper estimation of true and false positive rates can allow us to recover false negatives, minimise false positives and even discover important knowledge. Later in the thesis, an example will be given of how the likely set of relevant genes can be extracted from labelled genetic data. Previous data mining techniques could rank genes or build models but, typically, could not estimate the correct size of the model. The estimation of true and false positive rates is presented before the other work as it has applications in the later chapters.

Thus this thesis has two main contribution areas. Clustering (MAXCLUS) and outlier detection (T*, FASTOUT). Aditionally a simple but novel classifier is provided (GSEP) and an algorithm for error rate estimation (SERA) introduced in Appendix A.

1.1 Background

In our previous work [65], a new parameter-free 2D clustering algorithm was introduced (TURN*). This was particularly effective in collecting clusters and separating out noise. This ability to distinguish noise from clusters suggested an application for outlier detection. While efficient in 2D, the method used for finding nearest neighbours scaled exponentially with dimensionality. There was therefore an interest in how to extend this work to arbitrarily high dimensionality. While pursuing this, a framework (MAXCLUS) was developed which required only 2D clustering but discovers candidate subspaces of any size in which clustering can be done to confirm the existence of significant clusters. This framework can be implemented using TURN* or almost any other clustering algorithm. A more efficient projected clustering method was also developed.

In the work on MAXCLUS, it was observed that the clusters could be expanded beyond the discovered borders to improve collection – reduce false negative rates. It was also noted that noise is suppressed by increasing dimensionality as sparsity increases. Investigating how false positive and negative rates can be estimated led to the SERA algorithm [67] and its application to discovering a bounded small set of predictor features from datasets with binary labelling, e.g. for some disease condition, something that has been very difficult to do previously. This is presented in Appendix A.

Applying MAXCLUS to genome data is impractical due to the extremely large dimensionality. However, given labelled samples, the simple algorithm GSEP was developed that generated a weighted list of prognosticator genes/features, the top n of which can be fed to MAXCLUS yielding candidate subsets of genes/features and even clusters within these subsets based on, for example, up and down regulation. The size of n can be readily chosen based on the weights which are normalised so as to be comparable across datasets. This is a generic approach that can be applied to other types of data with very high dimensionality. This is complimentary to the purely statistical approach of SERA mentioned above (See Appendix A). Alternatively, SERA can be used to compute n.

Turning to the related problem of outlier detection, the TURN* algorithm generates various statistics while scanning across resolutions. A resolution is like a grid imposed on the data space which can be coarse or fine. This scanning is key to its automatic setting of its own internal parameters. The availability of statistics suggested various heuristics for outlier detection. Various schemes were tested and one (ROF) was found, which worked excellently on a 3D engineering database [63]. However, the scaling problem remained. Subsequently, this and another, based on entropy, were developed as T*ROF and T*ENT for high-dimensional outlier detection. The automated nature of TURN* has, inevitably, some increased cost. Further research on a very simple but novel subspace nearest neighbour approach led to the highly efficient and effective FASTOUT algorithm.

The research on outlier detection to date has focussed on the low dimensional concept of finding individual 'unusual' data. Because of this, researchers would modify labelled data with balanced classes to create a main and a rare class(es) in order to test their outlier algorithms. This was never properly motivated and is, in fact, peculiar. Smallness of the class is not, in itself, normally a distinguishing feature. Indeed, outlier algorithms can be used on labelled data without creating rare classes and, further, no researcher has shown particular success in separating very small classes from large ones in high dimensional data. In this thesis, we motivate theoretically and demonstrate empirically how outlier detection can separate classes of any reasonable size by exploiting their difference in variance rather than spatial location. This is effective even for concentric classes if there are sufficient dimensions.

The sequence of this thesis largely follows this chronology except that the issue of measurement error is addressed first.

1.2 Thesis Statement

It has been established (e.g. [31]) that the relative differences between points diminishes with increasing dimensionality in most cases. At the same time, sparsity typically increases exponentially with dimensionality. This presents severe challenges for clustering and outlier detection. The next two hypotheses address this problem:

Hypothesis 1. Clustering*: High dimensional datasets can be clustered using low-dimensionality techniques, specifically, through the discovery of relevant subspaces in which meaningful clusters likely exist. Given such candidate subspaces, clusters can be extracted.*

Hypothesis 2. Outlier Detection*: A meaningful measure of outlierness can be assessed in a high-dimensional space by computing an outlier score in many low-dimensional subspaces and taking the sum of such scores.*

If an outlier ranking can be computed, does this really tell us anything about the individual points if they are produced by some source with a certain variance? Since, for tailed distributions, such a point can attain any rank with some statistical probability, what valid information can be extracted from this ranking? The next hypothesis addresses these questions.

Hypothesis 3. Classification through variance*: If two or more classes with different variances are present in a dataset, the members of these classes will tend to separate in the outlier ranking, permitting an approach to classification.*

While multiple contributions and results are presented herein the principal thesis behind this work is that, while low dimensional data mining can often be used to yield information about individual data points and classes, high dimensional analysis yields more acute information about certain properties of inherent classes within the data, even though information about individual points is depreciated.

This fundamental notion is based on theoretical results that are not new but whose implications have not so far been formally noted and thus not explicitly exploited in the research areas relevant to this thesis, specifically outlier detection. The laws of nature are always there but science gradually documents them and, in many cases, application follows after theoretical understanding.

1.3 Contributions

The principal algorithmic contributions are as follows:

- A framework (MAXCLUS) for discovering subspaces likely to contain significant clustering and for extracting the clusters.
- An efficient and effective implementation of MAXCLUS, GCLUS.
- An efficient and effective classifier for binary high-dimensional problems, GSEP.
- An outlier detection framework (T*) for high-dimensional problems and two efficient and effective implementations, T*ENT and T*ROF. T*ENT and T*ROF are shown as effective classifiers for binary-class problems.
- A highly efficient and effective outlier detection algorithm (FASTOUT) for high-dimensional problems based on a novel linear cost subspace nearest neighbour method. FASTOUT and its variant FASTOUT-R are shown as effective classifiers for multi-class problems.
- An algorithm for computing false positive and false negative rates for complex experimental scenarios (SERA) presented in Appendix A.

In addition, Hypothesis 3 – classification through variance – is shown to be theoretically feasible and practically attainable with very high effectiveness.

1.4 High-Dimensional Data Mining

High dimensionality has become important recently with the vast increase in multi-attribute data collection. While this is widely acknowledged, there is still a strong tendency to continue with the tools and approaches that served us well in low dimensionality. As we discuss, high-dimensional spaces are fundamentally different from low-dimensional ones. In high-dimensionality. the approximations we employ to achieve high efficiency break down. For example, instead of a handful of outliers, every point in the dataset becomes an outlier candidate. Finding candidates for clustering together is questionable.

While there has been much concern over the ‘curse’ of high-dimensionality, certain ‘blessings’ can be identified [55]. However, these have scarcely been exploited before now for clustering and outlier detection. This thesis identifies a particular advantage of high dimensionality and presents several approaches to exploit it.

When data is clustered, some points may be identified as singletons or members of small clusters. Thus, clustering can lead to the identification of outliers. In this work, for the first time, we show how and why outlier detection algorithms can be used for clustering in high-dimensionality. In this context, we also put forward a new definition of an outlier in high-dimensional space.

Another reason for addressing both problems, outlier detection and clustering, is that the challenges are essentially the same, particularly with respect to high dimensionality. As dimensionality increases, sparsity tends to increase exponentially. With this, identifying clusters becomes much harder due to low density while ranking points as outliers is similarly challenged. Both problems usually involve density determination or the computation of nearest neighbours. In principle, the algorithms only differ in whether points/regions are collected due to high or low density. Therefore, the main issues of interest are substantially overlapping.

1.4.1 The Nature of the Problem

Let us assume a dataset $D = \{P, A\}$ with points P and attributes A, where $n = |P|$ and $m = |A|$. This can be represented by an $n \times m$ matrix containing values $x_{ij}, 1 \leq i \leq n, 1 \leq j \leq m$, where x_{ij} is the value of point $i \in P$ on attribute $j \in A$. The traditional clustering seeks clusters C consisting of sets of points such that the intracluster similarity is maximised and the intercluster similarities are minimised and all the attributes are used to determine the similarity between points. However, as datasets with larger dimensionality m became common it was obvious that this approach was not satisfactory. As the number of attributes increases, the utility of the full attribute space as a discriminant between points declines [31]. Beyer et al. [31] showed that the difference between the distance of a point to its furthest and nearest neighbours goes to zero as the number of dimensions goes to infinity in all but a highly restricted class of problems (See Section 1.4.2). Further, in most cases, only a subset of attributes is relevant in distinguishing a cluster and indeed, knowing which attributes are relevant can be of great importance. For example, in a genome microarray, there may be 10,000 genes but usually only a very small number, often just one or two, are really discriminatory for a given disease condition.

Thus, the subspace clustering problem can be defined as finding a set of clusters C, based on the same principle of maximizing intracluster similarity and minimising intercluster similarity, such that for any cluster numbered k, $C_k = \{P', A'\},\ P' \subseteq P,\ A' \subseteq A$. That is, a cluster consists of a set of points and a set of attributes. Within the subspace of those attributes A' a clustering is observed of the set of points P'. Even if in the full attribute space, P' are not statistically significant as a cluster with respect to mutual similarity and dissimilarity to others, they are significantly so in the subspace A'. The problem can be cast in terms of regions such as a cover of hyper-rectangles over any C_k but translating between these two definitions is just a matter of some post-processing and is not fundamental to any algorithm[1].

The subspace outlier detection problem could be defined in an identical form as finding pairs $\{P', A'\}, P' \subseteq P, A' \subseteq A$. A pair is a set of points P' that are especially 'unusual' in their respective subspace A'. The points P' will not form a cluster unless it is very small below some specified threshold ϕ. These points may not be significantly unusual in the full-dimensional space but are significantly so in the subspace.

1.4.2 Complexity and Solutions

Clustering

Optimal partitioning of data, even in two dimensions, is inherently hard. It can be expressed as a number of problems that have been shown to be NP Hard such as optimal partitioning of a graph [22], the Discrete Clustering Problem [56] or clustering of the elements of a dissimilarity matrix (the Bandwidth Minimisation Problem (BMP) [74]). Therefore, clustering algorithms that attempt to provide an optimal partitioning of the data can only be efficient and accurate on those classes of problems that are not inherently hard. Thus it is important to understand what are the properties of those classes.

Fortunately, the work done on the BMP provides us with the necessary insight. The BMP is usually treated as a graph theoretic problem. The data points are the nodes and their similarities are the edges. Results have been found for the simplest case where the existence of an edge is either true or false. The bandwidth problem is hard in most classes of graphs. Papadimitriou [144] showed that it is NP-Complete for general graphs. Garey et al. [74] showed that the BMP remains NP-Complete even when restricted to trees of maximum degree 3. Even more restrictive cases have been shown to be NP-Complete [138, 139]. Chain graphs, interval graphs, 1-Caterpillars and (probably) bipartite permutation graphs have been shown to be polynomially solvable. However, the severe constraints on these types of graphs means that any real world partitioning problem will be hard. If the clusters we identify C^* are to approximate the true clusters, or partitions of the graph, C and if we could construct a graph considering each member of C as a node and retain all the edges between points as edges between nodes, then for the instance to be polynomially solvable the graph must fall in one of the classes just mentioned. In particular, if there are no intercluster edges, the clusters can be recovered in linear or log linear time (depending on the algorithm employed) by selecting any point and simply assigning all those other points with edges to it the same class number.

More formally, for a dataset D containing N points and a set of clusters C the data points D_i, D_j

[1]See Chapter 2 for different views on this

should be such that for all $\{i, j\} \mid 1 \leq i, j \leq N$

$$Similarity(D_i, D_j) = \begin{cases} 1 & \{D_i, D_j\} \in C_k \mid 1 \leq k \leq |C| \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

That is, links only exist intra-cluster and not inter-cluster. Then it is trivial to show that a clustering C can be found in $O(Nlog(N))$ or better steps. This is the starter problem used by researchers in synthetic datasets. What if we added some noise or even some inter-cluster links? As long as there exists some positive threshold ϕ such that

$$Similarity(D_i, D_j) \begin{cases} \geq \phi & \{D_i, D_j\} \in C_k \mid 1 \leq k \leq |C| \\ < \phi & \text{otherwise} \end{cases} \tag{1.2}$$

i.e. the intercluster links are above some threshold while the other links are below it, we can readily transform problem instance 1.2 into 'easy' problem instance 1.1 by applying a filter at value ϕ.

Many heuristics employed by clustering algorithms to increase efficiency follow this approach. In some cases, edges between points that have weights below the threshold ϕ are removed and then the points collected into clusters. In others, such as the density based methods TURN* [66] and DBSCAN [60] a point is selected and then all those other points that are connected to it by sufficient density are classed with it. The density threshold, however computed, at which the algorithm stops any line of point collection for a particular cluster or class amounts to the threshold ϕ. Many refinements are possible and have been explored but the essential idea is the same.

Subspace Search

It is important to note that subspace clustering involves two separate problems, searching for subspaces and clustering. The total number of subspaces s in D is

$$s = \sum_{k=1}^{n} \binom{n}{k} = 2^n - 1 \tag{1.3}$$

Given a uniform distribution of N points in a hypercube of edge size 1 and k dimensions, the number N' of points in a subspace of the same dimensionality and width $l \leq 1$ is

$$N' = Nl^k \tag{1.4}$$

Thus sparsity typically increases exponentially exactly as the increase in the number of potential subspaces. All current high-dimensional clustering approaches leverage this to restrict the search space implicitly or explicitly (unless applying local search). As the density can be expected to decline monotonically (i.e. is anti-monotonic), defining a fixed density threshold effectively accomplishes this. For a subspace S, requiring all its subspaces to be dense according to some global threshold, the 'a priori' approach, similarly restricts the space. Thus only small subspaces are typically found but this can be done efficiently.

Noting the decreasing expected density of larger subspaces and thus the limiting effect of a fixed density threshold, Assent et al. [95] attempted to construct an algorithm that adjusted the threshold. However, abandoning a downward closure property prevented an efficient solution and they had to reintroduce it later in the paper.

Earlier (Section 1.4.1) we referred to the work of Beyer at al. [31]. They show that if p, q are independent data points with arity m and d_m is a distance function and $DMIN_m = min\{d_m(p_i, q)\} \mid 1 \leq i \leq n, DMAX_m = max\{(d_m(p_i, q)\} \mid 1 \leq i \leq n$ and if, for any constant $p, 0 < p < \infty$,

$$\lim_{m \to \infty} var \frac{(d_m(p_1, q))^p}{E[(d_m(p_1, q))^p]} = 0 \tag{1.5}$$

then for every $\epsilon > 0$

$$\lim_{m \to \infty} P[DMAX_m \leq (1 + \epsilon) DMIN_m] = 1 \tag{1.6}$$

They showed empirically that for as little as 10-20 dimensions linear scan outperforms tree indices frequently employed to achieve sub quadratic distance calculations. They assert that for NN (nearest neighbour) queries to be meaningful in more than 10-20 dimensions, the data must consist of small, well-formed clusters and the query must be guaranteed to land within or very near one of these clusters. The reader will note something not previously mentioned in the literature, that this requirement is exactly analogous to the problem classes defined in equations 1.1 and 1.2 based on the formal results for the BMP. Whether we look at the low or high dimensional clustering problem a tractable solution for the broad class of problems requires that the intracluster distances or edge weights, if treated as a graph, have to be almost entirely removable by some heuristic.

1.4.3 Outlier Detection

A common problem in many data mining and machine learning applications is, given a dataset, to identify data points that show significant anomalies compared to the majority of the points in the dataset. These points may be noisy data, which one would like to remove from the dataset, or may contain information that is particularly valuable for the identification of patterns in the data.

Outlier detection, originally a domain of statisticians, deals with the problem of finding such anomalous data, called outliers. There is no unique established definition of an outlier and different applications may merit different definitions. This, and the general issue of increasing sparsity, can be particularly problematic as dimensionality increases.

In this thesis, a new conceptualisation of outliers in high-dimensional datasets is introduced. The basic idea of the approach is to accumulate an outlier score over all subspaces of a fixed (low) dimension k. First a standard definition of outliers in k dimensions (in our case a density-based definition, but it could be replaced by an arbitrary definition) is used to determine outliers in all these subspaces. Second, every data point is assigned an *outlier score* equal to the number of subspaces in which it was classified to be an outlier. This outlier score is finally used for ranking points with respect to their outlierness. Such an ‘ensemble’ approach has been taken before but only for single dimensions or small random samples of the majority of dimensions (See Section 2.3.1).

The motivation behind this approach is that points that are outliers in only a few low-dimensional subspaces could be so just by chance. If a point tends to be an outlier consistently over almost all subspaces of a given dimension k, we would much rather consider it to be a true outlier in the high-dimensional dataset. Thus, we are proposing a new definition for outliers in high-dimensionality, which is in line with and extends earlier definitions, as follows,

A high-dimensional outlier is an observation that consistently deviates from other observations in multiple subspaces in the global multi-variate space.

For sufficiently small k this approach is very efficient. We will show good effectiveness for two methods T*ENT and T*ROF which find outliers in high-dimensional data using only 2D subspaces. It is natural to ask whether extending to 3D and higher can yield further benefit. In some scenarios the answer turns out to be ‘yes’ and in the algorithm FASTOUT we provide a way of not only utilising higher dimensionality subspaces but even determining the optimum such size. One reason for using 2D is the exponential cost in the subspace size of enumerating subspaces but our experiments show that for data containing full-dimensional clusters, processing only a small sample of subspaces is sufficient to maintain high effectiveness.

While this ensemble approach gives an outlier ranking, it also opens an approach to separate underlying classes even if heavily overlapping. For the first time, this concept is explained and investigated along with new algorithms that utilise it to accomplish classification very effectively and efficiently.

Chapter 2 surveys the related work in clustering and outlier detection. Chapter 3 introduces the subspace clustering framework MAXCLUS and an implementation of it, GCLUS, along with a classifier, GSEP, which is used as a preprocessor for very high-dimensional problems. After that, outlier detection issues are addressed. Chapter 4 discusses some fundamental issues – what is the primary cause of outlierness in high dimensionality? How can we measure it and what, ultimately, are we measuring? Several novel outlier algorithms are introduced and their effectiveness and efficiency tested. Appendix A analyses issues regarding error rates and provides an algorithm for estimating them and some examples of how this can be applied.

Chapter 2

Related Work

2.1 Subspace Clustering: A Methodological Review

Subspace clustering, generically, is the task of finding clusters within subspaces or sub sets of dimensions of a dataset. As such it subsumes all methods that use less than the full dimensional space (more restricted definitions of this term will be referred to in Section 2.1.2).

Useful reviews of subspace clustering have been given by Kriegel et al. [111, 112] and earlier by Parsons et al. [145] and Huan et al. [124]. As made clear in the latter, there are two principal techniques applied to this problem. The first is feature transformation and the second feature selection. In feature transformation linear combinations of the features (dimensions) are formed using techniques such as Principal Component Analysis (PCA), Singular Value Decomposition (SVD) and Random Projection [64]. This works well in certain circumstances but also faces certain problems. These techniques usually require the computation of a covariance matrix, which is expensive, and retain all the features. The later means that subspace clusters can still be obscured by redundant features and the final result is more difficult to interpret. This is because the original features are combined so the results are not so easily associated with a specific feature set. Clustering algorithms incorporating this approach include the work of Ding et al. [53], Hinneburg and Keim [88], and Fern and Brodley [64].

Feature selection approaches can be divided algorithmically into Bottom-up and Top-down approaches (see Figure 2.1). From a problem orientated view, the methods can be seen as 'Subspace' (looking for all clusters in all subspaces), Projected, Hybrid and Bi-clustering [111]. There are also methods for non-axis parallel clustering. While a problem orientated view is arguably more satisfying, it is more difficult to neatly assign algorithms to this view and the distinctions between the types are also debatable.

Top down methods operate on the whole space iteratively adjusting weights on the dimensions based on the results of each stage until the results stabilise. Bottom up methods build subspaces starting with assessment of single dimensions. Typically dense units are sorted in each dimension and their projections into two and higher dimensions are collected. This is the method adopted by

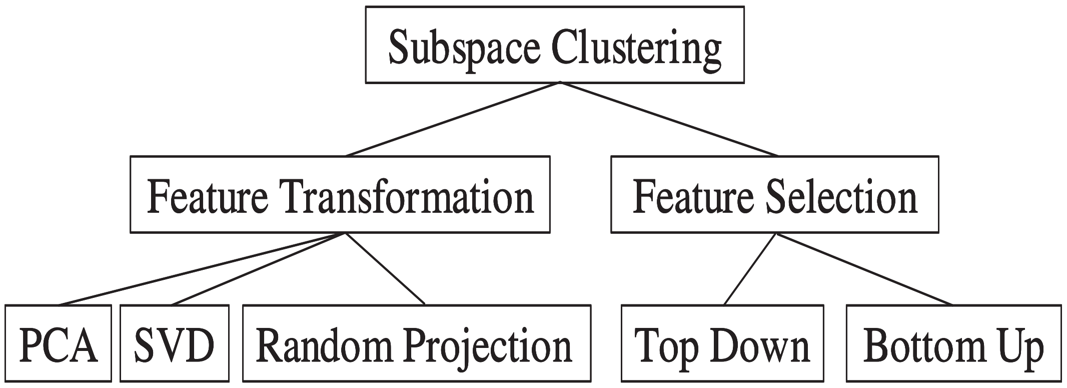

Figure 2.1: High level ontology of Subspace Clustering.

CLIQUE, the first of this type of algorithm [11]. 'Subspace' and Projected clustering are explained below (Section 2.1.2). Hybrid methods are those that incorporate elements of both clustering types. Bi-clustering was developed by Cheng and Church [48]. Other examples are [49, 27, 170, 126, 146]. It captures the similarity or coherence exhibited by a subset of objects on a subset of attributes while allowing for missing values. Data where there are strong linear or non-linear correlations between attributes is addressed by non-axis parallel methods or correlation clustering [8].

2.1.1 Feature Selection

Feature selection involves searching for feature subsets that meet some criterion. Commonly a greedy sequential search is performed through the feature space (e.g. [123, 147, 179]). Exhaustive search is intractable so some heuristic has to be employed to constrain the search space. Approaches generally fall into two categories, wrapper and filter methods. The wrapper methods use the mining algorithm iteratively adjusting weights on the dimensions to optimise the output. Filter approaches use some method to select features. Dash et al. [52] use entropy and make the point that using wrapper approaches leaves the process dependent on the parameter settings on which almost all clustering algorithms depend. At the same time they require a measure of goodness of clustering about which there is no general agreement in the unsupervised case. Their filter approach is based on the idea that spaces with clusters have lower entropy than those where the points are more evenly spread. They define entropy in terms of point to point distances, dense regions having many examples of low distances.

All those doing filtering of features in an unsupervised way have to get some measure of the likelihood of clusters existing in the subspace. Trying to estimate clustering without clustering as done, for example, by Dash et al., is certainly more approximate than an actual clustering and could be confounded by, for example, many small noise clusters. Their method involves many heuristics and parameters and they try to avoid the quadratic cost of distance computation by using a grid; but this faces even more severe computational costs in high dimensionality. This illustrates that any method that attempts to handle the full dimensional space in order to locate the subspaces, runs into great difficulties unless the dataset is especially suited to the method.

Kim et al. [104] discuss the fact that many different criteria can exist as to the goodness of clustering in a subspace and propose a method that employs the Pareto front, in effect a voting scheme among the various criteria. They approach the search problem using an Evolutionary Local Selection Algorithm (ELSA). Others have taken to random searching [2, 32, 64]. Fern and Brodley [64] point out that while random projection has promising theoretical properties it results in highly unstable clustering performance. They attempt to resolve this using a cluster ensemble approach. That is, they perform a series of random projections into a lower dimensional space followed by clustering and then combine the results. Random projection involves projecting the full space using a randomly generated transformation matrix. While this may reveal that a set of points are clustered together, the subspace in which this cluster actually exists is not known. This makes the results difficult to interpret.

2.1.2 Subspace Search

Subspace search methods can be divided into 'top down' and 'bottom up'. Kriegel et al. [111] distinguish between subspace clustering and projected clustering. According to [111], a subspace cluster is a region in some subspace with density larger than a given threshold. They formulate the subspace clustering problem as locating all such clusters. A projected cluster is the pair $\{C, D\}$ where C is a set of points and D is a set of attributes such that the data points in C project onto each attribute in D over a range that is distinguishable from the projection of the entire dataset and is not so distinguishable over the other attributes not in D. In practice, however, this distinction is moot as all the methods do or could output their results as a set (or set of sets) of $\{C, D\}$ pairs.

Projected Clustering

In CLIQUE [11] each dimension is partitioned into ξ equi-width bins. It scans the dataset once building the dense bins in each dimension. Then it combines the projections on to these bins, checking them against the database to ensure they are also dense, to build larger and larger subspaces. As CLIQUE exhaustively searches, the cost increases exponentially with the size of the subspaces searched. CLIQUE then computes minimal cluster descriptions by first greedily covering the region by a number of maximal rectangles and then discarding the redundant rectangles to create a minimal cover. This is then output as a DNF description. While CLIQUE outputs descriptions of dense regions, no attempt is made to distinguish subspaces containing specific sets of points or the clustered points themselves. Like all methods of this type, CLIQUE utilises an anti-monotonic property to limit the search space. By setting a global density threshold, an apriori approach can be adopted. Starting with 1 dimensional dense units that exceed the density threshold, the subspace size is increased such that $k+1$-dimensional regions are constructed using only dense k-dimensional regions.

ENCLUS [47] is a version of CLIQUE that uses an entropy measure rather than simple bin density. Subspaces with entropy below some threshold ω, suggesting that the points are more tightly packed, are considered good. This measure is anti-monotonic and is used for the apriori-like approach. CLIQUE and ENCLUS use a fixed grid size while the other group of bottom-up methods adapt the grid to the data. For example, a high frequency grid can be imposed and then adjacent bins with similar densities above the threshold are merged as is performed in the MAFIA algorithm [77]. Otherwise it is much like CLIQUE. Other adaptive grid algorithms are CBF [44] and CLTree [121]. CBF is faster than CLIQUE but at a small cost in precision. CLTree [121] uses a decision tree approach comparing relative emptiness with fullness and an information gain criteria to repeatedly partition the space into hyper-rectangular areas. Typically for a decision tree this approach tends to go too far and needs pruning. The authors give an example of a space containing two clusters that is partitioned into 14 areas. nCluster [122] is similar to CLIQUE except that it starts with overlapping windows of length δ. Unfortunately, this paper provides very limited empirical results.

Fromont et al. [70] investigated if adding semi-supervision in the form of 'must-link' and 'cannot-link' constraints to CLIQUE and nCluster improved their performance. Theses constraints add expert knowledge to the unsupervised clustering to ensure that pairs of point either appear together or do not.

The main difficulty of CLIQUE and similar approaches is in setting the key parameters.

Density

In principle density-based algorithms could find all subspace clusters even though clusters can be missed due to the various heuristics. Some algorithms do not attempt to find all clusters.

DOC [150] defines a global density threshold by requiring a minimum number of points α in a hypercube of fixed side w. DOC employs a random search starting from $2/\alpha$ randomly sampled seeds that define the cluster's hypercube. Randomly selected points are added to the seeded clusters. Attributes are deemed relevant if all points in the tentative cluster fall within the hypercube over those attributes from the seed and subsequently from the mediod. Clusters are deemed optimal if they maximise both the number of points and the number of relevant dimensions. A parameter β determines the relative importance of subspace dimensionality over the number of points. A single cluster is found with a certain probability and multiple runs are required to find multiple clusters. Results are very dependent on parameter settings. DOC combines the grid approach of bottom up methods with an iterative approach common to top down methods. In order to improve its efficiency FASTDOC [150] was proposed, which uses several heuristics at the cost of the guarantee of accuracy provided by DOC. Sensitivity to parameters and use of a single value w can be seen as drawbacks.

MINECLUS [178] is a development of DOC. Starting from a random seed, an optimal projected cluster is found by a deterministic process. They convert the problem into a frequent itemset mining task and the FP-Growth method [81] is used.

PRIM [68] is like DOC and MINECLUS in that it computes one dense axis-aligned ‘box’ at a time. Firstly, the box B covers all the data and then a smaller box $b*$ is removed such that $B \setminus b*$ yields the largest output mean value. This is referred to as ‘peeling’ and stops when the output box has less than a user-defined $mass\_min$. Then a ‘pasting’ process takes over where attempts are made to add boxes $b*$ in an inverse process with the same objective. A number of additional parameters are required to guide the process to which it appears rather sensitive.

DOC, MINECLUS and PRIM could be considered as a class of hyper-cube based techniques.

PreDeCon [34] is a development on the full-dimensional density-based clustering algorithm DBSCAN [60]. It assigns a subspace preference to each point p using the maximal dimension subspace which provides the best clustering of p. The subspace preference is based on the variance of points in the ϵ-neighbourhood of p and a threshold δ. Relevant dimensions are weighted by a constant $\kappa \gg 1$ while other dimensions are assigned weight 1. PreDeCon, like DBSCAN, does not require to know the number of clusters, can find arbitrary shapes and handles noise. On the other hand it can be sensitive to its input parameters, which may be hard to set.

EPCH [141] computes 1D or 2D histograms in order to identify ‘dense’ regions. A threshold for each histogram with mean μ and standard deviation σ is defined as $\mu + c \times \sigma$. This threshold is set high and dense regions identified and removed. This is iterated while the threshold is progressively reduced until no dense cells are detected. Neighbouring dense regions are then merged. Each point is assigned a ‘signature’ containing the dense regions it belongs to. Points are associated with others with similar signatures. This clustering goes on until at most $max\_no\_clusters$ is obtained.

P3C [137] initially computes 1D regions that are candidate projections of clusters. These 1D regions are combined into hyper-rectangular approximations of ‘cluster cores’ based on a statistical test. This test is based on the comparison of the real and expected number of points in the cluster core and is controlled by the parameter $Poisson\_threshold$. The cluster cores are then refined using the EM algorithm and outliers identified. Both disjoint and overlapping clusters can be computed. Like all algorithms depending on 1D computations, P3C is challenged if the 1D regions cannot be correctly computed. Also, if the clusters have low density, then the statistical tests are compromised.

FIRES [110] is a bottom-up framework that starts with 1D clustering using any method selected by the user. Small base clusters are dropped using an empirically determined threshold of 25% of the average base cluster size. Base clusters of low quality are pruned with quality being a function of size and dimensionality. A k-nearest neighbour graph is induced on these base clusters using the number of shared points. This graph is clustered using any clustering algorithm of choice such as DBSCAN, which is employed in the paper. The algorithm appears sensitive to its three main parameters, k, ϵ and $minPts$.

DBSCAN [60] detects arbitrarily sized clusters using the concept of reachability - points connected by regions of high density are clustered together. SUBCLU [99] runs the DBSCAN [60] clustering algorithm in various subspaces using the anti-monotonic nature of reachability to prune the search. It inherits the advantages of DBSCAN along with its sensitivity to its parameters.

SCHISM [156] applies a grid to the data space and looks for 'interesting' subspaces defined as having more points than expected under uniform distribution. To realise this concept a global threshold is defined for subspaces larger than v which is a function of the grid width and data size n. This permits pruning using the anti-monotonic property of a global density threshold. Below v, a variable threshold that is the minimum of a global threshold u and a statistical computation, based on the dimensionality and n, is used. This does not permit any guarantee of an anti-monotonic property and, therefore, finding all interesting subspaces. A depth-first search with backtracking is done starting from the interesting 1D spaces. To overcome redundancy in the subspaces detected, similar subspaces are merged based on a similarity parameter. Using a statistical measure is intuitive but the complexities and parameters pose a challenge. Results are reported in terms of coverage and entropy. Coverage is defined as the number of points which are accurately labelled as not being outliers. For a clustering C, entropy is defined as $E(C) = -\sum_{C_j}(\frac{n_j}{n}\sum_i p_{ij}log(p_{ij}))$, where $p_{ij} = \frac{n_{ij}}{n}$, C_j is the j^{th} cluster in C, n_j are the number of points in C_j and n_{ij} is the number of points in C_j that actually belong to class i.

DUSC [95] folows SUBCLU's definition of a cluster except that each point has a density measure and is considered a core point if its density is F times larger than the expected value under uniform distribution. Initially, the authors objected to global thresholds as they do not reflect the rapidly declining expected density of larger subspaces. On the other hand, being unable to prune the search space without an anti-monotonic property, their view was modified to introduce a global density threshold. The same authors [96] subsequently proposed depth-first search rather than the breadth-first search of typical apriori approaches together with merging of clusters, which improved the run time and removed redundancy in the process of clustering.

Unless only the maximal subspaces are retained, a hierarchy of subspaces will be discovered by many algorithms. DiSH [4] looks for the largest subspace (highest dimensionality) that is associated with each point p. On each attribute, the η-neighbourhood of each point with more than μ points is retained as 'relevant'. A similarity measure is defined between points depending on their number of shared relevant attributes. A visualisation tool is provided for the output.

Partitional

Partitional clustering attempts to decompose the data into a set of disjoint clusters usually guided by a target number k. PROCLUS [7] was the first of its type. It uses the k-medoids technique as in the clustering algorithm CLARANS [142] and uses a locality analysis to find the set of dimensions associated with each cluster. It requires k, the number of clusters and l, the average dimensionality of the subspaces. A random set of k well separated medoids is selected and a set of dimensions is chosen for each medoid based on the points closest to it and the dimensions on which these are most close. Points are then assigned to their nearest medoid based on the chosen set of dimensions and Manhattan segmental distances. The total number of dimensions is limited to kl. The clustering is evaluated, highly disperse medoids replaced, and the process iterates. The method tends to find hyper-spherical clusters and similar sized subspaces and is sensitive to the parameter settings. PROCLUS uses sampling which improves speed at the risk of missing some clusters. ORCLUS [8] extended PROCLUS to seek non-axis parallel subspaces but the running time is cubic in the dimensionality.

FINDIT [167] is short for a Fast and Intelligent subspace clustering algorithm using Dimension voting. The key idea is that closeness in many dimensions is more important than being very close on a few. FINDIT uses a medoid approach and sampling for better speed.

SSPC [175] is based on k-medoids and is similar to PROCLUS with the relevance score of HARP [174]. That is, an attribute A is relevant if the variance of the set of point values X over A is m times the total variance over A. The objective function ϕ is maximised when $s_{ij}^2 + (\mu_{ij} - \hat{\mu}_{ij})^2$ is smaller than a threshold $\hat{s}_{ij}^2$ where $s_{ij}^2, \mu_{ij}, \hat{\mu}_{ij}$ are the sample variance, sample mean and sample median over cluster i and attribute j. A number of mediods are selected as cluster seeds and points are added such that ϕ is maximised. Points that don't improve ϕ are marked as outliers. The 'worst' cluster of the 'best' clusters discovered so far is recomputed in the next iteration. SSPC benefits when some training data is available but can cluster without training data. Our empirical results show it to be quite effective but, without some labelled data, only some runs find the correct solution due to the random start. It also needs to know k and certain other parameters.

Top-down and Model Based Approaches

Top-down approaches usually set weights on each dimension and then refine them. In model-based clustering the data are assumed to be generated by a mixture of distributions, each representing a cluster. Domeniconi et al. [54] introduced a Locally Adaptive Clustering algorithm (LAC) which starts from a set of k centroids and a set of weights and adapts the weights to approximate a group of k Gaussians from the dataset while Candillier et al. [41] use a maximum likelihood estimation approach (SSC) to approximate the data as a set of k Gaussians. An example of using an objective function is Tseng and Kao's CST [160] which employs a simplified Hubert's Γ statistic to validate the clustering. COSA [69] is an offering from the statistical community, which uses iterative runs of any clustering algorithm to adjust the feature weights with a penalty based on the negative entropy of the weight distribution.

FPC [45] models the data as a mixture of K 'extended' Gaussians. The extended distribution has weights w_{ik} on each attribute i based on its relevance to the Gaussian component k. A threshold is required for selecting relevant attributes from the weights. The clustering process uses the maximum likelihood principle and the iid assumption for the data points on every attribute. FPC can generate ovelapping clusters but is sensitive to its initialisation.

STATPC [135] looks for axis-parallel dense regions that make sense statistically. Since there could be considerable redundancy in the set of such regions, a reduced set is generated that 'explains' the total set, Achieving this is couched as an optimisation problem and approximated, as an optimal solution is intractable. Results are based on statistical measures. A major advantage is that the only parameter setting required is the level of error that one is prepared to accept.

Hierarchical

Hierarchical clustering methods build a tree of clustering results using a top-down divisive or a bottom-up agglomerative approach. Agglomerative approaches are more common and typically start with each point as a separate cluster and progressively combine them.

Yip et al. developed HARP [174] and then the more efficient SSPC [175]. HARP is a hierarchical clustering method. Its intuition is to maximise the number of relevant attributes per cluster. Relevance of an attribute a for a cluster C is defined as $R(C, a) = 1 - \sigma^2(C, a)/\sigma^2(D, a)$ where D is the whole dataset. The merging process optimises the relevance of the merged cluster. Starting with each point as a cluster the internal merging thresholds are progressively loosened until the desired k clusters is achieved. While intended to resolve the problem of mandatory parameters it still requires two and is very slow due to the hierarchical approach.

Biclustering

δ-clusters [170] brought out the concept of coherent clusters. It captures the coherence exhibited by a subset of objects on a subset of attributes while allowing for missing values. They present the FLOC algorithm and claim it efficiently produces near-optimal clustering. This approach is motivated especially by bioinformatics where finding related genes, say, means seeing similar response patterns even if the absolute response values differ. They claim that the biclustering approach of Cheng and Church [48] is a special case of their δ-clusters. The method starts from k seeds and is quadratic in N and D. Other developments on the idea of biclustering are [169] and Melkman and Shaham [133] who use a randomised approach to avoid an exhaustive search. Li et al. [119] perform subspace identification in document clustering via an iterative alternating optimization process. This literature is quite extensive. A comprehensive survey of linear and non-linear correlation clustering is provided by Kreigel et al. [111] and Madeira and Oliveira [127].

Categorical

A few algorithms are specifically intended for data with categorical attributes. STIRR [75] seeks to find associations beyond traditional frequent itemset mining as follows. An item of interest is given a weight, this weight propagates to those items that co-occur with it in frequent items. From those items the weight further propagates and this process iterates in a manner similar to applications of spectral graph theory. After each iteration, the sets of weights from each iteration are adjusted so that they remain orthonormal. The second and later sets represent non-principal basins and contain both positive and negative weights. The extremes of these tend to be associated with well-separated clusters. STIRR envisions a dataset D as a simple undirected graph with attribute values as nodes and edges E between attribute values that co-occur in a transaction. This concept was taken up later by the authors of both CACTUS [73] and CLICK [182].

SUBCAD [72] partitions the data into a user specified k partitions based on an objective function which within full space Q includes compactness within a subspace P and separation in $Q \setminus P$. It initialises with k well scattered seeds based on a sample and full dimensionality dissimilarity and then matches points to the closest seed based on the objective function.

CACTUS [73] follows STIRR in viewing the data as a graph between attribute values. The graph is modified by pruning all edges that represent less than some threshold α times the expectation of co-occurences. Then the cluster-projection on each attribute of up to all attribute pairs containing the attribute (the intersection) is computed. In short, cliques are identified where nodes are single attribute value sets and subsequently sets of attribute (value sets) of increasing size.

The CLICK [182] concept is very like CACTUS. A somewhat different method is used to find maximal k-partite cliques in the attribute value graph. As a post-processing, given a parameter $minsup$, they try to cover the maximal number of tuples with the minimum number of cliques.

2.1.3 Conclusion

As we see there are a number of different approaches with advantages and drawbacks. Hierarchical algorithms are very slow and require key parameters. Projected clustering is often faster but suffers from the exponential cost of searching all possible projected subspaces, the answer is pruning but this can cost accuracy. k-medoid approaches are very dependent on the initial seeds and also have key parameters. Like all methods that optimise an objective function there can be many iterations making the running time impractical for larger problems and the algorithm may stop in a local minimum. Algorithms with an objective function are also usually biased toward spherical clusters. If well-defined clusters exist in some subspaces, it should be possible to find them in an efficient, accurate and deterministic fashion. This is one motivation for this thesis.

2.2 Subspace Clustering: An Effectiveness Review

As we discuss in Section 1.4.2 the problem is a combination of two intractable problems. Optimal partitioning of data is hard with respect to the number of datapoints, and searching the space of possible subspaces is intractable with respect to the number of dimensions. However, we have seen that a hard clustering instance can sometimes be transformed into an easy one and the subspace search can also be achieved efficiently through the use of an anti-monotonic property. Thus, all successful algorithms will contain heuristics for achieving this on a wide variation of datasets. In order to validate if the algorithm is in fact achieving this, we need to present it with a genuinely hard problem where success can be quantified. As reviewed more formally later, synthetic datasets are typically 'easy' instances of the problem while real world datasets are typically 'hard' (An instance is 'easy' if the clusters are dense and disjoint). Therefore we can sort our review of the existing approaches into two categories based on the author's own or other's evaluation. *Type A: Algorithms validated on labelled data, preferably publicly available datasets;*

Type B: Algorithms validated on synthetic data.

2.2.1 Type A – Algorithms validated on labelled data

This category considers quantitative accuracy results on datasets accessible to other authors. However, since this is quite rare, we also include work that provides some degree of reproducibility on, ideally, publicly available datasets.

Achtert et al. (COPAC) [5] report results for the original (n = 699) Wisconsin Breast Cancer Database [33]. They list the sizes of 6 pure clusters and one 'noise' cluster by class but do not discuss the nature of the subspaces identified. Assent et al. (DUSC) [95] give results for four public datasets (glass, pendigits, vowel [33], shapes [101]). Unfortunately, two of these (vowel, shapes) are not unambiguously referenced. Both of these are also pattern recognition problems for which unsupervised methods would not normally be particularly successful as the classes are heavily overlapping and individual class members may be phase shifted relative to each other. DUSC did surprisingly well on one and less so on the other but, perhaps due to lack of space, the results are hardly discussed, nor is the nature of the subspaces found.

Moise et al. (P3C) [137] report 67% F1 accuracy on a colon cancer dataset. The F1 accuracy of a cluster is the harmonic mean of its precision and recall. They also discuss qualitatively results on a housing dataset. Elsewhere [136] Moise reports a 55% accuracy on the UCI Glass dataset and just over 60% on the *E. coli* dataset [33] (accuracies approximately inferred from graphs). P3C for categorical data was tested on the Congressional Votes (90%), Post-Operative Patient (82%), Hepatitis (70%), Contraceptive Method Choice (59%), and Flags (45%) UCI data sets [33]. Moise et al. subsequently developed STATPC [135, 136] which was tested on Pima Indians Diabetes (75%), Liver Disorders (59%), Wisconsin Breast Cancer Prognostic (WPBC) (63%), Glass (60%) and Iris (80%) [33]. STATPC was also tested on the colon tumour dataset [16] (66%) and leukemia dataset [78] (74%).

PRIM [68] was tested on a database of Garnets and a marketing questionnaire dataset neither of which appear to be readily available. Selected results were presented showing reasonably high effectiveness. It was also compared with CART [37], a decision tree induction algorithm, and generally outperformed it.

MINECLUS [178] was tested in comparison with PROCLUS and DOC on the Iris, Soybean, Votes and Mushroom UCI datasets [33]. With the right parameters all methods returned accuracy better than 84% on these though DOC was too slow to complete the Mushroom dataset.

EPCH [141] was tested on the UCI Image Segmentation dataset [33] with 180 records and 19 numerical attributes. Confusion matrices are given for EPCH and ORCLUS with EPCH performing best.

Achtert et al. (HiSC) [3] clustered a labelled dataset of metabolic screening data (43 attributes) of 2000 newborns (apparently not publicly available). They provide a reachability graph and found many pure or nearly pure clusters along with some mixed clusters though quantitative results are not given. Similar success was reported for 4C by Böhm et al. [35].

Sequeira and Zaki [156] compared their method SCHISM with CLIQUE on a PenDigit [14] dataset which contains 7,494 16-dimensional vectors. The results are presented as a confusion matrix. SCHISM's best result was a 69% coverage and 0.365 entropy (defined in Section 2.1.2) while for CLIQUE they obtained 60.7% and 0.49.

Fromont et al. [70] used labelled data and selected pairs of labels at random to construct constraints and modified CLIQUE and nCluster to handle these. Graphing coverage and quality (inverse entropy) on the PenDigit, Image, Glass [33] and Shapes [102] datasets, they show that coverage decreases and quality increases with increased constraints as would be expected.

A cluster of papers have attempted yeast gene expression datasets. This data is unlabelled but there exists a database (SGD) [125] of known relations between genes so some authors [34, 110, 48, 170, 8, 99, 35, 156] mention some of the genes that clustered together and that are known to co-regulate. While accuracy has not been quantified, some reproducible success is demonstrated.

CLICK [182] was tested on the Mushroom, Congressional Vote, Reuters 21578 [33] and two text datasets [155, 165]. Far more clusters were generated than labelled classes but many of these clusters were 'pure'. Success with the text data was limited but some meaningful clusters were derived.

CACTUS was also tested on a combination of the [155, 165] text datasets. Some qualitative results showing reasonable results are given. STIRR [75] was tested on the same text datasets and various results given to show that the output is reasonable. STIRR provides weights and not clusters. At best it separates between items with weight $w > 0$ and $w < 0$.

SUBCAD [72] was tested on the Wisconsin Breast Cancer (9 attributes) (WBC), Soybean and Congressional Voting datasets [33]. Accuracy on the malignant class of the WBC was 66.1%. The Soybean data has 47 samples and 35 attributes and only 3 samples were 'misclassified'. In the Congressional data 91.95% were correctly 'classified'.

2.2.2 Type B – Quantitative results on Synthetic Data

This category is for papers that give quantitative results on well-described synthetic data such that their experiments could be reproduced to a reasonable degree. End users would like to see accuracy results for both subspace discovery and cluster detection but as this is rare, partial results are in-

cluded. The overall dimensionality is referred to by *D*, the subspace size by d and the total number of data points by *N*. The synthetic clusters are assumed to be dense and disjoint unless specified.
Bottom-up: Agrawal et al. (CLIQUE) [11] searched for $d = 5$ dimensional clusters hidden in *D* = 5-50 dimensions. The clusters were correctly found but with some extra artifactual clusters sometimes reported by the algorithm. Two algorithms designed for full-dimensional clustering were also run on the same data, BIRCH [184] and DBSCAN [60]. BIRCH was effective up to *D* = 10 dimensions and DBSCAN up to *D* = 7 dimensions. An extension of CLIQUE, ENCLUS [47] was applied to *d*=5 clusters in a *D*=10 dataset with *N*=300K points. These clusters were correctly recovered. Goil et al. (MAFIA) [77] compared their method with CLIQUE on an *N*=400K, *D*=10, *d*=4 dataset and recovered the subspaces correctly. CLIQUE was slow but also found the subspaces. No data is given for the accuracy on point clustering but MAFIA is stated to have done much better than CLIQUE. Kailing et al. (SUBCLU) [99] compared it to CLIQUE on various small datasets with *D*=10-15, *d*=1-5 and 1-6 clusters. CLIQUE did poorly on subspace discovery while SUBCLU missed a cluster in two of the six datasets. Point accuracy is not given. SCHISM [156] was run on various datasets and achieved very low entropies with varying coverage depending on the parameter settings.

Moise et al. [137] compared their P3C with several algorithms on a *N*=10K, *D*=100 dataset with various sized clusters. P3C achieved full accuracy on finding the correct number of clusters and outperforms the other algorithms on point accuracy and subspace attribute identification as evidenced by graphs. Parsons et al. as an objective third party assessed MAFIA and FINDIT [167] on widely ranging synthetic datasets. Both algorithms did well but were not 100% accurate reflecting the approximation choices made. Woo et al., FINDIT's authors, on *N*=100K and *D* varying up to 50 reported high accuracy on points and subspaces with up to 50% noise. Liu et al. (CLTree) [121] tested on up to *N*=500K, *D*=20, *d*=2-20 achieved high if not 100% accuracy. Chang and Jin (CBF) [44] tested on *N*=1M, *D*=8-16 showed much faster execution than CLIQUE at a cost of slightly lower precision.

EPCH [141] was compared with PROCLUS and ORCLUS on synthetic data ($N \leq 200K, D = 20$) suited to the latter two methods and the advantage of EPCH was, in most cases, demonstrated.
Top-down: Aggarwal et al. (PROCLUS) [7] achieved quite high accuracy on synthetic data where *N*=100K, *D*=20 and clusters varied up to *d*=7. Conveniently, a confusion matrix is provided though it is only a partial report. Procopiuc et al. with DOC [150] provide an approximate way to discover a single cluster at a time. Even though expensive (cubic in *N* and *D*), it was shown how the algorithm could accurately detect the rotation of human face images. On *N*=100K, *D*=200, *d*=15-20, subspace accuracy was from 50-85% with high accuracy on data points. Böhm et al. with PreDeCon [34] (DBSCAN adapted for projected clustering) achieved full accuracy on *D*=2-50, *d*=2-10, N unknown. COSA on *N*=100, *D*=10K achieved good accuracy on detecting a subspace with *d*=15. Achtert et al. (DISH) [4] achieved 99% accuracy for precision and recall on *D*=3-16 datasets. It was compared with HiSC [3], PROCLUS and PreDeCon and better success is claimed without quantitative results

being given.

Hybrid: MINECLUS [178] was tested in comparison with PROCLUS and DOC for *d*=5-10, N unknown. *D* was varied from 20 to 80. All 3 algorithms achieved accuracy over 90%. DOC was the most sensitive to *D*.

Kriegel et al. [110] presented a hybrid method (FIRES) which was compared with SUBCLU and CLIQUE. One cluster had *d*=10 but other details are unknown. Precision approached 100% with SUBCLU doing fairly well. Yip et al. [175] developed SSPC as a semi-supervised *k*-medoid approach but it can cluster unsupervised. They compared it with their previous algorithm HARP [174], PROCLUS and CLARANS [142], a non-projected *k*-medoid algorithm. They employed the Adjusted Rand Index [173] an effective assesment of accuracy also adopted in our experiments. Studying *N*=1000, *D*=100, *d*=5-40, CLARANS did poorly. The others achieved close to 100% accuracy when $d \geq 15$ if the best parameter settings were selected. SSPC proved to be the most robust to parameter settings. SSPC had an ARI of about 97% in the absence of noise declining to about 80% with 25% noise. This is further investigated in Section 3.3.4.

Biclustering methods present graphs illustrating their output. Thus it is hard to compare accuracy across papers. Prevalent features in biclusters can be compared, but this does not appear to have been quantified. However, this literature is too extensive to cover properly here.

Correlation clustering methods: Aggarwal and Yu's method ORCLUS [8] with *D*=10, *N*=100,000 and *d*=7 gives good but not perfect accuracy. ORCLUS found non-axis parallel correlations missed by DBSCAN but missed spherical clusters found by DBSCAN illustrating the relative merits of these two approaches. Böhm et al. (4C) [35] show graphically their success with *N*=1000, *D*=10 and *d*=2-30.

CLICK [182] was compared with CACTUS and STIRR on some simple synthetic datasets ($D = 20$), including those which the authors of CACTUS had previously used to compare with STIRR. For example, CLICK detected 2 disjoint clusters as 2 clusters, CACTUS reported all the subsets for the 2 while STIRR failed to separate them at all, though this could be blamed on the initial setting of weights. STIRR did slightly better on overlapping clusters, though the results of CLICK and CACTUS are more readily interpreted.

Overall we see that, as expected, algorithms can achieve close to 100% accuracy on simple synthetic data but the shortage of results on real-world datasets probably reflects the hardness of these problems. Some types of problems are unsuited to unsupervised approaches including some pattern recognition problems, for example when the patterns are rotated unless there is some special type of preprocessing. However, it would be very helpful to both the research and user community if clear comprehensive accuracy results were given on standard clusterable real-world public datasets. Providing quantitative results, which could allow direct comparison without obtaining the code for competing algorithms (which is frequently impossible), may not have been a priority for authors to date. At the same time, clear presentation of the theoretical complexity of the algorithms is also not routinely found making efficiency comparisons moot. Comparing efficiency across papers based on the reports of experiments is unreliable as implementation quality and computer environments are rarely reproducible.

2.3 High-Dimensional Subspace Outlier Detection

Outlier detection has been important in science ever since the beginnings of scientific measurement and the first literature dates back over 200 years. For a long time this was the domain of statisticians but now it is an important part of data mining, machine learning, networks and many other areas in computer science.

There is no established definition of an outlier and different applications may merit different definitions. For example Chen et al. [46] state "Outliers are those data records that do not follow any pattern in an application". The most frequently quoted definition is due to Hawkins [83] who stated, "An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism". Blending both ideas, Ramaswamy et al. [151] state, "An outlier in a set of data is an observation or a point that is considerably dissimilar or inconsistent with the remainder of the data".

Barnett and Lewis [23] define an outlier as "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data". They go on to say that "what characterises the 'outlier' is its impact on the observer (not only will it appear extreme but it will seem, in some sense, surprisingly extreme)." Throughout their book they refer to the outlier as a 'surprising' observation and give examples of how subjective this can be. Statisticians seek to formalise their work so, in almost all published work, the data is modelled by a distribution and the outliers are treated as extreme values of this distribution or belonging to some other coexisting contaminant distribution.

The bulk of the work on outliers in statistics involves univariate data and, often, a single outlier. The approach to multivariate data, and this approach is almost the only one to be found in the literature, is that, given a multivariate observation x, a scalar value $R(x)$ is derived and then some heuristic, statistical or otherwise, is applied to the set of scalars $R_i(x)(i = 1, 2, ...n)$ to discern the outliers. In some cases (e.g [18, 76]), rather than a single scalar value, a function is defined $f(x, t)$ where t is some phase angle allowing a plot where different attributes get different weights as the angle varies. Another refinement is to remove a point x and compute a statistic and thereby accumulate $R(x)$. In any case, having derived $R(x)$, an attempt is usually made to estimate the upper 5% and 1% in order to identify the extreme values. This, of course, involves assuming a distribution for the $R(x)$.

The field of outlier detection is vast and is generally reviewed in the context of a particular application field. The books by Barnett and Lewis [23], Hawkins [83] and Rousseeuw and Leroy [154] provide excellent material on the approaches of the statistics community while Hodge and

Austin [90] have published a review primarily from an Artificial Intelligence viewpoint including the use of supervised and unsupervised neural networks. Novelty detection using neural networks and statistics has also been reviewed by Markou and Singh [130, 131]. Jones and Sielken [98] survey computer system intrusion detection so this is only briefly covered here. Kou et al. [109] have provided a survey of fraud detection techniques. Of particular interest for data mining are the reviews of Petrovskiy [148] and Chandola et al. [43]. Therefore, in this section the related work is restricted to data mining approaches that do not deal with specialised data types (like trajectory or time-series data). In particular, we discuss those that address higher dimensionality as well as those claiming high efficiency.

Computing scientists generally take one of two approaches. Either they follow statistics and attempt to model the data, typically to take advantage of the formal results that one can then derive, or no model is assumed and various heuristics are adopted. Some would distinguish between noise and outliers while others do not. In some situations we may simply be looking for those points that are on the periphery of a distribution. In others, we want to find points that do not fit our current model.

In most approaches, a distance or similarity is defined between the observations. This may include, for instance, nearest neighbour counts. Then a scalar that is related to the local density can be defined and a threshold can also be defined for declaring a point to be in a sparse region. For example, Knorr and Ng [105] state "An object O in a dataset is a $DB(p, D)$-outlier if at least a fraction p of the other objects in the dataset lies greater than distance D from O", while Ramaswamy et al. [151] base their definition on the distance of a point from its k^{th} nearest neighbour. This is used to rank the points and select the top n outliers.

Earlier (Section 1.4.3), we proposed a new definition for outliers in high-dimensionality as follows,"A high-dimensional outlier is an observation that consistently deviates from other observations in multiple subspaces in the global multi-variate space." This definition is in line with and extends earlier definitions.

2.3.1 Methodologies

The prevailing concept of an outlier is based on a notion of points being remote in some sense from the main structure of the data. While this concept is easy to understand in low-dimensional spaces, it becomes questionable as dimensionality increases since inter-point distances tend to equalise [31]. That is, as the number of attributes increases, the ability to distinguish points by their mutual separation declines strongly – every point tends to become an outlier.

An important and often-cited statistical approach to multivariate outlier analysis is the Mahalanobis distance [128], which is a generalisation of the standard univariate approach to multiple attributes and accommodates non-axis parallel distributions. The Mahalanobis distance uses group means for each attribute as well as their covariance matrix. That is, while the normalised univariate distance of a point, value x, from a distribution $\{\mu, \sigma\}$ is $\frac{x-\mu}{\sigma}$, the Mahalanobis distance for a vector $\mathbf{x} = \{x_0, x_1, ...x_n\}^T$ from a set of group means $\mu = \{\mu_0, \mu_1, ...\mu_n\}^T$ with convariance matrix S is

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T S^{-1} (\mathbf{x} - \mu)}$$

Clearly, if the data consists of several well-separated distributions, this can only be applied locally but if the data is in the form of a single distribution or several heavily overlapping distributions, the Mahalanobis distance provides the same formally and intuitively meaningful measure of remoteness as the normalised distance does in the univariate case.

The mean and standard deviation used in statistics to determine outliers are themselves potentially sensitive to outliers. An aberration in the data can significantly bias the mean, which has led to extensive research into making these measures robust usually by detecting and removing outliers. The most famous method for the robustification of the Mahalanobis distance is due to Rousseeuw called the Minimum Covariance Determinant (MCD) [153], which determines the estimator by a subset of points which minimizes the determinant of the variance-covariance matrix over all subsets of the same size. In Section 4.6, we compare various data mining approaches to the robust Mahalanobis distance.

As the literature on outlier detection is vast, we only review that which specifically looks at subspaces or leverages subspace analysis for outlier detection.

Knorr and Ng [107] look for the smallest set of attributes to explain why an outlier is exceptional. They also consider if an outlier is *dominated* by other outliers. Previously [106] they introduced the notion of a $DB(p, D)$ outlier (section 2.3). Based on this, in [107], they define a point x as a non-trivial outlier in space A if it is not an outlier in any subspace $B \subset A$. Further, A is a *strongest* outlying space if there are no outliers in any $B \subset A$. Then all outliers in B are strongest outliers. If A is not a strongest outlying space, x is a weak outlier. The concepts of strong and weak outliers are utilised to provide intensional knowledge about the 'exceptional' nature of the object. They provide two algorithms, one that enumerates the outliers in all spaces starting from the 1D spaces and proceeding to all 2D spaces, all 3D spaces, etc. The second looks for greater efficiency by starting at some kD and whenever an outlier is found in some subspace the proper subsets of that space are inspected. Within a specific attribute space, a nested loop or cell based approach is used for finding the outliers. The output provides intensional knowledge about the outliers from the discovered subspace hierarchy. The main difficulty is in the complexity which is exponential in the number of attributes inspected. This approach seeks to find out why particular points are exceptional but does not seek to rank all or some points, though this may be a possible extension with some further work.

Aggarwal and Yu [9, 10] seek to detect subspaces in which the density is exceptionally lower than average. Each attribute is divided in ϕ ranges each containing an equal number of objects and sparsity coefficients are computed for d'-dimensional cubes. Objects in abnormally low density

Table 2.1: Existing and New Subspace outlier approaches. ‡Described in this thesis.

Citation	Type	Approach
SOE1 [85]	Ensemble	Sum 1D local densities
F.Bagging [114]	Ensemble	LOF in random large subspaces
T*ROF‡, T*ENT‡	Ensemble	From clustering over a range of resolutions
FASTOUT‡	Ensemble	From nearest neighbour clustering scheme
HighDOD [183]	Apriori pruning	Scores based on $\sum KNN$ distances
PODM [172]	Apriori pruning	Scores based on entropy
FPOF [84]	Frequent itemset	Transactions with fewer frequent itemsets
Evolutionary [10]	Local search	For most sparse cells
[21]	Clustering	Unclustered points are outliers

hypercubes are reported as outliers. Since the number of cubes is exponential in the dimensionality and there is no upward or downward closure property to prune the search, the authors propose an evolutionary algorithm to conduct the search. The risk is that the number of local optima may increase exponentially with dimensionality though the authors found results comparable to those found by brute force search. As the method involves local search, it is not intended to score (and hence rank) all points.

Zhang and Wang [183] propose an algorithm called High-Dimension Outlying Subspace Detection (HighDOD). Like the work of Aggarwal and Yu [10], they seek outlying subspaces and enumerate the outliers from there. They exploit the establised heuristic of the sum of the distances to the k nearest neighbours ($KNNSet$) to define an Outlier Degree (OD) such that the OD of a point p in space s is

$$OD_s(p) = \sum_{i=1}^{k} dist(p, p_i) | p_i \in KNNSet(p, s)$$

A point is considered an outlier if its OD is greater than a global threshold T_G which is a constant factor C times the RMS value of the OD values of all the points for all the individual attributes $T_G = C\sqrt{\sum_{i=1}^{d} \overline{OD}_i^2}$ or a local threshold T_L. For downward pruning of subspaces $T_L \leq \frac{1}{C}\overline{OD}_i, \forall i \in s$, for upward superspace selection $T_L \geq C\overline{OD}_i, \forall i \in s$ based on the monotonic property of OD. The novelty lies in their two approaches to pruning the search space, one global and one local. The global requires the OD for a point to be within a range of T_G. To reduce computation, prior probabilities for pruning each subspace size is computed based on a sample. The computation is still exponential with respect to dimensionality but benefits from these improvements. They provide run times on a few UCI datasets and show that the previous approach [10] applied to a synthetic dataset produces results close to that achieved by selecting subspaces at random. In the context of validation they define an outlier strength of a point p as the number of subspaces in which p is an outlier as a percentage of the total. While efficiency of the method is well-studied, no effectiveness results are given.

He et al. [84] describe a method for detecting outliers by computing a Frequent Pattern Outlier Factor (FPOF). This is the sum of the support of all the itemsets in a transaction t divided by the number of frequent patterns in the dataset with more than a minimum support $minisupport$. They also generate a list of itemsets $X \subseteq t$ with the largest values of $||X|| - ||t \cap X|| * support(X)$ as explanatory of the outlier. Simply, the itemsets considered outliers are described by those common itemsets not contained in them. This work is especially interesting because they apply their algorithm and some competing algorithms to two medical datasets from [33]. One (Lymphography, 18 attributes) contains rare classes and the other, the Wisconsin Breast Cancer Data (9 attribute dataset), has a rare class induced in it (8% of the data are 'malignant'). FPOF did best on the WDBC data, finding all the malignant data in the top 14% of the data according to outlier ranking, an accuracy of 57.4%. Both datasets were discretised as the method requires categorical data. This work is arguably a subspace algorithm as the itemsets are sets of attributes.

He et al. [85] attempt to formulate a unified framework for outlier detection in subspaces. The concept is to combine the outlier factors in different subspaces. This framework has as input the target database, the number of desired outliers, the set of subspaces and a combination function amongst others. In addition to formulating this framework which brings together the existing approaches, they present results on a very simple algorithm (SOE1) that only analyses individual attributes and then combines the scores. A histogram is built of each attribute and the value of the segment in which a point falls, a kind of local density, is taken as its measure. These are combined over all attributes. The ranking of these scores is an inverse outlier ranking. The authors used already digitised data, otherwise a parameter(s) would be required for defining the segment widths of non-categorical attributes. They investigate the use of different combination operators for the individual outlier factors. Using $\sum$ or $\prod$ achieved almost equal effectiveness (we therefore adopted $\sum$ in our experiments with SOE1). They also compare with several other algorithms and show, for example, that this method outperforms their earlier work FPOF [84]. Working only on single attributes is very attractive from an efficiency point of view but it is easy to provide examples where this approach might lead to a misclassification (see Section 4.1.4). In our work we show that combining a few attributes in a subspace for analysis does improve the results and this is evidenced in comparison with SOE1.

Breunig et al. [39] pioneered finding density-based outliers. They introduced the local outlier factor (LOF) which measures the degree of outlierness by comparing a point's relative density as compared to those of its nearest neighbours. This allows a ranking of outliers and provides a strongly local view. It can be sensitive to its key parameter *MinPts*. Lazeric and Kumar [114], having found in their previous work [115] that the LOF [39] was the most effective out of many outlier detection methods, but questioning its effectiveness in high-dimensionality, investigated if an ensemble voting approach using results from randomly selected subspaces could improve full dimensional LOF. The approach was to choose a constant number (50 in their experiments) of randomly selected subspaces from dimensionality d of size between $d/2$ and $d-1$ out of d dimensions. They observed that for a small 8D synthetic dataset where all the dimensions were relevant, full-dimensionality LOF performed best. However, on real-world data where a small outlier class was compared with a large normal class, the two voting schemes they propose gave improvements of 4%-14% except in a few cases. They note, without detailed investigation, that the best results came from cases where the outlier class was to be expected to differ in characteristics from the normal class. Where several 'normal' classes were combined and compared with one or several rare classes treated as a group, effectiveness declined. This is entirely to be expected and our work herein sheds more light on these observation. It should also be noted that even though they employ an ensemble approach on subspaces, the subspace size is still $O(d)$. We tested this approach in low dimensional spaces (Section 4.6).

Mao et al. [172] have pursued an approach of the kind found in ENCLUS [47] where an entropy measure was used for apriori pruning of subspace clustering. In this work, an outlier degree is computed based on the number of dense cells and their density within the subspace. As this is not very sensitive for detecting outliers an entropy measure $PO(s) = \sum_{\theta \in s}(\frac{|\theta|}{Np(\theta)}p(\theta) \leq \zeta)$ where θ is the subset of attributes, $Np(\theta)$ is the number of points in a cell, $p(\theta)$ is the density of the cell in the subspace and ζ is a user supplied threshold. Both these measures involve parameters as does the equi-width binning procedure. The algorithm (PODM) was not compared with other algorithms or tested beyond dimensionality of 25, presumably due to the usual scaling efficiency challenges.

Assent et al. [21] propose using a DBSCAN derived approach to subspace clustering and thereby identifying the outliers. Empirical results for outlier detection are not given except that the subspace clustering was superior on certain datasets to SCHISM and SUBCLU.

2.4 Assessing Error

Storey and Tibshirani [159] discuss extensively the problem faced by researchers using microarrays. The essence is that there are typically thousands of genes being tested and a confidence level has to be established for significance for each gene. Traditional p-value cutoffs of 0.01 or 0.05, widely employed in science by convention, lead to a potential

$$FP = pd \tag{2.1}$$

false positives. With dimensionality d large, FP becomes unacceptably large.

The initial means of handling this was to tighten the criteria. The Šidák correction for multiple comparisons [161] tightens the criteria for the probability of a false positive at the feature level (α_c) to yield a desired probability at the experimental level (α_e). Assuming all the features are independent, the correction is

$$\alpha_c = 1 - \sqrt[d]{1 - \alpha_e} \tag{2.2}$$

The Bonferroni correction [1, 36] results from the observation that, for α_c small, using approximations a simpler form of Equation 2.2 can be derived, that is

$$\alpha_c = \frac{\alpha_e}{d} \tag{2.3}$$

The usual application of these corrections is to choose α_c such that $\alpha_e \leq 0.05$ or $FP < 1$ (with reference to Equation 2.1). If d is large, this approach is extremely conservative. That is, α_c is so small to likely cause many false negatives [159]. While false positives are expensive to handle, false negatives can mean the whole work fails. To reduce this, some authors have applied this correction less stringently (e.g. [86, 62, 38]). Lee et al. [118] reported their results at several threshold levels but this makes it difficult to interpret.

Holms proposed a step-down approach that is less conservative [89]. The features are first ranked by their probability of being targets. Then the threshold probability is also varied according to the ranking such that for feature rank k, the threshold probability p is

$$p = \frac{\alpha_e}{d - k + 1} \tag{2.4}$$

This also assumes the features are independent. A similar approach that has been shown to allow for some feature dependencies is the False Discovery Rate (FDR) correction [29, 30].

To improve on this, the q-statistic was proposed [158, 159]. The q value is similar to the p value except that it measures significance in terms of the false discovery rate (FDR) rather than the false positive rate (FPR). The FPR is the rate at which non-target features, sometimes referred to as truly null, are designated significant, i.e. observed as targets. The FDR differs in that it is the rate that features designated significant are not targets. With certain provisos the FDR can be written as the probability $Pr(\text{feature } i \text{ is non-target} \mid \text{feature } i \text{ is significant})$ and the FPR as $Pr(\text{feature } i \text{ is significant} \mid \text{feature } i \text{ is non-target})$. This is a more sophisticated way of tightening the criteria for acceptance of a feature as significant where the number of significant features is much less than the number of non-target ones.

In addition, Westfall and Young proposed a bootstrapping method [164] that does not involve any assumption of independence. The disadvantage to this method is that it is typically expensive to compute and strictly empirical [57].

In the section on GSEP (3.5) we have proposed using standard p-value levels but dividing the samples in two or more groups. This reduces the likelihood of false positives by the power of the number of groups. Working with labelled genome data, this method can yield classification results better than or equal to the best achieved so far.

These approaches per se do not explicitly model the estimated impact of the noise or errors in the data, if known, or, if not known, apply a conventional assumed error rate (e.g. 5%) to estimate the impact on the statistical model, given the experimental parameters such as the number of samples and the number of features (points and dimensions).

In the next chapter it is proposed to estimate, given known or estimated error rates,

1 the expected number of false positives, and

2 the true number of targets given the observed number.

Chapter 3

An Efficient and Effective Approach to High-Dimensional Clustering

In high dimensionality, full-space clustering becomes ineffective as the minimum and maximum distances between points tends to become the same [31]. This motivates subspace approaches. On the other hand, the number of possible subspaces is exponential in the dimensionality making any exhaustive search intractable, so all existing approaches, other than local search methods, depend on an anti-monotonic property with respect to subspace size, such as density, for limiting the search.

In this chapter a novel approach is presented which unusually finds candidate subspaces first. This is done by using two-dimensional (2D) clustering and then looking for clusters in these subspaces using a standard projected clustering approach (viz. [11, 137]). The problem is viewed as an undirected graph with weighted edges and subspaces are found as maximal cliques after parameter-free pruning of edges with low weights indicative of little or no clustering between (pairs of) attributes that act as nodes in the graph.

Several methods [75, 73, 182], all intended for categorical valued data only, have taken a graphical approach using attribute values as nodes. STIRR [75] envisions a dataset D as a simple undirected graph with attribute values as nodes and edges E between attribute values that co-occur in a transaction. In CACTUS [73] the graph is modified by pruning all edges that represent less than some threshold α times the expectation of co-occurrences. Then the cluster-projection on each attribute of up to all attribute pairs containing the attribute (the intersection) is computed. That is, cliques are identified where nodes are single attribute value sets. CLICK [182] uses a somewhat different method to find maximal k-partite cliques in the attribute value graph. As a post-processing, given a parameter $minsup$, it tries to cover the maximal number of tuples with the minimum number of cliques.

All of these require categorical (or low-nomial data) and are quite different from the graphical approach proposed in this thesis. Out approach uses the attribute itself as a node (as opposed to attribute values), can handle any kind of data including real-valued data, and has an entirely different construct for edges. Rather than simply representing co-occurrence of attribute values, in our approach, edges represent a clustering relationship between attributes as explained in more detail below.

The advantage of 2D clustering is that it is a well-studied area and very effective methods exist (e.g. [60, 66]). Results are easy to check visually and no ‘curse’ of dimensionality problem exists [25]. Enumerating all 2D pairs is generally tractable and for very large dimensionality, as in the data used in Section 3.5, this method can be used after feature reduction. In the results presented here, this is provided by an algorithm GSEP (Section 3.5).

The approach presented here is both a framework, which we term MAXCLUS for Maximal clique Clustering, which prescribes a series of stages in each of which any specific appropriate algorithm can be inserted, and a very efficient and simple implementation GCLUS (Gene Clustering), the name inspired by the application to genome assays described in Section 3.5. First the MAXCLUS framework is itemised and then the details of GCLUS given.

3.1 The MAXCLUS Framework

As a framework, the stages of MAXCLUS are as follows:

1 Preprocessing such as normalisation.

2 Detection of dense areas in individual attributes and, optionally, removal of redundant dimensions.

3 Pairwise clustering of all remaining attribute pairs and collection of clustering statistics and their expression as a Clustering Relationship Graph.

4 Pruning of the graph edges with little evidence of clustering and any other pruning heuristics appropriate to the dataset.

5 Clique enumeration giving candidate subspaces.

6 Extraction of the clusters in the candidate subspaces based on the pairwise clustering information.

Each stage requires a particular approach. For example, the intial pairwise clustering may be best done using different algorithms for different types of attributes. Clustering binary attributes is quite a different task from clustering real values attributes.

Thus, in this thesis, we are primarily presenting a general approach to subspace discovery or feature selection and provide an implementation using a projected clustering algorithm to illustrate the effectiveness and efficiency of this approach to subspace cluster detection. Before discussing the implementation GCLUS, an illustration of the flexibility of the framework is noted.

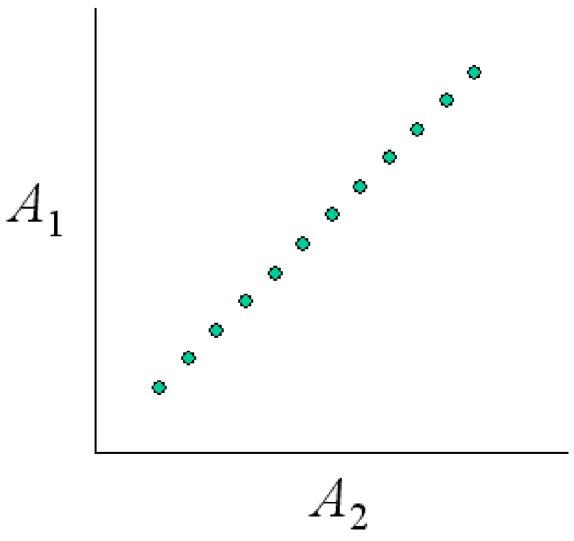

Figure 3.1: A dataset which has two linearly correlated attributes, A_1 and A_2, and no dense areas with respect to A_1 or A_2 taken singly.

3.1.1 Axis-Parallel vs Non-Axis Parallel Clustering

Projected clustering is also sometimes referred to as axis-parallel clustering. Non-axis parallel clustering methods (e.g. [8]) have been developed by authors pointing to circumstances such as that illustrated in Figure 3.1. In this figure, one can see that the clear relationship between attributes A_1 and A_2, by way of a linear correlation, could not be inferred by simply inspecting the individual attributes A_1 and A_2. Thus, the downward closure property could be said to be inapplicable.

MAXCLUS can be taken as a framework in which any clustering method can be inserted in the 2D clustering phase. This includes linear or non-linear regression to handle issues such as that presented in Figure 3.1 on attribute pairs that have a strong correlation. The slope of the regression can also be different between each pair of attributes thus providing similar benefits as the state-of-the-art non-axis parallel algorithms.

3.2 GCLUS

3.2.1 Definitions

Definition 1. *Projected Clustering*

Given a *dataset* $D \subset \mathbb{R}^d$ in d dimensions where each dimension represents a different *attribute* of the attribute set A: For every point $x \in D$ and every $i \in \{1, \dots, d\}$ we denote by x_i the value in the i^{th} attribute of x, i.e., $x = (x_1, \dots, x_d)$. If $k \leq d$ and $S = \{j_1, \dots, j_k\} \subseteq \{1, \dots, d\}$, then D_S denotes the set $\{(x_{j_1}, \dots, x_{j_k}) \mid x \in D\}$, i.e., the projection of D on the attributes indexed by S. Without loss of generality, we assume all attributes are normalised to $[0, 1]$.

A projected clustering result on S is the set of points and attributes $P_S \subseteq D_S$ such that the

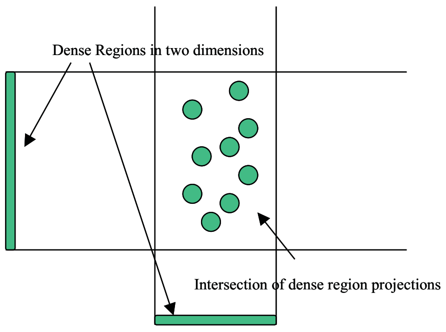
Dense Regions in two dimensions
Intersection of dense region projections

Figure 3.2: Density of intersecting areas of dense region projections is much greater than that of non-intersecting areas indicating a clustering relationship.

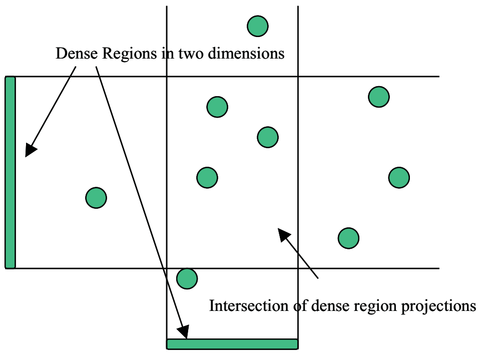
Dense Regions in two dimensions
Intersection of dense region projections

Figure 3.3: Density of intersecting areas of dense region projections is similar to non-intersecting areas indicating no clustering relationship.

minimal bounding hypershape H_P that encloses P_S has a density ρ_{H_P} which is significantly greater than would be expected by chance at the level α (See Sections 3.2.3 and 3.2.7). H_P is composed of the hyper rectangles created by the projection of P_S on S and has volume $vol(H_P)$.

The clustering result could be output as a set of clusters $C = \{S', P'\}$ such that $P' \subseteq D$ is a set of points that exist within a set of ranges S' over attributes $S \subseteq A$. Intuitively, these areas S' of the attributes S are projected as suggested by Figure 3.2 to form the hypershape that encloses C and contains P'.

Definition 2. *Clustering Relationship Graph*

The clustering relationship graph (CRG) for D is the complete graph $G = \{V, E\}$ where $V = \{1, \ldots, d\}$ and $E_{i,j} = \{v_i, v_j\},\ v_i, v_j \in V,\ i \neq j,\ 1 \leq i, j \leq d$. Let w_{ij} be the weight on edge E_{ij} such that for the projected clustering result P_{ij} on attribute set $\{i, j\}$ containing p points

$$w_{ij} = \frac{p}{|D|}$$

This simple definition was adopted as it proved as effective in our experiments as more complex definitions such as $w_{ij} = \rho_{H_P} = \frac{p}{vol(H_P)|D|}$.

Figure 3.2 illustrates a clustering relationship while Figure 3.3 is an example of two attributes with no mutual clustering and hence no such relationship. These scenarios result in high and low edge weights respectively.

Definition 3. *CRG Partition*

For the set of sorted edge weights $W = \{w_0, \ldots, w_m\},\ m = \binom{d}{2}$ let the smoothed series be W^s where $w_i^s = \frac{\sum_{i-1}^{i+1} w_i}{3}$. Then the threshold $T = w_{j-1}^s$ where $j = i$ for $max(w_i^s - w_{i-1}^s),\ 2 \leq i \leq m - 1$. That is T is just below the largest value jump in the sorted and smoothed series. Then the partitioned CRG $G' = \{V', E'\}$ where $\forall E'_{ij}, w_{ij} > T$. This approach is motivated in Section 3.2.2.

Definition 4. *Candidate Subspaces*

The set of candidate subspaces $S = \{S_1, S_2, \ldots\},\ S_i \in V',\ 1 \leq i \leq |S|$ is the set of all maximal cliques of size greater than $q = 2$ in the partitioned graph. These maximal subspaces are referred to as Strong Subspaces.

Definition 5. *Strong Subspace Cluster.*

A Strong Subspace Cluster $C_S = \{p_a, p_b, \ldots\},\ p_i \in P_S$ is a cluster in strong subspace S.

3.2.2 The Algorithmic Idea

If a 2D clustering is performed using the attribute pair $\{A_i, A_j\}$, then an estimate of the relationship between the pair as it contributes to the overall clustering can be made. We refer to this as the Clustering Relationship statistic w. If all $\binom{d}{2}$ pairs are clustered, a graph can be constructed where

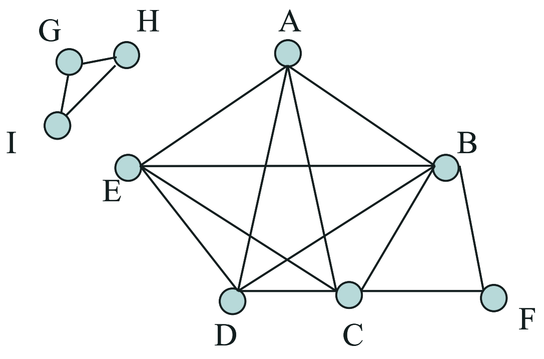

Figure 3.4: Clustering Relationship Graph

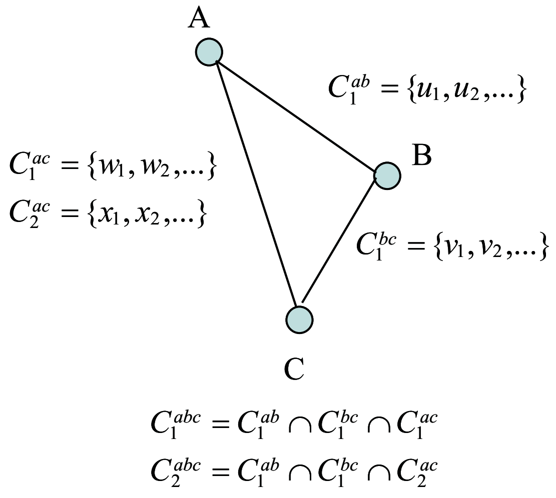
A
$C_1^{ab} = \{u_1, u_2, ...\}$
B
$C_1^{ac} = \{w_1, w_2, ...\}$
$C_2^{ac} = \{x_1, x_2, ...\}$
$C_1^{bc} = \{v_1, v_2, ...\}$
C
$C_1^{abc} = C_1^{ab} \cap C_1^{bc} \cap C_1^{ac}$
$C_2^{abc} = C_1^{ab} \cap C_1^{bc} \cap C_2^{ac}$

Figure 3.5: Clustering from a Clustering Relationship Graph

the attributes form the nodes and the edges are weighted by w_{ij} for each pair of nodes $\{A_i, A_j\}$. This is termed the Clustering Relationship Graph (CRG). This graph can be partitioned to remove unimportant edges (low weights) and then the maximal cliques in the graph will define the important maximal, non-redundant but potentially overlapping, candidate subspaces.

Figure 3.4 is an example where the edges with weights below some threshold have been removed. This has left a 5-clique overlapping with a 3-clique and a disjoint 3-clique.

These subspaces are only candidates at this stage as they may not contain any clusters large enough to be of interest to the end-user. Finally, these subspaces can be clustered. In our implementation, the information retained from the intitial 2D clustering is exploited to provide the final clusters with minimal further cost. On each 2D clustering, the clusters are labelled $\{1, 2, \dots\}$ and then, in the final stage, all points with the same labels on all edges of the nodal clique are clustered. For example, if values for a gene represent two states, up and down regulation, then the samples can have 4 possible states in the mutual clustering of 2 genes: up+up, up+down, down+up, down+down, which could be labelled 11,12,21,22. Thus the final clusters represent samples with the same regulation on all the genes in the clique.

Each edge E_{ij} of the CRG $G = \{A, E\}$ is associated with a clustering result $\mathscr{C}^{ij}$, as well as the scalar weight w_{ij}. For an edge E_{ij} on the attribute pair $\{A_i, A_j\}$ on which k clusters were found, the clustering result $\mathscr{C}^{ij} = \{C_1^{ij}, C_2^{ij}, ..., C_k^{ij}\}$. For example, see Figure 3.5. As illustrated in this figure, the clustering results on the edges of a (usually maximal) clique size $|S|$ in the CRG can be intersected to provide $\mathscr{C}^S = \{C_1^S, C_2^S, ...\}$ which is the final strong subspace clustering we seek (in the figure the output clusters are $\{C_1^{abc}, C_2^{abc}\}$).

As the figure indicates, when we collect a cluster in a $|S|$ size subspace we utilise at most one cluster C^{ij} per edge E_{ij}. Thus, if there are multiple clusters on multiple edges, the number of clusters collectable on the final subspace can be quite large. If one only required to know which points cluster within a given subspace, then a union of the clustering result could be output.

A projection clustering $\{S', P'\}$ is generally achieved by intersecting the ranges S' to collect the points P'. This is done for all the subspaces of interest, potentially the entire exponential sized space. It is straightforward to show that the clusters collected using only attribute pair clustering are equivalent. For completeness, this proof is now given.

Suppose there is a dataset D containing n points p_i, etc. and a cluster in a subspace s containing attributes a, b, etc. That is $s = \{a, b, c, ...\}$. On each (non-noise) attribute in s at least one dense area or cluster C is found containing points p_j, etc., e.g.

$$C^a = \{p_i^a, p_{i'}^a, ...\} \mid 1 \leq i, i', ... \leq n, \tag{3.1}$$

$$C^b = \{p_j^b, p_{j'}^b, ...\} \mid 1 \leq j, j', ... \leq n, \text{ etc.} \tag{3.2}$$

Then the output cluster C^s of the projected clustering method is given by

$$C^s = C^a \cap C^b \cap C^c \cap ... \tag{3.3}$$

Now, the pairwise clustering process first finds a series of two-dimensional clusters. E.g.

$$C^{ab} = \{p_j^{ab}, p_{j'}^{ab}, ...\} \mid 1 \leq i, i', ... \leq n, \tag{3.4}$$

$$C^{ac} = \{p_j^{ac}, p_{j'}^{ac}, ...\} \mid 1 \leq j, j', ... \leq n, \text{ etc.} \tag{3.5}$$

where, assuming a projection clustering process has been used in the attribute pair spaces,

$$C^{ab} = C^a \cap C^b, \text{ and} \tag{3.6}$$

$$C^{ac} = C^a \cap C^c, \text{ etc.} \tag{3.7}$$

$$\therefore C^{abc} = C^{ab} \cap C^{ac} \cap C^{bc} \tag{3.8}$$

$$\therefore C^{abc} = C^a \cap C^b \cap C^a \cap C^c \cap C^b \cap C^c \tag{3.9}$$

$$\therefore C^{abc} = C^a \cap C^b \cap C^c \tag{3.10}$$

$$\therefore C^s = C^a \cap C^b \cap C^c \cap ... \tag{3.11}$$

which is the same output as for the full-dimensional projected clustering process. As we will show, the Clustering Relationship Graph is used to determine which edge clustering results to intersect, saving the exhaustive search over all possible subspaces.

3.2.3 1D Clustering

We choose to base our whole scheme on 1D clustering, that is the dense areas detected on each dimension. This is the approach taken by projection clustering algorithms such as CLIQUE [11] and MAFIA [77]. We made that choice because it is very fast, and easy to interpret. However, separating the underlying distributions on each dimension is critical for success with the final clustering. CLIQUE combines adjacent dense bins based on a simple threshold. This fails to discriminate when distributions significantly overlap as they almost always do in the real world. MAFIA has a parameter to only merge bins of a similar size. This could cut the tails of sharply peaked distributions. Others [137] adopt a kernel method such as EM to model the data based on, say, k Gaussians. We adopted a method that does not require knowledge of k and isolates peaks based on detection of intervening troughs. If we know k this is straightforward but otherwise a heuristic is needed to determine when a trough is real, rather than due to noise.

To determine dense areas in a dimension we use binning. Bins B are equiwidth segments of a dimension such that the mean bin size $m = N/|B|$. $|B| = O(N)$ so m is a constant. We scan across the bins and mark the start of the first dense area when a bin frequency exceeds the mean m. The end of the dense area is detected based on

a) the bin frequency dropping below the mean or, optionally,

b) a local minimum being detected.

In case 'b' a new dense area would start here. If the data is very noisy then 'b' will cause many spurious groupings but if the noise is slight, then this option facilitates the separation of overlapping distributions. Continuous areas flagged as dense are expanded by a factor β (see Section 3.2.5). In the results reported in Section 3.3, the optional heuristic 'b' was only used for the Glass data.

It is straightforward to choose between (a) and (b). If we use (b) and find that there are a large number of small clusters in the subspaces, then (a) may be preferred as it indicates that stochastic effects have likely caused spurious minima and thus many dense regions to be detected on one or more attributes in the clique. Of course, the heuristics can be improved by many means such as smoothing, etc. but that is not the focus of this work and the simple heuristics adopted have produced good results.

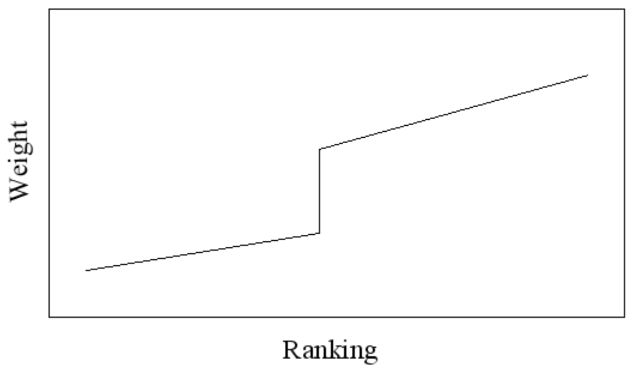

Figure 3.6: Sorted CRG edge weights on ideal (synthetic) data.

In Definition 1 it was stated that areas with statistically significantly greater density are to be selected. In the GCLUS implementation of MAXCLUS, the following approach has been taken: Let the the mean and standard deviation of the bin totals be $\{\mu_b, \sigma_b\}$. A dense region consists of k consecutive bins, with bin totals $\{n_1, n_2, \ldots, n_k\}$, that meet the criteria (a or b). Then a total bt is computed

$$bt = \frac{1}{\alpha\sigma_b}\sum_{i=1}^{k}(n_i - \mu_b) \tag{3.12}$$

Here α is equivalent to a statistical threshold and a region is considered dense for $bt > 1$. Dense regions are labelled $\{1, 2, \ldots\}$ as they are discovered scanning across the attribute. Suppose an attribute has values such that a high value represents an 'up' regulated gene and a low value a 'down' regulated gene or equivalent, then, if the gene is important for the diagnosis, groupings will appear in one or both areas giving rise to regions labelled '1' or '1' and '2'. The projection onto these regions is employed in the clustering, as usual for projection clustering, and the clusters found derive their labels from the labels on the attributes involved (for genome application examples see Section 3.5.2).

After scanning across an attribute, if we find no dense areas, then the attribute can be optionally removed.

It should be noted that if an attribute has a uniform value distribution, but its values have clear boundaries and thus sharp distinctions between classes, these classes will be detected.

3.2.4 CRG Partitioning

An important step is determining the threshold for removal of edges from the CRG. If a synthetic dataset is created containing only noise and clusters that only exist in certain subspaces $S = \{i, j, \ldots\}, S \in V,\ |S| < |V|$, then a clear distinction can be seen for the values of $w_{pq},\ p, q \notin S$, which have low values due to a lack of clustering and $w_{ij},\ i, j \in S$, which have high values. Thus the sorted weight set will look something like Figure 3.6. Even in highly noisy real data this pattern will exist to some extent. Finding the principal step in the ranked weights typically yields a useful solution and the user can choose to step the threshold up or down the ranking to reduce or increase the size of the output candidate subspace set. To reduce stochastic effects, a moving average is first applied to the weights before bisection.

3.2.5 False Positives and Negatives

A false positive (FP) is an item identified as a cluster member which does not properly belong there. A false negative (FN) is a cluster member that is not identified as such. While there is considerable work on false positives and negatives in the statistics community, the implications in subspace clustering have hardly been discussed. Everyone is aware of the 'curse' of dimensionality already mentioned. However, this also means that the number of true positives detected drops off sharply with increasing subspace size.

For what we believe is the first time in this area, we are suggesting a simple strategy to deal with this, namely to expand the collection area around the detected dense areas. In the results section, we show empirically the effect of this on the controlled situation of a synthetic dataset (Table 3.2). This table shows how the accuracy for a 100K dataset increases markedly by employing this strategy. To evaluate this strategy theoretically we assess the effect of this on both false positives and negatives as follows:

We know that for a fixed number N of randomly distributed noise points in database DB, the number of such points in any strong subspace cluster C in strong subspace S decreases exponentially with $s = |S|$. Let us consider an example to show what the effect of increasing sparsity has for both FP and FN. Projection clustering, for example, involves defining a dense region along one attribute and considering that as a projection of a cluster built from the intersection of similar dense areas of other attributes. We refer to this dense area as the 'collection area'. Since clusters tend to have a mean and a tailed distribution, this collection area will cut the tails at a certain range from the mean. Suppose we have fixed our collection area around a single normal distribution cluster peak at $\pm 1\sigma$ (standard deviation) of the distribution about the mean. This will give a 68% collection rate. If we double the collection area on each attribute to $\pm 2\sigma$, then the collection rate on each dimension increases to 95% (Chebyshev theorem). That means a false negative rate of only 5%. Suppose the true positive rate with respect to a single attribute is r. Then in a subspace of size s it is r^s. If, for example, $r = 0.68$ ($\pm 1\sigma$), and $s = 4$ (4D), we will collect only 21.4% of the distribution (cluster) members. If $r = 0.95$ ($\pm 2\sigma$), the collection rate is 81.5%.

This shows how the false negative rate can be sharply reduced by expanding the collection area but what is the effect on false positives? By doubling the collection area, the volume of the subspace area has increased by 2^s which means the potential number of false positives (FP) increases by a factor of 2^s. For example, suppose we collect over $\pm 2\sigma$ and $s = 4$ (4D) and 50% of the points in the dataset are random noise the rest being in this single cluster. Further let the dataset have a value range on each attribute of $\pm 4\sigma$. Then, the cost of decreasing false negatives from 88.6% to 19.5%

is an increase in false positives from just 0.4% to 6.25%. In 5D, the FP rate goes from 0.01% to just 3.1%. Thus the strong dilution of noise with increasing dimensionality typically makes it very advantageous to expand the collection area beyond the detected regions around the peaks.

The expansion factor employed in GCLUS is a scalar value β.

Algorithm 1 MaximalCliqueEnumeration(G)

```
Input: $G = \{P, E\}$, Minimum clique size $minClqSize$
Output: Set of maximal cliques $C$, clique index and count $k$

$order \leftarrow BuildOrderTable()$ {Compute order of each node}
$C \leftarrow \emptyset$
$k \leftarrow 0$
for all $i \in P$ do
  if $(order[i] \geq minClqSize)$ then
    if $(connected \leftarrow AddIfConnected(i, C_k))$ then
      $C_k.minimumOrder \leftarrow order[i]$
      $startFrom \leftarrow i$
      while $((FindConnectedCliques(startFrom, k)) > 0)$
    end if
    if $(connected$ and $order[i] \geq minClqSize)$ then
      $i \leftarrow i - 1$ {$i$ may exist in another clique so use it in the next round}
    end if
  end if
end for
return $C$
```

Algorithm 2 $AddIfConnected(i, C_k)$

```
Input: point $i$ and current clique $C_k$
Output: whether $i$ has been added

if $(C_k = \emptyset$ or $Connected(i, C_k))$ then
  $AddTo(i, C_k)$
  return true
else
  return false
end if
```

3.2.6 Maximal Clique Enumeration.

The most readily available code for this problem [40] finds disjoint cliques but in some cases failed to handle intersecting cliques so we have developed an algorithm (Algorithm 1, 2 and 3) that also provides some improved efficiency with respect to memory usage. This algorithm enumerates all maximal cliques, not just disjoint cliques. As it starts with the nodes with the highest order, the largest cliques are enumerated first. A minimum clique size can be specified. MAXCLUS/GCLUS Stage 2 generates results for clique size 1 and Stage 3, for clique size 2, so this minimum was set to clique size 3 in our experiments.

Algorithm 3 FindConnectedCliques($startFrom, k$)

```
Input: a point $startFrom \in P$, index of $C, k$
Output: last member of $C_k$ or null and $k$

FindConnectedCliques(startFrom, k)
$s \leftarrow startFrom$
for all $i \in P \geq s$ do
    if $(order[i] \geq minClqSize$ and $order[i] \geq C_k.order$ and $i \notin C_k$ and $AddIfConnected(i, C_k))$ then
        $C_k.minimumOrder \leftarrow GetLowestOrderIn(C_k))$
        if $(C_k.minimumOrder < Size(C_k))$ then
            {this clique can't be increased, start new clique}
            {collect those that could participate in overlapping cliques}
            $minCount \leftarrow 0$
            for all $j \in C_k$ do
                if $(order[j] = C_k.minimumOrder)$ then
                    $order[j] \leftarrow 0$
                    $minCount \leftarrow minCount + 1$
                end if
            end for
            if $Size(C_k) > minCount$ then
                for all $j \in C_k$ do
                    if $(order[j] > 0)$ then
                        $order[j] \leftarrow order[j] - minCount$
                        if $(order[j] > 1)$ then
                            $AddTo(j, C_{k+1})$
                            $Update(C_k.minimumOrder)$
                        end if
                    end if
                end for
                $k \leftarrow k + 1$ {Go to next clique}
                if $(Size(C_k) > 0)$ then
                    $startfrom \leftarrow FirstPointIn(C_k)$
                    return true
                else
                    return false
                end if
            end if
        end if
        if $((FindConnectedCliques(i, k)) == 0)$ then
            return false
        end if
    end if
end for
if $(Size(C_k) > minClqSize)$ then
    if $IsMaximal(C_k)$ then
        $k \leftarrow k + 1$ {Go to next clique}
    else
        $DropLastMember(C_k)$
        $startfrom \leftarrow LastMember(C_k)$
        return true
    end if
else
    if $(Size(C_k) > 1)$ then
        $DropLastMember(C_k)$
        $startfrom \leftarrow LastMember(C_k)$
        return true
    else
        $SetOrderOfFirstMemberTo(C_k, 0)$
    end if
end if
$C_k \leftarrow \emptyset$
return false
```

Table 3.1: The Stages of MAXCLUS/GCLUS and User Options.

Stage	Action	User Involvement
1	Pre-processing	Automated, simple normalization
2	Detection of dense regions in single attributes	Automatic with two modes
3	Pairwise clustering	Automated
4	Pruning the C.R. Graph	Automatic. Option to accept more edges.
5	Maximal Clique Enumeration	Automated
6	Subspace and cluster enumeration	Automated

3.2.7 User Adjustable Parameters

Table 3.1 shows the stages of the algorithm and the options for user intervention provided in our demonstration implementation of the framework. There are no mandatory parameters and just two options for intervention.

In Stage 2, the method described in Section 3.2.3 is employed using a fixed parameter $\alpha = 0.75$ for all experiments. This might appear a rather low value or rather *ad hoc* and thus requires some consideration. There are two principal scenarios for features that are being processed: 1) a feature with one or more distinct peaks A_p, and 2) a purely noise or redundant feature A_n. In the case of A_p, because of the peaks, the regions beyond the peaks have a mean well below μ_b. The low value of α ensures any meaningful peak is detected because it puts only a slight restriction on its size while acting as a substantial barrier to detecting stochastic variations beyond the peak regions. An illustration of this is given in Section 3.5.2 (see Figure 3.10 and Table 3.15).

For A_n, it allows many single bins to register as peaks but the likelihood of a string of k bins declines exponentially with k. Again, two approaches can be taken to this situation. The MAXCLUS framework allows for the removal of redundant features by any chosen method. One could apply another statistical test such as Kolmogorov-Smirnov as employed, for instance, by Moise et al. [135] at some standard probability level or some other redundant feature reduction method as done in Section 3.5 on GSEP. Alternatively, one can retain the features and rely on the low clustering that results from these spurious peaks to be pruned in the later stages of the process. This approach has been taken for the smaller datasets in the Results section. Thus taking any low but non-zero figure for α is conservative rather than *ad hoc* and the process is most unlikely to be sensitive to the exact value as we observed in our experiments.

In Stage 4, a threshold for pruning the Clustering Relationship Graph is automatically determined. However, in real world datasets, it may be difficult to find a clear separating step between 'strongly' clustered and 'weakly' clustered attributes. If the user wishes to detect smaller or larger subspaces, then the computed step can be increased or decreased to include or exclude some edges.

These two options are orthogonal. the first concerns the number of clusters while the second, the size of subspaces. Thus setting them is simple as they are independent.

An important innovation in this work is the expansion of the dense regions found in Stage 2 to

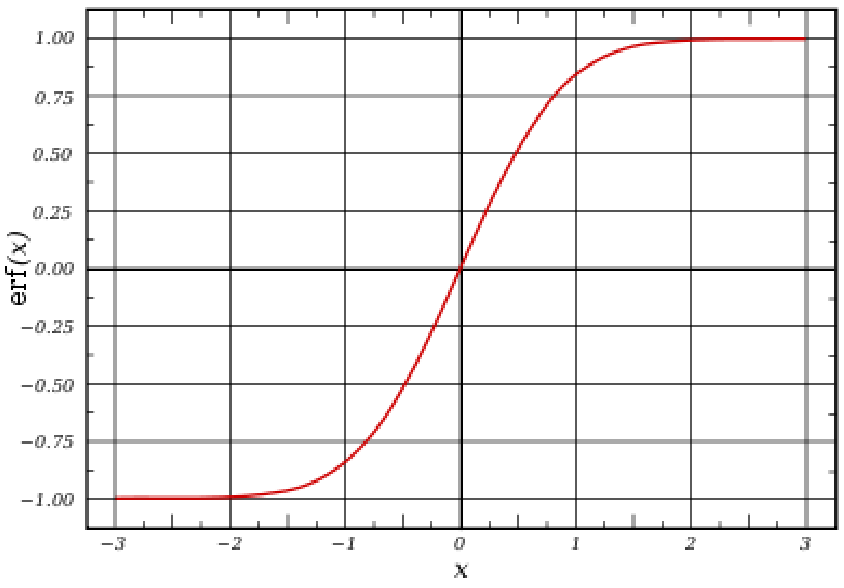

Figure 3.7: The error function for the standard normal distribution from [166].

reduce false negatives. This is controlled by parameter β. In all our experiments, this was set at 0.65 (65% expansion) except when we were comparing $\beta = 0$ to $\beta = 0.65$. This parameter is optional and if set to zero, the algorithm comes on a par with earlier projected clustering algorithms in their use of projections onto dense areas on single attributes. However, our experiments described below indicate that using this expansion does improve effectiveness.

Let us consider if the use of a fixed expansion value is reasonable or whether the expansion should be scaled up with the number of subspaces. If we have identical and independent attributes and a false negative likelihood of $p_1 = 0.05$ on a single attribute, then over k attributes we can reasonable estimate the likelihood as

$$p_k = 1 - (1 - p_1)^k$$

which is approximately kp_1 for p_1 small. Let us assume a standard normal distribution with error function $e(x) = erf(\frac{x-\mu}{\sigma\sqrt{2}})$ for cluster boundary (from μ) x. The tail probability or Q function is

$$Q(x) = 1 - \Phi(x) \tag{3.13}$$
$$Q(x) = \frac{1}{2}(1 - e(x)) \tag{3.14}$$

where $\Phi(x)$ is the cumlative probability distribution function (Equation 4.7). Then if increased dimensionality causes Q to increase by a factor of k, the new boundary x' we require to maintain constant the experimental probability is given by

$$e(x') = \frac{e(x)}{k} + 1 - \frac{1}{k}$$

While there is no closed form solution for this equation, inspection of the graph of $e(x)$ (Figure 3.7) shows that if $x = 1$, the values of k that are covered by our experiments will yield a value of x' approaching 2. This suggest that our choice of expanding by 65% is conservative but reasonable.

3.2.8 Limitations of Projection Clustering

The MAXCLUS framework can be adapted to almost any clustering approach. Our implementation uses projected clustering which is very fast but will not find all possible cluster shapes. For example, if a dense region was completely enclosed in another dense region with little gap between them, it is likely they would not be differentiated. Such clusters within clusters would be found by density based methods such as TURN* [66] or DBSCAN [60]. We have implemented MAXCLUS using TURN* but report this projected clustering method because it proved as effective and more efficient on the real world data tested. Clusters within clusters would be important in image processing but are rather rare in medical and other similar types of applications.

3.2.9 Theoretic Efficiency

In stage 1, the data is read in, normalised and placed into bins in such a way that the bin totals and the ids of the points in each bin can be retrieved. This is linear in run time in the number of points N and number of attributes d. In stage 2, the bin totals are scanned and the dense areas flagged. This is $O(Nd)$. At this stage, the noise only dimensions are optionally discarded.

Stage 3 computes the degree of clustering between all the members of the clustered dimensions CD. Let $a = |CD|$. This involves $\binom{a}{2}$ operations[1]. While this is potentially worst-case $O(d^2)$ the computation for each feature pair is a single pass over the points in the dense bins, typically a small fraction of N for any given dimension, and is thus very fast compared to typical clustering algorithms, which are rarely close to $O(N)$ per clustering and may require repeated iterations. Thus this stage is $O(Nd^2)$.

Stage 4, sorts the edge statistics which for a quick sort is $O(d^2log(d^2))$. The bisection is linear in the number of edges. Stage 5 finds the maximal cliques. If the number of cliques is small or the cliques are relatively small and mainly disjoint, this operation is close to linear in a. In most cases it will not significantly affect the overall efficiency by exceeding $O(d^2)$. Having found the cliques, Stage 6, determining the clusters, is linear or log-linear in the number of points in the dense areas and a. The complexity depends on the sorting technique used. If the number of classes per edge is small, then a linear hash technique is advised. Overall, therefore, assuming the CRG is adequately pruned so that the maximal clique enumeration does not hit a 'hard' graph, the efficiency of the whole algorithm is $O(N(d + d^2log(d^2)))$. If the subspaces are relatively small, then the approach is strongly linear as reflected in our scaleup graphs (Figures 3.8 and 3.9).

[1] $\binom{n}{k} = C(n,k) = \frac{n!}{k!(n-k)!}$

While two algorithms may have similar theoretic efficiency, many passes over the data, repeated iterations or a complex operation on each pass takes much longer than a single low complexity pass and for a very large N, that can be exceedingly significant to the user. The great simplicity of this approach, where no stage takes more than a single pass and the operations are very simple, makes it well suited to very large databases.

3.2.10 Memory Usage

Memory use is $O(Nd)$ but at no stage is there any need to hold full information from more than two attributes in main memory while working with the whole dataset. Clique enumeration only involves $O(d^2)$ space. Final clustering requires space for the clustered points on the clique edges for the subspace S being clustered, which is $O(N|S|^2)$. Typically, if d is large $|S| \ll |CD|$ and memory usage is $O(N)$.

3.3 Experimental Results

The experiments reported in this section used a 3GHz single processor P4 with 1GB of memory. GCLUS was implemented in C++. We tested GCLUS on both real and synthetic data. In the synthetic datasets the clusters were Gaussian within their relevant dimensions and random elsewhere, means (μ) were selected randomly and deviations σ_e were in the range $1 \leq \sigma_e \leq 3$. A proportion of random noise points were added. There were no missing values. Subspace overlap was created by simply including common dimensions in two or more clusters.

We obtained Java class files for a number of classic and recent subspace clustering algorithms. We appreciate the authors of Biosphere and SemBiosphere [176] for this. The algorithms are DIANA [100], CLARANS [142], K-Prototype [93], CAST [28], PROCLUS [7], HARP [174], SSPC [175] and BAHC [177]. DIANA is a divisive hierarchical clustering method while BAHC is a generic agglomerative hierarchical method. K-Prototype is k-means generalized to allow for categorical data when presented. This group of algorithms spans most approaches that have been put forward to date.

3.3.1 Accuracy

Accuracy has two principal components: the accuracy of subspace determination and that of cluster recovery. For the first, we use precision and recall as they are usually defined using true positives (TP) for the correctly identified dimensions, false positives (FP) for the dimensions erroneously identified and false negatives (FN) for the missed dimensions.

$$Precision(P) = \frac{TP}{TP + FP} \tag{3.15}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{3.16}$$

For cluster or point accuracy we use the true positive and false positive rates for the cluster points and the Adjusted Rand Index (ARI) [173]. The ARI takes into account false and true positives and negatives and is thus a sound measure of clustering results on data for which we know the expected result. A perfect score is 1.0 (100%) and no correctness of partitioning receives 0.0 (0%). The ARI is based on the Rand Index [152] and measures how similar the computed partition V is to the actual clustering U. Denote a, b, c and d as the number of point pairs which are in the same cluster in U and V, in the same cluster in U but not in V, in the same cluster in V but not in U and in different clusters in both U and V. Then the ARI is defined as

$$ARI(U, V) = \frac{2(ad - bc)}{(a + b)(b + d) + (a + c)(c + d)} \tag{3.17}$$

Precision, recall and ARI are presented throughout as percentages.

There are alternatives to the ARI. The Silhouette Coefficient [108] estimates the effectiveness of clustering based on the ratio of the intracluster and intercluster dissimilarities. However, this requires that a dissimilarity function has to be defined. Each algorithm uses its own method which is not always fully exposed or described. The Silhouette Coefficient and related clustering validation methods, many of which are biased towards spherical clusters, are best suited to situations where no class labels are provided as opposed to a method like ARI that uses the ground truth provided by the labels. One can use the class labels to compute an entropy as done by Assent et al. [95]. We found this produces results comparable to the ARI approach but is less comprehensive as the entropy does not take into account the proportion of points clustered. Thus one has to report the coverage separately. The ARI is particularly useful as all parts of the result are included in a single accuracy value.

For disjoint clusters that are produced by some mechanism that approximates a normal distribution, the algorithm can produce almost any desired level of point true positives (TPR) for non-overlapping clusters by adjusting the expansion factor β (see Section 3.2.1 and Section 3.2.5). Table 3.2 shows the effect of β on point TP and FP for a 100,000 point dataset. Setting $\beta = 0.65$ produced near 100% TPR at a small cost in speed and the point false positive rate. The last line in Table 3.2 illustrates the effect of collecting less aggressively ($\beta = 0$). The number of noise points that are incorporated in the clusters is extrememly small but the true positive rate drops markedly. In all cases, the point false positive rate is close to zero despite a 20% noise level in the data. As the clusters existed in disjoint subspaces, the attribute values of their points in the rest of the space were randomly distributed thus further contributing to the noise. This illustrates the expected suppression of noise due to the sparsity effect of higher dimensionality as well as the benefit of expanding the detected clusters.

Table 3.2: GCLUS results for 50 dimensions with 2 disjoint 5-dimensional clusters and 20% noise. False positive rates are the highest collected from either the noise or the other cluster. *No dense area expansion, $\beta = 0$. Other results $\beta = 0.65$.

N	Time (secs)	Subspace Precision & Recall	Points: True Positives	Points: False Positives
1,000	0.07	100%	98%	1 point
10,000	0.75	100%	98%	0.4%
100,000	10.7	100%	97%	0.18%
100,000*	9.43	100%	71%	0.04%

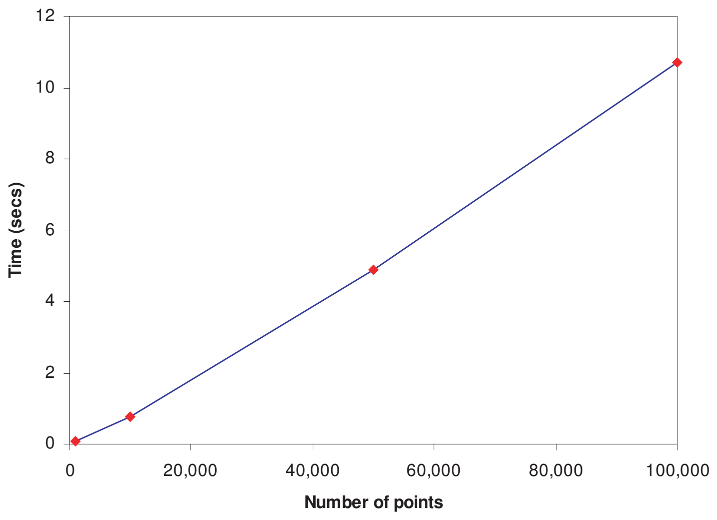

Figure 3.8: Scaling by increasing dataset size N.

3.3.2 Scaling by Data Size

The results for scaling by the data size N verify that the algorithm is approximately linear in N (Table 3.2 and Figure 3.8). The slight non-linearity comes from the final stage of extracting the points from the clustered dimensions. 10,000 points with 50 dimensions, 10 of them containing clusters took under a second. Times included reading in the dataset from disk but not outputting a list of the clustered points to the disk though they do include computing accuracy rates from class labels provided in the test dataset. This was also true for all the algorithms in our comparison experiments.

Some papers report scaling results. The recent state-of-the-art algorithm SSPC [175] showed approximate linear scaling with N but took about 100,000 seconds for a simple data set with 100K points as did PROCLUS in their experiments. 1,000 points took about 250 seconds. We present our results with these algorithms below. Parsons et al. in their review showed MAFIA and FindIt scaling approximately linearly with N with MAFIA completing its task at a speed close to our

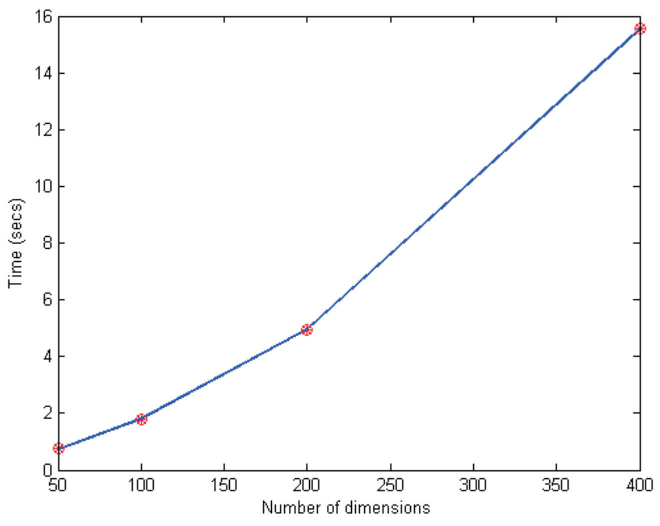

Figure 3.9: Scaling by increasing dimensionality D.

GCLUS results (though their computer specifications are not given) while FindIt was about 10 times slower though the gap narrowed on very large datasets. However, MAFIA is fast due to aggressive pruning and FindIt due to sampling and Parsons et al. found a resulting loss of accuracy for both [145]. In their experiments, both MAFIA and FindIt were not able to identify the subspaces with full accuracy on a simple synthetic dataset very similar to those used in our experiments. The many other algorithms that require model fitting or various kinds of iterative methods may be too costly for large N though useful for certain domains.

3.3.3 Scaling by Dimensionality

Theoretically, we expect the algorithm to scale linearly in overall dimensionality d and by the quadratic of the clustered dimensions CD. Figure 3.9 shows a slight upward curve well below the worst-case theoretical, as we anticipated in our discussion of the theoretical complexity.

3.3.4 Noise

Some authors report noise contamination results. Candillier et al [41] say that their method (SSC) provides 90% purity in the presence of 20% noise and that LAC [18] gives only 76%. They do not give results for different dimensionality but as our results and simple probability theory can show, noise contamination should drop sharply with subspace size and, even at only 10 dimensions, for GCLUS this is effectively 0%. As each cluster is a noise contributor in its non-relevant dimensions for every other cluster covering those dimensions, noise levels of 90+% are often present without interfering with clustering accuracy.

As SSPC is by far the best of the comparison algorithms and can handle noise, we compared SSPC and GCLUS on two synthetic datasets. The first (D1) had 1200 points 3 clusters and no noise.

The second (D2) had 2400 points, the same three clusters and 50% noise. As SSPC does better on disjoint subspaces we used this type of cluster. GCLUS was run once on each dataset and achieved 99.8% ARI on D1 and 99.3% on D2. Subspace precision and recall were 100% on both runs. SSPC, being non-deterministic, was run 10 times and the best result (achieved once out of 10) was 94.3% ARI on D1 and 59.3% on D2. Subspace precision and recall for SSPC were 91.7% on D1 and 86.7% on D2 for these runs. These results show that GCLUS was little affected by noise while SSPC had a clear decline in its cluster identification accuracy.

3.3.5 Real Datasets

One standard source for labelled data is the UCI repository [33]. This is primarily for machine learning and many of the datasets are quite challenging for an unsupervised approach. This may explain why few researchers report unsupervised results for these datasets. One would always expect supervised approaches to be more accurate than unsupervised ones. While the GCLUS algorithm handles mixed binomial, multinomial and real-valued continuous attributes, the best test is on mainly real-valued data. If all attributes are binomial, then the task is equivalent to frequent itemset mining. Relatively few public labelled datasets have all or mainly real-valued data and it is difficult to find any that are very large. Some datasets are not linearly separable or separable without some labelled data. However, certain datasets have been widely used by researchers for testing their algorithms and we present our results here on three of these.

As several of the algorithms are non-deterministic we report the best subspace accuracy and ARI obtained but the average was much lower. K-Prototype, HARP and to a lesser extent SSPC showed quite wide variations in their results though they never outperformed GCLUS. Table 3.6 shows the results on the real-world datasets. As this shows, GCLUS, our straightforward projection clustering implementation of MAXCLUS, outperformed k-means, heirarchichal clustering, a divisive top-down approach and other well-known and effective algorithms.

Unfortunately, we were unable to obtain satisfactory results in a reasonable time, or obtained very poor results, on all of the datasets with CLARANS and CAST so no results for these are reported in the summary Table 3.6.

Iris: The Iris dataset [33] is a commonly used test set with 150 samples made up of three equal classes and four real-valued attributes. This dataset is challenging simply because it is rather small and small datasets are difficult to separate using a binning approach for clustering, which we have adopted. It is also quite noisy. However, small N is important because many datasets (e.g. microarray) have small sample sizes due to the high cost of data collection.

GCLUS applied to the Iris dataset returned two cliques, attributes $\{1, 2, 3\}$ and $\{1, 2, 4\}$. The first gave two clusters and separated type 1 from the other two with an ARI of 98.6% for that task. The second produced three clusters separating all three classes with an ARI of 96.7% (Table 3.3). This is comparable to results obtained by supervised classification, e.g. Jiang and Zhou [97] using

Table 3.3: GCLUS classification accuracy on the Iris dataset (single subspace). Overall ARI 96.70%.

	Type 1	Type 2	Type 3
Total	50	50	50
Correctly classified	39	42	45
Incorrect	1	2	4
Unclassified	10	6	1

Table 3.4: GCLUS classification accuracy on the Glass dataset (single subspace). Overall ARI 65.85%.

	Float	Non-float	Other
Total	87	76	51
Correctly classified	68	34	17
Incorrect	11	2	2
Unclassified	8	40	32

kNN classifiers with a neural network ensemble who achieved around 96% accuracy. On inspection of the data, it is obvious that the 4th attribute gives the best chance of splitting the data into three classes and this can be inferred directly from the GCLUS cluster label output which tells us which attributes discriminated between which clusters. On the comparison algorithms, PROCLUS achieved an ARI of 94.73%, BAHC 81.45%, DIANA 64.46%, HARP 64.34%, K-Prototype (k-means) 63.69%, CAST put all points in one cluster and SSPC could not complete the task. Previously, the recent algorithm STATPC was tested on this dataset [136] (see also Section 2.2.1). An F1 accuracy of 80% was reported. F1 is the mean of precision and recall and the GCLUS F1 accuracy is 89.04%.

Glass: This dataset consists of chemical assays on glass. There are several classes, some rather rare. Principally, we can split the samples into three, float glass, non-float glass and unspecified others. There are 214 instances and 9 continuous attributes. GCLUS output a subspace that separated the three classes with an accuracy (ARI) of 65.9% (Table 3.4). Wang [162] testing a range of kNN classifiers achieved a best result of 66.8% using Neighbourhood Counting and 10-fold cross-validation.

On the comparison algorithms, we were unable to get a result with SSPC. Harp gave an ARI of 15.52% and PROCLUS an average of 22.51% with a best ARI of 54.54%. K-Prototype had a best ARI of 27.16%. STATPC was previously tested on this dataset [136] and yielded an F1 accuracy of 60%. In the same work, an F1 accuracy of 55% was reported for P3C (see Section 2.2.1). The GCLUS F1 result is 68.39%.

Wisconsin Diagnostic Breast Cancer (WDBC): The WDBC database [33] has 569 samples and 30 real-valued attributes. There are two classes, malignant and benign. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics

Table 3.5: GCLUS classification accuracy on the WDBC dataset. Overall ARI 95.22%.

	Normal	Malignant
Total	357	212
Correctly classified	324	149
Incorrect	30	23
Unclassified	3	40

Table 3.6: Accuracy (ARI in %) of various algorithms on real-world datasets. Best results obtained on 10 runs given for non-deterministic algorithms. *Failed to complete, would not run or very low score.

Algorithm	Iris	Glass	WDBC
GCLUS	**96.70**	**65.90**	**95.22**
DIANA	64.46	*	*
K-Prototype	63.69	27.16	49.14
PROCLUS	94.73	54.54	74.34
HARP	64.34	15.52	58.54
SSPC	*	*	70.61
BAHC	81.45	*	*

of the cell nuclei present in the image. GCLUS output a subspace of attributes {1, 2, 4, 5, 6, 7, 8, 17, 21, 22, 23, 24, 25, 26, 27, 28, 29} with an accuracy of 95.22% (Table 3.5). The cluster label indicated that the discrimination was entirely due to attribute 28 as was verified from the Stage 2 1D clustering.

Jiang and Zhou [97] using kNN classifiers and a neural net ensemble and at least 250 points as training data achieved 100% accuracy. With 200 training points, accuracy was 96%. The best of the comparison algorithms, SSPC, achieved a mean ARI of 39.46% with a best result of 70.61% over 10 runs, HARP achieved an ARI of 58.54%, PROCLUS an average of 10.48% with a best ARI of 74.34%. K-Prototype had a best ARI of 49.14%. SSPC typically selected around 25 attributes for each of the clusters.

3.3.6 Applying SERA

The SERA algorithm (Appendix A) can be used to estimate the false positive and negative rates. For example, let us take the GCLUS results on the WDBC (Table 3.5). GCLUS found two groups. The one that was mainly representing the malignant samples contains 172 samples. Without labels, we first ask what is the likelihood of false positives due to noise at the $p = 0.05$ level? SERA shows that this is essentially zero. Thus, the actual false positives are not due to noise. They are, in fact, due to the heavy overlap of the two distributions. In Chapter 4 we investigate how to deal with this problem. Next we ask how many false negatives there are if our collection is at a $p = 0.95$ level? SERA indicates 207, which is rather close to the true value of 212. Even though there are 23

Table 3.7: Subspace clustering algorithms, their mandatory parameters and outlier handling. Non-deterministic algorithms require the number of iterations. 'Opt' means there are optional parameter(s).

Algorithm	Nb. of clusters required	Deter-ministic	Other parameters	Outliers handled	Overlap allowed
GCLUS	No	Yes	Opt.	Yes	Yes
DIANA	Yes	Yes	Opt.	No	No
CLARANS	Yes	No	Yes	No	No
K-Prototype	Yes	No	Opt.	No	No
CAST	No	No	Yes	Yes	No
PROCLUS	Yes	No	Yes	Yes	No
HARP	Yes	Yes	Yes	Yes	No
SSPC	Yes	No	Yes	Yes	Yes
BAHC	Yes	Yes	Opt.	Yes	No

'normal' samples in this cluster, there is a similar number of malignant samples in the other cluster.

If we now look at the predominantly 'normal' cluster. Again SERA indicates that there are likely no noise false positives. If our collection is at a $p = 0.95$ level, the estimated true size of the group is 415. This is too high indicating that for this dominant group, this p value is too low. In fact, since there are only 3 unclassified 'normal' samples, this is equivalent to $p = 0.999$. Here the expansion factor β has essentially done a near perfect job. The difference in the results for the two groups is not surprising. When expanding, the expansion stops when the expansion of two dense areas meet. In such a case, the area between the clusters is apportioned based on their relative size. Depending on the skew of each distribution, the false negatives may be in this area, beyond it or on the opposite tail. Thus, depending on the actual data class distributions, the expansion process can result in a differential degree of success in collection. Therefore, SERA is a useful tool for probing results but can only give estimations based on its inherent assumptions.

3.3.7 Comparing Algorithms

Table 3.7 lists the algorithms we tested and their mandatory parameters (e.g. the final number of clusters, number of rounds of iteration), and whether they handle outliers and overlapping subspaces. Each algorithm may have many more non-mandatory parameters. CAST is the only comparison algorithm not requiring foreknowledge of the final number of clusters but it has other more obscure parameters.

Table 3.8 reports the accuracy and speed results we obtained for varying N on synthetic data. Clusters were Gaussian distributions and noise was randomly distributed. Relevant dataset parameters are in the table captions. Clearly SSPC is the best of the comparison algorithms though it falls short of the accuracy of GCLUS and runs very much slower even when the relative speed of Java and C++ is taken into account. In fact, with execution time running into days, it proved impractical

Table 3.8: Subspace identification accuracy, cluster accuracy (ARI) and speed for various algorithms on a synthetic dataset. Total dimensions d = 100, 4 disjoint clusters involving 24 dimensions, N = 1000 or 10,000 points and 20% random noise. Implementation: GCLUS C++, others Java. *Best result out of 10.

Algorithm	Iterations	Subspace Precision	Subspace Recall	Speed 1K (secs)	Speed 10K (secs)	ARI
GCLUS	N/a	**100%**	**100%**	**0.63**	**3.54**	**99.50%**
DIANA	N/a	N/a	N/a	59	6045	21.20%
CLARANS	10	N/a	N/a	100	956	2.40%
K-Prototype	10	N/a	N/a	99	2044	23.30%
CAST	10	N/a	N/a	417	41136	3.60%
PROCLUS	10	57.6%*	55%*	552	3918	35.20%
HARP	N/a	35%	86.70%	135	9941	22.40%
SSPC	10	93.8%*	87%*	277	2470	80.90%
BAHC	N/a	N/a	N/a	40	3816	7.90%

Table 3.9: Scaling with increased dimensionality for various algorithms. Total dimensions d = 100 (A) and 200 (B), 4 disjoint clusters involving 24 (A) and 48 (B) dimensions, N = 1000 points and 20% noise.

Algorithm	Iterations	Speed for A (secs)	Speed for B (secs)	Scaleup
GCLUS	N/a	**0.63**	**0.81**	**1.29**
DIANA	N/a	59	86.14	1.46
CLARANS	10	100	190	1.90
K-Prototype	10	99	250	2.53
CAST	10	417	929	2.23
PROCLUS	10	552	891	1.61
HARP	N/a	135	553	4.10
SSPC	10	277	430	1.55
BAHC	N/a	40	105	2.63

to test any of these algorithms on 100K points which GCLUS finished in a few seconds. Table 3.9 shows how the various algorithms responded to a doubling of the dimensionality, again GCLUS had the edge followed by DIANA and SSPC.

3.3.8 Overlapping Subspaces and Clusters and ‘Hard’ and ‘Fuzzy’ Clustering

Consider Figure 3.4 which has 3 subspaces, 2 overlapping. We tested this scenario in comparison with the same sized subspaces but where each one is disjoint. The 3-clique which shares 2 dimensions with the 5-clique is very likely to produce some dilution of accuracy as a spurious cluster containing points from the 5-clique can be found by chance in the 3-clique. This is what we observed. In fact, a small heavily overlapping subspace like this is the most difficult case. In both cases GCLUS achieved 100% precision and recall for the subspaces but the ARI for the disjoint case was 99.6% while that for the overlapping case was 91.8% due to the expected small spurious cluster. However, 81% of these points had already been assigned to the 5-clique. If we introduce a weak exclusivity requirement, that is that each point is only allowed one cluster label for overlapping clusters (the largest subspace assigned first), then this spurious cluster disappears (drops below the minimum cluster size threshold) and our full accuracy is restored.

If the overlap is smaller or both subspaces larger, this would of itself suppress the likelihood of a false cluster appearing. Over the course of many experiments we found that GCLUS handles subspace overlaps effectively. Virtually all algorithms to date have employed strong exclusivity - one cluster label assigned per input point. However, there are certainly cases where this is too strong. In a single microarray sample, there may be genes expressed from several different pathways - overlapping gene clusters. Thus providing choice over the exclusivity level is another powerful feature of the GCLUS approach. Exclusivity can be assigned one of three levels. None: points can appear in clusters in an arbitrary number of subspaces. Weak: a point may appear in only one cluster in one of multiple overlapping subspaces, the largest. Strong: strictly one cluster label per point.

Strong exclusivity is normally referred to as ‘hard’ clustering while none or weak exclusivity is described as ‘fuzzy’.

SSPC is the only one of the comparison algorithms that can handle cluster overlaps. On this small but significant problem, SSPC correctly identified the subspaces on 2 out of 10 attempts with a range of ARI values from 41.9% to 89.8%, which it achieved when finding the correct subspaces. SSPC is non-deterministic due to its random seeds.

3.4 Application to a Large High-Dimensional Dataset

So far small datasets were tested which were amenable to the various algorithms being applied. To demonstrate MAXCLUS/GCLUS on larger, high-dimensional datasets it was employed on a dataset from Alberta Perinatal Health. The Province of Alberta maintains a database of records of births.

The data for Central and Northern Alberta was available for this study for the years 1992 to 2003 inclusive. That provided a cohort of 236,718 births without missing data from 92 hospitals and health centers and home births attended by a midwife. The dataset has 239 mostly low-nomial features after exclusion of date and time which were not suitable for this analysis.

The purpose of the experiment was to provide useful output for physicians and so cliques of attributes were sought that included at least one 'input' feature and one 'output' feature. The features were marked up based on advice from physicians, where ambigous. An input feature would be some data collected from the mother prior to admission while an output feature would be, for example, some intervention or complication that arose. As can be seen from Table 3.10, the attribute subsets found are very realistic combinations.

The likelihood of a case participating in every edge of a clique naturally declines with clique size and this table shows just how sharply cluster size by cases dropped off with attribute clique size in this dataset.

3.5 GSEP and GCLUS on Genome and Proteomic Data

In this section we consider the following problem: Given a very high dimensional dataset with a binary labelling of at least some samples, find subsets of features which distinguish the two sample classes.

We will apply GCLUS/MAXCLUS after selecting a smaller set of features which distinguish the two classes. To do this, we developed the novel approach GSEP which acts as a highly efficient and effective classifier and outputs a weighted list of prognosticator features.

Finding the genes responsible for different disease states given experimental and control assays is an intensively studied problem. Many approaches have been applied including clustering and classification. Most classification techniques reduce the intensity values to binomial in a preprocessing step. However, this step inevitably discards a considerable amount of information and can make the output sensitive to the threshold values involved. The data is typically very noisy so this type of approach can be seen as a way of suppressing noise but, at the same time, thresholding is typically noise sensitive.

A simple approach is to compare between groups using standard statistics. A t-test on each gene can indicate whether it separates the groups but if there are many genes, false positives can proliferate. In Section 2.4 the related work on this problem is discussed. Virtually all approaches amount to tightening the criteria for the probability of a false positive at the feature level (α_c) to yield a desired probability at the experimental level (α_e). GSEP provides such a tightening without raising the feature level threshold unreasonably high as in many previous approaches.

Table 3.10: Largest clusters (by cases) detected in the Alberta peri-natal data. Attribute index number in brackets. Recoded refers to an attribute that was adjusted in the preliminary data cleaning by the medical researchers.

Cases	Attribute 1	Attribute 2	Attribute 3	Attribute 4
21657	Gestational age recoded (22)	Anti2less (202)	Birthweight recoded (223)	
10861	Number of Prior C/S (26)	Anaesthesia Other (127)	C/S Primary/Repeat (140)	
10447	Previous cesarean section (49)	Anaesthesia Other (127)	C/S Primary/Repeat (140)	
6154	Gestational age recoded (22)	Membranes ruptured before 37 (61)	Steroids (203)	
6085	Gestational age recoded (22)	NICU Admission (188)	Birthweight recoded (223)	
4226	Pregancy $\geq$41 wks gestation (23)	Ind Pregnancy$\geq$42 wks (103)	Prostaglandin Vaginal (112)	
3407	Gestational age recoded (22)	General (126)	Birthweight recoded (223)	
3399	Number of Prior C/S (26)	Age $\geq$35 yrs (28)	C/S Primary/Repeat (140)	
3344	Less than or equal to 34 week (24)	Steroids (203)	ETT & PPU (179)	
3253	Age $\geq$35 yrs (28)	Previous cesarean section (49)	C/S Primary/Repeat (140)	
3220	Age $\geq$35 yrs (28)	C/S Primary/Repeat (140)	Birthweight recoded (223)	
3099	Gestational age recoded (22)	Ind Pregnancy$\geq$42 wks (103)	ARM (113)	
2414	Multiple Baby (8)	Gestational age recoded (22)	Anaesthesia Other (127)	Birthweight recoded (223)
2396	Multiple Baby (8)	Gestational age recoded (22)	C/S Multiple Pregnancy (157)	Birthweight recoded (223)
2351	Multiple Baby (8)	Gestational age recoded (22)	ETT & PPU (179)	Birthweight recoded (223)
2332	Multiple Baby (8)	Gestational age recoded (22)	Elective Primary (144)	Birthweight recoded (223)

3.5.1 GSEP

Given a dataset D of samples P and d features (viz. assays and genes per assay) we seek a model consisting of d' prognosticator features and their relative weights $w = \{w_0, w_1 \dots w_{d'}\}$. A weight of 1.0 indicates an average classifying or separating effectiveness between experimental and control groups of 1.0 standard deviation (based on the log_2 values).

As this part of the approach is supervised, it requires a labelled training set. The preprocessing applied to the data consists of the standard method of dividing all values on each sample by the sample mean and then taking the log_2 of each value [57]. At this stage, a novel approach is introduced. The training set is divided into four groups, two for the experimental and two for the controls. These groups are selected randomly while ensuring they are approximately balanced in size. That is the experimental group is divided in two approximately equal sized groups $\{E_1, E_2\}$ and so is the control group $\{C_1, C_2\}$. Then for each feature g_i, E_1 is compared to C_1 and E_2 to C_2 and only if the separation $dist(g_i)$ is statistically significant (student's t test) at the threshold level (STL) t_{min} in the same direction for both cases then the feature is used for voting on the test set.

After a feature has passed this test, the two groups are combined giving $\{\mu, \sigma\}$ for experimental and control groups. Then, $\forall s_j \in P,\ 1 \leq j \leq |P|,\ \forall g_i,\ 1 \leq i \leq d$ the normalised distances are (where $s.g.x$ indicates the value of feature g in sample s)

$$dist_{ij}^{E} = \frac{|s_j.g_i.x - \mu_E|}{\sigma_E} \tag{3.18}$$

$$dist_{ij}^{C} = \frac{|s_j.g_i.x - \mu_C|}{\sigma_C} \tag{3.19}$$

$$Dist_{ij} = |dist_{ij}^{E} - dist_{ij}^{C}| \tag{3.20}$$

If g_i is closer (based on Equation 3.18 and 3.19) to the group mean to which s_j belongs, according to its label, then $Dist_{ij}$ is added to the feature weight w_i, otherwise it is subtracted.

The weights depend on a knowledge of the labels and are not used in classifying the test set but are provided as part of the output as a guide to the relative efectiveness of the features in classifying the labelled samples.

The reason for splitting the training set is straightforward: If one tosses an unbiased coin once, the chance of a head is $\frac{1}{2}$. To get two heads in a row, the probability is only $\frac{1}{2^2}$. Thus, with the reasonable assumption of independence for the randomly assigned groups, the likelihood of a false positive is reduced by the squareroot. Therefore, we could have a statistical threshold of just

$$\alpha_c = \frac{\alpha_e}{\sqrt{d}} \tag{3.21}$$

to achieve the 'not more than one' false positive level and higher STLs can be used up to the Bonferroni correction $\frac{\alpha_e}{d}$ to even more strongly suppress false positives and reduce the ouput model. This is discussed further in Section 3.5.1. If the number of samples is sufficient, then the training set could be divided into k sets and Equation 3.21 could employ the k^{th} root but medical data usually has insufficient samples due to the cost of collection so we confine ourselves to investigating two-way splits. For a reasonable approximation to normal distributions to meet the criteria for the statistical test, around 30 samples are ideally required in each comparison. However, since our purpose is not to achieve statistical exactness, this criteria need not prevent the use of this approach unless the number of samples is very low.

Bad Vote Limit Filter

If there are misclassifying examples in the training set, as in the Colon Tumour data discussed below, then depending on the random assignment of those examples to the training groups, a feature may decline in separating effectiveness. A filter was added to control this, the Bad Vote Limit Filter (BVLF). Quite simply, after computing the means and standard deviations of the groups for the training data, a check is made for how often these values would lead to a bad vote on the training data. That is, for each feature $g_i, 1 \leq i \leq d$ and each sample $s_j \in P, 1 \leq j \leq |P|$ there is a vote v_B such that

$$v_B = \begin{cases} 0 & dist(g_i) \Rightarrow \text{label} \\ 1 & \text{otherwise} \end{cases}$$

and

$$Bad(g_i) = \frac{1}{|P|} \sum_{j=1}^{|P|} v_B$$

If the proportion of bad votes $Bad(.)$ exceeds some threshold, the Bad Vote Limit Filter, then that gene is skipped. This is a control on the effect of outliers on the gene means. While the voting scheme $w_{g_i} \in \Re$ could be sensitive to outliers, the BVLF being an integer voting scheme controls this.

If we have just two classes, then this parameter can be set automatically by choosing a value that is close to but not more than that which produces the highest accuracy on the training set. This is justified because if the BVLF is too low, it will not have any effect on the results. If it is too high it will eliminate genes that usefully discriminate by being too restrictive on the size of the model.

This filter tends to control the more marginal features. Thus, whether one correctly classifies 88% or 90% of a test set, the model output of the leading features will likely be identical. Thus this filter tends to improve classification accuracy without substantially affecting the model.

Validation

For evaluation, we adopted Multifold Cross-Validation. Since the number of assays available is usually small and larger samples more precisely support statistical assumptions of normality while reducing stochastic effects, we used the standard hold-one-out method of cross-validation. This maximizes the size of the training set. One assay is set aside as the test and the rest are used as the training set. This is repeated over all the N assays. The effectiveness is the proportion of test assays correctly identified. In most cases, this was the evaluation method adopted by other researchers on the datasets tested.

Output Weighting

Output weights are generated to evaluate the separating effectiveness of each feature. The separation achieved on the training data is computed and added or subtracted to the weight for each test item depending on whether it was correctly assigned. Finally the total is divided by the size of the test set. These weights are not used in the prediction method but can be very useful to the user in interpreting the results. Firstly they allow sorting of the features by effectiveness and then they also reveal which features make very little contribution. The size of the output list is sensitive to the statistical test threshold but, whatever the size, the weights can be used to readily distinguish the key features. The major contributors remain the same whatever the length of the list reducing any concern about false negatives.

Parameter Sensitivity

Compared with the Bonferroni threshold of $\frac{\alpha_e}{d}$ (Equation 2.3, Section 2.4), a minimum t-value threshold (STL) of $\alpha_c = \frac{\alpha_e}{\sqrt{d}}$ can be used. Since Bonferroni is generally considered too conservative (see Section 2.4), this is advantageous. If it is desired to vary the model size, the STL can be varied from that of Equation 2.3 to that of Equation 3.21. We observed that the effectiveness may change only slightly over a considerable range, depending on the application, until it starts to drop off more sharply as the model rapidly increases.

The other parameter is the Bad Vote Filter Limit (BVFL). This is an optional extra that can slightly increase effectiveness when misclassifying examples exist in the training data. The BVFL needs to be set 10-15% below the target effectiveness or it will start to degrade the model. As discussed in Section 3.5.1, this can be set automatically based on the results on the training data.

Complexity

The algorithm performs a single pass over the samples for each feature and thus is linear in $|P|$ and d. It also requires only $O(|P|d)$ space. The BVLF inserts a loop over $|P|$, but this is only run on those features identified as prognosticator features and this is typically $O(1)$. For example, in the colon tumour set, between 11 and 100 genes were identified out of 2000, depending on the STL, with the 11 being the key genes whatever the model size. Biologically, we do expect that only an $O(1)$ proportion of genes tested will be significant contributors to a particular pathology, and this real-world reality is a key reason this problem is very fast to solve.

3.5.2 Results

Unless stated, values are subject to log_2 normalisation.

Table 3.11: Prognosticator genes, $n = 11$, accuracy 91.94% (57/62), Bad Vote Limit Filter 80%.

Gene	Weight	Description
R87126	1.42	MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)
R36977	1.262	P03001 TRANSCRIPTION FACTOR IIIA ;.
M36634	1.225	Human vasoactive intestinal peptide (VIP) mRNA, complete cds.
M22382	0.829	MITOCHONDRIAL MATRIX PROTEIN P1 PRECURSOR (HUMAN);.
H08393	0.812	COLLAGEN ALPHA 2(XI) CHAIN (Homo sapiens)
H43887	0.709	COMPLEMENT FACTOR D PRECURSOR (Homo sapiens)
M63391	0.703	Human desmin gene, complete cds.
T86473	0.394	NUCLEOSIDE DIPHOSPHATE KINASE A (HUMAN);.
X63629	0.306	H.sapiens mRNA for p cadherin.
T92451	0.179	TROPOMYOSIN, FIBROBLAST AND EPITHELIAL MUSCLE-TYPE (HUMAN);.
M76378	0.028	Human cysteine-rich protein (CRP) gene, exons 5 and 6.

Colon Tumour Dataset

This dataset [16] has 62 Affymetrix oligonucleotide arrays from biopsies of colon tumours, 40 are labeled positive and 22 are labeled negative. Values for 2000 genes are provided for each sample. Using multifold cross-validation, GSEP obtained an accuracy of 91.94% using an STL equivalent to $t_{min} > 3.4$ and BVLF of 0.80 and 11 out of the 2,000 attributes. This exceeds the highest accuracy we have seen reported [17, 26, 71, 163, 92] except for [129] (see next section). However, whether it identified 55, 56 or 57 samples correctly did depend on the random seed used for forming the training subsets since a few samples appear to misclassify and the effectiveness of the run varied depending on which training subset these were placed in.

Classification

About 10 genes gave a slight effect in the opposite direction but never as much as a weight of -0.1. This is likely a consequence of the few misclassifying samples. Genes that contribute less than 0.1 weight for or against are thus discarded for the final clustering stage. Since it took 11 genes to achieve the high accuracy reported, any smaller subspaces detected are most unlikely to cover all samples. Thus we cannot simply assess their accuracy by sensitivity or coverage as well as selectivity or separation of classes. However, genes which cluster together may share some functionality and may prove of value for diagnosis.

Huang and Kecman [92] applied Recursive Feature Elimination Support Vector Machines to this dataset and obtained a best error rate of 11.16%, which improved on the nearest shrunken centroid method (13.45%) [50]. Huang and Kecman provide several lists of the top 10 genes which vary considerably depending on the parameter settings but 5 of those listed in Table 3.11 are listed in their first top 10 and the number 1 gene in Table 3.11 appears consistently in their tables. Mahata and Mahata [129] ranked the genes by minimum probability of classification errors (MPE) for each gene using a Bayesian decision-making algorithm. They report accuracies from 85.9% to 96.77% depending on the number of genes used and the size of the test set. The best result was using just

Table 3.12: Maximal Cliques (≥ 3) found for Principal Genes in the Colon Cancer Dataset.

Clique	Genes				ARI %
1	M22382	R36977	M36634		58.2
2	M22382	X63629	M36634		53.8
3	M22382	H43887	M36634		53.0
4	M63391	M22382	R36977	M36634	49.6
5	H43887	M22382	R36977	M36634	43.4

Table 3.13: The confusion matrices for Cliques 1 and 2 (Table 3.12).

	Positive	Negative
Cluster 1	2	18
Cluster 2	14	3

	Positive	Negative
Cluster 1	0	21
Cluster 2	9	1

5 genes and a 50% test set. While this is a higher percentage than GSEP, GSEP was tested on classifying all 62 samples (using hold-one-out multifold cross-validation) and almost all papers we cite conclude that several samples appear to be misclassified (i.e. mislabelled). There may be a way of dividing the data into two sets (test and training) that minimises the effect of these on the test set accuracy and explains the unusually high accuracy reported by [129].

The number one gene from Table 3.11 is listed as number one in the table of [129] along with 6 others from Table 3.11. They cite M63391 and M36634 as important because they are muscle specific. [80] report that M63391 is down-regulated in colon-cancer patients. M76378 is said to involved in development, cellular differentiation and cell proliferation and has been associated with colon cancer [87]. M22382 has been correlated with the tumoural grade in both primary tumour and lymph node metastasis [42].

Clustering

The clustering was restricted to those genes that had showed a weight of ≥ 0.1 and further restricted to those for which a separating distribution could be found by the clustering algorithm. The clustering algorithm GCLUS has no knowledge of the class of any data point and is trying to separate dense regions around peaks in the distribution despite the high level of noise. Six genes participated in the various clusters found

The principal cliques found are shown in Table 3.12 and the confusion matrices for the leading two clusters are given in Table 3.13. From this we can deduce that these small gene groups, while not covering all samples, do offer a reasonably high degree of discrimination between malignant and benign colon tumours based on this small dataset and the assay size of 2000 genes.

Alon et al. [17] applied a two-way deterministic annealing clustering algorithm which builds a

Table 3.14: 47 prognosticating M/Z values with weights.

Peptide	Weight	Peptide	Weight	Peptide	Weight
MZ243.78535	2.243	MZ25.401403	1.573	MZ261.88643	1.109
MZ244.07686	1.986	MZ247.295	1.528	MZ261.28267	1.094
MZ244.36855	1.938	MZ25.49556	1.466	MZ464.76404	1.085
MZ244.66041	1.931	MZ433.90794	1.448	MZ247.58861	1.081
MZ244.95245	1.880	MZ463.55767	1.444	MZ417.35068	1.016
MZ245.24466	1.786	MZ463.1559	1.423	MZ416.96946	1.003
MZ245.53704	1.703	MZ435.46452	1.404	MZ3998.9223	0.987
MZ434.68588	1.647	MZ463.95962	1.347	MZ4010.7341	0.977
MZ245.8296	1.628	MZ4001.2832	1.333	MZ417.73207	0.960
MZ434.29682	1.615	MZ25.589892	1.325	MZ4008.3703	0.944
MZ246.70832	1.609	MZ4000.1027	1.279	MZ4011.9162	0.941
MZ247.00158	1.609	MZ464.36174	1.244	MZ8023.3507	0.661
MZ25.307419	1.602	MZ261.58446	1.204	MZ25.21361	0.437
MZ435.07512	1.600	MZ435.85411	1.177	MZ462.7543	0.155
MZ246.12233	1.596	MZ433.51923	1.169	MZ4024.9313	0.091
MZ246.41524	1.589	MZ4002.464	1.134		

binary tree of genes and tissues which was then ordered to place more correlated genes or tissues close to each other. They appear to have obtained an accuracy for identifying ‘positive’ samples close to 90%. They report that a few ‘positive’ samples do not appear to show the characteristics of the other tumour samples.

Ovarian Cancer Dataset

This dataset [140] has 253 samples of proteomic spectra from 91 ’normal’ samples and 162 diagnosed cases of ovarian cancer in different stages. Each sample contains 15,154 peptides defined by their mass/charge (M/Z) ratio. The spectra was obtained by mass spectroscopy on serum samples. Using multifold cross-validation, we obtained an accuracy of 97.63% using an STL equivalent to $t_{min} > 3.9$ and BVLF of 0.815 and 47 out of the 15,154 attributes. 38 of these had weights of over 1 standard deviation indicating the great potential effectiveness of this small subset for discriminating the cancerous cases (Table 3.14).

Petricoin et al., using a proprietary genetic algorithm, achieved 94% accuracy on an earlier version of this dataset. Alexe et al. [12] reported 100% accuracy using LAD [51]. LAD sets cut values on key attributes to separate the classes. The difficulty with this is that for any given dataset, if a LAD model exists that will achieve 100% accuracy, the researcher will likely find it especially using leave-one-out validation, as employed by these authors, where one can tune the model to almost the entire dataset. This, however, gives no surety that such accuracy can be achieved using that model on any other dataset where stochastic or other effects may well cause values to cross the LAD cutoffs. This problem only applies to techniques like LAD, which use specific ’cutoff’ values.

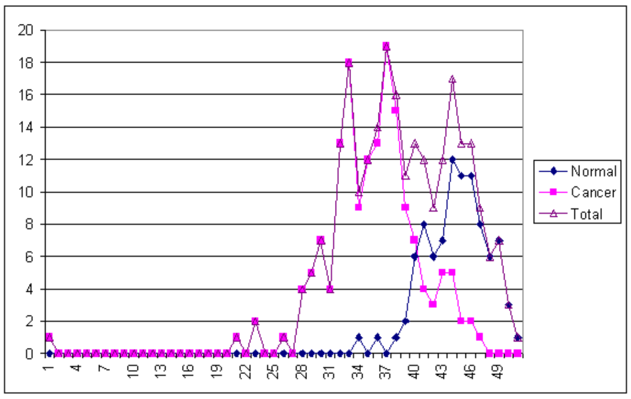

Figure 3.10: Class distribution on attribute MZ246.41524.

Table 3.15: Peaks identified by GCLUS on attribute MZ246.41524. If we assume only two classes, this has an ARI of 60.8%.

	Bins	Normal	Cancerous
1	$30 \leftrightarrow 33$	1%	27%
2	$34 \leftrightarrow 38$	4%	42%
3	$39 \leftrightarrow 41$	22%	9%
4	$42 \leftrightarrow 50$	73%	9%
Outliers		0%	12%

Clustering

Classifiers and clustering algorithms that require some domain information, principally the number of classes or clusters k, are in a sense artificial. The value k may be a realistic value but often it may be imposed on a more complex reality. This dataset clearly illustrates this. It is provided with only two classes, normal and cancerous but we do know that ovarian cancer has four stages described by medical science and all these stages are represented in this dataset. The classification result shows that proteomic analysis of serum is a powerful tool for detecting this disease and thus we can expect that it will also reveal other medically important states such as, perhaps, differences between the stages of this cancer.

The subspace clustering algorithm, GCLUS, presented in this paper, since it looks for peaks in the data on each attribute and does not impose any limit on their number, may be sensitive to more classes than we might expect. For example, if we look at attribute MZ246.41524 which is quite representative of many of our prognosticating attributes, we see a number of distinct peaks. Figure 1 shows the data for this attribute separated by class in terms of the bin totals after binning by GCLUS into 51 bins labeled 0-50. Table 3.15 shows the separate regions around the peaks selected by GCLUS.

GCLUS identified four peaks as described in Table 3.15. The peaks below bin 30 were detected but dropped as not statistically significant and their members classed as outliers. While the ARI

Table 3.16: Confusion matrix for clusters in clique: MZ243.78535 MZ244.36855 MZ463.55767 MZ261.88643 MZ3998.9223.

	Normal %	Cancer %
Cluster 1	63.7	11.7
Cluster 2	0.0	17.9
Cluster 3	1.1	34.6

Table 3.17: Confusion matrix for clusters in clique: MZ245.8296 MZ434.29682 MZ246.70832.

	Normal %	Cancer %
Cluster 1	79.1	7.4
Cluster 2	9.9	13.0
Cluster 3	0.0	27.8
Cluster 4	1.1	16.1
Cluster 5	0.0	20.3

for the separation on this single feature, assuming two classes, is only 60.8%, we can see that the separation is as good as we are likely to achieve on this data given the class overlap. There are, indeed, two major peaks in the cancer group while there is one or perhaps two in the normal group. Additional heuristics could be added to merge peak three into peak four and recover the first dropped peak but researchers know that fitting heuristics to one piece of data leads to overfitting and is not a robust approach for general application. The subsequent clustering output of GCLUS, since it uses projected clustering, is dependent on the initial one dimensional clustering and we can see that the output may be useful even if the ARI values, assuming only two classes, are not comparable with the classification result.

Tables 3.16, 3.17 and 3.18 show some subspaces output by GCLUS of sets of the proteomic spectra attributes with the clusters found in them and the breakdown by the given classes. The first subspace (Tables 3.16) would support the hypothesis that there are two classes within the cancer assays, the second (Tables 3.17) gives three such classes and the third just one (Tables 3.18). As different attributes are involved, it is possible that different components of the proteomic spectra provide different degrees of diagnostic specificity. We have also seen that the number of peaks detected on a given attribute is subject to both stochastic effects and the heuristics employed. This is inevitable, whatever the clustering technique employed. However, in cases of such importance, the small number of attributes involved could readily be manually marked for the peaks to be selected and then the subspaces and clusters would then reflect this expert input and could lead to important and reliable conclusions. In any case, the small subspaces reported in the tables still provide considerable discriminatory power.

Table 3.18: Confusion matrix for clusters in clique: MZ25.589892 MZ4000.1027 MZ464.36174.

	Normal %	Cancer %
Cluster 1	27.5	15.4
Cluster 2	17.6	1.9
Cluster 3	26.4	0
Cluster 4	4.4	71.6

Table 3.19: Lung Cancer confusion matrix. Subspace gene labels $1024\_at,\ 1059\_at,\ 1097\_sat$.

	ADCA %	MPM %
Cluster 1	22.0	0
Cluster 2	47.3	29.0
Cluster 3	26.7	61.3

Lung Cancer Dataset

This dataset [79, 19] has 181 samples each with 12,533 features. The samples are divided into 150 Adenocarcinoma (ADCA) and 31 Mesothelioma (MPM). This dataset illustrates a different situation from the previous discussed ones. While more complex classifiers (e.g. biclustering [20]) classified the samples with high accuracy up to 96.13%, GSEP achieved 92% on the ADCA samples (on a very wide range of t_{min} and BVLF) while not exceeding 23% on the MPM. However, if the BVLF was set high (e.g. 70%) and t_{min} low (0.2), then no features were sufficiently consistent to correctly identify more than 1 MPM sample, while a 52 gene model identified all the ADCA's correctly (100%). This result was quite robust to variations in the parameters. Eleven of the genes had weights greater than 1.0.

While the more complex classifiers provide a better accuracy overall, the advantage of GSEP is that it gives rather clear insight into the utility of using individual features to predict individual classes, which was not a focus of the earlier work [79, 20]. In this dataset, it is obvious that there is a set of features that consistently predict ADCA while this is not the case for MPM. While some success can be achieved for MPM samples, this only occurs if the filter which controls the effect of outliers is suppressed. This would suggest that whatever heuristics learnt by an algorithm to identify this class alone in this dataset might not extend to other such datasets, though it would continue to be possible to distinguish the classes using the genes that identify ADCA.

On the other hand, if the MPM samples were, in fact, made of two or more fairly balanced classes with different gene expressions, then the BVLF would prevent GSEP from identifying the group as a whole. That is, if there is no real single group or class as defined by a specific subset of features, then it will not be found. Such a situation could arise, for example, if the disease passed through different stages with distinct gene profiles.

GCLUS had some success seperating the classes based on sub groups of the strongest 11 genes output by GSEP (Table 3.19).

3.5.3 GSEP vs SERA

GSEP and SERA (Appendix A) were tested on the same datasets and some differences can be observed especially on the Lung Cancer Dataset. GSEP could not identify genes that distinguished the majority of the MPM samples (though the GSEP + GCLUS output did better) while the SERA related method found several. This reflects both charateristics of the data and differences in the methods. GSEP takes a more stringent view than the SERA related method requiring each voting gene to have sample gene values closer to the expected class mean than the other class mean on most samples. While this was the case for the ADCA samples, very few MPM sample genes met this criteria. SERA, on the other hand, only looks for sample genes that have values slightly greater than the control mean for all samples. This weak criteria reflects the fact that out of a very large number of features, such as 12,000, only a few meet even this criteria. This is acceptable because we only expect a few genes to contribute to a single disease condition.

If the SERA criteria was tightened to the GSEP level, then there would be a relative effect where fewer genes would meet the criteria, which would change the computed outcomes. Suppose we are using the method to estimate the number of true positives, then this number would decline or the experimental feature level statistical threshold could be reduced to maintain the number. In other words, the original experiment and the analysis of it are components in the error estimation and should properly be calibrated on data with known outcomes. As this is beyond the scope of this thesis, the method as presented in this thesis in Chapter A is calibrated based on what seems reasonable.

3.5.4 Conclusion

While this section is entirely couched in terms of medical diagnostic problems, the method is generic and can be applied to many problems especially where $N \ll d$, which is difficult for many other more complex approaches. The novel highly efficient approach GSEP can provide researchers with a clear and easy to interpret weighted list of prognosticator genes that can immediately be applied to classify further samples, even manually. The internals are sufficiently simple that the researcher can also clearly understand how the list is generated, which gives confidence in the output, and the weights are also simple to interpret. The small effectiveness advantage that can be achieved on some problems is an added bonus.

With GSEP as a pre-processor, the novel subspace clustering approach GCLUS also provides a very efficient way of finding key groups of features and the clusters amongst them. The cluster labels it provides show which characteristic of each feature, such as up or down regulation of genes, is responsible for each cluster.

Chapter 4

Outlier Detection in High-Dimensionality

4.1 Fundamental Considerations

As with clustering, outlier detection also suffers from the tendency for the maximum and minimum interpoint distances to become relatively the same with increasing dimensionality. While this effect was described and demonstrated by Beyer et al [31] and cited by numerous authors, we ask what is the fundamental cause of this problem? From this question, we can identify that there are certain benefits to this tendency that can be exploited. While outlier detection certainly becomes increasingly meaningless, certain features of the underlying classes, if such exist in the data, can be determined with increasing precision. This phenomenon has been widely exploited in machine learning and other areas but the implications for outlier detection have not been previously noted or investigated. The most important is that we can actually identify classes in the data as long as they differ in their variance. While authors have validated outlier approaches by trying to separate 'rare' classes on the presumption that these classes are outliers *per se* by being rare, we show that they likely succeeded, where they did, for another reason. These classes, regardless of size, differ from the dominant/other classes by a property which becomes important as dimensionality increases. We refer to this as 'classification by variance' and this is discussed in the next section. After that we introduce a number of novel outlier detection methods and compare them against the state-of-the-art in subspace and full space outlier detection, as this phenomenon we describe should be exploitable by any effective outlier method that can provide a ranking of points by their outlierness.

The contributions of the work presented in this chapter are twofold.

- A clear understanding of the basis of how outlier methods can be used for classification and under what circumstances;
- Several novel outlier detection methods that outperform existing methods in many datasets.

We can also identify two problem definitions.

- To generate and validate a ranking of points in a dataset based on their inherent 'outlierness' or 'unusualness'.
- To separate classes in data that differ in their variance from each other regardless of the degree of spatial separation.

The motivation and implications of outlier detection in high-dimensionality are now considered in more depth.

4.1.1 Outlier Detection and Classification Through Variance

In this thesis we introduce an entirely new concept in data mining, classification through variance rather than spatial location. This is accomplished using outlier detection methods and came about as a result of our research into outlier detection on high dimensional data. Several new methods for outlier detection and various competing methods are tested to demonstrate this effect. The success of the new methods, as seen from the results sections below, is notable, however a number of important questions have to be addressed. Firstly, a formal basis of this concept is required and it would be desirable to explore it extensively using synthetic datasets which would allow us to clearly distinguish this phenomenon from possible spatial effects. If classes can be separated by variance alone, then concentric and identically shaped distributions, differing only in variance should be separable, something that cannot be achieved using supervised classification, such as Support Vector Machines (SVMs), or unsupervised classification, i.e. clustering. This has to be demonstrated in the controlled environment of a synthetic dataset and this is addressed extensively in Section 4.5. In this section, we discuss the curses and blessings of high dimensionality and show how the primary blessing, concentration of measure, can be exploited to yield class separation.

4.1.2 Curses and Blessings of Higher Dimensionality

Many papers in data analysis, including outlier detection and clustering in higher dimensionality, cite the 'curse of dimensionality' first described as such by Bellman [25]. Donoho [55], reviewing a large body of work in Mathematics, enumerates one curse and three blessings. The curse is that described by [25], which is based simply on the exponential increase in the number of subspaces as dimensionality increases. Alternatively, one can note the exponential increase in the volume of the data space, which, for a d dimensional space with normalised side l, is l^d. This means the data, if at all homogeneous in the space, is increasingly sparse.

The first blessing is referred to as 'concentration of measure' (CofM), first described by V. Milman [134] for a common property of probabilities on product spaces in high-dimensionality. It asserts that a 'reasonable' function $f : X \rightarrow \Re$ defined on a 'large' probability space X 'almost always' yields values close to the mean of f on X. A reasonable function would have a finite and unique mean and this is the case for a broad class, Lipschitz functions[1]. In fact, CofM has been shown to apply to many types of dependent variables [58]. CofM is a generalisation of the law of large numbers.

For example, given a Lipschitz function on a d-dimensional sphere on which we place a uniform measure P, then for a random variable $X, d \to \infty$

$$P\{|f(x) - Ef(x)| > t\} \leq C_1 e^{-C_2 t^2} \tag{4.1}$$

where C_1, C_2 are constants independent of f and d. Thus, the measure is nearly constant and the tails behave, at worst, as a scalar Gaussian random variable with absolutely controlled mean and variance [55]. This principle is not confined to a sphere and has wide applicability. One probabilistic aspect of concentration of measure is that a random variable that depends (smoothly) on the influence of many independent variables, but not especially on any one, is essentially constant at the mean value [116]. The requirement of smoothness is provided by functions being Lipschitz.

Another example is coin tosses. If a coin of unknown bias p is thrown m times, then from the Chernoff bound

$$\forall \epsilon > 0, P\left(\left|\frac{\sum_i^m Xi}{m} - p\right| \geq \epsilon\right) \leq 2e^{-2\epsilon^2 m} \tag{4.2}$$

For example, if a balanced coin is thrown once there is complete uncertainty regarding the outcome. If the coin is thrown 1000 times, the number of heads will, in all probability, be rather close to 500. The larger the number of tosses, the more predictable the outcome, which well illustrates the blessing.

The other blessings cited by Donoho are not unrelated to this phenomenon but are not discussed here as they are not directly relevant to this thesis.

4.1.3 Data Mining and Concentration of Measure

CofM is closely related to the law of large numbers and the Central Limit Theorem which are basic to probability theory. Even though we have not seen any explicit reference to CofM in the related data mining literature, results based on probability theory, notably the often cited work of Beyer et al. [31], exploit this phenomenon. Beyer et al. showed that for a large class of problems, distance approximations designed to speed calculations, in particular tree indices, break down above just a few dimensions. Beyer et al. derive their result subject to a condition that the measures examined – interpoint distances – converge on a mean with increasing dimensionality. They showed that this was valid for a wide range of dataset scenarios [31].

The phenomenon which makes distance computations decreasingly useful in the high-dimensional space, as shown by [31], is thus a consequence of the convergence on the mean (as in Equations 4.1 and 4.2), which is in itself a blessing if we wish to estimate the mean or exploit this convergence.

[1]A Lipschitz continuous function is such that a line joining any two points on the graph of the function is never steeper than a certain constant, the Lipschitz constant for the function. To prove a finite mean, the constant should be finite.

Thus concentration of measure is itself the problem and the blessing. The problem is that individual points or subspaces cease to be ‘special’, i.e. notably different from others, which is the very basis of the notion of outlierness (for example). On the other hand, while this notion is being taken away, determination of the mean $\mathbf{c}$ or probability p is being facilitated. One important utility of that with reference to outlier detection is explored in this thesis.

4.1.4 Class Separation by Variance

Having touched on the key idea, we can proceed to take advantage of both theoretical and empirical work on this measure to prove the following theorem:

Assertion 1: In general, an ensemble of subspaces of size m can provide a measure that distinguishes classes $\{A, B, ..\}$ if each class has (at least) different variances σ and has a Lipschitz distribution and m, the size of the ensemble, is greater than some empirical constant m'.

Let the dataset be D with d independent dimensions and the arity of the subspaces measured be k. We proceed to discuss this assertion as follows:

First we need to show that CofM applies, starting with determining the number of independent variables in the application. For this, we define an outlier score measure, which can be applied to a subspace, and then estimate the number of independent sets of subspaces in an ensemble. This result is then generalised. This is necessary because if subspaces larger than 1D are combined, it can be objected that these subspaces are not independent as they are made of combinations of the original attributes.

Second, if the measure employed converges on its mean, we show that this allows classes with different variance to separate within a ranking of the outlier scores.

Consider a two-dimensional subspace $\{A_i, A_j\}$ (e.g. Figure 4.1) in which a simple generic approach to outlier determination is followed as in [85] which applies a grid and uses the count in the grid as the (inverse) measure of sparsity for scoring the points. [85] only applied this to the single dimensions but Figure 4.1 is an example of where there is a clear benefit if a larger subspace grid is used. Let $\delta_l(x)$ be the density measure (cell count) for point x on attribute or attribute set l. In Figure 4.1 point x has $\delta_i(x) = 4, \delta_j(x) = 4, \delta_{ij}(x) = 1$. Purely for the sake of this example, let the outlier score over m spaces (1D, 2D or kD) be

$$\phi(x) = \sum_{l}^{m} \frac{N}{\delta_l(x)}$$

In the context of CofM, where we consider a set of measures $\mathbf{X} = \{X_1, X_2, ..., X_m\}$ and we expect with high likelihood the mean of the sum over $\mathbf{X}$ to converge on some c, then the mean of the measures $\phi(x_i), \forall x_i \in D$ would be expected to converge on some constant related to the probability of the x_i being an outlier subject to those measures conforming to the requirements of CofM. This will be the case if we are sampling the tails of a distribution where the cumulative probability distribution function under those tails is finite. That is, there is some fixed probability of

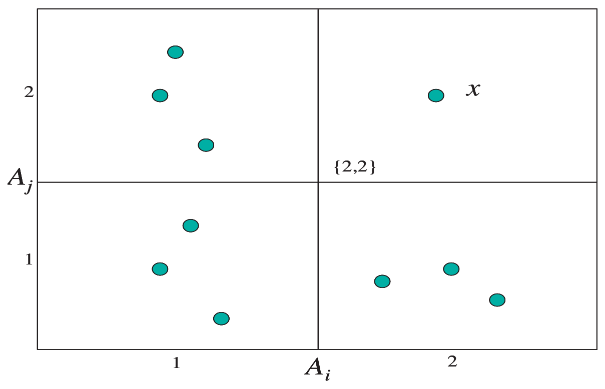

Figure 4.1: Local density measure example on two attributes.

a point becoming an outlier and we are accumulating an estimate of that probability by a series of independent measures (in fact, strict independence is not required [58]). In the following analysis and in subsequent sections, the division by the number of measures m to convert the sum to the mean is safely omitted because m is fixed and thus the same for all classes, and therefore falls out in any relative comparison between classes. That is, c can be safely considered to be multiplied by m so the sum is converging on mc.

Thus in the 1D analysis, $\phi(x)$ is accumulated over m $(= d)$ independent variables. In the 2D case, subspaces overlap due to shared attributes but in each space $\{A_i, A_j\}$ the only dependency of $\delta_{ij}(x)$ on the attributes is an upper (lower) bound provided by exactly one attribute. If all $k > 1$ size subspaces of d attributes are inspected such that $m = \binom{d}{k}$, at most one attribute will provide the upper bound for $\binom{d-1}{k-1}$ subspaces, another for at most $\binom{d-2}{k-1}$ and so forth. This results in a set of at least $d - k + 1$ independent subsets of subspaces of decreasing size where each set has a common dependency, albeit weak. Thus, in the 'worst case' scenario, the number of independent measures is still at least $d - k + 1$. This result applies to any measure that shows a dependence on at most one attribute. If a measure was dependent on a combination of 2 or more attributes, then the number of independent variables would be larger than d as the number of distinguishable subspaces size $k > 1$ is greater than d. Thus the bound is general to any measure: Any outlier method computed on ensembles of $\binom{d}{k}$ subspaces will have at least $d - k + 1$ independent sets of subspace results. For a fixed k, especially with $k \ll d$, this is $O(d)$.

So far we have assumed the attributes are independent. This is reasonable because, whatever the dimensionality of a dataset, it can always be transformed into a possibly smaller number of essentially independent dimensions by any of the standard means.

Therefore, the concentration of measure phenomenon will apply to the outlier measure ensemble score ϕ if 1) the measures are Lipschitz and 2) the number of attributes is sufficient for CofM to practically apply. Regarding (1), underlying classes are usually Lipschitz unless strong boundary effects are evident. The measure we are taking here amounts to an estimate of a part of the proba-

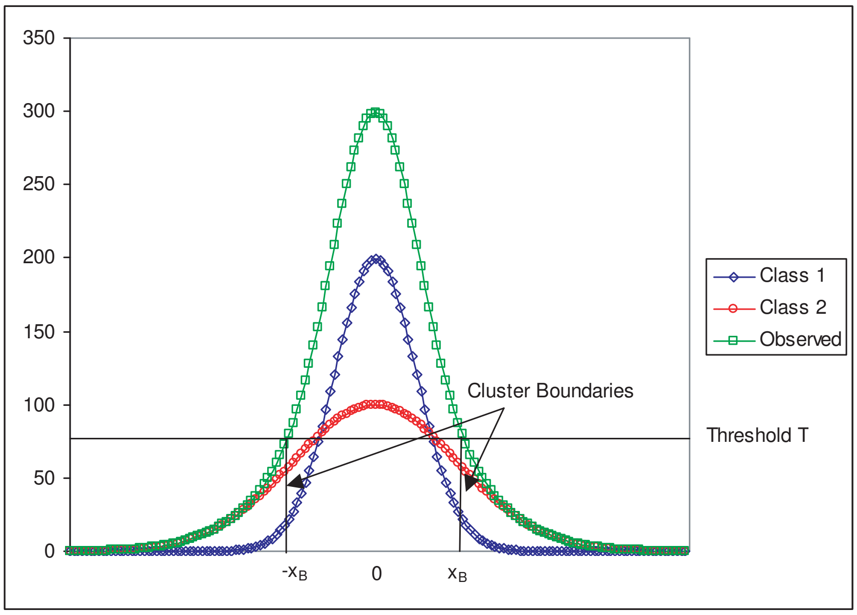

Figure 4.2: Clustering a mixture of two Gaussians.

bility distribution function which will be Lipschitz if the underlying distribution is. Regarding (2), Equation 4.2 shows an exponential convergence on the mean suggesting convergence even for low dimensionality. The empirical results of Beyer et al. [31] suggest 10-15 dimensions are sufficient. Later we present our results (Section 4.5.12) showing how this phenomenon rapidly intensifies at similar low dimensionality. Thus we can take $m' \approx 15$.

Now we have to show how the classes are separable. Figure 4.2 shows an example of a difficult case for class separation, separating two concentric classes. There is no spatial differentiation to exploit but the difference in the variance results in one class contributing the majority of the distribution tails. In one dimension, that would only allow us to collect a small number of points from the extreme tails, which likely belong to high variance Class 2, however, as dimensionality increases, CofM means that the points belonging to each class C_i will tend to yield a measure relatively close to some constant c_i. If the measure has a dependency on the class variance, then $c_1 \neq c_2$ if the variances differ causing the classes to separate in the measure ranking.

Let us consider some common approaches to the problem. Following [85] we can define a measure on each point x based on the local density observed around x. This density will be dependent on the observed distribution which, in Figure 4.2, is the sum of the probability distribution functions (PDF) of the two classes. However, the position of x relative to the mean is solely due to the PDF of its class C_i. Suppose we take an approach that applies a threshold T and only those points detected in areas with local density below T are inspected ($\forall x \in D \mid \delta(x) < T$). T effectively creates a cluster boundary (e.g. as in Figure 4.2). We can either define a measure that is based on local density as in the earlier part of this proof or on the distance from the cluster, however defined.

This will yield a real value estimate of 'outlierness' but is also a direct consequence of the PDF of the underlying class. Alternatively, we can take a binary measure giving equal score to any points found beyond the cluster boundary. However, over multiple attributes or subspaces, this measure is again dependent on the PDF of the underlying class since the variance of the class determines the likelihood of the point appearing in the tail. This binary score approach is like the coin flipping case of Equation 4.2. For a probability distribution function $F(N, \mu, \sigma)$, the outlier measure will converge on the probability p_i for attribute i where (assuming no boundary effects)

$$p_i = 1 - \frac{\int_{-x_B}^{x_B} F(N, \mu, \sigma)}{\int_{-\infty}^{\infty} F(N, \mu, \sigma)}$$

If we treat N and μ as constants and set the total cumulative distribution function as C, for example, by normalising the distributions, then

$$p_i = 1 - \frac{1}{C} f(x_B, \sigma)$$

making clear the dependence of p on σ. From this we can expect that such an approach to outlier scoring will cause entire classes with different σ to converge their scores around different values.

As it has been shown that the tails of the (outlier) measure (for Lipschitz functions) are approximately Gaussian (Equation 4.1), if we plot the ranked outlier scores for all $x \in D$ such that for rank $i,\ x_i \geq x_{i+1}$, then each class will produce an 'S' shaped curve. The outlier score of most points in a class will cluster around some constant giving a close to linear plot with 'tail' points having scores increasingly larger (smaller) than this constant. Exactly this can be seen in the figures in Section 4.7.2.

Therefore, under the appropriate conditions discussed, the assertion can be accepted.$\square$

The intuition behind this in terms of an outlier method can be expressed as follows:

If a point is contained in a class of high variance, then this point is likely to be an 'outlier' in many low-dimensional subspaces, and vice versa.

Assuming a density-based *Outlier* method applied in 2-dimensional subspaces (the argument easily carries over to other cases) on a dataset D containing 2 classes, a low variance class A and a higher variance class B, we can argue as follows:

First, assume a point x is ranked high in the outlier ranking returned by the method. Then, for 'many' 2-dimensional subspaces D_{ij} of D, x is assigned a high outlier degree in D_{ij} by the corresponding *Outlier* method. Since we employ a density-based algorithm for outlier detection in the 2-dimensional subspaces, this means that, for 'many' 2-dimensional subspaces D_{ij} of D, x is isolated (i.e., not in a dense region). Since the class A has low variance, points in A are expected to be not isolated in 'most' 2-dimensional subspaces. Hence x is more likely to belong to B than to belong to A. Consequently, if a point has a $score$ value above a certain threshold, this point is most likely to belong to B.

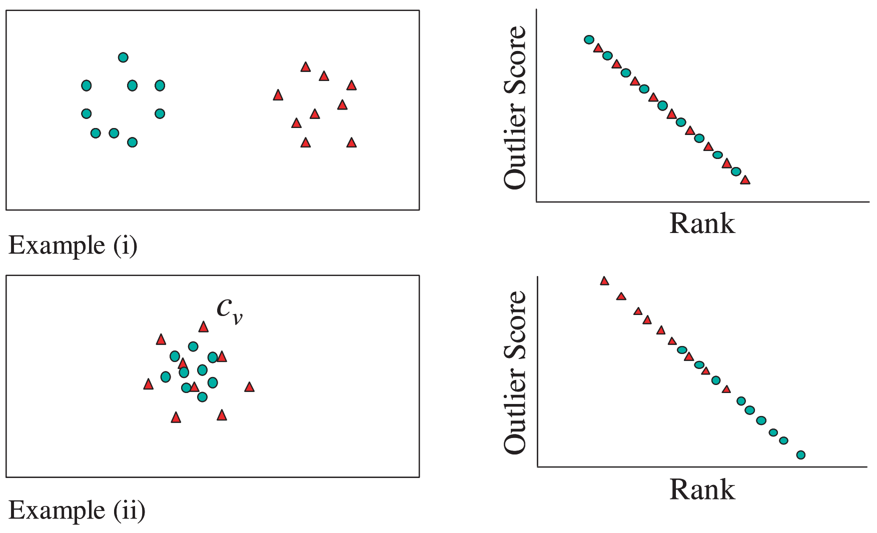

Figure 4.3: In Example (i) two classes with equal variance show no separation in the outlier ranking. In Example (ii), the difference in variance leads to a degree of separation.

Second, assume a point x is ranked low in the outlier ranking returned by the method. Then, for 'most' 2-dimensional subspaces D_{ij} of D, x is *not* considered an outlier in D_{ij} by the corresponding *Outlier* method. This means that, for 'most' 2-dimensional subspaces D_{ij} of D, x is in a dense region. Since the class B has high variance, points in B are expected to be isolated in 'many' 2-dimensional subspaces. Hence x is more likely to belong to A than to belong to B. Consequently, if a point has a $score$ value below a certain threshold, this point is more likely to belong to A.

In particular, if the difference between the variance of A and the variance of B is high enough, and if d is high enough to provide a sufficient number of 2-dimensional subspaces, we expect thus to separate basically *all points in* B from *all points in* A – just because basically all points in B will have higher outlier scores than any point in A.

Since this observation is totally independent of the mean μ around which the points in a class are distributed, our algorithm works well even if two classes overlap completely – as long as the two classes differ significantly in variance.

Figure 4.3 illustrates the general phenomenon. In Example (i), there are two equal variance classes and this results in their members being well mixed in the outlier score ranking. However, in Example (ii), the variances of the two classes are different and there is a tendency for the higher variance class c_v to populate the higher outlier score rankings. If the classes are heavily overlapping, in any given subspace, only a few members of c_v will *visually* appear as outliers. However, as their underlying variance is higher, accumulating over multiple subspaces, eventually almost all can differentiate themselves from the lower variance class.

4.2 T*: A 2D Approach

In this section, a new framework for outlier detection in an arbitrary number of dimensions is provided. This is based on rankings obtained by investigating low-dimensional subspaces (as opposed to Feature Bagging [114]) that consist of more than one attribute (as opposed to SOE1 [85]). New methods are provided that exploit 2D ensembles. In Section 4.5, a method for 3D and higher subspaces is provided. Comparative results for these three and other state-of-the-art algorithms are then provided. In the related work on outlier detection, Section 2.3.1, we discussed that among ensemble outlier methods, the state-of-the-art is provided by *Feature Bagging* [114] and *SOE1* [85]. These represent two contrasting choices regarding subspace size. While SOE1 uses only 1D ensembles, Feature Bagging uses $d/2$ and higher. These algorithms are tested along with leading full-dimensional methods, the two distance-based methods *ORCA* [24] and *Robust Mahalanobis Distance (RMD)* [153], and the density-based *LOF* method [39].

Initially researchers lacked adequate ground truth for testing their algorithms and generally relied on presenting some sample results showing that the output was reasonable. Then the concept arose [9] that test datasets with labelled classes where one or more classes were 'rare' or were induced to be 'rare' could act as a validator. It was presumed that rare classes, being rare, should exhibit some greater outlierness. This was extended to induced rare classes, where the new class was derived by random selection from an existing larger class. The class selected was typically a class of interest, such as a diseased sample class in a medical dataset. Typically, such classes are likely to exhibit greater variance. Thus subsequent authors (e.g. [114, 85]) validated their work on such data. While the results reported are certainly interesting, it is important to consider explicitly why the results were obtained. After all, smallness, as of itself, is unlikely to be the cause of the higher outlierness of a class unless a clustering algorithm is used that flags small clusters as outliers.

On the basis of the theoretical considerations already introduced (Section 4.1.1 and following), we choose not to pursue the 'rare' class notion and rather investigate whether classes of all sizes can be separated. Problems we address initially are binary class separation problems in which the two classes are assumed to differ in *variance*. We expect that, for two underlying classes A and B of different variance, our outlier ranking basically separates all points in A from all points in B, *even if the two classes overlap completely and are of the same size*. As already discussed, this is because points in the class of high variance are more likely to be outliers consistently in many low-dimensional subspaces and are thus ranked higher in the resulting outlier ranking (with some provisos). In Section 4.7 separating more than 2 classes is investigated.

Our experimental results show that this approach works and the new methods we introduce are frequently more effective than existing methods when applied to this task. We will also review the results from state-of-the-art classifiers on the test sets and this shows that our unsupervised approaches are frequently competitive with and sometimes superior to such supervised approaches.

4.2.1 The T* Framework

We assume a *dataset* $D \subset \mathbb{R}^d$ in d dimensions. Each of the dimensions represents a different *attribute*. For every point $x \in D$ and every $i \in \{1, \dots, d\}$ we denote by x_i the value in the i^{th} attribute of x, i.e., $x = (x_1, \dots, x_d)$. If $k \leq d$ and $S = \{s_1, \dots, s_k\} \subseteq \{1, \dots, d\}$, then D_S denotes the set $\{(x_{s_1}, \dots, x_{s_k}) \mid x \in D\}$, i.e., the projection of D on the attributes indexed by S.

For every finite set Z, let $|Z|$ denote the cardinality of Z.

4.2.2 Algorithmic Idea

In this section we assume an efficient algorithm *Outlier*, which, given a subspace D_S for a 'small' set S of attributes (for T*, in our experiments, $|S| = 2$ is sufficient) and a point $x \in D_S$, determines a degree of 'outlierness' of x in D_S such that

$$out(x, D_S)$$

denotes this degree.

We will describe two such algorithms in detail in Section 4.3; however, other heuristics could be applied as the *Outlier* algorithm in this framework. Note that we have not defined what an 'outlier' in D_S is, since there is no unique commonly accepted definition of that term. Different versions of the algorithm *Outlier* correspond to different interpretations of the term 'outlier in D_S'.

The outline of our algorithmic approach to outlier detection in D is as follows. Intuitively, our algorithm accumulates an outlier score for every data point, by counting how often and to which degree it is an 'outlier' in a set of all subspaces of a low dimension k. The top-ranked points are considered outliers.

1. Compute an outlier score for every point in D.

   (a) Set the parameter $k \ll d$ (size of subspaces).

   (b) Let $S_1 \subseteq \{1, \dots, d\}$, ..., $S_z \subseteq \{1, \dots, d\}$ be all subsets of $\{1, \dots, d\}$ of size k, i.e., $|S_i| = k$ for all $i \in \{1, \dots, z\}$.

   (c) For every point $x \in D$, compute an outlier score

   $$score(x) = \sum_{1 \leq i \leq z} out(x, D_{S_i})$$

   Note here that we assume the values of out to be calibrated over the different subspaces, such that a high out value in one subspace cannot become predominant over those obtained in other subspaces. As highlighted above (4.1.4), there is no need to divide this score by z to create the mean as this would not affect the ranking of the outliers.

2. Rank the points in D with respect to their outlier scores.

(a) Compute a ranking $x^1, \dots x^{|D|}$ over the points in D such that the values of $score(x^i)$ are non-increasing when i is increasing.

(b) If queried for the top N outliers in D, return $x^1, \dots, x^N$.

A key question in this approach is over which subspaces we should accumulate the outlier scores, i.e., how to choose the dimensionality k. On the one hand, taking all subspaces of a relatively high dimension k would be intractable. On the other hand, as argued in Section 2.3.1 and 4.1.4, it is not advisable to accumulate only over subspaces of dimension 1 (as in [85]) and when d is large, so remains $d/2$ (as in [114]).

As the experiments described in Section 4.4 demonstrate, a value of $k = 2$ is sufficient to outperform standard outlier detection methods on typical datasets, while at the same time resulting in a very efficient and effective method.

4.2.3 Considerations Regarding the Dimensionality d of the Original Dataset

We have discussed that a high enough dimensionality d of the original dataset is *key* for our method to work effectively (Section 4.1.4) due to the phenomenon of concentration of measure. Depending on the exact heuristics employed there can be other restrictions. If the method generates a binary score on each ensemble, sufficient ensembles are needed to rank all points if such a ranking is required and the number of unique subspaces is limited by d and k. We will present both such binary score heuristics and real-valued heuristics. Each has its advantages.

Note that none of the closely related work on outlier detection attempts genomic or proteomic data. Outlier detection (classically) seeks to find individual anomalies or rare classes in the presence of major classes or distributions. Genomic and proteomic data typically have a very small number of interesting features and no large coherent class(es). Thus such data has not been the focus of research on outlier detection so far and therefore we have also not applied the outlier methods to this kind of data. The datasets we use for empirical evaluation below have between 20 and 166 attributes. Datasets over 10-15 dimensions can be considered ‘high’ due to the onset of high-dimensional effects [31].

4.2.4 Application to Unsupervised Class Separation

T* can be used for unsupervised (binary) class separation even beyond the ‘rare classes’ case, if the given dataset fulfills the following conditions.

- The data is of high enough dimensionality for T* to produce a discriminative ranking of the data points.
- The given data separate in two classes, A and B.[1]

[1]We assume underlying true populations $\mathcal{A}$ and $\mathcal{B}$ and denote by A and B the respective fixed sets of datapoints contained in the given dataset D.

- Class B is of higher variance than class A.

Under the given assumptions, we propose the following straightforward unsupervised class separation method.

1. Run T* on the given dataset D to obtain a ranking $x^1, \ldots, x^{|D|}$ of the points in D.
2. For every $t \in \{1, \ldots, |D|\}$ assign a likelihood of the point x^t being in B in a way such that this likelihood decreases as t increases. Alternatively, if the class ratio $|A|/|B|$ is known, label the points $x^1, \ldots, x^{|B|}$ with B and the remaining points with A. This provides 'fuzzy' or 'hard' classification.

4.3 Two *Outlier* Algorithms Employable by T*

As stated in Section 4.2.1, one could use different variants of *Outlier* algorithms. We describe two variants here, both building on TURN* [66], a state-of-the-art clustering algorithm which has shown to outperform various others (DBSCAN, CHAMELEON, CURE, ROCK, Wavecluster, and k-Means) and has the additional advantage of being parameter-free. Since TURN* has proven to work best in 2 dimensions so far, but scales exponentially in d, we restrict our use of the T* framework to $k = 2$ in this paper. However, our experimental results show that this already yields excellent effectiveness on real-world datasets.

To summarize the TURN* method, we first need to introduce the concepts of nearest neighbours and of clusters, given a real-valued parameter r called *resolution*.

Neighbours and Nearest Neighbours. Let $x, x' \in D$, $S = \{i, j\} \subseteq \{1, \ldots, d\}$, $i \neq j$. x is a *neighbour* of x' in D with respect to S if $|x_i - x'_i| \leq r$ and $|x_j - x'_j| \leq r$. x is a *nearest neighbour* of x' with respect to S (an S-NN for short) if x is a neighbour of x' with respect to S and there is no $y \in D$ with

$$\begin{aligned} y_j \neq x'_j \quad & \text{and } (x_i < y_i \leq x'_i \text{ or } x_i > y_i \geq x'_i) \text{ or} \\ y_i \neq x'_i \quad & \text{and } (x_j < y_j \leq x'_j \text{ or } x_j > y_j \geq x'_j)\,. \end{aligned}$$

Internal Points and Clusters. Let $S = \{i, j\} \subseteq \{1, \ldots, d\}$, $i \neq j$, and $x, x' \in D$. x is *internal* in D_S if x has at least 4 S-NNs. x and x' are called *reachable* in D_S if there are internal points $n_0, \ldots, n_z$ in D_S such that x is an S-NN of n_0, n_m is an S-NN of n_{m+1} for all $m < z$, and n_z is an S-NN of x'. A *cluster* in D_S is a maximal set of points that are pairwise reachable in D_S.

TURN* consists of two modules. The first component is a clustering algorithm, which, given a dataset D and a resolution r, assigns all points in D to clusters. Clusters are built starting with a not yet touched internal point and recursively adding nearest neighbours for every internal point reached. The second component varies the resolution fed to the first component in order to find the 'best' clustering, repeatedly calling the first component. The reader is referred to [66] for details. It is important to note that the resolution parameter is adjusted by TURN* and need not be dealt with by the user.

In what follows we describe two *Outlier* methods; the corresponding variants of T* are called T*ENT and T*ROF.

4.3.1 T*ENT – Finding the Best Resolution by Entropy

The *Outlier* method in T*ENT varies the resolution selection criterion in the second TURN* component. It calls TURN* and collects a series of mean cluster density values (see below) of the clusterings obtained while varying over different resolution values r. The entropy of the clusterings are computed for all the resolution values at which the mean density changes its trend, i.e., at which the series shows a 'knee' suggesting an area of stability in the clustering results. Finally the clustering with the lowest entropy is selected as this likely has the least number of outliers. Once this clustering is found, all points in clusters that are smaller than a certain threshold θ are defined outliers. The outlier degree out for a given point is then simply 1 if the point is an outlier (in a cluster of size $< \theta$) and 0 if the point is not an outlier (in a cluster of size $\geq \theta$). In particular, the outlier degree is just a binary value expressing whether or not we consider a point an outlier rather than a real value expressing how much we consider a point an outlier.

In more detail, the behaviour of the *Outlier* method in T*ENT, applied to a 2-dimensional dataset $D_{\{i,j\}}$, can be described as follows.

1. Run TURN* on $D_{\{i,j\}}$.
2. Let $r_1, r_2, \ldots, r_T$ be the sequence of resolutions TURN* goes through.
3. For every resolution r_t, $1 \leq t \leq T$, compute a mean density with respect to the corresponding TURN* clustering as follows. For every $x \in D$ compute a local density

   $$\left(\sqrt{L_i(x)^2 + R_i(x)^2} + \sqrt{L_j(x)^2 + R_j(x)^2}\right)^{-1}.$$

   Here $L_i(x)$ is the closest nearest neighbour (for resolution r) whose value in attribute i is not higher than x_i (and accordingly with j instead of i); $R_i(x)$ is the closest nearest neighbour (for resolution r) whose value in attribute i is not smaller than x_i (and accordingly with j instead of i). The mean density is then the mean over all local densities of non-outlier points $x \in D$.
4. Detect all the resolutions r_t^* for which there is a change in the second differential of the series of mean density values.[2]

[2]This technique is routinely used in time series analysis to render a series stationary [132].

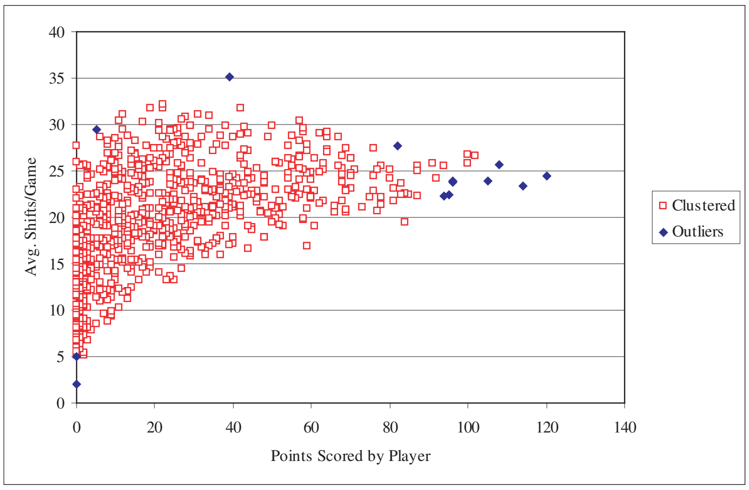

Figure 4.4: Outliers in a 2-dimensional subspace, automatically detected by the *Outlier* method in T*ENT (sample attribute pair, NHL data, $\theta = \min\{100, \frac{|D|}{100}\}$). Filled points are flagged as outliers in this 2-dimensional subspace, the others are not considered outliers in this space.

5. Of all those resolutions r_t^*, pick the one for which the corresponding clustering C has the lowest entropy value given by

$$H = \sum_{c \in C} p_c ln(p_c)$$

where p_c is the probability of a datapoint falling in cluster c.

6. For every $x \in D$, let

$$out(x, D_{\{i,j\}}) = \begin{cases} 1\,, & \text{if } x \text{ is in a cluster of size} < \theta\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

Note that the *Outlier* method in T*ENT requires setting the parameter θ. We address this point in Section 4.4.

For illustration, consider Figure 4.4 for an NHL dataset [143] in which each point is a hockey player described by 16 attributes. The figure shows outliers flagged by the method used in T*ENT in the 2-dimensional space spanned by the attributes showing (i) the number of points scored by a player over the season, and (ii) the average number of shifts a player had per game. This attribute pair is unusual in that two low ranking players are flagged as outliers. In most pairs, only the high ranking players stand out.

4.3.2 T*ROF – Accumulating Outlierness over Different Resolutions

The *Outlier* method in T*ROF applies TURN* without the stopping criterion for optimal resolutions. It simply computes the out value of a point x as the resolution-based outlier factor (ROF) over all different resolutions that TURN* goes through over the resolution range. The ROF is the sum of the ratios of cluster sizes of the cluster the point x is contained in, as resolution changes. ROF was previously applied successfully to a 3-dimensional engineering dataset, cf. [63], but not developed for higher dimensionality. In more detail, the behaviour of the *Outlier* method in T*ROF, applied to a 2-dimensional dataset $D_{\{i,j\}}$, can be described as follows.

1. Run TURN* on $D_{\{i,j\}}$.

2. Let $r_1, r_2, \ldots, r_T$ be the sequence of resolutions TURN* goes through.

3. For every $x \in D$, let

$$out(x, D_{\{i,j\}}) = \sum_{1 \leq t \leq T-1} \frac{|C(x, r_t)| - 1}{|C(x, r_{t+1})|},$$

   where, for every t, $C(x, r_t)$ denotes the cluster to which the first component of TURN* assigns the point x, when this component is run with resolution r_t.

Note that T*ROF has no user-definable parameters and produces a real-valued outlier score.

4.3.3 A Remark on Complexity

T*ENT and T*ROF have a run time cost of $O(d^2|D|log|D|)$. For all of the $O(d^2)$ attribute pairs, all of the points in D have to be sorted according to their attribute values along both dimensions; this is what dominates the run time. The space complexity is dominated by holding a $|D| \times d$ matrix in memory. However, the algorithm only requires two attributes to be processed at any one time so the minimum size is $O(2|D|)$.

4.4 Experimental Results

In the framework of T*, many different heuristics for determining outliers in low dimensionality could be plugged in. We propose two novel and effective methods. One based on entropy, T*ENT, and the other based on the Resolution-based Outlier Factor (ROF) [63], called T*ROF. Since Feature Bagging uses LOF as a basis for determining outlier scores in subspaces, we also experimented with using LOF in the T* framework (T*LOF). While the Feature Bagging method of [114] uses a relatively small sample of high dimensional spaces, T*LOF enumerates all the 2D spaces using LOF. These three are compared with Feature Bagging, SOE1, Robust Mahalanobis Distance (RMD), ORCA, and full-dimensional LOF.

As it is difficult to find data with any ground truth ranking of outliers, we consider two types of datasets for evaluation.

Type 1. The data is ranked and we expect a relationship between this ranking and outlierness. In this case, the accuracy of T*is measured by a correlation with the provided rank. As type 1 datasets we used three National Hockey League (NHL) datasets [143], which are popular validation sets in outlier detection research. In professional sports data, the bulk of the players are often hard to differentiate from each other but the top players tend to stand out on many attributes. Players could potentially be outliers for many reasons, not just because they are top players in the league. However, due to the fact that many attributes (e.g., number of goals scored or number of assists) have many players with 'bad' values, we expect low-ranked players to be in a fairly dense area and thus unlikely to be outliers. For illustration, consider Figure 4.4. More isolated points can be found in the upper right corner, where both attributes have their higher and 'better' values. It is reasonable, therefore, to expect a stronger correlation of an outlier ranking for the leading players than for the less successful players.

Type 2. The data is labelled in two classes A and B and we expect B to contain outliers more often than A. In this case, the accuracy of the outlier algorithms is measured by testing how many of the topmost ranked outliers are in B, using the actual number of points in B as a cutoff. We also measure area under the ROC curve (AUC). Type 2 data have been exploited this way for validating outlier detection methods before [10, 114] using 'rare' classes. In our case an extensive range of UCI datasets [33] are used; they all have higher dimensionality and have data of two classes that are likely to differ in variance. In particular, medical data are often of type 2.

4.4.1 Parameters

T*ROF has no user adjustable parameters. T*ENT has a parameter θ which determines the minimum cluster size for a cluster to be considered 'major' and thus not contribute to the outlier result. In all the Type 1 experiments and most of the Type 2 experiments, this was set by the heuristic advised in [66] for TURN*, that is $\theta = \min\{100, \frac{|D|}{100}\}$. On two datasets, WDBC and Parkinsons, somewhat better results were obtained when this value was reduced, showing some sensitivity to this setting. SOE1 builds a histogram of the data and thus requires either digitised data or a digitisation parameter. For each experiment, a series of different settings were tried and the best results reported. SOE1 proved quite sensitive to this parameter. A mean could have been reported but then this risked a bias due to the range of values tested. LOF and Feature Bagging have a parameter $minpts$ but we found that a value of 30 generally gave the best results as previously reported. RMD and ORCA required no parameter settings.

4.4.2 Evaluation on Type 1 Data.

The NHL datasets for seasons 2003/04, 2005/06, 2006/07 (there was no 04/05 season) each provide an official rank and values in 15 or 16 attributes for about 1000 hockey players. The attributes vary from having fairly continuous values to having no more than four possible values.

The algorithms are evaluated under the assumption that top players are frequently outliers. Table 4.1 shows the results for the three most successful algorithms. Every 'Cor.' value is the Pearson Correlation between the outlier score and the NHL ranking. For 2006/07 the top seven outliers determined by T*ENT method contain the players with NHL ranks 3,4,5,6,8 (2,3,4,5,6,7,9,10 for 2005/06 and 1,2,4,6 for 2003/04) and none with rank lower than 60 out of about 1000 players. T*ROF also put high ranking players at the top of the outlier scores. SOE1 had less obvious success with the leading players. (None of the other algorithms yielded a statistically significant correlation so the ranking appears quite random.) To further illustrate the results, it is also shown for each algorithm, down to which depth in the resulting ranking one has to look in order to find any 5 out of the top 10 NHL-ranked players.

Table 4.1: Correlation coefficients (abbreviated by Cor.; all values significant at $p < 0.0005$) and top 10 outlier players (in order) by NHL ranking number. Players that actually have one of the top 10 NHL ranking numbers are highlighted in boldface. '5/10' denotes the rank at which 5 out of the top 10 (by NHL rank) players are covered (lower numbers are better).

	2003-2004		
Method	Cor.	Top Outliers	5/10
T*ENT	0.852	**1**,**2**,41,57,**6**,21,19,37,**4**,16	14
T*ROF	0.686	16,**2**,41,**6**,18,**5**,12,33,**3**,**4**	**10**
SOE1	0.843	37,91,34,90,41,131,7,28,155,**1**	47
	2005-2006		
Method	Cor.	Top Outliers	5/10
T*ENT	0.851	**2**,**3**,**4**,11,16,**6**,**10**,**7**,**5**,**9**	**7**
T*ROF	0.690	15,**9**,**6**,**7**,83,**5**,20,**3**,**10**,**4**	8
SOE1	0.858	11,15,16,59,27,72,165,35,**9**,51	33
	2006-2007		
Method	Cor.	Top Outliers	5/10
T*ENT	0.858	60,**4**,**8**,**6**,**5**,**3**,14,11,17,24	**6**
T*ROF	0.706	**6**,**4**,**5**,34,**9**,47,48,**3**,**7**,27	8
e SOE1	0.864	**5**,60,90,77,105,53,**3**,17,55,24	44

This type of data drew a clear line between two groups of algorithms as the other comparison algorithms performed poorly on or could not handle this data. RMD could not complete the NHL data, ORCA did not achieve a significant correlation (e.g. 2003/04: $r = -0.071$), and LOF, as published, and thus Feature Bagging, are not suitable for certain attributes of the NHL dataset due to many identical data points and gave non-significant results when the task was attempted.

Note that the correlation achieved by SOE1 is always higher than that achieved by T*ROF and basically always as high as that achieved by T*ENT, even though T*ROF and T*ENT were clearly superior when it came to putting the best players at the top of the ranking. Figure 4.5 shows that the good correlation value SOE1 achieves is due to the higher correlation in the middle 'bulk' of the data, i.e., it tends to rank the 'less interesting' players slightly better than our methods. T*ENT and T*ROF are consistently better at ranking players at the top of the ranking.

Type 2 Data.

These results are found in Section 4.6 after the introduction of the FASTOUT algorithm. FASTOUT is introduced in the next section to explore ensemble outlier detection using 3D and higher sub-

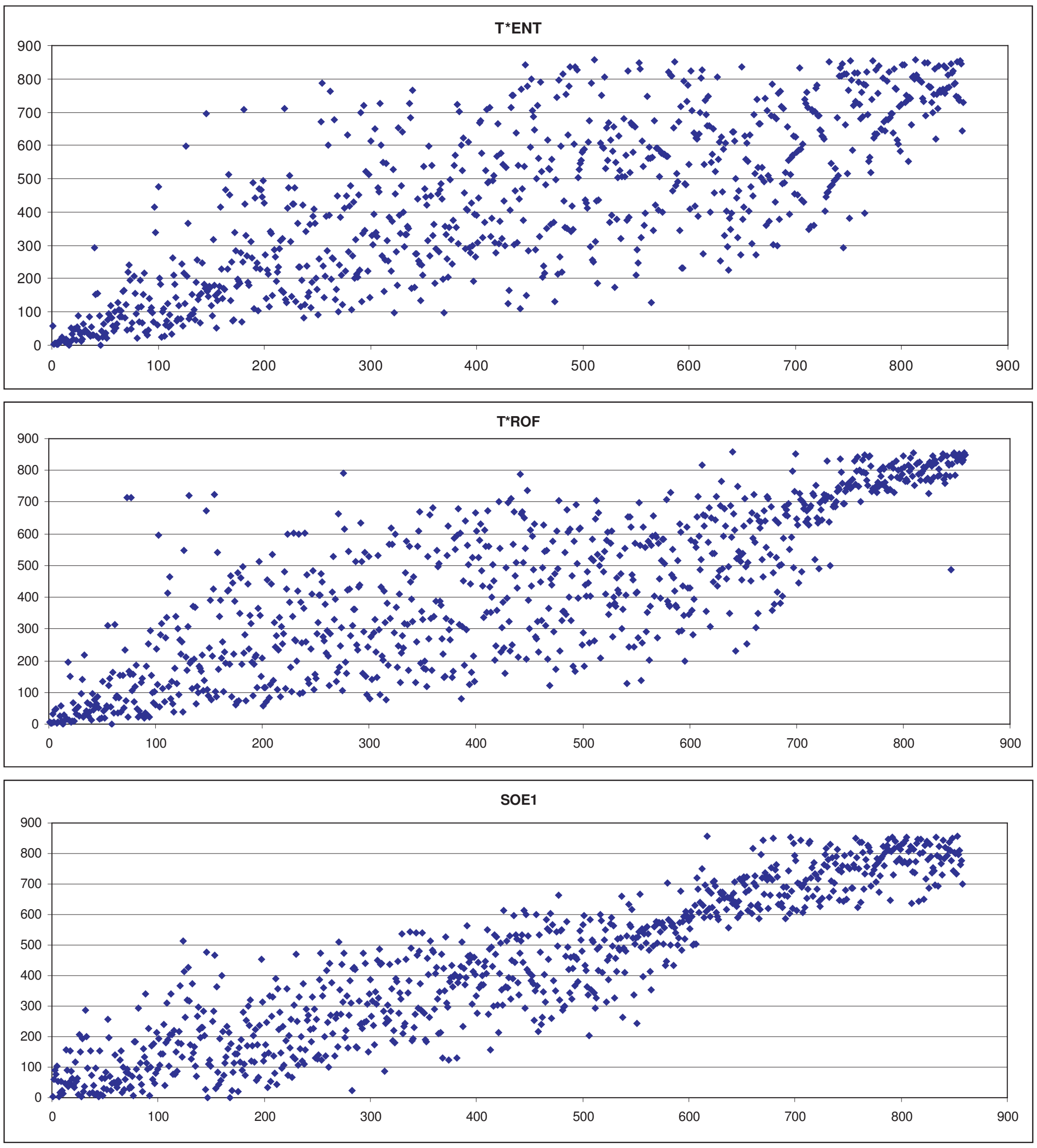

Figure 4.5: Correlation graphs (inverse outlier ranking on the horizontal axis versus NHL ranking on the vertical axis) for SOE1, T*ENT and T*ROF on NHL dataset 2006/07. The overall correlation achieved is 0.858 for T*ENT, 0.706 for T*ROF and 0.864 for SOE1.

spaces.

4.5 FASTOUT

FASTOUT employs a novel subspace ensemble outlier detection approach based on a new linear cost nearest neighbour technique. This approach can exploit an arbitrary size of subspace rather than 1D [85], 2D (T*ENT, T*ROF) or αD, where $\alpha \geq d/2$ for d dimensions [114]. We also show that only a sample of all the possible subspaces is sufficient to achieve high effectiveness, making it very efficient.

Earlier we discussed various approaches to outlier scoring. In particular, it was shown that a binary scoring based on 'clustered'/'not-clustered' in each subspace accumulated over sufficient subspaces (to be defined later) should provide the desired potential for class separation on the basis of differences in class variance. We therefore developed an efficient and effective algorithm for clustering within a given subspace (of any size), using an efficient nearest neighbour scheme. The algorithm is referred to as FASTOUT.

We start by providing certain definitions and then introduce our methodology for exploiting these for outlier detection. We then show how outlier detection can be used to sort, unsupervised, binary or even higher multiple underlying classes even when their distributions are fully overlapping.

4.5.1 Definitions

We assume a *dataset* $D \subset \mathbb{R}^d$ with d dimensions $A = \{1, 2, \ldots, d\}$. For every point $x \in D$ and every $i \in \{1, \ldots, d\}$ we denote by x_i the value in the i^{th} attribute of x, i.e., $x = (x_1, \ldots, x_d)$.

Definition 6. *k-Subspace S and subspace projection D_S.*

$S = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, d\}$. D_S is the projection of D on S.

Definition 7. *Subspace (Nearest) Neighbours.*

Let $x, x' \in D_S$. Then x is a nearest neighbour to x' with respect to S (an S-NN) iff 1) for any categorical attribute $A_i \in A$, $x_i = x'_i, \forall s_i \in S$. That is, the Hamming distance between x and x' is zero for S; or 2) for any numerical attribute $A_j \in A$, $|x_i - x'_i| \leq \epsilon, \forall s_i \in S$ for some $\epsilon \geq 0$.

Definition 8. *Subspace Cluster C.*

Let $x, x' \in D_S$. Then x is said to be *reachable* from x' if there are points $n_0, \ldots, n_z$ in D_S such that x is an S-NN of n_0, n_m is an S-NN of n_{m+1} for all $m < z$, and n_z is an S-NN of x'. A *cluster* C in D_S is a maximal set of points that are pairwise reachable in D_S. Let $\mathscr{C} = \{C_1, \ldots, C_q\}$ be the set of all clusters in D_S

Definition 9. *Outlier.*

Let $x \in D$. Then x is an outlier in D_S iff $\forall C_i \in \mathscr{C},\ x \notin C_i$ or $x \in C_i, |C_i| < \rho$, i.e. C_i is small. $O_S \in D_S$ is the set of outliers for subspace S.

Definition 10. *Outlier Score (Binary).*

Let $x \in D$ and $E = \{S_1, \ldots, S_{|E|}\}$ be the set of all subspaces. The outlier score $\phi(x) = \sum_{S \in E} f(S, x)$ where

$$f(S,x) = \begin{cases} 1 & x \in O_S \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Definition 11. *Outlier Ranking.*

$\forall x \in D$ the outlier ranking is the sorted list $\{1, \ldots, |D|\}$ indexed by i of outlier scores such that $\phi(x_i) \geq \phi(x_{i+1})$.

4.5.2 Finding Nearest Neighbours

In order to find the close neighbours of each point we hash the points into bins on each attribute. The number of bins $NumBins$ is $O(|D|)$ and for attribute A_i

$$BinWidth(A_i) = \frac{Max(A_i) - Min(A_i)}{NumBins} \tag{4.4}$$

where $Max(.)$ and $Min(.)$ represent the maximum and minimum values of the points on an attribute. As $NumBins$ is dependent on the size of the dataset, we define a parameter Q independent of dataset size, which is defined as

$$Q = \frac{|D|}{NumBins} \tag{4.5}$$

Q determines the average number of points in a bin and is O(1) as $NumBins$ is chosen to keep Q a fixed constant.

Unless the data is already normalised over some range, a pass over the database allows the minimum and maximum values of each attribute to be computed. From this, the bin width on each attribute is computed and another pass over the data allows the points to be assigned a bin number and a count be made for each bin. Then a $|D| \times d$ index can be created of the points sorted in each bin. This allows enumeration of the points in a bin without any searching at the cost of one more pass over the data (Algorithm 4).

To find the neighbours of a point $x \in D$ over subspace $S \subseteq A$, where the attributes are numerical, we look for all points that are within half a bin width ($w/2$) of x on all members of S. Let x be in bin j. Then we calculate the distance on attribute $s_i \in S$ over all points in bin j and keep those within $w/2$ of x. If x is in the first half of the bin j then we repeat over bin $j-1$, or, if in the second half, over bin $j+1$. this is repeated over all attributes in S keeping only those points that are neighbours on all members of S. For categorical attributes, points with the same value are clustered. This process has a complexity of $O(|D|d)$ with an approximately linear dependency on Q.

Algorithm 4 *RankedOutliers()* – Detecting, Scoring and Ranking Outliers

```
Input sample size $sample$, subspace size $k$, parameter $Q$
Output set of points $S$ sorted (ranked) by outlier score

$\forall a \in A$ Compute Bin Widths ($Q$)
$\forall x \in D\ \ \forall a \in A$ Count Numbers in Bins
$\forall x \in D\ \ \forall a \in A$ Assign to Bins
for $i \leftarrow 1$ to $sample$ do
  Choose Random set of $k$ attributes $s$
  $c \leftarrow 0$
  for all $x \in D,\ x.clustered = 0$ do
    $c \leftarrow c + 1$
    $x.clustered \leftarrow c$
    Collect vector $v$ of all unclustered points that are neighbours of $x$ or neighbours of members
    of $v\ \ \forall a \in s$ {Collect recursively until no additions can be made. Neighbour defined in text}
    if $v = \emptyset$ then
      $x.clustered \leftarrow 0$
    else
      $\forall q \in v,\ q.clustered \leftarrow c$
    end if
  end for
  for all $x \in D$ do
    if $x.clustered > 0$ then
      $x.clustered \leftarrow 0,\ x.score \leftarrow x.score + 1$
    end if
  end for
  $\forall x \in D,\ \ S \leftarrow x$
end for
Sort $S$ by $score$
return $S$
```

Algorithm 5 *Theta()* – Gets Ranked Outlier List and Computes θ

```
Input sample size $sample$, subspace size $k$, points to rank $n$, parameter $Q$
Output $\theta$ and set of points $S$ sorted (ranked) by outlier score

$S \leftarrow RankedOutliers(sample, k, Q)$ {Algorithm 4}
$\theta \leftarrow$ Number of Points with Score Equal To $n$ in $S$
return $\{\theta, S\}$
```

Algorithm 6 *OptimisedRanking()* – Automates Parameter Setting

```
Input maximum acceptable value of $\theta$ target {normally 1}
Input sample size $sample$, points to rank $n$
Output optimised ranked outlier list $S$

$k \leftarrow 3$
$Q \leftarrow 5$
$maxQ \leftarrow n/4$
$stepQ \leftarrow 10$
$last\theta \leftarrow n$
$\{\theta, S\} \leftarrow Theta(sample, k, Q)$
while ($\theta > target$ or $\theta < last\theta$ or $Q < maxQ$) do
  if ($\theta \leq target$) then
    return S
  end if
  if ($\theta \geq last\theta$) then
    {Effectiveness declining so try next $k$}
    $k \leftarrow k + 1$
    $Q \leftarrow Q - stepQ$ {Only need to back up 1 step}
    $last\theta \leftarrow n$
  else
    {Try next $Q$}
    $Q \leftarrow Q + stepQ$
    $last\theta \leftarrow \theta$
  end if
  $\{\theta, S\} \leftarrow Theta(sample, k, Q)$
end while
return $S$
```

4.5.3 Creating Clusters in Subspaces

After collecting a vector v of the nearest neighbours of a point $p \in P$, each member can be assigned a cluster number c. Then all the neighbours of each point $u \in v$, that have not been given a cluster number, are collected and also assigned c. This continues through a recursive process until no more points can be assigned. Since a point is only assigned a cluster number once, the process is $O(|D|)$ (Algorithm 4).

4.5.4 Outlier Detection

All points not assigned a cluster number, due to not having any neighbours or being in a very small cluster, are flagged as outliers in the given subspace (Algorithm 4). We have consistently used 1% of the dataset as the minimum cluster size as this proved robust (see Section 4.5.12).

4.5.5 Outlier Ranking Using Subspaces

Given a subspace size $k = |S|$, all possible subspaces or a sample of them are clustered and the number of times each point is an outlier is tallied. This becomes the outlier score for each point. These scores are then sorted and thus the points are ranked for outlier tendency or 'outlierness' (Algorithm 4).

4.5.6 Subspace Sampling

Enumerating all possible subspaces, rapidly becomes intractable (even with potential parallelisation of the algorithm). Thus, we experimented with a random sample from all the possible subspaces of a particular size k. Using sampling provides a large benefit in efficiency (see Section 4.7.4).

4.5.7 Semi-Supervised Class Assignment

If we have a ranking in which the classes are in different regions of the list and we have some labels, it is possible to attempt to classify unlabelled points based on their proximity to labelled data. No training is implied here but having got a final ranking, the class of each point is predicted based on the votes of the nearest labelled points. The class with the most votes is assigned. Ties are broken arbitrarily. In our implementation, classes are assigned a number as they are seen and lower numbers are preferred.

We tested a voting scheme where the nearest z labelled points voted for the class of unlabelled point x. Varying z from 2 to 10 had very little effect on the results on all test datasets except where very small classes were concerned where a smaller number is naturally more useful. Thus $z = 2$ was adopted.

4.5.8 Large Datasets and a Real Valued Measure

The binary 'clusterd/not clustered' measure for outlier scoring has two key advantages. It is easy to motivate on the basis of Equation 4.2 and Theorem 1. Simply put, if a point is not clustered it is likely in the tail of whatever distribution it might belong to and we do not, therefore, have to determine that membership. If we proposed a real-valued measure based on the distance from its distribution mean (for example) and there were several clusters found, it would be very difficult to justify which cluster to take. Selecting the nearest would obviously be unsafe as normal and many other distributions have infinite tails. Secondly, the binary approach provides an effective way of automatically setting the key parameters.

On the other hand, the binary approach has certain disadvantages. If the dataset is very large and thus, for efficiency, the ensemble size $m \ll |D|$, θ, the number of coequal points can become large. This has implications for finding class boundaries and the semi-supervised assignment of class labels (Section 4.5.7) may be influenced by artifacts in the presentation order of the data. One way of handling the later is to randomly sort the data prior to scoring and the semi-supervised approach can be applied to the sorted but unscored data to provide a base line, which amounts to guessing based on the label frequency in the training data. Both these approaches are applied in Section 4.7.7. However, while these two strategies can give us confidence in the results, they do not resolve the main issues. It would, therefore, be advantageous to find a real-valued measure that satisfactorily reproduces the characteristics of the binary measure ϕ.

The binary measure's differentiation of points is related to the frequency of points similarly scored on the subspace. That is, if a point is not clustered but most others are, then its score will have more weight on the final result than if many points are not clustered. To capture this, a measure ϕ_F is defined that is the normalised inverse product of the entropy of the point and the entropy of the subspace. That is, for a point i in cluster C_j where $p_i = p_{C_j} = \frac{|C_j|}{|D|}$

$$\phi_F = \frac{log(\frac{1}{|D|})^2}{|D| p_i log(p_i) \sum_{C_k \in C} p_k log(p_k)} \tag{4.6}$$

Thus ϕ_F decreases both due to the point being in a larger cluster (less likelihood of being an outlier) and to the higher entropy of the subspace (many outliers). ϕ_F provides a strongly non-linear bias to exceptional outliers. $\phi_F \exists [1, \infty]$ but the asymptote occurs when all points are in a single cluster and that is not scored. This modification of FASTOUT is referred to as FASTOUT-R.

Since ϕ_F gives almost identical results on the synthetic data to ϕ, fixing k and Q can be done by using the binary approach with a small ensemble (for efficiency) and choosing the k and Q that give a θ minimum to use for scoring the data with ϕ_F.

All the results are for the binary approach except in Section 4.6 and Section 4.7.7 as the purpose is to demonstrate the phenomenon of class separation in the most straightforward way.

4.5.9 Categorical Attributes

Large datasets typically contain a mixture of categorical and real valued/multinomial data. Categorical values such as 'http' and 'ftp' have no quantitative relationship. FASTOUT handles these automatically as indicated in the Definition section (4.5.1).

With some categorical attributes and large $|D|$ it could occur that a point exists in very dense bins on all attributes of a subspace. This results in a large cluster being discovered and is an unnecessary cost for outlier detection. Therefore a heuristic was added that if the least dense bin inhabited by a point is larger than $\alpha|D|$, the point is not classed as an outlier but placed in a cluster with all other such points. This heuristic is purely optional and improves efficiency without any significant effect on accuracy. This was only used on the KDD Cup 99 data with $\alpha = 0.01$ (Section 4.7.7).

4.5.10 Validation

In order to validate the outlier ranking, it is necessary to understand the mechanics that cause outliers in high dimensional spaces. Due to the many attributes, each point has multiple chances to appear remote in an attribute. Thus, as discussed above, if we score all points in a dataset for a given subspace size we can expect that if the points inherently belong to several sources or clusters with different variances, the points will separate in their outlier scores according to which cluster they belong since the tendency to outlierness will be a function of the variance. Thus, while we can argue that the low dimensional concept of an outlier as a point with some special uniqueness becomes increasingly inapplicable with increasing subspace size, we can use outlier scores to distinguish cluster membership and we can use the ability to separate the members of different classes to verify the effectiveness of the outlier ranking.

For example, in medical data we will expect that disease conditions will produce more extreme values on the measured attributes than normal cases. Thus, after scoring the samples, the diseased class should appear predominantly in the higher ranking, i.e. have higher outlier scores. Given labelled data with, say, two classes $\{B, C\}$ where B is expected to yield more outliers, we define the effectiveness as the proportion of members of B that appear in the top $|B|$ places in the ranking.

4.5.11 Application to Classification

This separating effect will apply even if the clusters are entirely overlapping spatially (e.g. Figure 4.6 right). This is because the separation depends only on the variance, not the position of the clusters. Thus, outlier scoring leads to an entirely novel means for separating clusters in situations where spatial clustering is essentially impossible. Since many real world situations involve heavily overlapping clusters, this approach could be of important practical utility.

4.5.12 Optimising Parameter Setting

Class separation depends on the members of different classes occupying different regions in the outlier ranking and knowing the ranks at which the predominance of one class gives way to the next. Later we will address how these 'cut' points may be determined. At this stage, let knowledge of these points be assumed. If there are 2 classes, let the cut point be at rank r. If the point at r has score ϕ_r, let θ be the number of points with identical score ϕ_r. θ is determined in Algorithm 5.

FASTOUT has a number of parameters. The key sensitive parameters are subspace size k and bin width Q. For FASTOUT, these are set automatically by Algorithm 6 based on an objective. This objective is based on the observation that high accuracy coincides with small θ. For example, Tables 4.6 and 4.7 show how values of $\theta \simeq 1$ coincide with the highest accuracy. A minor exception is for $k = 2$ which tends to produce lower accuracies than $k \geq 3$, likely due to the smaller sample size. Algorithm 6 scans across a range of values of Q for increasing k until $\theta = 1$ or θ values for the current k are higher than for $k - 1$ indicating the optimum has already been passed. Normally scans start at $k = 3, Q = 5$.

We also tested the sensitivity to our choice of 1% of the data size for the minimum cluster size (ρ, see Definition 9). On synthetic and real-world data this typically worked well. In some cases in the real-world data, 1% of the data was in low single digits and in these cases a slightly larger value (≈ 10) produced better results. In Algorithm 6 a stopping condition $maxQ$ is defined but this limit was never reached in our experiments. It simply reflects the fact that if the number of bins becomes very small, outlier detection is moot.

4.6 Experimental Results for FASTOUT, T*ROF and T*ENT on Type 2 Data.

The new and comparison algorithms were tested on all UCI Repository [33] binary problem datasets, with 20 or more attributes, which they would all reasonably be expected to complete. This meant requiring non-categorical data without missing values or a very large number of attributes (≥ 500). While these limitations could be reduced by modifying certain of the algorithms, they serve as a useful set of conditions to define a group of datasets that is both broad in scope and free of any question of experimenter bias. Out of the 12 datasets that meet these criteria, two were left out each being one of a pair of datasets based on the same data. For example, the SPECTF data with the highest number of attributes was chosen. The 'Hill-Valley' dataset was omitted since the task should not be amenable to this approach. Whether a sequence has a peak or a trough does not imply any likely difference in variance between the classes. On the other datasets, it is reasonably possible that the two classes may have different variances due to some meaningful tendency in the data.

Table 4.2 gives an overview of these datasets showing that they represent various class balances including a preponderance of the class B expected to be more often contributing outliers. Hardly

Table 4.2: Class sizes, dataset size and number of attributes.

	$\lvert D\rvert$	d	Target class	Other class
Ionosphere	351	32	126	225
SPECTF	267	44	212	55
Sonar	208	60	111	97
WDBC	569	30	212	357
Parkinsons	195	22	148	48
Spambase	4601	57	1813	2788
Creditcard	1000	24	700	300
Musk	476	166	207	269
Insurance	5822	85	348	5474

Table 4.3: Area under the curve (AUC) and percent accuracy on the target class for various UCI datasets. †Could not compute covariance matrix.

	Ionosphere		**SPECTF**		**Sonar**		**WDBC**		**Parkinsons**	
	AUC	%	AUC	%	AUC	%	AUC	%	AUC	%
FASTOUT	0.8400	85.78	0.7816	87.74	0.6449	62.16	**0.9578**	**87.74**	**0.7514**	**85.71**
FASTOUT-R	0.8297	85.33	0.7915	87.74	0.5894	60.36	0.9211	82.08	0.6835	82.99
T*ROF	0.8915	74.60	0.8849	90.57	**0.9255**	**83.51**	0.9574	**87.74**	0.7405	83.67
T*ENT	0.8429	84.89	**0.8959**	**92.45**	0.6235	61.26	0.8474	85.85	0.5319	80.27
SOE1	0.7691	80.00	0.7642	84.43	0.601	56.76	0.9203	80.66	0.7456	80.95
FBAG	0.6048	69.78	0.436	78.30	0.5018	54.05	0.5161	39.63	0.4239	72.11
LOF	0.5836	66.22	0.4154	77.83	0.4954	54.96	0.4699	35.85	0.4386	72.79
ORCA	0.4575	60.89	0.5569	80.19	0.5056	49.48	0.5091	39.62	0.5169	76.19
RMD	**0.9479**	**87.40**	0.7527	84.36	0.5857	62.73	0.9131	77.25	†	
T*LOF	0.4297	63.56	0.3753	75.94	0.4836	52.25	0.4834	38.21	0.4249	72.79

any would meet the usual 'rare' class criterion. For example, the WDBC set consists of 569 samples labelled in two classes, 212 malignant and 357 benign with 30 attributes for each sample. In this dataset as in many medical scenarios, normal biopsies tend to have quite strongly clustered results while abnormal ones exhibit a higher variance.

The results on the UCI datasets are summarised in Tables 4.3 and 4.4. Our results show that overall FASTOUT, T*ROF and T*ENT were the most effective though RMD and SOE1 performed well. RMD is also hampered by high run time and SOE1 by parameter sensitivity. The most effective approaches also require minimal parameter setting. T*ROF is parameter free and FASTOUT automatically adjusts its parameters. These therefore have a clear edge as effective and user friendly approaches.

An extensive, but not necessarily exhaustive, literature search for the best results on these datasets using supervised classifiers yielded the following:

Ionosphere: Wang [162] using a range of kNN classifiers and 10-fold cross-validation (MCV) achieved an accuracy of 87.8%. A Support Vector Machine (SVM) ensemble tested with 10-fold MCV achieved 95.20% [103].

Table 4.4: Area under the curve (AUC) and percent accuracy on the target class for various UCI datasets. †Could not compute covariance matrix. *Exceeded time limit.

	Spambase		**Credit Card**		**Musk**		**Insurance**	
	AUC	%	AUC	%	AUC	%	AUC	%
FASTOUT	0.7294	59.96	0.4675	69.29	0.6911	**72.12**	0.5363	8.05
FASTOUT-R	0.7255	72.74	0.5274	72.00	**0.7736**	69.57	0.4763	8.33
T*ROF	**0.9077**	**79.15**	0.3237	63.14	0.7055	58.45	0.5213	4.31
T*ENT	0.7278	78.26	**0.7689**	**89.14**	0.3297	28.99	0.5386	10.06
SOE1	0.7078	58.36	0.4422	69.14	0.3359	29.95	**0.5489**	**10.63**
FBAG	0.4742	37.51	0.5201	68.57	0.4967	43.48	0.4999	8.33
LOF	0.4668	36.62	0.494	69.14	0.4906	43.00	0.4994	6.61
ORCA	0.4909	38.44	0.5551	71.71	0.5084	46.38	0.5359	8.62
RMD	†		†		†		†	
T*LOF	0.4825	39.33	0.5168	71.57	0.4958	42.51	*	

SPECTF (Cardiac): Previously, the CLIP3 algorithm was used to generate classification rules that were 81.34% accurate (as compared with cardiologists' diagnoses) [113]. Ali et al. [13] report that the best Linear Dimensionality (LD) reduction Quadratic classifier had an error of 4.44% while the LD Linear classifier studied had an error rate of 17.64%.

Sonar: The Sonar dataset is known to be difficult to separate. Harmeling et al. [82] tested a wide variety of supervised methods including Gaussian mixtures, Support Vector Data Description (SVDD), Parzen, a k-means based approach, and several k-NN methods. Their concept was using various local density based measures to rank the points for their degree of being typical, as an outlier method does. Results varied from AUC 0.596 (SVDD) to 0.870 (new γ kNN method).

WDBC: Jiang and Zhou used a neural network to edit the training data for kNN classifiers [97]. With a minimum of 250 points used as training data, 100% separation of the classes was been achieved. With 200 training points they achieved 96% accuracy. Ali et al. [13] report that the best Linear Dimensionality (LD) reduction Quadratic classifier had an error of 4.02% while the LD Linear classifier studied had an error rate of 2.99%.

Parkinsons: No comparable supervised classification results were located in our literature search.

Spambase: Neural Expert networks have been reported with an accuracy of 85% [149] using 10 fold MCV.

Credit Card: This is the Statlog (German Credit Card) dataset. Eggermont et al. used Genetic Programming and compared with C4.5 with Bagging and Boosting and other algorithms. The best result was a 27.1% misclassification rate [59].

Musk: This is a highly studied dataset. For example, Zafra and Venture report accuracy with SVMs of 87% and 93% with a genetic algorithm [181]. This dataset is only nominally two-class as both musks and non-musks are made up of multiple classes and thus is possibly not suitable for the outlier approach.

Insurance: This was part of the COIL 2002 competition and proved difficult for all algorithms.

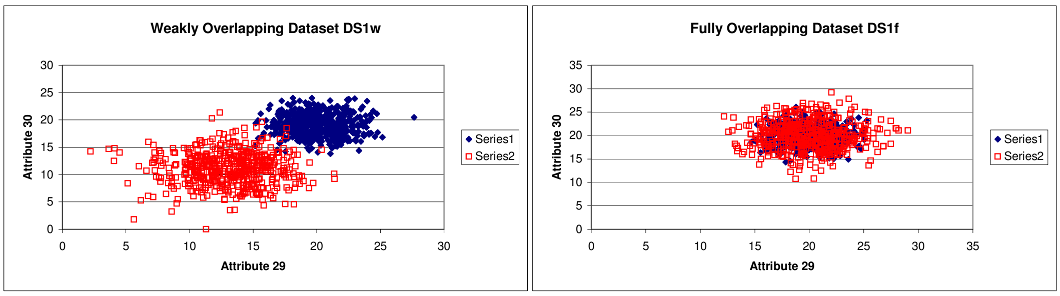

Figure 4.6: Example plots of the weakly and fully overlapping datasets DS1w and DS1f on two typical attributes.

Caravan ownership is to be predicted based on other factors in an insurance application. The best results were only a few percentage points better than the best in Table 4.4 (See e.g. [180]).

The very best supervised classifiers outperform the outlier methods on most datasets tested but the margins are generally small. Remarkably, one of our unsupervised approaches achieved the best result on two datasets – Sonar (T*ROF) and Credit Card (T*ENT). FASTOUT, FASTOUT-R, T*ENT and T*ROF are remarkably competitive considering that they simply exploit the difference in variance between the classes and operate unsupervised. Obviously, different datasets are suited to different algorithms but overall FASTOUT and T*ROF put in the best performance with respect to effectiveness. FASTOUT-R is significantly more efficient, requiring 10% or less subspace sample size to produce similar results to FASTOUT which is, itself, substantially more efficient than the T* methods that fully enumerate the 2D subspaces.

4.7 Exploring Class Separation with FASTOUT

In order to elicit the understanding of the concept of class separation through variance, it is advantageous to investigate synthetic datasets with known characteristics. After that, we look at the scheme that classifies using the outlier ranking and a small proportion of labelled data . These experiments were performed using a 2.6GHz AMD processor and 2G of RAM.

4.7.1 Synthetic Datasets

We hypothesised that if a dataset contained two clusters with different variances, then the two groups of points can be separated because the points with the greater variance will be outliers more often. If the variances are the same, then this should fail. This is illustrated in Figure 4.3. While the outliers are all scored correctly, only in the case where the underlying classes differ in variance do their members' scores inhabit (largely) different regions of the outlier ranking. We investigated this effect with synthetic datasets with weakly spatially overlapping clusters, labelled 'w', and then with fully overlapping clusters, labelled 'f'. E.g. DS1f means fully overlapping classes with standard deviation differing by 1. Figure 4.6 illustrates these two cases in a sample 2D space.

Table 4.5: Synthetic datasets with d identical attributes and their classes in the form $\{N, \mu, \sigma\}$ ($size$, $mean$, $standard\ deviation$); r_i indicates randomly assigned location on each attribute.

Designation	d	Classes
DS0w	30	$\{500, r_1, 2\}, \{500, r_2, 2\}$
DS1w	30	$\{500, r_1, 2\}, \{500, r_2, 3\}$
DS2w	30	$\{500, r_1, 2\}, \{500, r_2, 4\}$
DS1f	30	$\{500, 0, 2\}, \{500, 0, 3\}$

Table 4.6: Effectiveness in separating two clusters in synthetic datasets. $Q = 35$, Sample $= 2000$. Accuracy values above and θ values below (see Section 4.5.12). *All points are equally outliers.

k	1	2	3	4	5
DS0w	49.8%	50.0%	48.6%	*	*
DS1w	58.4%	93.0%	100%	*	*
DS2w	59.4%	97.8%	100%	95.8%	*
θ, DS0w	894	12	4	1000	1000
θ, DS1w	672	2	1	1000	1000
θ, DS2w	760	4	1	523	1000

Table 4.5 lists the characteristics of the data sets. All the datasets contain classes with Gaussian distributions over all attributes differing only in class size, variance and in some cases class mean μ.

Two scoring approaches are taken. Firstly, for binary class data the area under the ROC curve (AUC) is computed. Secondly, if the class C with the highest variance has size n, then the percentage of members of C in the top n ranked outliers is computed. this is referred to as the 'accuracy'. These two measures tend to be closely related so both values are not always given in the results.

Table 4.6 shows the results for the weakly overlapping datasets. This shows very clearly that this approach can separate classes based on a variance difference. Even with classes $\sigma = 2$ vs. $\sigma = 3$, 100% of the class members appeared in distinct regions of the outlier ranking. On the other hand, where there was no difference in variance (Dataset DS0w), no separation took place. This is as postulated in the theoretical discussion (Section 4.1.1).

A more challenging case is when the classes are concentric as we might expect the central core

Table 4.7: Effectiveness on the fully overlapping dataset DS1f, $d = 30$ for various Q and sample size 2000. Accuracy values across subspace sizes k above and θ values below. *All points are equally outliers.

k	1	2	3	4	5
Q=35	68.4%	93.2%	76.4%	*	*
Q=70	58.0%	88.8%	92.6%	85.0%	*
Q=100	53.4%	89.4%	94.0%	91.0%	85.0%
$\theta, Q = 35$	709	3	75	1000	1000
$\theta, Q = 70$	883	13	1	14	1000
$\theta, Q = 100$	924	33	1	2	28

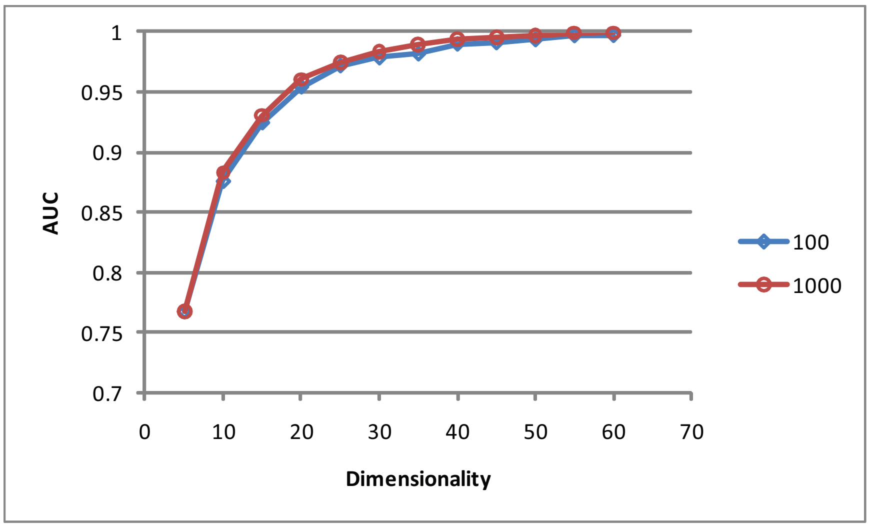

Figure 4.7: AUC at different dimensionalities for DS1f type datasets at two different sampling sizes.

Table 4.8: Synthetic datasets with identical attributes A and their classes in the form $\{N, \mu, \sigma\}$ ($size$, $mean$, $standard\ deviation$); r_i indicates randomly assigned location on each attribute.

Dataset	d	Classes
DS3f	30, 60, 90	$\{500, 0, 2\}, \{500, 0, 3\}, \{500, 0, 4\}$
DS4f	30	$\{500, 0, 1\}, \{500, 0, 2\}, \{500, 0, 3\}, \{500, 0, 4\}$
DS4Uf	30	$\{500, 0, 1\}, \{100, 0, 2\}, \{500, 0, 3\}, \{100, 0, 4\}$
DS4U2f	30	$\{30, 0, 1\}, \{100, 0, 2\}, \{300, 0, 3\}, \{500, 0, 4\}$
DS4U3f	30	$\{15, 0, 1\}, \{30, 0, 2\}, \{30, 0, 3\}, \{500, 0, 4\}$

of the cluster to contain indistinguishable members of both classes. Table 4.7 shows that, with the right parameter setting, a 94.0% separation occurs for DS1f. Figure 4.7 shows that over a range of dimensions from 5 to 60, AUC values quickly rise to better than 0.99 (at $d \geq 40$). Thus, given sufficient dimensions, the concentric case is only slightly more difficult than the weakly overlapping case.

4.7.2 Multiple Class Separation

Having confirmed that both non-concentric and concentric classes can be separated and that separating concentric classes is a more difficult task, we now consider only such classes. So far only balanced classes were analysed. Table 4.8 shows the characteristics of a number of synthetic datasets with multiple classes and both balanced and unbalanced class sizes.

Tables 4.9, 4.10 and 4.11 show the effectiveness of FASTOUT for three class synthetic data DS3f $\{C_1, C_2, C_3\}$ at three different dimensionalities. There is a clear improvement in separation with increasing dimensionality. Tables 4.12, 4.13, 4.14 and 4.15 show that this effectiveness of separation extends to 4 classes even if the classes are unbalanced (Datasets DS4f, DS4Uf, DS4U2f and DS4U3f). This data shows that large low variance classes separate well as do high small variance

Table 4.9: Separation of 3 fully overlapping clusters of differing variance (DS3f with $d = 30$)

	Class (Deviation σ)		
Ranked Points	4	3	2
Top 500	85.4%	14.6%	0%
Mid 500	14.6%	79.4%	6.0%
Bottom 500	0%	6.0%	94.0%

Table 4.10: Separation of 3 fully overlapping clusters of differing variance (DS3f with $d = 60$)

	Class (Deviation σ)		
Ranked Points	4	3	2
Top 500	94.0%	6.0%	0%
Mid 500	6.0%	92.2%	1.8%
Bottom 500	0%	1.8%	98.2%

Table 4.11: Separation of 3 fully overlapping clusters of differing variance (DS3f with $d = 90$)

	Class (Deviation σ)		
Ranked Points	4	3	2
Top 500	96.8%	3.2%	0%
Mid 500	3.2%	96.6%	0.2%
Bottom 500	0%	0.2%	99.8%

Table 4.12: Separation of 4 fully overlapping clusters of differing variance (DS4f).

	Class (Deviation σ)			
Ranked Points	4	3	2	1
Top 500	77.8%	22.2%	0%	0%
Next 500	22.2%	70.4%	7.4%	0%
Next 500	0%	7.4%	92.6%	0%
Bottom 500	0%	0%	0%	100.0%

Table 4.13: Separation of 4 fully overlapping unbalanced clusters of differing variance (DS4Uf).

	Class (Deviation σ)			
Ranked Points	4	3	2	1
Top 100	60.0%	8.0%	0%	0%
Next 500	40.0%	89.8%	11.0%	0%
Next 100	0%	2.2%	89.0%	0%
Bottom 500	0%	0%	0%	100.0%

Table 4.14: Separation of 4 fully overlapping unbalanced clusters of differing variance (DS4U2f).

	Class (Deviation σ)			
Ranked Points	4	3	2	1
Top 30	60.0%	12.0%	0%	0.2%
Next 100	40.0%	71.0%	5.7%	0%
Next 300	0%	17.0%	94.0%	0.2%
Bottom 500	0%	0%	0.3%	99.8%

Table 4.15: Separation of 3 small and one larger fully overlapping clusters of differing variance (DS4U3f).

	Class (Deviation σ)			
Ranked Points	4	3	2	1
Top 15	93.3%	3.3%	0%	0%
Next 30	6.7%	96.7%	0%	0%
Next 30	0%	0%	100%	0%
Bottom 500	0%	0%	0%	100%

classes relative to them (e.g. Table 4.15). However, larger higher variance classes can partially overlap with smaller classes of similar variance. For example, $\sigma = 3$ and $\sigma = 4$ classes separate with markedly lower effectiveness than $\sigma = 2$ and $\sigma = 4$ classes. It is clear that the greater the difference in variance as well as the larger the dimensionality, the greater the effectiveness of separation. This is well illustrated in Figure 4.7 and Tables 4.9, 4.10 and 4.11.

4.7.3 Class Variance Estimation

Consider the 3 class synthetic data presented above (DS3f). This dataset consists of 3 normal distributions $\{C_1, C_2, C_3\} = \{\{0, 2\}, \{0, 3\}, \{0, 4\}\}$ where each Gaussian is defined by its mean and standard deviation $\{\mu, \sigma\}$. Figure 4.8 shows 3 fully overlapping Gaussians. The cumulative probability distribution (Φ) for a Gaussian evaluated up to (cluster boundary) X_B is

$$\Phi(X_B) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{X_B} e^{\frac{-(u-\mu)^2}{2\sigma^2}} du \tag{4.7}$$

The binary valued process of outlier detection first clusters and then assigns an outlier score of 1 to points falling outside the cluster. Earlier (Section 4.1.1) it was argued that, summing over multiple subspaces, is sufficient to permit an estimation of the probability of a point being an outlier. If this is so, then the cumulated outlier score of a class should approximate the cumulative probability distribution (Equation 4.7) with respect to the boundary of the cluster $\pm X_B$ That is for the points $x_j,\ 1 \leq j \leq N_i$ in $C_i = \{N_i, \mu_i, \sigma_i\}$

$$\frac{1}{N_i} \sum_{C_i} \phi(x_j) \simeq \alpha\ \Phi(\mu_i, \sigma_i, X_B) \tag{4.8}$$

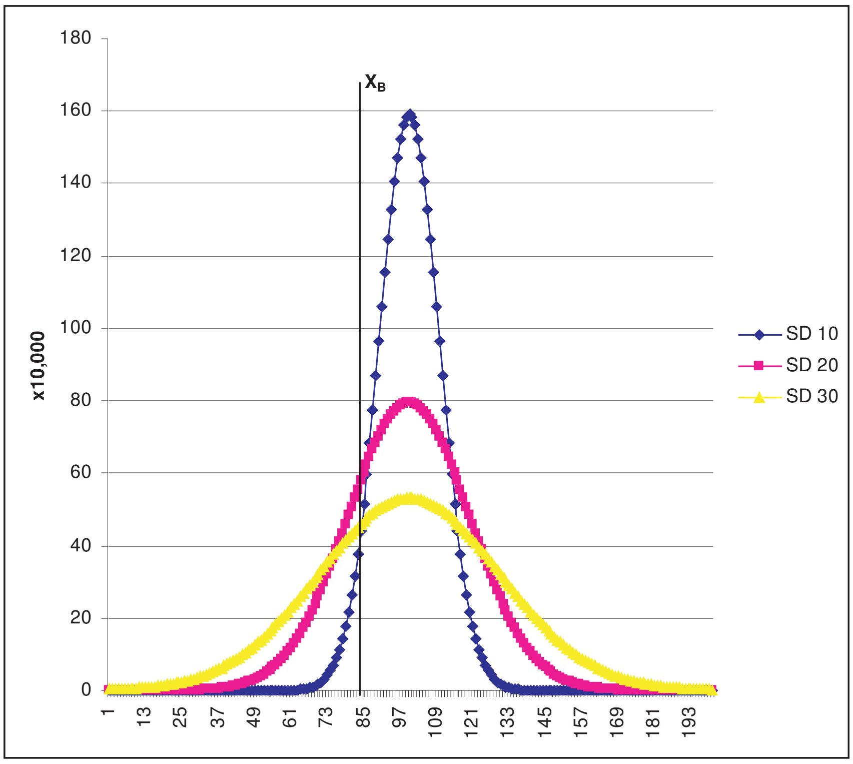

Figure 4.8: Three Gaussians with equal mean and differing variance and cluster boundary X_B.

where α is a constant of proportionality for constant subspace size.

If we set the approximate boundary of the clustering of DS3f, say, at $X_B = 5.1$, then the relative proportions of $\Phi(X_B)$ for the three classes is 3.44%, 29.32%, and 67.24%. If we total the outlier scores for the three classes in the DS3f synthetic dataset, their relative proportions are 2.29%, 30.10% and 67.60%, which is closely similar, given stochastic effects in the randomly generated synthetic dataset and approximations in the numerical integration (Table 4.16). Dataset DS4f yielded a similar result (Table 4.16) as also did the unbalanced datasets, for example DS4Uf (Table 4.17). Thus the binary-valued outlier measure of FASTOUT provides a very good approximation to the probability distribution function for normal distributions.

From this we could estimate the σ_i if we assume normal distributions and can estimate X_B relative to μ. Φ for a Gaussian has no closed form solution, so numerical integration is used to approximate it. The approach we take is an iterative approach where a series of solutions are computed within a reasonable range of the parameters and the best selected and then refined by computing within an increasingly tight region. The ratios of the L.H.S. of Equation 4.8 computed for the discovered classes is used as the objective and ratios of the R.H.S. computed for various values of σ and X_B with progressive refinement to give a fit within some ϵ. Thus α drops out.

Excellent solutions already exist for separating classes with well separated means. The approach described here is for the difficult case of heavily overlapping classes and this permits us to assume

Table 4.16: Comparison of $\Sigma\phi$ to $\Phi(X_B)$ for two synthetic datasets. $\Sigma\phi$ for actual classes.

DS3f				
Class (σ)		2	3	4
$\Sigma\phi$ $\Phi(X_B = 5.1)$		2.29% 3.44%	30.10% 29.32%	67.60% 67.24%
DS4f				
Class (σ)	1	2	3	4
$\Sigma\phi$ $\Phi(X_B = 1.6)$	1.72% 5.73%	24.13% 23.23%	35.36% 32.83%	38.79% 38.21%

Table 4.17: Comparison of $\Sigma\phi$ to $\Phi(X_B)$ for unbalanced class dataset DS4U2f with correction for dataset size N as per Equation 4.8.

Class (σ)	1	2	3	4
N	500	300	100	30
$(\Sigma\phi)/N$ $\Phi(X_B = 1.2)$	6.56% 9.95%	27.24% 24.56%	32.49% 31.03%	33.71% 34.48%

the means are approximately identical. Therefore, for spherically symmetric distributions, we can assume an approximately spherical form of the clustering which can be captured by its radius X_B, rather than some complex shape. This assumption is only relevant if we try to reverse engineer the variances from the outlier scores.

4.7.4 Sampling

Full enumeration (FE) of all subspaces is intractable for larger k. Therefore we investigated sampling. If the sample size is S' then the number of subspaces summed for the outlier measure $S^* = min(S', FE)$. Sampling was tested on DS1f over $d = 5$ to 60 for $S' = \{100, 1000\}$. Figure 4.7 shows the results. In our other experiments a similar pattern emerged with only a small effectiveness gain for larger samples or full enumeration.

4.7.5 Scaling

Synthetic datasets are often used for demonstrating scaling. The advantage is that the structure of the data is known but the disadvantage is that the simple type of datasets often used may not adequately test the algorithms. Purely for the purpose of this scaling experiment, the WDBC dataset was used as a seed to create datasets $10\times$, $50\times$ and $100\times$ the size. This was done by creating multiple points from each point adding a random amount of jitter to each, not exceeding 1%, thus preserving the core characteristics of the dataset. The class labels cannot be applied safely to these larger datasets so no investigation was made into effectiveness. Similarly, to test scaling with respect to the number of attributes, the WDBC data was used as a seed with up to 10% jitter to minimize correlation effects.

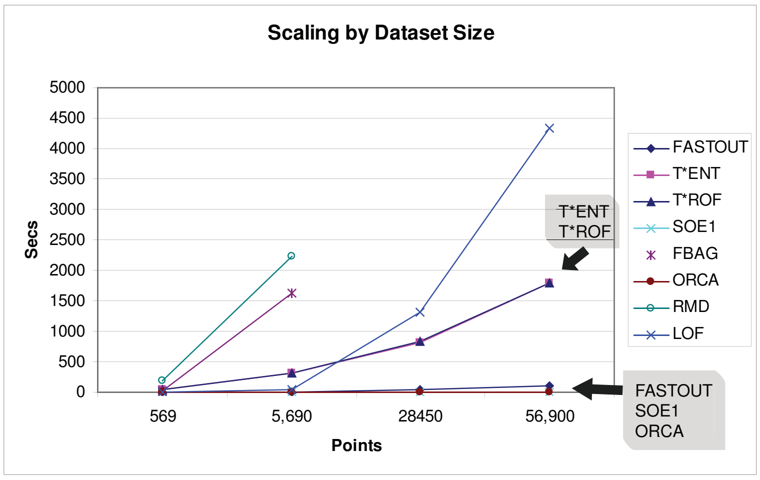

Figure 4.9: Scaling by dataset size for comparison outlier algorithms.

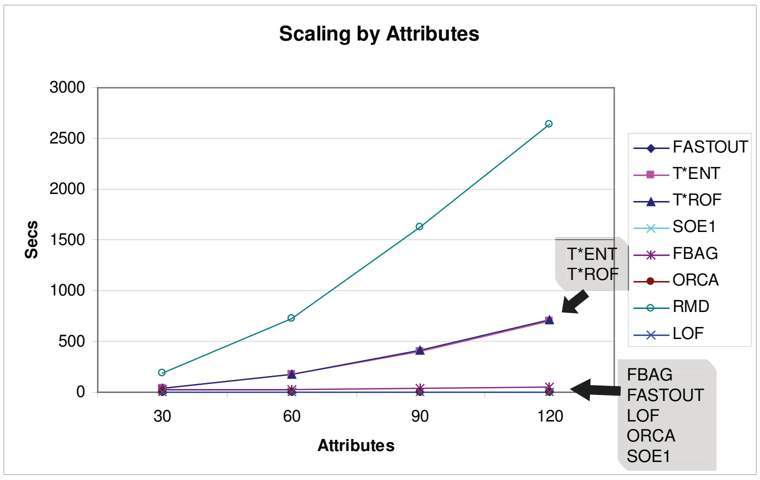

Figure 4.10: Scaling by dimensionality for comparison outlier algorithms.

Table 4.18: Run time: scaling by dataset size for various algorithms (30 attributes) and scaling by number of attributes (569 data points). †Could not complete.

Algorithm	Dataset Size				# Attributes			
	569	5,690	28450	56,900	30	60	90	120
FASTOUT	0.7	7.2	43.5	96.7	1.0	1.9	2.0	2.1
T*ENT	43.4	317.4	821.0	1782.2	43.4	174.8	405.0	704.5
T*ROF	43.6	321.0	828.4	1798.3	43.6	178.3	411.5	717.0
SOE1	0.0	0.1	0.7	1.5	0.0	0.0	0.0	0.1
FBAG	20.4	1620.5	†	†	20.4	30.2	42.9	53.8
LOF	0.5	44.0	1309.7	4331.9	0.5	0.8	1.2	1.5
ORCA	0.6	0.8	1.0	1.2	0.6	0.8	1.2	1.2
RMD	188.5	2225.1	†	†	188.5	729.9	1619.1	2635.2
T*LOF	77.7	5907.9	†	†	77.7	393.7	1178.4	1990.8

Figure 4.9 shows run time over increasing dataset sizes and Figure 4.10 shows run time over increasing dimensionality. The data can be found in Table 4.18. RMD, Frequent Bagging (FB) and T*LOF could not complete the larger datasets in a reasonable time. This is because RMD is cubic in complexity due to the matrix manipulation and FB has a run-time of about 50 x LOF which is itself quadratic for higher dimensionality where tree indices are not helpful. T*LOF is $\binom{d}{k} \times$ LOF and run times are not shown in the Figures but appear in Table 4.18. While T*ENT and T*ROF scale reasonably, FASTOUT, SOE1 and ORCA are highly efficient. The efficiency of FASTOUT reflects both the efficiency of the algorithm and the use of sampling. FASTOUT-R results are not given because the core algorithm has the same efficiency as FASTOUT but is usually run at a smaller sample size so the net efficiency is even greater by the relative sample size. The γ efficiency heuristic was not employed in any of these experiments. If applied, it further enhances the efficiency of the FASTOUT and FASTOUT-R algorithms.

4.7.6 Automatic Class Separation

In Section 4.1.2 it was theoretically anticipated that the shape of the outlier ranking graph would exhibit an 'S' shape for each class of different variance as long as the class had a smooth probability distribution function. This is clearly visible in Figure 4.11 especially for the highest variance class in the $d = 30$ case and all the classes where $d = 60$. The higher the class variance, the larger the class mean outlier score and the other scores are concentrated around this value (region R2, Figure 4.11 lower left) with a spreading out at the extreme of each tail (regions R1 and R3). As the relative difference between the variances declines, the overlap between the class outlier score tails increases. While it is clear, in the 30D case, that the SD 2 and SD 4 class members separate well (in fact with 99.4% effectiveness) the SD 3 class overlaps sufficiently so that the upper graph which shows the full rank/score graph does not exhibit any clear 'join' between the classes.

In the 60D case (Figure 4.11), the SD 2 and SD 3 classes separate enough that there is an obvious inflexion in the full rank/score graph between them. A smaller, but still clearly visible, inflexion is

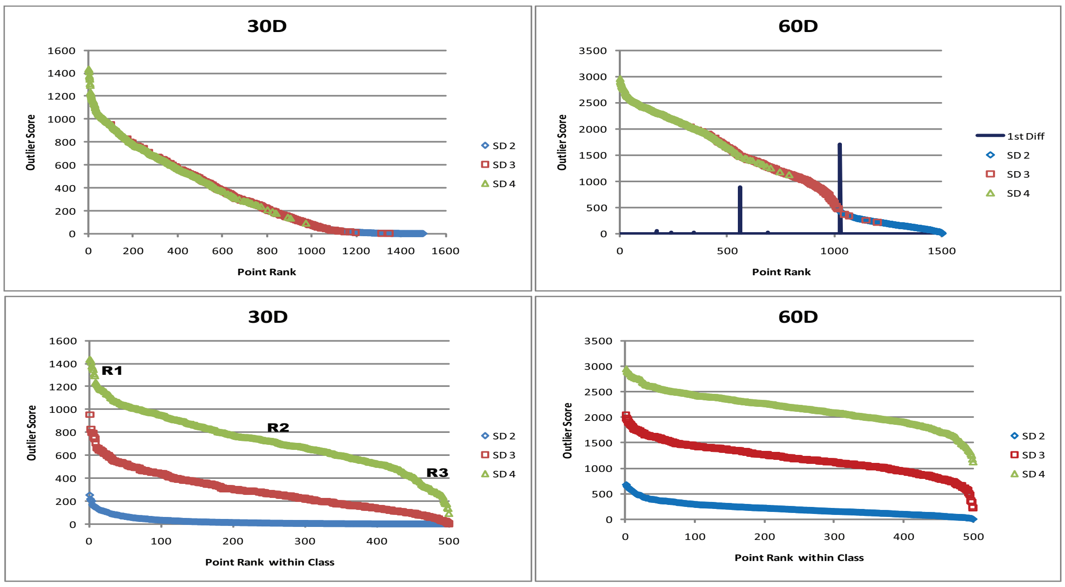

Figure 4.11: Three class synthetic data (DS3f) with $d = 30$ and 60 showing improved separation for higher dimensionality. Complete outlier ranking graph at top, ranking within classes at bottom.

seen between SD 3 and SD 4. In these cases we can locate these ‘joins’ algorithmically. As the graph is not strictly linear but shows some tendency to a steeper slope in the regions where classes overlap, some simple heuristics can be devised for detecting such locations in addition to either visual inspection or some semi-supervised approach.

Let G be the $\{\phi, \text{rank}\}$ graph as in Figure 4.11. The separation or ‘cut’ points we wish to identify are associated with inflection points in G. Finding the inflection points is simple in principle as the sign of the second differential of the series changes at these points. The difficulty lies in the small stochastic variations. Dataset DS3f was generated randomly and any realistic dataset will have some stochastic variations. Thus a moving average ($N/50$ points) was employed to smooth the series and the first differential of that taken. A histogram of the first differential was generated and a low pass filter applied at the most frequent value f. This yields zero values for the most linear regions of G.

From the first differential, the peaks of all the regions of non-zero values are taken. As the moving average introduces a ‘delay’ in the peak, the index of the peak is reduced by $n/100$, i.e. half the window. End-effects will usually occur and are not shown in most example figures. The result on DS3f, amplified ten times for presentation purposes is seen in Figure 4.11 top right. The inflections at 500 and around 1000 are well marked. As each class contains 500 points this is essentially correct. Obviously the difference between the $\sigma = 1$ and $\sigma = 2$ classes will be more pronounced than that between the $\sigma = 2$ and $\sigma = 3$ classes as the difference in the variance (σ^2) is in the ratio of $1 : 0.44$. Looking for the leading peaks in the smoothed totals of the first differential was more useful than looking for a change of sign in the second differential as this occurs at every peak, whatever size, and is thus more prone to stochastic effects.

This approach can be used when a suitable shape is seen in output graph G. Generally, the real world data with which we experimented did not permit sufficient separation to be suited to this. Therefore, in the next section, we propose a semi-supervised approach to extracting the classes from the output.

4.7.7 Semi-Supervised Class Assignment

In this section, the real-valued measure version of FASTOUT is used, FASTOUT-R as well as the binary version. FASTOUT-R was developed to deal with very large datasets as discussed in Section 4.5.8.

In order to test the concept of classifying points based on labelled neighbours in the ranked list, the KDD Cup 99 data was used [6]. This dataset is for detecting computer intrusion. Each tuple in the dataset has 41 attributes, 3 being categorical. The winner's result is reported for the test dataset with 311,029 tuples which has 36 classes. Competitors had access to a large amount of training data though the training data has a different probability distribution from the test data. As FASTOUT/FASTOUT-R does not train, we used only the test data (for which labels are available) and randomly selected 1/3 of this data as available labels, excluding the point to be classified, for the voting scheme after all points were ranked. From these, only the 2 nearest labelled items are actually being used. A baseline was computed by applying the scoring scheme on the randomised data vector prior to outlier scoring by FASTOUT/FASTOUT-R . This quantifies how well one could do by 'guessing' on the basis of the relative label frequencies in the 'training' data.

The choice of 1/3 is conservative but using even lower ratios only resulted in slightly lower effectiveness. The results for various ratios for the best score reported in Table 4.20 were $\{100\% : 0.1128,\ 50\% : 0.1195,\ 33.33\% : 0.1232,\ 25\% : 0.1259,\ 20\% : 0.1255\}$.

The classes were combined as indicated by [6] to yield 5 classes (one normal and the others are different types of attack). Cost per item was then computed according to the cost matrix provided by [6][3]. ACM used this cost to determine the winner. This allows comparison with the quoted Cup results [6] with the proviso just mentioned regarding the source of labels for the two methods. It was not expected that the FASTOUT-R approach, which simply exploits differences in variance between the distributions of different classes, would give similar effectiveness to supervised classification. However, the results are strongly suggestive of its practical utility for such applications (Table 4.19 and 4.20).

The results for both the binary (FASTOUT) and the real-valued measure approach (FASTOUT-R) are given in Table 4.20. The Cup winner data is taken from [6]. The binary scheme achieved a cost between the high accuracy of the real-valued score (FASTOUT-R) and the base line. This was a consequence of the very large number of coequal points due to the small ensemble size relative to $|D|$. Increasing the sample size for FASTOUT (Binary) by $10\times$ lowered the per item cost by 9%.

[3]The cost formula especially punishes class 3 and 4 tuples being classified as normal

Table 4.19: Confusion matrices for FASTOUT and the KDD Cup 99 winner.

FASTOUT (ϕ_R)

Actual	Predicted					
Class	normal	1	2	3	4	% correct
normal	57673	589	2760	26	1980	95.2%
1	1402	2294	409	2	59	55.1%
2	5535	778	223123	3	414	97.1%
3	156	12	19	35	6	15.4%
4	2087	268	5736	11	8087	50.0%
% correct	86.3%	58.2%	96.6%	46.1%	89.0%	

KDD Cup 99 Winner

Actual	Predicted					
Class	normal	1	2	3	4	% correct
normal	60262	243	78	4	6	99.5%
1	511	3471	184	0	0	83.3%
2	5299	1328	223226	0	0	97.1%
3	168	20	0	30	10	13.2%
4	14527	294	0	8	1360	8.4%
% correct	74.6%	64.8%	99.9%	71.4%	98.8%	

Table 4.20: Total cost per transaction for the KDD Cup 99 dataset. Cost is per tuple.

	Baseline	FASTOUT	FASTOUT	FASTOUT-R	**FASTOUT-R**	Winner
Ensemble size	n/a	200	2000	50	200	n/a
Cost/item	0.8854	0.5858	0.5349	0.1551	**0.1232**	0.2331

FASTOUT-R typically runs at smaller sample sizes than the binary-valued measure. Using a sample size of 50 (25% of the lower binary sample) already produces a cost 66.5% of the Cup winner's cost. Increasing the sample size by $4\times$ improved this to 52.9%. These results were obtained without any training. FASTOUT also did better than subsequent attempts at this problem (e.g. [120], cost 0.2445).

4.7.8 A Real-time Intrusion Detection Scheme

Section 4.7.7 shows a good result for detecting computer intrusion but this is most useful if intrusions can be detected in real-time. Computing a representative reference sample (that can fit within main memory) has to be done offline but once computed and the outlier scores for each subspace in the ensemble are held in memory, additional tuples can be processed in O(1) time. Each incoming tuple is checked for its nearest neighbour in each subspace and assigned that score. This is valid because the full process would cluster the two together and assign them the same score. The values for each subspace are summed and then the label is derived from the nearest points on the ranking of the reference sample. In a dual core machine, the second thread could periodically update the reference sample.

The same approach could be used in other scenarios also. This is left for future work as it is not part of the focus of this thesis.

Chapter 5

Conclusion and Future Work

In this thesis, we have presented a number of advances in various fields. In particular, we have demonstrated the utility of low dimensional approaches to high dimensional clustering and outlier detection problems. Our framework MAXCLUS provides an efficient and effective way of discovering candidate subspaces for clustering and then enumerating the clusters therein. The user can specify whether the clustering is 'hard' or 'fuzzy' at at least three levels enabling various types of information to be obtained about the points. With the new classifier GSEP as a preprocessor, very high-dimensional problems such as proteomic and genomic data can be handled resulting in small sets of features (petides, genes, etc.) which effectively distinguish the diagnostic classes.

In the field of outlier detection, several novel algorithms suited to high-dimensional data have been presented (T*ENT, T*ROF, FASTOUT, FASTOUT-R). It has been shown that these algorithms out perform the state-of-the-art in ranking outlierness in many datasets regardless of whether they contain rare classes or not. Previous work focussed solely on such rare classes. Our research into high-dimensional outlier detection has shown that our approach can even lead to a powerful means of classification for heavily overlapping classes given sufficiently high dimensionality and that this phenomenon occurs solely due to the differences in variance among the classes. On some difficult datasets, this unsupervised approach yielded better separation than the very best supervised classifiers and on other data, the results are competitive with state-of-the-art supervised approaches. The elucidation of this novel approach to classification opens a new field in data mining, classification through differences in variance rather than spatial location.

5.1 Future Work

This new view on outlier detection can lead to many avenues of research including those that can not be immediately envisaged. We have shown that one algorithm, FASTOUT-R, beats the KDD Cup '99 winner in computer network intrusion classification and we have suggested a modification that would permit the implementation of this in real-time. Testing of this is left to future work. Many more such applications are clearly possible.

It is also likely that even better versions of the FASTOUT-R heuristic may be forthcoming with further research. Many of our new algorithms are well suited to parallel implementation due to their compartmentalised or ensemble approach. The efficiency improvements from this have also not been tested. In our synthetic work on FASTOUT, we used classes with uniform variances across all attributes. This is unlikely to be the case in the real data we studied, however, it would be interesting to study through synthetic data how the classification process is affected by wide variations in variances on different attributes for single classes (non-hyperspherical). In addition, while the SERA algorithm discussed in Appendix A is available in code for arbitrarily high-precision to handle false positive estimation in extremely high-dimensionality, the relative slowness of this code and the likely scope for more efficient approaches to or approximations of the calculation call for further research.

Appendix A

Appendix A: True and False Positive Rates In Higher Dimensional Problems

A.1 The Problem of Measurement Error

In almost every data mining project, one has to take account of false positives (FP) and false negatives (FN). This issue becomes especially acute in very-high dimensional experiments such as those employing microarrays. While we will discuss this problem with frequent references to microarrays, the implications are entirely general to any problem with N samples and d dimensions with particular relevance where d is large.

In this discussion, 'targets' are features, in the case of feature selection, or points that we seek in the case of clustering. The term 'features' refers to dimensions or attributes. Those of interest show some pattern, for example, indicating up or down regulated genes. In the microarray scenario, there are usually a small number of samples, which are treated as points and a large number of genes which are treated as dimensions. In other data mining contexts, the targets could be data points each of which has values over a set of attributes. This is illustrated in Figure A.1. Thus the semantics depend on the circumstances but the algorithm that we will present later for estimating true positive and false negative rates is entirely general.

While a dataset contains a set of T targets, we observe the set O. The true positives are $T \cap O$, the false negatives are $T \backslash (T \cap O)$ and the false positives are $O \backslash (T \cap O)$. False positives are also called type I errors while false negatives are type II errors.

In the data mining community we often incorrectly assume that our data is free of error. This allows us to trust our results. If we obtain a result such as a cluster containing a set of points P, we typically report P as the target cluster members. However, in any real world scenario, this approach is unsafe. Virtually all data is noisy and this noise leads to both false positives and false negatives. As dimensionality increases, this problem becomes increasingly acute as sparsity increases exponentially with dimensionality and, thus, the likelihood of observing a cluster similarly declines under standard approaches.

Whether we are clustering or doing feature selection, we are looking for consistency however this is defined. In a cluster, all points that attained the same criterion/criteria are collected. Similarly, a sub set of features is often collected because the features are all considered to have fulfilled some criterion such as co-regulated genes. To understand the following analysis and method, it is important to note that the approach being presented is a generic statistical approach that can equally be applied to groups of points in a feature space or groups of features in a dataset as suggested by Figure A.1. The problem can be couched as groups of 'similar' objects or 'identical'. For example, points with values above some threshold can be rated as similar if the extent of their exceeding the threshold is retained, or identical if they are simply classed as being greater than the threshold. The 'identical' case is simpler to express statistically and is being handled in this chapter. The 'similar' case can always be reduced to the 'identical' case. This is similar to the many results in graph theory based on unweighted edges which are still helpful for understanding weighted graphs.

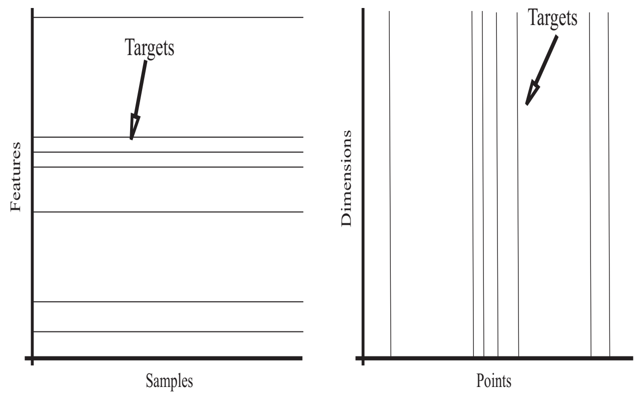

Figure A.1: Examples of targets in different data mining scenarios. On the left, the targets are features showing significance across samples. On the right the targets are points clustered or classified based on similarity on a set of dimensions.

For example, let us imagine a survey made of 100 students. Each is asked 50 questions. Suppose 20 of the students do genuinely belong to a group that should give similar or identical answers on a subset of the questions. What chance have we of observing a subspace cluster of all 20? Answers may be misrecorded or misinterpreted. If we find a group of 18, we should ask what is the likely true size of the group? Are any of these 18 not really part of the group (False Positives)? Which of the students not clustered but closely similar should be included (False Negatives)? Obviously such questions are important but, surprisingly, have received little attention.

Once one has a series of samples S with dimensionality d such as a number of microarray tests of patients with the same disease and we are looking for a target set of activated genes out of thousands measured, the statistical problem becomes non-trivial. This thesis provides an algorithm for computing the answer to this problem and presents how this answer can be applied to improve research results and reduce research costs. Specifically, the problems of both false positives and false negatives are addressed.

In addition, as dimensionality increases, increasing sparsity causes the false negative rate to increase. It is therefore important to be able to estimate this effect. That leads to the final hypothesis:

Hypothesis 4. Error rate estimation*: Given P points and A relevant features, the false positive and false negative rates can be estimated subject to certain assumptions.*

This Appendix provides the means to make such an estimate based on standard statistical calculations.

A.2 Methodology

As has been often mentioned in the literature, if we have a 5% chance of a false positive (FP) and a large number of features then we will have many false positives. 10,000 genes will be expected to give 500 FPs. Lee [117], in the context of microarray studies, argues that this is a reason for replicating the tests on each chip. This doubles the cost but greatly reduces (without eliminating) the FP rate. Another way of eliminating FPs is developed here based on a single round of microarray testing.

The question we address is how many targets (true positives) or non-targets (false positives) will be seen consistently if we test many samples? That is, if we have s samples, how many will appear with a measured value considered non-null on every sample? From this we can answer the questions, 'how many samples do we really need?' and 'how can we eliminate false positives?'.

The underlying statistical question is, essentially, 'given n coins with a weighted probability of coming up heads of p and d tosses of each, what is the probability of seeing at least the same k coins coming up heads in every round of tosses?' If there is no error or noise then $p = 1$.

In a typical high dimensional problem such as a microarray (MA) study where there are both target and non-target features, we can ask two questions both of the form 'given n genes with a probability of testing positive p and d samples, what is the chance that at least k unique genes are positive in every sample (that is, consistently up or down regulated)?' There are usually two questions of interest because, typically, we have a small number of target genes which are being sought and which we can expect to observe at some high likelihood p_t, even though we do not initially know which they are, and then a very large number of other genes from which we may get false positives due to error or noise with a probability of $p_f = 1 - p_t$, making the reasonable assumption that both will be subject to the same error rate or noise level.

The error rate can sometimes be known. For example, microarray manufacturers provide estimates and, in general, controlled experiments can be made on known quantities. However, this is often difficult or expensive so, in most situations, the error rate or degree of stochasticity is unknown. While this might appear to be a major hurdle, given the importance of the issue to science, the research community has long since adopted a convention of assuming an acceptable error rate of

5%, which means that if a measure is made with better than 95% certainty, it is accepted. In terms of probability one seeks $p < 0.05$ or, sometimes, $p < 0.01$. Other p-values may be used when there is good reason to do so. In the algorithm provided in this paper, any confidence level can be input.

Our approach assumes that the stochastic processes resulting in errors due to the samples or the measuring technology are independent between features. The method given also assumes that the error rates are the same for all features or points, but it could readily be modified if a distribution of these was known.

Algorithm 7 SERA – Statistical Error Rate Algorithm

```
Input: Number of samples $s$, number of features $n$, number of significant features observed $k$,
probability $p$ of seeing a significant feature in a sample.
Output: Probability of seeing $k$ significant features
Note: $choose(a, b)$ computes $\binom{a}{b}$

/*Main Function*/
$ComputeProbability(n, s, k, p)$ {
$\{Mx, Px\} \leftarrow BuildMatrix(p, n, k)$ /*Algorithm 8*/
return $Compute(n, s, k)$
}

$Compute(n, s, k)$ {
$val \leftarrow 0$
for $w \leftarrow 0$ to $n - k$ incrementing by 1 do
   $val \leftarrow val + choose(n, n - w) \times Px(n, n - w) \times CompLine(n, w, s - 1)$
end for
return $val$
}

$CompLine(n, w, s)$ {
/*if we've seen this parameter combination before*/
/*return the stored value, otherwise compute and store*/
if $AlreadyComputed(n, w, s)$ then
   return $GetStored(n, w, s)$
end if
$val \leftarrow 0$
for $v \leftarrow 0$ to $n - k$ incrementing by 1 do
   $val \leftarrow val + Mx(v, w) \times$
      $CompLine(n, max(w, v), s - 1)$
end for
$SetStored(val, n, w, s)$
$AlreadyComputed(n, w, s) \leftarrow true$
return $val$
}
```

Since this problem has been framed in terms of coin tosses, we can derive a statistical formulation to evaluate the probability. The problem is related to the binomial distribution. The binomial distribution deals with a sequence of n tosses of a weighted coin. The probability of getting exactly k successes in n trials is given by the Probability Mass Function:

Algorithm 8 Build Look up Value Matrix

```
Input: Number of features $n$, number of significant features observed $k$, probability $p$ of seeing a significant feature in a sample.
Output: $Mx, Px$ which store the precalculated values
Note: $choose(a, b)$ computes $\binom{a}{b}$

$BuildMatrix(p, n, k)$ {
/*Reset temporary value vectors*/
$\forall n, k\ Px(n, k) \leftarrow p^k(1-p)^{(n-k)}$
for $w \leftarrow n$ to $k$ decrementing by 1 do
  $ptemp \leftarrow Px(n, w)$
  for $v \leftarrow n$ to $k$ decrementing by 1 do
    if $(w + v \geq n + k)$ then
      $Mx(n - w, n - v) \leftarrow choose(n, w) \times ptemp$
    else if $(w \leq v)$ then
      $Condition1(w, v, choose(v, w))$
    else
      $Condition2(w, v)$
    end if
  end for
end for
return $Mx, Px$
}

$Condition1(w, v, val)$ {
$tot \leftarrow val$
$j \leftarrow w - 1$
$i \leftarrow 1$
while $(j \geq k)$ do
  $tot \leftarrow tot + choose(v, j) \times choose(n - v, i)$
  $j \leftarrow j - 1$
  $i \leftarrow i + 1$
end while
$Mx(n - w, n - v) \leftarrow tot \times ptemp$
}

$Condition2(w, v)$ {
$tot = choose(n - v, w - v)$
$j \leftarrow v - 1$
$i \leftarrow w - v + 1$
while $((j \geq k))$ do
  $tot \leftarrow tot + choose(v, j) \times choose(n - v, i)$
  $j \leftarrow j - 1$
  $i \leftarrow i + 1$
end while
$Mx(n - w, n - v) \leftarrow tot \times ptemp$
}
```

$$Pr(k) = \binom{n}{k} p^k (1-p)^{(n-k)} \tag{A.1}$$

In the problem addressed in this paper, we are tossing n labelled weighted coins a sequence of d times and we want to know how often the same set of *at least* k coins come down 'heads' every time.

Suppose we want to compute how many ways we have of getting at least k heads on every throw of n weighted coins. On the first throw of all the coins, as per Equation A.1, we have $\binom{n}{k}$ ways of getting k heads with a probability $p^k(1-p)^{(n-k)}$. Since we are interested in 'at least' k heads, we have to compute the probability of $k+1$ heads, $k+2$ heads, etc. up to all n coins being heads. This is expressed as the Cumulative Distribution Function:

$$Pr(X \geq k) = \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{(n-j)} \tag{A.2}$$

On the second round, we have to check, for each of the first possible sets of throws, the chance of a match with all possible k or more heads combinations in the second throw, i.e. where k or more coins have heads in both rounds of throws. This has to be continued for all of the d rounds. This can be expressed through the following equations.

$$Pr(n,s,k,p) = \sum_{w=0}^{n-k} \binom{n}{n-w} Px(n, n-w, p) f_3(n, w, s-1, p) \tag{A.3}$$

Equation A.3 is the principal expression and simply reduces to Equation A.2 for a single round of coin tosses. Equation A.4 computes the probability of any particular set of heads and tails.

$$Px(a,b,p) = p^b (1-p)^{a-b} \tag{A.4}$$

Function f_3, defined in Equation A.5, handles the branching computation for multiple rounds.

$$f_3(n,w,s,p) = \begin{cases} \sum_{v=0}^{n-k} Mx(n-v, n-w, p) f_3(n, max(w,v), s-1, p) & s \geq 1 \\ 1 & \text{otherwise} \end{cases} \tag{A.5}$$

$$Mx(a,b,p) = Px(n,a,p) \times \begin{cases} \binom{n}{a} & a+b \geq n+k \\ f_1(a,b,\binom{b}{a}) & w \leq b \\ f_2(a,b) & \text{otherwise} \end{cases} \tag{A.6}$$

$$f_1(w,v,x) = x + \sum_{j=w-1, i=1}^{j=k, i=w-k} \binom{v}{j} \binom{n-v}{i} \tag{A.7}$$

$$f_2(w,v) = \binom{n-v}{w-v} + \sum_{j=v-1, i=w-v+1}^{j=k, i=v-k} \binom{v}{j} \binom{n-v}{i} \tag{A.8}$$

Equation A.6, which defines the function Mx called in Equation A.5, appears complex but this is a consequence of handling the 'at least k' condition. If one takes exactly k, then $w = v$ and only the second choice is applicable unless one takes the trivial condition of $n = k$ where the first choice applies. For exactly k, $f_1 = 1$ because for each ordered seqence of k heads, there is exactly 1 matching possibility in each subsequent throw so the probability declines exponentially with the number of throws as expected. All the sums involve a count down from 'all n' heads to k heads. At each node in the tree there are 1 or more possible matches due to k or more heads which match the entire branch so far.

It is clear that there is a tree of computations with a fixed branching factor. This suggests an intractable computation for large values of the parameters as the number of computations increases exponentially. Fortunately, we can greatly accelerate the computation. Firstly, it can be seen that the comparison between each pair of rounds involves the same computations, so a matrix of values can be prebuilt and used for lookup (Algorithm 8).

Further, we can formulate the problem using a recursive algorithm and the return value from each call depends only on the parameters passed. Thus we can keep a table of calling parameters and return values and look up the result in this table whenever we find a parameter match (Algorithm 7). Since the redundancy is high, the problem of exponential blowup is substantially reduced. The principal cost in the calculation turns out to be the computations of $\binom{n}{k}$ for the lookup table where $n > 1000$ primarily because high precision math libraries are required that take a great deal more computing power on a 32 bit machine. If a table of these values was precomputed on a very fast computer or computing cloud, then the remaining computations for any given problem would return very quickly.

Fortunately, we can greatly accelerate the computation. Firstly, it can be seen that the comparison between each pair of rounds involves the same computations, so a matrix of values can be prebuilt and used for lookup (Algorithm 8).

We have not found in the literature any previous algorithm that provides this computation so we are unable to offer a comparison.

A.3 Empirical Examples

As discussed above, there are two separate applications of this algorithm. One is to estimate the false positive rate. That is, given n points or features that are non-target and therefore have a low probability of being false positives, how many will be seen consistently? For example, in a microarray experiment with n features, assuming they are (almost) all non-target, how many will show as significantly regulated on every sample. Or, in a clustering application, out of n points with d dimensions being considered, how many false positives can we expect to see? Here we assume that, in order for a false positive to appear in a d dimensional cluster, it has to be falsely given significance on all d dimensions. This would apply explicitly to applications like projection clustering [11, 47]

Table A.1: The number of target features designated significant in every sample by an application and the number that can be inferred to exist at a 95% confidence level for different numbers of samples (e.g. 1 sample, 5 samples, etc.).

Designated Significant	Inferred Targets for No. of Samples				
	1	5	10	20	30
5	6	8	9	10	10
10	12	14	16	16	17
15	17	20	22	23	23
20	23	27	28	29	30
25	28	32	34	35	36
30	34	38	40	41	42
50	55	62	64	65	66
100	109	120	122	124	125

and, arguably, implicitly to other algorithms.

The second application is regarding the target features that we expect to observe with high probability. If we observe m, how many really exist? Answering that allows us to estimate the true positive and false negative rates.

A.3.1 False Negatives and the True Positive Rate

Some sample computations are shown in Table A.1. The first column is the number of points or features observed and the other columns indicate the estimated true number of targets for various numbers of dimensions or samples, respectively. An error rate of 5% is assumed. For example, for 10 samples, if we see 10 features testing positive on all the samples, then, at the 95% confidence level, we can expect there are 16 target features, so we observe about 62%. If we observe a total of 30 features, then the real number is about 40 so 75% of the likely target features have been observed.

Once the number of false negatives has been estimated, from which the false negative rate can be derived, features which were not significant (or unclustered points, etc.) but which had a high similarity to those selected as targets can be added. For example, suppose 75 points are detected in a 20 dimensional subspace cluster but we estimate there should have been 100. All of the 75 points are clustered because they have similar or identical values for the 20 dimensions. Other points may have such similarity but on fewer dimensions and could be ranked on the number of dimensions on which they were similar. The top 25 could then be incorporated into the cluster. Since measurement error/noise has likely caused some points to fall below the chosen similarity threshold, these are the points that most likely represent the missed cluster members. It is important to note that whether one is looking for sets of features or sets of points, the same statistical approach can be applied.

Naturally, the number of points clustered in the above example is a function of the heuristics, such as thresholds, employed. The assumption is that a threshold which seems 'reasonable' to a researcher will likely correspond to the conventional 5% error rate. This assumption is the foundation of the 5% convention throughout the research community.

Researchers can use our algorithm on their own set of parameters so we just give examples of usage here to illustrate the utility.

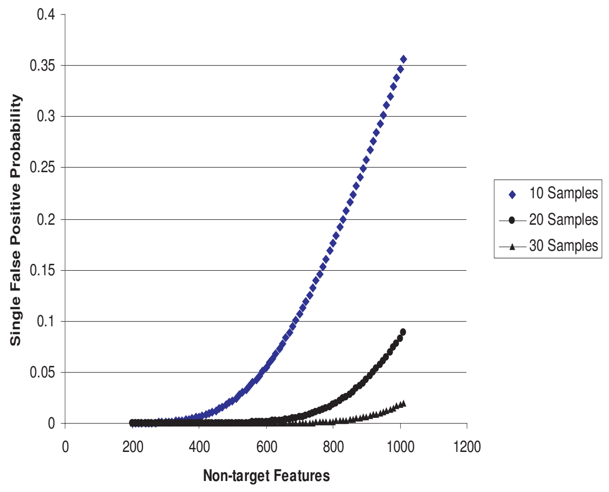

Figure A.2: False Positive likelihood for a single feature to test positive on all samples with increasing number of non-target features (e.g. genes) tested (0.4 = 40%). Probability of a FP for a single feature on a single sample = 5% (0.05).

A.3.2 False Positives

If we have a large number m of non-target features in our test arrays, then we can expect some FPs. The question is how many and at what size of m can any be expected?

For example, if there are 20 samples and we are looking to see if a single non-target feature tests positive in every sample then up to $m = 200$, the likelihood is vanishingly small (Figure 1). At $m = 1000$, the probability is about 8% and rising fairly quickly. Thus if we have a microarray with more than 1000 genes on it some FPs are quite possible. For 10 samples the likelihood is over 35% while for 30, at $m = 1000$, the likelihood is under 1%. This is illustrated in Figure A.2.

Obviously therefore, one strategy for eliminating false positives is to have a large number of samples. However, for microarrays, each sample can cost thousands of dollars to collect and process. Another alternative for the genome researcher is proposed in this paper taking advantage of the fact that we can estimate the number of FPs to be expected. This is outlined in Section A.4 below.

For the data mining researcher, the utility is in helping correct for false negatives while tracking false positives since both impact effectiveness. The algorithm can be incorporated into clustering and other algorithms as a post-processor. For example, after producing a clustering result, the algorithm can be used to estimate the number of false negatives for each cluster or class. Suppose the estimated number is m. Then the m points nearest to the cluster could be incorporated. Another approach is to use it to estimate the overall error rate. An example of how these can be done is given in Section A.3.3.

Table A.2: The top genes from the Colon Tumour database.

Rank	No.	Score	Label
1	377	40	Z50753 H.sapiens mRNA for GCAP-II/uroguanylin precursor
2	765	38	M76378 Human cysteine-rich protein (CRP) gene, exons 5 and 6
3	493	37	R87126 MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)
4	581	37	T51571 P24480 CALGIZZARIN
5	1892	37	U25138 Human MaxiK potassium channel beta subunit mRNA, complete cds

The following section gives a real world example in the high noise, very high dimensional application of microarray analysis. This example is of particular interest because it appears that no other approach to this data has yielded a small set of target genes. Where genes have been positively associated with cancers, the number of genes involved is very small. The preprocessing is particular to microarray data but the subsequent computations have very general application. On the other hand, this is not presented as a definitive way of identifying the correct features for these particular datasets. SERA does not perform either clustering or feature selection and thus does not provide candidate genes so, for the sake of giving an example, a very simple method is described in Section A.3.3 for identifying a subset and thus providing a count. Once that is provided, SERA gives us estimates of FN and FP rates from which we can adjust or revisit our results.

A.3.3 Colon Tumour Dataset

Direct Analysis

This dataset [16] has 62 Affymetrix oligonucleotide arrays from biopsies of colon tumours, 40 are labelled positive and 22 are labelled negative. Our interest is in determining which genes are associated with the malignant condition. Values for 2000 genes are provided for each sample.

For this analysis, the 40 positive samples are assessed for being consistently different than the negative cases. That is, the negative samples provide the standard levels against which the positive samples are to be tested. For each gene a count is made of how often it is up or down regulated over the samples. The data was normalised in a standard way by taking the log_2 of the values divided by their mean value over all attributes for each sample [57]. Minus and plus superscripts are used to identify the groups. The mean $\overline{x}_i^-$ and standard deviation σ_i^- of the negative samples for feature i were computed for each gene and the values for the positive genes x_i^+ were converted such that

$$x_i^{+'} = \frac{(x_i^+ - \overline{x}_i^-)}{\sigma_i^-} \tag{A.9}$$

This gives normalized deviations of the positive group from the negative group means. In order to accumulate the set of genes in the positive samples that are consistently up or down regulated, genes were checked for being greater or less than the mean of the other group by some small amount, here at least 0.05σ. For the experimental group, only one gene (40 samples) was consistently different in the same direction than the control group. Applying the algorithm shows that 4 other genes are likely to be target genes, rounding to the nearest integer. Thus we can predict that the top 5 genes are strong candidates for being associated with this disease condition. These genes are given in Table A.2. The highest scoring gene appears in a cluster of genes that may be implicated in Leukemia [78]. The likelihood of a false positive is small, as can be inferred from Figure A.2.

Alon et al. applied clustering to this dataset [15] based on a deterministic-annealing algorithm and showed that this clustering could separate the two classes but was not effective in identifying a few genes that likely play a fundamental role in this disease condition. After clustering, Alon et al. removed the 500 most significant genes and found that their algorithm still gave a successful clustering of the two classes. The various existing approaches to multiple test corrections can regulate the number of targets identified by adjusting thresholds but cannot identify the correct number. Mahata and Mahata [129], using a minimum probability of error approach with a complement naive Bayes classifier, took the approach of trying different numbers of genes until they got the highest classification result. For the colon tumour data they came to a value of 5 genes, identical to our result. According to Mahata and Mahata "Selection of the minimum number of important individual genes as biomarkers is still an open problem" [129]. SERA provides a statistical solution to this problem. This dataset is discussed in Section 3.5.2. The top three genes are ranked in the top 10 for colon cancer by Huang and Kecman [92]. Genes ranked 2 and 3 here are cited by Mahata and Mahata [129]. These two were identified by GSEP. Gene 1, Z50753, has been cited as important for colon cancer by Xiong et al. [168].

Suppose no gene had scored 40 over 40 samples. This would indicate that there may be less than 5 target genes. To further investigate, we could take the highest scoring gene. Suppose that gene had a score of 20. Then if we apply the algorithm on the basis of one positive on 20 samples then we get a result, after rounding, of 3 anticipated false negatives.

Clearly the method adopted for any particular application contributes to the error rate. For example, different clustering methods will yield different sized clusters. In the example above, the number of genes detected is influenced by the choices made in selecting them. Thus we do not know exactly what our overall error rate is without labelled data to test against, in this case genome level labelling. However, we can certainly improve on the raw result by making the conventional assumption of a 5% error rate as we have above, and using that to derive the number of false negatives and positives.

Application to Classification and Clustering

A number of different types of algorithms have been used to classify or cluster this dataset [15, 26, 71, 163]. Accuracy rates compared against the diagnostic labels range up to 92% (achieved by GSEP, Section 3.5). That means, for the group of 40 positive samples, these algorithms classified up to 36 correctly. Naturally we considered why we had not achieved 100% accuracy. A number of the genes appeared to be misclassifying the remaining samples. However, it is interesting to ask

Table A.3: The top 8 peptides identified from the Ovarian Cancer data [140].

Rank	No.	Score	Label
1	2237	162	MZ434.68588
2	2238	162	MZ435.07512
3	2236	162	MZ434.29682
4	2235	161	MZ433.90794
5	2239	160	MZ435.46452
6	181	159	MZ2.7610465
7	2240	158	MZ435.85411
8	2234	158	MZ433.51923

how many out of 40 should we expect to see, given that there must be a non-zero error rate inherent in the data as well as consequential to the heuristics of the classifier?

In order to apply SERA, we need an estimate of the number of dimensions relevant to the detection of the samples. If we knew this then we can find the error rate. Alternatively, if we assume an error rate then we can get an estimate of the number of relevant dimensions. First, let us take the result of the last section, that is that there are 5 relevant dimensions. Then, SERA shows that finding 36 out of 40 indicates only about a 1% error rate so this is actually an excellent experimental result. If the error rate is, in fact, higher, then the number of relevant dimensions is less.

Of course, SERA is computing based on 36 out of 40 samples having proved positive on all 5 dimensions every time. The actual algorithms will not necessarily require this or even investigate in this way. However, implicit in any algorithm is both the concept of a selected feature (or point) being consistent in its significance, that is significant frequently if not every time, and then a post processing stage, as we discussed above, where the best candidates are selected according to some threshold. This secondary stage accommodates somewhat for the errors or noise. The judicious choice of such thresholds actually achieves the low error rate indicated by SERA as it is a statistical assessment of the likelihood of such a result.

A.3.4 Additional Examples

Ovarian Cancer Dataset

This dataset [140] has 253 samples of proteomic spectra from 91 ‘normal’ and 162 diagnosed cases of ovarian cancer in different stages. Each sample contains 15,154 peptides defined by their mass/charge (M/Z) ratio. The spectra was obtained by mass spectroscopy on serum samples. Of the 162 diagnosed cancer samples, 3 peptides indicated a positive result on every sample. Applying the SERA algorithm at a 95% confidence level suggests that the true number is 8. Table A.3 shows the top 8 peptides. This dataset is discussed in Section 3.5.2. 7 out of 8 of the peptides listed above are identified by GSEP. Peptides 2 and 3 are cited as important in this dataset by Yap et al. [171]. Peptide 2 was also cited by Sorace and Zhan as predictive [157]. Peptide 5 was identified as highly predictive by Liu [94].

Table A.4: The top 24 genes identified from the Lung Cancer data for the Mesothelioma samples [79].

Rank	No.	Score	Label	Rank	No.	Score	Label
1	173	31	1158_s_at	13	94	31	1086_at
2	134	31	1121_g_at	14	22	31	1020_s_at
3	177	31	1161_at	15	143	31	1130_at
4	139	31	1126_s_at	16	9	31	1009_at
5	109	31	109_at	17	104	30	1095_s_at
6	130	31	1119_at	18	80	30	1073_at
7	8	31	1008_f_at	19	81	30	1074_at
8	147	31	1134_at	20	26	30	1024_at
9	10	31	100_g_at	21	110	29	1100_at
10	108	31	1099_s_at	22	141	29	1129_at
11	89	31	1081_at	23	57	29	1052_s_at
12	164	31	114_r_at	24	33	29	1030_s_at

Table A.5: The top 7 genes identified from the Lung Cancer data for the ADCA samples [79].

Rank	No.	Score	Label
1	150	131	1137_at
2	76	131	106_at
3	172	130	1157_s_at
4	160	123	1146_at
5	53	123	1049_g_at
6	18	121	1017_at
7	69	121	1063_s_at

Lung Cancer Dataset

This dataset [79, 19] has 181 samples each with 12,533 features. The samples are divided into 150 Adenocarcinoma (ADCA) and 31 Mesothelioma (MPM). These are two different disease conditions. Of the 31 Mesothelioma samples, 16 were positive on every sample. This indicates a likely 24 prognosticator genes (Table A.4).

Of the 150 ADCA samples, 2 were positive on 131 samples. No genes had a higher score. This indicates a likely count of 7 prognosticator genes (Table A.5).

On this dataset Gordon et al. [79] selected the 8 genes with the greatest statistically significant difference between the groups and a high expression level in at least one of the groups. Five were from the MPM group and 3 from the ADCA group. These gave a 95% differential accuracy. Despite their excellent result, there is no way of knowing from this approach what the correct number of genes is. Since the form of the labels in the dataset we used are not standard gene labels, direct comparison to other research was not possible. This dataset is also discussed in Section 3.5.2.

A.4 Optimising Microarray Results: An approach to Minimising Costs by Reducing Needed Experiments

We have shown that the number of positives - e.g. target genes - seen in every sample is typically 50% - 75% of the underlying number present. This helps us decide how many of the genes which test positive in less than all of the samples are to be accepted. We have also shown that false positives can occur but the likelihood drops sharply with increasing number of samples.

From the above we can propose the following way of eliminating false positives. Even though this involves a second round of testing of the samples taken from patients, the second round is much cheaper due to the small number of genes tested for and the number of samples required can be much reduced so lowering the overall cost. Lee [117] argues for replicating the tests on each chip to reduce false positives, which is typically much more costly.

Since we have an estimate of the number of possible false positives fp, if fp is above some small value that indicates a significant risk of assigning importance falsely to a gene (e.g. $fp > 0.5$) and, given that we have identified a set of potential target genes G, new tests can be performed on only the top $|G'|$ ranking genes such that $G \subseteq G'$.

More formally, let the genes testing repeatedly positive across all samples be G^*. Let the estimate of the number misclassified using the algorithm described here be $Est(p)$ based on the error rate p determined by the experimenters from experiments or as sufficiently conservative to account for all likely measurement errors. Then of the ranked set of genes G the top g candidates that were not in G^*, where $g \geq Est(p)$ along with G^* are retested.

That is, retest those genes identified and those that came close to being identified as implicated in the disease. The probability of a false positive appearing twice declines exponentially with the number of retesting rounds and thus, in most cases, is well below any meaningful threshold after a second round.

This approach greatly reduces the retesting costs by minimizing the likelihood that any false positives will occur across all the samples without the high cost of replication tests using microarray chips. Such replication would only be important if the number of samples was small.

Variations on this approach can be utilised in many applications.

A.5 Conclusions

This appendix presents an algorithm which could prove useful for both data mining researchers and those planning genomic, proteomic and other high-dimensional studies. The problem of false positives and negatives is a major issue for researchers.

Even the most advanced genome work being done today relies on ranking features by feature level t-tests as the primary analysis level. For example, the outstanding work on breast cancer by Easton et al [61]. While the smallest p-values are likely associated with the most interesting features, it would be advantageous to estimate the probability of a false positive as well as the likely number of false negatives associated with whatever number of features are selected as most significant. Both of these could assist in determining the correct number of features to select and the reliability of this selection.

As we have shown, the presented algorithm can give both an estimate of the number of false negatives and the likelihood of false positives. This algorithm can be incorporated into data mining applications to take advantage of these results to improve effectiveness.

We have also shown that this approach can lead to an estimate of the number of target features in situations where other approaches can only give a general ranking. Finally, medical researchers can use the estimates to reduce costs and procedures by minimising the number of samples that have to be taken.

Bibliography

[1] H Abdi. *N.J. Salkind (ed.): Encyclopedia of Measurement and Statistics*, chapter Bonferroni and Sidak corrections for multiple comparisons. Sage, 2007.

[2] Dimitris Achlioptas. Database-friendly random projections. In *Symposium on Principles of Database Systems*, 2001.

[3] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, Ina Müller-Gorman, and Arthur Zimek. Finding hierarchies of subspace clusters. In *Proc. 10th Europ. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2006.

[4] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, Ina Müller-Gorman, and Arthur Zimek. Detection and visualization of subspace cluster hierarchies. *Lecture Notes in Computer Science*, pages 152–163, 2007.

[5] Elke Achtert, Christian Böhm, Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Robust, complete and efficient correlation clustering. In *Proceedings of the 7th SIAM International Conference on Data Mining (SDM)*, 2007.

[6] ACM. ACM KDD Cup '99 results. `http://www.sigkdd.org/kddcup/`, 1999.

[7] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 61–72, 1999.

[8] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 70–81, 2000.

[9] C.C. Aggarwal and P.S. Yu. Outlier detection for high dimensional data. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pages 37–46, 2001.

[10] C.C. Aggarwal and P.S. Yu. An efficient and effective algorithm for high-dimensional outlier detection. *VLDB Journal*, 14(2):211–221, 2005.

[11] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[12] G Alexe, S Alexe, LA Liotta, E Petricoin, M Reiss, and PL Hammer. Ovarian cancer detection by logical analysis of proteomic data. *Proteomics*, 4:766–783, 2004.

[13] Mohammed Liakat Ali, Luis Rueda, and Myriam Herrera. On the performance of Chernoff-distance-based linear dimensionality reduction techniques. *Advances in Artificial Intelligence*, 4013:467–478, 2006.

[14] F. Alimoglu and E. Alpaydin. Methods of combining multiple classifiers based on different representations for pen-based handwriting recognition. In *Proc. 5th Turkish Artificial Intelligence and Artificial Neural Networks Symposium*, 1996.

[15] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA*, 96(12):67456750, 1999.

[16] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. `http://microarray.princeton.edu/oncology/affydata/index.html`, 2007.

[17] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *In Proc. of the National Academy of Sciences of the USA*, 96:6745–6750, 1999.

[18] D.F. Andrews. Plots of high-dimensional data. *Biometrics*, 28(4):125–136, 1972.

[19] Nasimeh Asgarian and Russell Greiner. Diagnostic datasets. Data source: `http://www.cs.ualberta.ca/~greiner/RESEARCH/OLD-BiCluster/Diagnosis.html#D1`, 2007.

[20] Nasimeh Asgarian and Russell Greiner. Using rank-one biclusters to classify microarray data. Technical Report CPDC-TR-9906-010, University of Alberta, 2007.

[21] I. Assent, R. Krieger, E. Mller, and T. Seidl. Subspace outlier mining in large multimedia databases. In *Parallel Universes and Local Patterns, Dagstuhl Seminar 07181*, 2007.

[22] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56:153–167, 2004.

[23] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley, 1994.

[24] S.D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time randomization and a simple pruning rule. In *Proc. SIGKDD*, 2003.

[25] Richard Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[26] Amir Ben-Dor, Laurakay Bruhn, Nir Friedman, Iftach Nachman, Michel Schummer, and Zohar Yakhini. Tissue classification with gene expression profiles. *Journal of Computational Biology*, 7(3-4):559–583, 2000.

[27] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *RECOMB '02: Proceedings of the 6th Annual International Conference on Computational biology*, pages 49–57, New York, NY, USA, 2002. ACM.

[28] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[29] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B*, 57:289–300, 1995.

[30] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals. of Statistics*, 29(4):1165–1188, 2001.

[31] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful. In *Int. Conf. on Database Theory*, pages 217–235, 1999.

[32] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. *Knowledge Discovery and Data Mining*, pages 245–250, 2001.

[33] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. `http://archive.ics.uci.edu/ml/`, 1998.

[34] Christian Böhm, Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. Density connected clustering with local subspace preferences. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 27–34, Washington, DC, USA, 2004. IEEE Computer Society.

[35] Christian Böhm, Karin Kailing, Peer Kröger, and Arthur Zimek. Computing clusters of correlation connected objects. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 455–466, New York, NY, USA, 2004. ACM.

[36] C.E. Bonferroni. *Il calcolo delle assicurazioni su gruppi di teste*. pages 13-60. Rome, 1935.

[37] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[38] R. B. Brem, G. Yvert, R. Clinton, and L. Kruglyak. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 296:752–755, 2002.

[39] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proc. SIGMOD Conf.*, pages 93–104, 2000.

[40] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16:575–577, 1973.

[41] L. Candillier, I. Tellier, F. Torre, and O. Bousquet. Ssc: Statistical subspace clustering. In *Proc. Of Machine Learning and Data Mining in Pattern Recognition: 4th International Conference (MLDM'05)*, pages 100–109, 2005.

[42] F. Cappello, S. David, F. Rappa, F. Bucchieri, L. Marasa, T.E. Bartolotta, et al. The expression of hsp60 and hsp10 in large bowel carcinomas with lymph node metastases. *BMC Cancer*, 5(1):139, 2005.

[43] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *To Appear in ACM Computing Surveys*, 2009.

[44] Jae-Woo Chang and Du-Seok Jin. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *SAC '02: Proceedings of the 2002 ACM Symposium on Applied computing*, pages 503–507, New York, NY, USA, 2002. ACM.

[45] L. Chen, Q. Jiang, and S. Wang. A probability model for projective clustering on high dimensional data. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 755–760. IEEE Computer Society, 2008.

[46] Zhixiang Chen, Jian Tang, and Ada Wai-Chee Fu. Modeling and efficient mining of intentional knowledge of outliers. In *Proc. of the Seventh International Database Engineering and Applications Symposium (IDEAS'03)*, 2003.

[47] Chun-Hung Cheng, Ada Wai-Chee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proc. Of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 84–93, 1999.

[48] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 93–103, 2000.

[49] H. Cho, I. Dhillon, Y. Guan, and S. Sra. Minimum sum squared residue co-clustering of gene expression data. In *Proceedings of SDM*, 2004.

[50] F Chu and L. Wang. Gene expression data analysis using support vector machines. In Wunsch II Donald C, editor, *Proceedings of the 2003 IEEE International Joint Conference on Neural Networks*, page 226871, 2003.

[51] Y. Crama, P. L. Hammer, and T. Ibaraki. Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, 16:299–326, 1988.

[52] M. Dash, K. Choi, P. Scheuermann, and H. Liu. Feature selection for clustering - a filter solution. In *Proceedings of the IEEE International Conference on Data Mining*, pages 115–122, 2002.

[53] C. Ding, X. He, H. Zha, and H.D. Simon. Adaptive dimension reduction for clustering high dimensional data. In *Proc. of the 2nd IEEE International Conference on Data Mining*, pages 147–154, 2002.

[54] Carlotta Domeniconi, Dimitris Papadopoulos, Dimitrios Gunopulos, and Sheng Ma. Subspace clustering of high-dimensional data. In *Proc. of the 4th SIAM International Conference on Data Mining (SIAM'04)*, 2004.

[55] David Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. In *Proc. of the American Mathematical Society Conference on Mathematical Challenges of the 21st Century*, 2000.

[56] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Mach. Learn.*, 56(1-3):9–33, 2004.

[57] Sorin Drăghici. *Data analysis for DNA microarrays*. Chapman and Hall, 2003.

[58] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomised Algorithms*. Cambridge University Press, 2009.

[59] Jeroen Eggermont, Joost N. Kok, and Walter A. Kosters. Genetic programming for data classification: Partitioning the search space. In *Proc. of the 2004 Symposium on applied computing (ACM SAC04)*, pages 1001–1005. ACM, 2004.

[60] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining (KDD'96)*, 1996.

[61] Easton et al. Genome-wide association study identifies novel breast cancer susceptibility loci. *Nature*, 447:1087–1093, 2007.

[62] W. G. Fairbrother, Sharp Yeh, R. F., P. A., and C. B. Burge. Predictive identification of exonic splicing enhancers in human genes. *Science*, 297:1007–1013, 2002.

[63] Hongqin Fan, Osmar R. Zaiane, Andrew Foss, and Junfeng Wu. A nonparametric outlier detection for effectively discovering top-n outliers from engineering data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'06)*, pages 557–566, 2006.

[64] X. Z. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proc. of the 20th International Conference on Machine Learning (ICML 2003),*, 2003.

[65] Andrew Foss. Automatic categorical data clustering and spatial data clustering by consecutive resolution refinement. Master's thesis, University of Alberta, Spring 2002.

[66] Andrew Foss and Osmar R. Zaïane. A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. In *Proc. of the IEEE International Conference on Data Mining (ICDM'02)*, pages 179–186, 2002.

[67] Andrew Foss and Osmar R. Zaiane. The estimation of true and false positive rates in higher dimensional problems and its data mining applications. In *Proc. Foundations of Data Mining Workshop, in conjunction with IEEE International Conference on Data Mining (ICDM 2008)*, 2008.

[68] J. Friedman and N. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9:123–143, 1999.

[69] J. Friedman and J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society B*, 66(4):815–849, 2004.

[70] Elisa Fromont, Adriana Prado, and Céline Robardet. Constraint-based subspace clustering. In *Proc. of SIAM 2009*, 2009.

[71] T.S. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16:906–914, 2000.

[72] Guojun Gan and Jianhong Wu. Subspace clustering for high dimensional categorical data. *SIGKDD Explor. Newsl.*, 6(2):87–94, 2004.

[73] V. Ganti, J. Gehrke, and R. Ramakrishnan. CACTUS - clustering categorical data using summaries. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 73–83. ACM Press, 1999.

[74] M. R. Garey, R. L. Graham, D. S. Johnson, and D. E. Knuth. Complexity results for bandwidth minimization. *SIAM J. Appl. Math.*, 34(3):477–495, 1978.

[75] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proceedings of 24rd International Conference on Very Large Data Bases*, pages 311–322. Morgan Kaufmann, 1998.

[76] R. Gnanadesikan. *Methods for statistical data analysis of multivariate observations*. Wiley, 1977.

[77] S. Goil, H. Nagesh, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. Technical Report CPDC-TR-9906-010, Northwestern University, Evanston IL, 1999.

[78] T. R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[79] Gavin J. Gordon, Roderick V. Jensen, Li-Li Hsiao, Steven R. Gullans, Joshua E. Blumenstock, Sridhar Ramaswamy, William G. Richards, David J. Sugarbaker, and Raphael Bueno. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research*, 62:4963–4967, 2002.

[80] G.M. Groisman, S. Polak-Charcon, and H.D. Appelman. Fibroblastic polyp of the colon: clinicopathological analysis of 10 cases with emphasis on its common association with serrated crypts. *Histopathology*, 48(4):4317, 2006.

[81] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *IEEE TKDE*, 2000.

[82] S. Harmeling, G. Dornhege, D. Tax, F Meinecke, and K-R Müller. From outliers to prototypes: Ordering data. *Neurocomputing*, 69:1608–1618, 2006.

[83] D. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[84] Z. Y. He, X. F. Xu, Z. X. Huang, and S. C. Deng. A frequent pattern discovery method for outlier detection. In *Proc. of the 6th 5th International Conference on Web-Age Information Management, WAIM04*, pages 726–732, 2004.

[85] Zengyou He, Xiaofei Xu, and Shengchun Deng. A unified subspace outlier ensemble framework for outlier detection in high dimensional spaces. In *Proc. of the 6th International Conference, WAIM 2005*, pages 632–637, 2005.

[86] I. Hedenfalk, D. Duggan, Y. D. Chen, M. Radmacher, M. Bittner, R. Simon, P. Meltzer, B. Gusterson, M. Esteller, M. Raffeld, Z. Yakhini, A. Ben-Dor, E. Dougherty, J. Kononen, L. Bubendorf, W. Fehrle, S. Pittaluga, S. Gruvberger, N. Loman, O. Johannsson, H. Olsson, B. Wilfond, G. Sauter, O. P. Kallioniemi, A Borg, and J. Trent. Gene-expression profiles in hereditary breast cancer. *N. Engl. J.Med.*, 344:539–548, 2001.

[87] K.J. Helzlsouer, T.P. Erlinger, and E.A. Platz. C-reactive protein levels and subsequent cancer outcomes: results from a prospective cohort study. *Eur J Cancer*, 42(6):7047, 2006.

[88] Alexander Hinneburg and Daniel A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. of the 25th International Conference on Very Large Data Bases*, pages 506–517, 1999.

[89] Y. Hochberg and A.C. Tamhane. *Multiple Comparison Procedures*. John Wiley and Sons, 1987.

[90] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[91] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):1330, 1963.

[92] Te Ming Huang and Vojislav Kecman. Gene extraction for cancer diagnosis by support vector machines - an improvement. *Artificial Intelligence in Medicine*, 35(1–2):185–194, 2005.

[93] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[94] Liu Huiqing. *Effective use of data mining technologies on biological and clinical data*. PhD thesis, National University of Singapore, 2004.

[95] Assent I., Krieger R., Müller E., and Seidl T. DUSC: Dimensionality unbiased subspace clustering. In *Proc. IEEE International Conference on Data Mining (ICDM'07)*, pages 409–414, 2007.

[96] Assent I., Krieger R., Müller E., and Seidl T. INSCY: Indexing subspace clusters with in-process-removal of redundancy. In *Proc. IEEE International Conference on Data Mining (ICDM'08)*, pages 719–724, 2008.

[97] Y. Jiang and Z.-H. Zhou. Editing training data for kNN classifiers with neural network ensemble. *Lecture Notes in Computer Science 3173*, 2004.

[98] Anita Jones and Robert Sielken. Computer system intrusion detection: A survey. `http://www.cs.virginia.edu/~jones/IDS-research/Documents/jones-sielken-survey-v11.pdf`, 2000.

[99] Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 246–257, 2004.

[100] L. Kaufman and P. J. Rousseeuw. *An Introduction to Cluster Analysis*. Wiley, 1990.

[101] E. Keogh. `http://www.cs.ucr.edu/{\verb+~+}eamonn/shape/shape.htm`, 2006.

[102] E. Keogh, L.Wei, X. Xi, S-H Lee, and M. Vlachos. Lb Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proc. VLDB 2006*, pages 882–893, 2006.

[103] Hyunsoo Kim and Se Hyun Park. Data reduction in support vector machines by a kernelized ionic interaction model. In *Proc. of SDM. 2004*, 2004.

[104] Y. S. Kim, W. N. Street, and F. Menczer. Feature Selection in Unsupervised Learning via Evolutionary Search. In *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, 2000.

[105] Edwin M. Knorr and Raymond T. Ng. A unified approach for mining outliers. In *Proceedings of the 7th CASCON Conference*, pages 236–248, 1997.

[106] E.M. Knorr and R.T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. VLDB Conf.*, pages 392–403, 1998.

[107] E.M. Knorr and R.T. Ng. Finding intensional knowledge of distance-based outliers. In *Proc. VLDB Conf.*, pages 211–222, 1999.

[108] D. Knuth. *The Art of Computer Programming: Sorting and Searching, Vol. 3*. Addison-Wesley, 1973.

[109] Y. Kou, C.T. Lu, S. Sirwongwattana, and Y.P. Huang. Survey of fraud detection techniques. In *Proceedings of the 2004 International Conference on Networking, Sensing, and Control*, pages 749–754, 2004.

[110] Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Sebastian Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 250–257, Washington, DC, USA, 2005. IEEE Computer Society.

[111] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1–58, 2009.

[112] H.P. Kriegel, P. Kröger, and A. Zimek. Detecting clusters in moderate to high dimensional data. In *Proc. IEEE International Conference on Data Mining (ICDM'07)*, 2007.

[113] Lukasz A. Kurgan, Krzysztof J. Cios, Ryszard Tadeusiewicz, Marek R. Ogiela, and Lucy S. Goodenday. Knowledge discovery approach to automated cardiac SPECT diagnosis. *Artificial Intelligence in Medicine*, 23:149, 2001.

[114] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *Proc. ACM SIGKDD*, pages 157–166, 2005.

[115] Ar Lazarevic, Aysel Ozgur, Levent Ertoz, Jaideep Srivastava, and Vipin Kumar. A comparative study of anomaly detection schemes in network intrusion detection. In *In Proc. of the 3rd SIAM International Conference on Data Mining*, 2003.

[116] Michael Ledoux. *The Concentration of Measure Phenomenon*, volume 89 of *Mathematical Surveys and Monographs*. AMS, 2001.

[117] Jae K. Lee. Analysis issues for gene expression array data. *Clinical Chemistry*, 47:1350–1352, 2001.

[118] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E.G. Jennings, H.L. Murray, D.B. Gordon, B. Ren, J.J. Wyrick, J. Tagne, T.L. Volkert, E. Fraenkel, D.K. Gifford, and R.A. Young. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298:799–804, 2002.

[119] Tao Li, Sheng Ma, and Mitsunori Ogihara. Document clustering via adaptive subspace iteration. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 218–225, New York, NY, USA, 2004. ACM.

[120] Xiangyang Li and Nong Ye. A supervised clustering algorithm for computer intrusion detection. *Knowledge and Information Systems*, 8(4):498–509, 2005.

[121] Bing Liu, Yiyuan Xia, and Philip S. Yu. Clustering through decision tree construction. In *CIKM '00: Proceedings of the 9th International Conference on Information and Knowledge Management*, pages 20–29, New York, NY, USA, 2000. ACM.

[122] G. Liu, J. Li, K. Sim, and L. Wong. Distance based subspace clustering with flexible dimension partitioning. In *Proc. of 23rd Intl. Conf. on Data Engineering*, pages 1250–1254, 2007.

[123] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Boston, 1998.

[124] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491 – 502, 2005.

[125] J. Liu and W. Wang. Saccharomyces genome database. `http://www.yeastgenome.org`, 1997-2007.

[126] Jinze Liu and Wei Wang. Op-cluster: Clustering by tendency in high dimensional space. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, page 187, Washington, DC, USA, 2003. IEEE Computer Society.

[127] S. Madeira and Arlindo Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):24–45, 2004.

[128] P.C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Science of India*, 12:49–55, 1936.

[129] P. Mahata and K. Mahata. Selecting differentially expressed genes using minimum probability of classification error. *Journal of Biomedical Informatics*, 40(6):775–786, 2007.

[130] Markos Markou and Sameer Singh. Novelty detection: A review - part 1: Statistical approaches. *Signal Processing*, 83:2481–2497, 2003.

[131] Markos Markou and Sameer Singh. Novelty detection: A review - part 2: Neural network based approaches. *Signal Processing*, 83:2499–2521, 2003.

[132] T. Masters. *Neural, Novel and Hybrid Algorithms for Time Series Prediction*. John Wiley & Sons, 1995.

[133] Avraham A. Melkman and Eran Shaham. Sleeved coclustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD04)*, pages 635–640, New York, NY, USA, 2004. ACM.

[134] Vitaly Milman. The heritage of P. Levy in geometrical functional-analysis. *Asterisque*, 157:273–301, 1988.

[135] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In Y. Li, B. Liu, and S. Sarawagi, editors, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 533–541. ACM Press, 2008.

[136] Gabriela Moise. *Data mining in heterogeneous biological and bio-medical data sets*. PhD thesis, University of Alberta, 2008.

[137] Gabriela Moise, Jörg Sander, and Martin Ester. P3C: A robust projected clustering algorithm. In *Proceedings of ICDM'06*, pages 414–425, 2006.

[138] Burkhard Monien. The NP-completeness of the bandwidth minimization problem. *SIAM J. Appl. Math.*, 34:477–495, 1978.

[139] Burkhard Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM J. Alg. Disc. Meth.*, 7(4):505–512, 1986.

[140] NCIFDA. Proteomics dataset. Data source (version 6-19-02): `http://home.ccr.cancer.gov/ncifdaproteomics/ppatterns.asp`, 2007.

[141] K. K. E. Ng, A. W. Fu, and C. W. Wong. Projective clustering by histograms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):369–383, 2005.

[142] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago, Chile proceedings*, pages 144–155, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann Publishers.

[143] NHL. Official web site: `www.nhl.com`, 2008.

[144] Ch. H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

[145] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6(1):90–105, 2004.

[146] Jian Pei, Xiaoling Zhang, Moonjung Cho, Haixun Wang, and Philip S. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, page 259, Washington, DC, USA, 2003. IEEE Computer Society.

[147] Jose Manuel Pena, Jose Antonio Lozano, Pedro Larranaga, and Inaki Inza. Dimensionality reduction in unsupervised learning of conditional gaussian networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(6):590–603, 2001.

[148] M.I. Petrovskiy. Outlier detection algorithms in data mining systems. *Program. Comput. Softw.*, 29(4):228–237, 2003.

[149] Valerii Aleksandrovich Petrushin and Latifur Khan (Eds). *Multimedia data mining and knowledge discovery*. Springer, 2007.

[150] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 418–427, New York, NY, USA, 2002. ACM.

[151] S. Ramaswamy, R Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 427–438, 2000.

[152] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:622–626, 1971.

[153] P.J. Rousseeuw and K. Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[154] P.J. Rousseeuw and A.M. Leroy. *Robust regression and outlier detection*. John Wiley, 1996.

[155] J. Seiferas. Bibliography on theory of computer science. `http://liinwww.ira.uka.de/bibliography/Theory/Seiferas`.

[156] Karlton Sequeira and Mohammed Zaki. SCHISM: A new approach to interesting subspace mining. *Int. J. of Business Intelligence and Data Mining*, 1(2):137–160, 2004.

[157] J. M. Sorace and M. Zhan. A data review and re-assessment of ovarian cancer serum proteomic profiling. *BMC Bioinformatics*, 4(24), 2003.

[158] J. D. Storey. A direct approach to false discovery rates. *J. R. Stat. Soc. B*, 64:479–498, 2002.

[159] J.D. Storey and R. Tibshirani. Statistical significance for genome-wide studies. *PNAS*, 100:9440–9445, 2003.

[160] Vincent S. Tseng and Ching-Pin Kao. Efficiently mining gene expression data via a novel parameterless clustering method. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2(4):355–365, 2005.

[161] Z. Šidák. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62:626–633, 1967.

[162] Hui Wang. Nearest neighbors by neighborhood counting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(6):942 – 953, 2006.

[163] J. Wang, T.H. Bo, I. Jonassen, O. Myklebost, and E. Hovig. Tumor classification and marker gene prediction by feature selection and fuzzy c-means clustering using microarray data. *BMC Bioinformatics*, 4(1):60, 2003.

[164] P.H. Westfall and S.S. Young. *Resampling based multiple testing: Examples and methods for p-value adjustment*. John Wiley and Sons, 1993.

[165] G. Wiederhold. Bibliography on database systems. `http://liinwww.ira.uka.de/bibliography/Database/Wiederhold`.

[166] Wikipedia. Error function. `http://en.wikipedia.org/wiki/Error_function`, 2009.

[167] Kyoung Gu Woo, Jeong Hoon Lee, Myoung Ho Kim, and Jeong Hoon Lee. Findit: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255–271, 2004.

[168] Momiao Xiong, Xiangzhong Fang, and Jinying Zhao. Biomarker identification by feature wrappers. *Genome Research*, 8(11):1878–1887, 2010.

[169] Jiong Yang, Wei Wang, Haixun Wang, Wei Wang, and Philip S. Yu. Enhanced biclustering on expression data. In *Proceedings of IEEE Conference on Bioinformatics and Bioengineering (BIBE)*, pages 321–327, 2003.

[170] Jiong Yang, Wei Wang, Haixun Wang, and Philip S. Yu. Delta-clusters: Capturing subspace correlation in a large data set. In *Proceedings of 18th International Conference on Data Engineering (ICDE)*, pages 517–528, 2002.

[171] Ghim-Eng Yap, Ah-Hwee Tan, and Hwee-Hwa Pang. Learning causal models for noisy biological data mining: an application to ovarian cancer detection. In *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pages 354–359. AAAI Press, 2007.

[172] Mao Ye, Xue Li, and Maria E. Orlowska. Projected outlier detection in high-dimensional mixed-attributes data set. *Expert Systems with Applications: An International Journal*, 36(3):7104–7113, 2009.

[173] K. Yeung and W. Ruzzo. An empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[174] Kevin Y. Yip, David W. Cheung, and Michael K. Ng. Harp: A practical projected clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(11):1387–1397, 2004.

[175] Kevin Y. Yip, David W. Cheung, and Michael K. Ng. On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In *Proc. of the IEEE International Conference on Data Engineering (ICDE'05)*, pages 329–340, 2005.

[176] Kevin Y. Yip, Peishen Qi, Martin H. Schultz, David W. Cheung, and Kei-Hoi Cheung. Sembiosphere: A semantic web approach to recommending microarray clustering services. In *Proc. of the Pacific Symposium on Biocomputing*, pages 188–199, 2006. `http://psb.stanford.edu/psb-online/proceedings/psb06/yip.pdf`.

[177] Kevin Y. Yip, Peishen Qi, Martin H. Schultz, David W. Cheung, and Kei-Hoi Cheung. Sembiosphere online service. `http://yeasthub2.gersteinlab.org/sembiosphere/index.jsp`, 2006.

[178] M. L. Yiu and N. Mamoulis. Iterative projected clustering by subspace mining. *IEEE TKDE*, 17(2):176–189, 2005.

[179] L. Yu and H. Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *Proceedings of the 20th International Conference on Machine Learning*, pages 856–863, 2003.

[180] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proc. of KDD '02*, 2002.

[181] Amelia Zafra and Sebastián Ventura. Multi-objective genetic programming for multiple instance learning. In *Lecture Notes in Computer Science, Machine Learning: ECML '07*, 2007.

[182] M. Zaki, M. Peters, I. Assent, and T. Seidl. CLICKS: an effective algorithm for mining subspace clusters in categorical datasets. In R. Grossman, R. J. Bayardo, and K. P. Bennett, editors, *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 736–742. ACM Press, 2005.

[183] J. Zhang and H Wang. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowl. Inf. Syst.*, 10(3):333–355, 2006.

[184] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, 1996.