Bridging Microarray Platforms To Extend the Utility of Gene Expression Profiles

John Allen

Master of Science

School of Informatics

University of Edinburgh

2004

Abstract

Gene expression profiling experiments provide a wealth of data for use in the analysis of the transcriptome and have yielded results that have helped in the classification of disease type and suggestions of possible biological pathways. Two major types of platforms exist, cDNA platforms where ESTs are spotted onto a glass microscope slide and oligonucleotide platforms where each gene is represented by small lengths of approximately 20 base pairs of DNA. Due to this difference in set-up and subsequent use the two platforms tend to yield different results when the data is analysed by a common method such as a clustering algorithm.

This project seeks to find a general tool to cluster expression profile results from both sets of data simultaneously. A new clustering algorithm that has been presented at the Royal Statistical Society London in May 2004 will be utilised in the tool. Its claimed advantage over existing algorithms is that it can cluster objects on only small subsets of their attributes that show a signal above or below the background noise of the majority of the data. This type of data structure is typical of that produced by the microarray platforms and suggests the application of the new algorithm to their study.

Acknowledgements

This research was carried out jointly with the Institute of Biomathematics & Statistics Scotland and the School of Informatics at the University of Edinburgh.

The research and period of study from October 2003 to September 2004 was funded by a Masters Training Package studentship from the Engineering and Physical Sciences Research Council (EPSRC).

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*John Allen*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Gene Microarrays

Gene microarray experiments provide high throughput of gene expression levels from one or more biological samples. Their use in determining disease-subclass and patient prognosis is ever growing (Hubank [1]), with over 1,000 publications in the first half of 2003 alone (Ferl *et al.* [23]). By analysing the expression levels from the microarrays any interesting patterns found can be used to suggest an individual gene, or groups of genes, that are responsible for the different biological behaviour of the samples. Microarray data however is notoriously noisy, both random and systematic errors caused by mistakes in the printing of the slides, preparation of the DNA probes, light intensity measurements of the dyes used to mark samples and sample contamination itself can occur (Miller *et al.* [2] & Medvedovic [4]). The data produced from these experiments is also sparse due to the cost in preparing the samples, performing the experiment and analysing the data.

The expression levels of many of the genes that are measured in the data sets are irrelevant to the distinction, for example between two disease sub-classes, and it is thought that only a few genes, acting in concert, are responsible for differences in sample behaviour. These genes are generally over-expressed in one type of sample and under-expressed in another (and vice versa). This is illustrated in Figure 1.1 which shows levels of expression of genes responsible for distinguishing between two sample

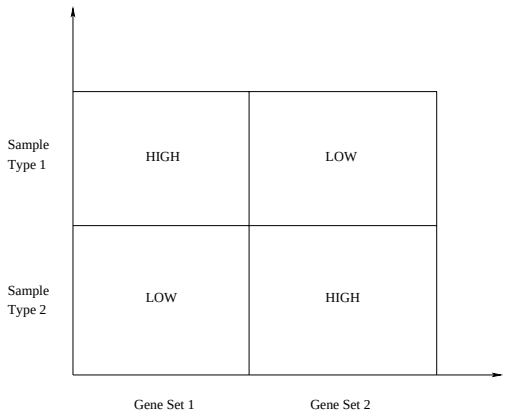

Figure 1.1: Ideal Differential Gene Expression to Determine Disease Classification

sets. Of course this representation is not so simple in reality, noise would mean that the distinction between types of samples would be somewhat blurred.

The goal of analysing the expression data then is to determine this small set of signal genes through the noise of the vast majority of the genes that have no contribution to the signal. This can be done by a variety of statistical means (Brazma & Vilo [5], Smyth, Yang & Speed [3]) and once this has been achieved a variety of machine learning algorithms, such as supervised learning and clustering techniques, can then be used to determine internal structures and relationships within the data and from those the predictions of characteristics of samples and genes. Putative gene regulatory networks can also be modelled using Boolean, Bayesian or relevance networks (Butte [6], Szallasi [7] & Friedman [8]).

1.2 An Example of Analysis of Microarray Data

The following work deals with the use of gene expression results from microarray experiments on tissues taken from patients having either acute lymphoblastic leukaemia (ALL) or acute myeloid leukaemia (AML). The data has been analysed by a number of research groups and illustrates the numerous different methods of analysis available to test hypotheses.

Golub *et al.* [11] attempted to find genes whose expression pattern was strongly

correlated with the class distinction. They developed a method called ”neighbourhood analysis” to find correlations between genes which had expression levels uniformly high in one class and low in another. Roughly 1100 genes were found by this method and the best 50 were chosen. Genes were also split into two equal groups, those with high expression in ALL and low in AML and vice versa. The set of genes were applied to the test set via self organising maps to find clusters of samples in this set. Good results, with around 85% precision, were obtained for the AML and ALL labels and also for the sub class of ALL data (T-cell and B-cell types). Poor results for clustering patient response were observed; this was thought to be due to extra genes, outside those being analysed, that affect drug performance.

Ben-Dor *et al.* [12] applied the CAST clustering algorithm implemented in the BioClust analysis software, together with a list of machine learning algorithms which included support vector machines (SVMs), boosting and nearest neighbours to classify the test set. A set of genes were selected from the 7129 genes available via a feature selection method called ’threshold number of misclassification’ (TNoM) which scores genes having quite different values in the AML and ALL classes. Roughly 150 genes were chosen via this method and the resulting classification into ALL and AML types was generally very successful with precisions up to 90%.

Xing, Jordan & Karp [13] used information gain ranking on each gene against the class label to select the ’best’ set of genes to use. Three classifiers were then used on the datasets with 360 genes being selected. The classifiers employed K nearest neighbours (KNN), a Gaussian model and a logistic regression algorithm. These gave excellent results with precisions up to 99% when compared to using a randomly selected gene set of the same size which gave averages of 36% error.

Getz, Levine & Domany [14] employed a novel coupled two way clustering (CTWC) technique, using ‘super-paramagnetic clustering’ (SPC), to find patterns in the data. First a subset of genes was selected by using only those genes that had an average expression, taken over all samples, that was over a certain threshold. In the clustering technique initially gene clusters using the samples as attributes were found, then only those genes were used to cluster samples. This loop continued until stable clusters of genes and samples were found. Very successful results were obtained that revealed

one gene cluster that clustered the AML and ALL samples into two sets. Further simulations, using different gene clusters, found two clusters in the AML sample set that differentiated patients based on the success of their treatment and likewise another gene cluster separated the ALL samples into T-cell and B-cell sub-types.

1.3 Types of Microarrays

There are two dominant type of DNA microarrays. The first is made by 'spotting' cDNA probes at specific locations onto a glass side. The cDNAs are usually polymerase chain reaction (PCR) products or expressed sequence tags (ESTs) that are 500 to 1,000 bases in length and their spacing on the slide allows for more than 10,000 spots to be placed on one chip. Typically there is one probe per gene. For technical reasons the information that is obtained from these microarrays generally gives the relative ratio or concentration of a given transcript when two conditions are compared. Samples of mRNA from two experiments are labelled with different dyes, pooled and hybridized to the microarray by competitive hydridisation.

The second type are high-density oligonucleotide arrays commercially available from Affymetrix. These microarrays contain between 11 and 20 pairs of oligonucleotide probes for a target RNA, for which one of the pair is the reverse complement to an ideally unique 25-mer in the RNA and the other contains a mutated middle base pair and serves as a measure of stray signal. Using the differences between these two intensities, the Affymetrix software used with the chips judges the reliability of each probe pair and calculates a qualitative and quantitative measurement of expression level. The construction of these chips is different to that used in the cDNA chips as the probes are synthesised *in situ* onto the array using a combination of light-directed combinatorial synthesis and photolithography.

The two different platforms have their own distinct advantages. Oligonucleotide arrays can be designed and made directly from sequence information without the need for physical intermediaries such as cDNA or PCR products. The large number of probes used per gene are used to increase detection redundancy, and their short length can be used to target the most unique regions of genes thereby improving the ability to

discriminate between, for example, related members of a gene family. Spotted cDNA arrays have the advantage of versatility as gene sets can be changed, expanded or modified to include alternatively spliced genes or newly discovered genes. There is also the possibility of constructing a custom array for any organism of choice and to spot unknown cDNAs for gene discovery purposes.

1.3.1 Bridging Microarray Platforms

Expression measurements made across the two microarray platforms are not directly comparable due to the differences in low-level hybridisation and analysis between the two techniques. One of the first studies (Kuo *et al.* [9]) to compare published measurements of, in theory, the same cancer cell lines measured from the two platform types demonstrated that this was the case. There is also the problem that one gene represented on one platform is not present on the other.

However, a method of bridging the data sets will be mooted in this work. In the case where the **same** sample set is analysed on the different platforms the genes common to both platforms can be used. By taking the expression measurement that shows the **least** favourable sign of a signal from the two possible measurements and collating these for the common set of genes between the platforms a set of gene expression levels will be produced. This gene set will then represent the best predictor as it has the likelihood of including any genes that may have a large systematic error but will now have little impact on any prediction made. Consider for example a gene on one platform producing a strong signal that has a poor signal on the other. By choosing the poor signal the risk of including a false strong signal from the first platform will be reduced. Of course the disadvantage of this method is that the size of the gene predictor set is reduced but the hope is to remove those gene expression measurement that may bring about false predictions.

1.4 Scope of This Work

In this work a new heuristic clustering algorithm (COSA by Friedman & Meulman [15]) that can cluster objects based on subsets of attributes that have significantly dif-

ferent signals between these objects will be the tool used to investigate microarray data. The claimed advantage of this technique is that it can ‘see through’ the noise of the vast majority of the gene and can find the small signal gene set without the use of any of the statistical approaches referenced above.

The next chapter introduces the COSA algorithms together with a summary of their derivation and the parameters used. Chapter 3 tests the algorithms on simple data then Chapter 4 expands the test to synthetic data sets that attempt to mimic microarray-like data sets. This task is important before applying the algorithm to any real data as its parameters need to be tested thoroughly to assess their likely impact when determining clusters. Chapter 5 then deals with the application of the algorithm to a real data set involving disease classification and prediction of patient survival times. It is here that the bridging technique introduced above will be tested alongside the more traditional method of analysing microarray platforms independently. Clusters of samples, sets of genes producing these patterns and patient survivability will be compared to existing ones found from previous studies.

Chapter 2

Clustering on Subsets of Attributes - COSA

2.1 Introduction

The COSA approach is to attempt to uncover distinct groups of objects that have similar joint values over a subset of attributes. The method can also be extended to focus on specific attribute values that are high or low, or both, in what is termed 'targeted clustering'. These properties of COSA thus make it suited to the analysis of gene expression values since, ideally, these have similar attribute value patterns (§1.1) to those described above. The recent paper of Friedman & Meulman [15] describes a method to implement this approach and this will be summarised in this chapter. The paper introduces two different versions of COSA: COSA1 that can be used to calculate so called COSA distances, or better described as dissimilarities, that can then be used with a partition like clustering algorithm such as Kmeans or fuzzy cmeans, and COSA2 which outputs dissimilarities that can be input to a hierarchical clustering algorithm. A version of COSA2 was provided as an executable that could be used in the *R* statistical software package running on Linux and Windows platforms. The intention for this work was to code both COSA1 and COSA2 algorithms using Matlab. This would then give platform independent code for both algorithms, the ability to input different distance measures into the algorithms (which was imperative for the bridging technique)

and to provide versions of the algorithms that could be used outwith any external package such as *R*. The existence of an executable for COSA2 provided a sound baseline for testing the Matlab version.

2.2 Summary of COSA

2.2.1 Attribute Value Data and Selection

The principal aim of clustering N objects is to produce subgroups of objects such that those objects in each group are more similar to one another than to those from an alternative group. If N_l is the number of objects assigned to the lth group and W_l is a weight applied to that group then this aim can be defined as minimising a criterion Q such as the weighted average over all groups $G_l(1 <\leq l \leq L)$, of the within group mean distance, D_{ij} between pairs of objects assigned to the same group, i.e.

$$Q = \sum_{l=1}^{L} \frac{W_l}{N_l^2} \sum_{G_l(i)} \sum_{G_l(j)} D_{ij}. \tag{2.1}$$

Objects $\mathbf{x_i} = (x_{i1}, \ldots, x_{ik}, \ldots, x_{in})$, are usually described by a set of n measured attributes and when this is the case the value of D_{ij} can be expressed as a weighted average of the individual attribute distances d_{ijk}, such that

$$D_{ij} = \sum_{k=1}^{n} w_k d_{ijk} \tag{2.2}$$

where

$$d_{ijk} = \delta_{ijk}/s_k, \tag{2.3}$$

with

$$\delta_{ijk} = |x_{ik} - x_{jk}| \quad \text{or} \quad |x_{ik} - x_{jk}|^2 \tag{2.4}$$

and

$$\{w_k \geq 0\}_l^n \quad \text{with} \quad \sum_{k=1}^{n} w_k = 1. \tag{2.5}$$

The value of s_k in 2.3 above provides a scale for measuring the spread or dispersion of the $\{x_{ik}\}_{i=1}^{N}$ values over all objects and is often taken to be

$$s_k = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \delta_{ijk} \tag{2.6}$$

which can be approximated as

$$s_k \approx IQR(\{x_{ik}\}_{i=1}^{N})/1.35 \tag{2.7}$$

when considering robustness measures to counteract any outliers in the data and where IQR is the interquartile range.

The weights w_k can be chosen to reflect a prior knowledge that particular attributes are more relevant than others to clustering the objects - the goal of feature selection, or to give the same influence when set to $\{w_k = 1/n\}_{k=1}^{n}$. Using this approach clusters of objects that *simultaneously* have small dispersions on all, or a subset, of the attributes will be found where the dispersions S_{kl} are given by

$$S_{kl} = \frac{1}{N_l^2} \sum_{G_l(i)} \sum_{G_l(j)} d_{ijk}. \tag{2.8}$$

or when using medians for robustness,

$$S_{kl} = \frac{1}{N_l} \sum_{i \in G_l} median\{d_{ijk}\}_{j \in G_l} \tag{2.9}$$

It can be shown that clustering based on using the distances in equation 2.2 with equal or different attribute weights minimises the arithmetic mean of the attribute dispersions within each cluster when the criterion Q is being minimised.

2.2.2 Clustering on Different Subsets of Attributes

Although the goal of feature selection can be helpful, clustering on different subsets of attributes cannot be achieved by this approach. It is achieved however by defining a separate attribute weighting $\mathbf{w}_l = \{w_{kl}\}_{k=1}^{n}$ for each grouping G_l where

$$\{w_{kl} \geq 0\}_{l}^{n} \quad \text{and} \quad \sum_{k=1}^{n} w_{kl} = 1. \tag{2.10}$$

and then minimising Q.

This approach however will yield to solutions that put a maximal unit weight on the attribute with the smallest dispersion within each group G_l and the groups will thus cluster on single attribute. To overcome this problem a penalty incentive involving the negative entropy of the weight distribution for each group is introduced, this being

$$e(\mathbf{w}_l) = \sum_{k=1}^{n} w_{kl} \log w_{kl}. \tag{2.11}$$

The minimum for the above function occurs when the weights are equal and is correspondingly larger as the weights become more unequal. Using this approach the distances D_{ij} are transformed to

$$D_{ij} = \sum_{k=1}^{n} (w_{kl} d_{ijk} + \lambda w_{kl} \log w_{kl}) + \lambda \log n \tag{2.12}$$

and the solution for the weights w_{kl} when minimising Q now becomes,

$$w_{kl} = \frac{\exp(-S_{kl}/\lambda)}{\sum_{k=1}^{n} \exp(-S_{kl}/\lambda)} \tag{2.13}$$

where increased weight is placed on attributes with smaller dispersions within each G_l group and this increase is controlled by the meta-parameter λ. The λ parameter can then be considered as a scale parameter in this heuristic algorithm. Setting it to zero places all weight on the attribute k with the smallest dispersion S_{kl}, whereas setting it to infinity forces all attributes to be given equal weight within each cluster G_l. It therefore has to be chosen with care when applying COSA a data set.

It can be shown that a new Q criterion can now be derived by inserting (2.12) and (2.13) into (2.1) where

$$Q = \sum_{l=1}^{L} W_l \left[-\lambda \log \left(\frac{1}{n} \sum_{k=1}^{n} \exp(-S_{kl}/\lambda) \right) \right] \tag{2.14}$$

and minimising this function actually minimises the inverse exponential mean of the attribute dispersions with the scale parameter λ, S_{kl}/λ, within each cluster.

2.2.3 Minimising the Clustering Criterion

The global minimisation of the above Q criterion is probably NP hard and is solved by a dual optimisation method as follows:

1. Initialise all weights to $w_{kl} = 1/n$.
2. Define interpoint distances D_{ij} using equations 2.15 or 2.16 below.
3. Apply a clustering algorithm to assign clusters based on these weights.
4. Calculate new weight values based on equations 2.8, 2.9 and 2.13.
5. Calculate new distances based on these new weight values and loop back to item 2 and repeat until the weights converge.

The distances D_{ij} are defined in either of two ways.
Distance one is given by

$$D(1)_{ij} = D_{ij}^{\eta}\left(\max(w_{k,l_i}, w_{k,l_j})\right) \tag{2.15}$$

known as the *maximum weight* distance and Distance two, the *maximum whole* distance is given by

$$D_{ij}^{2} = max(D_{ij}^{\eta}[w_{k,l_i}], D_{ij}^{\eta}[w_{k,l_j}]). \tag{2.16}$$

The terms D_{ij}^{η} are chosen to be based on a weighted inverse exponential mean (§2.2.2) of d_{ijk} with a scale parameter η i.e.

$$D_{ij}^{\eta} = -\eta \log \sum_{k=1}^{n} w_k e^{-d_{ijk}/\eta}. \tag{2.17}$$

The above choice for D_{ij}^{η} is based on the fact that the initial weights $\mathbf{W} = /1/n/$ are probably far from their global minimising values and the likelihood of converging to a suboptimal solution of 2.14 is very high. A homotopy strategy is introduced such that these distances evolve from their initial value of

$$D_{ij}^{\lambda} = -\lambda \log \sum_{k=1}^{n} w_k e^{-d_{ijk}/\lambda} \tag{2.18}$$

to their final value of

$$\lim_{\eta\to\infty} D_{ij}^{\eta} = \sum_{k=1}^{n} w_k d_{ijk} \tag{2.19}$$

which are the ordinary distances from equation 2.2 via the use of a homotopy parameter α where $\eta = \eta + \alpha.\lambda$. This method ensures there is a high probability of retreating out of a suboptimal solution to 2.14 at the start of the optimisation but ensuring that this probability is lowered as the process evolves and in a way mimics the process of simulated annealing.

Two important considerations must be made here. First too large a value of η will cause too fast an evolution of the distances to their normal forms with a result that ordinary clustering will result, and secondly too high a value of λ will again cause COSA to approximate to ordinary clustering by placing equal weight on all the attributes. The authors point out that if the clustered groups tend to concentrate on small subsets of the data the value of α should be taken to be small ($\alpha \leq 0.1$) causing a slow evolution to the ordinary distances.

This homotopy parameter is then, once again, to be used with care when performing a data mining exercise with COSA.

2.2.4 COSA1 Algorithm Based on Kmeans

An algorithm called COSA1 is now given that uses the above methods and implements a simple Kmeans algorithm to determine the clusters from the COSA dissimilarities D_{ij}.

1. Initialise $\mathbf{W} = \{1/n\}$, $\eta = \lambda$, number of cluster partitions c.
2. Randomise partition centres.
3. Start loop of COSA1.
4. Start Kmeans loop.
5. Compute the distances, $\{D_{ic}\}_1^N$ from each point to each centre with equations 2.15 or 2.16 and using 2.17.

6. Form clusters based on the Kmeans algorithm.
7. Form new cluster centres based on these clusters.
8. Return to step 4 until centres converge.
9. Compute weights $\mathbf{W} = \{\mathbf{w}_l\}_1^L$ equations 2.8 & 2.13.
10. Set $\eta = \eta + \eta.\lambda$.
11. Return to step 3 until **W** stabilises.

2.2.5 COSA2 Algorithm for Hierarchical Clustering

An alternative to the COSA1 algorithm is one in which each object is treated as a separate entity, not in a partitioned cluster, and the dissimilarities D_{ij} can be calculated for all objects. These values can be input into a standard hierarchical clustering algorithm (e.g. single, average, complete etc…linkage) and the usual dendrogram displayed. Clusters can then be extracted by eye or some automated technique by drilling down from the root of the dendrogram. To enable COSA to determine the dissimilarities D_{ij}, values of the dispersion, S_{ki} and hence the weights, w_{ki} attached to each attribute for each object must be sought. This is achieved by considering, for each point, the nearest neighbours that are closest to it in terms of the COSA dissimilarity D_{ij}. If $KNN(i)$ is the K closest objects to i based on D_{ij} then

$$KNN(i) = \{j | D_{ij} \leq d_{i(K)}\} \tag{2.20}$$

where $d_{i(K)}$ is the Kth order statistic of $\{D_{ij}\}_{j=1}^N$ sorted in ascending values. A value for the dispersion for each object's attribute can then shown to be

$$S_{ki} = \frac{1}{K} \sum_{j \in KNN(i)} d_{ijk} \tag{2.21}$$

or for robustness, especially when dealing with real data that has outliers, the median of the d_{ijk} values can be used. From this a value of the weights w_{ki} for each object can be deduced as

$$w_{ki} = \frac{\exp(-S_{ki}/\lambda)}{\sum_{k=1}^{n} \exp(-S_{ki}/\lambda)} \tag{2.22}$$

where the matrix of weights $\mathbf{W}$ is now an n by N matrix representing a weight for each object for each of its attributes.

From these derivations an algorithm, COSA2, for calculating COSA dissimilarities between all sets of objects for use in a hierarchical clustering algorithm can be given:

1. Initialise: $\mathbf{W} = \{1/n\}; \eta = \lambda$.
2. Start COSA2 Loop.
3. Compute dissimilarities D_{ij} - equations 2.15 or 2.16 using 2.17.
4. Compute the K nearest COSA neighbours for each point - equation 2.20.
5. Compute weights $\mathbf{W} = \{w_{ki}\}$ - equation 2.21 & 2.22.
6. Update $\eta = \eta + \alpha.\lambda$.
7. End COSA2 loop, item 2, when $\mathbf{W}$ stabilises.
8. Output the dissimilarities D_{ij}.

It can also be shown that the quality criterion Q for minimisation when using this approach for hierarchical clustering can be given as the following expression

$$Q = \sum_{1}^{n} \left[\frac{1}{K} \sum_{j \in KNN(i)} D_{ij} + \lambda \sum_{K=1}^{n} w_{ki} \log(w_{ki}) \right]. \tag{2.23}$$

2.2.6 Targeted Clustering

Instead of concentrating on distinct groups of objects that have similar joint values on subsets of the attributes the distances d_{ijk} between objects for a particular attribute k can be modified to reflect similarity to a desired value as well as between the objects themselves. For example it may be of interest to find groups of samples where subsets of genes have similar values and are either close to a high or low, or both high or low

expression value. These would reflect the goal of finding samples having genes that are differentially expressed (high or low) with dual targeting or only either high or low co-expression with single targeting.

Single targeting can be achieved by modifying the distances on an attribute, k having a target value t_k as

$$d_{ijk} = \max[d_k(x_{ik}, t_k), d_k(x_{jk}, t_k)] \tag{2.24}$$

where

$$d_k(x_{ik}, t_k) = |x_{ik} - t_k|/s_k \tag{2.25}$$

where s_k is defined by equation 2.6 or 2.7. Here the distances d_{ijk} will be small if both the x_{ik} and x_{jk} values are close to each other *and* close to the target value t_k.

Dual targeting can be achieved by setting

$$d_{ijk}(t_k, u_k) = \min[d_{ijk}(t_k), d_{ijk}(u_k)] \tag{2.26}$$

where u_k is some upper target value on the kth attribute and t_k is a corresponding lower target value. With this approach the distances d_{ijk} are small only when x_{ik} *and* x_{jk} are either *both* close to t_k *or both* close to u_k.

By setting the values of t_k and u_k to the values near the maximum and minimum data values of the attributes (a good choice would be the 5% and 95% percentiles, though this can be varied) the COSA clustering algorithm should then seek clusters of attribute values near these levels while ignoring potentially more dominant clusters with only moderate attribute values.

2.2.7 Missing Values

Any real world data mining task is usually plagued by problems with data that is missing. The task of representing this data by using only the given data is the subject of a huge field of research and has many avenues of approach. A number of references detailing this work is given by Smith [10]. The COSA authors recommend an approach where the weight w_k of attribute k is assigned a value of zero in the distance calculation

of equation 2.17 if its value is missing on *either* object i or j. If two objects have no non missing value in common then they are assigned an infinite distance so that they are not placed in the same cluster.

However this approach can lead to situations where objects that appear to be in the same cluster are actually forced to be apart. A simple case would be a scenario where one object has a full set of attributes, a second only half but having the other very close to the first and similarly a third but with the other half of attributes missing. All three objects look identical to each other but this approach of modelling the missing data would lead to the last two objects being forced apart.

For this reason it was decided not to follow this approach but instead to analyse the data before input to COSA to find the best method of modelling and replacing the missing data.

2.2.8 Importance Measure for Attributes

Once a cluster is found by COSA the set of attributes that have the most relevance to the cluster can be found by using the value of the dispersion S_{kl} given by equations 2.8 or 2.9 over all objects in the cluster l for a particular attribute k. An attribute score I_{kl} can be given as

$$I_{kl} = [S_{kl} + \varepsilon]^{-1} \tag{2.27}$$

where ε^{-1} represents a maximum score (when the dispersion S_{kl} is zero). This value represents the inverse measure of the spread of x_{ik} values within a group G_l relative to the spread over the whole set of objects. Of course this is the very criterion by which COSA decides on how to cluster objects on subsets of such attributes. The larger the value of I_{kl} the greater the likelihood it will contribute to the resulting clustering pattern. It can be calculated, for large data sets, by the quick calculation

$$I_{kl} = IQR_k(G_l)/IQR_k(N) \tag{2.28}$$

where IQR is the interquartile range.

Such values then are useful, once the clustering has been revealed, to decide on, for example, the most important genes that contribute to a cluster of patient samples found by COSA.

2.3 Summary

The COSA algorithms have been introduced as a potential method to extract gene expression signals that could determine, for example, disease classifications for patient samples resulting from data from microarray experiments. Their advantage seems to lie in the fact that, given the whole dataset of all genes the very small set of genes that control such classifications may be found and that such genes can then be listed by an importance score.

Both the COSA1 and COSA2 algorithms described here have been coded using Matlab and were tested on very basic data sets as described in Chapter 3. Chapter 4 deals with the extension of tests by the enlargement of the data sets to represent fictitious microarray data. The tests included the varying of various parameters that control these heuristic algorithms to gauge their effect. Once these effects were known the routines were exposed to actual real microarray data as detailed in Chapter 5.

Chapter 3

COSA Tests on Simple Data

To understand the mechanisms of the COSA algorithms, and to evaluate their potential performance three simple data sets were constructed. The aim of this was to discover any potential flaws in either COSA1 and/or COSA2 before the algorithms were applied to the more complicated microarray type data. Both *R* and Matlab versions of the COSA2 algorithm and the Matlab version of COSA1 were run in these tests.

3.1 Test Data Set 1

The first test data consisted of a set of twenty two-dimensional points, split so that one half of the data set was aligned with one axis and the other half of the set aligned to the other axis. The COSA algorithms should then form clusters of each of these subsets of the data sets since their dispersions in one of the dimensions will be small and in the other large.

The data set is illustrated in Figure 3.1 where a normal Kmeans algorithm has been run to find two clusters shown whose centres are represented as the large diagonal crosses. This shows that the two sets of data points cannot be split into the two sets of horizontal and vertical data via this method.

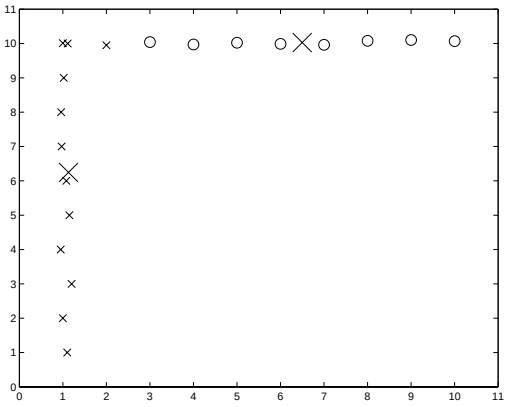

Figure 3.1: Test Data Set 1 With Kmeans

3.1.1 COSA1 Performance

Here COSA1 found the clusters as expected and inspection of the weights assigned to each attribute in each of the two clusters (found from equations 2.8, 2.9 & 2.13) found high weights associated with attributes with low dispersions in the clusters and vice versa.

3.1.2 COSA2 Performance

Again COSA2 managed to split the data points as expected with strong weights found for attributes with low dispersions in the clusters and vice versa (equations 2.21 & 2.22).

Plots of the clusters found, together with their centres are shown in Figure 3.2. One interesting point found is that the two points which can be considered to be in either cluster (the points in the top left of the plots) are switched between COSA1 and COSA2. For these points in COSA2 the weights, w_{kl} (equation 2.22) were not as strong (0.78/0.21) as the other weights which were close to unity/zero.

One disadvantage of the COSA algorithms can be seen, however, from the choice of these data sets. These sets were chosen so that objects would cluster as they are parallel to the coordinate axes and thus the dispersions of the subsets of attributes forming each cluster is small. Under a rotational transformation of the axes this would

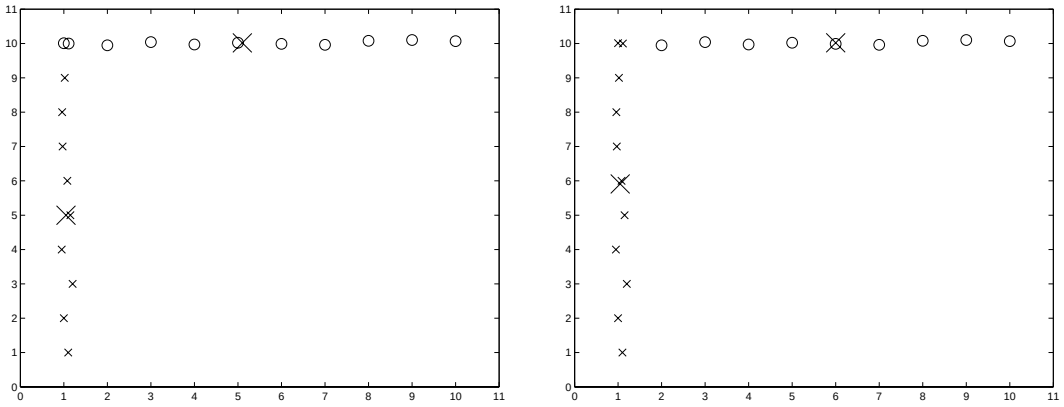

Figure 3.2: Test Data Set 1 With COSA1 and COSA2

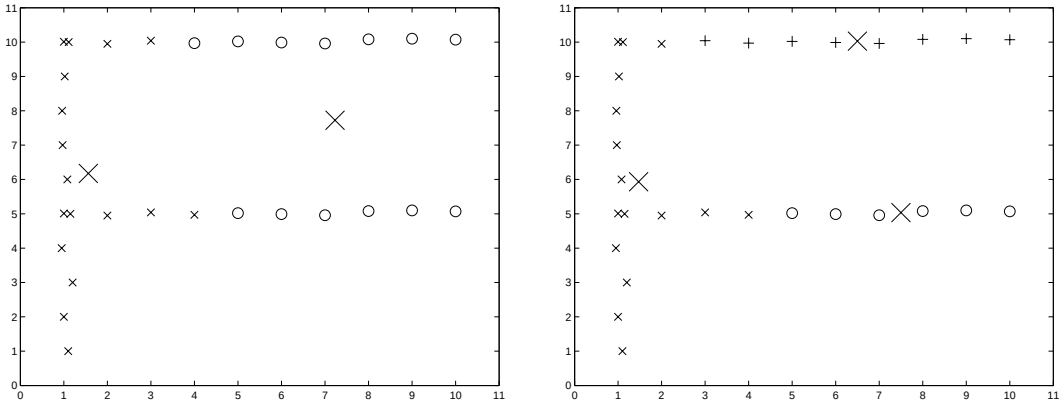

Figure 3.3: Test Data Set 2 With Kmeans

not be the case. This is then one immediate drawback of the COSA algorithms in that they are sensitive to such transformations.

3.2 Test Data Set Two

The test data set was then expanded by adding a second row of points that ran parallel to the x-axis. This data set is shown in Figure 3.3 where a normal Kmeans algorithm has been run with two and then three preferred clustering partitions requested.

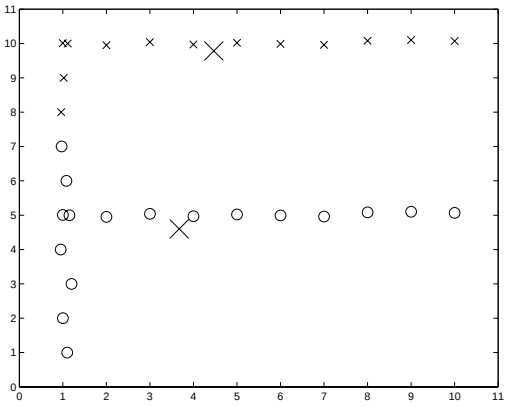

Figure 3.4: Test Data Set 2 With COSA1 - 2 Centres Requested

3.2.1 COSA1 Performance

It was at this position where COSA1 ran into problems. Figure 3.4 shows the result of COSA1 when two clusters were requested.

Here the clustering is not as expected, since the two clusters would be expected to be one set consisting of the vertical points and the other the two sets of horizontal points. This would mean the first set of points would have a low dispersion with respect to the *x* attribute and high for the *y* attribute. Correspondingly the other cluster would have high dispersion on the *x* attribute and a smaller dispersion on the *y* attribute (though not as small as in the case of test data 1 since the points occur at two distinct values of *y*). Instead the clustering found a mixture of both sets of these points. Inspection of the weights associated with each attribute in each cluster showed that the weight given to the *y* attribute was always greater then the corresponding value attached to the *x* attribute. This discrepancy was thought to be due to the fact that only two centres were requested where perhaps three was a more natural choice.

A further test was then performed with the number of preferred clusters set to three. This simulation produced favourable clusters (shown in Figure 3.5) but the algorithm did not converge and constantly flicked between the clusters shown in the figure (in the lower row of horizontal data these points can be seen).

In order to ascertain the reason for this the quality of the clustering (simply the sum of the within cluster COSA dissimilarities between each point in a cluster and

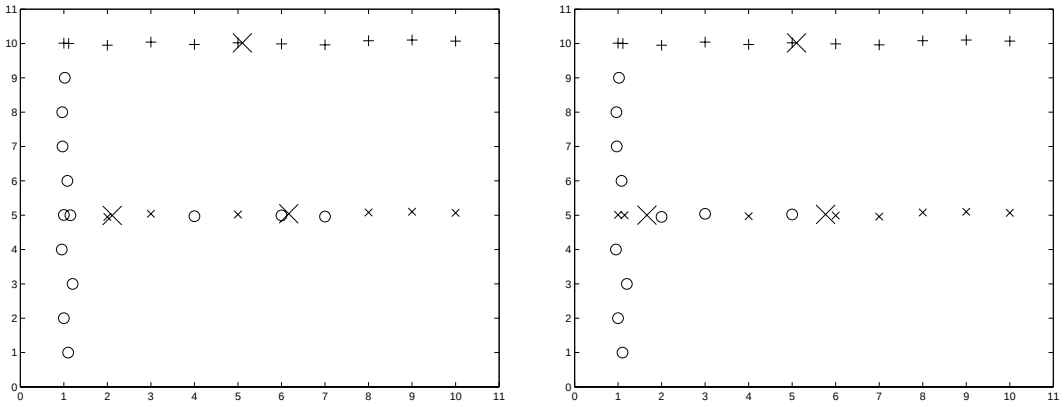

Figure 3.5: Test Data Set 2 With COSA1 - 3 Centres Requested

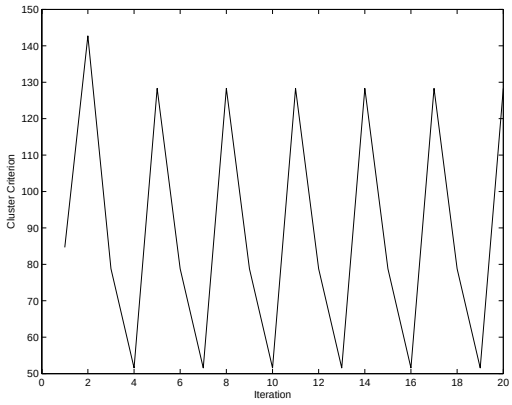

Figure 3.6: Quality of Clustering Associated with COSA1

their centre) was investigated at each iteration of the Kmeans algorithm. The outcome is shown in Figure 3.6.

The oscillations observed from this test were also found when investigating the differences in the centres of each cluster as the Kmeans algorithm within COSA1 proceeded. It is clear then that some problem exists with the algorithm as the data sets are made more complex. One possible explanation is that the dispersions and resulting weights are calculated on clusters that are defined by the hard rule of the number of partitions requested *before* the COSA1 algorithm has converged. This is not the case for COSA2 where the dispersions are calculated on clusters found by the K nearest neighbour methods described in §2.25 which allows for fuzzy (or overlapping) clus-

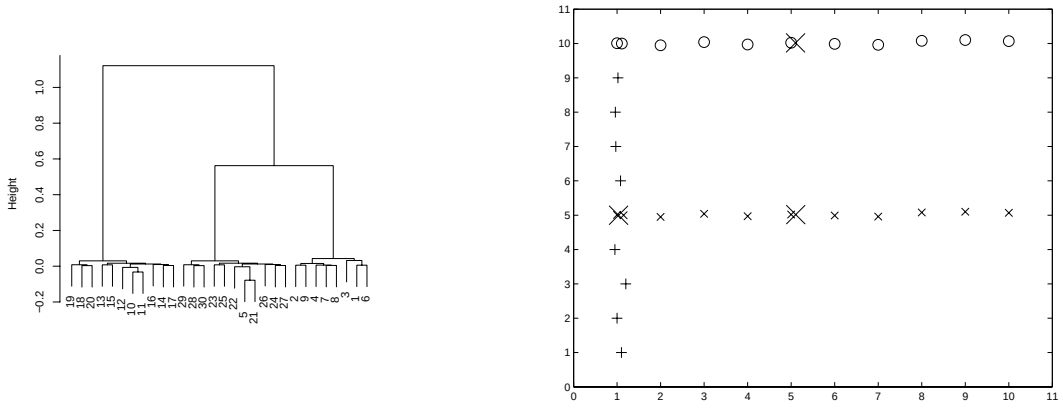

Figure 3.7: Test Data Set 2 With COSA2

ters of points and the final clusterings are not decided until COSA2 has converged and the resulting dissimilarties are then investigated.

3.2.2 COSA2 Performance

COSA2 managed to separate the data into three distinct clusters and this is shown in Figure 3.7 where the dendrogram, using average linkage on the COSA dissimilarities, is plotted alongside the clusters with their centres. The labelled points are numbered 1-10 for the vertical cluster, 11-20 for the upper horizontal and 21-30 for the lower horizontal. Here COSA2 has split the points into two well defined clusters initially (the upper horizontal one having low dispersion on y, the remaining horizontal and vertical sets having high dispersions both attributes). This second cluster itself has two well defined sub-clusters, the middle horizontal and vertical elements.

Inspection of the weights associated with each attribute for each cluster showed an excellent correlation between attribute dispersion and cluster, i.e. large dispersion and small weight and vice versa.

3.3 Test Data Set Three

A final more complex data set was then constructed. Again a two dimensional data set was used consisting of N objects, of which S represent a signal against the background

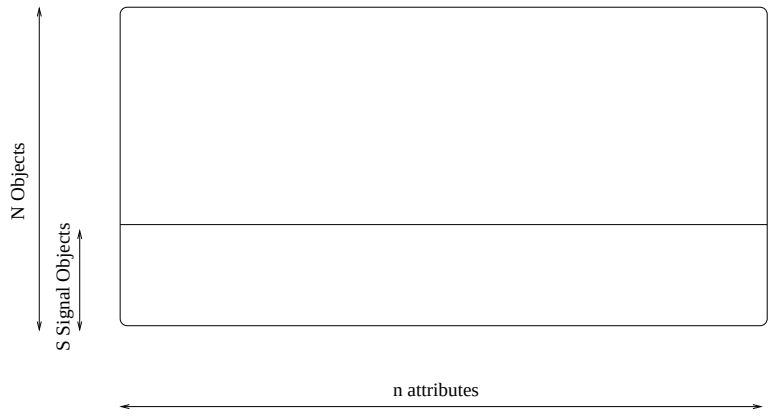

Figure 3.8: Construction of Data Sets for Test 3

of the random noise of the $N-S$ objects. This was modelled by drawing $N-S$ objects from a n dimensional normal distribution with μ set to zero and σ set to unity with the signal drawn from another normal distribution with μ set to 1.5, 2.5, 3.5 and 4.5 and σ set to 0.2 to form the set. The whole sample set was then was standardised to have zero mean and unit variance on all attributes. The construction of these data sets are illustrated in Figure 3.8 with N being set to 100, S to 15 and n to 2 for these tests.

The aim of this experiment was to test the ability of each clustering algorithm to separate the signal from the noise. This should have been easier with the data sets with μ set to 4.5 than with 1.5 since by virtue of the standardisation of the data the signal points alone have smaller variance when set with an initial higher offset average. The results of the clustering would give a guide to the strengths of each algorithm since the small dispersion of both attributes belonging to the signal data should be recognised by both COSA1 and COSA2 as being distinct from that of the background noise which had greater variance on its attributes. The power of the clusterings should also be seen to increase as μ is increased. In the first instance a normal Kmeans algorithm was ran on all four of the data sets and the results are shown in Figure 3.9.

Kmeans was successful at separating the signal with μ set to 4.5 and 3.5, five false positives were obtained at $\mu = 2.5$ and many at $\mu = 1.5$. This result highlighted the properties of the Kmeans algorithm, clusters that are more well defined from each other (a greater distance apart and more tightly clustered) are easier to separate into

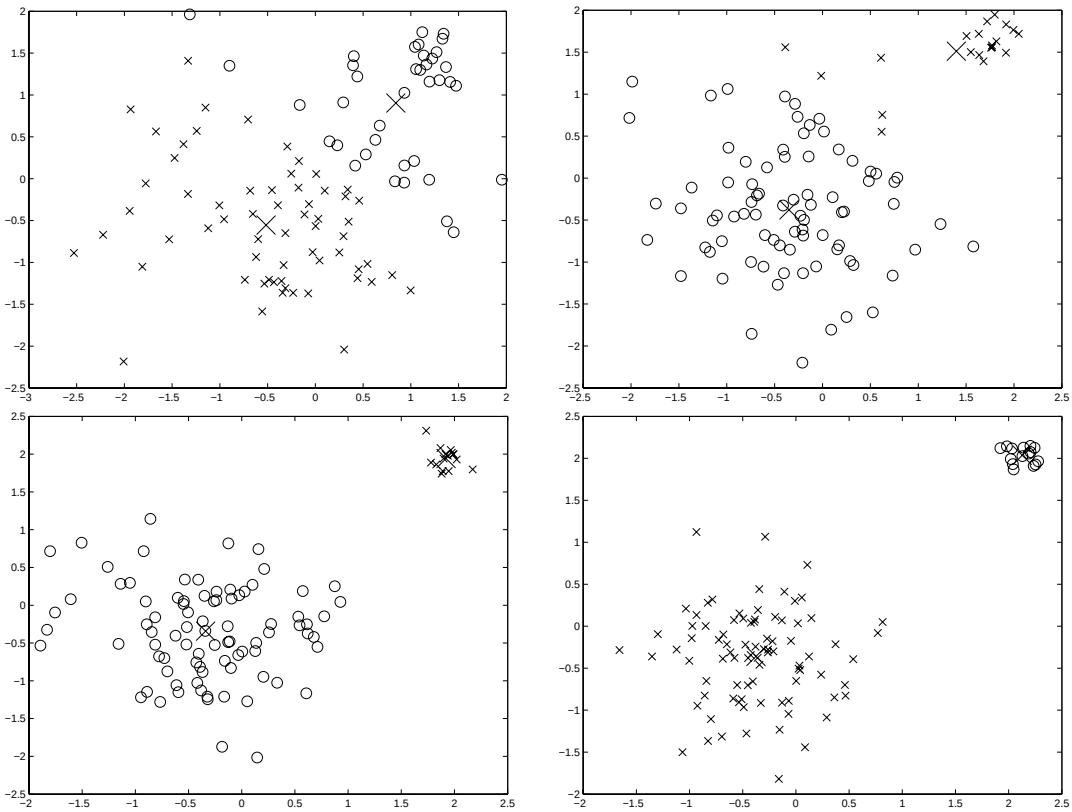

Figure 3.9: Test Data Set 3 with Kmeans

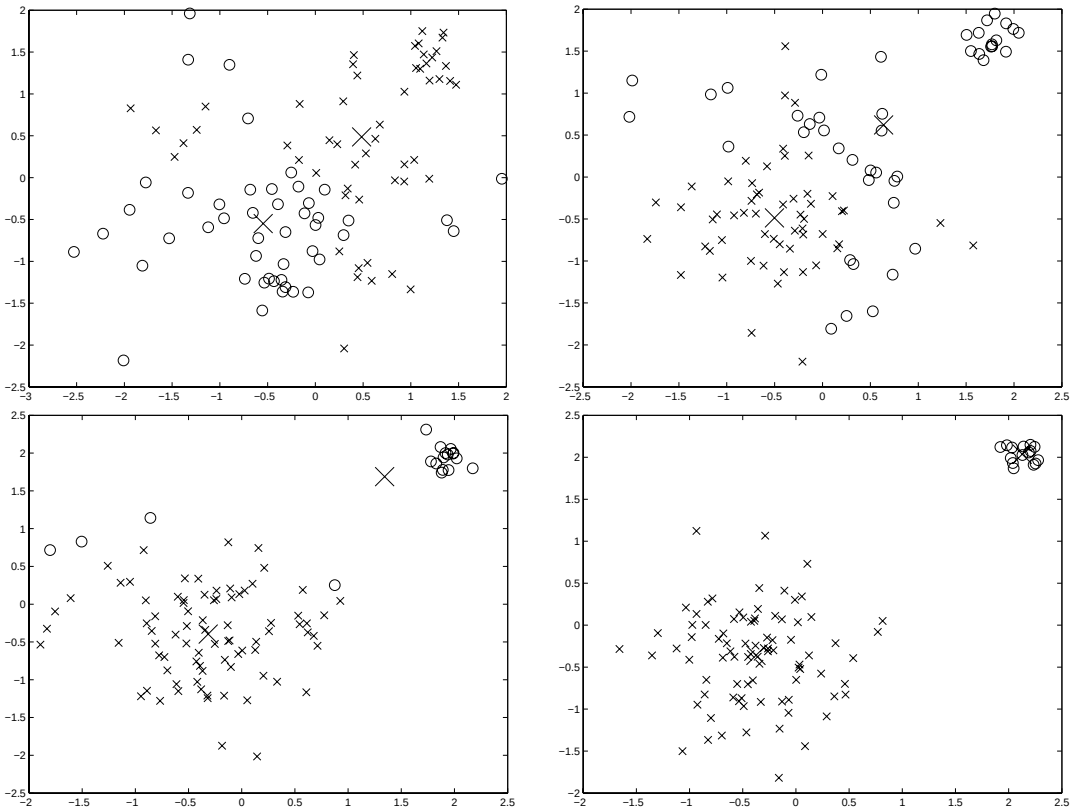

Figure 3.10: Test Data Set 3 with COSA1

distinct groups.

3.3.1 COSA1 Performance

The results using COSA1 on the data sets are displayed in Figure 3.10.

COSA1 performed less ably, with, though full success at $\mu = 4.5$, four false positives found with μ set to 3.5 and many with μ set to 2.5 and 1.5. The oscillations that occurred with test data set 2 also occurred with the last three of these data sets.

3.3.2 COSA2 Performance

The results using COSA2 on the data sets are displayed in Figure 3.11.

COSA2 performed best with full separation of the signal when μ was set to 4.5 and 3.5, only one false positive appeared with μ set to 2.5 and only three when μ was set to

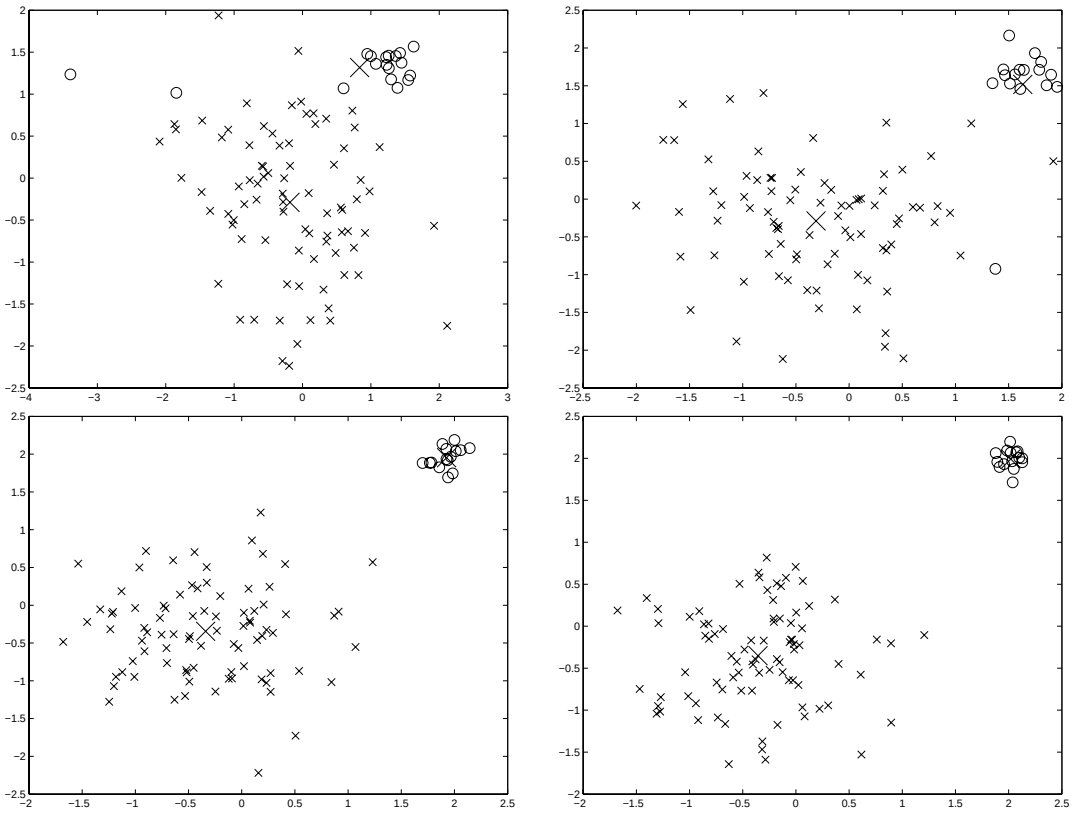

Figure 3.11: Test Data Set 3 with COSA2

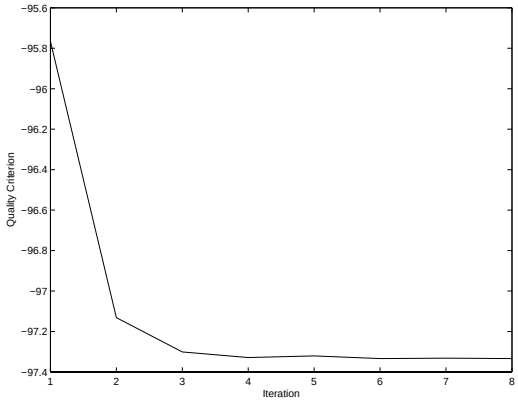

Figure 3.12: Quality of Clustering with Iteration Step Using COSA2

1.5.

The power of COSA2 could then be seen over Kmeans. The data set with μ set to 1.5 is hard to separate with Kmeans since the clusters are close together whereas COSA2 splits the data reasonably successfully since it is not searching for normal Euclidian distances between the points but dissimilarity values that measure low dispersions of attribute values of sets of data points.

The algorithm also converged to its solution very quicky, typically the number of iterations was between five and eight. A plot of the cluster quality criterion, represented by equation 2.23, was made for one of these simulations. This is shown in Figure 3.12. and shows that the minimisation of this quantity is achieved both smoothly and quickly.

3.4 Discussion

COSA1 and COSA2 were coded in Matlab and tested on a number of datasets. COSA2 proved to be excellent, predicting results well and being robust and fast. COSA1 proved less reliable, giving often poor results and proving unstable with the algorithm not converging due to oscillations caused when calculating the new cluster centres. The decision was then made to abandon COSA1 and keep COSA2 for the microarray dataset simulations to be described in the next two chapters.

Chapter 4

COSA2 Tests on Synthetic Microarray Data Sets

4.1 Introduction

The COSA2 routine, both the *R* version supplied by the authors of COSA and the version in Matlab written in this work were then tested on a synthetic dataset built to represent a typical data set of expression values resulting from microarray experiments. The idea here was to investigate the effect of the various parameters of the algorithm on any resulting clustering groups. Any important parameters that caused variations in these groups would be noted ready for the simulation experiments performed in the next chapter on real gene expression data sets. The tests would also reveal any differences between the Matlab and the *R* versions of the algorithm.

The data set to be constructed consisted of a group of N objects described by n attributes (genes) and was made up of two distinct parts, a signal and background noise group, the former of which should be revealed by a clustering algorithm. The signal group consisted of S objects randomly drawn from a n dimensional normal distribution with μ set to 1.5 and σ set to 0.2, and the background noise group constructed from a sample of $N - S$ objects drawn from an n dimensional normal distribution with μ set to zero and σ set to unity. Only the first no attributes of the signal group were taken from the former normal distribution, its other $n - no$ attributes were drawn from a normal

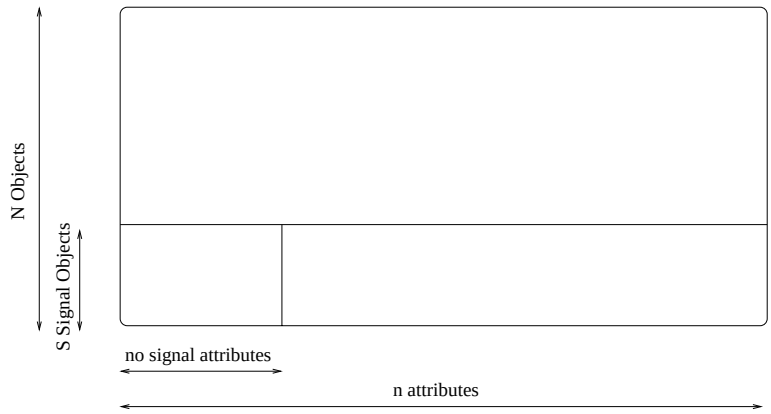

Figure 4.1: Construction of Synthetic Microarray Data

distribution similar to the latter distribution. After generation the pooled sample was standardised to have zero mean and unit variance on all attributes. The synthetic data set thus differed between the signal and noise groups on only the first *no* attributes. A figure illustrating this data set is shown in Figure 4.1.

4.1.1 The Power of the COSA2 Algorithm

Before the tests were run some simulations on the synthetic data were performed to illustrate the efficacy of the COSA2 algorithm. Synthetic data sets were generated with N set to 100, S set to 15, n set to 1000 and *no* set to 5, 10 and 20. Distances/dissimilarities between objects were calculated using normal Euclidian distance between attributes as well as both dual targeting and non-targeted distances using COSA2, were then calculated for each data set. An average linkage hierarchical clustering algorithm was then applied to these values. Figure 4.2 shows the results.

From the figure it can be observed that no clusters were produced using Euclidian distance whereas with COSA2, with no targeting set, a cluster started to appear with *no* set to 10 which then became more defined with *no* = 20. With targeted distance a cluster had already appeared when *no* was set to 5 and became a very distinct cluster with *no* = 20. It should also be noted that with targeted distances the cluster appears, not at the root of the dendrogram but among the leaf level since the algorithm is searching for a small subset of attributes that are close to a given target value as well to each other.

This fact then had implications for automatically finding clusters using dendrograms as the clusters chosen are those found coming straight off the root of the dendrogram. When using targeted distance with COSA2 the dendrograms would then have to be inspected by eye to check for any clustered objects of interest.

The important attributes that decided this clustering in the case of *no* being set to 10 could then be found using equations 2.27 or 2.28. Figure 4.3 shows the most important attributes and, as expected, these are grouped in the range 1 to 10.

A further plot, Figure 4.4, further illustrates this. Here histograms are plotted for the 'best' five attributes as found above, for the signal objects, noise objects and both signal and noise objects. It can be seen that the signal objects are defined by a narrow range of attributes values that drive the clustering of this object set, whereas the noise objects have a large variance around zero. The signal and noise objects taken together show some signs of a bimodal distribution.

4.2 COSA2 Tests

The following tests were then performed on this test data in order to test the effectiveness of COSA2 to split the signal group from the group of objects representing the background noise:

1. The effect of the strength of the signal (no/n) for non-targeted, single targeted (high or low) and dual targeted distances (high or low together).

2. The ability of normal clustering using a Euclidian distance with average linkage to separate the signal form the noise.

3. The effect of the signal to noise ratio (S/N).

4. The effect of the number of nearest neighbours K used to define clustering (§2.25).

5. The weighting scheme used to define the interobject dissimilarities (equations 2.15 or 2.16, §2.2.3).

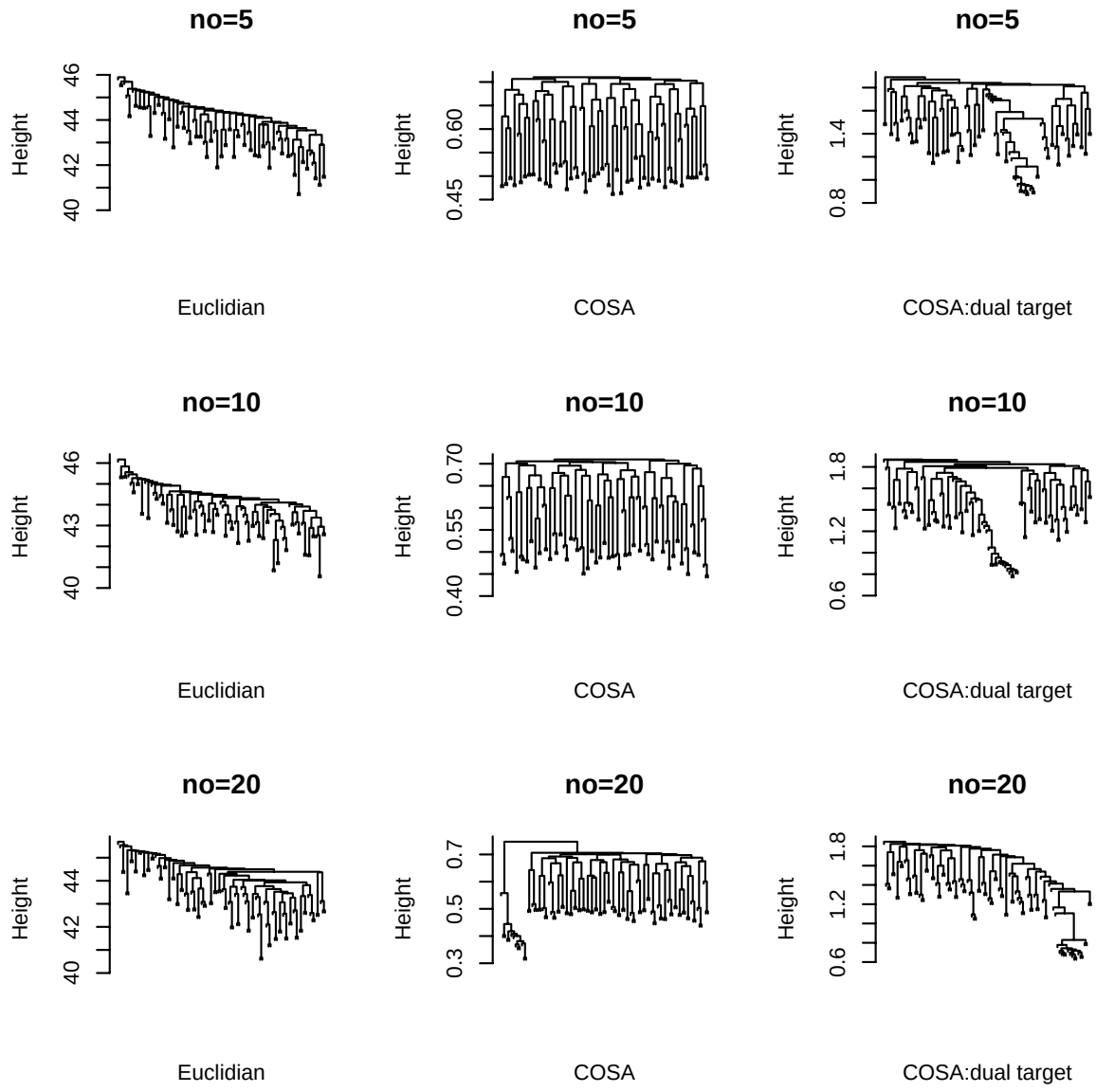

Figure 4.2: Average Linked Hierarchical Clusterings using Euclidian Distance, COSA2 & COSA2 Targeted Distances

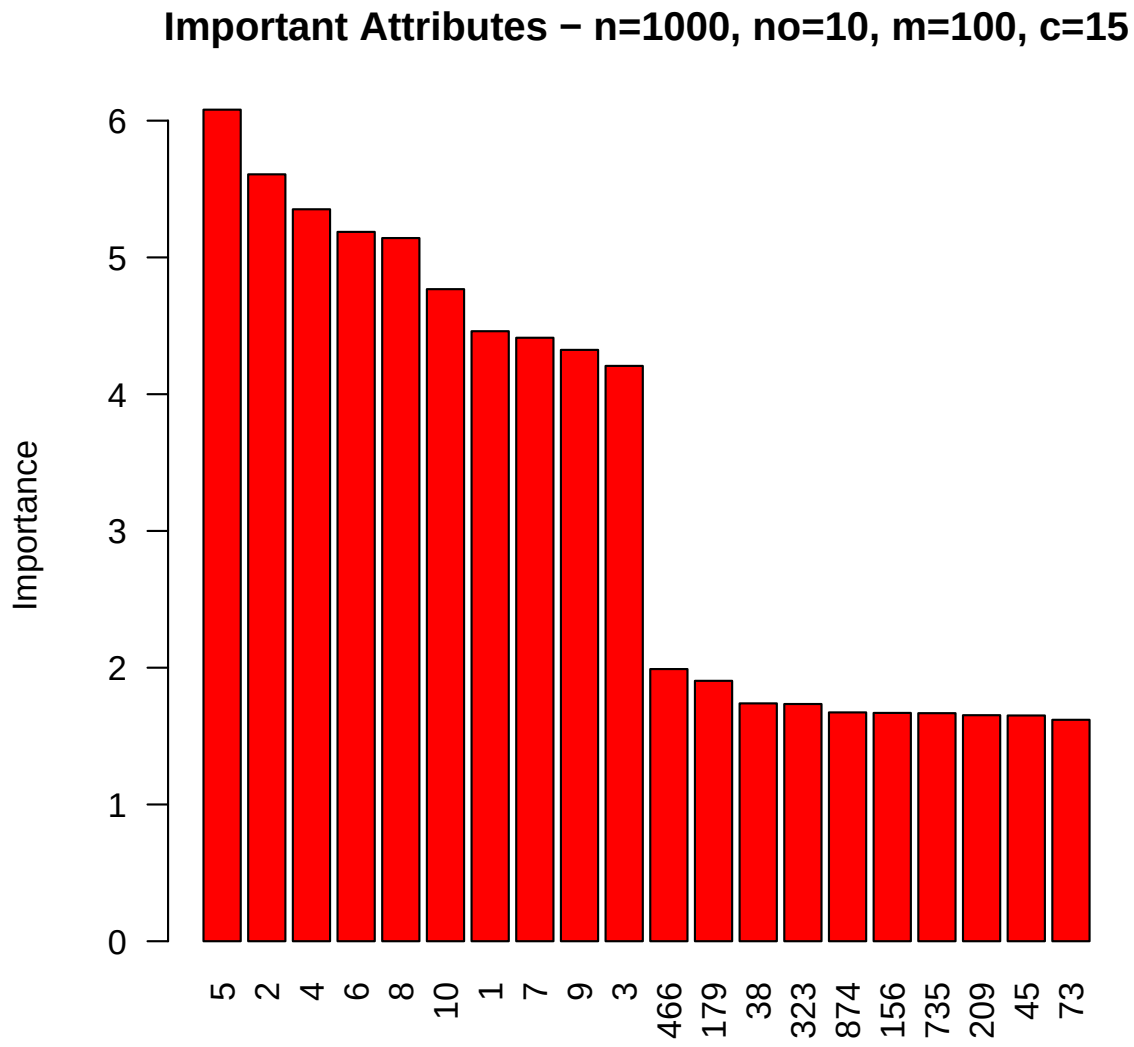

Figure 4.3: Important Attributes for Clustering in Figure 4.2 With *no* = 10

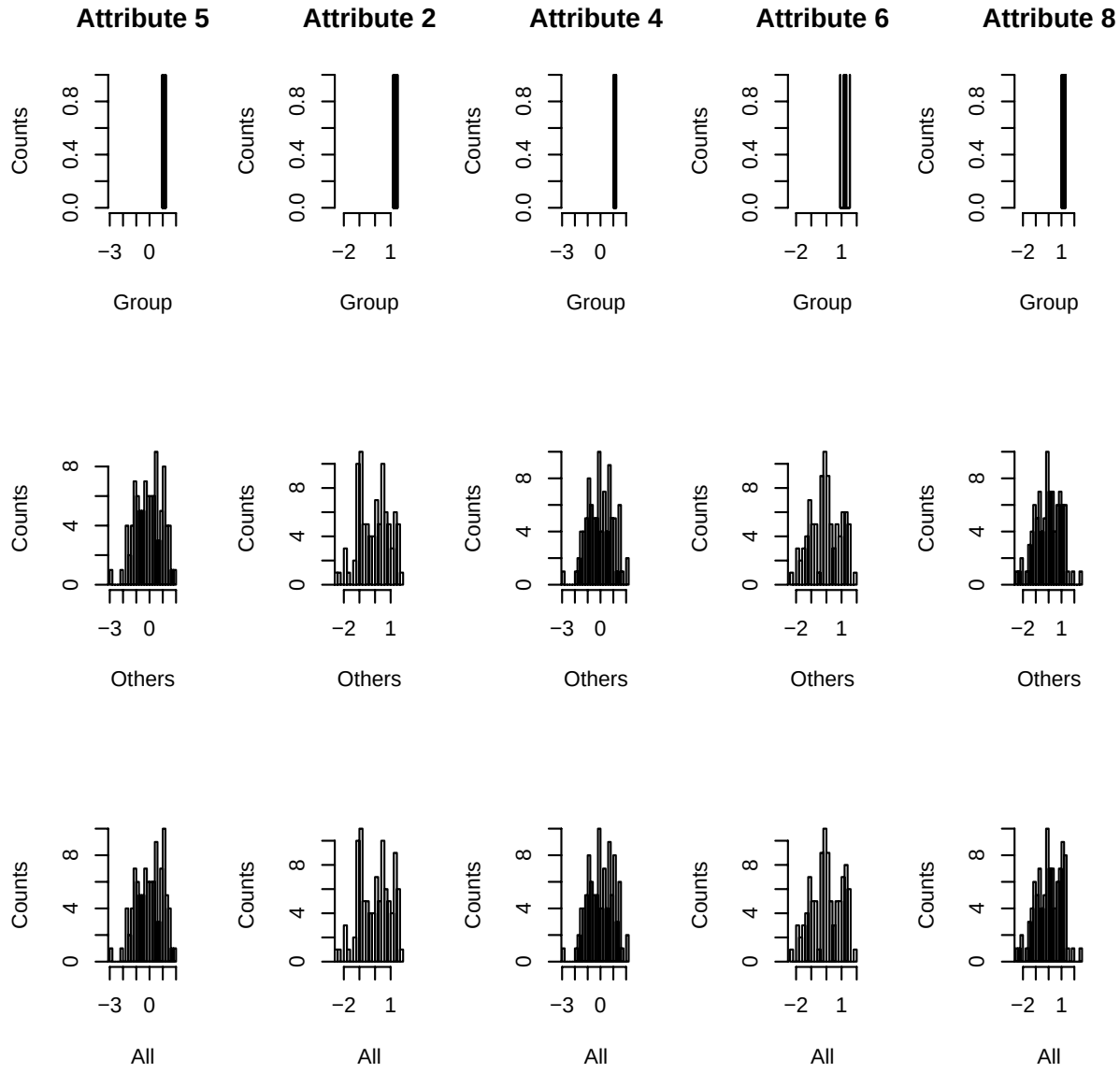

Figure 4.4: Histograms of the Important Attributes in Figure 4.3

6. The effect of the scale parameter λ (§2.2.2).

7. The effect of the homotopy parameter α (§2.2.3).

In these tests a number of synthetic data sets were produced for each parameter value of interest to give a statistical variation. Clusterings were found by applying an average linkage hierarchical clustering algorithm to the dissimilarities calculated from these data sets. Two clusters were then taken to be the those emerging from the root of the dendrogram. From these, values of the true positives (TPs) - those signal objects found in the cluster representing the signal, false positives (FPs) - noise objects found in the calculated signal cluster, true negatives (TNs) - noise objects found in the cluster representing the noise and false negatives (FNs) - signal objects that were placed in the calculated noise cluster. Values of precision (alternatively named as specificity or accuracy) = $\frac{TP}{TP+FP}$ and recall (alternatively termed sensitivity or coverage) = $\frac{TP}{TP+FN}$ could then be calculated from these confusion matrices.

These simulations were then repeated for the Matlab version of COSA2 coded for this work. Any significant differences between the precision and recall values were noted via a paired *t* test ($P < 0.05$).

In all test cases the number of attributes, n was set to 1,000 and the number of objects, N, set to 100.

4.2.1 Strength of Signal for COSA2

Here the parameters of COSA2 were set to $\lambda = 0.2$, $\alpha = 0.0$, the maximum weight dissimilarity type (equation 2.15), the value of the K nearest neighbours set to $\sqrt{N}$ and the number of signal objects, S, set to 15. The signal attribute number, *no* was then varied from close to zero up to N. The plots of precision and recall are shown in Figure 4.5. No values are shown after *no* = 35 as these values were 1.0 in each case. No significant differences were observed when run with the Matlab version of COSA2.

The simulations showed that the signal data can be retrieved with a signal attribute number of around 25 with this ability falling off rapidly when the number of attributes is around 15.

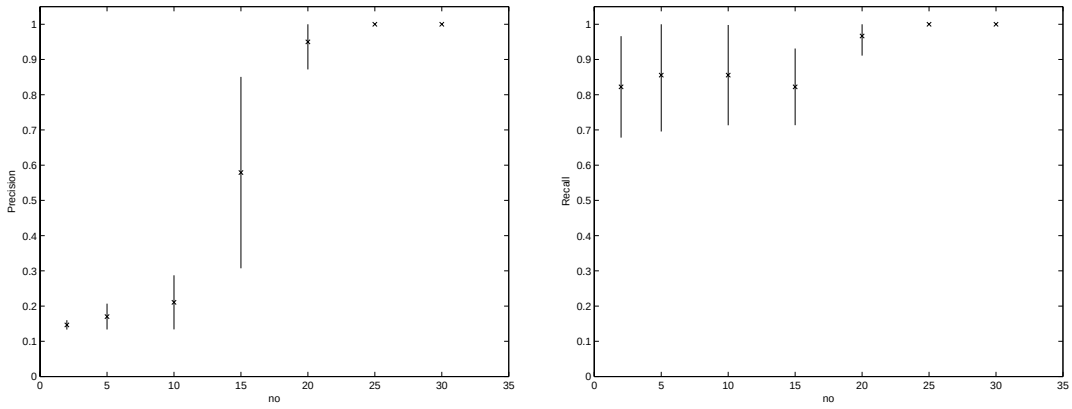

Figure 4.5: Precision & Recall for Strength of Signal for COSA2

Another experiment was conducted where, for a small number of signal attributes (actually *no* = 3), the scale parameter, λ, was varied in a large range of small values in order to attempt to find the signal since small values of λ will focus on a small number of attributes (§2.2.2). However with λ values varying from 0.1 to 1e-5 no increase in the precision or recall values were found. The possible reason for this is that the noise was completely obscuring that of the signal, no matter how small λ was made.

4.2.2 Strength of Signal for COSA2 With Lower Targeting of Distances

The above simulations were then repeated but this time searching for sub-sets of attributes having similar values and close to the 5% percentile of each attribute of the data. From the earlier discussion (§4.1.1) of the resulting dendrograms found with this method it was not possible to automatically find clusters and this had to be done by eye for each data set (§4.1.1).

For this test no clusters could be found for any of the signal strength values which was to be expected as the signal was contained at the upper end of each signal attribute distribution.

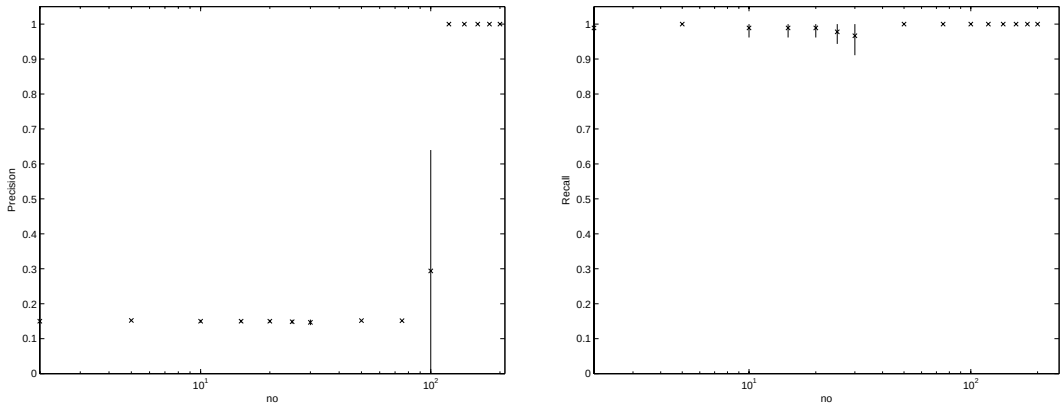

Figure 4.6: Precision & Recall for Strength of Signal Using Euclidian Distance

4.2.3 Strength of Signal for COSA2 with Upper and Dual Targeting of Distances

Again the above simulations were repeated but this time altering the distance between attributes to be targeted at an upper limit (95% percentile of each attribute) and at the upper or lower limit (dual targeting) or each attribute of the data. This time highly defined clusters (precision and recall always 1.0) were obtained with signal strengths starting from as low as *no* =5. Since the signal for each signal attribute was located at the end of its distribution this is to be expected. No differences were again seen when the simulations were repeated with the Matlab version of COSA2.

The high power of COSA2 in retrieving even very small signals is revealed here. By restricting the search to the targeted distances the algorithm is then more likely to find targeted clusters of interest rather than other more dominant clusters that prove to be of perhaps less interest.

4.2.4 Strength of Signal for Hierarchical Clustering With Euclidian Distance

Again the above simulations were repeated but this time the distances between objects were calculated by using normal Euclidian distance rather than using COSA2. The precision and recall plots are shown in Figure 4.6.

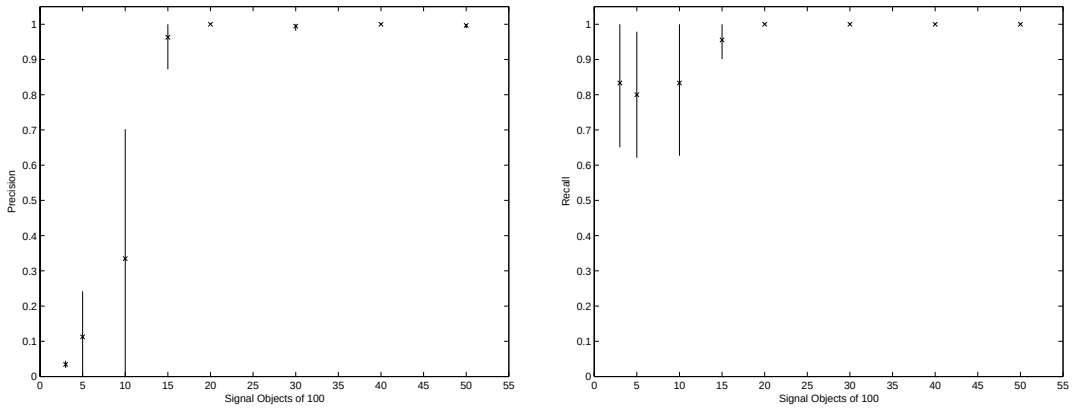

Figure 4.7: Precision & Recall for Signal to Noise Ratio

Here the advantage of the COSA2 algorithm is evident with normal Euclidian distance being only able to split the signal and noise objects with a signal strength of around 110 attributes.

4.2.5 Signal to Noise Ratio

In this simulation the number of attributes contributing to the signal was kept fixed at no = 20 and the number of signal objects, S, was varied from almost zero to the number of objects, N. All other parameters for COSA2 were as used for the above tests. Again there was no significant differences observed when run with the Matlab version of COSA2.

Plots of precision and recall are shown in Figure 4.7. Values are only shown for a signal to noise ratio of 50%, a mirror of the graph occurs for the ratio 50% to 100%.

The figure reveals that for a signal strength of 20 attributes of 1000 the signal could be split from the noise if the signal to noise ratio was over about 15%. The poor performance with a very low signal (and also conversely for a very high signal, for example close to 100%) can be explained by the fact that the importance of each attribute is reduced as the dispersion for the objects of interest will be close to the dispersion over all objects (see equation 2.28). These limitations can then be used as a guide when applying to real data sets.

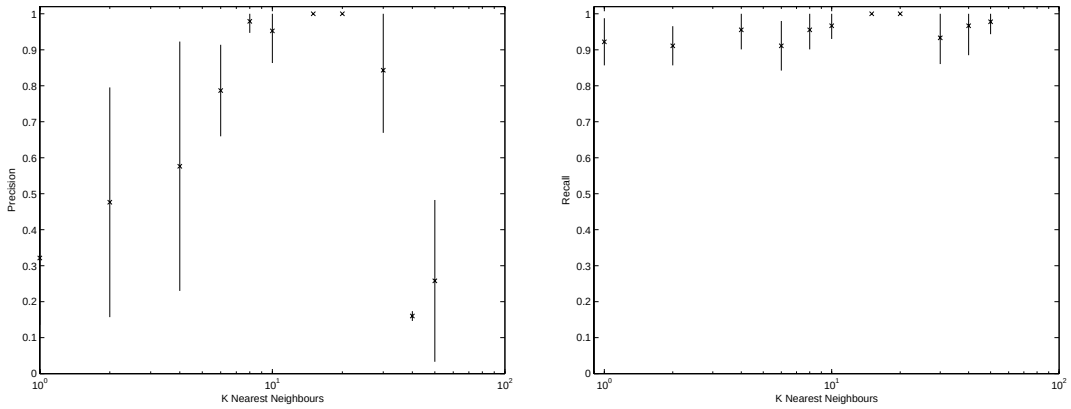

Figure 4.8: Precision & Recall for K Nearest Neighbours Used to Define Clusters

4.2.6 Number of K-Nearest Neighbours Used to Define Clusters

Here the value of the *K* nearest neighbours used to define the clustering in the COSA2 algorithm was investigated. The authors of COSA2 claim that setting *K* to be $\sqrt{N}$ is sufficient and this claim was tested here. For this test the parameters for COSA2 were as the last test but now with the fixed number of signal objects, *S* set back to 15. Plots of the precision and recall of the clustered signal objects are shown in Figure 4.8.

The plots show that the *K* nearest neighbours can be set in the range of around 7 to 20 objects which is covered by the default value of $\sqrt{N}$ though the values of 1.0 for precision and recall may indicate that using a slightly higher value of *K*, say 15, may produce better results. Again no significant differences were revealed with the use of the Matlab version of COSA2.

4.2.7 A Comparision of the Different Dissimilarity Methods Used

As previously described (§2.2.3) two methods can be used to define the dissimilarity between objects. This was investigated by repeating the method used in §4.2.1 but this time using the alternative dissimilarity definition of maximum whole distance (equation 2.16).

The resulting values of the precision and recall were then plotted against the values obtained from the maximum weight distance (equation 2.15) to investigate if any statistical difference occurred. These plots are shown in Figure 4.9.

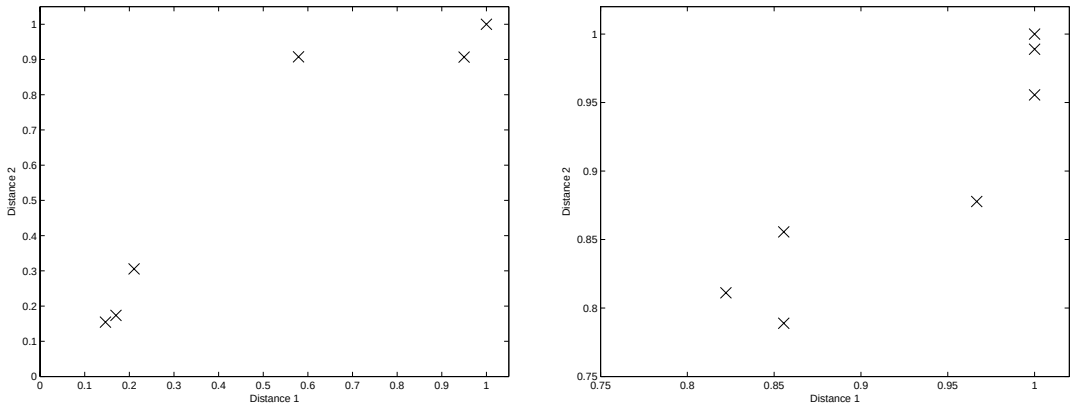

Figure 4.9: Cross Plots of Precision & Recall for Alternative Dissimilarity Definitions

The precision cross plot shows no significant difference between either weighting scheme for the two dissimilarity measures (paired *t* test with $P < 0.05$), though the recall cross plot shows a significant difference ($P = 0.023$) in favour of the first measure. This corroborates with that found by the authors who reported a "sometimes superior performance" (§5 of [15]). A similar difference was noted when using the Matlab version of COSA2.

4.2.8 The Effect of the Scale Parameter λ

By setting the signal attribute number, *no* to 20 and the number of signal objects, *S* to 15 the value of λ was modified from 0.01 to 1.0. The resulting plots of precision and recall for the signal objects are shown in Figure 4.10.

The effect of λ can be seen from the precision curve. For very small $\lambda < 0.07$ and for very large $\lambda > 0.4$ the precision drops dramatically. The effect of large λ can be understood from the fact that the dissimilarity measure now approaches that of the ordinary Euclidian distance (equations 2.2 - 2.4) with the consequences described in §4.2.4. For small values of λ more focus is placed on a smaller number of attributes that may be able to pick out the signal as they have the smallest dispersion (equation 2.8). However it transpires now that fewer objects in a cluster will have influence on the estimated weights through equations 2.8 & 2.13 with the result that the variance of these weights are increased with a consequent reduction in the power of the algorithm.

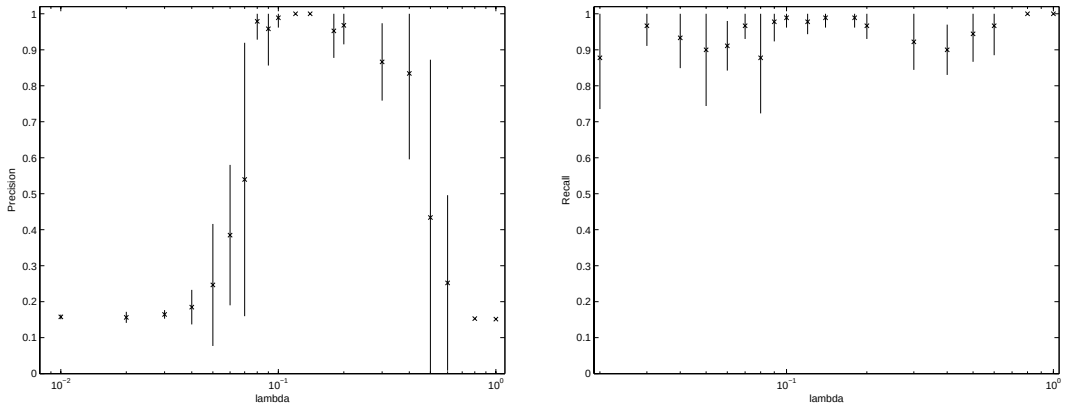

Figure 4.10: Precision & Recall of Signal as the Scale Parameter λ is Varied

It is thus of *crucial* importance when using COSA2 that a value of λ is chosen that is not too high or too low. This can be done by simple experimentation, i.e. varying λ until stable clusters are attained. However by having some prior idea of the scale of those small distances, d_{ijk}, on those attributes, k, upon which groups of objects will preferentially cluster a value could be set for λ. This scenario though is hard to determine in practice with 'real-world' data sets that are noisy and contain many missing values.

It was also interesting to see the variation in the convergence of the COSA2 algorithm. For small $\lambda < 0.07$ the number of iterations was large (> 20) and usually had to be terminated before the difference in weights reached the prescribed tolerance level, perhaps indicating that a solution was difficult to find. For large values of $\lambda > 0.4$ the number of iterations was few (sometimes only 2 or 3) reflecting the quick transition to Euclidian distance. For values of λ that gave good precision and recalls of the signal the number of iterations was typically between 5 and 10. Again the Matlab version of COSA2 revealed a similar scenario.

4.2.9 Variation of the Homotopy Parameter α

As previously described the homotopy parameter α is used to avoid becoming trapped in local optima as the minimisation of the objective function in COSA is performed. It effectively controls the rate at which the inverse exponential distance approaches the

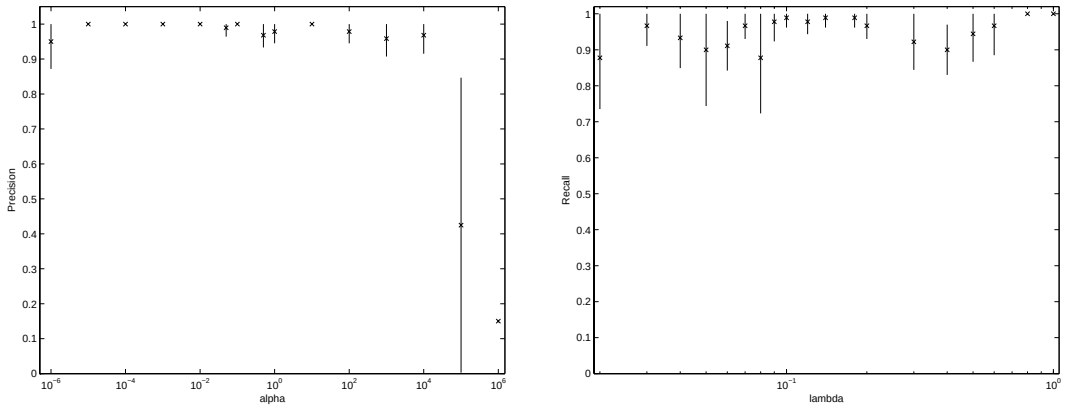

Figure 4.11: Precision & Recall Values for Signal Recovery as the Homotopy Parameter α is Varied

ordinary distance (equation 2.19). Too small an evolution will cause the solution to be trapped in local minima, too large and the ordinary distance is approached too quickly. Plotted in Figure 4.11 are the precision and recall curves for the signal recovery as α is modified from zero to around 10^5.

The values found proved misleading as though the behaviour at low $\alpha < 1.0$ is expected, for values greater than this, where the distance would evolve too quickly to the ordinary distances, good precision and recall occur and this was surprising. When the Matlab version of COSA2 was run with the same data however precision values were obtained that were around 1.0 until α became 1.0 and the precision then dropped dramatically (the recall values were always 1.0 for each value of α considered). These values are shown in Figure 4.12.

Here the precision behaves as it should as the value of α was increased over 1.0. Inspection of the weight changes (equation 2.22 and step 7 in §2.2.5) as the iteration procedure evolved with $\alpha > 1.0$ confirmed this. An initial small weight change when $\eta = \lambda$ in the first step of the COSA2 algorithm was then changed to a large weight change when α was applied to η (step 6 in §2.2.5) with poor results obtained as the distances approached the ordinary distances too soon.

The other outcome of this experiment was that α did not seem to improve the outcome of the signal recovery when it was used. In fact the homotopy method did not seem to affect the results when $\alpha < 1.0$ and a value of zero could be used.

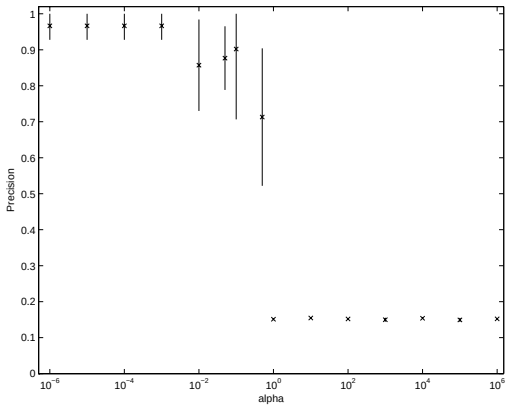

Figure 4.12: Precision Values as Homotopy Parameter Value α is Varied Using the Matlab Version of COSA2

4.3 Bridging Common Data Sets

In order to bridge two datasets that have common objects and attributes as discussed in §1.3.1 the distances d_{ijk} are taken from each set and a new single distance matrix is formed by taking the *maximum* distance for each d_{ijk}. In effect this has the effect of overlapping the signal regions to produce only one signal region from the two datasets. When clustered the bridged data would be expected to cluster on the common signal objects with the important attributes arising from the overlapped signal attributes. This scenario is illustrated in Figure 4.13.

A synthetic bridged dataset was formed from two synthetic data sets as described above. Here one set of data was formed with a signal where S = 20 and no = 25, and the other with S = 15 and no = 30 but this time starting at attribute 5. Running COSA2 produced a cluster of 15 objects on the overlapped signal of 20 attributes with the important attributes being those between 5 and 25 inclusive, justifying the above argument. These attributes are shown in Figure 4.14.

4.4 Conclusions

The COSA2 algorithm, both that provided by the authors and the Matlab version produced for this work has been tested thoroughly for all variable parameters. In particular

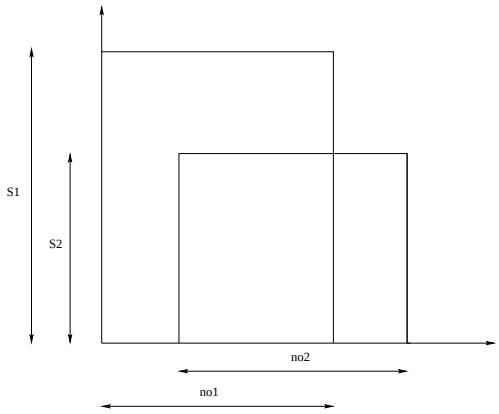

Figure 4.13: Concepts Involved in Bridging Data Sets

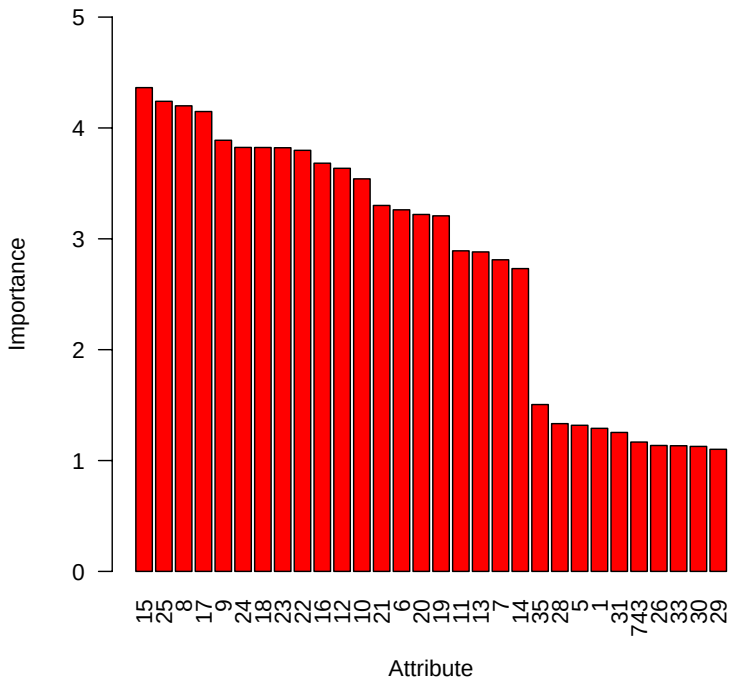

Figure 4.14: Important Attributes for the Bridged Data Set

limitations have been placed on the signal to noise ratio as well as signal strength in determining its recovery and how this can be increased if the attributes of the signal are close to a high or low value of interest by using targeted clustering. The parameter that has to be used with care is the scale parameter λ, whose effect must be investigated in any simulation study. The homotopy parameter α did not have an effect on any outcome and its value can be set to zero.

The bridging scenario which has been implemented in the Matlab version of COSA2 has also been discussed and succeeded when used with two synthetic datasets.

Chapter 5

Application of COSA - Diffuse Large B-Cell Lymphoma

5.1 Introduction

Microarray experiments have been used to predict patient survival times, notably for Breast Cancer (Van de Vijver *et al.* [17]) and T-cell acute lymphocytic leukaemia (T-ALL) (Chiaretti *et al.* [16]). Diffuse Large B-cell Lymphoma (DLBCL) is a disease that has been researched by a number of different groups to predict patient survival times and more importantly, for this work, has been studied on both cDNA arrays by Alizadeh *et al.* [18], with extended work by Rosenwald *et al.* [21], and on oligonucleotide arrays by Shipp *et al.* [20]. An attempt has also been made to bridge the data from the two platforms by Wright *et al.* [22] of which a summary was made by Ferl *et al.* [23].

This data has been made available to the public and makes for a good case for analysis by the COSA2 algorithm. The initial aims were then, for each set of microarrays, to independently find any interesting clustering patterns of the samples involved (such as patient survival, patient IPI index or type of DLBCL) and to find the most important genes that cause these patterns. These sample clustering patterns and genes could then be compared to those found in the above studies.

Each of these studies is first summarised below followed with a description of the

analysis by the COSA2 algorithm and a discussion of the findings.

5.2 Data from cDNA Arrays

Alizadeh *et al.* [18] studied 39 DLBCL samples, with approximately 8500 gene clones on Lymphochip cDNA arrays. Hierarchical clustering of the data revealed two types of DLBCL. One type expressed genes characteristic of germinal B cells (known as Germinal Centre B-cell like, or GCB DLBCL) and the other expressed genes characteristic of mitogenically activated blood B cells (termed Activated B-cell like, or ABC DLBCL). Other gene expression signatures were also found; those related to proliferation cells, T-cells and lymph-node biology were also differentially expressed among the clustered patient samples. However the signature belonging to the germinal centre B-cells was found to vary independently of the other three and this signature was the one deemed most important in clustering the samples. Using Kaplan-Meier analysis [25] of the overall survival of DLBCL patients, grouped using the basis of the germinal B-cell expression profile, revealed greater survivability of those patients with GCB DLBCL over those with ABC DLBCL. These survival figures were significantly greater than those obtained by grouping the patients according to their International Prognostic Index (IPI - an index based on patient age, stage of disease and other factors). The conclusion reached by the authors was that molecular classification of tumours based on gene expression measurements could identify clinically significant subtypes of cancer that were previously undetected. A 360 sub-set of genes that differentiated the GCB and ABC samples together with the three other genetic signatures including their ontologies and all patient data can be found at http://llmpp.nih.gov/lymphoma.

The work of Rosenwald *et al.* [21] extended the above work by analysing a larger data set of 274 patients of which 240 were publicly available, with a similar set of gene clones on the Lymphochip cDNA arrays. The patient data indicated type of DLBCL as found by the analysis detailed below, together with survival statistics and IPI index. By using a 100 gene subset of the GCB/ABC predictor of Alizadeh *et al.* [18] they were able, by hierarchical clustering of these genes, to obtain three distinct clusters of patient samples. These were the GCB and ABC DLBCL groups as described above

and a third group, termed type III DLBCL, which did not express either set of genes at a high level. Again Kaplan-Meier survival estimates indicated that those patients with GCB DLBCL outlived those with the ABC type, and independently of their IPI index. The type III DLBCL had similar survival estimates to the ABC DLBCL cases. Although these results seemed to agree with the previous study the authors noted that 35% of patients with ABC DLBCL still survived after five years.

A further simulation was carried out to in order to identify those genes whose expression correlated with the patient outcome. Hierarchical clustering then grouped these genes into 4 separate gene-expression signatures, akin to the previous work of Alizadeh *et al*. The groups were identified as those expressed in germinal B-cells, proliferating cells, lymph node cells and the major histocompatability-complex (MHC) class II signature. A small 16 gene predictor was then constructed from each of these groups of the genes that were significantly associated with survival. This group was then found to predict better survival of the patients than that used with the GCB/ABC gene predictor or those found using only the patients IPI index. The predictor's great power was revealed in the fact it could be used to subdivide patients within each subgroup into distinct risk groups that were not recognised by the GCB/ABC subgroup predictor or IPI data. In particular the predictor identified the few patients in the high-risk group with a high IPI level who were long term survivors as well as those patients in the low risk IPI group with a GCB type DLBCL who transpired to have a low survival time.

The 100 gene GCB/ABC predictor, together with the four gene signatures including their ontologies and all the patient and raw Lymphochip data can be found at http://llmpp.nih.gov/DLBCL.

It is also worth mentioning at this point the work done by Hastie *et al.* [19] on the Alizadeh *et al.* work with their statistical tool called 'Gene Shaving'. This tool uses principal components analysis to find the top 'super-genes' that have the maximum variance of expression data. Important genes can then be found by finding those genes that have the largest inner product with the 'super-gene' (or in reality genes are dispensed with that have low inner products with the 'super-genes'). This leads to clusters of genes that have coherent expression patterns and large variation across experimental

conditions. The study differs from hierarchical clustering in that genes may belong to more than one cluster and that the clustering may be supervised by an outcome measure such as survival times or type of disease. The technique can be unsupervised and partially, or fully supervised by using known properties of the samples or the genes to find any interesting groupings. Groups of genes resulting from unsupervised learning as well as supervised learning were used to find clusters of samples that were related to patient survival. Two groups of samples were discovered by this process and these were found to have significantly different survival times. Lists of these genes are made available in the paper.

5.3 Data from Oligonucleotide Arrays

The study of Shipp *et al.* [20] involved measuring the gene expression profiles of 58 patient samples on Affymetrix oligonucleotide microarrays containing 7129 probes representing 8,817 genes. Patient data included survival statistics after treatment and IPI indicies. The initial analysis involved finding those genes that were most highly correlated with patient samples that were labelled cured versus those labelled fatal or refractory. A list of the top fifty differentially expressed genes representing each category was found and this then formed the basis of a cross validation prediction model which used supervised learning methods such as SVMs and nearest neighbour methods. Good predictions of the patient survival were made by the model and a small thirteen gene set resulted from this. Kaplan-Meier analysis of the survival probabilities of the two groups showed a distinct difference between those patients marked as cured and those marked with fatal/refractory disease and improved upon similar analysis using the IPI data indicating that this predictor provided additional information that was not present in only the clinical prognosis information.

The group then sought to validate this model against the cell-of-origin model of Alizadeh *et al.* described earlier. In a sense this was their attempt to bridge data between the two platforms though such comparisons are difficult due to the different genes measured by different methods on both platforms and the fact that the patient samples were different. However the group proceeded to find the 90 genes that were represented on

the Affymetrix array that were included in Alizadeh *et al.*'s predictor. Hierarchical clustering of the samples using only this smaller gene set still split the samples of Alizadeh *et al.* into the GCB and ABC DLBCL types indicating that 90 overlapping genes were sufficient to make this distinction. The 90 gene predictor was then applied to the patient samples of their own study. Again a good split of two sets of patients was observed and these sets were highly correlated with the cell of origin distinction, though it must be emphasised that this was not a 100% distinction. However the split of samples did not reflect the correlation between DLBCL type and patient survival, indicating that this set of genes may have reflected cell-of-orgin but could not explain the clinical variability in this particular data when measured on this type of platform. This may be caused by the genes on the cDNA platform showing a false high or low expression level, perhaps from some systematic error caused by the type of platform and experimental measurement akin to the platform type; indeed the true measurement may be the poor result on the alternative platform. The discrepancy in results may also appear to corroborate that found by the Rosenwald *et al.* work where risk groups within the type of DLBCL had to be found from other gene signatures to obtain good predictions of patient survival. The observed difference could also be caused by the genes of the cDNA that were not present on the Affymetrix oligonucleotide array with a consequence that the latter's predictor was missing some of the most discriminating genes and was thus enriched for genes that only differentially expressed between the GCB/ABC subgroups with a small statistical significance.

The group also sought to find support for their 13 gene outcome predictor in the expression data of Alizadeh *et al.*. Three genes from the 13 were found to be represented on the cDNA Lymphochip and all of them were found to be highly correlated with outcome though nothing of course could be said of the remaining ten genes. If all genes were present it would have been interesting to determine if any of the predictor genes did not produce a high correlation with outcome perhaps indicating potential systematic errors in the measurement of the gene expression levels on one or both of the platforms.

This study reveals the reasoning to bridge arrays concurrently. By using only those genes common to both platforms to build a single predictor can the expression levels

of a gene be reviewed and any discrepancies noted. The overriding problem though for this to succeed is that both the same sample set must have been used on each platform and this is not the case for this data.

Data for this study and list of predictor genes with their ontologies and all patient data can be found at http://genome.wi.mit.edu/MPR/lymphoma.

5.4 Bridging the Platforms

The work of Wright *et al.* [22], which had been summarised by Ferl *et al.*, attempts to build a single patient survival model from the Rosenwald *et al.* and Shipp *et al.* data. Their reasoning was to find those genes that discriminate the DLBCL subgroups with most significance; 27 such genes on the cDNA platform were found and formed the basis of a predictor. A linear predictor score was constructed for each sample based on the summation of the expression value of each gene in the predictor group multiplied by a scaling factor whose value was chosen to depend on the strength that a gene could differentiate between the sample subgroups. The choice of scaling value was that of the *t* statistic generated by a *t* test that determined the difference in gene expressions between the DLBCL subgroups. A simple application of Baye's rule could then determine which DLBCL group that a sample outside those used to build the predictor belonged to. A 90% cut-off level was also applied so that samples having scores of less than this value were placed in a third group with no specification. The predictor was tested by splitting the cDNA platform samples into two groups of training and validation samples, and the resulting validation samples showed good rates of survival for the GCB cases and poor rates for the ABC cases when analysed by Kaplan-Meier analysis. A further success was obtained in reclassifying some of the type III subgroup found by Rosenwald *et al.*

The predictor was then applied to the oligonucleotide data. To do this 14 genes were found to be represented on the Affymetrix chip that were also present in the 27 gene predictor set on the cDNA Lymphochip. This set of genes were used to create a new DLBCL group predictor by using the same scaling coefficients that were used in the predictor on the cDNA chip but modifying the expression values of the genes

to match the mean and variance of the corresponding genes on the cDNA chip. This modification was necessary to account for any systematic measurement differences between the two different platforms.

Application of this linear predictor score proved promising as the samples which split into the two DLBCL groups, when analysed by Kaplan-Meier survival analysis, showed a distinct difference in survival times, an improvement on the hierarchical clustering method performed by Shipp *et al.* discussed earlier. The success of the predictor can be traced to the fact that it employs only a small group of genes that best discriminate the DLBCL types in the larger 274 sample study rather than the larger gene predictor from the smaller group of samples from Alizadeh *et al.*, containing many genes that predict the differences with lower statistical significance. Adjustment of the Affymetrix predictor using the cDNA Lymphochip gene data to help model the systematic measurement differences between the the two array platforms also helped in this success.

There are however some potential flaws in this approach. In particular, as pointed out in the summary from Ferl *et al.* the process involved in building the predictor is only unidirectional. It would be interesting to see if a predictor based on the oligonucleotide data and applied to the cDNA would produce similar results. The predictor used is also only a linear naive Bayes classifier and it would be interesting to apply more advanced methods from machine learning to the problem. Another problem is in the assumption that the ‘best’ genes on the cDNA platform revealed the ‘true result’ and that their expression levels were not in fact an artifact of some systematic error associated in the build and subsequent experimental technique on that platform. This scenario could be revealed by using the bridging process described in this work, i.e. taking each expression level measurement from each common gene and choosing the one that gives the worst differential level over all samples.

The gene expression data used in this study, together with the 27 and 14 genes used in the predictors, including their associated ontologies, can be found at http://llmpp.nih.gov/DLBCLpredictor.

5.5 COSA analysis

The COSA2 algorithm was applied to each data set independently in order to find any stable clusters of samples when varying the scale parameter λ for both targeted (differentially expressed genes) and non-targeted distances. The important genes contributing to these clusterings could then be found by the importance measure as detailed in §2.2.8. These genes could then be inspected with the predictors and gene signatures taken from the studies described earlier to reveal any interesting sample groupings and genes involved in producing these clusters. Patient survival times could then be calculated for the sample groups to determine if any significant difference could be observed.

5.5.1 cDNA Array Data

Here the data was taken from the Wright *et al.* study. Due to local computational limits the number of samples had to be reduced from 274 to 60, ensuring a similar ratio of the three types of DLBCL were present in the smaller sample set as was in the full set. This data had many missing values. Missing data can be handled in a number of ways (Smith [10]); statistical models can be used to predict values for the missing data from the given data, or more simply the values can just be set to the mean of the attribute they belong to. A recent study by Li & Gui [24] which uses the Rosenwald *et al.* data identified, for each gene, eight genes that were the nearest neighbours according to Euclidian distance. The missing data was then filled with the average of the nearest neighbours. In this study a very crude approximation was made, the missing values were set to zero which may have ramifications for the results. The sample set, along with patient statistics (follow-up times, IPI index and status) and DLBCL type is shown in Appendix A.1.

5.5.1.1 Sample Clusters Found

The value of λ was varied between the values of 0.1 and 0.4, as this was the region found to produce stable clusters in the tests on λ as described in §4.2.8. Since there was prior knowledge that differentially expressed genes would probably be present

clusterings involving two distinct groups were sought. However this proved to be difficult from the dendrograms produced from the COSA dissimilarities and was confounded by the results of targeting clustering which seemed to produce smaller sized clusters that were sometimes a mixture of the clusters found by the non-targeting approach. Choosing clusters was thus a difficult and time consuming exercise. Few definite clusters were found emerging from the root of the dendrograms, although a utility provided by the authors of COSA [15] allowed a point and click mechanism for choosing clusters when viewed in the *R* software package which then highlights the clusters by surrounding them with a box.

A sample of a few of these dendrograms are illustrated in Figure 5.1 where the dendrograms represent $\lambda = 0.1$ for both non-targeted and targeted along the top of the diagram and those for $\lambda = 0.2$ for non-targeted and targeted along the bottom; all dendrograms were produced using average linkage.

A set of two clusters that was the best fit from the dendrograms in Figure 5.1 was chosen. Those resulting from the clusters formed where $\lambda = 0.3$ and 0.4 were discarded as they showed significantly more variability (or as significant as the human eye could deduce).

The two clusters, one having 28 patients, the other 32 are shown in Table 5.1, together with the patient status, IPI Index and type of DLBCL. The patient indicies can be found from the list of the sixty patients chosen from the Rosenwald *et al.* study which is included in Appendix A.1.

The results are disappointing, the clusters showed no significant difference in patient status (χ^2 test, P=0.74), IPI Index (P=0.25) and DLBCL type (P=0.057) although the latter was close to the significant test ($P < 0.05$) indicating a possible correlation between the different DLBCL type and the clusters.

Patient survival is shown in Figure 5.2 which was modelled by Kaplan-Meier analysis with the top curve showing the survival for cluster one. Applying G-ρ tests, with ρ set to zero (the log-rank test [26]), to the survival curves showed no significant difference between the two plots ($P = 0.265$).

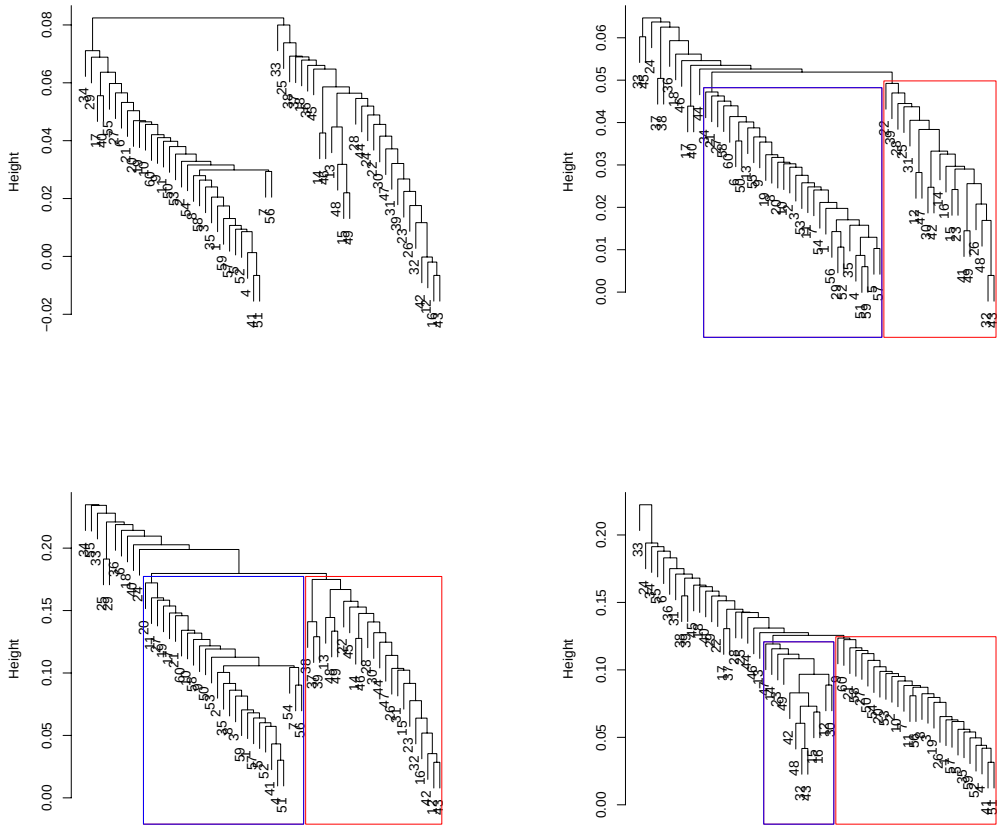

Figure 5.1: Dendrograms Found from the cDNA Data

5.5.1.2 Important Genes

The closeness to a possible correlation between DLBCL type led to an investigation of the most important genes contributing to the two clusters found using COSA2. By the importance measure method defined in §2.2.8, equations 2.27 & 2.28, the twenty most important genes were calculated for each cluster and both are shown in Figure 5.3 and listed in Appendix B.1. The table indicates whether any of the genes were present in the cell signatures and DLBCL predictor of Rosenwald *et al.*, the predictors of Shipp *et al.* and the genes found by the 'Gene Shaving' method. In the figures of importance measure the genes are labelled with their unigene cluster if known, else their Genbank

Patient ID	Status	IPI Index	DLBCL Type
1-11,17,19-21	15 Alive	6 High	19 GCB
27,29,34,35	17 Dead	15 Medium	8 ABC
40,41,50-60		10 High	5 Type III
		1 N.A.	
12-16,18,22-26	11 Alive	3 High	8 GCB
28,30-33,36-39	17 Dead	9 Medium	12 ABC
42-49		14 High	8 Type III
		2 N.A.	

Table 5.1: Sample Clusters Found in the cDNA Study

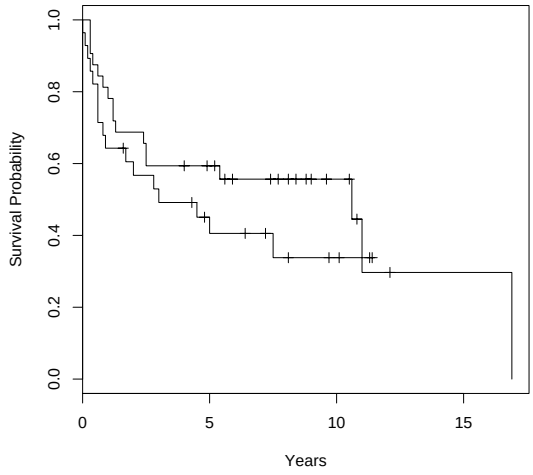

Figure 5.2: Kaplan-Meier Model Using cDNA Clusters

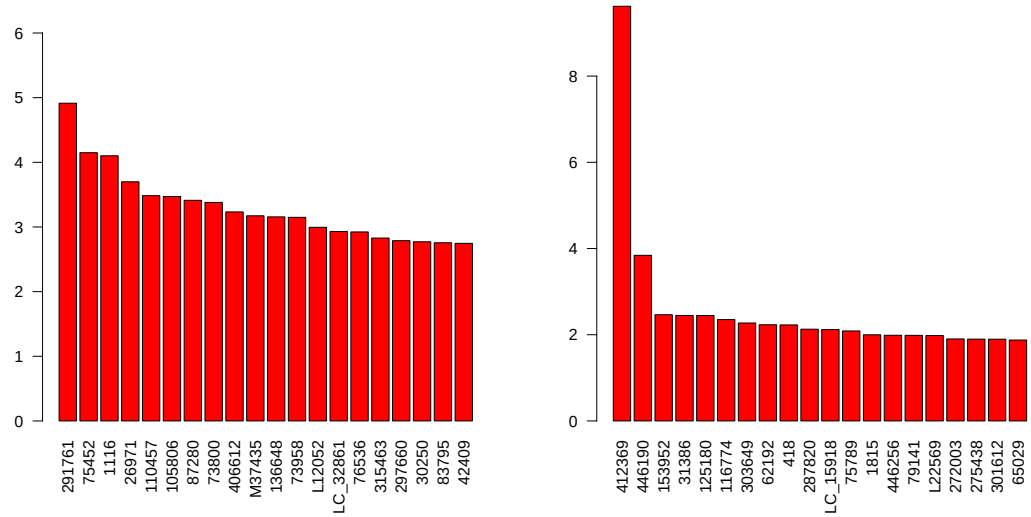

Figure 5.3: Important Genes Found from cDNA Study

accession number if known or else their description.

It was noted that one gene clone in the first sample cluster and two gene clones in the second sample cluster had maximum importance. This was traced back to the fact that these attributes had many missing values and as a consequence the dispersions over the clusters were zero. These genes were removed from the plots and the tables, though their presence indicated that a better treatment of the missing data was needed.

Very few of the listed genes cross referenced with those found in the literature mentioned earlier. In the first cluster only one gene was present in the 100 gene DLBCL type predictor, two in the lymph-node cell signature and two from the proliferation cell signature as found by Rosenwald *et al.* In the second cluster four genes were found in same authors' lymph-node cell signature. No genes were found that corresponded to any of the authors' other signature lists, nor any described by Shipp *et al.*, nor any found in the 'gene shaving' method. It was also noted that no gene was common to either of the two clusters and that the gene represented by unigene cluster id 412369 and found in the set producing the second cluster had a substantially higher importance number (close to 9) that was at least twice as large as any other importance value.

5.5.2 Oligonucleotide Array Data

The data set used in this simulation was that used in the Wright *et al.* study. This had been normalised by the authors as follows. Genes were only used that were listed on more than half of the samples and their signal was multiplied by a factor to make the median value of these genes equal to 1,000. All signal values that were less than 50 were set to 50 and finally a $\log_2$ transformation was then applied.

5.5.2.1 Sample Clusters Found

In this case the clusters found were more stable over the same range of λ as used in the cDNA data study though complete linkage rather than average had to be used to reveal them. Targeted clustering again did not provide any more further information regarding the structure of clusters. Examples of two dendrograms found from the COSA dissimilarities are shown in Figure 5.4, one for $\lambda = 0.1$ the other for $\lambda = 0.4$.

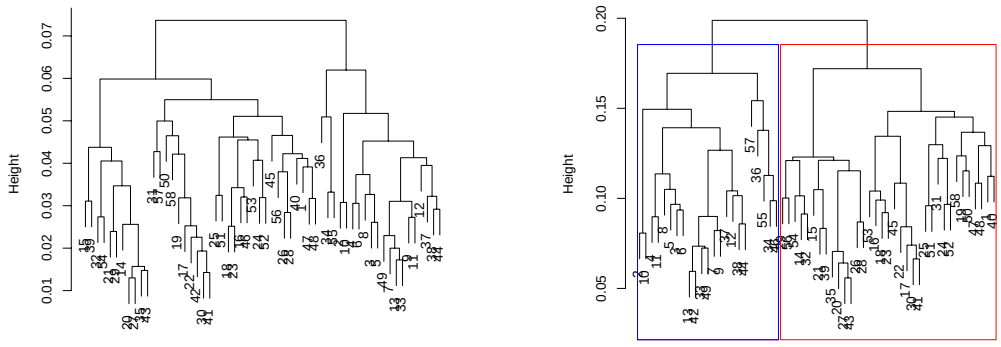

Figure 5.4: Dendrograms Found from the Oligonucleotide Data

The two clusters, one having 23 patients, the other 35 are shown in Table 5.2, together with the patient status, IPI Index and patient outcome. The patient indicies can be found from the list of the fifty-eight patients chosen from the Shipp *et al.* study which is included in Appendix A.2.

The clusters showed no significant difference with regard to patient status (χ^2 test

Patient ID	Status	IPI Index	Outcome
2-13,33,34	13 Alive	0 High	12 w/o disease
36-38,42,44,46	10 Dead	7 High Int.	11 w disease
49,55,57		5 Low Int.	
		10 Low	
		1 N.A.	
1,14-32,35	18 Alive	2 High	20 w/o disease
39-41,43,45,47	17 Dead	10 High Int.	15 w disease
48,50-54,56,58		6 Low Int.	
		16 Low	
		1 N.A.	

Table 5.2: Sample Clusters Found in the Oligonucleotide Study

$P = 0.91$), patient outcome ($P = 0.92$) or IPI Index ($P = 0.85$ with the high intermediary values being placed in the high category and likewise for the low values for this study).

Patient survival is shown in Figure 5.5 which was modelled with Kaplan-Meier analysis with the top curve showing the survival for patients in cluster one. Applying the G-ρ tests as described above to the survival curves showed no significant difference between the two plots ($P = 0.431$).

5.5.2.2 Important Genes

Using the importance measure for the genes in each cluster revealed the reason for the similarity between the two clusters. The number of genes having maximum importance in each cluster numbered around 750 and this was due to the preprocessing of the expression levels as described above. By setting the expression levels to a constant value if their value was below a certain threshold caused their dispersion to be zero and due to the fact that many of the genes had this low value (around 1,700) across most samples led to zero dispersions for many of them for each sample cluster. The clusters found for this particular simulation by COSA are thus driven not by a group of genes having a signal but rather by the genes with random low expression values

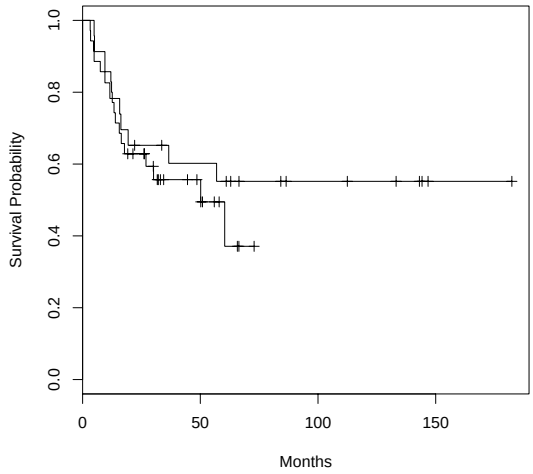

Figure 5.5: Kaplan-Meier Model Using Oligonucleotide Clusters

representing the noise. Ways to counteract this failing are discussed in the 'Further Work' section in the concluding chapter of this work.

5.5.3 Bridging the Data Sets

Unfortunately, due to the fact that the samples sets involved in all the studies discussed previously are different the proposed bridging technique discussed in this work could not be used. However, by looking at each set independently finding interesting gene expression patterns resulting from sample clusterings was attempted but the problems associated with the oligonucleotide data prevented this.

Chapter 6

Conclusion

The recent COSA algorithm proposed by Friedman & Meulman [15] that clusters objects on subsets of their attributes has been discussed (§2) and two versions have been built; COSA1 using an internal Kmeans clustering algorithm, and COSA2 which outputs dissimilarities for use in hierarchical clustering algorithms. Both program versions were written in Matlab and provide both a platform independent means of running clustering studies and a flexible method of inputting different distance calculations between object attributes. When tested on very simple data sets (§3) the COSA1 algorithm revealed problems in finding stable clusters due to convergence problems (§3.2.1). COSA2 on the other hand was very successful with these simulations and this algorithm was then used with tests on synthetic microarray data sets (§4). Here again COSA2 proved its success in being able to separate a small gene signal above a background of random noise represented by the vast majority of the genes. The power of the algorithm was illustrated in its ability to find such a signal well before a hierarchical clustering algorithm, using normal Euclidian distances, was able to. The targeted clustering method available with the COSA algorithms was also tested here and showed a substantial improvement again on retrieving even smaller signals than with no targeting. Since both COSA algorithms are heuristic the many parameters that control their output was also tested at this stage. Limits of signal to noise ratio and strength of signal that could be retrieved over a certain size of noise were found. Variations in the precision and recall of objects within well defined signal clusters gave limits on the internal parameters used in the COSA2 algorithm. One parameter, the

scale parameter, which effectively controls the number of attributes contributing to a cluster was identified as the one to be used with care.

With COSA2 fully tested it was then applied to real microarray data (§5) from data taken from samples of patients having diffuse large B-cell lymphoma (DLBCL). Such data has been measured on both cDNA and oligonucleotide microarrays and many studies have been performed to try to find gene signatures that classify the two types of DLBCL as either Germinal centre B-cell like (GCB) or Activated B-Cell like (ABC). The studies have also tried to find the link between DLBCL type and patient survival times with the hypothesis being that those with GCB type have a more higher survivability and also to search for other genetic signatures that could explain the difference. COSA2 was then applied to each of the two datasets independently in an attempt to discover any important sample clusterings and the most important genes that caused them. Sample groups were found, though the variation of the scale parameter mentioned earlier made for a difficult process in the the extraction of stable clusters. The samples clusters found in each case proved disappointing with no clusters showing a significant difference when compared with respect to patient outcome, status, prognosis index or survivability. The distinction between GCB and ABC type of DLBCL was also not found. The possible reasons for this are stated in the section on 'Future Work' below.

A possible method for bridging data from both types of microarrays concurrently in order to trap any systematic error in one or both of the values produced was also discussed (§1.3.1 & 4.3). The COSA2 algorithm coded for this work was written to accept such data from two platforms. This method was tested on some artificial data and proved successful (§4.3) though it could not be applied to the DLBCL data as the samples involved were different.

6.1 Future Work

One of the claims of the COSA algorithm is that data such as microarray data can be input immediately into it, without any need to weed out 'noisy' genes that would not contribute to the signal and which could be found by preprocessing the data with

statistical tools such as a *t* test. Instead COSA can take the whole data and find interesting sample clusters regardless of the noise of the vast majority of genes. This is an attractive option when dealing with microarray data and when testing on synthetic microarray data this approach was found to be successful. However with the real data from the DLBCL studies the algorithm seemed to fail to produce any meaningful results as sample clusters seemed to be randomly chosen. Inspection of the importance measurement of the genes found in each of the clusters revealed many genes from the oligonucleotide data that had the maximum importance, i.e. they had no dispersion over the objects in the cluster. Of course this small dispersion is the force driving the clustering in COSA but the preprocessing of the data by Wright *et al.* set many of the genes with small signals to a constant value. Here COSA did its job but simply found those genes that had been modified this way. The preliminary simulations on this data then need to be repeated with the raw data from Shipp *et al*. For the cDNA data again small numbers of genes were found to have a maximum importance and these were traced to genes whose values were taken to be zero as they were missing. This very naive way of modelling missing data then introduced certain genes that would have very small dispersions if they contained a lot of missing values. Again this clustering seemed to be driven by genes representing the noise of the data.

The conclusion here is then that real data cannot be simply thrown blindly at these algorithms and that the important data mining task of initially inspecting data to look for trends and missing data cannot be ignored. Once missing data is found it must be decided to be either missing at random (MAR) or emerging from a systematic source and modelled appropriately via the methods detailed by Smith [10] and references within.

Once this task has been completed on both data sets the procedures applied in §5.2 & 5.3 should be repeated to discover and interpret any interesting clusters. A further simulation should also be performed using hierarchical clustering with normal Euclidian distances rather than the COSA dissimilarities to determine whether COSA is able to produce significantly more informative groupings of samples and genes than this more conventional method.

Appendix A

Sample Lists used in DLBCL Studies

A.1 cDNA Microarray Data

The following set of 60 patient samples is a subset of the one used by Rosenwald *et al.* [21]. They are listed with their patient ID as used in this study, follow-up time in years, status (Dead or Alive), DLBCL type (GCB, ABC or Type III) and IPI Index (Low, Med or High).

A.2 Oligonucleotide Microarray Data

The following set of 58 patient samples is the one used by Shipp *et al.* [20]. They are listed with their patient ID as used in this study, IPI Index (Low, Low intermediate, High intermediate and High), follow-up time in months, status (Dead or Alive with or without disease) and outcome (expressed as 0 if without disease, 1 with).

Patient ID	Follow up	Status	DLBCL Type	IPI Index	Patient ID	Follow up	Status	DLBCL Type	IPI Index
1	4	Alive	GCB	Low	2	4.9	Alive	GCB	Med
3	5.6	Alive	GCB	Low	4	12.1	Alive	GCB	Med
5	10.5	Alive	GCB	Low	6	9.6	Alive	GCB	High
7	9	Alive	ABC	High	8	7.4	Alive	GCB	Low
9	8.8	Alive	GCB	Med	10	8.1	Alive	GCB	Low
11	5.2	Alive	Type III	Low	12	8.1	Alive	GCB	Med
13	11.3	Alive	GCB	Low	14	4.3	Alive	GCB	Low
15	11.4	Alive	ABC	Med	16	7.2	Alive	Type III	Low
17	8.4	Alive	GCB	Med	18	5	Alive	GCB	Low
19	10.8	Alive	ABC	Med	20	7.7	Alive	ABC	NA
21	5.9	Alive	GCB	Low	22	9.7	Alive	ABC	Med
23	6.4	Alive	ABC	Low	24	10.1	Alive	ABC	Med
25	1.6	Alive	Type III	Low	26	4.8	Alive	ABC	Low
27	11	Dead	Type III	Med	28	5	Dead	GCB	Med
29	0.3	Dead	ABC	High	30	0.8	Dead	Type III	Med
31	0.6	Dead	ABC	Low	32	0.6	Dead	GCB	Med
33	0.9	Dead	ABC	Med	34	0.3	Dead	GCB	Med
35	2.5	Dead	Type III	Med	36	0.2	Dead	ABC	Low
37	0.6	Dead	GCB	Low	38	0	Dead	Type III	NA
39	0.3	Dead	ABC	High	40	2.5	Dead	ABC	Low
41	1.3	Dead	GCB	Low	42	0.1	Dead	Type III	High
43	1.7	Dead	Type III	Low	44	3	Dead	GCB	NA
45	2	Dead	Type III	Low	46	0.4	Dead	ABC	Med
47	2.8	Dead	ABC	Low	48	7.5	Dead	ABC	Low
49	4.5	Dead	Type III	High	50	0.6	Dead	ABC	Med
51	5.4	Dead	GCB	Med	52	0.3	Dead	GCB	High
53	0.4	Dead	Type III	Med	54	0.8	Dead	Type III	Med
55	2.4	Dead	GCB	Med	56	1.2	Dead	GCB	Med
57	1.2	Dead	ABC	High	58	16.9	Dead	GCB	Low
59	1	Dead	ABC	Med	60	10.6	Dead	GCB	High

Table A.1: Samples Used in the cDNA Study

Patient ID	IPI Index	Follow up	Status	Outcome
1	Low	72.852	Alive w/o disease	0
2	Low	43.082	Alive w/o disease	0
3	Low intermediate	144.197	Alive w/o disease	0
4	High intermediate	61.016	Alive w/o disease	0
5	Low	86.459	Alive w/o disease	0
6	Low	84.197	Alive w/o disease	0
7	High intermediate	112.459	Alive w/o disease	0
8	Low	133.18	Alive w/o disease	0
9	Low	22.098	Alive w/o disease	0
10	Low intermediate	182.361	Alive w/o disease	0
11	Low	66.426	Alive w/o disease	0
12	N.A.	146.754	Alive w/o disease	0
13	Low intermediate	62.918	Alive w/o disease	0
14	Low intermediate	50.918	Alive w/o disease	0
15	Low	26.295	Alive w/o disease	0
16	N.A.	48.557	Alive w/o disease	0
17	High intermediate	55.934	Alive w/o disease	0
18	Low	12.59	Dead w/o disease	0
19	Low intermediate	50.164	Dead w/o disease	0
20	High intermediate	58	Alive w/o disease	0
21	Low intermediate	66.393	Alive w/o disease	0
22	Low	65.738	Alive w/o disease	0
23	Low	50.197	Alive w/o disease	0
24	Low	26.918	Dead w/o disease	0
25	Low	34.426	Alive w/o disease	0
26	Low	26.033	Alive w/o disease	0
27	Low	30.033	Alive w/o disease	0
28	Low intermediate	31.705	Alive w/o disease	0
29	Low	32.164	Alive w/o disease	0
30	Low	19.18	Alive w/o disease	0
31	Low	33.049	Alive w/o disease	0
32	Low	21.377	Alive w/o disease	0

Table A.2: Samples Used in the Oligonucleotide Study - Outcome 0

Patient ID	IPI Index	Follow up	Status	Outcome
33	Low	15.738	Dead w/disease	1
34	High intermediate	11.574	Dead w/disease	1
35	High intermediate	3.377	Dead w/disease	1
36	Low	36.59	Dead w/disease	1
37	High intermediate	5.049	Dead w/disease	1
38	Low	9.475	Dead w/disease	1
39	High	3.213	Dead w/disease	1
40	Low intermediate	4.885	Dead w/disease	1
41	High intermediate	12	Dead w/disease	1
42	High intermediate	4.885	Dead w/disease	1
43	High intermediate	60.361	Dead w/disease	1
44	Low intermediate	16.262	Dead w/disease	1
45	High intermediate	16.426	Dead w/disease	1
46	High intermediate	9.475	Dead w/disease	1
47	High intermediate	15.574	Dead w/disease	1
48	High intermediate	17.77	Dead w/disease	1
49	Low intermediate	56.918	Dead w/disease	1
50	Low	13.344	Dead w/disease	1
51	Low intermediate	12.295	Dead w/disease	1
52	Low	44.557	Alive w/disease	1
53	High intermediate	4.623	Dead w/disease	1
54	High	7.508	Dead w/disease	1
55	High intermediate	19.344	Dead w/disease	1
56	Low	30.131	Dead w/disease	1
57	Low	33.607	Alive w/disease	1
58	High intermediate	13.902	Dead w/disease	1

Table A.3: Samples Used in the Oligonucleotide Study - Outcome 1

Appendix B

Lists of the Important Genes Found in DLBCL studies

B.1 cDNA Microarray Data

The two tables detail the most important gene clones for each of the two clusters found from the COSA simulations of the data from Rosenwald *et al.* They are listed with their accession ID, unigene cluster ID, description (ontology) and a note indicating whether the clone occurred in any of the signatures or predictors as dicussed in Chapter 5. The key is as follows:

- GR - present in the germinal B-cell signature of Rosenwald *et al.*
- PR - present in the proliferating cell signature of Rosenwald *et al.*
- LR - present in the lymph-node cell signature of Rosenwald *et al.*
- MR - present in the MHC Class II cell signature of Rosenwald *et al.*
- PredR - present in the 100 gene predictor for GCB/ABC/Type III DLBCL of Rosenwald *et al.*
- PredS - present in the 19 gene predictor for patient outcome of Shipp *et al.*

- PredC - present in the 90 common gene predictor for GCB/ABC DLBCL of Rosenwald *et al.* & Shipp *et al.*
- GS - present in the genes found by “Gene Shaving ” by Hastie *et al.*

B.2 Oligonucleotide Microarray Data

No genes were listed for this study due to the problems encountered with the preprocessing of the gene expression data causing the most important genes to be simply random noise.

Accession ID	Unigene ID	Description	Note
	Hs.291761	Homo sapiens transcribed sequence	
NM_005345	Hs.75452	heat shock 70kDa protein 1A	
L04270	Hs.1116	lymphotoxin beta receptor	LR
AF220560	Hs.26971	B/K protein	
AF071593	Hs.110457	Wolf-Hirschhorn syndrome candidate 1	PR
NM_006433	Hs.105806	granulysin	LR
	Hs.87280	hypothetical protein MGC35578	
M25322	Hs.73800	selectin P	
AB014568	Hs.406612	unc-84 homolog B (C. elegans)	
M37435			
	Hs.136648	Homo sapiens transcribed sequences	
M29474	Hs.73958	recombination activating gene 1	PR
L12052			
		LC_32861	
Y12781	Hs.76536	transducin (beta)-like 1X-linked	
U16261	Hs.315463	interleukin 24	
U15637	Hs.297660	TNF receptor-associated factor 3	
AF055376	Hs.30250	v-maf musculoaponeurotic fibrosarcoma	
X15949	Hs.83795	interferon regulatory factor 2	PredR
AF151904	Hs.42409	CGI-146 protein	

Table B.1: Top 20 Important Genes Found from the cDNA Data Using Cluster 1

Accession ID	Unigene ID	Description	Note
	Hs.412369	Homo sapiens transcribed sequence	
	Hs.446190	Homo sapiens transcribed sequences	
X55740	Hs.153952	5'-nucleotidase, ecto (CD73)	
BC008666	Hs.31386	secreted frizzled-related protein 2	
X06562	Hs.125180	growth hormone receptor	
X68742	Hs.116774	integrin, alpha 1	
NM_002982	Hs.303649	chemokine (C-C motif) ligand 2	LR
M16553	Hs.62192	coagulation factor III (thromboplastin, tissue factor)	
U09278	Hs.418	fibroblast activation protein, alpha	LR
X02761	Hs.287820	fibronectin 1	LR
		LC_15918	
X92845	Hs.75789	N-myc downstream regulated gene 1	
M24122	Hs.1815	myosin, light polypeptide 3, alkali;	
	Hs.446256	Homo sapiens transcribed sequences	
U43142	Hs.79141	vascular endothelial growth factor C	
L22569			
M24173	Hs.272003	hemoglobin, zeta	
	Hs.275438	histone deacetylase 7A	
NM_005253	Hs.301612	FOS-like antigen 2	
L13698	Hs.65029	growth arrest-specific 1	LR

Table B.2: Top 20 Important Genes Found from the cDNA Data Using Cluster 2

Appendix C

Matlab Source Code

Matlab source code to run both the COSA1 and COSA2 routines can be found on the University of Edinburgh's Informatics DICE system, under the project/cosa/matlab directory of user s0343361. Included here are the headers of the routines for reference purposes.

Note that a matlab routine, quantile.m, which mimics the quantile function found in the *R* statistical package, is needed to run some of these routines. It can be downloaded from http://home.online.no/~pjacklam. Alternatively, if using the Matlab Stats toolbox the quantile functionality can be found by replacing the quantile function with the percentile.m function.

C.1 COSA

C.1.1 cosa1.m

This code implements the COSA1 algorithm.

```
function[clusters, weights, iter, qual1, qual2, centrediff] =
        cosa1(xdata, nclusters, lambda, alpha, niter, iweight, itargs,
              lquant, hquant, irobust, idebug)
%
% Matlab version of COSA1 algorithm by Friedman & Meulman 2004
```

```
%
% Input:
%
% xdata     - input data, nosobjsxnosfeats
% nclusters - number of clusters
% lambda    - the scale parameter
% alpha     - the homotopy parameter
% niter     - max number of iterations
% iweight   - distance decided on max. weights (33 - iweight=0) or
%             max. whole distances(30 - iweight=1)
% itargs    - targeted distances, 0-none, 1-lower, 2-upper, 3-both
% lquant    - lower quantile
% hquant    - upper quantile
% irobust   - 0 for mean calculations,
%             1 for medians calculation of dispersions
% idebug    - 0 for no debug, 1 for extra debug information,
%             3 for plots of 2d clusters and debug
%
% Output:
%
% clusters - row of clusters
% weights  - matrix of weights  - nclustersxnosfeats
% iter     - the last iteration step
% qual1    - quality of clustering 1
%            ( eqn 16 with mods just for centres to each point)
% qual2    - quality of clustering 2 ( eqn 23)
% centrediff - diff of kmeans centres
%
```

C.1.2 cosa2.m

This code implements the COSA2 algorithm.

```
function[diss, dissmatrix, weights, iter] = cosa2(xdists, lambda, alpha,
                                    niter, iweight, knear, irobust, idebug)
%
% Input:
%
% xdists    distances - nosobjsxnosobjsxnosfeats
% lambda  - the scale parameter
% alpha   - the homotopy parameter
% niter   - max number of iterations
%
% iweight - distance decided on weights (eqn 33 - iweight=0) or
%           whole distances(eqn 30 - iweight=1)
% knear   - nearest neighbours used in dispersion calc - default is
%           sqrt(nosobjs)
% irobust - robustness parameter 0 = means, 1 - medians for dispersion
%           calculations
% idebug  - 0 no debug, 1 debug output,
%           2 = plots of cluster quality at each iteration, no debug
%
% Output:
%
% diss       - vector of dissimilarities - nosobjsx(nosobjs-1)/2
%              lower traingle of dissmatrix columns
% dissmatrix - matrix of dissimilarities - nosobjsxnosobjs
%              useful for R program
% weights    - matrix of weights  - nosobjsxnosfeats
% iter       - the last iteration step
%
```

C.2 Attribute Distance Calculations

C.2.1 dists.m

This code calculates the distances between attributes of a data set.

```
function[dist,diff] = dists(xdata, itargs, irobust, idist, lquant, hquant)
%
% Matlab version of COSA2 algorithm by Friedman & Meulman 2004
% Computation of distances (euclidian and squared euclidian
%                            - see idist arg. )
% with input of targeted attributes
%
% Input:
%
% xdata is array of data, rows are nosobjs objects,
%              columns are nosfeats features/attributes/dimensions
% itargs - targeted data: 0 - no target, 1 low target = lower quantile
%                           2 - high target higher quantile,
%                           3 - both lower and upper
% irobust - 0 - use means, 1 - use medians in dispersion calculations
% idist   - 0 - use absolute difference |xik-xjk|, 1 - use square of this
% lquant  - lower quantil values for itargs >=1
% hquant  - upper quantile for itargs >=2
%
% Output:
%
% dist - matrix of distances - nosobjsxnosobjsxnosfeats
% diff - matrix of actual distances (not divided by dispersion of attribute)
%        nosobjectsxnosobjsxnosfeats
```

C.2.2 distsupdate.m

This code updates the distances between attributes of a data set for use in the COSA1 algorithm.

```
function[dist] = distsupdate(xdata, diffsexist, istart, itargs, irobust,
                             idist, lquant, hquant)


%
% Computation of distances (euclidian and squared euclidian -
%                             see idist arg. )
% with input of targeted attributes
%
% Additional routine that updates distances on certain rows/columns
% given an existinbg ditsnce matrix - for use with COSA1
%
% Input:
%
% xdata is array of data, rows are nosobjs objects,
%           columns are nosfeats features/attributes/dimensions
% diffsexist - existing actual distance array calculated from dists
% istart - start row of xdata where updated dists need be calculated
% itargs - targeted data: 0 - no target,
%                         1 low target = lquant quantile
%                         2 - high target hquant quantile,
%                         3 - both lower and upper
% irobust - 0 - use means, 1 - use medians in dispersion calculations
% idist   - 0 - use absolute difference |xik-xjk|, 1 - use square of this
% lquant  - lower quantil values for itargs >=1
% hquant  - upper quantile for itargs >=2
%
% Output:
%
```

```
% dist - matrix of distances - nosobjsxnosobjsxnosfeats
%
```

C.2.3 distsbridge.m

This code calculates the maximum distances between attributes of two data sets for use in bridging two data sets.

```
function[dist,dist1,dist2] = distsbridge(xdata1, xdata2, itargs, irobust,
                                                    idist, lquant, hquant)


%
% Computation of distances (euclidian and squared euclidian -
%                           see idist arg. )
% with input of targeted attributes.
%
% This is the bridged version where two data sets are input
% and a single distance matrix
% is output based on the maximum distance between points along an attribute
%
% Input:
%
% xdata1 is array of data, rows are nosobjs objects,
%                 columns are nosfeats features/attributes/dimensions
% xdata2 is array of data, rows are nosobjs objects,
%                  columns are nosfeats features/attributes/dimensions
% itargs - targeted data: 0 - no target,
%                         1 low target = lower quantile
%                         2 - high target higher quantile,
%                         3 - both lower and upper
% irobust - 0 - use means, 1 - use medians in dispersion calculations
% idist   - 0 - use absolute difference |xik-xjk|, 1 - use square of this
% lquant  - lower quantil values for itargs >=1
```

```
% hquant  - upper quantile for itargs >=2
%
% Output:
%
% dist  - matrix of distances - nosobjsxnosobjsxnosfeats
% dist1 - matrix of distances for data 1 - nosobjsxnosobjsxnosfeats
% dist2 - matrix of distances for data 2 - nosobjsxnosobjsxnosfeats
% above two for checks only
```

C.3 Attribute Importance

C.3.1 importance.m

This code determines the importance of each attribute in a group of objects.

```
function[att, imp] = importance(xdatadists, cluster, maximp)
%
% Find importance of attributes in a cluster using
% inverse dispersion measure
%
% Input:
%
% xdatadists - input data distances, nosobjsxnosobjsxnosfeats
% cluster   - cluster of interest, array of size  sizeclus
% maximp    - the maximum importance number (1/nos of attributes of interest)
%
% Output:
%
% att  - attributes in decreasing order of importance
% imp  - associated importance of the attributes
%
```

C.3.2 importancef.m

This code determines the importance of each attribute in a group of objects.

```
function[att, imp] = importancef(xdata, cluster, maximp)
%
% Find importance of attributes in a cluster on (inverse) dispersion measure
% FAST calculation
%
% Input:
%
% xdata     - input data, nosobjsxnosfeats
% cluster   - cluster of interest, array of size  sizeclus
% maximp    - the maximum importance number (1/nos of attributes of interest)
%
% Output:
%
% att  - attributes in decreasing order of importance
% imp  - associated importance of the attributes
%
```

Bibliography

[1] Hubank, M. Gene expression profiling and its application in studies of haematological malignancy. *British Journal of Haematology* **124**, 577-594 (2004).

[2] Miller, L. D. Optimal gene expression analysis by microarrays. *Cancer Cell* **2**, 353-361 (2002).

[3] Smyth, G. K., Yang, Y. H. & Speed, T. Statistical Issues in cDNA microarray data analysis. *Functional Genomics: Methods and Protocols* (2002).

[4] Medvedovic M. Determining the number of replicates needed to detect differentially expressed genes in DNA array experiments. Tutorial by Mario Medvedovic, University of Cincinnati Medical Centre. Available from medvedm@email.uc.edu.

[5] Brazma, A. & Vilo, J. Gene expression data analysis. *FEBS Letts.* **480**, 17-24 (2000).

[6] Butte, A. The use and analysis of microarray data. *Nature Reviews Drug Discovery* **1**, 51-960 (2002).

[7] Szallasi, Z. Genetic network analysis - from the bench to computers and back. *Tutorial 2nd ICSB* (2001).

[8] Friedman, N., Linial, M., Nachman I. & Pe'er D. Using Bayesian Networks to Analyse Expression Data. *Journal of Computational Biology* **1**, 601-620 (2000).

[9] Kuo, W. P. *et al.* Analysis of matched mRNA measurements from two different microarray technologies. *Bioinformatics* **18**, 405-412 (2002).

[10] Smith, M. W. What can I do about missing Data? http://www.herc.research.med.va.gov/FAQ_I9.htm (2004)

[11] Golub, T. R. *et al*. Molecular Classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* **286**, 531-537 (1999).

[12] Ben-Dor A. *et al*. Tissue classification with gene expression profiles. *Proc. IC-CMB* 1-32 (2000).

[13] Xing, E. P., Jordan, M. I. & Karp, R. M. Feature Selection for high-dimensional genomic microarray data. *Machine Learning: Proc. of 18th Int. Conf.* San Mateo CA (2001)

[14] Getz G., Levine E. & Domany E. Coupled two-way clustering analysis of gene microarray data. *arkiv.physics* **1**, 1-8 (2000).

[15] Friedman, J. H. & Meulman, J. J. Clustering objects on subsets of attributes. *Presented at the Royal Statistical Society London, May 2004*. Available from http://www-stat.stanford.edu/˜jhf/ftp/cosa.pdf (2004).

[16] Chiaretti, S. *et al*. Gene expression profile of adult T-cell acute lymphocytic leukaemia identifies distinct subsets of patients with different response to therapy and survival. *Blood* **103**, 2771-2778 (2004).

[17] Van de Vijver, M., J. *et al*. A Gene expression signature as a predictor of survival in breast cancer. *N. Engl. J. Med.* **347**, 1999-2009 (2002).

[18] Alizadeh, A. A. *et al*. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* **403**, 503-511 (2000).

[19] Hastie, T. *et al*. 'Gene Shaving' as a method for identifying distinct set of genes with similar expression patterns. *Genome Biology* **1(2)**:research0003.1-0003.21 (2000).

[20] Shipp, M. A. *et al*. Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature Medicine* **8**, 68-74 (2002).

[21] Rosenwald, A. *et al*. The use of molecular profiling to predict survival after chemotherapy for diffuse large B-cell lymphoma. *N. Engl. J. Med.* **346**, 1937-1947 (2002).

[22] Wright, G. *et al*. A gene expression-based method to diagnose clinically distinct subgroups of diffuse large B cell lymphoma. *PNAS* **100**, 9991-9996 (2003).

[23] Ferl, G. Z., Timmerman, J. M. & Witte, O. N. Extending the utility of gene profiling data by bridging microarray platforms. *PNAS* **100**, 10585-10587 (2003).

[24] Li, G. & Gui J. Partial Cox regression analysis for high-dimensional microarray gene expression data. *Bioinformatics* **20**, i208-i215 ISMB/ECCB (2004).

[25] Fleming, T. H. & Harrington, D. P. Non parametric estimation of the survival distribution in censored data. *Comm. in Statistics* **13** 2469-2486 (1984).

[26] Fleming, T. H. & Harrington, D. P. A class of rank test procedures for censored survival data. *Biometrika* **69** 553-566 (1982).