Improved biclustering of microarray data demonstrated through systematic performance tests

Heather Turner*, Trevor Bailey, Wojtek Krzanowski

Department of Mathematical Sciences, University of Exeter, Laver Building, North Park Road, Exeter EX4 4QE, UK

Received 1 September 2003; received in revised form 4 February 2004; accepted 10 February 2004

Abstract

A new algorithm is presented for fitting the plaid model, a biclustering method developed for clustering gene expression data. The approach is based on speedy individual differences clustering and uses binary least squares to update the cluster membership parameters, making use of the binary constraints on these parameters and simplifying the other parameter updates. The performance of both algorithms is tested on simulated data sets designed to imitate (normalised) gene expression data, covering a range of biclustering configurations. Empirical distributions for the components of these data sets, including non-systematic error, are derived from a real set of microarray data. A set of two-way quality measures is proposed, based on one-way measures commonly used in information retrieval, to evaluate the quality of a retrieved bicluster with respect to a target bicluster in terms of both genes and samples. By defining a one-to-one correspondence between target biclusters and retrieved biclusters, the performance of each algorithm can be assessed. The results show that, using appropriately selected starting criteria, the proposed algorithm out-performs the original plaid model algorithm across a range of data sets. Furthermore, through the rigorous assessment of the plaid model a benchmark for future evaluation of biclustering methods is established.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Biclustering; Two-way clustering; Overlapping clustering; Artificial microarray data; Performance evaluation; Bicluster quality measures

* Corresponding author. Tel.: +44-1392-264465; fax: +44-1392-264460.
E-mail address: heather.l.turner@exeter.ac.uk (H. Turner).
1. Introduction

There has been considerable interest recently in the analysis of microarray data. Microarray experiments are widely used to discover information about gene function. A typical experiment will investigate thousands of genes, recording their expression level over tens of samples, perhaps representing a range of experimental conditions or different classes. Genes with similar expression patterns over the samples are said to be co-regulated, which may indicate a common function. Likewise, samples with similar expression profiles may have attributes in common, for example they may be samples from patients with the same disease. With the aim of identifying such groups and samples, clustering has a natural role in the exploratory analysis of microarray data.

Most conventional clustering methods seek to find a one-way partition of the data, yet there are a number of scenarios that occur in microarray experiments that do not fit this model. Firstly, a gene may be involved in more than one biological process and exhibit an expression profile that is a combination of the regulation induced by each process. If there are other genes that are involved in only one of these processes, the structure should be represented by two overlapping clusters. Secondly, a group of genes may be co-regulated under limited conditions. In this case, the structure should be represented by a two-way cluster or bicluster, a group of genes and an associated group of samples over which the genes are co-regulated. Finally, genes may not be related to the subject of the investigation and exhibit near-constant expression profiles. Since conventional clustering methods cluster all the genes, a common practice is to filter out “uninteresting” genes on the basis of expression level, variance of expression profile or other ad hoc criteria. However, this can result in a large proportion of the data being thrown away which may include important genes. Gene selection should be an integral part of the clustering process, so that genes that do not exhibit interesting patterns are left unclustered. All these scenarios have their equivalents in terms of samples, for example a cluster of samples may only be distinguished by a cluster of genes.

Several clustering methods have been developed in recent years to address one or more of these issues. These include gene-shaving (Hastie et al., 2000), a one-way overlapping clustering method; context-specific Bayesian clustering (Barash and Friedman, 2002), a one-way clustering method in which clusters may be based on different subsets of samples; EMMIX-GENE (McLachlan et al., 2002) and interrelated two-way clustering (Tang et al., 2001), which cluster samples on the basis of gene clusters; “simultaneous” clustering (Pollard and van der Laan, 2002) which uses compositions of gene and sample clustering functions; coupled two-way clustering (Getz et al., 2000) a recursive two-way clustering method; rich probabilistic models (Segal et al., 2001) which models the dependency of expression levels on gene and array attributes which may include (possibly a priori) gene and array clusters; double conjugated clustering (Busygin et al., 2002), which identifies non-overlapping biclusters, and finally SAMBA (Tanay et al., 2002), order preserving submatrix clustering (Ben-Dor et al., 2002), biclustering (Cheng and Church, 2000) and the plaid model (Lazzeroni and Owen, 2002) which all identify non-exhaustive potentially overlapping biclusters.
The final group of clustering methods can accommodate all the scenarios described earlier, yet there are further methodological features that separate these methods. Order preserving submatrix clustering seeks genes with the same rank profile across a subset of samples. Although genes with the same rank profile must have an expression profile of the same pattern, the converse is not necessarily true, therefore we find this method unnecessarily prescriptive. SAMBA requires the gene expression levels to be converted to up-regulated (+1), down-regulated (−1) or unchanged (0) levels. There is clearly a risk that some observations will be misclassified in this conversion process, making it harder to discover the underlying structure of the data. To allow greater sensitivity, the SAMBA algorithm is being developed to allow multiple response levels (Tanay et al., 2002).

We prefer the use of the original gene expression levels to find biclusters with similar expression patterns. This is the aim of Cheng and Church’s biclustering method and the plaid model. Cheng and Church biclusters include all samples for which the genes are co-regulated, even if the expression is at the “normal” level for that gene. This type of bicluster includes “flat” biclusters with near constant gene expression over the samples, which are not of interest. The plaid model has the advantage that it estimates the normal level of expression for each gene, then seeks biclusters of genes that have similarly unusual expression levels over the biclustered samples. Since biologists are generally interested in identifying changes in gene expression and the samples in which they have changed, we find this type of bicluster to be the most appropriate.

The features of the plaid model make it an attractive method for clustering gene expression data. However, we believe that the efficiency of the original plaid model algorithm proposed by Lazzeroni and Owen (2002) is compromised by the relaxation of binary constraints on the cluster membership parameters in the model. Our first objective in this paper is to introduce a plaid model algorithm that takes advantage of the binary constraints on the cluster membership parameters, providing a simpler and more direct method of optimisation.

This raises the question of how to compare the performance of our proposed plaid model algorithm to the original algorithm. Despite the development of numerous biclustering and two-way clustering techniques for analysing microarray data, little systematic work has been conducted on evaluating the performance of these methods. There has been a focus on validating and interpreting the output from real data, a process clearly necessary for the methods to be of any use, but giving no indication of the error involved, particularly considering both dimensions of the resulting clusters. Typical approaches are comparing the retrieved clusters to those retrieved from randomised data to confirm the algorithm has found some true structure (Kluger et al., 2003; Tanay et al., 2002) or comparing the clusters to a known classification of samples or gene annotation (Tanay et al., 2002; Busygin et al., 2002; Tang et al., 2001; Getz et al., 2000). These two approaches are known as intrinsic validation and external validation.

Although validation against an external classification can be used to compare algorithms, this is not a solid basis from which to make general conclusions. Conformity to external classification does not imply conformity to underlying structure. Furthermore, due to practical considerations, such studies usually consider only two or three data sets which provide limited evidence for the superiority of a method.
In order to test an algorithm on a large number of data sets where the structure is known, one must turn to artificially constructed data. Some researchers have used simulated data to evaluate two-way clustering methods; however, this approach has not been used to its full potential. For example, simulations may only consist of one or two data sets (Segal et al., 2001; Getz et al., 2000) or data sets with only one cluster (Ben-Dor et al., 2002). The simulated data sets are generally simpler than those encountered in practice; they typically involve a relatively small number of genes and do not allow for error over all data points (Segal et al., 2001; Barash and Friedman, 2002—strictly one-way clustering, but results may be viewed as a set of biclusters that partition one dimension of the data matrix). Even though two-way clustering methods are examined, the clusters are evaluated on a one-way basis.

Our second objective in this paper is to introduce a rigorous approach to test the performance of biclustering methods. The key features of our approach are the construction of realistic microarray data sets, replicated testing over a range of cluster structures, and two-dimensional evaluation of the retrieved biclusters. Through this systematic approach, we are able to show that our proposed plaid model algorithm out-performs the original algorithm at recovering underlying cluster structure.

We assume throughout the paper that we are clustering a gene by sample expression matrix, but note that the plaid model may also be used to analyse data from other applications as illustrated in the paper by Lazzeroni and Owen (2002).

2. The plaid model

The plaid model consists of a series of additive layers intended to capture the underlying structure of a gene expression matrix. The model includes a background layer containing all the genes and samples, to account for global effects in the data. Any subsequent layers represent additional effects corresponding to biclusters that exhibit a strong pattern not explained by the general model.

In the plaid model the expression level, \( Y_{ij} \), \( i = 1, \ldots, n; j = 1, \ldots, p \) of the \( i \)th gene in the \( j \)th sample is modelled by

\[
Y_{ij} = \Theta_{ij0} + \sum_{k=1}^{K} \Theta_{ijk} \rho_{ik} \kappa_{jk} + \epsilon_{ij}
\]

\[
= (\mu_0 + \alpha_{i0} + \beta_{j0}) + \sum_{k=1}^{K} (\mu_k + \alpha_{ik} + \beta_{jk}) \rho_{ik} \kappa_{jk} + \epsilon_{ij},
\]

where \( k \) is a layer index starting at zero for the background layer running to \( K \), the number of biclusters; \( \Theta_{ijk} \) is the sum of mean, gene and sample effects in layer \( k \); \( \rho_{ik} \) is a binary cluster membership parameter defined for \( k \geq 1 \) and equal to one if the \( i \)th gene is in the \( k \)th bicluster, zero otherwise; \( \kappa_{jk} \) similarly indicates cluster membership for the \( j \)th sample, and \( \epsilon_{ij} \) is residual error. Variants of the model may be obtained by simplifying \( \Theta_{ijk} \), but the form presented here is most suitable for analysing microarray data.
The background layer is fitted first of all, then bicluster-specific layers are added one at a time until a pre-specified number is reached or no more significant layers can be found, as determined by a permutation test. The original algorithm for finding a bicluster and fitting the corresponding layer uses alternating ordinary least squares (OLS) and is summarised in Algorithm 1 (the layer index, $k$, is dropped for clarity).

Algorithm 1 first relaxes the constraints on $\hat{\beta}_i$ and $\hat{\kappa}_j$, allowing non-binary estimates, which are then gradually shifted towards binary solutions as the algorithm progresses. Even though the optimisation process is directed towards a binary solution, the final estimates for $\hat{\beta}_i$ and $\hat{\kappa}_j$ are still binary approximations of continuous least squares estimates. Furthermore, the gradual shifting of estimates introduces a discontinuity, which may prevent the algorithm converging to a superior solution to the final result. Indeed, the algorithm can only converge during the final $T$ iterations in which $\hat{\beta}_i$ and $\hat{\kappa}_j$ are rounded to binary values. Our experience suggests that the value of $T$ used in the publicly available PlaidTM program (Owen, 2004) is generally insufficient to reach convergence. As biclusters are added sequentially, non-convergence for one bicluster can have a knock-on effect on later biclusters.

Algorithm 1. Original plaid model algorithm

1: Compute $\hat{Z}$, the matrix of residuals from the model so far
2: Compute starting values $\hat{\beta}^{(0)}_i$ and $\hat{\kappa}^{(0)}_j$
3: for $s = 1 : S$ do
4: Compute $\hat{\mu}^{(s)}, \hat{\beta}_i^{(s)}$ and $\hat{\kappa}_j^{(s)}$ using OLS estimates with $\hat{\beta}_{i}^{(s-1)}$ and $\hat{\kappa}_{j}^{(s-1)}$
5: Compute $\hat{\beta}_i^{(s)}$ using OLS estimate with $\hat{\mu}^{(s)}, \hat{\beta}_i^{(s)}, \hat{\kappa}_j^{(s)}$ and $\hat{\kappa}_{j}^{(s-1)}$
6: Compute $\hat{\kappa}_j^{(s)}$ using OLS estimate with $\hat{\mu}^{(s)}, \hat{\beta}_i^{(s)}, \hat{\beta}_j^{(s)}$ and $\hat{\beta}_{j}^{(s-1)}$
7: Shift any $\hat{\beta}_i^{(s)}$ and $\hat{\kappa}_j^{(s)}$ to $0 + \frac{s}{2(S - T)}$ if greater than 0.5
8: Shift any $\hat{\beta}_i^{(s)}$ and $\hat{\kappa}_j^{(s)}$ to $0 - \frac{s}{2(S - T)}$ if less than 0.5
9: end for
10: Compute $\hat{\mu}^{(S+1)}, \hat{\beta}_i^{(S+1)}$ and $\hat{\kappa}_j^{(S+1)}$
11: Calculate candidate layer sum of squares $\text{LSS}_c = \sum_{i,j} (\hat{\mu} + \hat{\beta}_i + \hat{\beta}_j)\hat{\beta}_i\hat{\kappa}_j$
12: for $m = 1 : M$ do
13: Permute $\hat{Z}$
14: Repeat search for bicluster
15: Calculate layer sum of squares $\text{LSS}_m$
16: end for
17: if $\text{LSS}_c > \max(\text{LSS}_1, \ldots, \text{LSS}_M)$ then
18: Accept bicluster and search for next layer
19: else
20: stop

---

$^1$Rounded to zero if within predetermined tolerance.
In response to these issues, we propose the use of binary cluster membership parameters throughout the optimisation process to preserve continuity and allow quicker convergence. With reference to the original algorithm, Lazzeroni and Owen (2002) suggest that allowing  $\hat{r}_i$ and $\hat{k}_j$ to move too quickly towards binary values may cause the algorithm to “lock in” to a suboptimal solution. However, they provide no evidence to show that restricting the values of $\hat{r}_i$ and $\hat{k}_j$ to $0.5 \pm s/(S - T)$ actually improves performance, rather than simply slowing convergence. Our later results suggest that restricting to binary values for $\hat{r}_i$ and $\hat{k}_j$ throughout the algorithm leads to equivalent specificity (recovery of true structure) to the “gradual shifting” approach but also greater sensitivity (proportion of true structure in recovered structure).

Using binary cluster membership parameters also improves the computational efficiency of each iteration. When all genes and samples have “fuzzy” membership of a bicluster, the full data set must be used to estimate the effects, $\hat{\beta}_k$, $\hat{\alpha}_{ik}$ and $\hat{\alpha}_{jk}$, for the corresponding layer. On the other hand, if genes and samples have a “hard” membership of a bicluster, only expression levels for the biclustered genes in the biclustered samples are needed to fit the corresponding layer. For large microarray experiments, this results in a noticeable reduction in the computation involved.

3. Additive clustering

The basis of our alternative plaid model algorithm is provided by speedy individual differences clustering, or SINDCLUS (Chaturvedi and Carroll, 1994), an algorithm originally proposed to fit a range of additive clustering models.

Additive clustering was introduced by Shepard and Arabie (1979) for clustering a set of objects based on a similarity matrix. The original model, ADCLUS, is a special case of the plaid model in which rows and columns are clustered symmetrically and a constant effect is fitted to each layer instead of mean, row and column effects. Following the notation of the previous section, this may be written as

$$Y_{ij} = \mu_0 + \sum_{k=1}^{K} \mu_k \rho_{ik} \rho_{jk} + \epsilon_{ij}.$$  

Various algorithms have been proposed to fit this model (Shepard and Arabie, 1979; Arabie and Carroll, 1980; Mirkin, 1987; Tenenbaum, 1996; Lee, 1999, 2002) and generalisations of the model (DeSarbo, 1982). However, SINDCLUS is of particular interest as it was designed for large data sets and does not rely on features of similarity data that are unlike those of microarray data. In addition, SINDCLUS has a similar structure to the plaid model algorithm and allows non-symmetric clustering of rows and columns.

In SINDCLUS, the number of clusters is fixed and the parameters of the model are estimated one cluster at a time, alternating between fitting the cluster-specific effect and estimating the cluster membership parameters. The cluster effect is estimated using OLS, but the cluster membership parameters are estimated using binary least squares.
Given a model for data $Z_{ij}$ of the form
\[ Z_{ij} = \rho_i x_j + e_{ij}, \]
in which $\rho_i$ is an unknown binary parameter and $x_j$ is known, binary least squares estimates $\hat{\rho}_i$ minimise
\[ \sum_i \sum_j (Z_{ij} - \rho_i x_j)^2. \]
For a particular value of $i$, we have
\[ \sum_j (Z_{ij} - \rho_i x_j)^2 = f_i(\rho_i) \]
as $Z_{ij}$ and $x_j$ are known for all $j$. Therefore, the total sum of squared errors is separable in $\rho_i$ and we can minimise the complete sum by separately minimising $f_i(\rho_i)$ for each $i$. As $\rho_i$ can only be 0 or 1, the minimisation can be feasibly achieved by a process of trial and error.

4. Binary least squares algorithm

Following the SINDCLUS method, we use binary least squares to fit the cluster membership parameters in the plaid model and thereby avoid the problems of relaxation. This section describes our algorithm in detail, starting with the core (learning cluster membership and fitting layer effects), then discussing certain optional features and initialisation.

In the following description of the core iterations, we suppose that $l-1$ layers have been fitted, so that the current residuals are given by
\[ \hat{Z}_{ij} = Y_{ij} - \hat{\Theta}_{ij0} - \sum_{k=1}^{l-2} \hat{\Theta}_{ijk} \hat{\rho}_k \hat{\kappa}_{jk}. \]
Dropping the layer subscript for clarity, we consider the search for an $l$th layer of the form
\[ \hat{Z}_{ij} = \Theta_{ij} \rho_i \kappa_j + \hat{e}_{ij} = (\mu + \alpha_i + \beta_j) \rho_i \kappa_j + \hat{e}_{ij} \]
that minimises the residual sum of squares, or, equivalently and more conveniently
\[ Q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} (\hat{Z}_{ij} - \Theta_{ij} \rho_i \kappa_j)^2. \]
The search is initialised with starting values for $\hat{\rho}_i$ and $\hat{\kappa}_j$.

4.1. Updating the layer effects

Given current binary values for $\hat{\rho}_i$ and $\hat{\kappa}_j$, the layer effects can be updated using the reduced set of data, $Z^\ast$, corresponding to the genes for which $\hat{\rho}_i$ equals one and the
samples for which $\hat{k}_j$ equals one. Least-squares-estimates for the layer effects are then identical to two-way analysis of variance estimates

$$\hat{\mu} = \hat{Z}_\mu^*, \quad \hat{\alpha}_i = \hat{Z}_\alpha^* \hat{\mu}, \quad \text{and} \quad \hat{\beta}_j = \hat{Z}_\beta^* \hat{\mu}.$$  

For genes and samples that are not in the bicluster, no effects are added. However, $\hat{\alpha}_i$ and $\hat{\beta}_j$ must be defined for all genes and samples, respectively, so that an iterative estimation procedure may be used. Therefore, we define $\hat{\alpha}_i = 0 \forall i$ and $\hat{\beta}_j = 0 \forall j$.

### 4.2. Updating the cluster membership parameters

We update the cluster membership parameters one at a time using binary least squares, assuming all the other parameters are fixed. In particular, given $\hat{\mu}, \hat{\alpha}_i, \hat{\beta}_j$ and $\hat{k}_j$, $\hat{\rho}_i$ is updated using binary least squares to fit

$$\hat{Z}_{ij} = \rho_i[\hat{k}_j(\hat{\mu} + \hat{\alpha}_i + \hat{\beta}_j)] + e_{ij}.$$  

Similarly given $\hat{\mu}, \hat{\alpha}_i, \hat{\beta}_j$ and $\hat{\rho}_i$, $\hat{k}_j$ is updated using binary least squares to fit

$$\hat{Z}_{ij} = \kappa_j[\hat{\rho}_i(\hat{\mu} + \hat{\alpha}_i + \hat{\beta}_j)] + e_{ij}.$$  

As in the original algorithm, $\hat{\rho}_i$ and $\hat{k}_j$ are updated in parallel, in other words when $\hat{\rho}_i$ is updated, the estimate from the previous iteration is used for $\hat{k}_j$ and vice versa, so that the order in which the cluster membership parameters are updated is not important.

The effects and the cluster membership are alternately updated until convergence (or the number of iterations exceeds a predefined limit).

### 4.3. Pruning biclusters

The original plaid model algorithm includes an option to prune out any ill-fitting genes and samples from a bicluster once the core iterations are completed. The cluster membership parameters are adjusted as follows:

$$\hat{\rho}_i = \begin{cases} 1 & \text{if } \hat{\rho}_i = 1 \text{ and } \sum_{j,\hat{k}_j=1} (\hat{Z}_{ij} - \hat{\Theta}_{ij})^2 < (1 - \tau_1) \sum_{j,\hat{k}_j=1} \hat{Z}_{ij}^2, \\ 0 & \text{otherwise,} \end{cases}$$

$$\hat{k}_j = \begin{cases} 1 & \text{if } \hat{k}_j = 1 \text{ and } \sum_{i,\hat{\rho}_i=1} (\hat{Z}_{ij} - \hat{\Theta}_{ij})^2 < (1 - \tau_2) \sum_{i,\hat{\rho}_i=1} \hat{Z}_{ij}^2, \\ 0 & \text{otherwise,} \end{cases}$$

where $\tau_1, \tau_2 \in (0, 1)$ specify for genes and samples, respectively, the minimum proportional reduction in residual sum of squares required for cluster membership. If all the genes or all the samples are pruned out, the algorithm is terminated. Otherwise, the layer effects are updated once more and the layer may be checked for significance as described in Section 2.

Since we prefer small, tight biclusters in which the feature of the bicluster is clear in all members of the bicluster, we use $\tau_1, \tau_2 \in (0.5, 0.7)$, high enough to suggest the
bicluster feature is an important component of the member expression profiles, but also allowing for other components of the data, i.e. overlapping biclusters or error.

4.4. Back fitting

After a layer has been added to the model, the layer effects for all layers in the current model may be re-estimated in the light of this additional structure. This is particularly important for the background layer as this represents the base expression level for all genes and samples and a good estimate will make it easier to identify a further bicluster, if it exists, in the next round of iterations.

The back fitting is done sequentially, fitting each layer to the residuals from the model excluding that layer. To reach convergence, several rounds of back fitting would generally be required. However, even one back fitting step will substantially improve the model, because the effect of the unusual expression levels corresponding to the most recent bicluster on the estimates of layer effects for previous layers is reduced when these effects are adjusted for the presence of the additional layer.

It is questionable whether extensive back fitting will improve the algorithm’s performance if the current model is only partially accurate in terms of cluster membership. However, the residuals used to search for the next bicluster should be reasonable approximations of the residuals that would be obtained if optimal layer effects were fitted in the current model. Therefore, we suggest using one to three back fitting steps as a tradeoff between accuracy of layer effects and computational burden.

4.5. Starting values

In the original algorithm, starting values for $\hat{\rho}_i$ and $\hat{\kappa}_j$ are derived from approximations of the first left and right singular vectors of the data matrix, since these are the best one-dimensional representations of the data. This is not appropriate for our proposed algorithm as binary starting values are required. In the context of clustering similarity matrices, Chaturvedi and Carroll (1994) used random binary vectors to initialise the SINDCLUS algorithm for fitting additive clustering models, but for large microarray data sets our experience is that a more sophisticated approach is needed.

The cluster membership parameters simply split the genes and samples into two groups: those that are in the bicluster and those that are not. Finding starting values for the cluster membership parameters can therefore be viewed as a univariate clustering problem, with a known number of groups. We use the $k$-means algorithm with $k=2$ to cluster the genes and the samples independently and then take the cluster with fewer members from each result to form the starting bicluster (since the minority group has “unusual” gene expression in the context of the given data set).

However, as discussed in Section 2, there is a possibility that restricting $\rho_i$ and $\kappa_j$ to binary values may cause the algorithm to “lock in” to a locally optimal solution. Therefore, we also consider a starting method that seeks to broaden the search by combining results from multiple starting values, each favouring a different feature of the data. We obtain one starting bicluster using the $k$-means algorithm as above and
three other starting biclusters that include genes and samples with, respectively, higher variance, higher skewness or lower kurtosis than the full data set.

Each starting bicluster is used to search for a bicluster to add to the current plaid model. The four resulting seed biclusters are then used to determine a “stronger” starting bicluster to search for a final candidate bicluster. The seed biclusters approximate one or more biclusters which may include false biclusters. We wish to find the true bicluster with the highest layer sum of squares (LSS) out of the true biclusters not yet added to the model. Therefore, we consider a gene or sample to be a strong candidate for cluster membership if it belongs to a seed bicluster with high LSS and we favour genes and samples that belong to more than one seed bicluster.

In particular, if \( m \) is the mean number of genes in the seed biclusters, then the genes selected for the new starting bicluster belong to the intersection, \( I \), of the gene sets of the \( b \) highest ranking seed biclusters in order of LSS, where \( b \) is the maximum number such that \(|I| \geq \max(3, 0.1m)\). The samples are selected in a similar manner. We impose a minimum size on the starting bicluster to avoid focusing the search on a small overlap between quite distinct biclusters.

5. Constructing artificial microarray data

Since we are motivated by the application of microarray experiments, we wished to test the performance of the original algorithm against our suggested method on convincing simulated microarray data. Simulated data allows systematic consideration of factors that may affect the performance of the clustering algorithms; however, the characteristic features of real microarray data need to be built in to the simulation if the evaluation is to give a good idea of how the method may perform in practice.

5.1. Decomposition of real microarray data

There is currently no generally accepted model for the error in gene expression measurements. Therefore, to obtain a realistic error distribution we dismantled a “null” microarray experiment in which there was a hypothesis of no difference between arrays.

The data came from a test experiment comparing RNA extraction using PAX tubes to RNA extraction using the Trizol method. There are two cDNA microarrays for each method, each using RNA from a human whole blood sample in the red channel and a standard reference in the green channel. Since the blood sample is the same for each array, we expect the gene expression levels to be the same on each array regardless of the method used to extract RNA from the blood sample.

We first examined the data to ascertain which normalisation procedures would be necessary to remove systematic experimental effects. Plotting signal intensities against printing order revealed substantial plate effects. We removed these effects using an early version of the robust plate normalisation technique available in the LIMMA R package (Smyth and Speed, 2003; WEHI, 2004; Ihaka and Gentleman, 1996) adapted for the printing order used on our arrays. Image plots of the background-corrected signals showed marked spatial effects. However, there was a large proportion of flagged data,
making a print-tip correction unsuitable due to the low number of usable spots in certain print-tip sections. Hence, a global LOESS spatial correction was performed on the background-corrected intensities for each channel (Colantuoni and Henry, 2004, similar to SNOMAD, Colantuoni and Henry, 2004). Following this an intensity-dependent normalisation was applied to the log-ratio (base 2) of red to green intensities. Finally, a scale correction was applied to make the four arrays comparable. The last two steps follow the methods of Yang et al. (2002). After removing genes with flagged or missing data on any of the arrays and averaging over replicates within arrays where applicable, there were 19,358 genes in the data set.

Following normalisation, the main effects (constituting the background layer) were removed from the data. The residuals from this process were taken as the empirical error distribution, assuming that the null hypothesis was true and there would be no further structure in the data. To validate this assumption, the original algorithm was run on the data and no significant biclusters were found.

5.2. Construction of artificial data

We designed an experiment to test the performance of the plaid model algorithms over a range of simulated microarray data sets with known cluster configuration. Full details of the design of our experiment are given in Section 7. To construct each set of data in the experiment, we simulated data with plaid model structure and added error sampled from the empirical error distribution discussed in the previous section. Each data set contained a pre-specified number of genes and samples. For the background layer, \( \mu \) was set to the mean of the RNA extraction data and the \( \alpha_i \) and \( \beta_j \) were, respectively, sampled from the gene and sample effects removed, as described in the previous section, during the decomposition of the null data set.

The number of biclusters was specified, then genes and samples were selected for cluster membership with fixed probability \( p_1 \) and \( p_2 \), respectively. If the number of genes (or samples) selected was greater than half the total number of genes (or samples) the complementary set was used instead. The bicluster was rejected if it was too similar to a previous bicluster as measured by various statistics introduced in Section 6.

For bicluster layers, the mean was sampled from \( U(4s_0, 2s_1) \) and randomly chosen to be positive or negative, where \( s_0 \) is the standard deviation of the error distribution and \( s_1 \) is the standard deviation of the gene effects from the RNA extraction data. This range was chosen so that the mean effect would be larger than that expected by chance but not an extreme effect. Gene and sample effects were both sampled from the empirical distribution of gene effects. The sample effects from the RNA extraction data were not used for bicluster layers because over thousands of genes, the sample effect is likely to be very small, whereas the sample effects may be large within a bicluster of co-regulated genes.

To check that the biclusters were detectable, a plaid model with the same cluster structure as the generating model was fitted to the simulated data, back fitting once after each layer was added to the model. The fitted model would not be exactly the same as the generating model due to the error added to the data and the limited back fitting. The residuals from this model would be those a plaid algorithm would find if
it correctly identified the cluster membership. The original algorithm was then used to find a further, false, bicluster. If the true biclusters had layer sums of squares that were greater than the false bicluster’s LSS, it was assumed that the biclusters were detectable (though not necessarily the optimal solution). Otherwise parts of the simulation were repeated to obtain a data set that passed this check.

6. Bicluster quality measures

To evaluate clustering results obtained in the performance evaluation, we adapted one-way quality measures that are commonly used in information retrieval (Strehl, 2002; Baeza-Yates and Ribeiro-Neto, 1999), the first two of which have also been used to associate external groups with two-way clusters in microarray data (Getz et al., 2000).

Suppose that we wish to compare a target bicluster $A$ and a retrieved bicluster $B$. Let

- $g_X$ be the number of genes in $X$,
- $s_X$ be the number of samples in $X$,
- $n_X = g_X s_X$ be the number of data points corresponding to $X$.

Our quality measures are defined as follows

- “sensitivity” $= \frac{g_A \cap B}{g_B} \times \frac{s_A \cap B}{s_B}$,
- “specificity” $= \frac{g_A \cap B}{g_A} \times \frac{s_A \cap B}{s_A}$,

and

- “$F_1$ measure” $= \frac{2(g_A \cap B)(s_A \cap B)}{n_A + n_B}$.

Sensitivity measures the proportion of data values in $B$ that are also in the target bicluster $A$ and specificity is the proportion of $A$ that has been retrieved in $B$. Both of these measures are simply the product of the one-way or marginal equivalents, as shown. The $F_1$ measure (Strehl, 2002; Baeza-Yates and Ribeiro-Neto, 1999) is the harmonic mean of the sensitivity and specificity, giving an overall measure of quality, which cannot be factorised into marginal components (the subscript refers to the equal weighting of sensitivity and specificity).

Strehl (2002) measured the quality of a set of clusters by finding the best match to each cluster in the target set and calculating the mean of the pair-wise quality measures, weighted by cluster size. We used a version of this method, adapted to our particular application.

In the performance evaluation, we fixed the number of retrieved biclusters to be the same as the number of biclusters in the artificial data set. Therefore, we chose to match biclusters via a one-to-one correspondence so that every retrieved bicluster contributed to the quality score.
The importance of a bicluster is not necessarily determined by the size of its membership, so we used the sum of squares of the corresponding layer (sum of squared simulated $\Theta_{ij}$) for the cluster weight. The one-to-one correspondence was determined by matching the target biclusters in decreasing weight order to the unmatched retrieved bicluster that gave the highest pair-wise $F_1$ measure. In the event that all the pair-wise $F_1$ measures were zero, the bicluster was not matched until all the other target biclusters had been considered.

7. Performance evaluation/results

The plaid model algorithms were tested on six types of data set, varying over two factors: the number of imposed biclusters and the type of overlap allowed between the biclusters. The number of biclusters was either 3, 5 or 10 and biclusters were allowed to overlap in either one dimension only or in both dimensions. When biclusters overlap in one dimension only, they may be considered as separate because the bicluster effects do not coincide.

The data sets in the evaluation were constructed with 6001 genes and 11 samples. This size was chosen so that the total number of elements was less than that in the empirical error distribution and represented a “typical” microarray experiment. The number of genes is approximately the size of the yeast genome and 11 arrays would be sufficient to investigate a small number of treatments. The dimensions of the data set were chosen to be odd numbers so that a bicluster could always be uniquely defined (i.e. the bicluster would be smaller than its three complements).

Samples were included in simulated biclusters with probability $p_2 = 0.5$ and genes were added with $p_1$ ranging evenly over the biclusters from 0.005 to 0.025, so that the expected bicluster size ranged from 30 to 150 genes, with five samples. For each type of cluster configuration, we simulated 10 data sets.

We tested the original algorithm and two implementations of our proposed algorithm, one for each of the methods to obtain starting values discussed in Section 4.5. To ensure that the type of algorithm was not confounded with the type of starting values, we also tested the original algorithm with $k$-means starting values. In our results, we refer to type of algorithm according to the method of least squares used to fit the cluster membership parameters: either OLS or binary least squares (BLS).

For each run, the algorithms were set to find the same number of biclusters as that in the constructed data sets without testing for significance, including empty biclusters if necessary. A high number of iterations was allowed to fit each layer, so that in general the fitted layer would be the result at convergence. Six iterations were used to find starting values, where applicable. Genes and samples were pruned with $\tau_1 = \tau_2 = 0.7$ and one round of back fitting followed the addition of each layer.

Fig. 1 shows the average $F_1$ measure, sensitivity and specificity over the 10 simulations for each method and each type of cluster configuration. The data used to produce these plots, along with corresponding standard deviations, are given in Table 1.

As expected, the quality of the retrieved cluster sets tends to decrease with increasing structural complexity, as shown by the $F_1$ measures in Fig. 1(A). It can also be seen in Table 1 that the variability of the quality measures decreases with increasing structural
complexity. This is because as the number of clusters increases the contribution of each individual cluster to the overall quality measure decreases and hence the effect on the quality measure of missing or identifying a particular cluster is not so great.

Overall, the BLS algorithms performed better than the OLS algorithms. However, it is also noticeable that the original OLS algorithm is improved by the use of \( k \)-means starting values as opposed to those derived from the singular vectors of the data matrix.

Looking at the sensitivity and specificity in Fig. 1(B) and (C), it is clear that the BLS algorithms are set apart by an improvement in sensitivity. This results from the clear cut definition of cluster membership, used at every iteration. Similarly, using \( k \)-means starting values for the original algorithm improves the sensitivity because there is a definite starting point, but the sensitivity does not reach the level of the BLS algorithms as fuzziness is allowed to creep in through the early OLS updates that are not rounded to binary values.

Interestingly, there is some evidence that the sensitivity of methods using binary starting values is higher for data sets with overlapping biclusters. Whilst, on the one hand, an overlap between biclusters increases structural complexity, it can also make the biclusters easier to find if both biclusters are regulated in the same direction. It would appear that such a double effect is easily identified when using binary starting values and results in a starting bicluster with high sensitivity.

Fig. 1(C) shows, as might be expected, that the specificity depends largely on the starting values: starting close to the true bicluster will make it easier to find. All the methods that use \( k \)-means starting values have similar specificity values on average and perform better than the original algorithm in this respect.
Table 1
Mean quality measures for each algorithm over simulations with the given cluster configuration (corresponding standard deviations are given in brackets)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Algorithm</th>
<th>Cluster Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3 separate</td>
</tr>
<tr>
<td>F1 measure</td>
<td>Ordinary (SV)</td>
<td>0.55 (0.20)</td>
</tr>
<tr>
<td></td>
<td>Ordinary (KM)</td>
<td>0.57 (0.20)</td>
</tr>
<tr>
<td></td>
<td>Binary (MULTI)</td>
<td>0.67 (0.15)</td>
</tr>
<tr>
<td></td>
<td>Binary (KM)</td>
<td>0.65 (0.19)</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>Ordinary (SV)</td>
<td>0.55 (0.24)</td>
</tr>
<tr>
<td></td>
<td>Ordinary (KM)</td>
<td>0.55 (0.22)</td>
</tr>
<tr>
<td></td>
<td>Binary (MULTI)</td>
<td>0.71 (0.18)</td>
</tr>
<tr>
<td></td>
<td>Binary (KM)</td>
<td>0.67 (0.21)</td>
</tr>
<tr>
<td>Specificity</td>
<td>Ordinary (SV)</td>
<td>0.61 (0.19)</td>
</tr>
<tr>
<td></td>
<td>Ordinary (KM)</td>
<td>0.70 (0.21)</td>
</tr>
<tr>
<td></td>
<td>Binary (MULTI)</td>
<td>0.69 (0.19)</td>
</tr>
<tr>
<td></td>
<td>Binary (KM)</td>
<td>0.71 (0.18)</td>
</tr>
</tbody>
</table>
Table 2
For each quality measure, the mean difference over all simulations between the BLS algorithm using $k$-means starting values and the given algorithm, followed by the standard error. Algorithms are described first by the method used to fit the cluster membership parameters and then by the method used to obtain starting values.

<table>
<thead>
<tr>
<th></th>
<th>OLS</th>
<th>BLS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Singular vectors</td>
<td>$k$-Means clusters</td>
</tr>
<tr>
<td>$F_1$ measure</td>
<td>0.137 (0.022)</td>
<td>0.066 (0.017)</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>0.195 (0.023)</td>
<td>0.135 (0.018)</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.091 (0.025)</td>
<td>$-0.001$ (0.020)</td>
</tr>
</tbody>
</table>

The specificity of most of the algorithms is similar between the separate and overlapping configurations for both five and 10 clusters. This reflects the fact that, although a high degree of overlapping was permitted in simulating the overlapping configurations, the random selection of bicluster members resulted in a relatively low degree of overlap in practice. On average 1.2% (s.d. 0.86%) of data values in the target clusters were in two clusters and 0.01% (s.d. 0.03%) were in three clusters.

Whilst the specificity of all the algorithms appears to be much the same, it is possible that the actual biclusters identified could be substantially different. Although we have not defined a method for comparing two retrieved bicluster sets, we can gain an idea of their similarity by comparing their respective cluster-level specificity measures. In particular, we can consider a target cluster and infer the minimum and maximum similarity in retrieval between the matched biclusters from two algorithms, by defining similarity in retrieval to be the proportion of the target bicluster that both biclusters found plus the proportion of the target bicluster that both biclusters did not find. On average, this range is approximately 50–70% for all pairwise comparisons, suggesting broadly similar results in terms of the target clusters identified and the degree to which they are retrieved.

Returning to the standard deviations in Table 1, there does not appear to be any difference in consistency of performance between the clustering algorithms. So taking the results as a whole, the highest performing algorithms are the BLS methods.

Comparing the two implementations of the BLS algorithm over all 60 simulated data sets, we find that the average $F_1$ measure and sensitivity are higher when starting values from seed biclusters are used. However, the difference is not significant, as can be seen in the comparison of all methods to the BLS algorithm using $k$-means starting values, shown in Table 2. It appears that the extra computation involved in finding seed biclusters is not worthwhile, and taking this into consideration, the BLS algorithm using $k$-means starting values comes out best over all.

8. Discussion

We have developed an improved and computationally efficient algorithm for fitting the plaid model. By using 60 simulated data sets constructed from real microarray data
components, we have shown that our algorithm performs consistently better than the original algorithm.

Quantifying two aspects of retrieval quality, we have been able to identify easily interpretable differences between the algorithms tested. As a consequence of developing a method that requires binary starting values, we have found that the sensitivity and specificity of the original algorithm can be increased by using binary starting values based on \( k \)-means clusters. However, switching to our method that uses binary cluster membership parameters throughout the iterations increases the sensitivity still further. The original plaid model algorithm has been criticised for producing large heterogeneous clusters (Segal et al., 2003); it appears that our alternative algorithm has gone some way to addressing this issue.

The quality of the clustering results in our evaluation serve as a guide to the quality one can expect in real-life applications. For a clustering structure of reasonable complexity, say five possibly overlapping biclusters, our results suggest that around 50% of the structure will be recovered and around 50% of the gene–sample pairs will be the result of spurious structure. This may be regarded as a conservative estimate of quality as we have based our evaluation on small bicluster sizes.

We have emphasised the two-dimensional quality of biclusters since the biclusters are naturally interpreted in terms of gene–sample pairs. For example, by our definition, a bicluster would need to obtain marginal sensitivity values of approximately 90% to achieve 80% sensitivity overall. Thus, our measures are more stringent than one-way quality measures, but they give an indication of the error involved if the biclusters are directly interpreted.

Our method for obtaining a quality measure for a cluster set penalises clustering results that partially represent the same target bicluster in two or more retrieved biclusters, since only one retrieved bicluster is matched to each target bicluster for comparison. One could avoid this by matching each retrieved bicluster to the best matching target bicluster, but this would ignore target biclusters which are not retrieved at all. Either method penalises retrieved biclusters that contain more than one target bicluster, since comparisons are made on a pairwise basis.

As regards specificity, perhaps a measure that looks for the elements of a target bicluster over all the retrieved biclusters needs to be considered. Care would need to be taken with overlapping biclusters, since an element should only be counted as retrieved if it is clear that it has been retrieved as a result of the effect represented by the target bicluster under consideration, and not another target bicluster. In respect of sensitivity, perhaps a measure over the whole cluster set would be more appropriate, measuring the proportion of target elements in all retrieved elements. A further issue is the interpretability of the cluster solutions; for example, cluster sets that do not form a natural one-to-one correspondence with the target biclusters may obscure the underlying structure. It might be necessary to consider this feature if a more flexible approach to obtaining quality measures for the whole cluster set were adopted.

The sensitivity of the plaid model algorithms could be improved by using a stopping rule as described in Algorithm 1, but this would not improve specificity. Clearly, relaxing the pruning criteria may increase specificity but may also decrease sensitivity,
so this device can not be relied upon to improve performance overall. Specificity may be improved by further rounds of back fitting, though this increases computation time and can add noise by adjusting fitted layer effects on the basis of incorrectly clustered data.

Performance might be improved by taking a more flexible approach to back fitting. This could be as simple as delaying bicluster pruning until after back fitting, so that the fit of the model is judged on the basis of better estimates of the layer effects. Alternatively, a few more iterations could be performed after back fitting to refine the cluster membership. These modifications would need to be considered in conjunction with the timing of the significance test—in the original algorithm the significance of a bicluster is determined before back fitting.

Specificity might also be increased by making use of auxiliary information about the genes and samples. If genes or samples are known to belong to certain groups, one might expect all the members of a group to be clustered together. The plaid algorithm could be modified to favour such biclusters by conducting early iterations on the groups directly, using the prior knowledge to direct the search for genes and samples individually.

Our method for generating artificial microarray data could be adapted to test the performance of any high-level analysis. For example, one could simulate data with an up-regulation on a subset of genes between two treatments to test methods for detecting differential expression. Some research has been done in this area using simulated microarray hybridization experiments incorporating various sources of experimental error (Wierling et al., 2002; Singhal et al., 2004; UPMC Health Systems, 2004). These methods also allow image analysis programs and normalisation procedures to be evaluated. Whilst there is great value in examining the whole data analysis process, our relatively simple method of simulating “cleaned” gene expression data allows more complex high level analyses to be studied in depth.

Other methods for generating error distributions could be investigated, for example Strimmer (2003) uses a quasi-likelihood approach for modelling error on gene expression measurements which makes minimal distributional assumptions and could be modelled on real data. However, our approach requires no distributional assumptions, so it can be used on gene expression data from any platform and with any data transformation, as long as any systematic experimental effects are removed by appropriate normalisation. Furthermore, our method of decomposing data from a “null” experiment to obtain an empirical error distribution simultaneously provides a distribution of expression levels, giving all the material necessary to simulate realistic data.

Acknowledgements

We thank Dr. Cheryl Hemingway for providing microarray data generated as part of a collaboration funded by the Wellcome Trust between Imperial College and the Relman Laboratory, Stanford University.
References


